Assiut University

Numerical Analysis Methods & Algorithms

Prepared by

Dr. Aml Abd El-Azim Farhat

Introduction

Numerical analysis has become an indispensable tool in solving complex mathematical problems that arise in science, engineering, and technology. With the rapid advancement of computational power, numerical methods and algorithms have grown in importance, offering practical solutions where analytical approaches fall short. This book, "Numerical Analysis: Methods and Algorithms," aims to provide an accessible yet comprehensive overview of the field, catering to beginners and those seeking to deepen their understanding of numerical techniques.

Numerical methods differ fundamentally from analytical methods in their approach to problem-solving. While analytical techniques seek exact solutions through mathematical expressions, numerical methods focus on approximating solutions with a desired level of accuracy. This distinction makes numerical analysis especially valuable for addressing real-world problems that often lack closed-form solutions, such as simulating fluid dynamics, optimizing complex systems, or solving nonlinear equations.

The numerical analysis applications are vast, ranging from engineering design and financial modeling to climate simulations and machine learning. These methods empower professionals and researchers to model and analyze otherwise computationally infeasible phenomena. By blending mathematical rigor with computational efficiency, numerical algorithms bridge the gap between theory and practice.

This book is dedicated to my beloved family, whose unwavering support has been my greatest source of strength, to my colleagues who have shared this journey of learning and discovery, and to my students, whose curiosity and determination continue to inspire me every day. May this work serve as a guide and a resource for all who embark on the fascinating journey of numerical analysis.

Contents

Chapter 1

Types of Errors and Their Propagations

Types of Errors in Numerical Analysis

Numerical analysis is a field of study that focuses on developing methods and algorithms to approximate solutions to mathematical problems. While numerical methods are essential for solving complex equations that do not have analytical solutions, their results are inherently approximate. These approximations lead to different types of errors. Understanding and computing these errors is crucial to ensure the reliability and accuracy of numerical solutions.

1. Types of Errors

Errors in numerical analysis can be broadly classified into the following categories:

Round-off Error

Round-off errors occur due to the finite precision of computer arithmetic. Computers cannot represent all numbers exactly, especially irrational numbers or numbers with many decimal places. For example, the value of π is approximated as 3.14159 in many systems, leading to small inaccuracies.

If a certain number is approximated to m decimal digits. Then the round of error er can be calculated as follows:

$$|er| = \frac{1}{2} \times 10^{-m},$$

where er is the round off error.

Truncation Error

Truncation errors arise when an infinite process, such as an infinite series or iterative method, is approximated by a finite process. For instance, using a Taylor series expansion up to a finite number of terms introduces truncation error.

Absolute Error

Absolute error measures the difference between the true value (T) and the approximate value (A):

$$Absolute\ Error\ = | T - A |.$$

This type of error is particularly useful when comparing results with slight variations.

Example 1.1 One of the most popular and often-used approximations is the approximation of $T = \pi$ by $A = 3.14$, so we can evaluate the absolute error of this operation as follows:

$$Absolute\ Error\ = | T - A | = | \pi - 3.14 | \approx 1.5926 \times 10^{-3}.$$

Relative Error

Relative error provides a normalized measure of error relative to the true value:

$$Relative\ Error = \left| \frac{T - A}{T} \right|$$

This type of error is particularly useful when comparing results with different magnitudes.

Example 1.2 The approximation of $T = 1000300$ by $A = 1000000$, we can evaluate the relative error of this operation as follows:

$$Relative\ Error = \left| \frac{T - A}{T} \right| = \left| \frac{1000300 - 1000000}{1000300} \right|$$
$$= 2.999 \times 10^{-4}$$

Percentage Error

Percentage error is the relative error expressed as a percentage:

$$Percentage\ Error = \left| \frac{T - A}{T} \right| \times 100\%$$

2. Assess Error Magnitude

Use norms, such as the L_2 -norm or L_∞ -norm, to assess the magnitude of errors in vector-based computations. Evaluate the condition number of the problem to estimate sensitivity to input errors.

3. Analyze Convergence

Examine how errors decrease as the step size or discretization parameter is refined.

4. Study Error Propagation

Trace how errors in initial conditions or intermediate steps impact the result.

5. Propagation of Error

When computations involve multiple steps or operations, errors from earlier steps can propagate and amplify in later stages. Analyzing error propagation is essential for designing stable algorithms.

Discretization Error

Discretization errors occur when continuous mathematical models are approximated using discrete methods. For example, solving differential equations numerically using finite difference methods introduces discretization errors.

Modeling Error

Modeling errors result from simplifying assumptions in the mathematical model. These errors are not inherent to the numerical method but stem from the difference between the real-world system and its mathematical representation.

6. Bound of Single-Value Function Error

In numerical analysis, the bound of single-value function error refers to the maximum possible deviation between the true value of a single-variable function and its approximation. This concept is crucial for estimating the reliability and accuracy of numerical approximations such as Taylor series, interpolation, or numerical integration. To compute the bound of a single-value function error, follow these steps:

The error E_f is often expressed as the difference between the actual function value $f(x)$ and its approximation $f(x^*)$ at $x = x^* + er$:

$$E_f = f(x) - f(x^*),$$

i.e. $$E_f = f(x^* + er) - f(x^*),$$

then, impose the Taylor's expansion of the 1st term we have

$$E_f = f(x^*) + \frac{er}{1!}f'(x^*) + \frac{er^2}{2!}f''(x^*) + O(er^n) - f(x^*).$$

Consequently, if E_f is minimal (and the second and higher derivatives f'' evaluated at x^* are not too big, and er is sufficiently small to disregard the terms of order $O(er^n)$for $n \geq 3$), we observe that

$$E_f \approx erf'(x^*),$$

thus

$$|E_f| \approx |er||f'(x^*)|$$

For $|er| \leq \frac{1}{2} \times 10^{-m}$, therefore

$$\left|E_f\right| \leq \frac{1}{2} \times 10^{-m}|f'(x^*)|$$

This implies to that the error bounds are

$$-\frac{1}{2} \times 10^{-m}|f'(x^*)| \leq E_f \leq \frac{1}{2} \times 10^{-m}|f'(x^*)|$$

Example 1.3 Evaluate the function error bounds of $f(x) = \sqrt{e^{3x} - 1}$ at $x^* = 1.25$

Solution:

For $f(x) = \sqrt{e^{3x} - 1}$, $f'(x) = \frac{3e^{3x}}{2\sqrt{e^{3x}-1}}$.

And $x^* = 2.15 \Rightarrow m = 2$.

$$\because \left|E_f\right| \leq \frac{1}{2} \times 10^{-m} |f'(x^*)|$$

$$\therefore \left|E_f\right| \leq \tfrac{1}{2} \times 10^{-2} \left| \frac{3e^{3(1.25)}}{2\sqrt{e^{3(1.25)}-1}} \right|,$$

$$\left|E_f\right| \leq 0.04949.$$

Hence

$$-0.04949 \leq E_f \leq 0.04949.$$

7. Bound of Multi-Variable Function Error

Let $f(\boldsymbol{x})$ be a multivariable function, and $\boldsymbol{x}^*$ be the point around which we approximate $f(\boldsymbol{x})$. Assume $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{er}$, where $\boldsymbol{er}$ is the error vector.

The goal is to derive the bound of the error $\boldsymbol{E_f}$ between $f(\boldsymbol{x})$ and $f(\boldsymbol{x}^*)$.

For a sufficiently smooth function $f(\boldsymbol{x})$, the Taylor series expansion around $\boldsymbol{x}^*$ is:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*) \cdot \boldsymbol{er} + \frac{1}{2} \boldsymbol{er}^{\boldsymbol{T}} H(\boldsymbol{\xi}) \boldsymbol{er} + \cdots$$

where $\nabla f(\boldsymbol{x}^*)$ is the gradient of f evaluated at $\boldsymbol{x}^*$, $H(\boldsymbol{\xi})$ is the Hessian matrix (matrix of second partial derivatives) evaluated at some intermediate point $\boldsymbol{\xi}$ on the line segment between $\boldsymbol{x}^*$ and $\boldsymbol{x}$,

and $\boldsymbol{er} = \boldsymbol{x} - \boldsymbol{x}^*$. The higher-order terms are omitted for simplicity, as they become negligible for small $\boldsymbol{er}$

The error $\boldsymbol{E}_f$ is the difference between the actual function value $f(\boldsymbol{x})$ and the approximation $f(\boldsymbol{x}^*)$:

$$\boldsymbol{E_f} = f(\boldsymbol{x}) - f(\boldsymbol{x}^*).$$

Substituting the Taylor expansion and neglect the too small terms we obtain:

$$\boldsymbol{E_f} = \nabla f(\boldsymbol{x}^*) \cdot \boldsymbol{er}$$

Thus, the total error is bounded by:

$$| \boldsymbol{E}_f | \leq \| \nabla f(\boldsymbol{x}^*) \| \| \boldsymbol{er} \|$$

Or

$$| \boldsymbol{E}_f | \leq \frac{1}{2} \times \sum_{i=1}^{n} 10^{-m_i} \left| \frac{\partial f(\boldsymbol{x}^*)}{\partial x_i} \right|$$

Example 1.4 Evaluate the function error bounds of the multi-variable function $f(\boldsymbol{x}) = x_1^3 x_2 + 4x_1 + 5$ at $\boldsymbol{x}^* = (5.31, 1.224)$

Solution:

$\frac{\partial f}{\partial x_1} = 3x_1^2 x_2 + 4,$ $\frac{\partial f}{\partial x_2} = x_1^3,$

$m_1 = 2,$ $m_2 = 3,$

$$\because |\, \boldsymbol{E}_f \,| \leq \frac{1}{2} \times \sum_{i=1}^{n} 10^{-m_i} \left| \frac{\partial f(\boldsymbol{x}^*)}{\partial x_i} \right|$$

$$\therefore |\, \boldsymbol{E}_f \,| \leq \frac{1}{2} (10^{-2} |3(5.31)^2 (1.224) + 4| + 10^{-3} |(5.31)^3|)$$

$$-0.6125 \leq E_f \leq 0.6125$$

Conclusion

Errors are an inevitable part of numerical analysis, but their impact can be minimized through careful algorithm design, error estimation, and analysis. By understanding the types of errors and their computation, numerical analysts can improve the accuracy and reliability of their results, ensuring that numerical methods remain powerful tools for solving real-world problems.

Exercises 1

1. A rod is measured to be 12.35 cm, but the true length is 12.50 cm.
    - Find the absolute error.
    - Calculate the relative error as a percentage.
2. A scale reports the weight of a package as 8.45 kg, but the actual weight is 8.50 kg.
    - Determine the absolute error.
    - Compute the relative error in percentage terms.
3. A cashier rounds prices to the nearest 0.05 USD.
    - If the original price of an item is 23.47 USD, determine the bound of the rounding error.
    - What is the maximum possible rounding error in this case?
4. A cylinder's volume is calculated using $V = \pi r^2 h$, where $r = 5.23\,cm$ and $h = 12.6\,cm$, both rounded to two decimal places.
    - Determine the bound of the rounding error in the volume computation.
5. A car's speed $v(t) = 2.5t^2$ is modeled, where ttt is time in seconds. Suppose $t^* = 4.00$ is measured with an error bound of $\Delta t = 0.02\,s$.

- Estimate the bound of the error in $v(t)$.

6. The temperature $T(t) = 25 + 3sin(0.5t)$ is measured at $t^* = 2.0\,s$, with a measurement error of $\Delta t = 0.01\,s$.
   - Find the bound of the error in $T(t)$.

---

7. The pressure $P(V,T) = nRTV$ depends on volume V, temperature T, and constants n, R. For $V = 10.0\,L$, $T = 300\,K$, and error bounds $\Delta V = 0.1\,L$ and $\Delta T = 1\,K$:
   - Calculate the bound of the error in $P(V,T)$.
8. The resistance $R = \frac{\rho L}{A}$, where $\rho = 1.68 \times 10^{-8}\,\Omega \cdot m$, $L = 2.00\,m$, and $A = 1.50 \times 10^{-6}\,m^2$. The error bounds are $\Delta L = 0.01\,m$ and $\Delta A = 0.05 \times 10^{-6}\,m^2$.
   - Estimate the bound of the error in R.
9. The height $h(t) = v_0 t - \frac{1}{2}gt^2$ is modeled, where $v_0 = 20.0\,m/s$, $g = 9.8\,m/s^2$, and $t = 2.00\,s$. If $\Delta t = 0.02\,s$:
   - Find the bound of the error in h(t)h(t)h(t).
10. A profit function $P(x,y) = 50x + 30y - 10xy$, where x and y are quantities of two products. If $x = 100$, $y = 50$, and the error bounds are $\Delta x = 1$, $\Delta y = 2$:
    - Determine the bound of the error in $P(x,y)$.

Chapter 2

Operators and Differences

In this chapter, we will explore several essential operators that serve as foundational tools in the development of interpolation numerical methods. These operators play a pivotal role in deriving widely used techniques such as the Newton-Gregory forward and backward interpolation formulas, as well as the Newton divided difference method. By delving into the properties and applications of these operators, we aim to build a deeper understanding of their significance in numerical analysis, laying the groundwork for efficient and accurate solutions to problems involving interpolation and data approximation.

Forward Difference Operator Δ

The forward difference operator defined as:

$$\Delta f(x) \coloneqq f(x+h) - f(x),$$

$$\Delta y_k \coloneqq y_{k+1} - y_k.$$

Satisfies the following properties

- $\Delta^n(\alpha y_k + \beta y_l) = \alpha \Delta^n y_k + \beta \Delta^n y_l$
- $\Delta^m \Delta^n y_k \coloneqq \Delta^{m+n} y_k.$

- We can simply expand the forward differences of a function as:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$ $\cdots$
x_0	y_0			
		$\Delta y_0 = y_1 - y_0$		
x_1	y_1		$\Delta^2 y_0 = \Delta y_1 - \Delta y_0$	
		$\Delta y_1 = y_2 - y_1$		$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0$
x_2	y_2			
$\vdots$	$\vdots$			

Example 2.1 Expand the following expressions

i) Δy_9,

ii) $\Delta(12y_{\frac{3}{2}} + 7y_{-5})$,

iii) $\Delta^2 y_6$,

iv) $\Delta^3 y_k$ and then find $\Delta^3 y_7$.

Solution:

i) $\Delta y_9 = y_{10} - y_9$.

ii) $\Delta\left(12y_{\frac{3}{2}} + 7y_{-5}\right) = 12\Delta y_{\frac{3}{2}} + 7\Delta y_{-5} = 12\left(y_{\frac{5}{2}} - y_{\frac{3}{2}}\right) + 7(y_{-4} - y_{-5})$.

iii) $\Delta^2 y_6 = \Delta\Delta y_6 = \Delta(y_7 - y_6) = (y_8 - y_7) - (y_7 - y_6) = y_8 - 2y_7 + y_6$.

iv) The third forward difference can be deduced as

$$\Delta^3 y_k = \Delta(\Delta^2 y_k) = \Delta(y_{k+2} - 2y_{k+1} + y_k)$$
$$= (y_{k+3} - y_{k+2}) - 2(y_{k+2} - y_{k+1}) + (y_{k+1} - y_k)$$
$$= y_{k+3} - 3y_{k+2} + 3y_{k+1} - y_k.$$

Therefore, for $k = 7$:

$$\Delta^3 y_7 = y_{10} - 3y_9 + 3y_8 - y_7.$$

Example 2.2 Let $f(x) = x^4 - 3x + 5$, construct the forward differences for the sequentially values $x = 1,2,3,4,5,6$

Solution:

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$
1	3					
		12				
2	15		50			
		62		60		
3	77		110		24	
		172		84		0
4	249		194		24	
		366		108		
5	615		302			
		668				
6	1283					

Backward Difference Operator ∇

The forward difference operator defined as:

$$\nabla f(x) \coloneqq f(x) - f(x-h),$$

$$\nabla y_k \coloneqq y_k - y_{k-1}.$$

Satisfies the following properties

- $\nabla^n(\alpha y_k + \beta y_l) = \alpha \nabla^n y_k + \beta \nabla^n y_l$
- $\nabla^m \nabla^n y_k \coloneqq \nabla^{m+n} y_k.$
- We can simply expand the backward differences of a function as:

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
x_{n-2}	y_{n-2}				
		∇y_{n-1}		$\nabla^3 y_n$	
		$= y_{n-1} - y_{n-2}$		$= \nabla^2 y_n - \nabla^2 y_{n-1}$	
x_{n-1}	y_{n-1}		$\nabla^2 y_n$		
			$= \nabla y_n - \nabla y_{n-1}$		
		$\nabla y_n = y_n - y_{n-1}$			
x_n	y_n				

Example 2.3 Expand the following expressions

i) $\nabla y_9,$

ii) $\nabla(12y_{\frac{3}{2}} + 7y_{-5})$,

iii) $\nabla^2 y_6,$

iv) $\nabla^3 y_k$ and then find $\nabla^3 y_7$.

