



UNIVERSITY OF MINES AND  
TECHNOLOGY (UMaT),  
TARKWA

MATHEMATICAL SCIENCES DEPARTMENT

# Numerical Analysis

Course Code: MN/MR/MC/EL/CE/CY/RN 363

*Compiled by*

**Albert Buabeng**

(PhD, MPhil, BSc)

## COURSE OUTLINE (SYLLABUS)

- Introduction
- Error analysis
- Iterative methods of solving systems of linear equations
- Iterative methods of solving systems of non-linear equations
- Interpolation
- Numerical solutions of ordinary and partial differential equations
- Application of computer programming

**OBJECTIVE:** The primary objective of this course is to provide you with a concrete idea of what numerical methods are and how they relate to engineering and scientific problem solving.

**OUTCOME:** At the end of the course, it is expected that students will understand the concept behind numerical methods and be able to apply them in finding solutions to problems relating to engineering and science. In particular, the students will become proficient in:

- Understanding the theoretical and practical aspects of the use of numerical methods;
- Implementing numerical methods for a variety of multidisciplinary applications;
- Establishing the limitations, advantages, and disadvantages of numerical methods.

**PREREQUISITES:** It is assumed that the student has some background in Algebra, Calculus and Computing.

**GRADING CRITERIA AND EVALUATION PROCEDURES:** The grade for the course will be based on class attendance, group homework, quizzes/ class test and a final end of term exams.

- **Attendance:** All students should make it a point to attend classes. Random attendance will be taken to constitute 10% of the grade.
- **Group Homework:** Two homework assignments worth 10% of the final grade. Homework will be assigned on regular basis and will be due exactly one week (before 5:00 pm) from the day the homework is issued to students

- **Group Presentation:** A group presentation worth 5% of the final grade will be conducted where necessary. Students will be assigned to a group with a task to research into and present their findings in class to member.
- **Quizzes/ Class Test:** Two quizzes worth 15% of the grade will be given during class. The quiz or test date will be announced one week in advance.
- **Final End-Of-Term Exams:** Final exam is worth 60% of the final grade.

## CONTACT

Email: [abuabeng@umat.edu.gh](mailto:abuabeng@umat.edu.gh)

VLE - Class Name: Numerical Analysis (For Engineers)

VLE - Class Code: NumAna@25

Zoom Meeting ID: 613 2022 1000

Password: 963018

### COURSE PLAN (SCHEDULE)

Date			Chapter	Activity
Week	Month	Day		
1	January	Week 1	1	Semester's Activities/Overview <b>Introduction to Numerical Analysis</b>
2		Week 2	2	<b>Introduction to Numerical Analysis</b> <b>Error Analysis</b> Accuracy and Precision Numerical Errors Types of Errors
3	February	Week 3		Numerical Differentiation Control of Numerical Errors Try Exercises
4		Week 4	3	<b>Numerical Solution to Nonlinear Eqns</b> Introduction Graphical Method Bracketing Methods Bisection Method
5		Week 5		Bisection Method (with MATLAB) False Position
6		Week 6		Open Methods Newton Raphson Secant Method <b>Quiz I</b>
8	March	Week 7	4	<b>Iterative Methods for Solving Linear Systems</b> Introduction Gauss-Siedel Method Jacobi Method
9		Week 8	5	<b>Numerical Interpolation</b> Introduction Newton Linear <b>Assessment of Course Delivery</b>
10		Week 9		Quadratic General Form (Newton Interpolation) Lagrange
11		Week 10	6	<b>Quiz II</b> <b>Numerical Solutions of ODEs and PDEs</b> Introduction Euler Method Runge Kutta Method
12		Week 11		Runge Kutta Method Conclusion
	April	Week 12 -		<b>First Semester Examinations</b>

## TABLE OF CONTENTS

Content	Page
<b>TABLE OF CONTENTS</b>	<b>i</b>
<b>LIST OF FIGURES</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>v</b>
<b>LIST OF EXAMPLES</b>	<b>vi</b>
<b>CHAPTER 1 INTRODUCTION TO NUMERICAL ANALYSIS AND ERROR ANALYSIS</b>	<b>1</b>
1.1 Objectives	1
1.2 You've Got a Problem	1
1.3 What is Numerical Analysis?	2
1.3.1 Numerical Methods Covered	4
1.4 Try Exercise	11
<b>CHAPTER 2 ERROR ANALYSIS (ROUND OFF AND TRUNCATION ERRORS)</b>	<b>13</b>
2.1 Objectives	13
2.2 You've Got a Problem	13
2.3 Errors	14
2.4 Accuracy and Precision	14
2.5 Numerical Errors	15
2.5.1 Sources of Numerical Errors	18
2.6 Types of Errors	19
2.6.1 Round off or Machine Error	19
2.6.2 Truncation Errors	19
2.7 Numerical Differentiation	21
2.7.1 Backward Difference Approximation of the First Derivative	22
2.7.2 Centred Difference Approximation of the First Derivative	22
2.8 Total Numerical Error	24
2.8.1 Control of Numerical Errors	25
2.9 Try Exercises	26
<b>CHAPTER 3 NUMERICAL SOLUTION TO NONLINEAR EQUATIONS</b>	<b>28</b>
3.1 Introduction	28
3.2 Roots: Bracketing Methods	29
3.2.1 Objectives	29

3.2.2	You've got a Problem	29
3.2.3	Roots in Engineering and Science	30
3.2.4	Graphical Methods	31
3.3	Bracketing Methods and Initial Guesses	33
3.4	Bisection Method	33
3.4.1	Assumptions	34
3.4.2	Algorithm: Bisection Method	34
3.4.3	Criterion for Termination	34
3.4.4	MATLAB M-file: Bisection	40
3.4.5	Merits of Bisection Method	40
3.4.6	Demerits of Bisection Method	40
3.5	Regula Falsi Method (Method of False Position)	40
3.5.1	Similarities with Bisection Method	41
3.5.2	Differences with Bisection Method	41
3.5.3	MATLAB M-file: Regula Falsi	44
3.5.4	Try Exercise	44
3.6	Roots: Open Methods	46
3.6.1	Objectives	46
3.6.2	Introduction	46
3.7	Newton-Raphson Method	47
3.7.1	MATLAB M-file: Newton Raphson	50
3.8	Secant Method	50
3.8.1	MATLAB M-file: Secant	54
3.8.2	Try Exercise	54
<b>CHAPTER 4 ITERATIVE METHODS FOR SOLVING LINEAR SYSTEMS</b>		<b>57</b>
4.1	Objectives	57
4.2	Introduction	57
4.2.1	What are Linear Algebraic Equations?	57
4.2.2	Linear Algebraic Equations in Engineering and Science	57
4.2.3	You've Got a Problem	58
4.3	Iterative Methods	60
4.4	Gauss-Seidel Iterative Method	60
4.4.1	Weakness of Gauss-Seidel Iterative Method	65
4.4.2	MATLAB M-file: Gauss-Seidel	68

4.5	Jacobi Iterative Method	68
4.6	Try Exercise	73
<b>CHAPTER 5 NUMERICAL INTERPOLATION (POLYNOMIALS)</b>		<b>74</b>
5.1	Objectives	74
5.2	You've Got a Problem	74
5.3	Introduction to Interpolation	75
5.3.1	Determining Polynomial Coefficients	75
5.4	Newton Interpolating Polynomial	77
5.4.1	Linear Interpolation	77
5.4.2	Quadratic Interpolation	79
5.4.3	General Form of Newton's Interpolating Polynomials	81
5.5	Lagrange Interpolating Polynomial	83
5.5.1	MATLAB M-file: Lagrange Interpolation	85
5.6	Try Exercises	85
<b>CHAPTER 6 NUMERICAL SOLUTION OF ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS</b>		<b>88</b>
6.1	Objectives	88
6.2	You've Got a Problem	88
6.3	Introduction	89
6.4	Euler Method	90
6.4.1	MATLAB M-file: Euler's Method	94
6.5	The Runge-Kutta (RK) Methods	94
6.5.1	Classical Fourth Order Runge-Kutta (RK4) Method	94
6.5.2	MATLAB M-file: 4th Order Runge-Kutta	99
6.6	Try Exercise	99
<b>REFERENCES</b>		<b>100</b>
<b>APPENDICES</b>		<b>101</b>
Appendix A An M-file to implement the Bisection Method		101
Appendix B An M-file to implement the Regula Falsi Method		102
Appendix C An M-file to implement the Newton Rapson Method		104
Appendix D An M-file to implement the Secant Method		105
Appendix E An M-file to implement the Guass-Seidel Method		106
Appendix F An M-file to implement the Lagrange Interpolation		108
Appendix G An M-file to implement the Euler's Method		109
Appendix H An M-file to implement the Runge-Kutta Method		110

## LIST OF FIGURES

Figure	Title	Page
1.1	Forces Acting on a Free-Falling Bungee Jumper	1
1.2	Some Engineering Areas of Primary Focus	5
1.3	Summary of the Numerical Methods Covered	6
1.4	Velocity-Time Plot of Bungee Jumper	7
1.5	Finite Difference to Approximate the First Derivative of $v$ with respect to $t$	8
1.6	Comparison of the Numerical and Analytical Solutions for the Bungee Jumper Problem	10
1.7	Storage Tank	11
2.1	(a) Inaccurate and Imprecise, (b) Accurate and Imprecise, (c) Inaccurate and Precise (d) Accurate and Precise	14
2.2	(a) Forward, (b) Backward, and (c) Centred Finite-Difference Approximations of the First Derivative	22
2.3	Trade-Off between Roundoff and Truncation Error	25
3.1	Difference b/n Roots and Optima	29
3.2	Plot of Equation (3.4) versus Mass	31
3.3	Occurrence of Roots in an Interval	32
3.4	Graphical Depiction of the Bisection Method	33
3.5	Plot of Equation (3.8) versus Mass	35
3.6	False Position	41
3.7	Fundamental difference b/n (a) Bracketing and (b), (c) Open Methods for Root Location	47
3.8	Newton-Raphson Method	47
3.9	Nonlinear Spring	55
3.10	Spherical Tank	56
4.1	System Involving Systems of Equations	58
4.2	Three Individuals Connected by Bungee Cords	59
4.3	Free-Body Diagrams	59
4.4	Difference b/n (a) the Gauss-Seidel, (b) the Jacobi Iterative Methods	69
5.1	Interpolating Polynomials	75
5.2	Two Linear Interpolations to Estimate $\ln 2$	78
5.3	Comparison of Quadratic and Linear Interpolation of $\ln 2$	80
5.4	Recursive Nature of Finite Divided Differences	82
5.5	Cubic Interpolation to Estimate $\ln 2$	83
5.6	Rationale behind Lagrange Interpolating Polynomials (First-Order)	84
6.1	Numerical evaluation of $\frac{dy}{dt} = y' = f(t, y)$	90
6.2	Slope Estimates of RK4 Method	95



## LIST OF TABLES

Table	Title	Page
1.1	Analytical Solution	7
1.2	Numerical Solution	10
2.1	Summary of Computation	18
3.1	Fundamental Principles used in Design Problems	30
3.2	Summary of Computation	37
3.3	Summary of Computation	40
3.4	Summary of Bisection Method	43
3.5	Summary of False Position Method	43
3.6	Summary of Newton-Raphson Method	48
3.7	Summary of Secant Method	52
3.8	Summary of Modified Secant Method	53
4.1	Summary of Gauss-Seidel Computation	65
4.2	Summary of Gauss-Seidel (Showing Weakness)	68
5.1	Fluid Mechanics Parameters	74
5.2	Divided Difference Table	83
5.3	Dissolved Oxygen (mg/L)	86
6.1	Summary of Euler's Method Computation	92
6.2	Summary of Euler's Method Computation	94

## LIST OF EXAMPLES

Example	Title	Page
1.1	Analytical Solution to the Bungee Jumper Problem	6
1.2	Numerical Solution to the Bungee Jumper Problem	8
2.1	Error Estimates for Iterative Methods	17
2.2	Roundoff or Machine Error	19
2.3	Finite-Difference Approximations of Derivatives	23
3.1	The Graphical Approach	31
3.2	Bisection Method	35
3.3	Bisection Method	38
3.4	The False-Position Method	41
3.5	A Case where Bisection is Preferable to False Position	43
3.6	False Position Method	43
3.7	Newton-Raphson Method	48
3.8	Newton-Raphson Method	48
3.9	Secant Method	50
3.10	Modified Secant Method	52
4.1	Gauss-Seidel Method	61
4.2	Gauss-Seidel Method	63
4.3	Gauss-Seidel Method	66
4.4	Jacobi's Iteration Method	69
4.5	Jacobi's Iteration Method	71
5.1	Determining Polynomial Coefficients with Simultaneous equations	76
5.2	Linear Interpolation	77
5.3	Linear Interpolation	78
5.4	Linear Interpolation	78
5.5	Linear Interpolation	78
5.6	Quadratic Interpolation	80
5.7	Newton Interpolating Polynomial	82
5.8	Lagrange Interpolating Polynomial	85
6.1	Euler's Method	91
6.2	Euler's Method	92
6.3	4 <sup>th</sup> Order Runge-Kutta (RK4) Method	95
6.4	4 <sup>th</sup> Order Runge-Kutta (RK4) Method	98

# CHAPTER 1

## INTRODUCTION TO NUMERICAL ANALYSIS AND ERROR ANALYSIS

### 1.1 Objectives

The objective of this chapter is to:

- Introduce what numerical methods are and how they relate to engineering and scientific problem-solving.
- Learn how mathematical models can be formulated based on scientific principles to simulate the behavior of a simple physical system.
- Understand how numerical methods afford a means to generate solutions in a manner that can be implemented on a digital computer.
- Learn about the different types of numerical methods.

### 1.2 You've Got a Problem

Suppose that a bungee-jumping company hires you (*Bungee jumping is an activity that involves jumping from a tall structure while connected to a large elastic cord*). You're given the task of predicting the velocity of a jumper (Figure 1.1) as a function of time during the free-fall part of the jump. This information will be used as part of a larger analysis to determine the length and required strength of the bungee cord for jumpers of different mass.

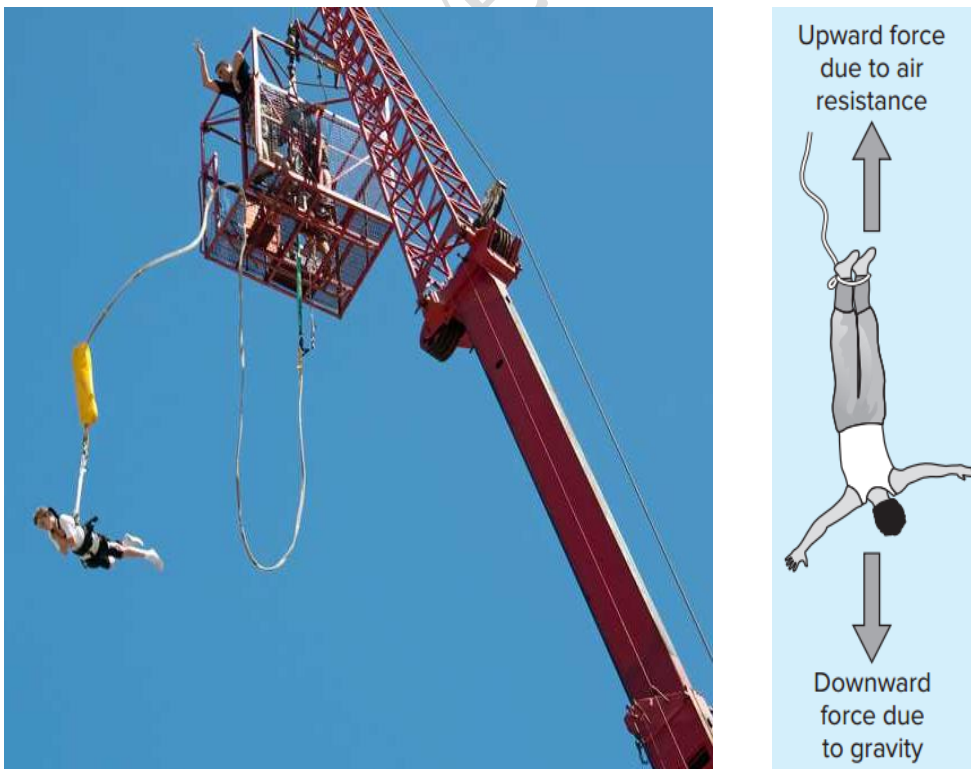


Figure 1.1 Forces Acting on a Free-Falling Bungee Jumper

You know from your knowledge of physics and fluid mechanics; the acceleration should be equal to the ratio of the force to the mass (Newton's second law). Based on this insight and your knowledge, you develop the following mathematical model for the rate of change of velocity with respect to time as shown in Equation (1.1)

$$\frac{dv}{dt} = g - \frac{c_d}{m} v^2 \quad (1.1)$$

where  $v$  = downward vertical velocity (m/s),  $t$  = time (s),  $g$  = the acceleration due to gravity ( $\cong 9.81 \text{ m/s}^2$ ),  $c_d$  = a lumped drag coefficient (kg/m), and  $m$  = the jumper's mass (kg). The drag coefficient is called "lumped" because its magnitude depends on factors such as the jumper's area and the fluid density.

Because this is a differential equation, you know that calculus might be used to obtain an analytical or exact solution  $v$  as a function of  $t$ . However, in the subsequent sections, we will illustrate an alternative solution approach. This will involve developing a computer-oriented numerical or approximate solution.

Aside from showing you how the computer can be used to solve this particular problem, our more general objective will be to illustrate

- i. what numerical methods are and
- ii. how they figure in engineering and scientific problem-solving. In so doing, we will also show how mathematical models figure prominently in the way engineers and scientists use numerical methods in their work.

### 1.3 What is Numerical Analysis?



Numerical analysis is a branch of Mathematics that deals with devising efficient methods for obtaining numerical solutions to difficult Mathematical problems. Most of the Mathematical problems that arise in science and engineering are very hard and sometimes impossible to solve exactly. Thus, an approximation to a difficult Mathematical problem is very important to make it easier to solve. Due to the immense development in computational technology, numerical approximation has become more popular and a modern tool for scientists and engineers. As a result, much scientific software is developed (for instance, Excel, Matlab, Mathematica, Maple, etc.) to handle more difficult problems efficiently and easily. These software contain functions that use standard numerical methods, where a user can pass the required parameters and get the results just by a single command without knowing the details of the numerical method. Thus, one may ask why we need to understand numerical methods when such software are at our hands?



In fact, there is no need for a deeper knowledge of numerical methods and their analysis in most of the cases in order to use some standard softwares as

an end user. However, there are at least seven reasons to gain a basic understanding of the theoretical background of numerical methods.

- i. Learning different numerical methods and their analysis will make a person more familiar with the technique of developing new numerical methods. This is important when the available methods are not enough or not efficient for a specific problem to be solved.
- ii. In many circumstances, one has more methods for a given problem. Hence, choosing an appropriate method is important for producing an accurate result in lesser time.
- iii. With a sound background, one can use methods properly (especially when a method has its own limitations and/or disadvantages in some specific cases) and, most importantly, one can understand what is going wrong when results are not as expected.
- iv. Numerical methods allow the usage “canned” software with insight. During your career, you will invariably have occasion to use commercially available pre-packaged computer programs that involve numerical methods. The intelligent use of these programs is greatly enhanced by an understanding of the basic theory underlying the methods.
- v. Many problems cannot be approached using canned programs. If you are conversant with numerical methods, and are proficient at computer programming, you can design your own programs to solve problems without having to buy or commission expensive software.
- vi. Numerical methods are an efficient vehicle for learning to use computers. Because numerical methods are expressly designed for computer implementation, they are ideal for illustrating the computer’s powers and limitations. After a successfully implementing numerical methods on a computer, and then apply them to solve otherwise intractable problems, you will be provided with a dramatic demonstration of how computers can serve your professional development. At the same time, you will also learn to acknowledge and control the errors of approximation that are part and parcel of large-scale numerical calculations.
- vii. Numerical methods provide a vehicle for you to reinforce your understanding of mathematics. Because one function of numerical methods is to reduce higher mathematics to basic arithmetic operations, they get at the “nuts and bolts” of some otherwise obscure topics. Enhanced understanding and insight can result from this alternative perspective.



