**A Course Project of Applicable Mathematics (U18OE401A)**

**ENTITLED**

# "Uniformly convergent finite difference operator method for function of single variable"


SUBMITED TO **: Department of Information Technology**
**KAKATIYA INSTITUTE OF TECHNOLOGY AND SCIENCE, WARANGAL**

Course Name : Applicable Mathematics

Course Code : U18OE401A

Section : 4IT2

**Course project presented by: K. Aryan**

Course Faculty Name : Dr. Narahari Raji Reddy

Course faculty email ID : nrr.mh@kitsw.ac.in

SIGN OF STUDENT

COURSE PROJECT

SIGN. OF GUIDE

(Dr. Narahari Raji Reddy)

COURSE PROJECT SUPERVISOR

KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE
(An Autonomous Institute under Kakatiya University, Warangal)
Opp: Yerragattu Gutta, Hasanparthy (Mandal), WARANGAL - 506 015, Telangana, INDIA.
काकतीय प्रद्योगिकी एवं विज्ञान संस्थान, वरंगल - ५०६ ०१५
కాకతీయ సాంకేతిక విజ్ఞాన శాస్త్ర విద్యాలయం, వరంగల్ - ౫౦౬ ౦౧౫
Estd 1980
KITSW
website: www.kitsw.ac.in | e-mail: principal@kitsw.ac.in | ✆: +91 870 2564888 | Fax: +91 870 2564320

# D E C L A R A T I O N

I, **K. Aryan**, hereby, declare that the work presented in this Course Project entitled, "***Uniformly convergent finite difference operator method for function of single variable***", has been carried out by us in the Department of Mathematics and Humanities, Kakatiya Institute of Technology and Science, Warangal, under the supervision of ***Dr. Narahari Raji Reddy,*** Assistant Professor.

Place: Warangal
Date: 08– 04 – 2025

(**K. Aryan)**
Course Project student

KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE
(An Autonomous Institute under Kakatiya University, Warangal)
Opp: Yerragattu Gutta, Hasanparthy (Mandal), WARANGAL - 506 015, Telangana, INDIA.
काकतीय प्रद्योगिकी एवं विज्ञान संस्थान, वरंगल - ५०६ ०१५
కాకతీయ సాంకేతిక విజ్ఞాన శాస్త్ర విద్యాలయం, వరంగల్ - ౫౦౬ ౦౧౫
Estd 1980 KITSW
website: www.kitsw.ac.in | e-mail: principal@kitsw.ac.in | ✆: +91 870 2564888 | Fax: +91 870 2564320

Dr. Narahari Raji Reddy
Mathematics & Humanities Dept.
Assistant Professor **KITS, Warangal**-506 015

Email: nrr.mh@kitsw.ac.in

Date: 08– 04 – 2025

# C E R T I F I C A T E

It is certified that the work contained in the Course Project entitled "***Uniformly convergent finite difference operator method for function of single variable***", by **K. Aryan of IV- Semester B.Tech. Information Technology**, has been carried out under my supervision in the partial fulfilment of the requirement of B.Tech.

degree during the academic year 2024-2025.

(Dr. Narahari Raji Reddy)
Course Project Supervisor

(Dr. K Shiva Shankar)
Associate Professor & Head
Dept. Mathematics & Humanities
KITS, Warangal

(Dr. T. Senthil Murugan)
Professor & Head
Dept. of Information Technology
KITS, Warangal

# Index

# A C K N O W L E D G E M E M T S

I am deeply indebted to Dr. Narahari Raji Reddy, Assistant Professor, Department of Mathematics and Humanities, KITS, Warangal for suggesting the topic and for his guidance at each stage of this Course project. I express my deep sense of gratitude for his invaluable and sustained guidance throughout the process of course project work. It is an only account of his guidance and timely help to present the course project in this shape.

I express my sincere thanks to Dr. K Shiva Shankar, Associate Professor and Head, Department of Mathematics and Humanities, KITS, Warangal, for his constant encouragement during my study period here.

Secondly, I wish to express a sincere acknowledgement to Dr. T. Senthil Murugan, Professor, Dept. of Information Technology, for all the guidance, kind comments.

It is our privilege thank to Prof. K. Ashoka Reddy, Principal, KITS, Warangal, for his valuable suggestions and constant encouragement during our study period here.

Last but not least, I am thankful to all my classmates for their constant encouragement during my study period here.

**K. Aryan**

# A B S T R A C T

The course project introduces the numerical analysis of different types of equations, describing the mathematical background for understanding numerical methods and giving information on what to expect when using them. As a reason for studying numerical methods as a part of a more general course on different types of equations, many of the basic ideas of the numerical analysis of different types of equations are tied closely to theoretical behaviour associated with the problem being solved. For example, the criteria for the stability of a numerical methods are closely connected to the stability of the different types of problems being solved.

We also include MATLAB programs to illustrate many of the ideas that are introduced in the course project. Much is to be learned by experimenting with the numerical solution of different types of equations.

Numerical methods vary in their behavior, and the many different types of problems affect the performance of numerical methods in a variety of ways.


# Chapter 1

# Introduction

The accurate solution of initial or boundary value problems is central to many scientific and engineering applications. Whether modelling the deflection of a beam under load,

the steady-state temperature distribution in a rod, or electrostatic potentials in a region, initial or boundary value problems provide a powerful framework for describing physical systems. However, most real-world initial or boundary value problems cannot be solved analytically due to the complexity of their governing differential equations or boundary conditions. This limitation underscores the importance of numerical methods, particularly the finite difference method (FDM), which provides a practical, efficient, and accessible approach to approximating solutions to such problems.