Solution:

i) $\nabla y_9 = y_9 - y_8$.

ii) $\nabla\left(12y_{\frac{3}{2}} + 7y_{-5}\right) = 12\nabla y_{\frac{3}{2}} + 7\nabla y_{-5} = 12\left(y_{\frac{3}{2}} - y_{\frac{1}{2}}\right) + 7(y_{-5} - y_{-6})$.

iii) $\nabla^2 y_6 = \nabla\nabla y_6 = \nabla(y_7 - y_6) = (y_7 - y_6) - (y_6 - y_5) = y_6 - 2y_5 + y_4$.

iv) The third backward difference can be deduced as

$$\nabla^3 y_k = \nabla(\nabla^2 y_k) = \nabla(y_k - 2y_{k-1} + y_{k-2})$$
$$= (y_k - y_{k-1}) - 2(y_{k-1} - y_{k-2}) + (y_{k-2} - y_{k-3})$$
$$= y_k - 3y_{k-1} + 3y_{k-2} - y_{k-3}.$$

Therefore, for $k = 7$:

$$\Delta^3 y_7 = y_7 - 3y_6 + 3y_5 - y_4.$$

Example 2.2 Let $f(x) = x^3 - 2x + 4$, construct the forward differences for the sequentially values $x = 1,2,3,4,5$

Solution:

x	f	∇f	$\nabla^2 f$	$\nabla^3 f$	$\nabla^4 f$
1	3				
		5			
2	8		12		

		17		6	
3	25		18		0
		35		6	
4	60		24		
		59			
5	119				

Central Difference Operator δ

The forward difference operator defined as:

$$\delta f(x) \coloneqq f\left(x+\frac{h}{2}\right)-f\left(x-\frac{h}{2}\right),$$

$$\delta y_k \coloneqq y_{k+\frac{1}{2}}-y_{k-\frac{1}{2}}.$$

Satisfies the following properties

- $\delta^n(\alpha y_k+\beta y_l)=\alpha\delta^n y_k+\beta\delta^n y_l$
- $\delta^m\delta^n y_k \coloneqq \delta^{m+n}y_k.$

Example 2.3 Expand the following expressions

i) $\delta y_2,$

ii) $\delta(10y_{\frac{3}{2}})$,

iii) $\delta^2 y_6,$

iv) $\delta^3 y_k$ and then find $\delta^3 y_7.$

Solution:

i) $\delta y_2 = y_{\frac{5}{2}} - y_{\frac{3}{2}}.$

ii) $\delta\left(10y_{\frac{3}{2}}\right) = 10\delta y_{\frac{3}{2}} = 10(y_2 - y_1).$

iii) $\delta^2 y_6 = \delta\delta y_6 = \delta\left(y_{\frac{13}{2}} - y_{\frac{11}{2}}\right) = \delta y_{\frac{13}{2}} - \delta y_{\frac{11}{2}} = (y_7 - y_6) - (y_6 - y_5).$

.

Example 2.2 Let $f(x) = x^2$, construct the central differences for the real values $x = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, \frac{9}{2}$

Solution:

x	f	δf	$\delta^2 f$	$\delta^3 f$
$\frac{1}{2}$	$\frac{1}{4}$			
		6		
$\frac{5}{2}$	$\frac{25}{4}$		0	
		6		2
$\frac{7}{2}$	$\frac{49}{4}$		2	
		8		
$\frac{9}{2}$	$\frac{81}{4}$			

Divided Differences [. , .]

x	y	$[\bullet,\bullet]$	$[\bullet,\bullet,\bullet]$	$[\bullet,\bullet,\bullet,\bullet]$ ⋯
x_0	y			
		$[x_0, x_1]$ $= \frac{f(x_1) - f(x_0)}{x_1 - x_0}$		
x_1	y		$[x_0, x_1, x_2]$ $= \frac{[x_1, x_2] - [x_0, x_1]}{x_2 - x_0}$	
		$[x_1, x_2]$ $= \frac{f(x_2) - f(x_1)}{x_2 - x_1}$		$[x_0, x_1, x_2, x_3]$ $= \frac{[x_1, x_2, x_3] - [x_0, x_1, x_2]}{x_3 - x_0}$
x_2	y			
⋮	⋮			

The divided difference of a real valued function $f(x)$ defined as follow:

$$[\mathrm{x}_0, \mathrm{x}_1] \coloneqq \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

$$[\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2] \coloneqq \frac{[\mathrm{x}_1, \mathrm{x}_2] - [\mathrm{x}_0, \mathrm{x}_1]}{x_2 - x_0}$$

$$[\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3] \coloneqq \frac{[\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3] - [\mathrm{x}_0, \mathrm{x}_1, \mathrm{x}_2]}{x_3 - x_0}$$

Example 2.3 Let $f(x) = x^3$, construct the divided differences for the real values $x = a, b, c, d$

Solution:

$$[a,b] = \frac{f(b)-f(a)}{b-a} = \frac{b^3-a^3}{b-a} = a^2+ab+b^2$$

$$[a,b,c] = \frac{[b,c]-[a,b]}{c-a}$$

$$= \frac{(b^2+bc+c^2)-(a^2+ab+b^2)}{c-a}$$

$$= a+b+c$$

$$[a,b,c,d] = \frac{[b,c,d]-[a,b,c]}{d-a}$$

$$= \frac{(b+c+d)-(a+b+c)}{d-a} = 1$$

x	f	$[.,.]$	$[.,.,.]$	$[.,.,.,.]$	$[.,.,.,.,.]$
a	a^3				
		a^2+ab+b^2			
b	b^3		$a+b+c$		
		b^2+bc+c^2		1	
c	c^3		$b+c+d$		0
		c^2+cd+d^2		1	
d	d^3		$c+d+e$		
		d^2+de+e^2			
e	e^3				

Example 2.4 Let $f(x) = x^4$, construct the divided differences for the real values $x = 1,3,4,5,7$

Solution:

x	$f(x)$	$[.,.]$	$[.,.,.]$	$[.,.,.,.]$	$[.,.,.,.,.]$
1	1				
		40			
3	81		8.3333		
		65		11.6666	
4	256		55		7.9583
		175		59.4166	
5	625		237.666		
		888			
7	2401				

The Derivative Operator *D*

$$\mathbf{D}f(x) \coloneqq \frac{\mathrm{df(x)}}{\mathrm{dx}} = f'(x) = lim_{h\to 0}\ \frac{f(x+h)-f(x)}{h},$$

$$\mathbf{D^2}f(x) \coloneqq \frac{\mathrm{d^2f(x)}}{\mathrm{dx^2}} = f''(x).$$

$$\mathbf{D^n}f(x) \coloneqq \frac{\mathrm{d^nf(x)}}{\mathrm{dx^n}} = f^{(n)}(x).$$

The Shift Operator E

The Importance of Shift Operator:

- Simplification of Difference Equations: The shift operator provides a concise way to represent finite difference equations, which are crucial in approximating solutions to differential equations.
- Connection Between Continuous and Discrete Systems: It bridges continuous systems (differential equations) and discrete systems (difference equations), making it essential for numerical methods like finite difference and numerical interpolation.
- Compact Notation: It simplifies the representation of iterative or recursive methods, particularly when analyzing numerical stability or convergence.
- Application in Numerical Differentiation: It is used to derive formulas for numerical differentiation and integration.

The shift operator E defined as:

$$\mathbf{E}f(x) \coloneqq f(x+h),$$

$$\mathbf{E}y_k \coloneqq y_{k+1}.$$

$$\mathbf{E^s}y_k \coloneqq y_{k+s}, \forall s \in R.$$

Properties of the Shift Operator

1. Linearity: The shift operator is linear, meaning:

$$\boldsymbol{E}(c_1 y_1 + c_2 y_2) = c_1 \boldsymbol{E}(y_1) + c_2 \boldsymbol{E}(y_2)$$

   where c_1 and c_2 are constants.

2. Commutativity: If E^m and E^n denote $m-$ and $n-$step shifts, then:

$$E^m \cdot E^n = E^{m+n}.$$

3. Identity Property: E^0 is the identity operator, such that:

$$E^0\big(y(x)\big) = y(x).$$

4. Inverse: The inverse of E, denoted E^{-1}, shifts the function or sequence backward:

$$E^{-1}\big(y(x)\big) = y(x-h)\,; \quad or \quad E^{-1}(y_n) = y_{n-1}$$

5. Iteration: Repeated application of E corresponds to a multiple-step shift:

$$E^m(y_n) = y_{n+m}; \qquad or \quad E^m\big(y(x)\big) = y(x+mh).$$

   Examples

1. Forward Difference Representation: The forward difference operator Δ is expressed using E:

$$\Delta y_n = y_{n+1} - y_n = E(y_n) - y_n = (E-1)y_n \quad \Rightarrow \Delta \equiv E - 1$$

2. Backward Difference Representation: The forward difference operator ∇ is expressed using E:

$$\nabla y_n = y_n - y_{n-1} = y_n - E^{-1}y_n = (1 - E^{-1})y_n \quad \Rightarrow \nabla \equiv 1 - E^{-1} \Rightarrow E = \frac{1}{1-\nabla}$$

3. Numerical Approximation of Derivatives: Using the shift operator, the derivative $y'(x)$ can be approximated by:

$$y'(x) = Dy(x) \approx \frac{E\big(y(x)\big) - y(x)}{h}$$

4. Useful relation via Taylor's expansion

$$Ey(x) = y(x+h) = y(x) + \frac{h}{1!}y'(x) + \frac{h^2}{2!}y''(x) + \frac{h^3}{3!}y'''(x) + \cdots$$

$$\therefore Ey(x) = y(x) + \frac{h}{1!}Dy(x) + \frac{h^2}{2!}D^2y(x) + \frac{h^3}{3!}D^3y(x) + \cdots$$

$$\therefore Ey(x) = \left[1 + \frac{hD}{1!} + \frac{(hD)^2}{2!} + \frac{(hD)^3}{3!} + \cdots\right]y(x)$$

$$\therefore Ey(x) = e^{hD}y(x)$$

i.e. $E \equiv e^{hD}$

The Mean Operator μ

Defined as:

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

$$\therefore \mu \equiv \frac{E^{\frac{1}{2}} + E^{-\frac{1}{2}}}{2}$$

Exercises 2

1. Given $f(x) = x^2$, compute $\Delta f(x)$ and $\nabla f(x)$ at $x = 2$ for $h = 1$.
2. Prove the relationship $\Delta = E - 1$.
3. For $f(x) = x^3$, compute $\Delta^2 f(x)$ and $\nabla^2 f(x)$.
4. Show that for $f(x) = ax^2 + bx + c$, $\Delta^2 f(x)$ is constant.
5. Show that $\Delta^{-1} f(x) = \sum f(x)$, where the sum represents the antithetical (discrete integration) operation.
6. If $f(x) = sin(x)$, find $Ef(x)$ at $x = \frac{\pi}{4}$ for $h = 0.1$.
7. Prove that $E^n f(x) = f(x + nh)$.
8. Verify that $E\Delta = \Delta E$.
9. Prove that $E = 1 + \Delta$ when acting on a function f(x).
10. Given $f(x) = e^x$, compute $\mu f(x)$ $h = 0.1$.
11. Show that $\mu = \frac{E+E^{-1}}{2}$.
12. If $f(x) = x^2$, find $\mu f(x)$ and compare it to $\frac{f(x+h)+f(x-h)}{2}$.
13. Show that $\frac{\Delta f(x)}{h} \approx Df$(x) for small h.
14. For $f(x) = x^3$, compute the error between $\frac{\Delta f(x)}{h}$ and $Df(x)$.
16. For $f(x) = x^2 + x$, compute $\delta f(x)$ for $h = 1$.

17. Show that $\delta = \frac{E-E^{-1}}{2}$.

18. Compare $\delta f(x)$ with $\frac{\Delta f(x)+\nabla f(x)}{2}$.

19. If $f(x) = cos\ (x)$, compute $\Delta\delta f(x)$ and $\delta\Delta f(x)$ for $h = 0.5$.

20. For $f(x) = x^3$, compute $\Delta^3 f(x)$, $\nabla^3 f(x)$,$[a, b]$, $[a, b, c]$, $[a, b, c, d]$, and $\delta^3 f(x)$.

Chapter 3

Interpolation over Equidistance Points

Interpolation is a fundamental concept in numerical analysis, where the goal is to construct a function that approximates a set of given data points. When the data points are equidistant, meaning they are evenly spaced along the independent variable's axis, the interpolation process becomes particularly efficient and structured. This special case is widely studied due to its simplicity and practical relevance in various applications.

Importance of Interpolation Over Equidistant Points

1. **Efficient Computation:** The equidistant spacing of points simplifies the mathematical expressions involved in interpolation. Polynomial interpolations like those using Newton’s forward and backward difference formulas are optimized for such datasets, reducing computational effort.
2. **Structured Methods:** The uniformity of equidistant points allows the use of difference operators such as forward, backward, and central differences. These operators are crucial for deriving interpolation formulas and approximating derivatives or integrals.

3. **Improved Numerical Stability:** While high-degree polynomial interpolation over arbitrary points can suffer from numerical instability (Runge's phenomenon), equidistant points allow controlled behavior when used with appropriate techniques like piecewise interpolation.

Applications of Interpolation Over Equidistant Points

1. **Signal Processing:** Equidistant interpolation is used in reconstructing signals sampled at uniform intervals. This forms the basis for digital audio, video processing, and other signal manipulation techniques.
2. **Data Approximation:** In fields like physics, engineering, and finance, equidistant interpolation helps approximate functions when analytical expressions are unavailable, enabling simulations and predictions.
3. **Numerical Solutions of Differential Equations:** Interpolation methods, especially with equidistant nodes, are employed to discretize continuous problems in finite difference methods.
4. **Image Resizing and Reconstruction:** Algorithms for scaling or reconstructing images often assume uniformly spaced pixel grids, utilizing interpolation for smooth transformations.
5. **Geophysics and Meteorology:** Equidistant interpolation assists in creating models of spatial phenomena based on

uniformly collected data, such as temperature distributions or seismic activity.

Interpolation over equidistant points offers numerous advantages, challenges like numerical instability in higher-order interpolations remain. To address this, piecewise interpolations such as splines or using Chebyshev nodes for non-uniform spacing in some scenarios are viable alternatives.

In conclusion, interpolation over equidistant points is a cornerstone of numerical analysis, enabling precise, efficient, and practical solutions across a wide range of scientific and engineering disciplines. Its structured nature ensures that it remains a preferred approach when working with uniformly sampled data.

Newton Forward Method

Given a dataset consisting of $n + 1$ equidistant points $\{x_i, y_i\}_{i=0}^{n}$, where:

- x_i are the independent variable values,
- y_i are the corresponding dependent variable values $\big(y_i = f(x_i)\big)$,

we aim to approximate the value of the function $y(x) = f(x)$ at a specific point x_s within the range of the given points, using the

Newton Forward Interpolation Formula.

Constraints and Assumptions:

1. The points x_i are equidistant, such that $h = x_{i+1} - x_i$ is constant.
2. The target point x_s lies within the range $[x_0, x_n]$.
3. The function $f(x)$ is sufficiently smooth over the interval, allowing accurate interpolation.

Mathematical Representation:

The Newton Forward Interpolation Formula for approximating $y(x_s)$ is given by:

$$y(x_s) \approx y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots + \frac{p(p-1)\cdots(p-n+1)}{n!}\Delta^n y_0$$

where:

- $p = \frac{x_s - x_0}{h}$ is the normalized distance from the first point.
- $\Delta^k y_0$ are the $k-th$ order forward differences starting from y_0.

Objective:

The primary goal is to compute $y(x_s)$ using this formula with minimal error, ensuring that the interpolation process efficiently leverages the equidistant nature of the data points and the forward difference scheme.

Practical Significance:

This method is particularly useful for:

- Evaluating tabulated data at intermediate points.
- Approximating function values in evenly spaced datasets.
- Serving as a foundational approach for higher-order interpolation and numerical differentiation.

By solving this problem, we enable accurate and efficient interpolation that finds applications in physics, engineering, finance, and other computational fields requiring numerical estimation of functions.

Derivation

We want to evaluate $y(x_s)$ so

$$y(x_s) \approx P_n(x_s) = E^p y_0 = (1+\Delta)^p y_0.$$

Then impose the binomial expansion we obtain

$$y(x_s) \approx P_n(x_s) = \left(1 + \frac{p}{1!}\Delta + \frac{p(p-1)}{2!}\Delta^2 + \frac{p(p-1)(p-2)}{3!}\Delta^3 + \cdots\right) y_0,$$

$$\therefore y(x_s) \approx P_n(x_s) = y_0 + \frac{p}{1!}\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots .$$

A useful Algorithm

Problem statement given as

x	x_0	x_1	x_2	$\cdots$	x_n
y	y_0	y_1	y_2	$\cdots$	y_n

Find $y_s = y(x_s)$ or $y'_s = y'(x_s)$

Solution Algorithm

- **Step 1 Calculate** $h = x_1 - x_0$ and $p = \dfrac{x_s - x_0}{h}$
- **Step 2 Construct Forward Differences Table**
- **Step 3 Substitute in**

$$y_s = y(x_s) \approx P_n(x_s) = y_0 + \frac{p}{1!}\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots$$

Example 3.1

The temperature of a city at specific times of the day is recorded in the table below:

Time (x)	6 AM	9 AM	12 AM	15 PM	18 PM
Temperature (y)	15°	18°	21°	25°	22°

Estimate the temperature at **10 AM**.