Numerical analysis includes three parts. The first part of the subject is about the development of a method to a problem. The second part deals with the analysis of the method, which includes the error analysis and the efficiency analysis.

Error analysis gives us the understanding of how accurate the result will be if we use the method and the efficiency analysis tells us how fast we can compute the result.

The third part of the subject is the development of an efficient algorithm to implement the method as a computer code. A complete knowledge of the subject includes familiarity in all these three parts. Thus, this course is designed to meet this goal.

### 1.3.1 Numerical Methods Covered

Euler's method was chosen for this introductory chapter because it is typical of many other classes of numerical methods. In essence, most consist of recasting mathematical operations into the simple kind of algebraic and logical operations compatible with digital computers. Figure 1.2 summarises the major areas covered in this course.

Chapters 1 and 2 introduces numerical analysis and the concept of errors. One may be surprised to see errors at the initial stage of the course when no methods are introduced. Error analysis is an important concept, which must be understood for the effective use of numerical methods.

Chapter 3 introduces various iterative methods for determining the root of nonlinear equations. As depicted in Figure 1.3, root location involves searching for the zeros of a function. In contrast, optimization involves determining a value or values of an independent variable that correspond to a "best" or optimal value of a function. Thus, as in Figure 1.3, optimization involves identifying maxima and minima. Although somewhat different approaches are used, root location and optimization both typically arise in design contexts.

Chapter 4 is devoted to solving systems of simultaneous linear algebraic equations Figure 1.3. Such systems are similar in spirit to roots of equations in the sense that they are concerned with values that satisfy equations. However, in contrast to satisfying a single equation, a set of values is sought that simultaneously satisfies a set of linear algebraic equations. Such equations arise in a variety of problem contexts and in all disciplines of engineering and science. In particular, they originate in the mathematical modelling of large systems of interconnected elements such as structures, electric circuits, and fluid networks. However, they are also encountered in other areas of numerical methods such as curve fitting and differential equations.

As an engineer or scientist, you will often have occasion to fit curves to data points. The techniques developed for this purpose can be divided into two general categories: regression and interpolation. Interpolation is used where the objective is to determine intermediate values between relatively error-free data points. Such is usually the case for tabulated information. The strategy in such cases is to fit a curve directly through the data points and use the curve to predict the intermediate values is discussed in Chapter 5.

Finally, Chapter 6 focuses on the solution of ordinary differential equations, Figure 1.3. Such equations are of great significance in all areas of engineering and science. This is because many physical laws are couched in terms of the rate of change of a quantity rather than the magnitude of the quantity itself. Examples range from population-forecasting models (rate of change of population) to the acceleration of a falling body (rate of change of velocity).

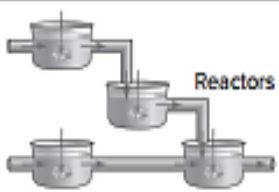

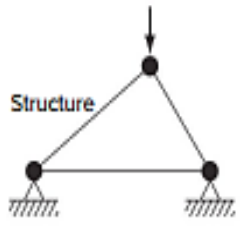
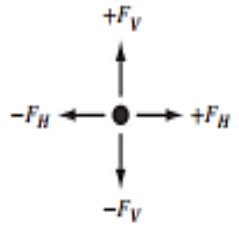
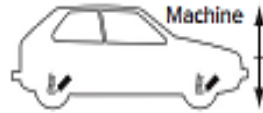
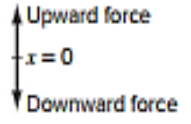
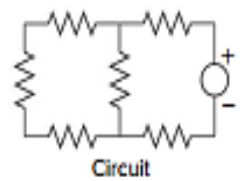
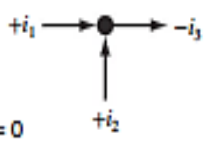
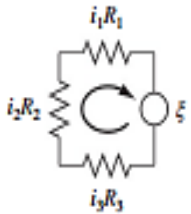
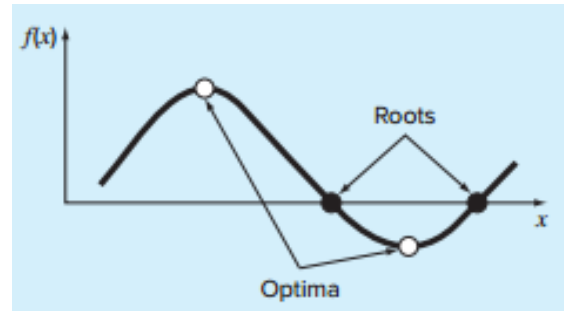
Field	Device	Organizing Principle	Mathematical Expression
Chemical engineering		Conservation of mass	Mass balance:   Over a unit of time period $\Delta \text{mass} = \text{inputs} - \text{outputs}$
Civil engineering		Conservation of momentum	Force balance:   At each node $\Sigma \text{ horizontal forces } (F_H) = 0$ $\Sigma \text{ vertical forces } (F_V) = 0$
Mechanical engineering		Conservation of momentum	Force balance:   $m \frac{d^2x}{dt^2} = \text{downward force} - \text{upward force}$
Electrical engineering		Conservation of charge	Current balance:   For each node $\Sigma \text{ current } (i) = 0$
		Conservation of energy	Voltage balance:   Around each loop $\Sigma \text{ emf's} - \Sigma \text{ voltage drops for resistors} = 0$ $\Sigma \xi - \Sigma iR = 0$

Figure 1.2 Some Engineering Areas of Primary Focus

## Chapter 2: Root of nonlinear equations

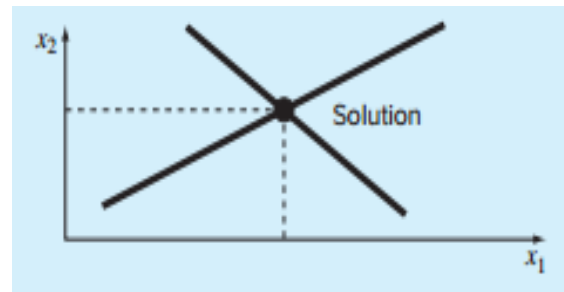
Solve for  $x$  such that  $f(x) = 0$



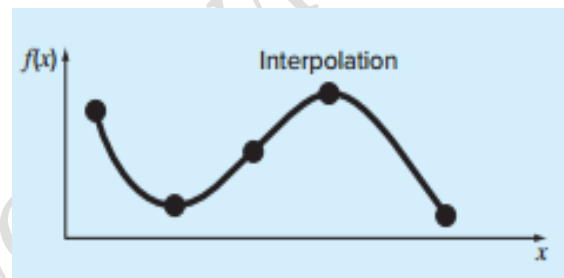
## Chapter 3: Linear algebraic equations

Given the  $a$ 's and the  $b$ 's, solve for the  $x$ 's

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ \vdots & \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned}$$



## Chapter 4: Polynomial interpolation



## Chapter 5: Initial value ODE

Given

$$\frac{dy}{dt} \approx \frac{\Delta y}{\Delta t} = f(t, y)$$

Solve for  $y$  as a function of  $t$

$$y_{i+1} = y_i + f(t_i, y_i)\Delta t$$

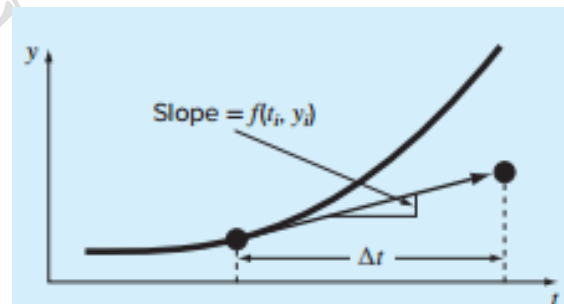


Figure 1.3 Summary of the Numerical Methods Covered

### Example 1.1 Analytical Solution to the Bungee Jumper Problem

A bungee jumper with a mass of 68.1 kg leaps from a stationary hot air balloon. Compute velocity for the first 12 s of free fall. Also determine the terminal velocity that will be attained for an infinitely long cord. Use a drag coefficient of 0.25 kg/m.

#### Solution

Equation (1.1) is a model that relates the acceleration of a falling object to the forces acting on it. It is a differential equation because it is written in terms of the differential rate of change ( $dv/dt$ ) of the variable that we are interested in predicting. However, the exact solution of Equation (1.1) for the velocity of the jumper cannot be obtained using simple algebraic manipulation. Rather, more advanced techniques such as those



of calculus must be applied to obtain an exact or analytical solution. For example, if the jumper is initially at rest ( $v = 0$  at  $t = 0$ ), calculus can be used to solve Equation (1.1) as shown in Equation (1.2)

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) \quad (1.2)$$

Inserting the parameters into Equation (1.2) yields

$$v(t) = \sqrt{\frac{9.81(68.1)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{68.1}} t\right) \quad (1.3)$$

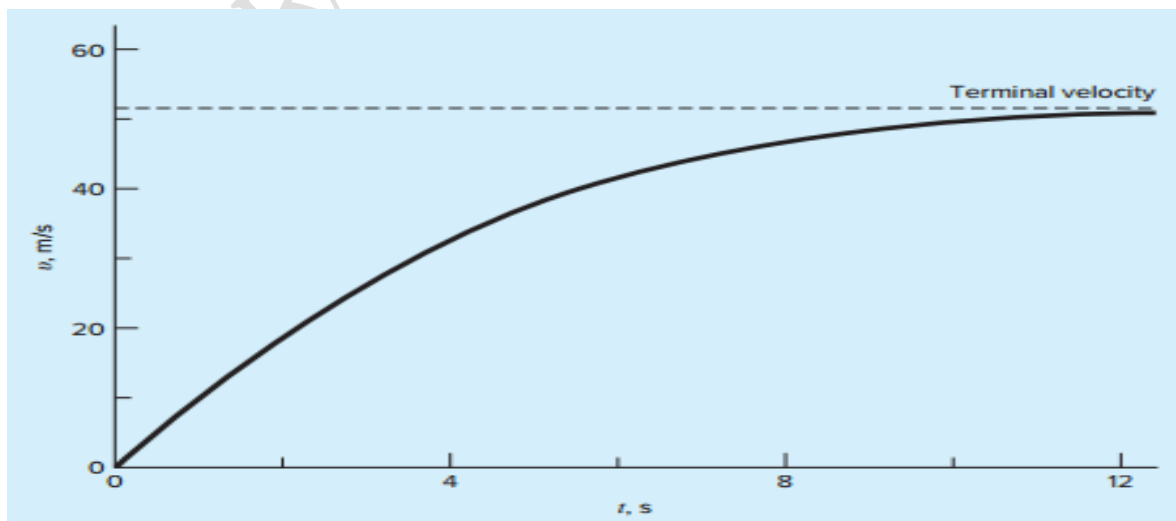
$$= 51.6938 \tanh(0.18977t) \quad (1.4)$$

which can be used to compute *(NB: set calculator to radian when dealing with trigonometric functions - i.e., involving sin, cos, tan; and hyperbolic functions, i.e. sinh, cosh, tanh etc.)*

**Table 1.1 Analytical Solution**

Iteration	$t(s)$	$v(m/s)$
	0	0.0000
1	2	18.7292
2	4	33.1118
3	6	42.0762
4	8	46.9575
5	10	49.4214
6	12	50.6175
$\vdots$	$\vdots$	$\vdots$
$\infty$	$\infty$	51.6938

According to the model, the jumper accelerates rapidly (Table 1.1 and Figure 1.4). A velocity of 49.4214 m/s (about 110 mi/hr) is attained after 10 s. Note also that after a sufficiently long time, a constant velocity, called the terminal velocity, of 51.6983 m/s (115.6 mi/hr) is reached. This velocity is constant because, eventually, the force of gravity will be in balance with the air resistance. Thus, the net force is zero and acceleration has ceased. Velocity increases with time and asymptotically approaches a terminal velocity.



**Figure 1.4 Velocity-Time Plot of Bungee Jumper**



Equation (1.2) is called an **analytical** or **closed-form solution** because it exactly satisfies the original differential equation. Unfortunately, there are many mathematical models that cannot be solved exactly. In many of these cases, the only alternative is to develop a numerical solution that approximates the exact solution.

### Example 1.2 Numerical Solution to the Bungee Jumper Problem

Numerical methods are those in which the mathematical problem is reformulated so it can be solved by arithmetic operations. This can be illustrated for Equation (1.1) by realising that the time rate of change of velocity can be approximated by (Figure 1.5)

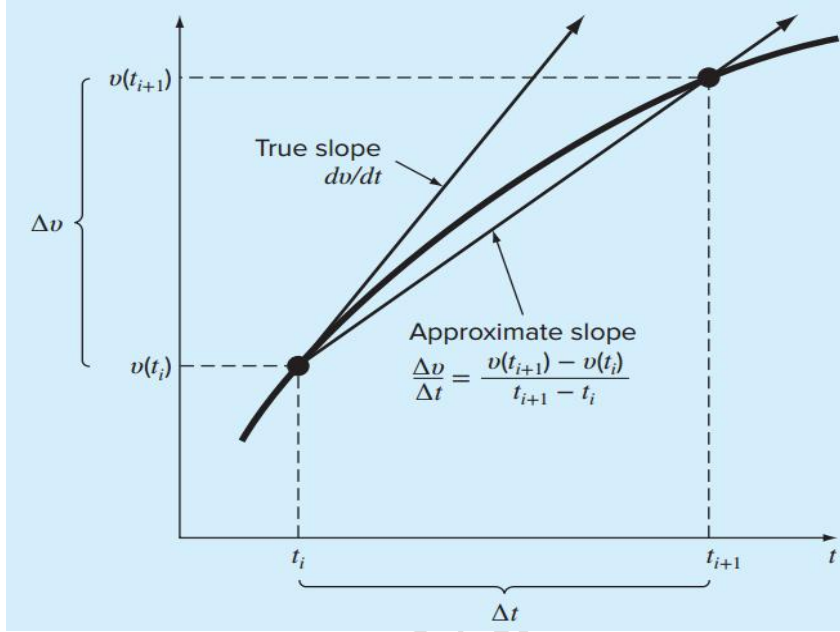


Figure 1.5 Finite Difference to Approximate the First Derivative of  $v$  with respect to  $t$

Hence,

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \quad (1.5)$$

where  $\Delta v$  and  $\Delta t$  are differences in velocity and time computed over finite intervals,  $v(t_i)$  is velocity at an initial time  $t_i$ , and  $v(t_{i+1})$  is velocity at some later time  $t_{i+1}$ . Note that  $\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t}$  is approximate because  $\Delta t$  is finite. Remember from calculus that Equation (1.5) is called a **finite-difference approximation** of the derivative at time  $t_i$ .

Substituting Equation (1.5) into Equation (1.1) yields

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c_d}{m} v(t_i)^2 \quad (1.6)$$

Equation (1.6) can then be rearranged to yield

$$v(t_{i+1}) = v(t_i) + \left[ g - \frac{c_d}{m} v(t_i)^2 \right] (t_{i+1} - t_i) \quad (1.7)$$

Notice that the term in brackets is the right-hand side of the differential equation itself [Equation (1.1)]. That is, it provides a means to compute the rate of change or slope of  $v$ . Thus, the Equation (1.7) can be rewritten more concisely as

$$v(t_{i+1}) = v(t_i) + \left[ g - \frac{c_d}{m} v(t_i)^2 \right] \Delta t \quad (1.8)$$

where the nomenclature  $v_i$  designates velocity at time  $t_i$ , and  $\Delta t = t_{i+1} - t_i$ .

We can now observe that the differential equation from Equation (1.1) has been transformed into an equation that can be used to determine the velocity algebraically at  $t_{i+1}$  using the slope and previous values of  $v$  and  $t$ . If you are given an initial value for velocity at some time  $t_i$ , you can easily compute velocity at a later time  $t_{i+1}$ . This new value of velocity at  $t_{i+1}$  can in turn be employed to extend the computation to velocity at  $t_{i+2}$  and so on. Thus, at any time along the way,

$$\text{New value} = \text{old value} + \text{slope} \times \text{step size} \quad (1.9)$$

This approach is formally called **Euler's method**. We'll discuss it in more detail when we turn to differential equations later in Chapter 6.

### Problem Statement

Performing the same computation as in Example 1.1 but using Equation (1.8) to compute velocity with Euler's method. Employ a step size of 2 seconds for the calculation.

### Solution

At the start of the computation ( $t_0 = 0$ ), the velocity of the jumper is zero. Using this information and the parameter values from Example 1.1, Equation (1.8) can be used to compute velocity at  $t_1 = 2$  s:

$$v(2) = 0 + \left[ 9.81 - \frac{0.25}{68.1} (0)^2 \right] * 2 = 19.62 \text{ m/s} \quad (1.10)$$

For the next interval (from  $t = 2$  to 4 s), the computation is repeated, with the result

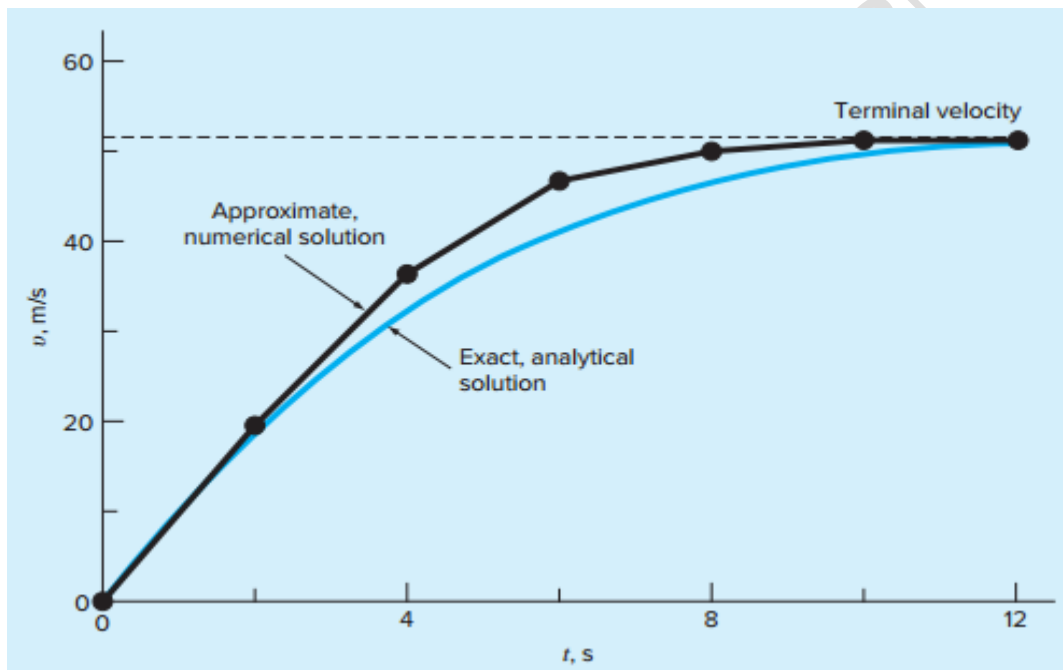
$$v(4) = 19.62 + \left[ 9.81 - \frac{0.25}{68.1} (19.62)^2 \right] * 2 = 36.4137 \text{ m/s} \quad (1.11)$$

The calculation is continued in a similar fashion to obtain additional values:

The results of the numerical solution (Table 1.2) are plotted in Figure 1.6 along with the exact solution. It is observed that the numerical method captures the essential features of the exact solution. However, because we have employed straight-line segments to approximate a continuously curving function, there is some discrepancy between the two results. One way to minimize such discrepancies is to use a smaller step size. For example, applying Equation (1.8) at 1 s intervals results in a smaller error, as the straight-line segments track closer to the true solution.