Numerical analysis plays a crucial role in approximating complex mathematical functions using simpler forms, such as polynomials. In Chapter 2, we explore interpolation, a fundamental technique that constructs a polynomial matching a given function at discrete points. This interpolating polynomial allows us to estimate values at non-tabular points, perform differentiation and integration, and solve equations efficiently. We focus on Newton's Interpolation Method, which is particularly useful for equi-spaced data. Additionally, we discuss key applications, including function prediction, numerical operations and highlighting the importance of interpolation in computational mathematics. Through this chapter, we establish a clear understanding of polynomial approximation and its practical significance in scientific computing.

Numerical differentiation plays a vital role in approximating derivatives when a function is given in tabular form rather than as an analytical expression. In Chapter 3, we explore methods for estimating derivatives using finite differences, particularly when exact differentiation is impractical. Given a function defined at discrete points we derive approximations for differentiation at both tabular and non-tabular points. Using the example, we apply the forward difference formula to the data and compare it with the exact analytical result, demonstrating the accuracy and limitations of numerical differentiation. This chapter provides essential techniques for derivative approximation, highlighting their significance in scientific computing and engineering applications where exact derivatives are unavailable.

Numerical integration provides essential tools for approximating definite integrals when analytical solutions are difficult or impossible to obtain. In Chapter 4, we examine fundamental techniques for computing $\int_{x=a}^{b} f(x)\, dx$ using discrete data points, focusing on the Trapezoidal Rule and

Simpson's Rule. These methods become particularly valuable when dealing with complex functions or experimental data available only at tabulated points.

Together, these chapters form a comprehensive journey from mathematical theory to computational application. They demonstrate the effectiveness of the finite difference method as a reliable tool for solving function of single variables and establish a solid framework for extending this work to more complex or multidimensional problems in future studies.

# Chapter 2

## INTERPOLATION AND APPROXIMATION

---

In this chapter, we shall discuss the problem of approximate a given function of single variable by a polynomial using interpolation.

Interpolation is a procedure for estimating a value between known values of data points. It is done by first determining a polynomial that gives the exact value at the data points, and then using the polynomial for calculating values between the points. When a small number of points is involved, a single polynomial might be sufficient for

interpolation over the whole domain of the data points. For example, Fig. 2-1 shows a plot of the stress-strain relationship for rubber. The red markers show experimental points that were measured very accurately, and the solid curve was obtained by using interpolation. It can be observed that the curve passes through the points precisely and gives a good estimate of values between the points.

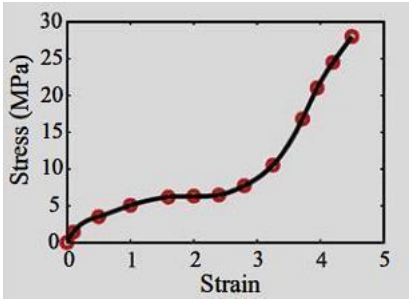



Fig. 2.1

Let $y = f(x)$ be a continuous function defined on some interval $[a, b]$ , and be prescribed at $n + 1$ distinct tabular points $x_0, x_1, x_2, \dots, x_n$ such that $a = x_0 < x_1 < x_2 < \cdots < x_n = b$. The distinct tabular points $x_0, x_1, x_2, \dots, x_n$ may be non-equispaced or equispaced, that is $x_i - x_{i-1} = h, i = 1,2, \dots n$.

The problem of polynomial approximation is to find a polynomial function $P_n(x)$, of degree $\leq n$, which fits the given data exactly, that is,

$$P_n(x_i) = f(x_i), i = 0,1,2, \dots n.$$

$\therefore$ The polynomial function $P_n(x)$ is called the interpolating polynomial.

The deviation of $P_n(x)$ from $f(x)$ , that is $f(x) - P_n(x)$ , is called the error of approximation.

**Newton's Forward Interpolation formula:**

If the required $'x'$ value is around the starting value of the table and data is equi spaced, then the most suited for interpolation is *Newton's forward difference Interpolation*.

Let the required value $= x, h = x_1 - x_0, p = \frac{x - x^0}{h}$.

The Newton's forward interpolation formula is

$$P_n(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots \dots \dots$$

Problem 1: Consider the one variable function of the form $y = x^2$ for $x \in [1,3]$ . Verify whether the interpolating polynomial function approximate with the given function using Newton's Interpolation formula.

**Solution:** Given that $y = x^2$ for $x \in [1,3]$

| $x$ | $x_0 = 1$ | $x_1 = 2$ | $x_3 = 3$ |
|---|---|---|---|
| $y = x^2$ | $y_1 = 1$ | $y_1 = 4$ | $y_3 = 9$ |

From the given data, we have $h = 1 - 0 = 1$,

The data is called equi-spaced data

The forward difference table is

| $x$ | $y$ | $\Delta y$ | $\Delta^2 y$ |
|---|---|---|---|
| $x_0 = \mathbf{1}$ | $\boldsymbol{y_0 = 1}$ | | |
| | | $\boldsymbol{\Delta y_0 = 3}$ | |
| $x_1 = 2$ | $y_1 = 4$ | | $\boldsymbol{\Delta^2 y_0 = 2}$ |
| | | $\Delta y_1 = 5$ | |
| $x_2 = 3$ | $y_2 = 9$ | | |