Solution:

We will use the Newton Forward Interpolation formula to calculate the temperature at $x = 10$ (time = 10 hours). The formula is:

$$y(x_s) \equiv P(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots$$

- **Step 1 Calculate**

$h = x_1 - x_0$ =9-6= $\boxed{3}$ and $p = \frac{x_s - x_0}{h} = \frac{10-6}{3} = \boxed{\frac{4}{3}}$

- **Step 2 Construct Forward Differences Table**

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
6	15				
		3			
9	18		0		
		3		1	
12	21		1		-9
		4		-8	
15	25		-7		
		-3			
18	22				

- **Step 3 Substitute in**

$$y(x_s) \equiv P(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots$$

$$y(10) \equiv P(10) = 15 + \frac{4}{3}(3) + \frac{\frac{4}{3}\left(\frac{4}{3}-1\right)}{2!}(0) + \frac{\frac{4}{3}\left(\frac{4}{3}-1\right)\left(\frac{4}{3}-2\right)}{3!}(1) + \frac{\frac{4}{3}\left(\frac{4}{3}-1\right)\left(\frac{4}{3}-2\right)\left(\frac{4}{3}-3\right)}{4!}(-9).$$

$$\therefore y(10) \approx 18.76^{\circ}$$

Newton Backward Method

Given a dataset consisting of $n+1$ equidistant points $\{x_i, y_i\}_{i=0}^{n}$, where:

- x_i are the independent variable values,
- y_i are the corresponding dependent variable values $\left(y_i = f(x_i)\right)$,

we aim to approximate the value of the function $y(x) = f(x)$ at a specific point x_s within the range of the given points, using the **Newton Backward Interpolation Formula**.

Constraints and Assumptions:

4. The points x_i are equidistant, such that $h = x_{i+1} - x_i$ is constant.
5. The target point x_s lies within the range $[x_0, x_n]$.
6. The function $f(x)$ is sufficiently smooth over the interval, allowing accurate interpolation.

Mathematical Representation:

The Newton Forward Interpolation Formula for approximating $y(x_s)$ is given by:

$$y(x_s) \approx y_n + p\nabla y_n + \frac{p(p+1)}{2!}\nabla^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_n + \cdots + \frac{p(p+1)\cdots(p+n-1)}{n!}\nabla^n y_n$$

where:

- $p = \frac{x_s - x_n}{h} or\ p = \frac{x_s - x_{n-1}}{h}$ is the normalized distance from the first point.
- $\Delta^k y_0$ are the $k-th$ order forward differences starting from y_0.

Objective:

The primary goal is to compute $y(x_s)$ using this formula with minimal error, ensuring that the interpolation process efficiently leverages the equidistant nature of the data points and the forward difference scheme.

Practical Significance:

This method is particularly useful for:

- Evaluating tabulated data at intermediate points.
- Approximating function values in evenly spaced datasets.
- Serving as a foundational approach for higher-order interpolation and numerical differentiation.

By solving this problem, we enable accurate and efficient interpolation that finds applications in physics, engineering, finance, and other computational fields requiring numerical estimation of functions.

Derivation

We want to evaluate $y(x_s)$ so

$$y(x_s) \approx P_n(x_s) = E^p y_n = (1-\nabla)^{-p} y_n.$$

Then impose the binomial expansion we obtain

$$y(x_s) \approx P_n(x_s) = \left(1 + \frac{p}{1!}\nabla + \frac{p(p+1)}{2!}\nabla^2 + \frac{p(p+1)(p+2)}{3!}\nabla^3 + \cdots\right) y_n,$$

$$\therefore y(x_s) \approx P_n(x_s) = y_n + \frac{p}{1!}\nabla y_n + \frac{p(p+1)}{2!}\nabla^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_n + \cdots.$$

A useful Algorithm

Problem statement given as

x	x_0	x_1	x_2	$\cdots$	x_n
y	y_0	y_1	y_2	$\cdots$	y_n

Find $y_s = y(x_s)$ or $y'_s = y'(x_s)$

Solution Algorithm

- **Step 1 Calculate**

$$h = x_1 - x_0 \text{ and } p = \frac{x_s - x_n}{h} \text{ or } p = \frac{x_s - x_{n-1}}{h}$$

- **Step 2 Construct Backward Differences Table**
- **Step 3 Substitute in**

$$y_s = y(x_s) \approx P_n(x_s) = y_n + \frac{p}{1!}\nabla y_n + \frac{p(p+1)}{2!}\nabla^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_n + \cdots$$

Example 3.2

The population of a city measured by (Millions) at specific years is recorded in the table below:

Year (x)	1995	1997	1999	2001	2003
Population (y)	12	13	15	15	16

Estimate the population at **2002**.

Solution:

We will use the Newton Backward Interpolation formula to calculate the population at $x = 2002$. The formula is:

$$y_s = y(x_s) \approx P_n(x_s) = y_n + \frac{p}{1!}\nabla y_n + \frac{p(p+1)}{2!}\nabla^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_n + \cdots$$

- **Step 1 Calculate**

$$h = x_n - x_{n-1} = 2003 - 2001 = \boxed{2} \text{ and } p = \frac{x_s - x_{n-1}}{h} = \frac{2002 - 2003}{2} = \boxed{-\frac{1}{2}}$$

- **Step 2 Construct Backward Differences Table**

x	y	∇y	∇y	$\nabla^3 y$	$\nabla^4 y$
1995	12				

		1			
1997	13		1		
		2		-3	
1999	15		-2		6
		0		3	
2001	15		1		
		1			
2003	16				

- **Step 3 Substitute in**

$$y(x_s) \equiv P(x) = y_n + p\nabla y_n + \frac{p(p+1)}{2!}13^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_0 + \cdots$$

$$y(2002) \equiv P(2002) = 16 + \left(-\frac{1}{2}\right)(1) + \frac{\left(-\frac{1}{2}\right)\left(-\frac{1}{2}+1\right)}{2!}(1) + \frac{\left(-\frac{1}{2}\right)\left(-\frac{1}{2}+1\right)\left(-\frac{1}{2}+2\right)}{3!}(3) + \frac{\left(-\frac{1}{2}\right)\left(-\frac{1}{2}+1\right)\left(-\frac{1}{2}+2\right)\left(-\frac{1}{2}+3\right)}{4!}(6).$$

$\therefore y(10) \approx 14.9 = 15$ approximately

Exercises 3

1. Use the forward Gregory formula to estimate the temperature at 7:30 AM.

Time (AM)	7: 00	8: 00	9: 00	10: 00
Temperature (C°)	20.5	22.1	23.4	25.0

2. Use the backward Gregory formula to estimate the population in 1945.

Year	1920	1930	1940	1950	1960
Population (in millions)	5.5	5.8	6.4	7.2	7.9

3. Predict the brightness of a star at 2.5 hours using the forward Gregory formula.

Time (h)	1	2	3	4	5	6
Brightness (Magnitude)	4.5	4.8	5.1	5.4	5.5	5.2

4. Estimate the velocity of fluid at a depth of 6.5 meters using the backward Gregory formula.

Depth (m)	4	5	6	7
Velocity (m/s)	1.2	1.5	1.8	2.1

5. Estimate the GDP at 2006 using the forward Gregory formula.

Year	2000	2005	2010	2015
GDP (Trillion USD)	12	14.2	16.8	19.5

6. Predict the drug concentration at 3.5 hours using the backward Gregory formula.

Time (hours)	1	2	3	4
Drug Concentration (mg/L)	15	12.5	10.8	9.2

7. Estimate the heat flux at a distance of 1.5 meters using the forward Gregory formula.

Distance (m)	1	2	3	4
Heat Flux (W/m^2)	55	55.2	60.8	67

---

8. Material Science

Problem: Predict stress at a strain of 0.0255 using the backward Gregory formula.

Strain	0.010	0.015	0.020	0.025
Stress (MPa)	100	120	140	180

Chapter 4

Interpolation over Non-Equidistance Points

Newton Divided Differences Method

Newton's Divided Differences Method is a widely used technique in numerical analysis for constructing an interpolating polynomial given a set of data points. Unlike other interpolation techniques, this method efficiently handles unevenly spaced data and provides a compact form for the interpolating polynomial. Its recursive structure and adaptability make it foundational in numerical mathematics and computational applications.

Newton's Divided Differences Method is a powerful technique for interpolating a function $y(x)$ given a set of discrete data points $\{x_i, y_i\}_{i=0}^{n}$. This method constructs an interpolating polynomial using a recursive formulation of divided differences, making it highly efficient and numerically stable, particularly when dealing with unequally spaced data.

Importance

Interpolation is a fundamental task in numerical analysis and engineering, providing estimates for unknown function values based on known data points. Newton's Divided Differences Method is especially useful because:

- It can handle non-equally spaced data points, unlike simpler methods like Lagrange interpolation.
- It allows for an incremental approach: if new data points are added, the polynomial can be updated without redoing all calculations.
- Its recursive nature provides a structured and compact representation of the polynomial.

Applications

Newton's Divided Differences Method finds applications in:

1. **Engineering**: Estimating values of functions in simulations and models.
2. **Science**: Interpolating experimental data for analysis.
3. **Computer Graphics**: Smooth rendering of curves.
4. **Signal Processing**: Reconstruction of sampled signals.
5. **Economics and Finance**: Interpolating and forecasting trends from discrete data.

Derivation of Newton's Divided Differences Formula

Given a set of points $\{x_i, y_i\}_{i=0}^{n}$, the goal is to construct an interpolating polynomial $P_n(x)$ such that: $P_n(x_i) = y_i$ for $i = 0,1, \dots, n$.

1. **Form of the Polynomial**: The polynomial is expressed incrementally as:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

2. **Divided Differences**: The coefficients $a_0, a_1, \dots, a_n$ are calculated using divided differences:
    - First-order divided difference:

$$[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}.$$

    - Second-order divided difference:

$$[x_i, x_{i+1}, x_{i+2}] = \frac{[x_{i+1}, x_{i+2}] - [x_i, x_{i+1}]}{x_{i+2} - x_i}.$$

    - Continue recursively for higher orders.

3. **Recursive Calculation**: The general formula is:

$$[x_{i0}, x_{i1}, \dots, x_{ik}] = \frac{f[x_{i1}, \dots, x_{ik}] - f[x_{i0}, \dots, x_{ik-1}]}{x_{ik} - x_{i0}}.$$

The coefficients a_i are directly related to the divided differences:

$a_0 = f(x_0)$, $a_1 = [x_0, x_1]$, $a_2 = [x_0, x_1, x_2]$, …, $a_n = [x_0, x_1, \dots, x_n]$.

Problem as

x	x_0	x_1	x_2	$\cdots$	x_n
y	y_0	y_1	y_2	$\cdots$	y_n

Find $y_p = y(x_p)$

Solution Algorithm

- **Step 1 Construct Divided Differences Table**

x	y	$1st\_Differnce$ $[.,.]$	$2nd\_Differnce$ $[.,.,.]$	$3rd\_Differnce$ $[.,.,.,.,.]$
x_0	y_0			
		$[x_0,x_1]=\frac{y_1-y_0}{x_1-x_0}$		
x_1	y_1		$[x_0,x_1,x_2]=\frac{[x_1,x_2]-[x_0,x_1]}{x_2-x_0}$	
		$[x_1,x_2]=\frac{y_2-y_1}{x_2-x_1}$		$[x_0,x_1,x_2,x_3]=\frac{[x_1,x_2,x_3]-[x_0,x_1,x_2]}{x_3-x_0}$
x_2 $\vdots$	y_2 $\vdots$	$\vdots$	$[x_1,x_2,x_3]=\frac{[x_2,x_3]-[x_1,x_2]}{x_3-x_1}$	

- **Step 2 Substitute in**

$$y_p = y(x_p) \approx P_n(x_p) =$$
$$y_0+(x_p-x_0)[x_0,x_1]+(x_p-x_0)(x_p-x_1)[x_0,x_1,x_2]+(x_p-x_0)(x_p-x_1)(x_p-x_2)[x_0,x_1,x_2,x_3]+\cdots$$

Example 4.1 Interpolate $y(x)$ at $x = 2.5$ using the data points:

$(x_0, y_0) = (1,2)$, $(x_1, y_1) = (2,4)$, $(x_2, y_2) = (3,6)$.

Solution:

x	1	2	3
y	2	4	6

- **Step 1 Construct Divided Differences Table**

x	y	$1st\_Differnce$ $[.,.]$	$2nd\_Differnce$ $[.,.,.]$
1	2		
		2	
2	4		0
		2	
3	6		

- **Step 2 Substitute in**

$$y_p = y(x_p) \approx P_n(x_p) =$$
$$y_0 + (x_p - x_0)[x_0, x_1] + (x_p - x_0)(x_p - x_1)[x_0, x_1, x_2] + (x_p - x_0)(x_p - x_1)(x_p - x_2)[x_0, x_1, x_2, x_3] + \cdots$$

$$y(2.5) \approx 2 + (2.5-1)(2) + (2.5-1)(2.5-2)(0) = 5.$$

Properties

- The degree of $P_n(x)$ is at most n.
- Divided differences are symmetric: their value remains unchanged regardless of the order of input points.

- Adding new data points requires minimal additional computation.

Lagrange Interpolation

Lagrange Interpolation is a polynomial interpolation method used to estimate the value of a function $y(x)$ at a given point x_s, using a set of known data points $\{x_i, y_i\}_{\{i=0\}}^{n}$. It is particularly useful when the exact form of the function is unknown, but its values at specific points are available.

Importance of Lagrange Interpolation

1. **Polynomial Approximation**: It provides a simple polynomial representation of a function, which can approximate the function over the range of given data points.
2. **Simplicity**: The method does not require solving systems of equations, unlike other interpolation methods such as Newton's divided difference.
3. **Versatility**: It is applicable in various fields such as numerical analysis, physics, engineering, and computer graphics, where function values are derived from discrete measurements or observations.

Mathematical Formulation

Given $n+1$ data points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, the Lagrange interpolating polynomial $P_n(x)$ is given by:

$$Pn(x) = \sum_{i=0}^{n} y_i L_i(x),$$

where $L_i(x)$ are the Lagrange basis polynomials defined as:

$$L_i(x) = \prod_{j=0 j \neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

The basis polynomial $L_i(x)$ is constructed such that $L_i(x_i) = 1$ and $L_i(x_j) = 0$ for $j \neq i$.

Derivation

1. **Basis Polynomial Construction**: To ensure $L_i(x_i) = 1$, the numerator of $L_i(x)$ must be zero for all $j \neq i$, and the denominator normalizes it to one:

$L_i(x) = \prod_{j=0 j \neq i}^{n} \frac{x - x_j}{x_i - x_j}.$

2. **Linear Combination**: The polynomial $P_n(x)$ is formed as a linear combination of the $L_i(x)$ polynomials, weighted by the corresponding function values y_i:

$$P_n(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x).$$

3. **Interpolation Property**: At each $x = x_k$, all basis polynomials $L_i(x)$ vanish except for $L_k(x_k) = 1$, ensuring $P_n(x_k) = y_k$.

Applications

1. **Numerical Computation**: Approximating functions when their analytical form is unknown.
2. **Data Analysis**: Filling in missing data points in datasets.
3. **Engineering**: Modeling relationships between variables in control systems or simulations.
4. **Computer Graphics**: Texture mapping and smooth curve generation.
5. **Astronomy**: Estimating planetary positions or trajectories from discrete data.

Example 4.2

Suppose we are given the data points $\{(1,1), (2,4), (3,9)\}$ and need to interpolate the value of $y(x_s)$ at $x_s = 1.5$.

1. **Form Basis Polynomials**: For $n = 2$:

$L_0(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{(x-2)(x-3)}{2}$,

$L_1(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -(x-1)(x-3)$,

$L_2(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{(x-1)(x-2)}{2}$.

2. **Construct Interpolating Polynomial**:

$P_2(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x)$,

$P_2(x) = \frac{(x-2)(x-3)}{2} - 4(x-1)(x-3) + 9\frac{(x-1)(x-2)}{2}$.

Evaluate at $x_s = 1.5$: Substituting $x_s = 1.5$ into $P_2(x)$, we find:

$P_2(1.5) \approx 2.25$.

Advantages

1. No need to recompute existing values when new data points are added (if starting from scratch).
2. Exact fit for $n+1$ points using an n-degree polynomial.

Limitations

1. **Oscillations**: In cases of evenly spaced points with a high-degree polynomial (Runge's phenomenon).
2. **Scalability**: Computationally expensive for large datasets as every new evaluation requires recalculating $L_i(x)$.

Conclusion

The Lagrange interpolation method provides an elegant and systematic approach to approximate a function using known data points. Despite its simplicity, careful consideration must be given to its limitations, especially for large datasets or non-uniform

distributions of data points. It remains a foundational technique in numerical analysis with widespread applications across disciplines.