**Table 1.2 Numerical Solution**

Iteration	$t(s)$	$v(m/s)$
	0	0
1	2	19.6200
2	4	36.4137
3	6	46.2983
4	8	50.1802
5	10	51.3123
6	12	51.6008
$\vdots$	$\vdots$	$\vdots$
$\infty$	$\infty$	51.6938



**Figure 1.6 Comparison of the Numerical and Analytical Solutions for the Bungee Jumper Problem**

Using hand calculations, the effort associated with using smaller and smaller step sizes would make such numerical solutions impractical. However, with the aid of the computer, large numbers of calculations can be performed easily. Thus, you can accurately model the velocity of the jumper without having to solve the differential equation exactly.

As in Example 1.2, a computational price must be paid for a more accurate numerical result. Each halving of the step size to attain more accuracy leads to a doubling of the number of computations. Thus, we see that there is a trade-off between accuracy and computational effort. Such trade-offs play prominently in numerical methods and constitute an important theme of this course.

## 1.4 Try Exercise

1. Explain what numerical analysis is?
2. What are some of the reasons for an engineer to gain a basic understanding of the theoretical background of numerical methods?
3. Numerical analysis is divided into three parts. What are they?
4. Use calculus to verify that Equation (1.2) is a solution of Equation (1.1) for the initial condition  $v(0) = 0$ .
5. Repeat Example 1.2. Compute the velocity to  $t = 12$  s, with a step size of (a) 1 and (b) 0.5 s. Can you make any statement regarding the errors of the calculation based on the results?
6. Newton's law of cooling says that the temperature of a body changes at a rate proportional to the difference between its temperature and that of the surrounding medium (the ambient temperature),

$$\frac{dT}{dt} = -k(T - T_a) \quad (1.12)$$

where  $T$  = the temperature of the body ( $^{\circ}\text{C}$ ),  $t$  = time (min),  $k$  = the proportionality constant (per minute), and  $T_a$  = the ambient temperature ( $^{\circ}\text{C}$ ). Suppose that a cup of coffee originally has a temperature of  $70^{\circ}\text{C}$ . Use Euler's method to compute the temperature from  $t = 0$  to 20 min using a step size of 2 min if  $T_a = 20^{\circ}\text{C}$  and  $k = 0.019/\text{min}$ .

7. A storage tank (Figure 1.7) contains a liquid at depth  $y$  where  $y = 0$  when the tank is half full. Liquid is withdrawn at a constant flow rate  $Q$  to meet demands. The contents are resupplied at a sinusoidal rate  $3Q \sin^2(t)$ . The mathematical expression for this system can be written as

$$\underbrace{\frac{d(Ay)}{dt}}_{\text{change in volume}} = \underbrace{3Q \sin^2(t)}_{\text{inflow}} - \underbrace{Q}_{\text{outflow}}$$

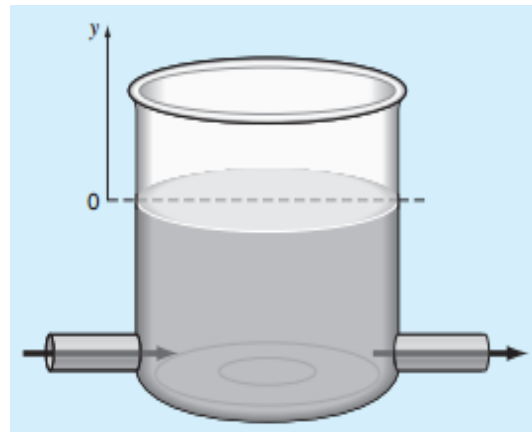


Figure 1.7 Storage Tank

However, since the surface area  $A$  is constant

$$\frac{dy}{dt} = 3 \frac{Q}{A} \sin^2(t) - \frac{Q}{A} \quad (1.13)$$

Use Euler's method to solve for the depth  $y$  from  $t = 0$  to 5 d with a step size of 0.5 d. The parameter values are  $A = 1250 \text{ m}^2$  and  $Q = 450 \text{ m}^3/\text{d}$ . Assume that the initial condition is  $y = 0$ .

8. For the same storage tank described in Question 7, suppose that the outflow is not constant but rather depends on the depth. For this case, the differential equation for depth can be written as

$$\frac{dy}{dt} = 3 \frac{Q}{A} \sin^2(t) - \frac{\alpha(1+y)^{1.5}}{A} \quad (1.14)$$

Use Euler's method to solve for the depth  $y$  from  $t = 0$  to 10 d with a step size of 0.5 d. The parameter values are  $A = 1250 \text{ m}^2$ ,  $Q = 450 \text{ m}^3/\text{d}$ , and  $\alpha = 150$ . Assume that the initial condition is  $y = 0$ .

9. You are working as a crime scene investigator and must predict the temperature of a homicide victim over a 5-hr period.
- Use Newton's law of cooling (Equation (1.12)) and Euler's method to compute the victim's body temperature for the 5-hr period using values of  $k = 0.12/\text{hr}$  and  $\Delta t = 0.5 \text{ hr}$ . Assume that the victim's body temperature at the time of death was  $37^\circ\text{C}$ , and that the room temperature was at a constant value of  $20^\circ\text{C}$  over the 5-hr period.
  - Further investigation reveals that the room temperature had actually dropped linearly from 20 to  $10^\circ\text{C}$  over the 5-hr period. Repeat the same calculation as in (a) but incorporate this new information.
  - Compare the results from (a) and (b) by plotting them on the same graph.

## CHAPTER 2

### ERROR ANALYSIS (ROUND OFF AND TRUNCATION ERRORS)

#### 2.1 Objectives

The primary objective of this chapter is to acquaint you with the major sources of errors involved in numerical methods. Specific objectives and topics covered are

- Understanding the distinction between accuracy and precision.
- Learning how to quantify error.
- Learning how error estimates can be used to decide when to terminate an iterative calculation.
- Understanding how roundoff errors occur because digital computers have a limited ability to represent numbers.
- Understanding why floating-point numbers have limits on their range and precision.
- Recognizing that truncation errors occur when exact mathematical formulations are represented by approximations.
- Knowing how to use the Taylor series to estimate truncation errors.
- Understanding how to write forward, backward, and centered finite-difference approximations of first derivatives.
- Recognizing that efforts to minimize truncation errors can sometimes increase roundoff errors

#### 2.2 You've Got a Problem

In Chapter 1, you developed a numerical model for the velocity of a bungee jumper. To solve the problem with a computer, you had to approximate the derivative of velocity with a finite difference

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \quad (2.1)$$

Thus, the resulting solution is not exact, that is, it has error.



In addition, the computer you use to obtain the solution is also an imperfect tool. Because it is a digital device, the computer is limited in its ability to represent the magnitudes and precision of numbers. Consequently, the machine itself yields results that contain error.

So, both mathematical approximations and digital computers cause our resulting model prediction to be uncertain. The question now is: How do you deal with such

uncertainty? In particular, is it possible to understand, quantify, and control such errors in order to obtain acceptable results? This section introduces you to some approaches and concepts that engineers, and scientists use to deal with this dilemma.

## 2.3 Errors



Engineers and scientists constantly find themselves having to accomplish objectives based on uncertain information. Although perfection is a laudable goal, it is rarely if ever attained. For example, despite the fact that the model developed from Newton's second law is an excellent approximation, it would never in practice exactly predict the jumper's fall. A variety of factors such as winds and slight variations in air resistance would result in deviations from the prediction. If these deviations are systematically high or low, then we might need to develop a new model. However, if they are randomly distributed and tightly grouped around the prediction, then the deviations might be considered negligible and the model deemed adequate. Numerical approximations also introduce similar discrepancies into the analysis.

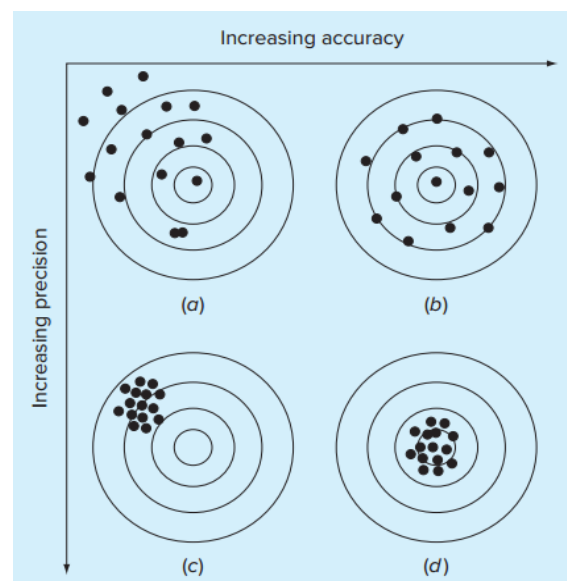
This section covers basic topics related to the identification, quantification, and minimisation of these errors. This is followed by dealing with the two major forms of numerical error: roundoff error (due to computer approximations) and truncation error (due to mathematical approximations). We also describe how strategies to reduce truncation error sometimes increase roundoff.

## 2.4 Accuracy and Precision



The errors associated with both calculations and measurements can be characterized with regard to their accuracy and precision. **Accuracy** refers to how closely a computed or measured value agrees with the true value. **Precision** refers to how closely individual computed or measured values agree with each other.

These concepts can be illustrated graphically using an analogy from target practice. The bullet holes on each target in Figure 2.1 can be thought of as the predictions of a numerical technique, whereas the bull's-eye represents the truth. **Inaccuracy** (also called **bias**) is defined as systematic deviation from the truth. Thus, although the shots in Figure 2.1c are more tightly grouped than in Figure 2.1a, the two cases are equally biased because they are both centred on



**Figure 2.1** (a) Inaccurate and Imprecise, (b) Accurate and Imprecise, (c) Inaccurate and Precise (d) Accurate and Precise



the upper left quadrant of the target. **Imprecision** (also called **uncertainty**), on the other hand, refers to the magnitude of the scatter. Therefore, although Figure 2.1b and Figure 2.1d are equally accurate (i.e., centred on the bull's-eye), the latter is more precise because the shots are tightly grouped.



Numerical methods should be sufficiently accurate or unbiased to meet the requirements of a particular problem. They also should be precise enough for adequate design. In this material, the term **error** is used to represent both the inaccuracy and imprecision of our predictions.

## 2.5 Numerical Errors



Numerical errors arise from the use of approximations to represent exact mathematical operations and quantities. For such errors, the relationship between the exact, or true result and the approximation can be formulated as

$$\text{True value} = \text{approximation} + \text{error} \quad (2.2)$$

By rearranging Equation (2.2), we find that the numerical error is equal to the discrepancy between the truth and the approximation, as in

$$E_t = \text{true value} - \text{approximation} \quad (2.3)$$

where  $E_t$  is used to designate the exact value of the error. The subscript  $t$  is included to designate that this is the “true” error. Note that the true error is commonly expressed as an absolute value and referred to as the absolute error.

One way to account for the magnitudes of the quantities being evaluated is to normalize the error to the true value, as in

$$\text{True fractional relative error} = \frac{\text{true value} - \text{approximation}}{\text{true value}} \quad (2.4)$$

The relative error can also be multiplied by 100% to express it as

$$\varepsilon_t = \left( \frac{\text{true value} - \text{approximation}}{\text{true value}} \right) 100 \quad (2.5)$$

where  $\varepsilon_t$  is the true percent relative error.

For example, suppose that you have the task of measuring the lengths of a bridge and a gold bar and come up with 9999 and 9 cm, respectively. If the true values are 10,000 and 10 cm, respectively, the error in both cases is 1 cm. However, their percent relative errors can be computed using Equation (2.5) as 0.01% and 10%, respectively. Thus, although both measurements have an absolute error of 1 cm, the relative error for the gold bar is much greater. We would probably conclude that we have done an adequate job of measuring the bridge, whereas our estimate for the gold bar may be questionable.

Notice that for Equations (2.3) and (2.5),  $E_t$  and  $\varepsilon_t$  are subscripted with a  $t$  to signify that the error is based on the true value. For the example of the bridge and the gold bar, we were provided with the true value. However, in actual situations such information is rarely available. For numerical methods, the true value will only be known when we deal with functions that can be solved analytically. Such will typically be the case when we investigate the theoretical behaviour of a particular technique for simple systems. However, in real-world applications, we will obviously not know the true answer **a priori**. For these situations, an alternative is to normalize the error using the best available estimate of the true value, that is, to the approximation itself, as in

$$\varepsilon_\alpha = \left( \frac{\text{approximation error}}{\text{approximation}} \right) 100\% \quad (2.6)$$

where  $\alpha$  signifies that the error is normalized to an approximate value. Note also that for real-world applications, Equation (2.3) cannot be used to calculate the error term in the numerator of Equation (2.6). One of the challenges of numerical methods is to determine error estimates in the absence of knowledge regarding the true value.

For example, certain numerical methods use iteration to compute answers. In such cases, a present approximation is made on the basis of a previous approximation. This process is performed repeatedly, or iteratively, to successively compute (hopefully) better and better approximations. For such cases, the error is often estimated as the difference between the previous and present approximations. Thus, percent relative error is determined using

$$\varepsilon_\alpha = \left( \frac{\text{present approximation} - \text{previous approximation}}{\text{present approximation}} \right) 100\% \quad (2.7)$$



The signs of Equations (2.3) through to (2.7) may be either positive or negative. If the approximation is greater than the true value (or the previous approximation is greater than the present approximation), the error is negative; if the approximation is less than the true value, the error is positive. Also, for Equations (2.4) to (2.7), the denominator may be less than zero, which can also lead to a negative error. Often, when performing computations, we may not be concerned with the sign of the error but are interested in whether the absolute value of the percent relative error is lower than a prespecified tolerance,  $Tol$ . Therefore, it is often useful to employ the absolute value of Equation (2.7). For such cases, the computation is repeated until

$$|\varepsilon_\alpha| < Tol \quad (2.8)$$

This relationship is referred to as a stopping criterion. If it is satisfied, our result is assumed to be within the prespecified acceptable level,  $Tol$ .

It is also convenient to relate these errors to the number of significant figures in the approximation. It can be shown (Scarborough, 1966) that if the following criterion is met, we can be assured that the result is correct to **at least**  $n$  significant figures.

$$Tol = (0.5 \times 10^{2-n})\% \quad (2.9)$$

### Example 2.1 Error Estimates for Iterative Methods

In mathematics, functions can often be represented by infinite series. For example, the Maclaurin series expansion (exponential function) can be computed using

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} \quad (2.10)$$

Thus, as more terms are added in sequence, the approximation becomes a better and better estimate of the true value of  $e^x$ . Starting with the first term,  $e^x = 1$ , add terms one at a time in order to estimate  $e^{0.5}$ . After each new term is added, compute the true and approximate percent relative errors with Equations (2.5) and (2.7), respectively. Note that the true value is  $e^{0.5} = 1.648721\dots$ . Add terms until the absolute value of the approximate error estimate  $\varepsilon_\alpha$  falls below a prespecified error criterion  $Tol$  conforming to three significant figures.

### Solution

First, Equation (2.9) can be employed to determine the error criterion,  $Tol$ , that ensures a result that is correct to at least three significant figures:

$$Tol = (0.5 \times 10^{2-3})\% = 0.05\% \quad (2.11)$$

Thus, we will add terms to the series until  $\varepsilon_\alpha$  falls below 0.05%.

The first estimate is simply equal to Equation (2.10) with a single term. Thus, the first estimate is equal to 1. The second estimate is then generated by adding the second term as in

$$e^x = 1 + x$$

for  $x = 0.5$

$$e^{0.5} = 1 + 0.5 = 1.5$$

Using Equation (2.5), this represents a true percent relative error of

$$\varepsilon_t = \left| \frac{1.648721 - 1.5}{1.648721} \right| \times 100\% = 9.02\%$$

Equation (2.7) can be used to determine an approximate estimate of the error, as in

$$\varepsilon_\alpha = \left| \frac{1.5 - 1}{1.5} \right| \times 100\% = 33.3\%$$

Because  $\varepsilon_\alpha$  is not less than the required value of  $Tol$ , we would continue the computation by adding another term,  $\frac{x^2}{2!}$ , and repeating the error calculations. The process is continued until  $|\varepsilon_\alpha| < Tol$ . The entire computation can be summarized as

**Table 2.1 Summary of Computation**

Iteration	Result	$\varepsilon_t$ (%)	$\varepsilon_\alpha$ (%)
1	1	39.3	
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645833333	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	0.0158

Thus, after six terms are included, the approximate error falls below  $Tol = 0.05\%$ , and the computation is terminated. However, notice that, rather than three significant figures, the result is accurate to five! This is because, for this case, both Equations (2.5) and (2.7) are conservative. That is, they ensure that the result is at least as good as they specify. Although, this is not always the case for Equation (2.5), it is true most of the time.

### 2.5.1 Sources of Numerical Errors



Error appears in different ways. On one hand, any measurement is subject to it (this why any measuring device is sold with an estimated margin of precision); this is intrinsic to Nature and one can only take it into account and try to assess its magnitude (give a bound). On the other hand, computations performed in finite precision arithmetic both propagate these errors and give rise to new ones precisely because the quantity of available digits is finite.

The following are some of the sources of error:

- **Measurement error:** already mentioned. This is unavoidable.
- **Truncation error:** happens whenever a number (datum or the result of a computation) has more digits than available and some of them must be “forgotten”.
- **Rounding error:** takes place when a number is rounded to a specified precision.
- **Accumulation error:** which appears when accumulating (with additions, essentially) small errors of the same sign a lot of times.

All the above errors may take place when working with finite arithmetic. Numerically computed solutions are subject to certain errors.

## 2.6 Types of Errors

Mainly there are two types of errors. They are roundoff errors and errors due to truncation.

### 2.6.1 Roundoff or Machine Error



Roundoff errors are errors arising from the process of rounding off during computation. These are also called chopping, i.e. discarding all decimals from some decimals on. It also arises because digital computers cannot represent some quantities exactly. They are important to engineering and scientific problem solving because they can lead to erroneous results. In certain cases, they can actually lead to a calculation going unstable and yielding obviously erroneous results. Such calculations are said to be **ill-conditioned**. Worse still, they can lead to subtler discrepancies that are difficult to detect.

There are two major facets of roundoff errors involved in numerical calculations:

- i. Digital computers have magnitude and precision limits on their ability to represent numbers.
- ii. Certain numerical manipulations are highly sensitive to roundoff errors. This can result from both mathematical considerations as well as from the way in which computers perform arithmetic operations.

#### Example 2.2 Roundoff or Machine Error

The expression  $10^{15} * (\pi - 3.14159265359) = -206.7615373566167...$

#### Roundoff rule to discard the $(k+1)^{th}$ and all subsequent decimals

- **Rounding down** If the number at  $(k+1)^{th}$  decimal to be discarded is less than half a unit in the  $k^{th}$  place, leave the  $k^{th}$  decimal unchanged. For example, rounding of 8.43 to 1 decimal gives 8.4 and rounding of 6.281 to 2 decimal places gives 6.28.
- **Rounding up** If the number at  $(k+1)^{th}$  decimal to be discarded is greater than half a unit in the  $k^{th}$  place, add 1 to the  $k^{th}$  decimal. For example, rounding of 8.48 to 1 decimal gives 8.5 and rounding of 6.277 to 2 decimals gives 6.28.
- If it is exactly half a unit, round off to the nearest even decimal. For example, rounding off 8.44 and 8.55 to 1 decimal gives 8.4 and 8.6 respectively. Rounding off 6.265 and 6.275 to 2 decimals gives 6.27 and 6.28 respectively.

### 2.6.2 Truncation Errors



These are errors corresponding to the fact that a finite (or infinite) sequence of computational steps necessary to produce an exact result is “truncated” prematurely after a certain number of steps. Also, truncation errors result from using an

approximation in place of an exact mathematical procedure. For example, in Chapter 1 we approximated the derivative of velocity of a bungee jumper by a finite-difference equation of the form [Equation (1.5)]

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \quad (2.12)$$

A truncation error was introduced into the numerical solution because the difference equation only approximates the true value of the derivative (recall Figure 1.5).