Let $x = x$, $h = 1$, $p = \frac{x - x^0}{1} = \frac{x - 1}{h}$

The Newton's forward interpolation formula is

$$P_n(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \cdots$$

$$\therefore P_2(x) = 1 + (x-1) * 3 + \frac{(x-1)(x-2)}{2!} * 2$$

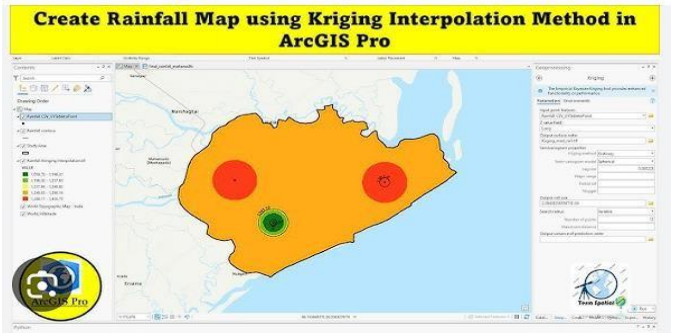
$$= 1 + 3x - 3 + x^2 - 3x + 2 = x^2$$

Hence the interpolating polynomial function coincides exactly with the given polynomial function.

Observations:

i) The approximating polynomial $P_n(x)$ can be used to predict the value o $f(x)$f at a nontabular points.

ii) The second use is to perform the required operations, like determination of roots, differentiation and integration etc. can be carried out using the approximating polynomial function $P_n(x)$




# Chapter 3

## NUMERICAL DIFFERENTIATION

---

In this chapter, we shall discuss the problem of approximating the derivative of a given function of single variable by numerical differentiation.

Differentiation gives a measure of the rate at which a quantity changes. Rates of change of quantities appear in many disciplines, especially science and engineering. One of the more fundamental of these rates is the relationship between position, velocity, and acceleration. If the position, $y$ of an object that is moving along a straight line is known as a function of independent variable, $x$, $y = f(x)$ the object's velocity, $v(x)$, is the derivative of the position with respect to variable, $x$, i.e.,

$$v = \frac{dy}{dx}$$

The velocity $v$ is also the slope of the position- curve.

Many models in physics and engineering are expressed in terms of rates. In an electrical circuit, the current in a capacitor is related to the time derivative of the voltage. In analyzing conduction of heat, the amount of heat flow is determined from the derivative of the temperature. Differentiation is also used for finding the maximum and minimum.

We assume that a function $f(x)$ is given in a tabular form at a set of $n+1$ distinct points $x_0, x_1, x_2, \dots, x_n$. From the given tabular data, we require approximation to the derivative $\left(\frac{dy}{dx}\right)_{x=x_0}$, where $x_0$ may be a tabular or a non-tabular point.

**Derivatives Using Newton's Forward Interpolation Formula:**

Consider the data $(x_i, f(x_i))$ given at equi-spaced points $x_i = x_0 + ih, i = 1,2, \dots, n$ where $h$ is the step length. The Newton's forward derivative formula for first order derivative is given

$$\left(\frac{dy}{dx}\right)_{x=x_0} = \frac{1}{h}\left[\Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0 \dots \dots\right]$$

Problem 2: Consider the function of the form $y = x^2$ for $x \in [1,3]$ . Calculate $\left(\frac{dy}{dx}\right)_{x=1}$ using numerical differentiation and compare it with its actual value. Write a user-defined MATLAB program that determines the errors of the function.

**Solution:** Given that $y = x^2$ for $x \in [1,3]$

Consider the equi-spaced points $x_i = x_0 + ih, i = 1,2$ where $h = 1$ is the step length.

| $x$ | $x_0 = 1$ | $x_1 = 2$ | $x_3 = 3$ |
|---|---|---|---|
| $y = x^2$ | $y_1 = 1$ | $y_1 = 4$ | $y_3 = 9$ |

(i) Numerical differentiation:

The forward difference table is

| $x$ | $y$ | $\Delta y$ | $\Delta^2 y$ |
|---|---|---|---|

| | | | |
|---|---|---|---|
| $x_0 = \mathbf{1}$ | $\boldsymbol{y_0 = 1}$ | | |
| | | $\boldsymbol{\Delta y_0 = 3}$ | |
| $x_1 = 2$ | $y_1 = 4$ | | $\boldsymbol{\Delta^2 y_0 = 2}$ |
| | | $\Delta y_1 = 5$ | |
| $x_2 = 3$ | $y_2 = 9$ | | |

In the given data, $h = x_1 - x_0 = 1$

The required derivative at $'x = x_0 = 1'$ is the starting value of the table and data is equispaced,

*Forward derivative* formula for first order derivative is

$$\left(\frac{dy}{dx}\right)_{x=x_0} = \frac{1}{h}\left[\Delta_0 - \frac{1}{2}\Delta^2 y_0 + \cdots\right]$$

$$\left(\frac{dy}{dx}\right)_{x=1} = \frac{1}{1}\left[3 - \frac{1}{2}(2)\right] = 2$$

Actual differentiation: Here $y = x^2$

$$\therefore \left(\frac{dy}{dx}\right)_{x=1} = (2x)_{x=1} = 2$$

Hence the Numerical first order differentiation value coincides exactly with the given first order derivative of polynomial function.