Exercises 4

1. Estimate the temperature at $t = 7{:}30\ AM$ using both Lagrange and Newton's divided differences formulas.

Time (AM)	7	9	10
Temperature (°C)	15.2	18.6	21.4

2. A ball's height is recorded at various times. Use interpolation to estimate the height at $t = 1.5\ s$.

Time (s)	1	3	4
Height (m)	4.9	19.6	44.1

3. Predict the stock price at $t = 2.5$ days.

Time (days)	1	2	3	4
Stock Price (USD)	150	155	165	155

4. Estimate the population in 1995 using interpolation.

Year	1990	2000	2010
Population (M)	50.5	55.8	62.3

5. Estimate the brightness of a star at $t = 2.5\ hours$.

Time (hours)	2	3	5
Brightness (magnitude)	3.8	4.1	4.5

6. Predict the CO_2 concentration at $t = 25\ min$ using Lagrange interpolation.

Time (min)	20	30	40
CO2 Concentration (ppm)	410	420	435

7. Estimate stress at a strain of 0.025.

Strain	0.010	0.020	0.030
Stress (MPa)	120	140	180

8. Use interpolation to estimate heat flux at $x = 1.5\,m$.

Distance (m)	2	5	10
Heat Flux (W/m^2)	50	55.5	62.3

9. Predict the rocket's altitude at $t = 4.5\,s$ using Newton's formula.

Time (s)	4	5	7
Altitude (km)	2.8	3.4	4.2

10. Approximate $f(1.5)$ for $f(x) = x^2 + 2x + 1$ using interpolation on the given points.

x	1	2	3	4	5
$f(x)$	4	9	16	25	47

Chapter 5

The Numerical Differentiation

Numerical differentiation involves approximating the derivative of a function using discrete data points. It is a crucial tool in applied mathematics, physics, engineering, and computational sciences, especially when dealing with functions for which analytical differentiation is difficult or impossible. By approximating derivatives, numerical differentiation helps in solving differential equations, analyzing experimental data, and modeling real-world phenomena.

Numerous categories can provide several numerical differentiation formulae, such as the following:

From the Taylor series expansion. The most common finite difference formulas are:

1. ## Forward Difference Formula

   For a function $f(x)$ at point x with step size h:

   $$f'(x) \approx \frac{f(x+h)-f(x)}{h}$$

   Error Term: $O(h)$.

Backward Difference Formula

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

Error Term: $O(h)$.

Central Difference Formula

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Error Term: $O(h^2)$.

Second Derivative (Central Difference)

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Error Term: $O(h^2)$.

The derivations of the above formulas come from Taylor's series expansions.

Example 5.1 Approximate the derivative of $f(x) = sin(x)$ at $x = \frac{\pi}{4}$ using a step size $h = 0.1$.

1. **Forward Difference**:

$$f'(x) \approx \frac{sin\left(\frac{\pi}{4} + 0.1\right) - sin\left(\frac{\pi}{4}\right)}{0.1}$$

Substituting values:

$$f'(x) \approx \frac{0.79389 - 0.70711}{0.1} \approx 0.868$$

Exact derivative: $f'(x) = cos(\frac{\pi}{4}) = 0.7071.$

Absolute Error: $|0.868 - 0.7071| = 0.161.$

2. **Central Difference**:

$$f'(x) \approx \frac{sin\left(\frac{\pi}{4}+0.1\right)-sin\left(\frac{\pi}{4}-0.1\right)}{2(0.1)}$$

Substituting values:

$$f'(x) \approx \frac{0.79389 - 0.60876}{0.2} \approx 0.925$$

Absolute Error: $|0.925 - 0.7071| = 0.218.$

An alternative class of numerical differentiation formulas can be derived using the interpolation techniques discussed in Chapters 3 and 4. By representing the function $f(x)$ as a polynomial of degree n interpolated at a general point x, the derivative at any desired point can be computed directly and efficiently.

Example 5.2 Via the following data find an approximate polynomial of $y(x)$ and $y'(x)$ then find $y'(2)$.

x	1	3	4
y	3	19	33

Solution:

Here, we will impose Lagrange method to interpolate at a general point x as follows:

$$L_0(x) = \frac{(x-3)(x-4)}{(1-3)(1-4)} = \frac{1}{6}(x^2 - 7x + 12),$$

$$L_1(x) = \frac{(x-1)(x-4)}{(3-1)(3-4)} = \frac{-1}{2}(x^2 - 5x + 4),$$

$$L_2(x) = \frac{(x-1)(x-3)}{(4-1)(4-3)} = \frac{1}{3}(x^2 - 4x + 3),$$

$$y(x) \approx P_2(x) = \sum_{i=0}^{2} L_i(x) y_i$$

$$= \frac{3}{6}(x^2 - 7x + 12) - \frac{19}{2}(x^2 - 5x + 4) + \frac{33}{3}(x^2 - 4x + 3)$$

$$= 2x^2 + 1$$

$$\therefore y'(x) \approx 4x$$

$$\therefore y'(2) \approx 4(2) = 8.$$

Generally, we can write

$$y'^{(x)} \approx \sum_{i=0}^{n} L'_i(x) y_i.$$

Additional explicit formulas for numerical differentiation y' and y'' can be developed by deriving complementary expressions based

on the Newton-Gregory forward interpolation method, as outlined below:

- $y'(x_p) \approx \frac{1}{h}\left[\Delta y_0 + \frac{2p-1}{2}\Delta^2 y_0 + \frac{3p^2-6p+2}{6}\Delta^3 y_0 + \frac{2p^3-9p^2+11p-3}{12}\Delta^4 y_0 + \cdots\right]$
- $y'(x_0) \approx \frac{1}{h}\left[\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + \frac{\Delta^5 y_0}{5}\cdots\right]$
- $y''(x_p) \approx \frac{1}{h^2}\left[\Delta^2 y_0 + (p-1)\Delta^3 y_0 + \frac{6p^2-18p+11}{12}\Delta^4 y_0 + \cdots\right]$
- $y''(x_0) \approx \frac{1}{h^2}\left[\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \frac{5}{6}\Delta^5 y_0 \cdots\right]$

• ***A useful Algorithm***

Problem as

x	x_0	x_1	x_2	$\cdots$	x_n
y	y_0	y_1	y_2	$\cdots$	y_n

Find
$(i)\ y'(x_p)$
$(ii)\ y'(x_0)$
$(iii)\ y''(x_p)$
$(iv)\ y''(x_0)$

Solution Algorithm

- **Step 1 Calculate** $h = x_1 - x_0$ and $p = \frac{x_p - x_0}{h}$
- **Step 2 Construct Forward Differences Table**
- **Step 3 Substitute in the following relations**

- $y'(x_p) \approx \frac{1}{h}\left[\Delta y_0 + \frac{2p-1}{2}\Delta^2 y_0 + \frac{3p^2-6p+2}{6}\Delta^3 y_0 + \frac{2p^3-9p^2+11p-3}{12}\Delta^4 y_0 + \cdots\right]$
- $y'(x_0) \approx \frac{1}{h}\left[\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + \frac{\Delta^5 y_0}{5} \cdots\right]$
- $y''(x_p) \approx \frac{1}{h^2}\left[\Delta^2 y_0 + (p-1)\Delta^3 y_0 + \frac{6p^2-18p+11}{12}\Delta^4 y_0 + \cdots\right]$
- $y''(x_0) \approx \frac{1}{h^2}\left[\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \frac{5}{6}\Delta^5 y_0 \cdots\right]$

Example 5.3 Via the following equidistance data find $y'(3)$ and $y'(2)$.

x	2	4	6	8	10
y	34	742	3842	12214	29890

Solution:

Solution Algorithm

- **Step 1 Calculate**

$h = x_1 - x_0 = 4-2 = \boxed{2}$ and $p = \frac{x_p - x_0}{h} = \frac{3-2}{2} = \boxed{\frac{1}{2}}$.

- **Step 2 Construct Forward Differences Table**

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
2	34				
		708			
4	742		2392		
		3100		2880	
6	3842		5272		1152

		8372		4032	
8	12214		9304		
		17676			
10	29890				

- **Step 3 Substitute in the following relations**

- $$y'(x_p) = \frac{1}{h}\left[\Delta y_0 + \frac{2p-1}{2}\Delta^2 y_0 + \frac{3p^2-6p+2}{6}\Delta^3 y_0 + \frac{2p^3-9p^2+11p-3}{12}\Delta^4 y_0 + \cdots\right]$$

$$y'(3) = \frac{1}{2}\left[708 + \frac{2\left(\frac{1}{2}\right)-1}{2}(2392) + \frac{3\left(\frac{1}{2}\right)^2-6\left(\frac{1}{2}\right)+2}{6}(2880) + \frac{2\left(\frac{1}{2}\right)^3-9\left(\frac{1}{2}\right)^2+11\left(\frac{1}{2}\right)-3}{12}(1152)\right]$$

$$= 318$$

- $$y'(x_0) = \frac{1}{h}\left[\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + \frac{\Delta^5 y_0}{5}\cdots\right]$$

$$y'(2) = \frac{1}{2}\left[708 - \frac{2392}{2} + \frac{2880}{3} - \frac{1152}{4}\right] = 92$$

- $$y''(x_p) = \frac{1}{h^2}\left[\Delta^2 y_0 + (p-1)\Delta^3 y_0 + \frac{6p^2-18p+11}{12}\Delta^4 y_0 + \cdots\right]$$

$$y''(x_p) = \frac{1}{2^2}\left[2392 + \left(\frac{1}{2}-1\right)(2880) + \frac{6(\frac{1}{2})^2-18(\frac{1}{2})+11}{12}(1152)\right] = 322$$

- $$y''(x_0) = \frac{1}{h^2}\left[\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \frac{5}{6}\Delta^5 y_0\cdots\right]$$

$y''(2) \approx \frac{1}{2^2}\left[2392 - 2880 + \frac{11}{12}(1152)\right] = 142$. Try to verify ($y = 3x^4 - x^2 - 10$)

Exercises 5

1. Find $y'(7.5), y'(7), y''(7.5)$, and $y''(7)$.

x	7: 00	8: 00	9: 00	10: 00
y	20.5	22.1	23.4	25.0

2. Find $N'(1925), N'(1920), N''(1925)$, and $N''(1920)$.

t	1920	1930	1940	1950	1960
$N(t)$	5.5	5.8	6.4	7.2	7.9

3. Predict $y'(1.5), y'(1), y''(1.5)$, and $y''(2)$.

t	1	2	3	4	5	6
y	4.5	4.8	5.1	5.4	5.5	5.2

4. Estimate $\frac{dV}{dL}(4.5), \frac{dV}{dL}(4), \frac{d^2V}{dL^2}(4.5)$, and $\frac{d^2V}{dL^2}(4)$.

L	4	5	6	7
V	1.2	1.5	1.8	2.1

5. Construct an approximate polynomial passes through the following points and then estimate $y'(2.5), y'(2), y''(2.5)$, and $y''(2)$.

x	2	5	10
y	50	55.5	62.3

6. Approximate f', f'' of $f(x) = cos(x)$ at $x = \pi$ using a step size $h = 0.01$.

Chapter 6

The Numerical Integration

Numerical integration is a branch of numerical analysis that focuses on calculating the integral of a function when an exact analytic solution is difficult or impossible to obtain. It provides a means to approximate the area under a curve by summing up finite sections of the graph. Numerical integration is widely used in science, engineering, economics, and many other disciplines where complex mathematical models are involved.

Importance of Numerical Integration

1. **Real-World Applicability:** Many functions arising from real-world scenarios are either not integrable in closed form or are defined by discrete data points, such as experimental measurements or simulations.
2. **Complex Functions:** For functions involving intricate expressions (e.g., trigonometric, exponential, or transcendental terms), numerical methods provide an efficient solution.

3. **Multi-Dimensional Problems:** In higher dimensions, numerical integration helps compute volumes, probabilities, and other quantities in complex domains.
4. **Approximation of Models:** It allows the approximation of models in physics, engineering, and biology where explicit solutions are not feasible.

Applications of Numerical Integration

1. **Physics and Engineering:** Computing work done by a force, evaluating energy states, or solving boundary value problems.
2. **Economics:** Estimating area under demand or supply curves for total revenue or cost analysis.
3. **Biostatistics:** Calculating survival probabilities or analyzing stochastic processes.
4. **Computer Graphics:** Rendering 3D images through ray tracing and shading models often uses numerical integration.
5. **Machine Learning:** Numerical integration is crucial in training probabilistic models, such as calculating expectations under a probability distribution.

Types of Numerical Integration

1. **Single-Variable Integration:** Deals with functions of one variable, e.g., finding the area under a curve on a 2D plane.
2. **Multiple Integration:** Extends the process to functions of two or more variables, such as finding volumes under surfaces.
3. **Definite Integration:** Focuses on finding the value of the integral over a specific interval $[a, b]$.
4. **Indefinite Integration:** Although less common in numerical contexts, it involves approximating antiderivatives.

Common Methods of Numerical Integration

1. **Simple Techniques:**
    - **Rectangular Rule**: Approximates the area using rectangles. Although straightforward, its accuracy depends on the function's smoothness.
    - Trapezoidal Rule: Uses trapezoids for approximation, offering better accuracy than the rectangular rule by considering linear interpolation.

Derivation

$$\int_{x_0}^{x_n} f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx$$

$$I_{trapz} = I_1 + I_2 + \cdots I_n$$

Here, we will compute I_1 and by similar way we can compute $I_2, I_3, \ldots, I_n$,

Now, we can interpolate the integrable function $f(x)$ over equidistance two points $[x_0, x_1]$.

x	f	Δf
x_0	f_0	
		Δf_0
x_1	f_1	

$f(x) \approx P_1(x) = f_0 + s\Delta f_0$,

where $s = \frac{x-x_0}{h}$, and $h = x_1 - x_0$.

By the substitution $s = \frac{x-x_0}{h} \Rightarrow x = x_0 + sh \Rightarrow dx = hds$,

$\Rightarrow x = x_0 \rightarrow s = 0$ and $x = x_1 \rightarrow s = 1$

$$\therefore I_1 = h\int_0^1 (f_0 + s\Delta f_0)ds$$

$$\therefore I_1 = h\left(f_0 s + \frac{s^2}{2}\,\Delta f_0\right)_0^1 = h\left(f_0 + \frac{1}{2}\,\Delta f_0\right)$$

$$= h\left(f_0 + \frac{1}{2}\,(f_1 - f_0)\right)$$

$$\therefore I_1 = \frac{h}{2}(f_0 + f_1).$$

Similarly, $\therefore I_2 = \frac{h}{2}(f_1 + f_2).$

$$\therefore I_3 = \frac{h}{2}(f_2 + f_3).$$

.

.

.

$$I_{n-1} = \frac{h}{2}(f_{n-2} + f_{n-1}).$$

$$I_{n-1} = \frac{h}{2}(f_{n-1} + f_n).$$

Then

$$I_{Trapz} = \frac{h}{2}(f_0 + 2(f_1 + f_2 + \cdots + f_{n-1}) + f_n)$$

- ***A useful Algorithm***

Problem as:

Find $\int_{x_0}^{x_n} f(x)dx$, *with* $\boxed{n}$ equidistant points; or step size $\boxed{h}$.

Solution Algorithm

- **Step 1 Calculate** $h = \frac{x_n - x_0}{n}$ or $n = \frac{x_n - x_0}{h}$ with $x_i = x_0 + ih$
- **Step 2 Calculate** $f(x)$ **at the point** $x_i, i = 0,1,\cdots n$ **in Tabled form**

x	x_0	x_1	x_2	$\cdots$	x_{n-2}	x_{n-1}	x_n
$f(x)$	f_0	f_1	f_2	$\cdots$	f_{n-2}	f_{n-1}	f_n

- **Step 3 Substitute in the following relations**
- $I_{Trapz} \approx \frac{h}{2}\left[f_0 + 2\left(f_1 + f_2 + \cdots f_{n-1}\right) + f_n\right]$,

Example 6.1

Compute an approximate value of $\int_0^1 \sqrt{e^x + 1}\, dx$, Using Trapezoidal formula via $h = 0.1$.

Solution:

$n = \frac{x_n - x_0}{h} = \frac{1-0}{0.1} = 10$, and $x_{i+1} = x_0 + ih$.

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1

$f(x)$	1.4142	1.4509	1.4904	1.5329	1.5785	1.6274	1.6799	1.736	1.7959	1.86	1.9282

$$I_{Trapz} \approx \frac{h}{2}\left[f_0 + 2\left(f_1 + f_2 + \cdots f_9\right) + f_{10}\right],$$

$$\therefore I_{Trapz} \approx \frac{0.1}{2}\left[1.4142 + 2\left(1.4509 + 1.4904 + \cdots 1.86\right) + 1.9282\right] = 1.64236$$

2. **Higher-Order Polynomial Methods:**
   - Simpson's $\frac{1}{3}$ Rule: Uses parabolic arcs to approximate the function, providing greater accuracy for smooth functions.

Derivation

$$\int_{x_0}^{x_n} f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)dx$$

$$I_{\frac{1}{3}simpison} = I_2 + I_4 + \cdots I_n$$

Here, we will compute I_1 and by similar way we can compute $I_2, I_3, \ldots, I_n$,

Now, we can interpolate the integrable function $f(x)$ over equidistance three points $\{x_0, x_1, x_2\}$.

x	f	Δf	$\Delta^2 f$

x_0	f_0		
		Δf_0	
x_1	f_1		$\Delta^2 f_0$
		Δf_1	
x_2	f_2		

$f(x) \approx P_2(x) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0,$

where $s = \frac{x-x_0}{h}$, and $h = x_{i+1} - x_i$.

By the substitution $s = \frac{x-x_0}{h} \Rightarrow x = x_0 + sh \Rightarrow dx = hds,$

$\Rightarrow x = x_0 \rightarrow s = 0$ and $x = x_2 \rightarrow s = 2$

$$\therefore I_2 = h\int_0^2 \left(f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0\right) ds$$

$$\therefore I_2 = h\left(f_0 s + \frac{s^2}{2}\,\Delta f_0 + \frac{\frac{s^3}{3} - \frac{s^2}{2}}{2}\Delta^2 f_0\right)_0^2$$

$$= h\left(2f_0 + 2\,\Delta f_0 + \frac{1}{3}\Delta^2 f_0\right)$$

$$\therefore I_2 = \frac{h}{3}(f_0 + 4f_1 + f_2).$$

Similarly, $\therefore I_4 = \frac{h}{3}(f_2 + 4f_3 + f_4).$

$$\therefore I_6 = \frac{h}{3}(f_4 + 4f_5 + f_6).$$

.