### *Using the Taylor Series to Estimate Truncation Errors*

Although the Taylor series will be extremely useful in estimating truncation errors throughout this lecture material, it may not be clear to you how the expansion can actually be applied to numerical methods. In fact, we have already done so in our example of the bungee jumper. Recall that the objective of Example 1.1 was to predict velocity as a function of time. That is, we were interested in determining  $v(t)$  which can be expanded in a Taylor series as

$$v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + \frac{v''(t_i)}{2!}(t_{i+1} - t_i)^2 + \dots + R_n \quad (2.13)$$

where  $R_n$ , a remainder term is also included to account for all terms from  $n+1$  to infinity:

$$R_n = \frac{v^{(n+1)}(\xi)}{(n+1)!} h^{n+1} \quad (2.14)$$

where the subscript  $n$  implies that this is the remainder for the  $n^{\text{th}}$ -order approximation and  $\xi$  is a value of  $t$  that lies somewhere between  $t_i$  and  $t_{i+1}$ .

Now let us truncate the series [Equation (2.14)] after the first derivative term:

$$v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + R_1 \quad (2.15)$$

Equation (2.15) can be solved for

$$v'(t_i) = \underbrace{\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}}_{\text{First-order approximation}} - \underbrace{\frac{R_1}{t_{i+1} - t_i}}_{\text{Truncation error}} \quad (2.16)$$

The first part of Equation (2.16) is exactly the same relationship that was used to approximate the derivative in Example 1.2 [Equation (1.5)]. However, because of the Taylor series approach, we have now obtained an estimate of the truncation error associated with this approximation of the derivative. Now representing the second part of Equation (2.16) as

$$\frac{R_1}{t_{i+1} - t_i} = O(t_{i+1} - t_i) \quad (2.17)$$

Thus, the estimate of the derivative of the first part of Equation (2.16) has a truncation error of order  $t_{i+1} - t_i$ . In other words, the error of our derivative approximation should be proportional to the step size. Consequently, if we halve the step size, we would expect to halve the error of the derivative.

## 2.7 Numerical Differentiation

Equation (2.16) is given a formal label in numerical methods as a **finite difference**. It can be represented generally as Equation (2.18)

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} + O(x_{i+1} - x_i) \quad (2.18)$$

or

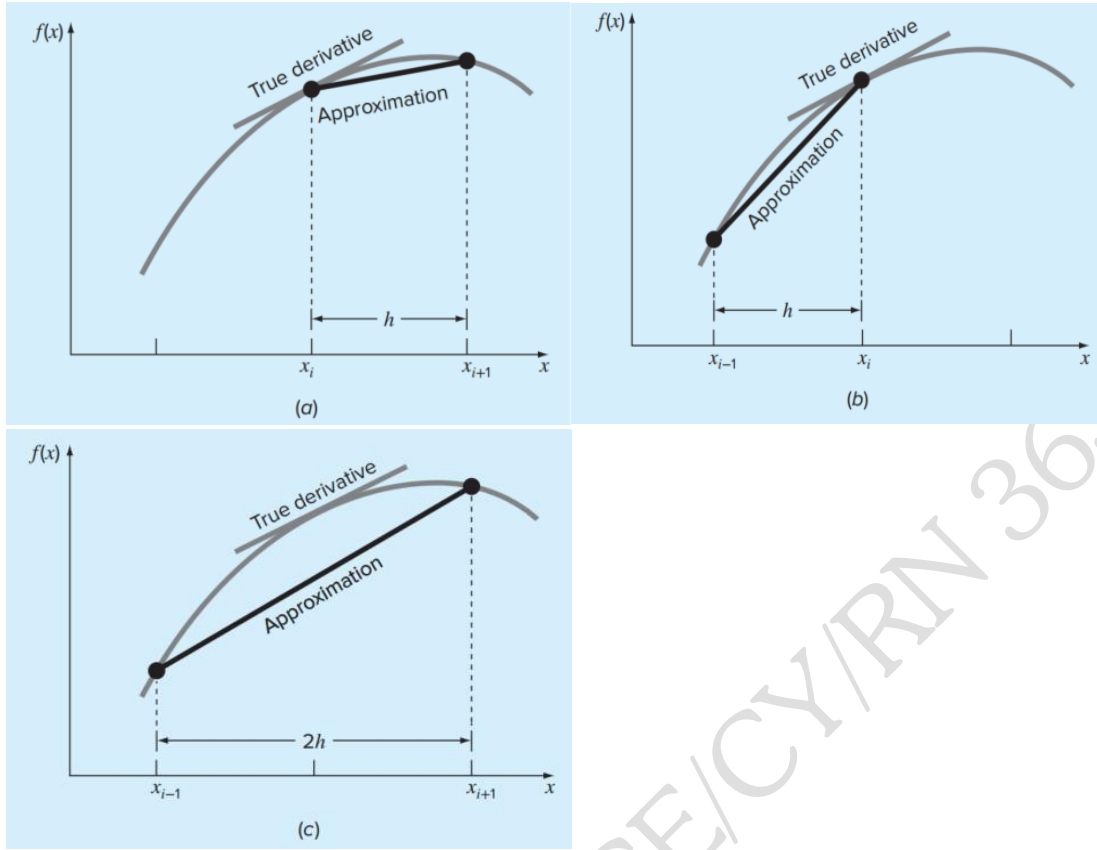
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \quad (2.19)$$

where  $h = x_{i+1} - x_i$  is called the step size, that is, the length of the interval over which the approximation is made. It is termed a “forward” difference because it utilizes data at  $i$  and  $i+1$  to estimate the derivative (Figure 2.2a).

This forward difference is but one of many that can be developed from the Taylor series to approximate derivatives numerically. For example, backward and centred difference approximations of the first derivative can be developed in a fashion similar to the derivation of Equation (2.16).

The former utilizes values at  $x_{i-1}$  and  $x_i$  (Figure 2.2b), whereas the latter uses values that are equally spaced around the point at which the derivative is estimated (Figure 2.2c). More accurate approximations of the first derivative can be developed by including higher-order terms of the Taylor series.

Finally, all the foregoing versions can also be developed for second, third, and higher derivatives. The following sections provide brief summaries illustrating how some of these cases are derived.



**Figure 2.2 (a) Forward, (b) Backward, and (c) Centred Finite-Difference Approximations of the First Derivative**

### 2.7.1 Backward Difference Approximation of the First Derivative

The Taylor series can be expanded backward to calculate a previous value on the basis of a present value, as in

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \dots \quad (2.20)$$

Truncating Equation (2.20) after the first derivative and rearranging yields

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{h} \quad (2.21)$$

where the error is  $O(h)$ .

### 2.7.2 Centred Difference Approximation of the First Derivative

A third way to approximate the first derivative is to subtract Equation (2.20) from the forward Taylor series expansion

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \dots \quad (2.22)$$

to yield



$$f(x_{i+1}) = f(x_{i-1}) + 2f'(x_i)h + 2\frac{f^{(3)}(x_i)}{3!}h^3 + \dots \quad (2.23)$$

which can be rearranged as

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1}))}{2h} - \frac{f^{(3)}(x_i)}{6}h^3 + \dots \quad (2.24)$$

or

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1}))}{2h} - O(h^2) \quad (2.25)$$

Equation (2.25) is a centred finite difference representation of the first derivative. Notice that the truncation error is of the order of  $h^2$  in contrast to the forward and backward approximations that were of the order of  $h$ . Consequently, the Taylor series analysis yields the practical information that the centred difference is a more accurate representation of the derivative (Figure 1.5c). For example, if we halve the step size using a forward or backward difference, we would approximately halve the truncation error, whereas for the central difference, the error would be quartered.

### Example 2.3 Finite-Difference Approximations of Derivatives

Use forward and backward difference approximations of  $O(h)$  and a centred difference approximation of  $O(h^2)$  to estimate the first derivative of

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

at  $x = 0.5$  using a step size  $h = 0.5$ . Repeat the computation using  $h = 0.25$ . Note that the derivative can be calculated directly as

$$f'(x) = -0.4x^3 - 0.45x^2 - 1.0x - 0.25$$

and can be used to compute the true value as  $f'(0.5) = -0.9125$ .

#### Solution

For  $h = 0.5$ , the function can be employed to determine

$$x_{i-1} = 0 \quad f(x_{i-1}) = 1.2$$

$$x_i = 0.5 \quad f(x_i) = 0.925$$

$$x_{i+1} = 1.0 \quad f(x_{i+1}) = 0.2$$

These values can be used to compute the forward difference [Equation (2.19)],

$$f'(0.5) \cong \frac{0.2 - 0.925}{0.5} = -1.45 \quad |\varepsilon_t| = 58.9\%$$

The backward difference [Equation (2.21)],

$$f'(0.5) \cong \frac{0.925 - 1.2}{0.5} = -0.55 \quad |\varepsilon_t| = 39.7\%$$

The centred difference [Equation (2.25)],

$$f'(0.5) \cong \frac{0.2 - 1.2}{1.0} = -1.0 \quad |\varepsilon_t| = 9.6\%$$

For  $h = 0.25$ ,

$$x_{i-1} = 0.25 \quad f(x_{i-1}) = 1.10351563$$

$$x_i = 0.5 \quad f(x_i) = 0.925$$

$$x_{i+1} = 0.75 \quad f(x_{i+1}) = 0.63632813$$

which can be used to compute the forward difference,

$$f'(0.5) \cong \frac{0.63632813 - 0.925}{0.25} = -1.155 \quad |\varepsilon_t| = 26.5\%$$

The backward difference,

$$f'(0.5) \cong \frac{0.925 - 1.10351563}{0.25} = -0.714 \quad |\varepsilon_t| = 21.7\%$$

and the centred difference,

$$f'(0.5) \cong \frac{0.63632813 - 1.10351563}{0.5} = -0.934 \quad |\varepsilon_t| = 2.4\%$$

For both step sizes, the centred difference approximation is more accurate than forward or backward differences. Also, as predicted by the Taylor series analysis, halving the step size approximately halves the error of the backward and forward differences and quarters the error of the centred difference.

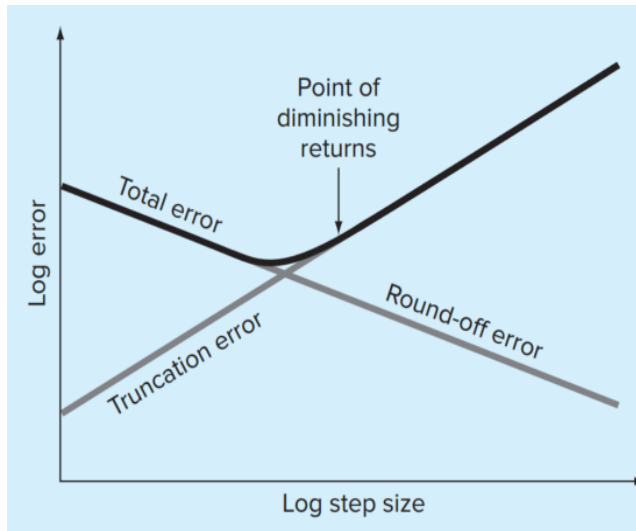
Besides first derivatives, the Taylor series expansion can be used to derive numerical estimates of higher derivatives (**Assignment**).

## 2.8 Total Numerical Error



The total numerical error is the summation of the truncation and roundoff errors. In general, the only way to minimize roundoff errors is to increase the number of significant figures of the computer. Further, we have noted that roundoff error may increase due to subtractive cancellation or due to an increase in the number of computations in an analysis.

In contrast, Example 2.3 demonstrated that the truncation error can be reduced by decreasing the step size. Because a decrease in step size can lead to subtractive cancellation or to an increase in computations, the truncation errors are decreased as the roundoff errors are increased.



**Figure 2.3 Trade-Off between Roundoff and Truncation Error**

Therefore, we are faced by the following dilemma: The strategy for decreasing one component of the total error leads to an increase of the other component. In a computation, we could conceivably decrease the step size to minimize truncation errors only to discover that in doing so, the roundoff error begins to dominate the solution and the total error grows! Thus, our remedy becomes our problem (Figure 2.3). One challenge that we face is to determine an appropriate step size

for a particular computation. We would like to choose a large step size to decrease the amount of calculations and roundoff errors without incurring the penalty of a large truncation error. If the total error is as shown in Figure 2.3, the challenge is to identify the point of diminishing returns where roundoff error begins to negate the benefits of step-size reduction.

When using MATLAB, such situations are relatively uncommon because of its 15 to 16-digit precision. Nevertheless, they sometimes do occur and suggest a sort of “numerical uncertainty principle” that places an absolute limit on the accuracy that may be obtained using certain computerized numerical methods.

### 2.8.1 Control of Numerical Errors

For most practical cases, we do not know the exact error associated with numerical methods. The exception, of course, is when we know the exact solution, which makes our numerical approximations unnecessary. Therefore, for most engineering and scientific applications we must settle for some estimate of the error in our calculations.



There are no systematic and general approaches to evaluating numerical errors for all problems. In many cases error estimates are based on the experience and judgment of the engineer or scientist.

- One should be prepared to perform numerical experiments to increase your awareness of computational errors and possible ill-conditioned problems. Such experiments may involve repeating the computations with a different step size or method and comparing the results.
- One may employ sensitivity analysis to see how our solution changes when we change model parameters or input values.

- One may want to try different numerical algorithms that have different theoretical foundations, are based on different computational strategies, or have different convergence properties and stability characteristics.
- When the results of numerical computations are extremely critical and may involve loss of human life or have severe economic ramifications, it is appropriate to take special precautions. This may involve the use of two or more independent groups to solve the same problem so that their results can be compared.

The roles of errors will be a topic of concern and analysis in all sections of this lecture material.

## 2.9 Try Exercises

1. Define the following terms
 

(a) Truncated Error	(c) Relative Error
(b) Roundoff Error	(d) Absolute Error
2. Determine the absolute and relative errors involved if  $x = 2/3$  is represented in normalised decimal form with 6 digits by
 

(a) round-off	(b) truncation
---------------	----------------
3. If  $x = 3.536$ , determine the absolute error and relative error when  $x$  is rounded
4. If the number  $x = 57.46235$  is rounded off to four significant figures, find the absolute error, relative error and the percentage relative error.
5. If the approximate value of  $\pi = \frac{22}{7}$  is 3.14, determine the absolute error, relative error and relative percentage error.
6. Determine the true error and true percentage relative error for the measured length of a track is approximated by 9999 cm and the true value is 10,000 cm.
7. The following infinite series can be used to approximate  $e^x$ :
 
$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$$
  - (a) Prove that this Maclaurin series expansion is a special case of the Taylor series expansion with  $x_i = 0$  and  $h = x$ .
  - (b) Use the Taylor series to estimate  $f(x) = e^{-x}$  at  $x_{i+1} = 1$  for  $x_i = 0.25$ . Employ the zero, first, second, and third order versions and compute the  $|\varepsilon_i|$  for each case.
8. The Maclaurin series expansion for  $\cos(x)$  is

$$\cos(x) = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots$$

Starting with the simplest version,  $\cos(x) = 1$ , add terms one at a time to estimate  $\cos(\pi/3)$ . After each new term is added, compute the true and approximate percent relative errors. Use your calculator or MATLAB to determine the true value. Add terms until the absolute value of the approximate error estimate falls below an error criterion conforming to two significant figures.

9. Use forward and backward difference approximations of  $O(h)$  and a centred difference approximation of  $O(h^2)$  to estimate the first derivative of the function  $f(x) = 25x^3 - 6x^2 + 7x - 88$ .

Evaluate the derivative at  $x = 2$  using a step size of  $h = 0.25$ . Compare your results with the true value of the derivative. Interpret your results on the basis of the remainder term of the Taylor series expansion.

10. Consider the function  $f(x) = x^3 - 2x + 4$  on the interval  $[-2, 2]$  with  $h = 0.25$ . Use the forward, backward, and centred finite difference approximations for the first and second derivatives so as to graphically illustrate which approximation is most accurate. Graph all three first derivative finite difference approximations

## CHAPTER 3

### NUMERICAL SOLUTION TO NONLINEAR EQUATIONS

#### 3.1 Introduction

Years ago, you learned to use the quadratic formula

$$= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (3.1)$$

to solve

$$f(x) = ax^2 + bx + c = 0 \quad (3.2)$$

The values calculated with Equation (3.1) are called the “roots” of Equation (3.2). They represent the values of  $x$  that make Equation (3.2) equal to zero. For this reason, roots are sometimes called the zeros of the equation.

Although the quadratic formula is handy for solving Equation (3.2), there are many other functions for which the root cannot be determined so easily. Before the beginning of digital computers, there were several ways to solve for the roots of such equations. For some cases, the roots could be obtained by direct methods, as with Equation (3.1)

Although there were equations like this that could be solved directly, there were many more that could not. In such instances, the only alternative is an approximate solution technique.

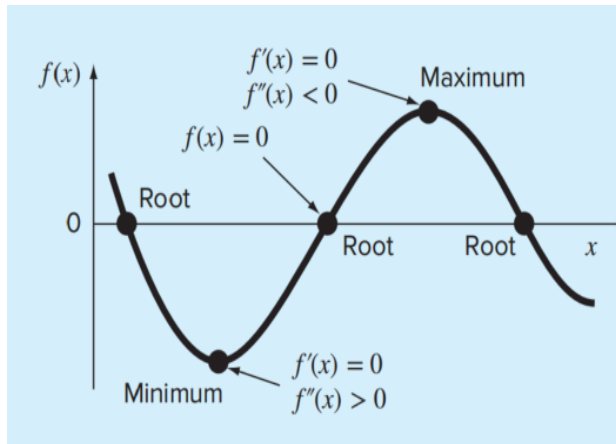
One method to obtain an approximate solution is to plot the function and determine where it crosses the  $x$  axis. This point, which represents the  $x$  value for which  $f(x) = 0$ , is the root. Although graphical methods are useful for obtaining rough estimates of roots, they are limited because of their lack of precision.

An alternative approach is to use trial and error. This “technique” consists of guessing a value of  $x$  and evaluating whether  $f(x)$  is zero. If not (as is almost always the case), another guess is made, and  $f(x)$  is again evaluated to determine whether the new value provides a better estimate of the root. The process is repeated until a guess results in an  $f(x)$  that is close to zero.



Such haphazard methods are obviously inefficient and inadequate for the requirements of engineering and science practice. Numerical methods present alternatives that are also approximate but employ systematic strategies to narrow in on the true root. As elaborated in this chapter, the combination of these systematic methods and computers makes the solution of most applied roots-of-equations problems a simple and efficient task.

Besides roots, another feature of interest to engineers and scientists are a function's minimum and maximum values. The determination of such optimal values is referred



**Figure 3.1 Difference b/n Roots and Optima**

root-location methods we just discussed. That is, both involve guessing and searching for a location on a function. The fundamental difference between the two types of problems is illustrated in Figure 3.1. Root location involves searching for the location where the function equals zero. In contrast, optimization involves searching for the function's extreme points.

## 3.2 Roots: Bracketing Methods

### 3.2.1 Objectives

The primary objective of this section is to acquaint you with bracketing methods for finding the root of a single nonlinear equation. Specific objectives and topics covered are

- Understanding what roots problems are and where they occur in engineering and science.
- Knowing how to determine a root graphically.
- Knowing how to solve a roots problem with the bisection method.
- Knowing how to estimate the error of bisection and why it differs from error estimates for other types of root-location algorithms.
- Understanding false position and how it differs from bisection

### 3.2.2 You've got a Problem

Medical studies have established that a bungee jumper's chances of sustaining a significant vertebrae injury increase significantly if the free-fall velocity exceeds 36 m/s after 4 s of free fall. Your boss at the bungee-jumping company wants you to determine the mass at which this criterion is exceeded given a drag coefficient of 0.25 kg/m.

You know from your previous studies that the following analytical solution can be used to predict fall velocity as a function of time:

to as optimization. As you learned in calculus, such solutions can be obtained analytically by determining the value at which the function is flat; that is, where its derivative is zero. Although such analytical solutions are sometimes feasible, most practical optimization problems require numerical, computer solutions. From a numerical standpoint, such optimization methods are similar in spirit to the

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) \quad (3.3)$$

Try as you might, you cannot manipulate this equation to explicitly solve for  $m$ , that is, you cannot isolate the mass on the left side of the equation.