**MATLAB Code:**

```
% Given data points
x = [1, 2, 3];
y = [1, 4, 9]; % y = x^2

% Compute forward differences (Δy)
delta_y = diff(y);

% Compute the first-order derivative using the corrected formula
% y'(i) = (1/2) * [Δy_0 + i]
% Here, Δy_0 is the first forward difference, and i = 1
delta_y0 = delta_y(1); % Δy_0 = 3 i = 1; % As per the
formula
y_prime = (1/2) * (delta_y0 + i);

% Display the result
disp('Computed first-order derivative:'); disp(y_prime);
```

```
% Verification: Analytical derivative of y = x^2
syms x_analytical; y_analytical =
x_analytical^2;
dy_dx_analytical = diff(y_analytical, x_analytical);

% Evaluate the analytical derivative at x = 1
dy_dx_value = double(subs(dy_dx_analytical, x_analytical, 1));

disp('Analytical first-order derivative at x = 1:');
disp(dy_dx_value);

% Check if the computed derivative matches the analytical derivative tolerance
= 1e-5;
if abs(y_prime - dy_dx_value) < tolerance
   disp('The computed derivative matches the analytical derivative.'); else
   disp('The computed derivative does not match the analytical derivative.'); end
```

**Output:**

```
>> discrete_newton

Computed first-order derivative:       2

Analytical first-order derivative at x = 1:      2

The computed derivative matches the analytical derivative.
```

# Chapter 4

## NUMERICAL INTEGRATION

In this chapter, we shall derive numerical methods to compute an integral numerically.

In many applications of science and engineering, we require to compute the value of the definite integral $\int_{x=a}^{b} f(x)\,dx$, where $f(x)$ may be given explicitly or as a tabulated data. Even when $f(x)$ is given explicitly, it may be a complicated function such that integration is not easily carried out. In this chapter, we shall derive numerical integration methods to compute the integral numerically.

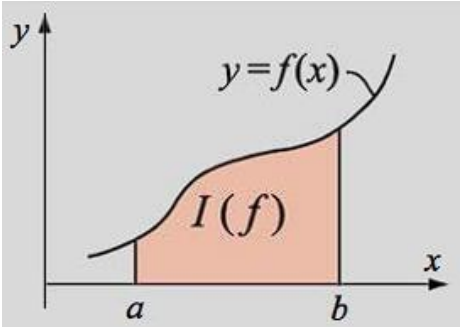



Fig 4.1

Graphically, as shown in Fig. 4-1, the value of the integral corresponds to the shaded area under the curve of $f(x)$ between $x = a$ and $x = b$.

In applications, evaluation of the integral may prove to be very complicated and not practical:

a) when we cannot find the antiderivative $F(x)$ of $f(x)$ and

b) when the integrand $f(x)$ is a tabulated function.

**Newton–Cotes Quadrature Formula**

Let $I = \int_a^b f(x)dx$. Divide $[a, b]$ into $n$ subintervals of equal length $\boldsymbol{h = \frac{b-a}{n}}$ by means of points $a = x_0 < x_1 < x_2 < \cdots \ldots \ldots < x_{n-1} < x_n = b$.

Hence by Newton's forward difference formula, we obtain

$$I = \int_{x_0}^{x_n} y\,dx = nh\left[y_0 + \frac{n}{2}\Delta y_0 + \frac{n(2n-3)}{12}\Delta^2 y_0 + \frac{n(n-2)^2}{24}\Delta^3 y_0 + \cdots\right] \quad \text{_____(1)}$$

which is called Newton-Cotes quadrature formula.

**Trapezoidal Rule:** Taking $n = 1$ in equation (1) and neglecting all differences higher than the first, we obtain

$$I = \frac{h}{2}[(y_0 + y_n) + 2(y_1 + y_2 + y_3 + y_4 + \cdots)] \quad \text{(2)}$$

- The Trapezoidal Rule can be applied to any number of subintervals, that is, when $n$ is even or odd.

**Simpson's $\frac{1}{3}$ Rule:** Taking $n = 2$ in equation (1) and neglecting all differences higher than the second, we obtain

$$I = \frac{h}{3}[(y_0 + y_n) + 4(y_1 + y_3 + y_5 + \cdots) + 2(y_2 + y_4 + \cdots)] \quad \text{(3)}$$

- Simpson's $\frac{1}{3}$ Rule can be applied when the number of subintervals is even.

**Simpson's $\frac{3}{8}$ Rule:** Taking $n = 3$ in equation (1) and neglecting all differences higher than the third, we obtain

$$I = \frac{3h}{8}[(y_0 + y_n) + 3(y_1 + y_2 + y_4 + y_5 + \cdots) + 2(y_3 + y_6 + y_9 + \cdots)] \quad \text{(4)}$$

- Simpson's $\frac{3}{8}$ Rule can be applied when the number of subintervals is a multiple of 3.

Problem 3: Consider the single variable function of the form $y = x^2$ for $x \in [1,3]$ .

Calculate $\int_{x=1}^{3} x^2\,dx$ using numerical integration of Trapezoidal rule and Simpson's rule for $h = 1$ & $h = 0.5$ and compare it with its actual value. Write a user-defined MATLAB program that determines the errors of the function.