.

.

$$I_{n-2} = \frac{h}{3}(f_{n-4} + 4f_{n-3} + f_{n-2}).$$

$$I_n = \frac{h}{3}(f_{n-2} + 4f_{n-1} + f_n).$$

Then

$$I_{\frac{1}{3}Simpison} = \frac{h}{3}(f_0 + 4(f_1 + f_3 + \cdots + f_{n-1}) + 2(f_2 + f_4 + \cdots + f_{n-2}) + f_n)$$

- ***A useful Algorithm***

Problem as:

Find $\int_{x_0}^{x_n} f(x)dx$, *with* $\boxed{n}$ equidistant points; or step size $\boxed{h}$.

Solution Algorithm

- **Step 1 Calculate** $h = \frac{x_n - x_0}{n}$ or $n = \frac{x_n - x_0}{h}$ with $x_i = x_0 + ih$
- **Step 2 Calculate** $f(x)$ **at the point** $x_i, i = 0,1,\cdots n$ **in Tabled form**

x	x_0	x_1	x_2	$\cdots$	x_{n-2}	x_{n-1}	x_n
$f(x)$	f_0	f_1	f_2	$\cdots$	f_{n-2}	f_{n-1}	f_n

- **Step 3 Substitute in the following relations**
- $I_{\frac{1}{3}Simpison} \approx \frac{h}{3}\left[f_0+4\left(f_1+f_3+\cdots f_{n-1}\right)+2\left(f_2+f_4+\cdots f_{n-2}\right)+f_n\right],$

Example 6.2

Compute an approximate value of $\int_2^3 \frac{x+2}{x^2-2x+2}\,dx$, Using $\frac{1}{3}$ Simpson's' rule via $h = 0.1$ and evaluate the absolute error.

Solution:

$n = \frac{x_n - x_0}{h} = \frac{3-2}{0.1} = 10$, and $x_{i+1} = x_0 + ih.$

x	2	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3
$f(x)$	2	1.8552	1.7213	1.5985	1.4864	1.3846	1.2921	1.2082	1.1320	1.0629	1

$$I_{\frac{1}{3}Simpison} \approx \frac{h}{3}\left[f_0+4\left(f_1+f_3+f_5+f_7+f_9\right)+2\left(f_2+f_4+f_6+f_8\right)+f_{10}\right],$$

$$I_{\frac{1}{3}Simpison} \approx \frac{0.1}{3}\Big[2+4\left(1.8552+1.5985+1.3846+1.2082+1.0629\right)= \\ +2\left(1.7213+1.4864+1.2921+1.1320\right)+1\Big]=1.4234$$

While the exact value can be obtained as follows:

$$I_{\text{exact}} = \int_2^3 \frac{x+2}{x^2-2x+2}dx = \int_2^3 \frac{\frac{1}{2}(2x-2)+3}{x^2-2x+2}dx$$
$$= \frac{1}{2}\int_2^3 \frac{2x-2}{x^2-2x+2}dx + 3\int_2^3 \frac{1}{(x-1)^2+1}dx$$
$$= \frac{1}{2}\ln\left(x^2-2x+2\right)_2^3 + 3\tan^{-1}(x-1)_2^3 = 1.42339$$

The absolute error

$$er = \left| I_{\text{exact}} - I_{\frac{1}{3}Simpison} \right| = \left|1.423397029127 - 1.4233958791728\right| = 0.0000011499541$$

- **Simpson's $\frac{3}{8}$ Rule:** Uses parabolic arcs to approximate the function, providing greater accuracy for smooth functions.

Derivation

$$\int_{x_0}^{x_n} f(x)dx = \int_{x_0}^{x_3} f(x)dx + \int_{x_3}^{x_6} f(x)dx + \cdots + \int_{x_{n-3}}^{x_n} f(x)dx$$

$$I_{\frac{3}{8}simpison} = I_3 + I_6 + \cdots I_n$$

Here, we will compute I_3 and by similar way we can compute $I_6, I_9, \ldots, I_n$,

Now, we can interpolate the integrable function $f(x)$ over equidistance four points $\{x_0, x_1, x_2, x_3\}$.

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$
x_0	f_0			
		Δf_0		
x_1	f_1		$\Delta^2 f_0$	
		Δf_1		$\Delta^3 f_0$
x_2	f_2		$\Delta^2 f_1$	
		Δf_2		
x_3	f_3			

$$f(x) \approx P_3(x) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0,$$

where $s = \frac{x-x_0}{h}$, and $h = x_1 - x_0$.

By the substitution $s = \frac{x-x_0}{h} \Rightarrow x = x_0 + sh \Rightarrow dx = hds$,

$\Rightarrow x = x_0 \rightarrow s = 0$ and $x = x_3 \rightarrow s = 3$

$$\therefore I_3 = h\int_0^3 \left(f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0\right) ds$$

$$\therefore I_3 = h\left(f_0 s + \frac{s^2}{2}\,\Delta f_0 + \frac{\frac{s^3}{3} - \frac{s^2}{2}}{2}\Delta^2 f_0 + \frac{\frac{s^4}{4} - s^3 + s^2}{6}\Delta^3 f_0 \right)_0^3$$

$$= h\left(3f_0 + \frac{9}{2}\,\Delta f_0 + \frac{9}{4}\Delta^2 f_0 + \frac{3}{8}\Delta^3 f_0\right)$$

$$\therefore I_3 = \tfrac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3).$$

Similarly, $\therefore I_6 = \tfrac{3h}{8}(f_3 + 3f_4 + 3f_5 + f_6).$

$$\therefore I_9 = \tfrac{3h}{8}(f_6 + 3f_7 + 3f_8 + f_9).$$

.

.

.

$$I_{n-3} = \tfrac{3h}{8}(f_{n-6} + 3f_{n-5} + 3f_{n-4} + f_{n-3}).$$

$$I_n = \tfrac{3h}{8}(f_{n-3} + 3f_{n-2} + 3f_{n-1} + f_n).$$

Then

$$I_{\frac{3}{8}Simpison} = \frac{3h}{8}(f_0 + 3(f_1 + f_2 + f_4 + f_5 \ldots + f_{n-2} + f_{n-1}) + 2(f_3 + f_6 + \cdots + f_{n-3}) + f_n)$$

- ***A useful Algorithm***

Problem as:

Find $\int_{x_0}^{x_n} f(x)dx$, *with* $\boxed{n}$ equidistant points; or step size $\boxed{h}$.

Solution Algorithm

- **Step 1 Calculate** $h = \frac{x_n - x_0}{n}$ or $n = \frac{x_n - x_0}{h}$ with $x_i = x_0 + ih$
- **Step 2 Calculate** $f(x)$ **at the point** $x_i, i = 0,1,\cdots n$ **in Tabled form**

x	x_0	x_1	x_2	$\cdots$	x_{n-2}	x_{n-1}	x_n
$f(x)$	f_0	f_1	f_2	$\cdots$	f_{n-2}	f_{n-1}	f_n

- **Step 3 Substitute in the following relations**

$$I_{\frac{3}{8}Simpison} = \frac{3h}{8}(f_0 + 3(f_1 + f_2 + f_4 + f_5 \ldots + f_{n-2} + f_{n-1}) + 2(f_3 + f_6 + \cdots + f_{n-3}) + f_n)$$

Example 6.2

Compute an approximate value of $\int_0^1 \frac{1}{x^2+1} dx$, Using $\frac{3}{8}$ Simpson's' rule via $h = 0.1$ and use the result to find an approximate value to π.

Solution:

$n = \frac{x_n - x_0}{h} = \frac{1-0}{0.1} = 10$, and $x_{i+1} = x_0 + ih$.

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$f(x)$	1	0.99	0.9615	0.9174	0.862	0.8	0.7356	0.6711	0.6097	0.5524	0.5

$$I_{\frac{3}{8}Simpison} = \frac{3h}{8}(f_0 + 3(f_1 + f_2 + f_4 + f_5 + f_7 + f_8) + 2(f_3 + f_6 + f_9) + f_{10})$$

$$I_{\frac{3}{8}Simpison} = \frac{3(0.1)}{8}(1 + 3(0.99 + 0.9615 + 0.862 + 0.8 + 0.6711 + 0.6097) + 2(0.9174 + 0.7356 + 0.5524) + 0.5) =$$

While the exact value can be obtained as follows:

$$I_{\text{exact}} = \int_0^1 \frac{1}{x^2+1} dx = \tan^{-1} x \Big]_0^1 = \tan^{-1} 1 - \tan^{-1} 0 = \frac{\pi}{4}.$$

Then suppose

$$I_{\text{exact}} \approx I_{\frac{3}{8}Simpison}$$

$$\frac{\pi}{4} \approx 0.772283753199$$

$$\pi \approx 4\left(0.772283753199\right) = 3.08913501279$$

- **Newton-Cotes Formulas:** Extend the trapezoidal and Simpson's rules to higher degrees of polynomial interpolation.

3. **Adaptive Methods:**
   - **Adaptive Quadrature:** Dynamically refines the intervals to achieve a desired accuracy, focusing computational effort where the function changes rapidly.
4. **Gaussian Quadrature:**
   - Involves a weighted sum of function values at specific points (roots of orthogonal polynomials). It is highly efficient for smooth functions over fixed intervals.
5. **Monte Carlo Integration:**
   - Uses random sampling to approximate the integral. It is particularly effective for high-dimensional integrals and probabilistic models.

Error Analysis in Numerical Integration

Accuracy in numerical integration depends on:

1. **Step Size (h):** Smaller step sizes reduce the error but increase computational cost.
2. **Method Used:** Higher-order methods generally reduce errors but may require more sophisticated calculations.
3. **Nature of the Function:** Smooth, well-behaved functions result in lower errors, while highly oscillatory or discontinuous functions increase difficulty.

Choosing the Right Method

- **For Simple Functions:** Trapezoidal or Simpson's rules suffice for basic applications.
- **For High Accuracy:** Gaussian quadrature or adaptive methods are preferred.
- **For High-Dimensional Problems:** Monte Carlo methods are typically the go-to approach

Exercises 6

1. The power consumption of a device follows the function $f(t) = 5 + 0.5t^2$ (in watts) over time t (in hours) from $t = 0$ to $t = 8$. Use the trapezoidal rule with $n = 4$ to estimate the total energy consumed in kilowatt-hours.

2. The velocity of water in a river varies as $f(x) = 4x - 0.2x^2$ (in m/s), where x is the distance (in meters) from one bank, measured from $x = 0$ to $x = 20$. Use $n = 5$ to estimate the average velocity across the river.

3. The temperature gradient in a rod is given by $f(x) = 100e^{-0.1x}$ (in °C) along its length from $x = 0$ to $x = 10$ (in cm). Use the trapezoidal rule with $h = 2$ to approximate the total temperature difference.

4. The height of an arch is given by $f(x) = 25 - x^2$, where x is measured from -5 to 5 (in meters). Use the trapezoidal rule with $n = 10$ to estimate the cross-sectional area.

5. The growth rate of a population is modeled by $f(t) = 200 + 50sin\,(t)$ (in people/year) over $t = 0$ to $t = 6$ years. Use Simpson's 1/3 rule with $n = 6$ to estimate the total population increase over the period.

6. The velocity of a car is given by $f(t) = 10t - t^2 + 2$ (in m/s) from $t = 0$ to $t = 8$ seconds. Use $n = 4$ to calculate the distance traveled using Simpson's 1/3 rule.

7. The rate of heat dissipation in a system is modeled as $f(x) = x3 - 2x2 + x + 5$, where x is the time in seconds, from $x = 0$ to $x = 4$. Use $h = 0.5$ and Simpson's 1/3 rule to approximate the total heat dissipation.

8. The rate of fuel consumption is given by $f(x) = 3x^x - 2x + 1$ (in liters/hour) over $x = 0.1$ to $x = 1$ hours. Use Simpson's 3/8 rule with $h = 0.1$ to estimate the total fuel consumed.

9. The rate of liquid flow into a tank is modeled as $f(t) = 5 + 3cos\,(t)$, where t is in seconds, over $t = 0$ to $t = 9$. Use Simpson's 3/8 rule with $n = 3$ to approximate the total volume of liquid.

10. The height of a projectile at time t is given by $f(t) = -4.9t2 + 20t + 50$ (in meters), from $t = 0$ to $t = 5$ seconds. Use $n = 3$ to calculate the total area under the height-time graph using Simpson's 3/8 rule.

Chapter 7

Solving 1st order Differential Equations

First-order differential equations are equations involving the derivative of a function and the function itself. They are fundamental in mathematics, physics, engineering, biology, and many other fields. The general form of a first-order differential equation is:

$$F(x, y, \frac{dy}{dx}) = 0$$

or, in explicit form:

$$\frac{dy}{dx} = f(x, y),$$

where $y = y(x)$ is the unknown function to be solved, x is the independent variable, and $f(x, y)$ is a given function.

Importance of First-Order Differential Equations

First-order differential equations play a critical role in modeling real-world phenomena and are often the starting point for understanding more complex systems. Their importance lies in their ability to describe:

- **Natural Processes:** They model physical, chemical, and biological systems, such as radioactive decay, population growth, and heat transfer.
- **Engineering Systems:** Electrical circuits, mechanical systems, and fluid flow often involve first-order dynamics.
- **Economics and Social Sciences:** They are used in modeling economic growth, resource consumption, and market trends.

These equations are foundational because they allow a systematic approach to problem-solving in a wide range of disciplines.

Applications of First-Order Differential Equations

1. **Physics:**
    - Motion under constant acceleration.
    - Exponential decay in radioactive materials.
    - RC (resistor-capacitor) circuits.
2. **Biology:**
    - Population dynamics (e.g., logistic growth models).
    - Spread of diseases (basic SIR models).
3. **Engineering:**

- Thermodynamics (heat transfer in simple systems).
- Control systems (PID controllers often involve first-order approximations).

4. **Economics:**
    - Modeling interest rates and investment growth.
    - Analysis of supply and demand dynamics.

Types of Exact and Analytical Solutions

1. **Separable Equations:**
2. **Linear First-Order Equations:**
3. **Exact Equations:**
4. **Homogeneous Equations:**
5. **Special Substitutions:** Techniques like Bernoulli's and Riccati equations involve transforming the equation into a simpler form.

Numerical Solutions of First-Order Differential Equations

While analytical solutions provide exact answers, many real-world problems lead to equations that cannot be solved explicitly. Numerical methods are essential for approximating solutions to

such equations, especially in academic and research settings. Common numerical techniques include:

Euler's Method:

Euler's Method is one of the simplest and most intuitive numerical methods for solving first-order ordinary differential equations (ODEs) of the form:

$$y' = f(x, y) \qquad y(x_0) = y_0.$$

It provides an approximate solution to the ODE by stepping forward in small increments using the slope of the function.

Derivation of Euler's Method

The derivation of Euler's method is based on the concept of the Taylor series expansion. Given the first-order ODE $y' = f(x, y)$, the Taylor series expansion for $y(x)$ around x_n is:

$$y(x_{n+1}) = y(x_n) + hf(x_n, y_n) + \frac{h^2}{2!}y''(x_n) + \cdots$$

Euler's method truncates the series after the first-order term:

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y_n),$$

where:

- x_n is the current point.
- y_n is the current approximation of $y(x_n)$.
- h is the step size.

Thus, the Euler's formula for advancing the solution is:

$$y_{n+1} = y_n + hf(x_n, y_n).$$

Steps of Euler's Method

1. **Initialize:** Start with the initial condition (x_0, y_0).
2. **Iterate:**
    - Compute the next value y_{n+1} using: $y_{n+1} = y_n + hf(x_n, y_n)$.
    - Update $x_{n+1} = x_n + h$
3. **Repeat:** Continue iterating until the desired range of xxx is covered.

Example 7.1

Solve the first-order differential equation:

$y' = x + y, \qquad y(0) = 1,$

using Euler's method with step size $h = 0.1$ over the interval $x \in [0,0.5]$.

Step-by-Step Solution

1. **Initial Condition:** $x_0 = 0, y_0 = 1.$
2. **Iterate:**

- For $n = 0$: $y_1 = y_0 + hf(x_0, y_0)$, $y_1 = 1 + 0.1(0 + 1) = 1.1$
- For $n = 1$: $y_2 = y_1 + hf(x_1, y_1)$, $y_2 = 1.1 + 0.1(0.1 + 1.1) = 1.22$
- For $n = 2$: $y_3 = y_2 + hf(x_2, y_2)$, $y_3 = 1.22 + 0.1(0.2 + 1.22) = 1.362$
- For $n = 3$: $y_4 = y_3 + hf(x_3, y_3)$, $y_4 = 1.362 + 0.1(0.3 + 1.362) = 1.5282$
- For $n = 4$: $y_5 = y_4 + hf(x_4, y_4)$, $y_5 = 1.5282 + 0.1(0.4 + 1.5282) = 1.721$.

Approximate Solution:

At $x = 0.5$, $y \approx 1.721$.