An alternative way of looking at the problem involves subtracting  $v(t)$  from both sides to give a new function:

$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t) \quad (3.4)$$

Now we can see that the answer to the problem is the value of  $m$  that makes the function equal to zero. Hence, we call this a “roots” problem. This Section will introduce you to how to obtain such solutions.

### 3.2.3 Roots in Engineering and Science

Although they arise in other problem contexts, roots of equations frequently occur in the area of design. Table 3.1 lists a number of fundamental principles that are routinely used in design work. As taught in previous courses, mathematical equations or models derived from these principles are employed to predict dependent variables as a function of independent variables, forcing functions, and parameters. Note that in each case, the dependent variables reflect the state or performance of the system, whereas the parameters represent its properties or composition.

**Table 3.1 Fundamental Principles used in Design Problems**

Fundamental Principle	Dependent Variable	Independent Variable	Parameters
Heat balance	Temperature	Time and position	Thermal properties of material, system geometry
Mass balance	Concentration or quantity of mass	Time and position	Chemical behavior of material, mass transfer, system geometry
Force balance	Magnitude and direction of forces	Time and position	Strength of material, structural properties, system geometry
Energy balance	Changes in kinetic and potential energy	Time and position	Thermal properties, mass of material, system geometry
Newton's laws of motion	Acceleration, velocity, or location	Time and position	Mass of material, system geometry, dissipative parameters
Kirchhoff's laws	Currents and voltages	Time	Electrical properties (resistance, capacitance, inductance)



An example of such a model is the equation for the bungee jumper's velocity. If the parameters are known, Equation (3.3) can be used to predict the jumper's velocity. Such computations can be performed directly because  $v$  is expressed explicitly as a function of the model parameters. That is, it is isolated on one side of the equal sign.

However, as posed at the start of the chapter, suppose that we had to determine the mass for a jumper with a given drag coefficient to attain a prescribed velocity in a set time period. Although Equation (3.3) provides a mathematical representation of the interrelationship among the model variables and parameters, it cannot be solved explicitly for mass. In such cases,  $m$  is said to be **implicit**.

This presents a real problem, as many design problems involve specifying the properties or composition of a system (as represented by its parameters) to ensure that it performs in a desired manner (as represented by its variables). Thus, these problems often require the determination of implicit parameters.

The solution to the problem is provided by numerical methods for roots of equations. To solve the problem using numerical methods, it is conventional to express Equation (3.3) by subtracting the dependent variable  $v$  from both sides of the equation to give Equation (3.4). The value of  $m$  that makes  $f(m) = 0$  is, therefore, the root of the equation. This value also represents the mass that solves the design problem.

The following section deal with a variety of numerical and graphical methods for determining roots of relationships such as Equation (3.4). These techniques can be applied to many other problems confronted routinely in engineering and science.

### 3.2.4 Graphical Methods



A simple method for obtaining an estimate of the root of the equation  $f(x) = 0$  is to make a plot of the function and observe where it crosses the  $x$  axis. This point, which represents the  $x$  value for which  $f(x) = 0$ , provides a rough approximation of the root.

#### Example 3.1 The Graphical Approach

Use the graphical approach to determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. Note: The acceleration of gravity is  $9.81 \text{ m/s}^2$ .

#### Solution

Figure 3.2 shows a plot of Equation (3.4) versus mass:

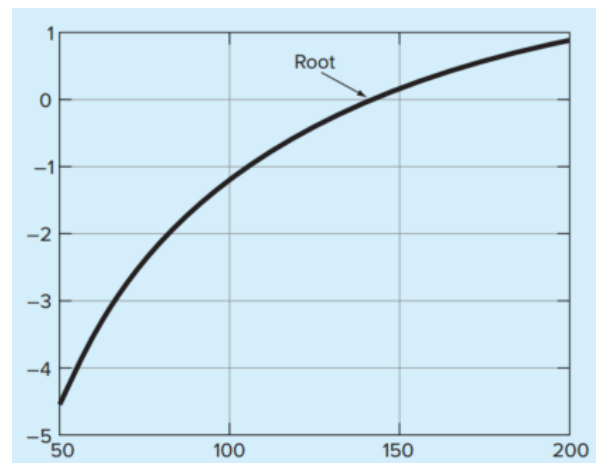


Figure 3.2 Plot of Equation (3.4) versus Mass

The function crosses the  $m$  axis between 140 and 150 kg. Visual inspection of the plot provides a rough estimate of the root of 145 kg. The validity of the graphical estimate can be checked by substituting it into Equation (3.4) to yield

$$f(m) = \sqrt{\frac{9.81(145)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{145}} \times 4\right) - 36 = 0.0456 \quad (3.5)$$

which is close to zero.

Graphical techniques are of limited practical value because they are not very precise. However, graphical methods can be utilized to obtain rough estimates of roots. These estimates can be employed as starting guesses for numerical methods discussed in this chapter.

Aside from providing rough estimates of the root, graphical interpretations are useful for understanding the properties of the functions and anticipating the pitfalls of the numerical methods. For example, Figure 3.3 shows a number of ways in which roots can occur (or be absent) in an interval prescribed by a lower bound  $x_l$  and an upper bound  $x_u$ . Figure 3.3b depicts the case where a single root is bracketed by negative and positive values of  $f(x)$ . However, Figure 3.3d, where  $f(x_l)$  and  $f(x_u)$  are also on opposite sides of the  $x$  axis, shows three roots occurring within the interval. In general, if  $f(x_l)$  and  $f(x_u)$

have opposite signs, there are an odd number of roots in the interval. As indicated by Figure 3.3a and Figure 3.3c, if  $f(x_l)$  and  $f(x_u)$  have the same sign, there are either no roots or an even number of roots between the values. Although these generalizations are usually true, there are cases where they do not hold.

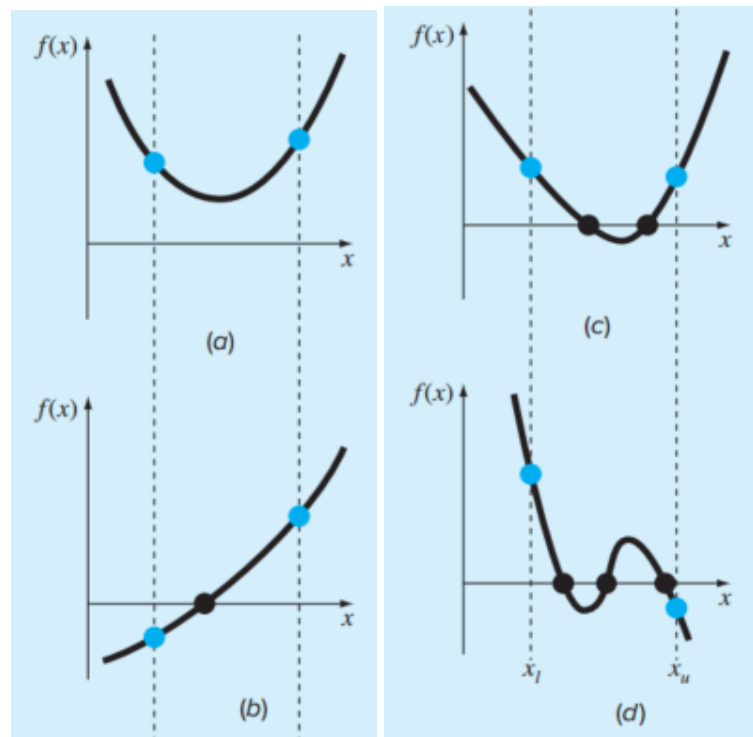


Figure 3.3 Occurrence of Roots in an Interval

### 3.3 Bracketing Methods and Initial Guesses

For many problems, it is preferable to have methods that come up with the correct answer automatically. Interestingly, as with trial and error, these approaches require an initial “guess” to get started. Then they systematically zoom in on the root in an iterative fashion.

The two major classes of methods available are distinguished by the type of initial guess. They are

- **Bracketing methods:** As the name implies, these are based on two initial guesses that “bracket” the root, that is, are on either side of the root. Bisection and False position methods are two known examples of the bracketing methods.
- **Open methods:** These methods can involve one or more initial guesses, but there is no need for them to bracket the root. Among the open methods, the Newton-Raphson is most commonly used. The most popular method for solving a non-linear equation is the Newton-Raphson method and this method has a high rate of convergence to a solution.

For well-posed problems, the bracketing methods always work but converge slowly (i.e., they typically take more iterations to home in on the answer). In contrast, the open methods do not always work (i.e., they can diverge), but when they do, they usually converge quicker.

### 3.4 Bisection Method

The bisection method is a variation of the incremental search method in which the interval is always divided in half. If a function changes sign over an interval, the function value at the midpoint is evaluated. The location of the root is then determined as lying within the subinterval where the sign change occurs. The subinterval then becomes the interval for the next iteration. The process is repeated until the root is known to the required precision. A graphical depiction of the method is provided in Figure 3.4.

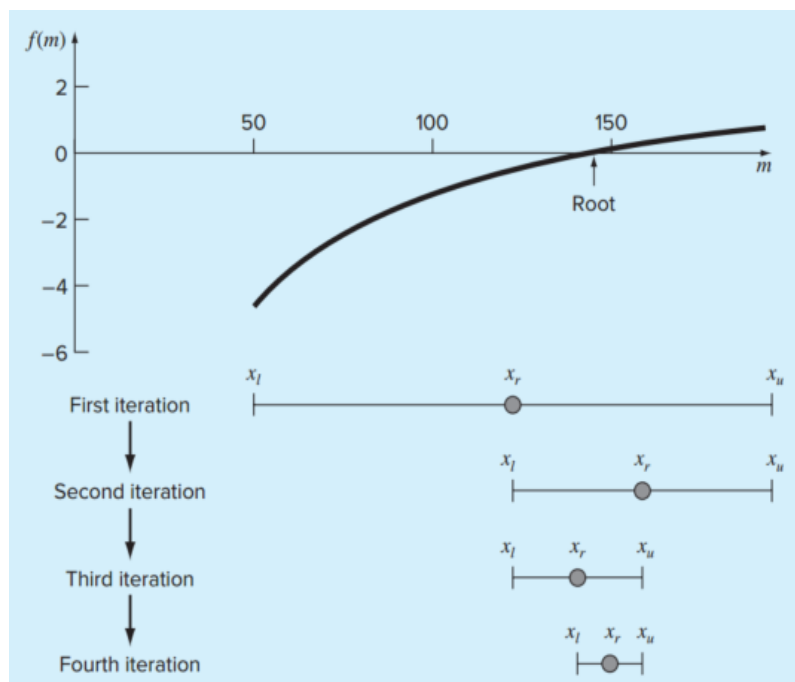


Figure 3.4 Graphical Depiction of the Bisection Method

The bisection method is based on the intermediate value theorem for continuous functions.

### ***Intermediate Value Theorem for Continuous Function***

*If  $f(x)$  is a continuous function and  $f(x_l)$  and  $f(x_u)$  have opposite signs, then at least one root lies in between  $x_l$  and  $x_u$ . If the interval  $(x_l, x_u)$  is small enough, it is likely to contain a single root. i.e., an interval  $[x_l, x_u]$  must contain a zero of a continuous function  $f$  if the product  $f(x_l)f(x_u) < 0$ . Geometrically, this means that if  $f(x_l)f(x_u) < 0$ , then the curve  $f(x)$  has to cross the  $x$ -axis at some point in between  $x_l$  and  $x_u$ .*

#### 3.4.1 Assumptions

- a)  $f(x)$  is a continuous function in interval  $[x_l, x_u]$
- b)  $f(x_l)f(x_u) < 0$

#### 3.4.2 Algorithm: Bisection Method

- a) Find middle point  $x_r = \frac{x_l + x_u}{2}$ .
- b) If  $f(x_r) = 0$ , then  $x_r$  is the root of the solution.
- c) Else  $f(x_r) \neq 0$ 
  - i. If  $f(x_l)f(x_r) < 0$  then root lies between  $x_l$  and  $x_r$ . So, we recur for  $x_l$  and  $x_r$ .
  - ii. Else If  $f(x_u)f(x_r) < 0$  then root lies between  $x_u$  and  $x_r$ . So, we recur  $x_u$  and  $x_r$ .
  - iii. Else given function doesn't follow one of assumptions.

Since root may be a floating-point number, we repeat above steps while difference between  $x_l$  and  $x_u$  is less than a value? (A very small value).

#### 3.4.3 Criterion for Termination

A convenient criterion is to compute the percentage error  $\varepsilon_r$  defined by Equation (2.7)

$$\varepsilon_r = \left( \frac{\text{present approximation} - \text{previous approximation}}{\text{present approximation}} \right) 100\%$$

or

$$|\varepsilon_\alpha| = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| 100\% \quad (3.6)$$

where  $x_r^{new}$  is the root for the present iteration and  $x_r^{old}$  is the root from the previous iteration. When  $\varepsilon_\alpha$  becomes less than a prespecified stopping criterion  $Tol$ , the computation is terminated.

### Example 3.2 Bisection Method

From Example 3.1 (determine the mass of the bungee jumper problem determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. Note: The acceleration of gravity is 9.81 m/s<sup>2</sup>).

$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t) \quad (3.7)$$

$$f(m) = \sqrt{\frac{9.81(m)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{m}} \times 4\right) - 36 \quad (3.8)$$

It is observed from Figure 3.5 that the function changes sign between values of 50 and 200.

The plot obviously suggests better initial guesses, say 140 and 150, but for illustrative purposes let's assume we don't have the benefit of the plot and have made conservative guesses.

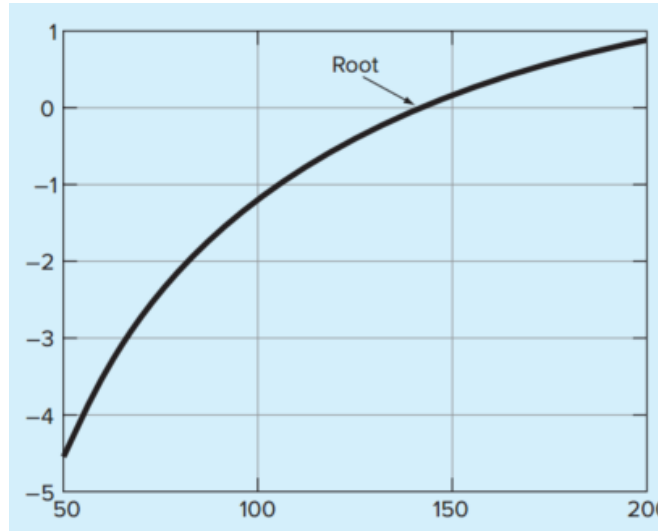


Figure 3.5 Plot of Equation (3.8) versus Mass

### Solution

First, check if  $f(x)$  is real and continuous in the interval from  $x_l$  to  $x_u$  and  $f(x_l)$  and  $f(x_u)$  have opposite signs, that is, if  $f(x_l)f(x_u) < 0$

$$f(50) = \sqrt{\frac{9.81(50)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{50}} \times 4\right) - 36 = -4.579$$

$$f(200) = \sqrt{\frac{9.81(200)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{200}} \times 4\right) - 36 = 0.860$$

Since  $f(50)f(200) = -4.579(0.860) = -3.938 < 0$ , then there is at least one real root between the interval  $x_l$  to  $x_u$ , that is between 50 to 200.

### 1<sup>st</sup> Iteration

The initial estimate of the root  $x_r$  lies at the midpoint of the interval

$$x_r = \frac{50 + 200}{2} = 125$$

Next, we compute the product of the function value at the lower bound and at the midpoint:

$$f(125) = \sqrt{\frac{9.81(125)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{125}} \times 4\right) - 36 = -0.409$$

$$\text{Hence, } f(50)f(125) = -4.579(-0.409) = 1.871$$

which is greater than zero, and hence no sign change occurs between the lower bound and the midpoint. Consequently, the root must be located in the upper interval between 125 and 200. At this point, the absolute relative approximate error  $|\varepsilon_a|$ , cannot be calculated as we do not have a previous approximation. Therefore, we create a new interval by redefining the lower bound as 125.

### 2<sup>nd</sup> Iteration

At this point, the new interval extends from  $x_l = 125$  to  $x_u = 200$ . A revised root estimate can then be calculated as

$$x_r = \frac{125 + 200}{2} = 162.5$$

Next, we compute the product of the function value at the lower bound and at the midpoint:

$$f(162.5) = \sqrt{\frac{9.81(162.5)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{162.5}} \times 4\right) - 36 = 0.359$$

$$\text{Hence, } f(125)f(162.5) = -0.409(0.359) = -0.147$$

Therefore, the root is now in the lower interval between 125 and 162.5. The upper bound is redefined as 162.5. Estimating an approximate percent relative error as using Equation (3.6)

$$|\varepsilon_a| = \left| \frac{162.5 - 125}{162.5} \right| 100\% = 23.08\%$$

The process can be repeated to obtain refined estimates.

### 3<sup>rd</sup> Iteration

The root estimate for the third iteration is calculated as

$$x_r = \frac{125 + 162.5}{2} = 143.75$$

Next, we compute the product of the function value at the lower bound and at the midpoint:

$$f(162.5) = \sqrt{\frac{9.81(143.75)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{143.75}} \times 4\right) - 36 = 0.021$$

$$\text{Hence, } f(162.5)f(143.75) = 0.359(0.021) = 0.008$$

Therefore, the root is now in the lower interval between 125 and 143.75. The upper bound is redefined as 143.75. The approximate percent relative error is estimated as

$$|\varepsilon_\alpha| = \left| \frac{143.75 - 125}{143.75} \right| 100\% = 13.04\%$$

The process is repeated until the result is accurate enough to satisfy your needs. That is, when  $\varepsilon_\alpha$  becomes less than a prespecified stopping criterion  $Tol$ , the computation is terminated.

Thus, after eight iterations  $|\varepsilon_\alpha|$  finally falls below  $Tol = 0.5\%$  (Table 3.2), and the computation can be terminated.

**Table 3.2 Summary of Computation**

Iteration	$x_l$	$x_u$	$x_r$	$ \varepsilon_\alpha $ (%)
1	50	200	125	
2	125	200	162.5	23.08
3	125	162.5	143.75	13.04
4	125	143.75	134.375	6.98
5	134.375	143.75	139.0625	3.37
6	139.0625	143.75	141.4063	1.66
7	141.4063	143.75	142.5781	0.82
8	142.5781	143.75	143.1641	0.41

Thus, we have employed the Bisection method to solve the problem posed at the beginning of the chapter, that you need to determine the mass at which a bungee jumper's free-fall velocity exceeds 36 m/s after 4 s of free fall given a drag coefficient of 0.25 kg/m. Thus, a result of  $m = 142.74$  kg is obtained after 21 iterations with an approximate relative error of  $\varepsilon_\alpha = 0.00005345\%$ .

### Example 3.3 Bisection Method

Given  $x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$ . Using the Bisection method find

- by performing three iterations to estimate the root of the above equation.
- find the absolute relative approximate error at the end of each iteration, and
- the number of significant digits at least correct at the end of each iteration if  $x_l = 0$  and  $x_u = 0.11$ .

#### Solution

Since  $f(x_l) = f(0) = (0)^3 - 0.165(0)^2 + 3.993 \times 10^{-4} = 3.993 \times 10^{-4}$

$$f(x_u) = f(0.11) = (0.11)^3 - 0.165(0.11)^2 + 3.993 \times 10^{-4} = -2.662 \times 10^{-4}$$

$$\text{Hence } f(x_l)f(x_u) = f(0)f(0.11) = (3.993 \times 10^{-4})(-2.662 \times 10^{-4}) < 0$$

So, there is at least one root between  $x_l$  and  $x_u$ , that is between 0 and 0.11

#### 1<sup>st</sup> Iteration

The estimate of the root is

$$x_m = \frac{x_l + x_u}{2} = \frac{0 + 0.11}{2} = 0.055$$

$$f(x_m) = f(0.055) = (0.055)^3 - 0.165(0.055)^2 + 3.993 \times 10^{-4} = 6.655 \times 10^{-5}$$

$$f(x_l)f(x_m) = f(0)f(0.055) = (3.993 \times 10^{-4})(6.655 \times 10^{-5}) > 0$$

Hence the root is bracketed between  $x_m$  and  $x_u$ , that is between 0.055 and 0.11.