**Solution:** Given that $y = x^2$ for $x \in [1,3]$

**(i)** Consider the equi-spaced points $x_i = x_0 + ih, i = 1,2$ where $h = 1$ is the step length.

| $x$ | $x_0 = 1$ | $x_1 = 2$ | $x_3 = 3$ |
|---|---|---|---|
| $y = x^2$ | $y_1 = 1$ | $y_1 = 4$ | $y_3 = 9$ |

By Trapezoidal rule for the given data, we have

$$\int_a^b f(x)dx = \frac{h}{2}[(y_0 + y_n) + 2(y_1 + y_2 + y_3 + y_4 + \cdots \ldots)]$$

$$\int_{x=1}^{3} x^2\,dx = \frac{1}{2}[(y_0 + y_2) + 2y_1]$$

$$= 0.5[(1 + 9) + 2 * 4] = 9.0$$

By Simpson's $\frac{3}{8}$ Rule for the given data, we have

$$\int_a^b f(x)dx = \frac{3h}{8}[(y_0 + y_n) + 3(y_1 + y_2 + y_4 + \cdots) + 2(y_3 + y_6 + \cdots)]$$

$$\int_{x=1}^{3} x^2\,dx = \frac{3}{8}(1)[(1 + 9) + 3(4)] = \frac{3}{8} * 22 = 8.25$$

**(ii)** Consider the equi-spaced points $x_i = x_0 + ih$, $i = 1,23,4$ where $h = 0.5$

| $x$ | $x_0 = 1$ | $x_1 = 1.5$ | $x_2 = 2$ | $x_3 = 2.5$ | $x_4 = 3$ |
|---|---|---|---|---|---|
| $y = x^2$ | $y_1 = 1$ | $y_1 = 2.25$ | $y_2 = 4$ | $y_3 = 6.25$ | $y_4 = 9$ |

By Trapezoidal rule for the given data, we have

$$\int_a^b f(x)dx = \frac{0.5}{2}[(y_0 + y_4) + 2(y_1 + y_2 + y_3)]$$

$$\int_{x=1}^{3} x^2\,dx = 0.25[(1+9) + 2*12.5] = 8.75$$

By Simpson's $\frac{1}{3}$ Rule for the given data, we have

$$\int_a^b f(x)dx = \frac{h}{3}[(y_0 + y_4) + 4(y_1 + y_3) + 2(y_2)]$$

$$\int_{x=1}^{3} x^2\,dx = \frac{0.5}{3}[(1+9) + 4*8.5 + 2*4] = \frac{0.5}{3}*52 = 8.6667$$

The integral is computed using the step lengths $h, \frac{h}{2}$. We compute the value of the integral with a number of step lengths using the same method. Usually, we start with a coarse step length, then reduce the step lengths and re compute the value of the integral. The sequence of these values converges to the exact value of the integral.

Actual Integration: $\int_{x=1}^{3} x^2\,dx = \left[\frac{x^3}{3}\right]_{x=1}^{3} = \frac{26}{3} = 8.6667$

| | $h = 1$ | $h = 0.5$ |
|---|---|---|
| (a) Trapezoidal rule | $I = 9.0$ | $I = 8.75$ |
| (b) Simpson's $\frac{1}{3}$ Rule or Simpson's $\frac{3}{8}$ Rule | $I = 8.25$ | $I = 8.6667$ |
| (c) Actual value | $I = 8.6667$ | $I = 8.6667$ |

The errors in the solutions are the following:

$$error, E = |I_{exact} - I_{appr.}|$$

| | $h = 1$ | $h = 0.5$ |
|---|---|---|
| (a) Trapezoidal rule | $E = 0.33$ | $E = 0.083$ |
| (b) Simpson's $\frac{1}{3}$ Rule or Simpson's $\frac{3}{8}$ Rule | $E = 0.416$ | $E = 0$ |

**MATLAB Code for h=1:**

```
% Given data points
x = [1, 2, 3]; y = [1, 4,
9]; % y = x^2

% Newton's forward interpolation formula
% f(n) = Y_0 + P * ΔY_0 + (P*(P-1)/2) * Δ^2Y_0
% P = (n - x0) / h, where h is the step size h =
1; % Step size x0 = x(1); % Initial x value
```

```
P = @(n) (n - x0) / h;

% Compute the forward differences
delta_y = diff(y);
delta2_y = diff(delta_y);

% Define the interpolation function
f = @(n) y(1) + P(n) * delta_y(1) + (P(n) * (P(n) - 1)) / 2 * delta2_y(1);

% Display the interpolation function
disp('Interpolation function f(n):'); disp(f);

% Evaluate the interpolation function at a specific point (e.g., n = 2.5) n_value
= 2.5; f_value = f(n_value);
disp(['Interpolated value at n = ', num2str(n_value), ':']); disp(f_value);

% Numerical integration using the trapezoidal rule
integral_trapezoidal = (h / 2) * (y(1) + y(3) + 2 * y(2));
disp('Numerical integration using trapezoidal rule:');
disp(integral_trapezoidal);

% Numerical integration using Simpson's 1/3 rule
integral_simpson = (h / 3) * (y(1) + y(3) + 4 * y(2));
disp('Numerical integration using Simpson''s 1/3 rule:');
disp(integral_simpson);

% Verification: Analytical integration of y = x^2 from 1 to 3
syms x_analytical; y_analytical = x_analytical^2;
integral_analytical = int(y_analytical, x_analytical, 1, 3);
disp('Analytical integration of y = x^2 from 1 to 3:');
disp(double(integral_analytical));
```

**Output:**

```
>> continuous_newton for h=1

Numerical integration using trapezoidal rule:      9

Numerical integration using Simpson's 1/3 rule:      8.666

Analytical integration of y = x^2 from 1 to 3:       8.666
```