The Exact Solution (General) of this problem is $y = Ce^x - x - 1$.

Using the initial condition $y(0) = 1 \Longrightarrow C = 2$.

Thus, the exact solution is $y = 2e^x - x - 1$.

Exact Value at $x = 0.5 \Longrightarrow y(0.5) = 2e^{0.5} - 0.5 - 1 = 1.797$

Absolute Error

The absolute error is the difference between the exact and approximate solutions:

$Error = | y_{exact} - y_{Euler} | = | 1.797 - 1.721 | = 0.076.$

Advantages and Limitations of Euler's Method

Advantages:

- Simple to implement.
- Intuitive and provides a basic understanding of numerical approximation.

Limitations:

- **Low Accuracy:** Truncates the Taylor series after the first term.
- **Stability Issues:** For stiff equations, the method can diverge.
- **Error Accumulation:** Errors grow linearly with the number of steps.

Euler's method is an excellent introductory numerical technique for solving first-order ODEs. While it is not the most accurate or robust method, it lays the foundation for understanding more advanced techniques, such as Runge-Kutta methods, which offer better accuracy and stability.

1. **Improved Euler (Heun's) Method:**
    - A refinement of Euler's method, improving accuracy by averaging slopes.

Runge-Kutta of order 2

The Runge-Kutta methods are powerful techniques for solving ordinary differential equations (ODEs). The second-order Runge-Kutta method (RK2) is a significant improvement over Euler's method, achieving higher accuracy while still being computationally straightforward. It is a two-stage method that estimates the slope more accurately by using an intermediate step.

We consider a first-order ODE:

$$y' = f(x, y), \qquad y(x_0) = y_0.$$

Derivation of the RK2 Method

RK2 improves upon Euler's method by using the weighted average of slopes:

1. **Predict the slope at the start of the interval** using Euler's method (k_1).
2. **Estimate the slope at a midpoint** or other intermediate point within the interval (k_2).

The general formula for RK2 is:

$$y_{n+1} = y_n + h\phi,$$

where:

$$\phi = b_1 k_1 + b_2 k_2,$$

with:

- $k_1 = f(x_n, y_n)$, the slope at the beginning of the interval.
- $k_2 = f(x_n + a_2\, h, y_n + h\,(a_{21}\, k_1))$, the slope at an intermediate point.

By choosing appropriate weights a_2, a_{21} , b_1, and b_2, the RK2 method can achieve second-order accuracy. A common choice is the **Heun's Method** (or trapezoidal rule):

- $a_2 = 1, a_{21} = 1, b_1 = b_2 = \frac{1}{2} a_2 = 1,\ a_{21} = 1, b_1 = b_2 = \frac{1}{2} a_2 = 1, a_{21} = 1, b_1 = b_2 = 21.$

This leads to the RK2 formula:

$$y_{n+1} = y_n + \frac{h}{2}[k_1 + k_2],$$

where:

$$k_1 = f(x_n, y_n), \qquad k_2 = f(x_n + h, y_n + hk_1).$$

The error of this formula is of order $O(h^2)$

Steps of RK2 Method

1. **Initialize:** Start with the initial condition (x_0, y_0).
2. **Compute k_1:**
    - $k_1 = f(x_n, y_n)$, the slope at the beginning of the interval.

3. **Compute k_2:**
    - $k_2 = f(x_n + h, y_n + hk_1)$, the slope at the end of the interval (using the predicted value from k_1.
4. **Update y:**
    - Compute the next value y_{n+1} using: $y_{n+1} = y_n + \frac{h}{2}[k_1 + k_2]$.
5. **Repeat:** Iterate the process for the desired range of x.

Example 7.2

Solve the differential equation:

$$y' = x + y, \qquad y(0) = 1,$$

using RK2 with step size $h = 0.1$ over the interval $x \in [0,0.3]$.

Solution:

1. **Initial Condition:**

$x_0 = 0, y_0 = 1.$

2. **First Step ($x = 0 \to 0.1$):**
    - Compute $k_1 = f(x_0, y_0) = 0 + 1 = 1.$
    - Compute $k_2 = f(x_0 + h, y_0 + hk_1) = f(0.1,1 + (0.1)(1)) = f(0.1,1.1) = 0.1 + 1.1 = 1.2.$

- Update: $y_1 = y_0 + \frac{h}{2}(k_1 + k_2) = 1 + \frac{0.1}{2}(1 + 1.2) = 1 + 0.11 = 1.11.$

3. **Second Step ($x = 0.1 \rightarrow 0.2$):**
   - Compute $k_1 = f(x_1, y_1) = 0.1 + 1.11 = 1.21.$
   - Compute $k_2 = f(x_1 + h, y_1 + hk_1) = f(0.2,1.11 + (0.1)(1.21)) = f(0.2,1.231) = 0.2 + 1.231 = 1.431.$
   - Update: $y_2 = y_1 + \frac{h}{2}(k_1 + k_2) = 1.11 + \frac{0.1}{2}(1.21 + 1.431) = 1.11 + 0.13205 = 1.24205.$
4. **Third Step ($x = 0.2 \rightarrow 0.3$):**
   - Compute $k_1 = f(x_2, y_2) = 0.2 + 1.24205 = 1.44205.$
   - Compute $k_2 = f(x_2 + h, y_2 + hk_1) = f(0.3,1.24205 + (0.1)(1.44205)) = f(0.3,1.386255) = 0.3 + 1.386255 = 1.686255.$
   - Update: $y_3 = y_2 + \frac{h}{2}(k_1 + k_2) = 1.24205 + 0.12(1.44205 + 1.686255) = 1.24205 + 0.156415 = 1.398465.$

With the exact solution $y = 2e^x - x - 1.$

$\therefore y(0.3) = 2e^{0.3} - 0.3 - 1 \approx 2(1.34986) - 0.3 - 1 = 2.69972 - 1.3 = 1.39972.$

$Error = |\ 1.39972 - 1.398465\ | = 0.001255.$

The RK2 method is a significant improvement over Euler's method, providing better accuracy with minimal extra computation. It is a practical choice for many problems where higher-order methods like RK4 might be unnecessary. However, its accuracy still depends on the step size, and for very stiff equations, other methods like implicit Runge-Kutta or adaptive methods might be required.

Runge-Kutta of order 4 Methods:

- Widely used due to their balance between computational cost and accuracy, particularly the fourth-order Runge-Kutta method:

  $y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$, where k_1, k_2, k_3, k_4 are intermediate slope evaluations, can calculated from:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$
$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$
$$k_4 = hf(x_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4], \qquad n = 0,1,2,\cdots$$

The error of this formula is of order $O(h^4)$

Example 7.3

Solve the differential equation:

$$y' = \frac{2x^2 - xy + y^2}{x^2} \quad , \qquad y(1) = 1,$$

using RK4 with step size $h = 0.1$ over the interval $x \in [1,1.2]$.

Solution:

Step 1: When $\boldsymbol{n = 0}$

$$k_1 = hf(x_0, y_0) = 0.1\left(\frac{2 - 1 + 1}{1}\right) = 0.2$$

$$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) = 0.205$$

$$k_3 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) = 0.2052$$

$$k_4 = hf(x_0 + h, y_0 + k_3) = 0.2105$$

$$\therefore y_1 = y_0 + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$\therefore y_1 = 1 + \frac{1}{6}[0.2 + 2(0.205) + 2(0.2052) + 0.2105] = 1.2052$$

Step 2: When $\boldsymbol{n = 1}$

$$k_1 = hf(x_1, y_1) = 0.2105$$

$$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) = 0.2159$$

$$k_3 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) = 0.2162$$

$$k_4 = hf(x_0 + h, y_0 + k_3) = 0.2218$$

$$\therefore y_2 = y_1 + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$\therefore y_2 = 1.2052 + \frac{1}{6}[0.2105 + 2(0.2159) + 2(0.2162) + 0.2218]$$
$$= 1.4212$$

$$\therefore y(1.2) = 1.4212$$

The exact solution of this problem is $y = x(1 + \tan(\ln(x)))$

Thus $y_{exact}(1.2) = 1.4212$

The absolute error $er = |y_{exact}(1.2) - y(1.2)| = 2.1569 \times 10^{-7}$

Multistep Methods:

The previously studied methods are referred to as single-step methods, which require knowledge of the solution at a single point to obtain the solution at the subsequent point. In contrast, multi-step methods necessitate the knowledge of the solution at multiple points to determine the desired solution. Such methods are derived through numerical integration of the differential equation. Adams, Milnes, Bashforth and Adams-Moulton

methods use information from multiple previous points to improve accuracy.

Adams Method

Adams's numerical method you're referring to is a **third-order Adams-Bashforth method**, used for solving ordinary differential equations (ODEs). This method is an explicit, multi-step method that estimates the value of the solution at the next step based on previous values. Specifically, the formula you're referring to is:

$$y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}), \qquad n = 2,3,4, \ldots$$

where $f_i = f(x_i, y_i)$ is the evaluation of the derivative at step n, y_n is the current approximation of the solution at step n, h is the step size.

- The indices $n, n-1, n-2$ refer to previous steps in the solution process.
- This method is used for solving the ODE $y' = f(x, y)$ over a given interval.

How It Works:

1. **Initial Values**: To use the third-order Adams-Bashforth method, you need values for y_0, y_1, y_2 (typically obtained via an single-step methods like Euler's method or Runge Kutta family or a lower-order Adams method). This is

because the method uses the current value and the two preceding values of f.

2. **Iterative Process**: Once you have the initial values and derivatives, you apply the formula iteratively to find the solution at subsequent steps. Each step uses the values from the previous three steps to compute the new value of y_{n+1}.
3. **Order of Accuracy**: The third-order method provides a more accurate estimate than simpler methods like Euler's method or the second-order Adams-Bashforth method, with an error term proportional to h^4.

Example of Step Computation:

For $n = 2$, the formula would be used to compute y_3 using the values of f_2, f_1, f_0.

$$y_3 = y_2 + \frac{h}{12}(23f_2 - 16f_1 + 5f_0)$$

This method continues iteratively as you move forward in time.

Pros and Cons:

- **Pros**: It's relatively accurate for solving ODEs, especially when high accuracy is needed and the step size h is not too large.
- **Cons**: It requires knowledge of previous function values (thus, a "multi-step" method), and it can be unstable or

inaccurate if the step size is not chosen appropriately or if the function has discontinuities.

$$y_{n+1} = y_n + \frac{h}{12}[23f_n - 16f_{n-1} + 5f_{n-2}], \qquad n = 2,3,4\cdots$$

Example 7.4

Find $y(0.03)$ for the ordinary differential equation:

$$y' = e^{25x-y}, \quad y(0) = 0, \qquad y(0.01) = 0.01129698,$$
$$y(0.02) = 0.02561793$$

using Adams method.

Solution:

We apply the iteration

$$y_{n+1} = y_n + \frac{h}{12}[23f_n - 16f_{n-1} + 5f_{n-2}], \qquad n = 2,3,4\cdots$$

For n=2

$$y_3 = y_2 + \frac{h}{12}[23f_2 - 16f_1 + 5f_0]$$

$$y_3 = y_2 + \frac{h}{12}[23f(x_2, y_2) - 16f(x_1, y_1) + 5f(x_0, y_0)]$$

$$y(0.03) = 0.02561793 + \frac{0.1}{12}\left[23e^{25(0.02)-0.02561793} - 16e^{25(0.01)-0.01129698} + 5e^{25(0)-0}\right]$$

$$\therefore y(0.03) = 0.04365781$$

With the absolute error

$$er = |y_{exact} - y_{Adams}| = 0.0000528$$

Milnes Method

Milne's numerical method is a multi-step method used to solve first-order ordinary differential equations (ODEs) of the form:

$$y' = f(x, y), \qquad y(x0) = y0.$$

It belongs to the **predictor-corrector** family, which estimates a solution at a future point using a prediction step and then refines it with a correction step. The method requires several previous points to calculate the next point, which is why it is classified as a multi-step method.

Key Concepts:

1. Predictor Step: The predictor formula estimates y_{n+1} using past values of y and $f(x, y)$. In Milne's method, the predictor is based on Milne's explicit method:

$$y_{n+1} = y_{n-3} + \frac{4h}{3}(2f_{n-2} - f_{n-1} + 2f_n),$$

where:

- h is the step size,

- $f_k = f(x_k, y_k)$ for each point k,
- $y_{n-3}, y_{n-2}, y_{n-1}$, and y_n are previously computed values.

2. Corrector Step: The corrector refines y_{n+1} by using the predicted value y_{n+1} in a formula based on Milne's implicit method:

$$y_{n+1} = y_{(n-1)} + \frac{h}{3}(f_{n-1} + 4f_n + f_{n+1}),$$

where $f_{n+1} = f(x_{n+1}, y_{n+1})$. This step uses the predicted y_{n+1} to evaluate f_{n+1}.

Procedure:

1. Initialization:
   - To start, you need the values y_0, y_1, y_2, y_3 . These can be computed using a one-step method like Runge-Kutta or other techniques.
2. Iteration for $n \geq 3$:
   - Predict y_{n+1} using the predictor formula.
   - Use this predicted y_{n+1} to compute f_{n+1} and plug it into the corrector formula.
   - Correct y_{n+1} iteratively until convergence (i.e., the value stabilizes).

Key Features:

- Efficiency: Being a multi-step method, it reuses previously computed points, making it computationally efficient for long intervals.
- Accuracy: The corrector step improves accuracy by incorporating feedback from the predicted value.
- Stability: The method is conditionally stable, meaning it works well with appropriate step size h.

Practical Notes:

- The predictor step is explicit, making it computationally simpler but less accurate.
- The corrector step is implicit, often requiring iteration for convergence.
- Proper choice of the initial values and step size h is crucial for stability and accuracy.

Milne's method is widely used in numerical solutions of ODEs due to its balance between computational efficiency and accuracy, particularly when the function $f(x, y)$ is well-behaved.

We can conclude Milne's method in the following iterations:

$$\text{Predict}\quad y_{n+1} = y_{n-3} + \frac{4h}{3}[2f_{n-2} - f_{n-1} + 2f_n]$$

$$\text{Correct}\quad y_{n+1} = y_{n-1} + \frac{h}{3}[f_{n-1} + 4f_n + f_{n+1}], \qquad n = 3,4,5,\cdots$$

Example 7.5

Find $y(0.04)$ for the ordinary differential equation:

$$y' = e^{25x-y},\quad y(0) = 0, \qquad y(0.01) = 0.01129698,$$
$$y(0.02) = 0.02561793, \qquad y(0.03) = 0.043710684$$

using Milne's method.

Solution:

Put $\boldsymbol{n = 3}$

$$\text{Predict}\quad y_4 = y_0 + \frac{4h}{3}[2f_1 - f_2 + 2f_3]$$

$$\text{Predict}\quad y_4 = 0 + \frac{4(0.01)}{3}(2e^{25(0.01)-0.01129698} - e^{25(0.02)-0.02561793}2e^{25(0.03)-0.043710684})$$

$$\text{Predict}\quad y_4 = 0.046027632$$

$$\text{Correct}\quad y_4 = y_2 + \frac{h}{3}[f_2 + 4f_3 + f_4]$$

$$\text{Correct}\quad y_4 = 0.02561793 + \frac{0.01}{3}(e^{25(0.02)-0.02561793} + 4e^{25(0.03)-0.043710684} + e^{25(0.04)-0.046027632})$$

$$\therefore y(0.04) = y_4 = 0.06664744127$$

With the absolute error

$$er = |y_{exact} - y_{Milnes}| = 0.00000138$$

2. **Adaptive Methods:**
    - Techniques like adaptive Runge-Kutta adjust the step size dynamically based on error estimates, crucial for handling stiff or highly variable equations.

Academic and Research Perspective

In academic settings, the study of numerical solutions focuses on their stability, convergence, and error analysis. This involves:

1. **Stability Analysis:**
    - Ensuring that numerical methods produce bounded solutions for stable problems.
2. **Convergence:**
    - Demonstrating that the numerical solution approaches the exact solution as the step size tends to zero.
3. **Error Analysis:**
    - Quantifying global and local truncation errors.

4. **Computational Efficiency:**
    - Research in efficient algorithms that minimize computation while maintaining accuracy is an ongoing pursuit, especially for large-scale problems in computational science.

In research, numerical solutions are indispensable in fields such as fluid dynamics, climate modeling, and neural network training, where first-order differential equations often arise as governing models. Modern research often integrates machine learning with numerical methods to improve prediction and efficiency.

This structured understanding of first-order differential equations highlights their theoretical importance and practical versatility. They serve as a bridge between mathematical rigor and real-world application, making them a cornerstone of scientific inquiry.