So, the lower and upper limits of the new bracket are  $x_l = 0.055$  and  $x_u = 0.11$ .

At this point, the absolute relative approximate error,  $|\varepsilon_a|$  cannot be calculated as we do not have a previous approximation. Also, the number of significant digits at least correct at the end of this iteration cannot be calculated.

#### 2<sup>nd</sup> Iteration

The estimate of the root is

$$x_m = \frac{x_l + x_u}{2} = \frac{0.055 + 0.11}{2} = 0.0825$$

$$f(x_m) = f(0.0825) = (0.0825)^3 - 0.165(0.0825)^2 + 3.993 \times 10^{-4} = -1.622 \times 10^{-4}$$

$$f(x_l)f(x_m) = f(0.055)f(0.0825) = (6.655 \times 10^{-5})(-1.622 \times 10^{-4}) < 0$$

Hence the root is bracketed between  $x_l$  and  $x_m$ , that is between 0.055 and 0.0825.

So, the lower and upper limits of the new bracket are  $x_l = 0.055$  and  $x_u = 0.0825$ .



The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration #2 is

$$|\varepsilon_a| = \left| \frac{x_m^{new} - x_m^{old}}{x_m^{new}} \right| \times 100 = \left| \frac{0.0825 - 0.055}{0.0825} \right| \times 100 = 33.33\%$$

The number of significant digits at least correct is 0, since an absolute relative approximate error of less than 5% is required for a one significant digit to be correct. Hence, none of the significant digits are at least correct in the estimated root of  $x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$  because the absolute relative approximate error is greater than 5%.

### 3<sup>rd</sup> Iteration

The estimate of the root is

$$x_m = \frac{x_l + x_u}{2} = \frac{0.055 + 0.0825}{2} = 0.06875$$

$$f(x_m) = f(0.06875) = (0.06875)^3 - 0.165(0.06875)^2 + 3.993 \times 10^{-4} = -5.563 \times 10^{-5}$$

$$f(x_l)f(x_m) = f(0.055)f(0.06875) = (6.655 \times 10^{-5})(-5.563 \times 10^{-5}) < 0$$

Hence, the root is bracketed between  $x_l$  and  $x_m$ , that is between 0.055 and 0.06875.

So, the lower and upper limits of the new bracket is  $x_l = 0.055$  and  $x_u = 0.06875$ .

The absolute relative approximate error,  $|\varepsilon_a|$  at the end if iteration #3 is

$$|\varepsilon_a| = \left| \frac{x_m^{new} - x_m^{old}}{x_m^{new}} \right| \times 100 = \left| \frac{0.06875 - 0.0825}{0.06875} \right| \times 100 = 20\%$$

The number of significant digits at least correct is 0. Because. none of the significant digits are at least correct in the estimated root of the equation, as the absolute relative approximate error is greater than 5%. Seven more iterations were conducted, and these iterations are shown in Table 3.3.

At the end 10<sup>th</sup> of iteration,  $|\varepsilon_a| = 0.1721\%$

Hence the number of significant digits at least correct is given by the largest value of  $m$  for which

$$\begin{aligned} |\varepsilon_a| < \varepsilon_s = 0.5 \times 10^{2-m} &\Rightarrow 0.1721 < 0.5 \times 10^{2-m} \\ 0.3442 < 10^{2-m} &\Rightarrow \log(0.3442) < 2 - m \\ m < 2 - \log(0.3442) \end{aligned}$$

So  $m = 2$

The number of significant digits at least correct in the estimated root of 0.06241 at the end of the 10<sup>th</sup> iteration is 2.

**Table 3.3 Summary of Computation**

Iteration	$x_l$	$x_u$	$x_m$	$ \varepsilon_a  \%$	$f(x_m)$
1	0.000	0.11	0.055		$6.655 \times 10^{-5}$
2	0.055	0.11	0.0825	33.33	$-1.622 \times 10^{-4}$
3	0.055	0.0825	0.06875	20.00	$-5.563 \times 10^{-5}$
4	0.055	0.06875	0.06188	11.11	$4.484 \times 10^{-6}$
5	0.06188	0.06875	0.06531	5.263	$-2.593 \times 10^{-5}$
6	0.06188	0.06531	0.06359	2.702	$-1.0804 \times 10^{-5}$
7	0.06188	0.06359	0.06273	1.369	$-3.176 \times 10^{-6}$
8	0.06188	0.06273	0.0623	0.6896	$6.497 \times 10^{-7}$
9	0.0623	0.06273	0.06252	0.3436	$-1.264 \times 10^{-6}$
10	0.0623	0.06252	0.06241	0.1721	$-3.0767 \times 10^{-7}$

#### 3.4.4 MATLAB M-file: Bisection

An M-file to implement bisection is presented in **Appendix A**. It is passed the function (func) along with lower ( $x_l$ ) and upper guesses.


#### 3.4.5 Merits of Bisection Method

- The iteration using bisection method always produces a root, since the method brackets the root between two values.
- As iterations are conducted, the length of the interval gets halved. So, one can guarantee the convergence in case of the solution of the equation.
- The Bisection method is simple to program in a computer.

#### 3.4.6 Demerits of Bisection Method

- The convergence of the bisection method is slow as it is simply based on halving the interval.
- Bisection method cannot be applied over an interval where there is a discontinuity.
- Bisection method cannot be applied over an interval where the functional values always takes the same sign.
- The method fails to determine complex roots.

### 3.5 Regula Falsi Method (Method of False Position)

 **Key Point** False position (also called the linear interpolation method) is another well-known bracketing method. It is very similar to bisection with the exception that it uses a different strategy to come up with its new root estimate. Rather than bisecting the interval, it locates the root by joining  $f(x_l)$  and  $f(x_u)$  with a straight line (Figure 3.6).

The intersection of this line with the  $x$  axis represents an improved estimate of the root. Thus, the shape of the function influences the new root estimate. Using similar triangles, the intersection of the straight line with the  $x$  axis can be estimated as

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \quad (3.9)$$

This is the false-position formula. The value of  $x_r$  computed with Equation (3.9) then replaces whichever of the two initial guesses,  $x_l$  or  $x_u$ , yields a function value with the same sign as  $f(x_r)$ . In this way the values of  $x_l$  and  $x_u$  always bracket the true root. The process is repeated until the root is estimated adequately. The algorithm is identical to the Bisection algorithm with the exception that Equation (3.9) is used.

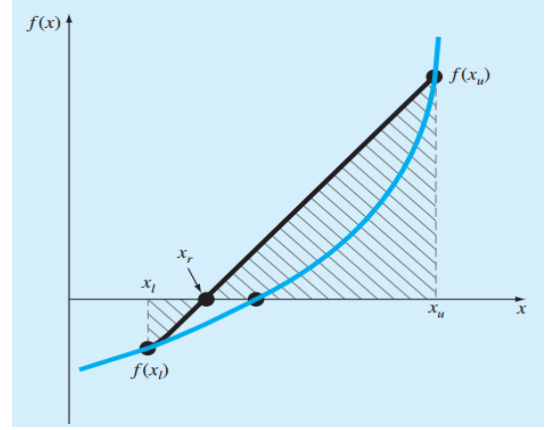


Figure 3.6 False Position

### 3.5.1 Similarities with Bisection Method

- Same Assumptions:** This method also assumes that function  $f(x)$  is continuous in  $[x_l, x_u]$  and given two numbers  $x_l$  and  $x_u$  are such that  $f(x_l)f(x_u) < 0$ .
- Always Converges:** like Bisection, it always converges, usually considerably faster than Bisection, but sometimes very much more slowly than Bisection.

### 3.5.2 Differences with Bisection Method

It differs in the fact that we make a chord joining the two points  $[x_l, f(x_l)]$  and  $[x_u, f(x_u)]$ . We consider the point at which the chord touches the  $x$  axis and named it as  $x_r$ .

### Example 3.4 The False-Position Method

Use false position to solve the same problem approached graphically and with bisection in Example 3.1 and Example 3.2.

#### Solution

As in Example 3.2, initiate the computation with guesses of  $x_l = 50$  and  $x_u = 200$ . First, check if  $f(x)$  is real and continuous in the interval from  $x_l$  to  $x_u$  and  $f(x_l)$  and  $f(x_u)$  have opposite signs, that is, if  $f(x_l)f(x_u) < 0$

$$x_l = 50 \quad f(50) = \sqrt{\frac{9.81(50)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{50}} \times 4\right) - 36 = -4.579387$$

$$x_u = 200 \quad f(200) = \sqrt{\frac{9.81(200)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{200}} \times 4\right) - 36 = 0.860291$$

Since  $f(50)f(200) = -4.579(0.860) = -3.938 < 0$ , then there is at least one real root between the interval  $x_l$  to  $x_u$ , that is between 50 to 200.

### 1<sup>st</sup> Iteration

$$x_r = 200 - \frac{0.860291(50 - 200)}{-4.579387 - 0.860291} = 176.2773$$

Next, we compute the product of the function value at the lower bound and at the midpoint:

$$f(176.2773) = \sqrt{\frac{9.81(176.2773)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{176.2773}} \times 4\right) - 36 = 0.566174$$

$$\text{Hence, } f(50)f(176.2773) = -4.579(0.566174) = -2.59251$$

Therefore, the root lies in the lower interval  $x_l$ , and  $x_r$  becomes the upper limit for the next iteration,  $x_u = 176.2773$ .

### 2<sup>nd</sup> Iteration

$$x_r = 176.2773 - \frac{0.566174(50 - 176.2773)}{-4.579387 - 0.566174} = 162.3828$$

Next, we compute the product of the function value at the lower bound and at the midpoint:

$$f(162.3828) = \sqrt{\frac{9.81(162.3828)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{162.3828}} \times 4\right) - 36 = 0.357508$$

$$\text{Hence, } f(50)f(162.3828) = -4.579(0.357508) = -1.6371$$

Therefore, the root lies in the lower interval  $x_l$ , and  $x_r$  becomes the upper limit for the next iteration,  $x_u = 162.3828$ , which has an approximate relative error of

$$|\varepsilon_a| = \left| \frac{162.3828 - 176.2773}{162.3828} \right| 100\% = 8.5566\%$$

Additional iterations can be performed to refine the estimates of the root. Although false position often performs better than bisection, there are other cases where it does not. As in the following Example 3.5, there are certain cases where bisection yields superior results.

### Example 3.5 A Case where Bisection is Preferable to False Position

Use bisection and false position to locate the root of

$$f(x) = x^{10} - 1$$

between  $x = 0$  and  $1.3$ .

#### Solution

Using bisection, the results can be summarized as Table 3.4.

**Table 3.4 Summary of Bisection Method**

Iteration	$x_l$	$x_u$	$x_r$	$ \varepsilon_a $ (%)
1	0	1.3	0.65	100.0
2	0.65	1.3	0.975	33.3
3	0.975	1.3	1.1375	14.3
4	0.975	1.1375	1.05625	7.7
5	0.975	1.05625	1.015625	4.0

Thus, after five iterations, the approximate error is reduced to less than 4%.

For false position, a very different outcome is obtained as shown in Table 3.5. After five iterations, the approximate error has only been reduced to about 17%.

**Table 3.5 Summary of False Position Method**

Iteration	$x_l$	$x_u$	$x_r$	$ \varepsilon_a $ (%)
1	0	1.3	0.09430	
2	0.09430	1.3	0.18176	48.1
3	0.18176	1.3	0.26287	30.9
4	0.26287	1.3	0.33811	22.3
5	0.33811	1.3	0.40788	17.1

### Example 3.6 False Position Method

Using regula falsi method, find a real root of the equation,  $f(x) = x^3 + x - 1$ , near  $x = 1$

Here note that  $f(0) = -1$  and  $f(1) = 1$ . Hence  $f(0)f(1) < 0$ , so by **intermediate value theorem** a root lies in between 0 and 1. We search for that root by regula falsi method and we will get an approximate root.

Set  $a_0 = 0$  and  $b_0 = 1$ . Then

$$x_0 = \frac{\begin{vmatrix} a_0 & b_0 \\ f(a_0) & f(b_0) \end{vmatrix}}{f(a_0) - f(b_0)} = \frac{\begin{vmatrix} 0 & 1 \\ -1 & 1 \end{vmatrix}}{1 - (-1)} = 0.5$$

and  $f(x_0) = f(0.5) = -0.375$ .

Since  $f(0)f(0.5) > 0$ , a root lies between 0.5 and 1. Set  $a_1 = x_0 = 0.5$  and  $b_1 = b_0 = 1$ .

Then

$$x_0 = \frac{\begin{vmatrix} a_1 & b_1 \\ f(a_1) & f(b_1) \end{vmatrix}}{f(a_1) - f(b_1)} = \frac{\begin{vmatrix} 0.5 & 1 \\ -0.375 & 1 \end{vmatrix}}{1 - (-0.375)} = 0.6364$$

and  $f(x_1) = f(0.6364) = -0.0158$ .

Since  $f(0.5)f(0.6364) > 0$ , a root lies between 0.6364 and 1. Set  $a_2 = x_1 = 0.6364$  and  $b_2 = b_1 = 1$ .

Then

$$x_0 = \frac{\begin{vmatrix} a_2 & b_2 \\ f(a_2) & f(b_2) \end{vmatrix}}{f(a_2) - f(b_2)} = \frac{\begin{vmatrix} 0.6364 & 1 \\ -0.1058 & 1 \end{vmatrix}}{1 - (-0.1058)} = 0.6712$$

and  $f(x_2) = f(0.6712) = -0.0264$ .

Since  $f(0.6364)f(0.6712) > 0$ , a root lies between 0.6712 and 1. Set  $a_3 = x_2 = 0.6712$  and  $b_3 = b_2 = 1$ .

Then

$$x_0 = \frac{\begin{vmatrix} a_3 & b_3 \\ f(a_3) & f(b_3) \end{vmatrix}}{f(a_3) - f(b_3)} = \frac{\begin{vmatrix} 0.6712 & 1 \\ -0.0264 & 1 \end{vmatrix}}{1 - (-0.0264)} = 0.6796$$

And  $f(x_3) = f(0.6796) = -0.0063 \approx 0$   $f(x_3) = f(0.6796) = -0.0063 \approx 0$ .

Since  $f(0.6796) \approx 0.0000$ , we accept 0.6796 as an (approximate) solution of  $x^3 + x - 1 = 0$ .

### 3.5.3 MATLAB M-file: Regula Falsi

An M-file to implement false position is presented in **Appendix B**.

### 3.5.4 Try Exercise

1. Find a real root of the following equations by Bisection/False position method.

- |                                |                                  |
|--------------------------------|----------------------------------|
| i. $2 \cos x - x = 0$ $[1, 2]$ | vii. $x^3 + 2x^2 + 10x - 20 = 0$ |
| ii. $x^3 - 5x = 6$             | viii. $2x - \log_{10} x = 7$     |
| iii. $x \log_{10} x = 1.2$     | ix. $e^x = 3x$                   |
| iv. $\tan x + \tanh x = 0$     | x. $x^3 - 5x + 1 = 0$            |
| v. $e^{-x} = \sin x$           | xi. $xe^x = \cos x$              |
| vi. $x^3 - 5x - 7 = 0$         | xii. $x^2 - \log_e x = 12$       |

xiii.  $3x - \cos x = 1$

xvii.  $x^3 - 5x + 3 = 0$

xiv.  $2x - 3\sin x = 5$

xviii.  $\cos x = \sqrt{x}$

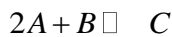
xv.  $2x = \cos x + 3$

xvi.  $xe^x = 3$

2. Use bisection to determine the drag coefficient needed so that an 80 kg bungee jumper has a velocity of 36 m/s after 9 s of free fall. Note: The acceleration of gravity is  $9.81 \text{ m/s}^2$ . Start with initial guesses of  $x_l = 0.1$  and  $x_u = 0.2$  and iterate until the approximate relative error falls below 2%. The mathematical expression for determining the drag coefficient is expressed as

$$f(c_d) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} \times t\right) - v(t)$$

3. (a) Determine the roots of  $f(x) = -12 - 21x + 18x^2 - 2.75x^3$  graphically. In addition, determine the first root of the function with  
(b) bisection and (c) false position. For (b) and  
(c) use initial guesses of  $x_l = -1$  and  $x_u = 0$  and a stopping criterion of 1%.  
4. Locate the root of  $f(x) = \sin(x) - x^2$  graphically, where  $x$  is in radians. Use a graphical technique and bisection with the initial interval from 0.5 to 1. Perform the computation until  $\varepsilon_a$  is less than 2%.  
5. A reversible chemical reaction



can be characterized by the equilibrium relationship

$$K = \frac{c_c}{c_a^2 c_b}$$

where the nomenclature  $c_i$  represents the concentration of constituent  $i$ . Suppose that we define a variable  $x$  as representing the number of moles of  $C$  that are produced. Conservation of mass can be used to reformulate the equilibrium relationship as

$$K = \frac{(c_{c,0} + x)}{(c_{a,0} - 2x)^2 (c_{b,0} - x)}$$

where the subscript 0 designates the initial concentration of each constituent. If  $K = 0.016$ ,  $c_{a,0} = 42$ ,  $c_{b,0} = 28$ , and  $c_{c,0} = 4$ , determine the value of  $x$ .

- (a) Obtain the solution graphically.

(b) On the basis of (a), solve for the root with initial guesses of  $x_l = 0$  and  $x_u = 20$  to  $\varepsilon_\alpha = 0.5\%$ . As an engineer, you have the option to choose either bisection or false position to obtain your solution. Justify your choice.

6. Mechanical engineers, as well as most other engineers, use thermodynamics extensively in their work. The following polynomial can be used to relate the zero-pressure specific heat of dry air  $c_p$  kJ/(kg K) to temperature (K):

$$c_p = 0.99403 + 1.671 \times 10^{-4} T + 9.7215 \times 10^{-8} T^2 - 9.5838 \times 10^{-11} T^3 + 1.9520 \times 10^{-14} T^4$$

Develop a plot of  $c_p$  versus a range of  $T = 0$  to 1200 K, and then use

- (a) bisection to determine the temperature that corresponds to a specific heat of 1.1 kJ/(kg K).
- (b) repeat the process using false position method.

### 3.6 Roots: Open Methods

#### 3.6.1 Objectives

The primary objective of this section is to acquaint you with open methods for finding the root of a single nonlinear equation. Specific objectives and topics covered are

- Recognizing the difference between bracketing and open methods for root location.
- Knowing how to solve a root problem with Newton-Raphson's method.
- Knowing how to implement both the secant and the modified secant methods.
- Knowing how to use MATLAB to estimate roots.

#### 3.6.2 Introduction

For the bracketing methods in Section 3.2, the root is located within an interval prescribed by a lower and an upper bound. Repeated application of these methods always results in closer estimates of the true value of the root. Such methods are said to be convergent because they move closer to the truth as the computation progresses (Figure 3.7a).

In contrast, the open methods described in this chapter require only a single starting value or two starting values that do not necessarily bracket the root. As such, they sometimes diverge or move away from the true root as the computation progresses (Figure 3.7b). However, when the open methods converge (Figure 3.7c) they usually do so much more quickly than the bracketing methods.