**MATLAB Code for h=0.5:**

```
% Given data
X = [1, 1.5, 2, 2.5, 3];  % X values Y
= [1, 2.25, 4, 6.25, 9]; % Y values
h = 0.5;                  % Step size
```

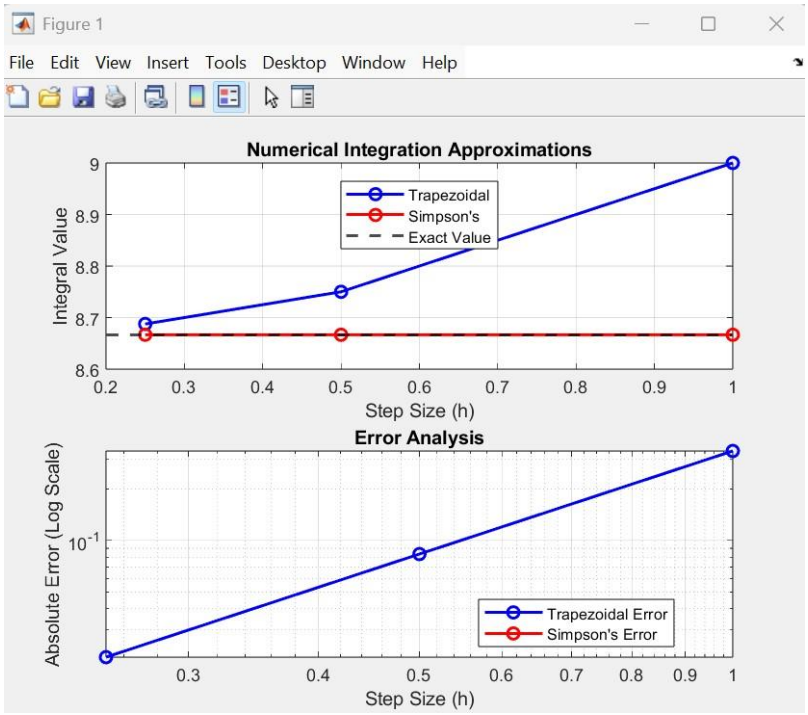
```
% Trapezoidal Rule
n = length(Y);          % Number of data points
trapezoidal_integral = (h/2) * (Y(1) + 2*sum(Y(2:end-1)) + Y(end));

% Simpson's 3/8 Rule
if mod(n-1, 2) == 0       % Check if the number of intervals is even     simpson_integral =
(h/3) * (Y(1) + 4*sum(Y(2:2:end-1)) + 2*sum(Y(3:2:end-2)) + Y(end)); else
   error('Simpson's 3/8 rule requires an even number of intervals.'); end

% Display results fprintf('Trapezoidal Rule Integral: %.4f\n',
trapezoidal_integral); fprintf('Simpson's 3/8 Rule Integral:
%.4f\n', simpson_integral);
```

Output:

```
>> discretetrapezoidal

Trapezoidal Rule Integral: 8.75

Simpson's 3/8 Rule Integral: 8.666
```

```
% Plot results
figure;

% Approximation vs Exact Value
subplot(2,1,1);
plot(h_values, trap_results, 'bo-', 'LineWidth', 1.5, 'DisplayName', 'Trapezoidal');
hold on; plot(h_values, simp_results, 'ro-', 'LineWidth', 1.5, 'DisplayName',
'Simpson''s'); yline(exact_integral, 'k--', 'LineWidth', 1.5, 'DisplayName', 'Exact
Value'); xlabel('Step Size (h)'); ylabel('Integral Value');
title('Numerical Integration Approximations');
legend('Location', 'best'); grid
on;

% Error Analysis (Log Scale) subplot(2,1,2);
loglog(h_values, trap_errors, 'bo-', 'LineWidth', 1.5, 'DisplayName', 'Trapezoidal Error');
hold on;
loglog(h_values, simp_errors, 'ro-', 'LineWidth', 1.5, 'DisplayName', 'Simpson''s Error');
xlabel('Step Size (h)');
ylabel('Absolute Error (Log Scale)');
```

```
title('Error Analysis');
legend('Location', 'best'); grid
on;
```




Graph for Simpson's Error represents zero error in error analysis.

# Chapter 5: Conclusion and Applications

The development and application of the finite difference method for solving single variable function, as outlined in Chapters 2, 3, and 4, represent a significant step in the numerical approximation of function. Through this progression, the project has showcased the method's ability to transform abstract mathematical models into concrete computational solutions capable of addressing a wide range of real-world problems.

Chapter 2 offered critical insight into the formulation and theoretical underpinning of the finite difference approach. By discretizing the problem domain and replacing single variable function with the corresponding nodes, it became possible to reduce complex function into solvable numerical approximation. This transformation lies at the heart of numerical analysis and enables powerful approximations where analytical methods fall short. The emphasis on both boundary conditions and consistency in the finite difference scheme ensured the model's applicability to diverse scenarios.

Numerical differentiation is carried out on data that are specified as a set of discrete points in chapter-3. In many cases the data are measured or are recorded in experiments, or they may be the result of large-scale numerical calculations. If there is a need to calculate the numerical derivative of a function that is given in an analytical form, then the differentiation is done by using discrete points of the function. This means that in all cases numerical integration is done by using the values of points

In Chapter 4, all cases the numerical integration is carried out by using a set of discrete points for the integrand. When the integrand is an analytical function, the location of the points within the interval $[a, b]$ can be defined by the user or is defined by the integration method. When the integrand is a given set of tabulated points (like data measured in an experiment), the location of the points is fixed and cannot be changed. the finite difference algorithm was put to the test through a series of benchmark problems. The comparisons between numerical and analytical solutions validated the correctness and accuracy of the implementation. Error analysis revealed the convergence behavior of the method, affirming the expected theoretical properties of second-order accuracy. The sensitivity of the solution to grid spacing was also explored, reinforcing the trade-offs between computational cost and solution precision.