Exercises 7

1. Population Growth

- Use Euler's method to find $y(2)$ with step size $h = 0.5$:

  $y' = 0.1y(1 - \frac{y}{500}), \qquad y(0) = 50.$

2. Newton's Law on Cooling

- Use Runge-Kutta 2 to $y(5)$ with h=0.1:

  $y' = -0.07(y - 25), \qquad y(0) = 90.$

3. RC Circuit

- Use Runge-Kutta 4 to find $y(1)$ with $h = 0.25$:

$y' = -y + 105x, \qquad y(0) = 0.$

4. Falling Object with Air Resistance

- Use Adams-Bashforth method to find $y(5)$ with $n = 10$:

  $y' = 9.8 - 0.2y^2, \qquad y(0) = 0.$

5. Predator-Prey Model

- Use Milnes method to find $y(10)$ with $h = 0.2$:
- $y' = 0.02y(1 - \frac{y}{500}) - 0.001y^2, \qquad y(0) = 100.$

6. Mixing Problem

- Use Euler's method to find $y(20)$ with $n = 40$:
- $y' = -0.1y + 2, \qquad y(0) = 50.$

7. Radioactive Decay

- Use Runge-Kutta 2 to find $y(1)$ with $h = 0.5$:

  $y' = x - 0.001y, \qquad y(0) = 100.$

8. Drug Concentration in Bloodstream

- Use Runge-Kutta 4 to find $y(5)$ with $h = 0.1$:

  $y' = 10x^2 - 0.3y^2, \qquad y(0) = 0.$

9. Logistic Growth

- Use Adams method to find $y(20)$ with n=40:

  $y' = 0.5y(1 - \frac{xy}{1000}), \qquad y(0) = 10.$

10. Circuit Inductance

- Use Milnes method to find $y(4)$ with h=0.2:

  $y' = 5 - x^3y^2, \qquad y(0) = 0.$

Chapter 8

Solving Non-Linear Equations

Nonlinear algebraic equations, often represented as $f(x) = 0$, appear in a wide range of scientific and engineering problems. Unlike linear equations, these equations do not generally have analytical solutions, and numerical methods are often employed to find approximate roots. Several techniques have been developed to address these challenges, each with specific strengths and limitations. Among the most common numerical methods are the **bisection method**, **false-position method**, **fixed-point iteration**, and **Newton-Raphson method**.

1. Bisection Method

The bisection method is a **bracketing approach** that relies on the intermediate value theorem. If $f(a)$ and $f(b)$ have opposite signs $(f(a) \cdot f(b) < 0)$, there exists at least one root x^* in the interval $[a, b]$. The method iteratively divides the interval into halves, selecting the subinterval where the sign change occurs.

The **bisection method** is a numerical technique for finding the root of a non-linear equation $f(x) = 0$ in a given interval $[a, b]$. This method is based on the intermediate value theorem, which

states that if $f(a)$ and $f(b)$ have opposite signs, there must be at least one root in $[a, b]$.

Algorithm of the Bisection Method

1. **Initial Interval**: Start with an interval $[a, b]$ such that $f(a) \cdot f(b) < 0$, ensuring that a root exists in the interval.
2. **Midpoint Calculation**: Compute the midpoint $c = \frac{a+b}{2}$.
3. **Check Sign**: Evaluate $f(c)$:
   - If $f(c) = 0$, c is the root.
   - If $f(a) \cdot f(c) < 0$, the root lies in $[a, c]$; set $b = c$.
   - Otherwise, the root lies in $[c, b]$; set $a = c$.
4. **Repeat**: Continue the process until the interval length $(b - a)$ is less than the desired accuracy ϵ.

Relation Between Number of Iterations and Accuracy

The number of iterations n required to achieve a given accuracy ϵ is given by:

$$n \geq \frac{log\left(\frac{b-a}{\epsilon}\right)}{log(2)}$$

where $[a, b]$ is the initial interval.

Example 8.1

Solve $x^3 - x - 1 = 0$ in the interval $[1,2]$ for accuracy $\epsilon = 10^{-3}$.

1. **Initial Values**:
   - $f(1) = 1^3 - 1 - 1 = -1$
   - $f(2) = 2^3 - 2 - 1 = 5$
   - Since $f(1) \cdot f(2) < 0$, a root exists in $[1,2]$.
2. **Number of Iterations**: Using the formula:

$$n \geq \frac{log\left(\frac{2-1}{10-3}\right)}{log(2)} = \frac{log(10^3)}{log(2)} = \frac{3}{log(2)} \approx 10$$

   Thus, about 10 iterations are needed.

3. **Perform Iterations**: Start with $[a, b] = [1,2]$ and $\epsilon = 10^{-3}$:

Iteration	a	b	$c = \frac{a+b}{2}$	$f(c)$	Interval Length
1	1	2	1.5	0.875	1
2	1	1.5	1.25	−0.328	0.5
3	1.25	1.5	1.375	0.228	0.25
4	1.25	1.375	1.3125	−0.062	0.125
5	1.3125	1.375	1.34375	0.080	0.0625
6	1.3125	1.34375	1.328125	0.009	0.03125

7	1.3125	1.328125	1.3203125	−0.027	0.015625
8	1.3203125	1.328125	1.32421875	−0.009	0.0078125
9	1.32421875	1.328125	1.32617175	0.00005	0.0039062

The root is approximately x=1.326 with accuracy $\epsilon = 10^{-3}$.

- **Advantages**: Guaranteed convergence if the function is continuous, and the interval contains a root.
- **Limitations**: Convergence is linear and can be slow, especially for functions with shallow gradients near the root.

2. False-Position Method (Regula Falsi)

The false-position method combines aspects of the bisection method and linear interpolation. Instead of halving the interval, a secant line is drawn between the points $(a, f(a))$ and $(b, f(b))$, and the root of the secant line serves as the next approximation. The interval is then updated based on the sign change.

The False Position Method (or Regula Falsi) is a numerical approach to solve nonlinear algebraic equations of the form $f(x) = 0$. It combines the ideas of the bisection method and linear interpolation to approximate the root.

Steps of the Method

1. Choose an interval $[a, b]$: Select two initial guesses a and b such that $f(a)f(b) < 0$ (indicating a root exists in $[a, b]$).
2. Linear interpolation to find c: Compute the root of the straight line connecting $(a, f(a))$ and $(b, f(b))$:

$$c = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

Update the interval:

   - Evaluate $f(c)$. If $f(a)f(c) < 0$, the root lies in $[a, c]$; set $b = c$.
   - If $f(b)f(c) < 0$, the root lies in $[c, b]$; set $a = c$.
3. Check for convergence: The process continues until the relative error or $| f(c) |$ is less than the desired accuracy (10^{-4} in this case).

Derivation

The formula for c comes from the equation of a line passing through two points:

Line equation: $y = f(a) + \frac{f(b)-f(a)}{b-a}(x - a)$.

Setting $y = 0$ (to find the x-intercept), we solve (Lost figure here)

$$0 = f(a) + \frac{f(b) - f(a)}{b - a}(c - a)$$

which simplifies to:

$$c = \frac{af(b) - bf(a)}{f(b) - f(a)}.$$

Example 8.2

Solve $10xe^x - 7 = 0$ in the interval $[0,1]$ with an accuracy of 10^{-4}.

1. Define $f(x)$:

$$f(x) = 10xe^x - 7$$

2. Initial values:
   - $a = 0,\ f(0) = 10(0)e^0 - 7 = -7,$
   - $b = 1, f(1) = 10(1)e^1 - 7 \approx 20.28$

Since $f(a)f(b) < 0$, a root exists in $[0,1]$.

3. Iterative steps: Using the false position formula:

$$c = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

We'll compute this iteratively until the desired accuracy is reached.

The root of the equation $10xe^x - 7 = 0$ in the interval $[0,1]$ is approximately:

$$x \approx 0.4475$$

Details of the Last Iteration:

- Interval: $[a, b] = [0.447465,1]$
- Computed root: $c = 0.447468$
- Function value at root: $f(c) \approx -4.236 \times 10^{-5}$ (satisfying the accuracy 10^{-4}.

Thus, the solution converges within the desired accuracy.

Iteration	a	b	c	$f(c)$
1	0.2575	1	0.2575	−3.6685
2	0.3717	1	0.3717	−1.6093
3	0.4181	1	0.4181	−0.6485
4	0.4362	1	0.4362	−0.2522
5	0.4432	1	0.4432	−0.0967
6	0.4458	1	0.4458	−0.0369
7	0.4469	1	0.4469	−0.014
8	0.4472	1	0.4472	−0.0053
9	0.4474	1	0.4474	−0.002
10	0.4474	1	0.4474	−0.0008

- **Advantages**: Faster convergence than bisection for some functions due to the secant approximation.
- **Limitations**: May converge slowly for functions where one endpoint remains fixed over multiple iterations.

4. Fixed-Point Iteration

The fixed-point method is a numerical technique used to solve nonlinear algebraic equations of the form:

$$f(x) = 0$$

This equation can be reformulated as:

$$x = g(x)$$

where $g(x)$ is a function derived from $f(x)$. The method involves iterating an initial guess x_0 using the formula:

$$x_{n+1} = g(x_n)$$

The iteration continues until the sequence $\{x_n\}$ converges to a fixed point x^*, which satisfies $x^* = g(x^*)$.

---

2. Steps of the Fixed-Point Method

1. **Reformulate $f(x) = 0$ as $x = g(x)$:** Choose an appropriate $g(x)$ such that $x = g(x)$ is equivalent to $f(x) = 0$.
2. **Ensure that $g(x)$ satisfies $|g'(x_i)| < 1, \forall i$**

3. **Initial Guess x_0:** Start with an initial guess x_0 close to the root.
4. **Iterative Formula:** Compute successive approximations using:

$$x_{n+1} = g(x_n)$$

5. **Convergence Check:** Stop the iteration when:

$$| x_{n+1} - x_n | < \epsilon$$

where ϵ is a pre-defined tolerance.

---

3. Convergence Analysis

The convergence of the fixed-point method depends on the properties of $g(x)$. Specifically, the method converges if:

$| g'(x) | < 1$ for all x in the neighborhood of the root

Proof of Convergence Condition:

Let x^* be the fixed point of $g(x)$, i.e., $x^* = g(x^*)$. Consider two successive iterates x_n and x_{n+1}:

$$x_{n+1} - x^* = g(x_n) - g(x^*)$$

Using the Mean Value Theorem for $g(x)$, there exists a point ξ between x_n and x^* such that:

$$g(x_n) - g(x^*) = g'(\xi)(x_n - x^*)$$

Taking the absolute value:

$$| x_{n+1} - x^* | = | g'(\xi) | \cdot | x_n - x^* |$$

If $| g'(\xi) | < 1$, then:

$$| x_{n+1} - x^* | < | x_n - x^* |$$

Thus, the sequence $\{x_n\}$.

Example 8.3

Solve $f(x) = x^2 - 2 = 0$ using the fixed-point method.

Solution:

Step 1: Reformulate

Rearrange $x^2 - 2 = 0$ as $x = g(x)$. Choose:

$$g(x) = \frac{1}{2}(x + \frac{2}{x})$$

This formulation ensures the iterations remain well-behaved.

Step 2: Initial Guess

Let $x_0 = 1.5$.

Step 3: Iteration

Using $x_{n+1} = g(x_n)$, compute successive approximations:

1. $x_1 = g(1.5) = \frac{1}{2}(1.5 + \frac{2}{1.5}) = 1.4167$
2. $x_2 = g(1.4167) = \frac{1}{2}(1.4167 + \frac{2}{1.4167}) = 1.4142$
3. $x\_3n = g(1.4142) = \frac{1}{2}(1.4142 + \frac{2}{1.4142}) = 1.4142$

Step 4: Convergence Check

The iterations converge to $x^* = 2 \approx 1.4142$.

The fixed-point method is a fundamental tool for solving nonlinear equations, with its convergence contingent on the appropriate choice of $g(x)$ and adherence to the condition $|g'(x)| < 1$. Properly applied, it is both efficient and accurate for a wide range of problems.

- **Advantages**: Simple to implement, and the method can converge rapidly if $g(x)$ satisfies specific conditions (e.g., the derivative $|g'(x)| < 1$ in the neighborhood of the root).
- **Limitations**: Convergence is not guaranteed if $g(x)$ is poorly chosen or if $|g'(x)| \geq 1$.

4. Newton-Raphson Method

The Newton-Raphson method is a derivative-based iterative technique that uses the tangent line at a point to approximate the root. Starting from an initial guess x_0, the next approximation is given by:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

The **Newton-Raphson method** is an iterative numerical technique for finding the roots of a nonlinear equation $f(x) = 0$. The method is renowned for its **quadratic convergence**, making it highly efficient when the initial guess is close to the actual root.

Method Derivation

Given $f(x) = 0$, let $x = r$ be the root. Assume an initial guess x_0. Using the Taylor expansion around x_0:

$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(c)}{2}(x - x_0)^2$, for some $c \in [x0, x]$.

Neglecting higher-order terms, we approximate:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

Setting $f(x) = 0$ to solve for x, we get:

$$x \approx x_0 - f(x_0)f'(x_0)$$

Thus, the iterative formula is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Convergence Conditions and Fixed-Point Connection

The Newton-Raphson method can be interpreted as a fixed-point iteration $x_{n+1} = g(x_n)$, where:

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

For the method to converge, the following conditions must be satisfied:

1. $g'(x)$ exists and is continuous near the root r.
2. $|\, g'(x) \,| < 1$ near r.

From $g'(x)$, we have:

$$g'(x) = 1 - f'(x)^2 - \frac{f(x)f''(x)}{f'(x)^2}.$$

At the root $x = r$, $f(r) = 0$, so:

$$g'(r) = 0.$$

This indicates that the convergence of the Newton-Raphson method is at least quadratic near r.

Proof of Quadratic Convergence

Let $e_n = x_n - r$ be the error at the n-th step. Expanding $f(x_n)$ using the Taylor series around r:

$$f(x_n) = f(r) + f'(r)e_n + \frac{f''(r)}{2}e_n^2 + O(e_n^3).$$

Since $f(r) = 0$, we have:

$$f(x_n) = f'(r)e_n + \frac{f''(r)}{2}e_n^2 + O(e_n^3).$$

The Newton-Raphson update becomes:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Substituting $f(x_n)$ and $f'(x_n) \approx f'(r)$, we get:

$$x_{n+1} = x_n - \frac{f'(r)e_n + \frac{f''(r)}{2}e_n^2}{f'(r)}.$$

Simplifying:

$$x_{n+1} = x_n - e_n - \frac{f''(r)}{2f'(r)}e_n^2.$$

Thus, the error at the next iteration is:

$$e_{n+1} = -\frac{f''(r)}{2f'(r)}e_n^2 + O(e_n^3).$$

This shows that $e_{n+1} \propto e_n^2$, proving quadratic convergence.

Example 8.4

Solve $f(x) = x^3 - 2x - 5 = 0$ using the Newton-Raphson method.

1. **Define the function and its derivative**:

$f(x) = x^3 - 2x - 5, \qquad f'(x) = 3x^2 - 2.$

2. **Initial guess**: $x_0 = 2.$

3. **Iterations**:

- Iteration 1: $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{2^3-2(2)-5}{3(2)^2-2} = 2.1$
- Iteration 2: $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 2.1 - \frac{2.1^3-2(2.1)-5}{3(2.1)^2-2} = 2.0943$

  Iteration 3: $x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 2.0943 - \frac{2.0943^3-2(2.0943)-5}{3(2.0943)^2-2} = 2.094551$

4. **Root**: After sufficient iterations, the root converges to $x \approx 2.094551$.

To find the r-th root of N ($\sqrt[r]{N}$), we define the function $f(x)$ such that its root corresponds to the desired root:

$$f(x) = x^r - N$$

Step 1: Compute the derivative of $f(x)$

$$f'(x) = rx^{r-1}$$

Step 2: Substitute into the Newton-Raphson formula

$$x_{n+1} = x_n - \frac{x_n^r - N}{rx_n^{r-1}}$$

Simplify the fraction:

$$x_{n+1} = x_n - \frac{x_n}{r} + \frac{N}{rx_n^{r-1}}$$

Combine terms:

$$x_{n+1} = \frac{(r-1)x_n^r + N}{rx_n^{r-1}}$$

This is the iterative formula for finding the r-th root of N.

Example 8.5:

Compute $\sqrt[3]{29}$ (the cube root of 29).

Solution:

Here:

- $r = 3$ (since we are calculating the cube root),
- $N = 29$.

Step-by-step process:

1. Start with an initial guess x_0 .
2. Use the formula:

$$x_{n+1} = \frac{(3-1)x_n^3 + 29}{3x_n^2}$$

Let's compute this numerically using $x_0 = 3$ as the initial guess.

$$n = 0: \qquad x_1 = \frac{2x_0^3 + 29}{3x_0^2} = \frac{2(3)^3 + 29}{3(3)^2} = 3.074$$

$$n = 1: \qquad x_2 = \frac{2x_1^3 + 29}{3x_1^2} = \frac{2(3.074)^3 + 29}{3(3.074)^2} = 3.07231$$

$$n = 2: \qquad x_2 = \frac{2x_1^3 + 29}{3x_1^2} = \frac{2(3.07231)^3 + 29}{3(3.07231)^2} = 3.07231$$

Using the Newton-Raphson method, the approximate value of the cube root of 29 is 3.07231.

Advantages: Quadratic convergence near the root, making it one of the fastest methods for well-behaved functions.

Limitations: Requires the computation of the derivative $f'(x)$, and convergence is not guaranteed if the initial guess is far from the root, or the function has inflection points.

Conclusion

Each numerical method for solving nonlinear equations has unique features suited to different types of problems. Bracketing methods like bisection and false-position offer robustness but may be slower, while open methods like fixed-point iteration and Newton-Raphson provide faster convergence but require more careful initial guesses and conditions. Understanding the

properties of the function $f(x)$ and the problem context is crucial for selecting the most appropriate method.