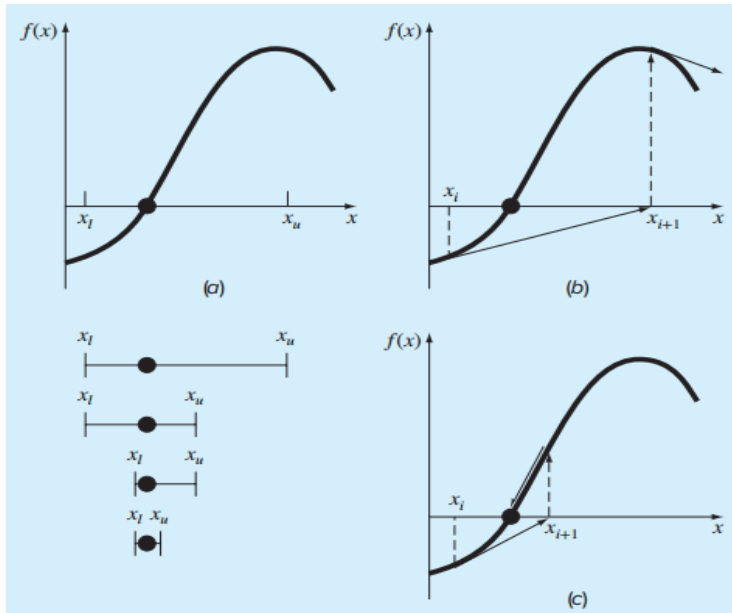


Figure 3.7 Fundamental difference b/n (a) Bracketing and (b), (c) Open Methods for Root Location

### 3.7 Newton-Raphson Method

Perhaps the most widely used of all root-locating formulas is the Newton-Raphson method (Figure 3.8). If the initial guess at the root is  $x_i$ , a tangent can be extended from the point  $[x_i, f(x_i)]$ . The point where this tangent crosses the  $x$  axis usually represents an improved estimate of the root.

The Newton-Raphson method can be derived based on this geometrical interpretation. As in Figure 3.8, the first derivative of  $x$  is equivalent to the slope:

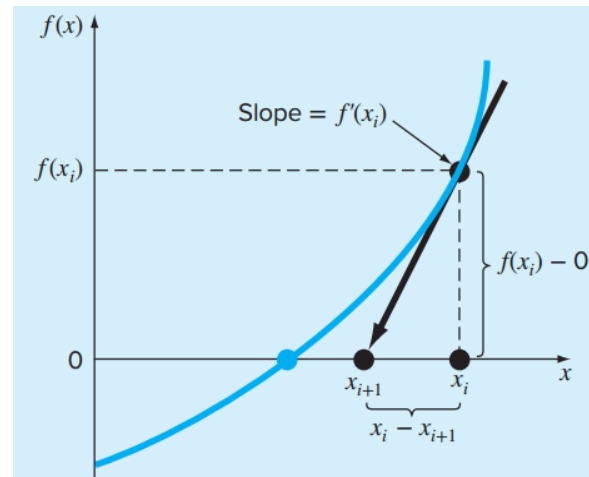


Figure 3.8 Newton-Raphson Method

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}} \quad (3.10)$$

which can be rearranged to yield

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (3.11)$$

which is called the Newton-Raphson formula.

**Assumption:** The function and its derivative should both be continuous.

### Example 3.7 Newton-Raphson Method

Use the Newton-Raphson method to estimate the root of  $f(x) = e^{-x} - x$  employing an initial guess of  $x_0 = 0$ .

#### Solution

The first derivative of the function can be evaluated as

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

which can be substituted along with the original function into Equation (3.11) to give

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

Starting with an initial guess of  $x_0 = 0$ , this iterative equation can be applied to compute

$$x_{i+1} = 0 - \frac{e^{-0} - 0}{-e^{-0} - 1} = 0.5000$$

Estimating an approximate percent relative error as using Equation (3.6)

$$|\varepsilon_a| = \left| \frac{0.5 - 0}{0.5} \right| 100\% = 100.00\%$$

0.5 is now assigned to  $x_i$ , for the next iteration. The process can be repeated to obtain the following results as shown in Table 3.6.

**Table 3.6 Summary of Newton-Raphson Method**

Iteration	$x_i$	$x_{i+1}$	$ \varepsilon_a $
1	0.0000	0.5000	100.0000
2	0.5000	0.5663	11.7093
3	0.5663	0.5671	0.1467
4	0.5671	0.5671	0.0000

Thus, the approach rapidly converges on the true root. Notice that the true percent relative error at each iteration decreases much faster.

### Example 3.8 Newton-Raphson Method

Given  $x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$ . Using the Newton-Raphson method with  $x_0 = 0.05$

- estimate the root of the above equation by performing three iterations.
- find the absolute relative approximate error at the end of each iteration, and
- the number of significant digits at least correct at the end of each iteration

### Solution

$$\text{Let } f(x) = x^3 - 0.165x^2 + 3.993 \times 10^{-4} \Rightarrow f'(x) = 3x^2 - 0.33x$$

#### 1<sup>st</sup> Iteration

The estimate of the root is

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} = 0.05 - \frac{(0.05)^3 - 0.165(0.05)^2 + 3.993 \times 10^{-4}}{3(0.05)^2 - 0.33(0.05)} \\ &= 0.05 - \frac{1.118 \times 10^{-4}}{-9 \times 10^{-3}} = 0.05 - (-0.01242) = 0.06242 \end{aligned}$$

The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration 1 is

$$|\varepsilon_a| = \left| \frac{x_1 - x_0}{x_1} \right| \times 100 = \left| \frac{0.06242 - 0.05}{0.06242} \right| \times 100 = 19.89\%$$

The number of significant digits at least correct is 0, as you need an absolute relative approximate error of less than 5% for one significant digit to be correct in your result.

#### 2<sup>nd</sup> Iteration

The estimate of the root is

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} = 0.06242 - \frac{(0.06242)^3 - 0.165(0.06242)^2 + 3.993 \times 10^{-4}}{3(0.06242)^2 - 0.33(0.06242)} \\ &= x_1 = \frac{-3.97781 \times 10^{-7}}{-8.90973 \times 10^{-3}} = 0.06242 - (4.4645 \times 10^{-5}) = 0.06238 \end{aligned}$$

The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration 2 is

$$|\varepsilon_a| = \left| \frac{x_2 - x_1}{x_2} \right| \times 100 = \left| \frac{0.06238 - 0.06242}{0.06238} \right| \times 100 = 0.0641\%$$

The number of significant digits at least correct is 2.

#### 3<sup>rd</sup> Iteration

The estimate of the root is

$$\begin{aligned} x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} = 0.06238 - \frac{(0.06238)^3 - 0.165(0.06238)^2 + 3.993 \times 10^{-4}}{3(0.06238)^2 - 0.33(0.06238)} \\ &= 0.06238 - \frac{4.42937 \times 10^{-11}}{-8.91171 \times 10^{-3}} = 0.06238 - (-4.9703 \times 10^{-9}) = 0.06238 \end{aligned}$$

The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration 3 is

$$|\varepsilon_a| = \left| \frac{x_3 - x_2}{x_3} \right| \times 100 = \left| \frac{0.06238 - 0.06238}{0.06238} \right| \times 100 = 0\%$$

The number of significant digits at least correct is 4, as only 4 significant digits are carried through in all the calculations.

### 3.7.1 MATLAB M-file: Newton Raphson

An M-file to implement Newton Raphson method is presented in **Appendix C**.

## 3.8 Secant Method

As in Example 3.7, a potential problem in implementing the Newton-Raphson method is the evaluation of the derivative. Although this is not inconvenient for polynomials and many other functions, there are certain functions whose derivatives may be difficult or inconvenient to evaluate. For these cases, the derivative can be approximated by a backward finite divided difference (Equation (2.21)) on page 22.

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \quad (3.12)$$

This approximation can be substituted into Equation (3.11) to yield the following iterative equation

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \quad (3.13)$$

Equation (3.13) is the formula for the **secant method**. Notice that the approach requires two initial estimates of  $x$ . However, because  $f(x)$  is not required to change signs between the estimates, it is not classified as a bracketing method.

Rather than using two arbitrary values to estimate the derivative, an alternative approach involves a fractional perturbation of the independent variable to estimate  $f'(x)$

$$f'(x_i) \cong \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i} \quad (3.14)$$

where  $\delta$  is a small perturbation fraction. This approximation (Equation (3.14)) can be substituted into Equation (3.11) to yield the following iterative equation:

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)} \quad (3.15)$$

Equation (3.15) is the modified secant method. As in the following Example 3.9, it provides a nice means to attain the efficiency of Newton-Raphson without having to compute derivatives.

### Example 3.9 Secant Method

Given  $x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$ . Using the Secant method with  $x_{-1} = 0.02$  and

$$x_0 = 0.05.$$

- estimate the root of the above equation by performing three iterations.
- find the absolute relative approximate error at the end of each iteration, and
- the number of significant digits at least correct at the end of each iteration

### Solution

#### 1<sup>st</sup> Iteration

The estimate of the root is

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)(x_0 - x_{-1})}{f(x_0) - f(x_{-1})} \\ &= x_0 - \frac{(x_0^3 - 0.165x_0^2 + 3.993 \times 10^{-4})(x_0 - x_{-1})}{(x_0^3 - 0.165x_0^2 + 3.993 \times 10^{-4}) - (x_{-1}^3 - 0.165x_{-1}^2 + 3.993 \times 10^{-4})} \\ &= 0.05 - \frac{[0.05^3 - 0.165(0.05)^2 + 3.993 \times 10^{-4}] \times [0.05 - 0.02]}{[0.05^3 - 0.165(0.05)^2 + 3.993 \times 10^{-4}] - [0.02^3 - 0.165(0.02)^2 + 3.993 \times 10^{-4}]} \\ &= 0.06461 \end{aligned}$$

The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration 1 is

$$|\varepsilon_a| = \left| \frac{x_1 - x_0}{x_1} \right| \times 100 = \left| \frac{0.06461 - 0.05}{0.06461} \right| \times 100 = 22.61\%$$

The number of significant digits at least correct is 0, as you need an absolute relative approximate error of less than 5% for one significant digit to be correct in your result.

#### 2<sup>nd</sup> Iteration

The estimate of the root is

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= x_1 - \frac{(x_1^3 - 0.165x_1^2 + 3.993 \times 10^{-4})(x_1 - x_0)}{(x_1^3 - 0.165x_1^2 + 3.993 \times 10^{-4}) - (x_0^3 - 0.165x_0^2 + 3.993 \times 10^{-4})} \\ &= 0.06461 - \frac{[0.06461^3 - 0.165(0.06461)^2 + 3.993 \times 10^{-4}] \times [0.06461 - 0.05]}{[0.06461^3 - 0.165(0.06461)^2 + 3.993 \times 10^{-4}] - [0.05^3 - 0.165(0.05)^2 + 3.993 \times 10^{-4}]} \\ &= 0.06241 \end{aligned}$$

The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration 2 is

$$|\varepsilon_a| = \left| \frac{x_2 - x_1}{x_2} \right| \times 100 = \left| \frac{0.06241 - 0.06461}{0.06241} \right| \times 100 = 3.525\%$$

The number of significant digits at least correct is 1, as you need an absolute relative approximate error of less than 5%. That is, when  $m$  is made the subject from  $3.525 < 0.5 \times 10^{2-m}$ .

### 3<sup>rd</sup> Iteration

The estimate of the root is

$$\begin{aligned} x_3 &= x_2 = \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} \\ &= x_2 - \frac{(x_2^3 - 0.165x_1^2 + 3.993 \times 10^{-4})(x_2 - x_1)}{(x_2^3 - 0.165x_2^2 + 3.993 \times 10^{-4}) - (x_1^3 - 0.165x_1^2 + 3.993 \times 10^{-4})} \\ &= 0.06241 - \frac{[0.06241^3 - 0.165(0.06241)^2 + 3.993 \times 10^{-4}] \times [0.06241 - 0.06461]}{[0.06241^3 - 0.165(0.06241)^2 + 3.993 \times 10^{-4}] - [0.06461^3 - 0.165(0.06461)^2 + 3.993 \times 10^{-4}]} \\ &= 0.06238 \end{aligned}$$

The absolute relative approximate error,  $|\varepsilon_a|$  at the end of iteration 3 is

$$|\varepsilon_a| = \left| \frac{x_3 - x_2}{x_3} \right| \times 100 = \left| \frac{0.06238 - 0.06241}{0.06238} \right| \times 100 = 0.0595\%$$

The number of significant digits at least correct is 1, as you need an absolute relative approximate error of less than 5% (Table 3.7).

**Table 3.7 Summary of Secant Method**

Iteration	I	$x_{i-1}$	$x_i$	$x_{i+1}$	$ \varepsilon_a $ %	$f(x_{i+1})$
1	0	0.02	0.05	0.06461	22.61	$-1.9812 \times 10^{-5}$
2	1	0.05	0.06461	0.06241	3.525	$-3.2852 \times 10^{-7}$
3	2	0.06461	0.06217	0.06238	0.0595	$2.0252 \times 10^{-9}$
4	3	0.06241	0.06238	0.06238	$3.64 \times 10^{-4}$	$-1.8576 \times 10^{-12}$

### Example 3.10 Modified Secant Method

Use the modified secant method to determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. Note: The acceleration of gravity is 9.81 m/s<sup>2</sup>. Use an initial guess of 50 kg and a value of  $10^{-6}$  for the perturbation fraction. Recall that the problem is expressed as

$$f(m) = \sqrt{\frac{9.81(m)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{m}} \times 4\right) - 36$$

### Solution

$$x_0 = 50 \quad f(50) = \sqrt{\frac{9.81(50)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{50}} \times 4\right) - 36 = -4.579387$$

$$x_0 + \delta x_0 = 50.00005 \quad f(50.00005) = \sqrt{\frac{9.81(50.00005)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{50.00005}} \times 4\right) - 36 \\ = -4.579381118$$

### 1<sup>st</sup> Iteration

Inserting the parameters into Equation (3.15) yields

$$x_1 = 50 - \frac{10^{-6}(50)(-4.57938708)}{-4.579381118 - (-4.57938708)} = 88.39931$$

Estimating an approximate percent relative error as using Equation (3.6)

$$|\varepsilon_a| = \left| \frac{88.39931 - 50}{88.39931} \right| 100\% = 43.4\%$$

### 2<sup>nd</sup> Iteration

$$x_1 = 88.39931 \quad f(88.39931) = \sqrt{\frac{9.81(88.39931)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{88.39931}} \times 4\right) - 36 \\ = -1.69220771$$

$$x_1 + \delta x_1 = 88.39940 \quad f(88.39940) = \sqrt{\frac{9.81(88.39940)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{88.39940}} \times 4\right) - 36 \\ = -1.6922.3516$$

$$x_2 = 88.39931 - \frac{10^{-6}(88.39931)(-1.69220771)}{-1.692203516 - (-1.69220771)} = 124.08970$$

Estimating an approximate percent relative error as using Equation (3.6)

$$|\varepsilon_a| = \left| \frac{124.08970 - 88.39931}{124.08970} \right| 100\% = 28.76\%$$

The process can be repeated to obtain the following results as shown in Table 3.8.

**Table 3.8 Summary of Modified Secant Method**

Iteration	$x_i$	$ \varepsilon_a $
0	50.0000	
1	88.3993	43.438
2	124.0897	28.762

3	140.5417	11.706
4	142.7072	1.517
5	142.7376	0.021
6	142.7376	0.000041

The choice of a proper value for  $\delta$  is not automatic. If  $\delta$  is too small, the method can be swamped by roundoff error caused by subtractive cancellation in the denominator of Equation (3.15). If it is too big, the technique can become inefficient and even divergent. However, if chosen correctly, it provides a nice alternative for cases where evaluating the derivative is difficult and developing two initial guesses is inconvenient. Since the derivative is approximated, it converges slower than Newton-Raphson method.

### 3.8.1 MATLAB M-file: Secant

An M-file to implement the secant method is presented in **Appendix D**.

### 3.8.2 Try Exercise

1. Use the Newton Raphson method to determine a root of  $f(x) = 2\cos x - x$  using  $x_0 = 1.5$ . Perform the computation until  $\epsilon_a$  is less than  $\epsilon_a = 0.01\%$ .
2. Use the Newton Raphson method to determine a root of  $f(x) = 2x^3 - 2x - 5$  using  $x_0 = 1.5$ . Perform the computation until  $\epsilon_a$  is less than  $\epsilon_a = 0.01\%$ .
3. Find a positive root of the equation  $x^4 + 2x + 1 = 0$  correct to 4 places of decimals. (Choose  $x_0 = 1.3$ )
4. Use the Newton Raphson method to determine a root of  $f(x) = -0.9x^2 + 1.7x + 2.5$  using  $x_0 = 5$ . Perform the computation until  $\epsilon_a$  is less than  $\epsilon_a = 0.01\%$ .
5. Determine the highest real root of  $f(x) = x^3 - 6x^2 + 11x - 6.1$  :
  - (a) Graphically.
  - (b) Using the Newton-Raphson method (three iterations,  $x_0 = 3.5$ )
  - (c) Using the secant method (three iterations,  $x_{-1} = 2.5$  and  $x_0 = 3.5$ ).
  - (d) Using the modified secant method (three iterations,  $x_0 = 3.5$ ,  $\delta = 0.01$ ).
6. Determine the lowest positive root of  $f(x) = 7\sin(x)e^{-x} - 1$  :
  - (a) Graphically.
  - (b) Using the Newton-Raphson method (three iterations,  $x_0 = 0.3$ ).
  - (c) Using the secant method (three iterations,  $x_{-1} = 0.5$  and  $x_0 = 0.4$ ).
  - (d) Using the modified secant method (five iterations,  $x_0 = 0.3$ ,  $\delta = 0.01$ ).



7. Use (a) the Newton-Raphson method and (b) the modified secant method ( $\delta = 0.05$ ) to determine a root of

$f(x) = x^5 - 16.05x^4 + 88.75x^3 - 192.0375x^2 + 116.35x + 31.6875$  using an initial guess of  $x = 0.5825$  and  $\varepsilon_a = 0.01\%$ . Explain your results.

8. Determine the real root of the equation  $xe^x = 1$  using the secant method. Compare your result with the true value of  $x = 0.567143\dots$

9. Use the secant method to determine the root, lying between 5 and 8, of the equation  $x^{22} = 69$ .

10. Use the secant and the modified secant methods ( $\delta = 0.01$ ) to solve the following equations, with the starting values given, correct to 3 decimal places:

(a)  $\cos x = 0.6x$   $x_{-1} = 1$  and  $x_0 = 1.1$

(b)  $2\cos x + 5x - 1 = 0$   $x_{-1} = -0.1$  and  $x_0 = 0$

(c)  $x^4 + x - 3 = 0$   $x_{-1} = 1.3$  and  $x_0 = 1.5$

(d)  $x^4 - 2 = 0$   $x_{-1} = 1.3$  and  $x_0 = 1.5$

11. The concentration of pollutant bacteria  $c$  in a lake decreases according to

$$c = 77e^{-1.5t} + 20e^{-0.08t}$$

Determine the time required for the bacteria concentration to be reduced to 15 using the Newton-Raphson method with an initial guess of  $t = 6$  and a stopping criterion of 1%.

12. Real mechanical systems may involve the deflection of nonlinear springs. In Figure 3.9, a block of mass  $m$  is released a distance  $h$  above a nonlinear spring. The resistance force  $F$  of the spring is given by

$$F = -\left(k_1 d + k_2 d^{\frac{3}{2}}\right)$$

Conservation of energy can be used to show that

$$0 = \frac{2k_2 d^{\frac{5}{2}}}{5} + \frac{1}{2}k_1 d^2 - mgd - mgh$$

Solve for  $d$ , given the following parameter values:  $k_1 = 40,000 \text{ g/s}^2$ ,  $k_2 = 40 \text{ g/(s}^2 \text{ m}^{0.5})$ ,  $m = 95 \text{ g}$ ,  $g = 9.81 \text{ m/s}^2$ , and  $h = 0.43 \text{ m}$ .

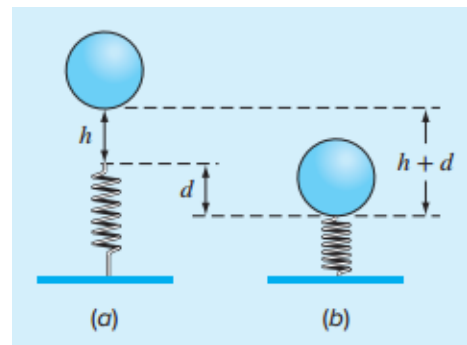


Figure 3.9 Nonlinear Spring

13. You are designing a spherical storage tank

Figure 3.10 to hold water for a small village in a developing country. The volume of liquid it can hold can be computed as

$$V = \pi h^2 \frac{(3R - h)}{3}, \text{ where } V = \text{volume [m}^3\text{]},$$

$h$  = depth of water in tank [m], and  $R$  =

the tank radius [m]. If  $R = 3$  m, what depth

must the tank be filled to so that it holds  $30 \text{ m}^3$ ? Use three iterations of the most efficient

numerical method possible to determine

your answer. Determine the approximate

relative error after each iteration. Also, provide justification for your choice of method.