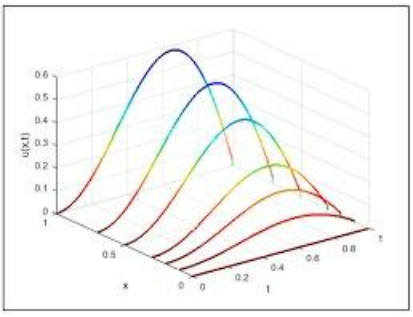
Collectively, these chapters illustrate how a numerical technique—grounded in sound mathematical principles and implemented through thoughtful programming—can produce reliable and insightful results. The finite difference method, while conceptually simple, proves to be a robust and scalable tool for solving a broad class of one variable function. This work not only affirms the power of numerical methods in applied mathematics but also paves the way for further exploration into more advanced topics such as adaptive meshing, multi-dimensional boundary value problems, and the use of modern software libraries for large-scale simulations.

In conclusion, this segment of the project has achieved its objective: to bridge theory and computation through the finite difference method, providing a practical and effective means of solving function of one variable. The experience gained from this study underscores the value of numerical analysis as both an academic discipline and a practical tool in the modern scientific and engineering landscape.

## Real Life Application

### 1. Inverse Problems in Boundary Value Problems

- Discrete differentiation models solve for derivatives from tabulated data, analogous to inverse problems where differential equation coefficients (e.g., material properties) are inferred from observed outputs. Newton's interpolation helps reconstruct missing data points in boundary conditions.

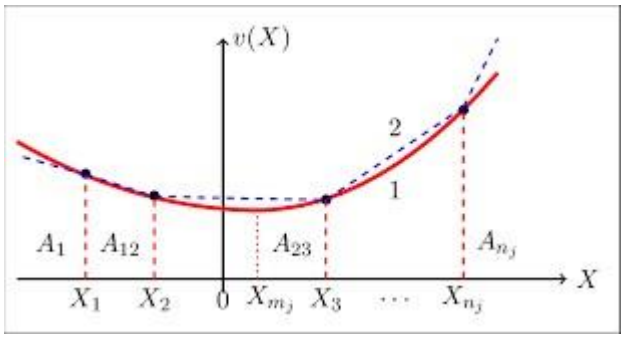



- Real-World Use:
  - Geophysics: Determining underground rock permeability from seismic wave data.
  - Engineering: Identifying heat conduction coefficients in thermal systems.

## 2. Determination of Interfaces in Computerized Tomography


- Trapezoidal/Simpson's integration approximates integrals of pixel intensities along projection paths, similar to Radon transform calculations in CT scans.
- Discrete differentiation detects edges (sharp changes) in reconstructed images.
- Application: Medical Imaging: Locating tumor boundaries by integrating X-ray absorption data from multiple angles.

## 3. Local Volatility Reconstruction (Dupire Formula)

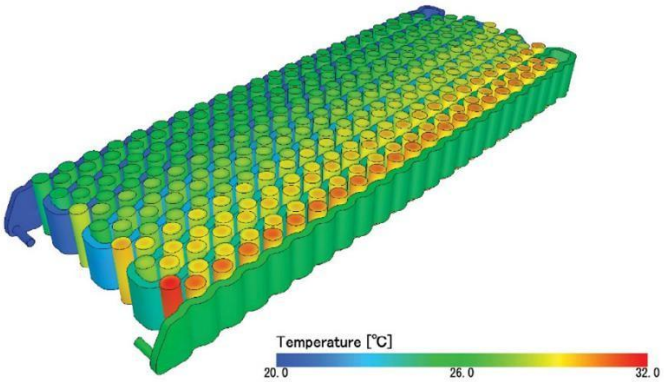



- Second-order derivatives mirror the Dupire formula, which computes volatility (σ) from option prices using:

$$\sigma(S,t)^2 = 2\frac{\frac{\partial C}{\partial T} + rS\frac{\partial C}{\partial S}}{S^2\frac{\partial^2 C}{\partial S^2}}$$

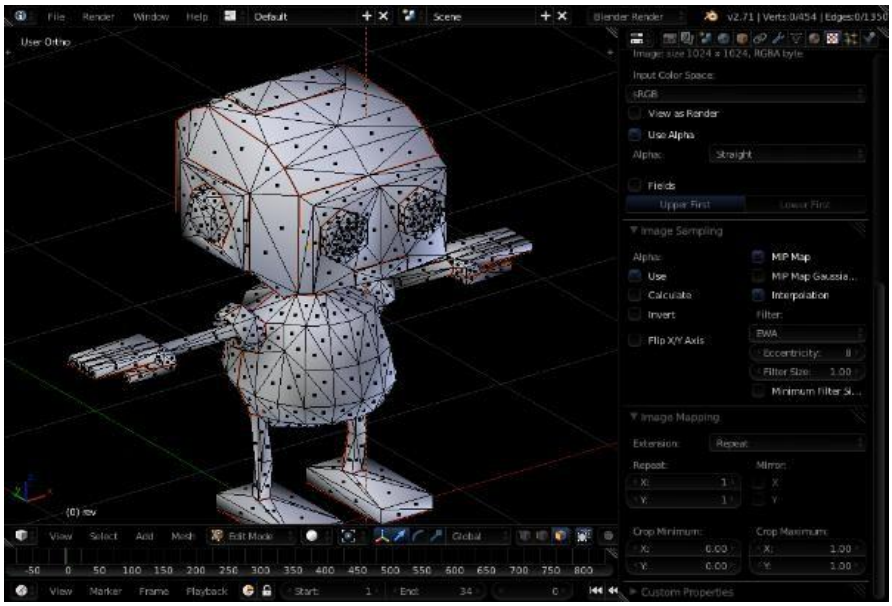
- Discrete differentiation approximates partial derivatives from market data.
- Use Case: Finance: Pricing exotic options by calibrating volatility surfaces to market data.