Exercise 8

Bisection Method

1. **Determine the break-even point in a business:**
   - $f(x) = x^3 - 2x^2 + 3x - 5$, Interval: $[1,3]$.
2. **Calculate the velocity at which drag equals a given force in a fluid:**
   - $f(x) = x^2 + 5x - 20$, Interval: $[2,6]$
3. **Find the root of an electronics circuit equation:**
   - $f(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$, Interval: $[0.5,2.5]$
4. **Solve a heat transfer equation in a rod:**
   - $f(x) = tan\,(x) - x$, Interval: $[0, \pi/2]$

False Position Method

5. **Solve a projectile motion problem for the angle θ:**
   - $f(x) = x^2 sin\,(2x) - 5$, Interval: $[0, \pi/4]$
6. **Determine the pressure in a fluid flow problem:**
   - $f(x) = l\,n(x) + x^2 - 4$, Interval: $[1,2]$
7. **Find the solution for an electrical power balance:**
   - $f(x) = e^x - 3x^2 + 5$, Interval: $[0,2]$
8. **Locate the root of a chemical reaction rate equation:**

- $f(x) = x^3 - 2x^2 + x - 1$, Interval: $[0.5,1.5]$

Fixed Point Method

9. **Determine the steady-state temperature in a cooling problem:**
   - $g(x) = 4 + x^2$, Initial guess: $x_0 = 1.5$
10. **Find the equilibrium concentration in a chemical reaction:**
    - $g(x) = e^{-x}$, Initial guess: $x_0 = 0.8$
11. **Solve a damping problem in mechanics:**
    - $g(x) = 1 + 2x$, Initial guess: $x_0 = 2.5$
12. **Determine the flow rate in a pipeline system:**
    - $g(x) = x - ln\,(x) - 3$, Initial guess: $x_0 = 1.5$

Newton-Raphson Method

13. **Find the natural frequency of a vibrating system:**
    - $f(x) = x^3 - 4x^2 + 10x - 6$, Initial guess: $x_0 = 2$
14. **Solve for the root in a thermodynamics pressure-volume relation:**
    - $f(x) = x^3 - 6x + 4$, Initial guess: $x_0 = 1.5$
15. **Find the intersection point in an optical lens system:**
    - $f(x) = e^x - x^2 - 2$, Initial guess: $x_0 = 0.5$

Chapter 9

Solving Linear Equations

Linear systems of equations are foundational in mathematics, engineering, and applied sciences, often arising in fields such as physics, economics, and computer science. These systems can be represented in the matrix form:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$A\boldsymbol{x} = \boldsymbol{b},$$

where A is an $n \times n$ coefficient matrix, $\boldsymbol{x}$ is an $n \times 1$ vector of unknowns, and $\boldsymbol{b}$ is an $n \times 1$ vector of constants. While analytical methods such as Gaussian elimination or matrix inversion are effective for small systems, they become computationally expensive and less robust for larger systems. This has led to the

development of numerical methods, which provide iterative or factorization-based approaches to solve such systems efficiently.

This chapter introduces three widely used numerical methods for solving linear systems: the **Jacobi Method**, the **Gauss-Seidel Method**, and **LU Factorization**. These methods offer distinct advantages and are suited to different scenarios, depending on the problem's size, sparsity, and convergence requirements.

---

1. Jacobi Method

The Jacobi Method is an iterative approach that reformulates the linear system into a sequence of successively approximated solutions.

Firstly, we rearrange the linear system to satisfy the following conditions

$$|a_{11}| > |a_{12}| + |a_{13}| + \cdots + |a_{1n}|$$

$$|a_{22}| > |a_{21}| + |a_{23}| + \cdots + |a_{2n}|$$

$$\vdots$$

$$|a_{nn}| > |a_{n1}| + |a_{n2}| + \cdots + |a_{nn-1}|$$

Secondly, rewrite the following equation as:

$$x_1 = \frac{1}{a_{11}}[b_1 - (a_{12}x_2 + \cdots + a_{1n}x_n)]$$

$$x_2 = \frac{1}{a_{22}}[b_2 - (a_{21}x_1 + \cdots + a_{2n}x_n)]$$

$$\vdots$$

$$x_n = \frac{1}{a_{nn}}[b_n - (a_{n1}x_1 + \cdots + a_{nn-1})]$$

Finally, apply the iteration:

$$x_1^{(i+1)} = \frac{1}{a_{11}}\left[b_1 - \left(a_{12}x_2^{(i)} + \cdots + a_{1n}x_n^{(i)}\right)\right]$$

$$x_2^{(i+1)} = \frac{1}{a_{22}}\left[b_2 - \left(a_{21}x_1^{(i)} + \cdots + a_{2n}x_n^{(i)}\right)\right]$$

$$\vdots$$

$$x_n^{(i+1)} = \frac{1}{a_{nn}}\left[b_n - \left(a_{n1}x_1^{(i)} + \cdots + a_{nn-1}^{(i)}\right)\right]$$

Example 9.1

Apply the Jacobean iterative method to find an approximate solution to the following linear system (Start with the initials $(x_0, y_0, z_0) = (0,0,0)$).

$$4x - y + 2z = 21,$$

$$x + y - 7z = -1,$$
$$-x + 12y + 5z = 12.$$

Solution:

We rewrite the system as:

$$\begin{aligned} 4x - y + 2z = 21, \quad & |4| > |-1| + |2| \\ -x + 12y + 5z = 12. \quad & |12| > |-1| + |5| \\ x + y - 7z = -1, \quad & |-7| > |1| + |1| \end{aligned}$$

Then

$$x = \frac{1}{4}(21 + y - 2z)$$

$$y = \frac{1}{12}(12 + x - 5z)$$

$$z = \frac{1}{7}(1 + x + y)$$

Then, we apply the iteration:

$$x_{i+1} = \frac{1}{4}(21 + y_i - 2z_i)$$

$$y_{i+1} = \frac{1}{12}(12 + x_i - 5z_i)$$

$$z_{i+1} = \frac{1}{7}(1 + x_i + y_i)$$

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} 5.25 \\ 1 \\ 0.143 \end{bmatrix}$	$\begin{bmatrix} 5.4285 \\ 1.3779 \\ 1.0357 \end{bmatrix}$	$\begin{bmatrix} 5.076636 \\ 1.020833 \\ 1.115221 \end{bmatrix}$	$\begin{bmatrix} 4.94759 \\ 0.95837 \\ 1.01392 \end{bmatrix}$	$\begin{bmatrix} 4.9826 \\ 0.9898 \\ 0.9865 \end{bmatrix}$	$\begin{bmatrix} 5.0041 \\ 1.0041 \\ 0.9960 \end{bmatrix}$

∴ The approximate solution is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5.0041 \\ 1.0041 \\ 0.9960 \end{bmatrix}.$$

---

2. Gauss-Seidel Method

The Gauss-Seidel Method improves upon the Jacobi Method by utilizing updated values of x as soon as they are available within an iteration.

Firstly, we rearrange the linear system to satisfy the following conditions

$$|a_{11}| > |a_{12}| + |a_{13}| + \cdots + |a_{1n}|$$

$$|a_{22}| > |a_{21}| + |a_{23}| + \cdots + |a_{2n}|$$

$$\vdots$$

$$|a_{nn}| > |a_{n1}| + |a_{n2}| + \cdots + |a_{nn-1}|$$

Secondly, rewrite the following equation as:

$$x_1 = \frac{1}{a_{11}}[b_1 - (a_{12}x_2 + \cdots + a_{1n}x_n)]$$

$$x_2 = \frac{1}{a_{22}}[b_2 - (a_{21}x_1 + \cdots + a_{2n}x_n)]$$

$$\vdots$$

$$x_n = \frac{1}{a_{nn}}[b_n - (a_{n1}x_1 + \cdots + a_{nn-1})]$$

Finally, apply the iteration:

$$x_1^{(i+1)} = \frac{1}{a_{11}}\left[b_1 - \left(a_{12}x_2^{(i)} + a_{13}x_3^{(i)} + \cdots + a_{1n}x_n^{(i)}\right)\right]$$

$$x_2^{(i+1)} = \frac{1}{a_{22}}\left[b_2 - \left(a_{21}x_1^{(i+1)} + a_{23}x_3^{(i)} + \cdots + a_{2n}x_n^{(i)}\right)\right]$$

$$x_3^{(i+1)} = \frac{1}{a_{33}}\left[b_3 - \left(a_{31}x_1^{(i+1)} + a_{32}x_2^{(i+1)} + \cdots + a_{3n}x_n^{(i)}\right)\right]$$

$$\vdots$$

$$x_{n-1}^{(i+1)} = \frac{1}{a_{n-1n-1}}\left[b_{n-1} - \left(a_{n-11}x_1^{(i+1)} + \cdots + a_{n-1n}^{(i)}\right)\right]$$

$$x_n^{(i+1)} = \frac{1}{a_{nn}}\left[b_n - \left(a_{n1}x_1^{(i+1)} + a_{n2}x_2^{(i+1)} + \cdots + a_{nn-1}^{(i+1)}\right)\right]$$

Example 9.2

Apply the Gauss-Seidel iterative method to find an approximate solution to the following linear system (Start with the initials $(y_0, z_0) = (0,0)$.

$$4x - y + 2z = 21,$$
$$x + y - 7z = -1,$$
$$-x + 12y + 5z = 12.$$

Solution:

We rewrite the system as:

$$4x - y + 2z = 21, \quad |4| > |-1| + |2|$$
$$-x + 12y + 5z = 12. \quad |12| > |-1| + |5|$$
$$x + y - 7z = -1, \quad |-7| > |1| + |1|$$

Then

$$x = \frac{1}{4}(21 + y - 2z)$$

$$y = \frac{1}{12}(12 + x - 5z)$$

$$z = \frac{1}{7}(1 + x + y)$$

Then, we apply the iteration:

$$x_{i+1} = \frac{1}{4}(21 + y_i - 2z_i)$$

$$y_{i+1} = \frac{1}{12}(12 + x_{i+1} - 5z_i)$$

$$z_{i+1} = \frac{1}{7}(1 + x_{i+1} + y_{i+1})$$

	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} 5.2500 \\ 1.4375 \\ 1.0982 \end{bmatrix}$	$\begin{bmatrix} 5.0603 \\ 0.9641 \\ 1.0035 \end{bmatrix}$	$\begin{bmatrix} 4.9893 \\ 0.9977 \\ 0.9981 \end{bmatrix}$	$\begin{bmatrix} 5.0003 \\ 1.0008 \\ 1.0002 \end{bmatrix}$	$\begin{bmatrix} 5.0001 \\ 0.9999 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 1 \\ 1 \end{bmatrix}$

∴ The approximate solution is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 1 \end{bmatrix}.$$

---

3. LU Factorization

LU Factorization is a direct method that decomposes the matrix A into a product of a lower triangular matrix L and an upper triangular matrix U:

$$A = LU.$$

The system $Ax = b$ can then be solved in two steps:

1) Solve $Ly = b$ using forward substitution.
2) Solve $Ux = y$ using back substitution.

LU Factorization is computationally efficient for multiple systems with the same coefficient matrix but different right-hand side vectors. It is also used as a building block for more advanced numerical techniques.

The LU decomposition offers several practical benefits:

1. Efficient Solution of Linear Systems

- Once A is decomposed into L and U, solving $Ax = b$ becomes more efficient:
- This avoids repeatedly solving the system for multiple right-hand sides b as L and U are reusable.

2. Reduction in Computational Complexity

- Direct methods like Gaussian elimination require $O(n^3)$ operations for each system solving. With LU decomposition, the factorization is performed once $O(n^3)$, but subsequent solves require only $O(n^2)$ for substitution.

3. Numerical Stability

- LU decomposition with partial pivoting (known as $PA = LU$ where P is a permutation matrix) can improve numerical stability, minimizing rounding errors during computation.

4. Facilitates Matrix Inversion

- If A is invertible, its inverse A^{-1} can be computed efficiently using L and U, reducing the computational effort compared to other methods.

5. Determinant Calculation

- The determinant of A can be easily computed as the product of the diagonal elements of U (or L), reducing computational overhead.

6. Applications in Eigenvalue and Singular Value Problems

- LU decomposition serves as a preprocessing step in numerical methods for computing eigenvalues and singular values of matrices.

7. Ease of Matrix Updates

- In cases where a matrix undergoes slight modifications (e.g., adding or removing rows/columns), updating L and U is computationally simpler than recomputing the entire decomposition.

8. Memory Efficiency

- The decomposition splits the problem into triangular matrices, which are more memory-efficient to store and manipulate.

In summary, LU decomposition is a powerful tool in numerical linear algebra, providing computational efficiency, stability, and flexibility in solving various matrix-related problems.

Example 9.3

A) Put the following matrix in the form of L. U. Factorization

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & 3 \\ 1 & 0 & -2 \end{pmatrix}$$

B) Use the resulted L and U matrices to solve the following linear system

$$x + y + z = 6$$

$$2x - y + 3z = 9$$

$$x \quad - z = -2$$

Solution:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & 3 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \\ -2 \end{bmatrix}$$

$$A\boldsymbol{x} = \boldsymbol{b}$$

Set

$$A = LU$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & 3 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$\therefore u_{11} = 1, \therefore u_{12} = 1, \qquad \therefore u_{13} = 1$

$\therefore l_{21} = 2, \qquad -1 = (2)(1) + u_{22} \quad \therefore u_{22} = -3,$

$3 = (2)(1) + u_{23}, \quad \therefore u_{23} = 1 \qquad \therefore l_{31} = 1$

$0 = (1)(1) + (-3)l_{32}, \qquad \therefore l_{32} = \frac{1}{3}$

$-1 = (1)(1) + \frac{1}{3}(1) + u_{33}, \qquad \therefore u_{33} = -\frac{7}{3}$

Thus

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -3 & 1 \\ 0 & 0 & -\frac{7}{3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \\ -2 \end{bmatrix}$$

Put

$$Ly = b$$

$$\therefore \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 9 \\ -2 \end{bmatrix}$$

1) Solve $Ly = b$ using forward substitution we have

$y_1 = 6, \qquad y_2 = -3, \qquad y_3 = -7$

2) Solve $Ux = y$ using back substitution.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -3 & 1 \\ 0 & 0 & -\frac{7}{3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ -7 \end{bmatrix}$$

$$\therefore (x, y, z) = (1,2,3)$$

Comparative Analysis

Jacobi and Gauss-Seidel are iterative methods and are suitable for large, sparse systems, particularly when an approximate solution is acceptable, or the coefficient matrix is well-conditioned.

LU Factorization, being a direct method, is advantageous for dense systems or when solving multiple systems with a fixed matrix A.

Each method has its strengths and limitations, and their effectiveness depends on the problem characteristics. Understanding these methods provides a robust foundation for tackling real-world linear systems in various scientific and engineering domains.

Exercises 9

Jacobi and Gauss-Seidel Method Exercises

1. **Traffic Flow Control**

   A city's traffic network is modeled as:

   $$10x - 2y + z = 28$$

   $$-3x + 15y + 2z = 63$$

   $$x + y + 20z = 53$$

   Use Jacobi and Gauss-Seidel methods to find x, y, z, starting with $(x_0, y_0, z_0) = (0, 0, 0)$.

2. **Heat Transfer in a Plate**

   The heat distribution in a plate is modeled as:

   $$4x - y - z = 15$$

   $$-x + 4y - z = 10$$

   $$-x - y + 4z = 10$$

   Find x, y, z with an initial guess of $(x_0, y_0, z_0) = (1, 1, 1)$.

3. **Chemical Reactions in a Reactor**

   Reaction rates are governed by:

   $$8x + y - z = 34$$

   $$x + 7y + z = 27$$

   $$-x + y + 6z = 22$$

   Solve using $(x_0, y_0, z_0) = (2, 2, 2)$.

4. **Structural Analysis in Engineering**

   Forces in a truss are modeled as:

$$5x - 2y + z = 12$$

$$-x + 4y - 2z = -1$$

$$x + y + 3z = 7$$

   Use Jacobi and Gauss-Seidel methods with $(x_0, y_0, z_0) = (0, 0, 0)$.

5. **Electrical Circuit Analysis**

   Circuit currents are modeled as:

$$10x - y + 2z = 27$$

$$-2x + 8y - z = -15$$

$$x - y + 6z = 25$$

   starting from $(x_0, y_0, z_0) = (0, 0, 0)$.

LU-Factorization Method Exercises

6. **Warehouse Optimization**

   Inventory management requires solving:

$$6x + 3y - z = 21$$

$$2x + y + 4z = 15$$

$$-x + 5y + 6z = 30$$

   Perform LU factorization to solve x, y, z.

7. **Portfolio Balancing**

   Asset allocations satisfy:

$$3x + 2y - z = 14$$

$$x + 4y + 3z = 20$$

$$-x + 2y + 5z = 25$$

   Solve using LU decomposition.

8. **Energy Minimization**

   Thermal system equations are:

$$7x + 2y - z = 32$$

$$2x + 8y + z = 25$$

$$x - y + 5z = 14$$

   Apply LU factorization to find x, y, z.

9. **Production System Optimization**

   The equations governing production rates are:

$$10x - 3y + 2z = 50$$

$$3x + 10y - z = 40$$

$$2x - y + 8z = 60$$

   Solve using LU decomposition.

10. **Vibration Analysis in Mechanics**

    System vibrations satisfy:

$$5x + 3y - z = 22$$

$$3x + 6y + 2z = 30$$

$$x + 2y + 7z = 28$$

Use LU decomposition to solve for x, y, z.