**4. Structural Analysis, Fluid Dynamics, Heat Transfer**




- Forward differences approximate stress/strain derivatives in finite element analysis.
- Numerical integration computes forces, fluxes, or energy
- Application:-Aerospace: Predicting wing deformation using numerical integration of load distributions.

**5. Computer Graphics & Animation**



- Numerical Interpolation generates smooth curves (Bézier splines) from keyframe points.
- Numerical Differentiation calculates normals/curvature for lighting/shading.
- Application:- Pixar Animation: Rendering motion paths via polynomial interpolation

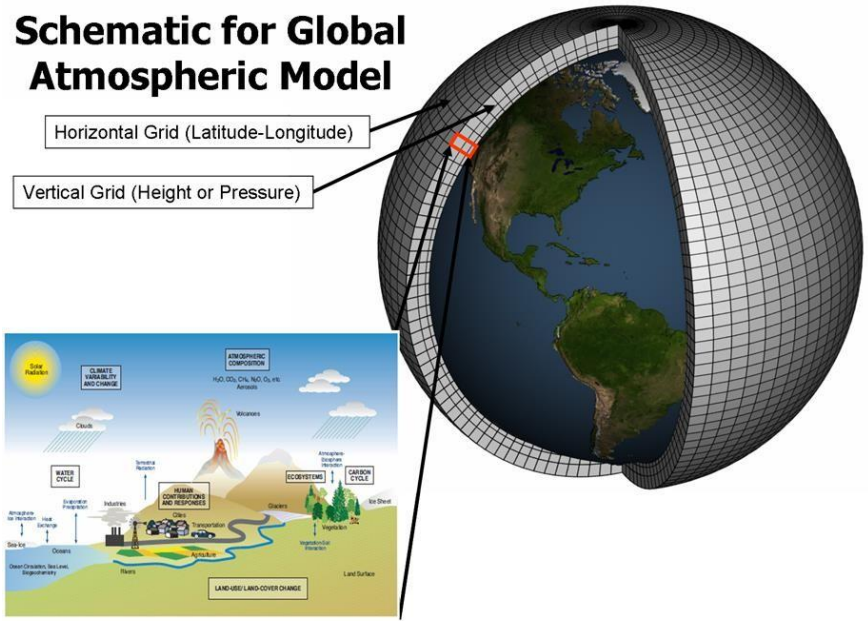
## 6. Option Pricing in Stock Markets


- Connection:
  - Black-Scholes PDE requires numerical integration for Monte Carlo simulations.
  - Discrete dividends are modelled using your Newton interpolation.
- Example:
  - Risk Management: Valuing path-dependent options with trapezoidal rule integration.

## 7. Medical Imaging (MRI/Ultrasound)

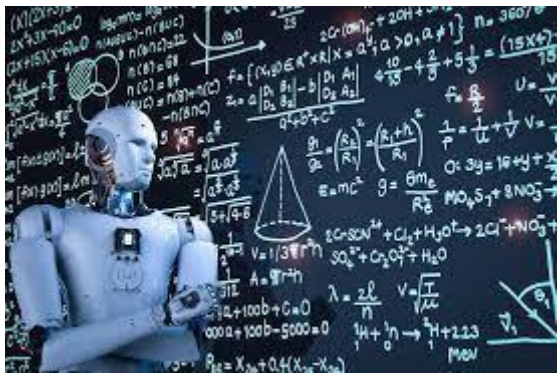
- Connection:
  - Fourier transforms rely on numerical integration to convert signal data to images.
  - Edge detection uses derivatives to highlight anatomical features.
- Use:
  - Tumor Detection: Integrating MRI signals to reconstruct 3D tumor volumes.

## 8. Climate Modelling




- Connection:
  - Solving PDEs for weather prediction uses your forward differences for spatial derivatives.
  - Trapezoidal integration aggregates $CO_2$ flux over geographic grids.
- Example:
  - Hurricane Tracking: Integrating wind velocity fields to predict paths.

## 9. Machine Learning & Data Science



- Connection:
  - Gradient descent employs your first-order derivatives (Model 2).
  - Numerical integration computes probabilities in Bayesian inference.
- Application:
  - Neural Networks: Training via backpropagation (chain rule of derivatives).

# References

**Text Books:**


1. S.R.K. Iyengar and R.K. Join, "*Numerical Methods*", New Age International Publishers.
2. Steven C. Chapra and R.P. Canale, " *Numerical Methods for Engineers*", Seventh Edition, MC Graw Hill Education.
3. Amos Gilat and Vish Subramaniam, "*Numerical Methods for Engineers and Scientists An Introduction with Applications using MATLAB*", WILEY Publisher.