**Extra information:** (a) For bracketing methods, initial guesses of 0 and  $R$  will bracket a single root for this example. (b) For open methods, an initial guess of  $R$  will always converge.

z

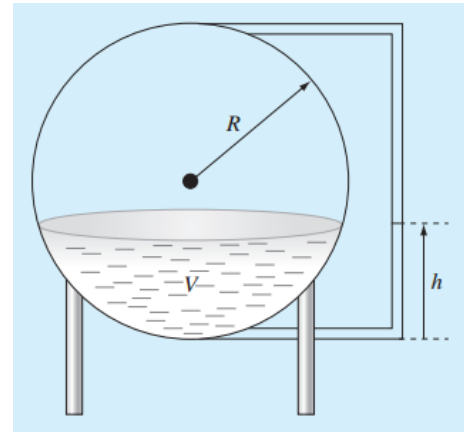


Figure 3.10 Spherical Tank

## CHAPTER 4

### ITERATIVE METHODS FOR SOLVING LINEAR SYSTEMS

#### 4.1 Objectives

The primary objective of this chapter is to acquaint you with iterative methods for solving simultaneous equations. Specific objectives and topics covered are

- Understanding the difference between the Gauss-Seidel and Jacobi methods.
- Knowing how to assess diagonal dominance and knowing what it means.

#### 4.2 Introduction

##### 4.2.1 What are Linear Algebraic Equations?

In Chapter 3, we determined the value of an unknown,  $x$ , that satisfied a single equation,  $f(x) = 0$ . Now, we deal with the case of determining the values  $x_1, x_2, \dots, x_n$ , that simultaneously satisfy a set of equations:

$$\begin{aligned}f_1(x_1, x_2, \dots, x_n) &= 0 \\f_2(x_1, x_2, \dots, x_n) &= 0 \\&\vdots \\f_n(x_1, x_2, \dots, x_n) &= 0\end{aligned}\tag{4.1}$$

Such systems are either linear or nonlinear. In this chapter, we deal with **linear algebraic equations** that are of the general form

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\&\vdots \\a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n\end{aligned}\tag{4.2}$$

where the  $a$ 's are constant coefficients, the  $b$ 's are constants, the  $x$ 's are unknowns, and  $n$  is the number of equations.

##### 4.2.2 Linear Algebraic Equations in Engineering and Science

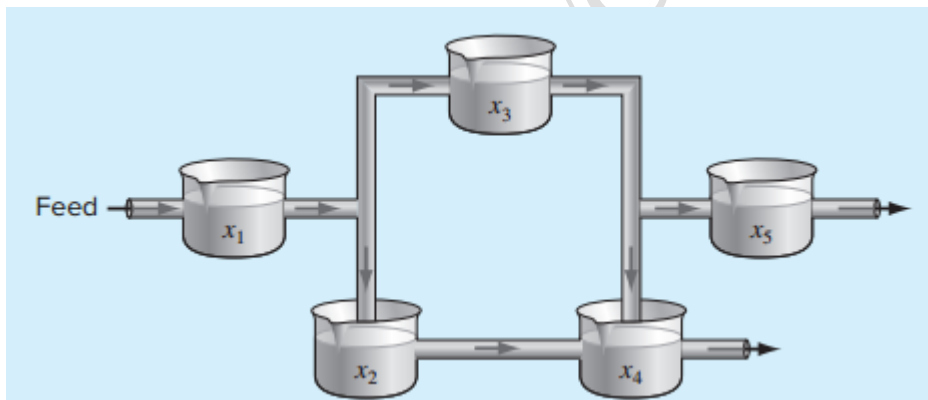
Many of the fundamental equations of engineering and science are based on conservation laws. Some familiar quantities that conform to such laws are mass, energy, and momentum. In mathematical terms, these principles lead to balance or continuity equations that relate system behaviour as represented by the levels or response of the quantity being modelled to the properties or characteristics of the system and the external stimuli or forcing functions acting on the system.

As an example, the principle of mass conservation can be used to formulate a model for a series of chemical reactors (Figure 4.1). For this case, the quantity being modelled

is the mass of the chemical in each reactor. The system properties are the reaction characteristics of the chemical and the reactors' sizes and flow rates. The forcing functions are the feed rates of the chemical into the system.

When we studied roots of equations in Chapter 3, you saw how single-component systems result in a single equation that can be solved using root-location techniques. Multicomponent systems result in a coupled set or system of mathematical equations that must be solved simultaneously. The equations are coupled because the individual parts of the system are influenced by other parts. For example, in Figure 4.1, reactor 4 receives chemical inputs from reactors 2 and 3. Consequently, its response is dependent on the quantity of chemical in these other reactors.

When these dependencies are expressed mathematically, the resulting equations are often of the linear algebraic form of Equation (4.2). The  $x$ 's are usually measures of the magnitudes of the responses of the individual components. Using Figure 4.1 as an example,  $x_1$  might quantify the amount of chemical mass in the first reactor,  $x_2$  might quantify the amount in the second, and so forth. The  $a$ 's typically represent the properties and characteristics that bear on the interactions between components. For instance, the  $a$ 's for Figure 4.1 might be reflective of the flow rates of mass between the reactors. Finally, the  $b$ 's usually represent the forcing functions acting on the system, such as the feed rate.



**Figure 4.1 System Involving Systems of Equations**

Aside from physical systems, simultaneous linear algebraic equations also arise in a variety of mathematical problem contexts such as trusses, reactors, and electric circuits. These result when mathematical functions are required to satisfy several conditions simultaneously. Each condition results in an equation that contains known coefficients and unknown variables. The techniques discussed in this chapter can be used to solve for the unknowns when the equations are linear and algebraic.

#### 4.2.3 You've Got a Problem

A mining company has two mines. One day's operation at Mine 1 produces ore that contains 20 metric tons of copper and 550 kilograms of silver, while one day's operation at Mine 2 produces ore that contains 30 metric tons of copper and 500 kilograms of

silver. Suppose the company operates Mine 1 for  $x_1$  days and Mine 2 for  $x_2$  days. How many days does the company needs to operate each mine in order to produce 150 tons of copper and 2825 kilograms of silver?

From the question above, a mathematical relation could be deduced as

$$20x_1 + 30x_2 = 150$$

$$550x_1 + 500x_2 = 2825$$

Thus, the problem reduces to solving a system of two simultaneous equations for the two unknown quantities.

Also, suppose that three jumpers are connected by bungee cords. Figure 4.2 being held in place vertically so that each cord is fully extended but unstretched. We can define three distances,  $x_1$ ,  $x_2$ , and  $x_3$ , as measured downward from each of their unstretched positions. After they are released, gravity takes hold and the jumpers will eventually come to the equilibrium positions shown in Figure 4.2b.

Suppose that you are asked to compute the displacement of each of the jumpers. If we assume that each cord behaves as a linear spring and follows Hooke's law, free-body diagrams can be developed for each jumper as depicted in Figure 4.3.

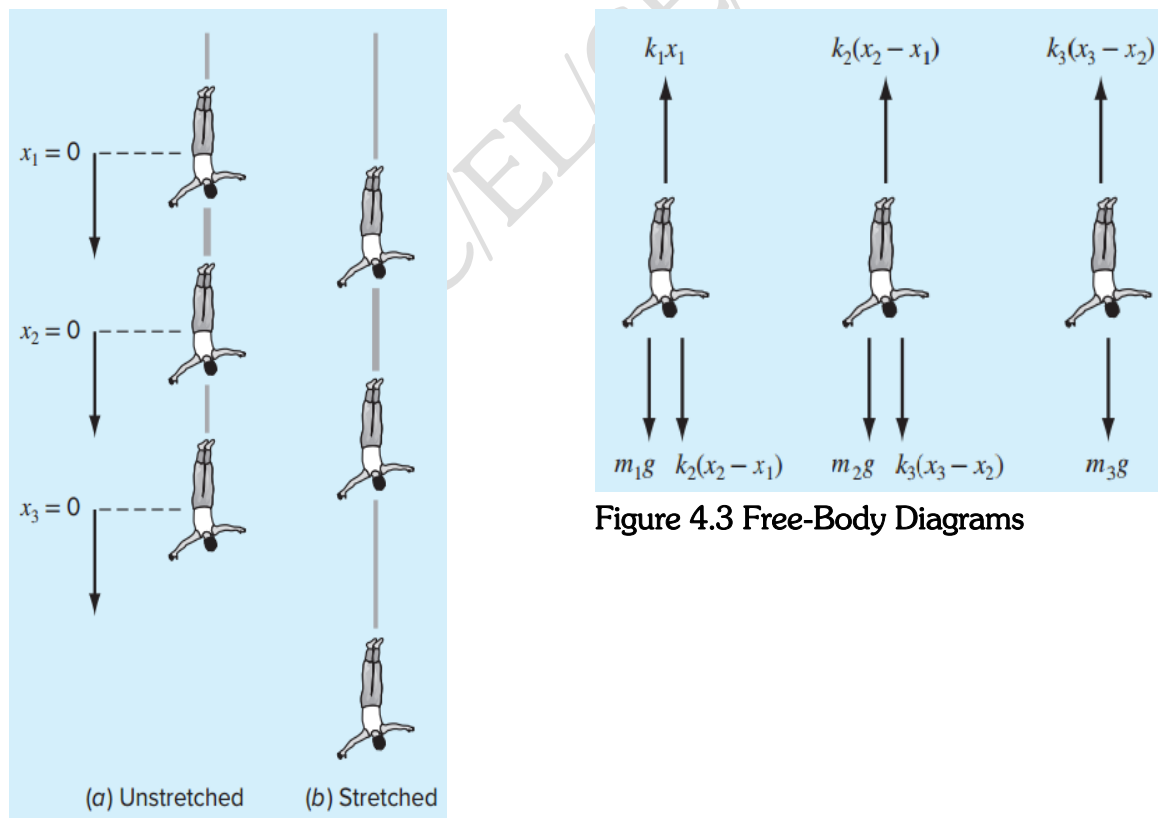


Figure 4.3 Free-Body Diagrams

Figure 4.2 Three Individuals Connected by Bungee Cords

Using Newton's second law, force balances can be written for each jumper

$$\begin{aligned}
m_1 \frac{d^2 x_1}{dt^2} &= m_1 g + k_2(x_2 - x_1) - k_1 x_1 \\
m_2 \frac{d^2 x_2}{dt^2} &= m_2 g + k_3(x_3 - x_2) + k_2(x_1 - x_2) \\
m_3 \frac{d^2 x_3}{dt^2} &= m_3 g + k_3(x_2 - x_3)
\end{aligned} \tag{4.3}$$

where  $m_i$  is the mass of jumper  $i$  (kg),  $t$  = time (s),  $k_j$  = the spring constant for cord  $j$  (N/m),  $x_i$  = the displacement of jumper  $i$  measured downward from the equilibrium position (m), and  $g$  = gravitational acceleration (9.81 m/s<sup>2</sup>). Because we are interested in the steady-state solution, the second derivatives can be set to zero,  $\frac{d^2 x_i}{dt^2} = 0$ .

Rearranging the in terms of  $x$ , yields

$$\begin{aligned}
(k_1 - k_2)x_1 - k_2 x_2 &= m_1 g \\
-k_2 x_1 + (k_2 - k_3)x_2 - k_3 x_3 &= m_2 g \\
-k_3 x_2 + k_3 x_3 &= m_3 g
\end{aligned} \tag{4.4}$$

Thus, the problem reduces to solving a system of three simultaneous equations for the three unknown displacements. Because we have used a linear law for the cords, these equations are linear algebraic equations.

Some widely used numerical techniques that employ simultaneous equations are the Gauss elimination, LU factorisation, Jacobi, Gauss Seidel etc.

### 4.3 Iterative Methods



Iterative or approximate methods provide an alternative to the elimination methods (refer to MN/MR/MC/EL/CE/RN 169: Linear Algebra). Such approaches are similar to the techniques we developed to obtain the roots of a single equation in the previous chapter. Those approaches consisted of guessing a value and then using a systematic method to obtain a refined estimate of the root. Because this section deals with a similar problem, obtaining the values that simultaneously satisfy a set of equations, we will be dealing with approximate methods which could be useful in this context. In this chapter, we will present approaches for solving both linear and nonlinear simultaneous equations.

### 4.4 Gauss-Seidel Iterative Method

The Gauss-Seidel method is the most commonly used iterative method for solving linear algebraic equations. It is also known as the **Liebman method** or the **method of successive displacement**. Assume that we are given a set of  $n$  equations:

$$[A]\{x\} = \{b\} \tag{4.5}$$

Suppose that for conciseness we limit ourselves to a  $3 \times 3$  set of equations. If the

diagonal elements are all nonzero, the first equation can be solved for  $x_1$ , the second for  $x_2$ , and the third for  $x_3$  to yield

$$x_1^j = \frac{b_1 - a_{12}x_2^{j-1} - a_{13}x_3^{j-1}}{a_{11}} \quad (4.6)$$

$$x_2^j = \frac{b_2 - a_{21}x_1^j - a_{23}x_3^{j-1}}{a_{22}} \quad (4.7)$$

$$x_3^j = \frac{b_3 - a_{31}x_1^j - a_{32}x_2^j}{a_{33}} \quad (4.8)$$

where  $j$  and  $j-1$  are the present and previous iterations.

To start the solution process, initial guesses must be made for the  $x$ 's. A simple approach is to assume that they are all zero. These zeros can be substituted into Equation (4.6), which can be used to calculate a new value for  $x_1 = \frac{b_1}{a_{11}}$ . Then we substitute this new value of  $x_1$  along with the previous guess of zero for  $x_3$  into Equation (4.7) to compute a new value for  $x_2$ . The process is repeated for Equation (4.8) to calculate a new estimate for  $x_3$ . Then we return to the first equation and repeat the entire procedure until our solution converges closely enough to the true values.

Convergence can be checked using the criterion that for all  $i$ ,

$$\varepsilon_{ai} = \left| \frac{x_i^j - x_i^{j-1}}{x_i^j} \right| \times 100\% \leq Tol \quad (4.9)$$

#### Example 4.1 Gauss-Seidel Method

Use the Gauss-Seidel method to obtain the solution for

$$\begin{aligned} 3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 &= 71.4 \end{aligned}$$

#### Solution

First, solve each of the equations for its unknown on the diagonal using Equations (4.6), (4.7) and (4.8) respectively.

$$x_1 = \frac{7.85 + 0.1x_2 + 0.2x_3}{3} \quad (4.10)$$

$$x_2 = \frac{-19.3 - 0.1x_1 + 0.3x_3}{7} \quad (4.11)$$

$$x_3 = \frac{71.4 - 0.3x_1 + 0.2x_2}{10} \quad (4.12)$$

### 1<sup>st</sup> Iteration

By assuming that  $x_2$  and  $x_3 = 0$ , Equation (4.10) can be used to compute

$$x_1^1 = \frac{7.85 + 0.1(0) + 0.2(0)}{3} = 2.616667 \quad (4.13)$$

This value, along with the assumed value of  $x_3 = 0$ , can be substituted into Equation (4.11) to calculate

$$x_2^1 = \frac{-19.3 - 0.1(2.616667) + 0.3(0)}{7} = -2.794524 \quad (4.14)$$

The first iteration is completed by substituting the calculated values for  $x_1$  and  $x_2$  into Equation (4.12) to yield

$$x_3^1 = \frac{71.4 - 0.3(2.616667) + 0.2(-2.794524)}{10} = 7.005610 \quad (4.15)$$

### 2<sup>nd</sup> Iteration

For the second iteration, the same process is repeated to compute

$$x_1^2 = \frac{7.85 + 0.1(-2.794524) + 0.2(7.005610)}{3} = 2.990557 \quad (4.16)$$

$$x_2^2 = \frac{-19.3 - 0.1(2.990557) + 0.3(7.005610)}{7} = -2.499625 \quad (4.17)$$

$$x_3^2 = \frac{71.4 - 0.3(2.990557) + 0.2(-2.499625)}{10} = 7.000291 \quad (4.18)$$

The method is, therefore, converging on the true solution. Additional iterations could be applied to improve the answers. However, in an actual problem, we would not know the true answer **a priori**. Consequently, Equation (4.9) provides a means to estimate the error. For example, for  $x_1$

$$|\varepsilon_{a1}| = \left| \frac{2.990557 - 2.616667}{2.990557} \right| \times 100\% = 12.5\% \quad (4.19)$$

For  $x_2$  and  $x_3$ , the error estimates are  $\varepsilon_{a2} = 11.8\%$  and,  $\varepsilon_{a3} = 0.076\%$ . Note that, as was the case when determining roots of a single equation, formulations such as Equation (4.9) usually provide a conservative appraisal of convergence. Thus, when they are met, they ensure that the result is known to at least the tolerance specified by  $Tol$ .



### Example 4.2 Gauss-Seidel Method

Use Gauss-Seidel Method to solve  $Ax = b$  where  $A = \begin{bmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{bmatrix}$ ,  $b = \begin{bmatrix} 106.8 \\ 177.2 \\ 279.2 \end{bmatrix}$

$$A = \begin{bmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 106.8 \\ 177.2 \\ 279.2 \end{bmatrix}$$

Assume an initial guess of the solution as  $[x_1 \ x_2 \ x_3]^T = [1 \ 2 \ 5]^T$ .

#### Solution

Rewriting the equations gives

$$\begin{aligned} x_1^{(k)} &= \frac{106.8 - 5x_2^{(k-1)} - x_3^{(k-1)}}{25} \\ x_2^{(k)} &= \frac{177.2 - 64x_1^{(k)} - x_3^{(k-1)}}{8} \\ x_3^{(k)} &= \frac{279.2 - 144x_1^{(k)} - 12x_2^{(k)}}{1} \end{aligned}$$

#### 1<sup>st</sup> Iteration

Given the initial guess of the solution vector as

$$\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}$$

we get

$$\begin{aligned} x_1^{(1)} &= \frac{106.8 - 5(2) - (5)}{25} = 3.6720 \\ x_2^{(1)} &= \frac{177.2 - 64(3.6720) - (5)}{8} = -7.8510 \\ x_3^{(1)} &= \frac{279.2 - 144(3.6720) - 12(-7.8510)}{1} = -155.36 \end{aligned}$$

The absolute relative approximate error for each  $x_i$  then is

$$|\varepsilon_a|_1 = \left| \frac{3.6720 - 1.0000}{3.6720} \right| \times 100 = 72.76\%$$

$$|\varepsilon_a|_2 = \left| \frac{-7.8510 - 2.0000}{-7.8510} \right| \times 100 = 125.47\%$$

$$|\varepsilon_a|_3 = \left| \frac{-155.36 - 5.0000}{-155.36} \right| \times 100 = 103.22\%$$

At the end of the first iteration, the guess of the solution vector is

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 3.6720 \\ -7.8510 \\ -155.36 \end{bmatrix}$$

and the maximum absolute relative approximate error is 125.47%.

## 2<sup>nd</sup> Iteration

The estimate of the solution vector at the end of iteration 1 is

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 3.6720 \\ -7.8510 \\ -155.36 \end{bmatrix}$$

Now we get

$$x_1^{(1)} = \frac{106.8 - 5(-7.8510) - (-155.36)}{25} = 12.0566$$

$$x_2^{(1)} = \frac{177.2 - 64(12.0566) - (-155.36)}{8} = -54.8828$$

$$x_3^{(1)} = \frac{279.2 - 144(12.0566) - 12(-54.8828)}{1} = -798.3568$$

The absolute relative approximate error for each  $x_i$  then is

$$|\varepsilon_a|_1 = \left| \frac{12.0566 - 3.6720}{12.0566} \right| \times 100 = 69.54\%$$

$$|\varepsilon_a|_2 = \left| \frac{-54.8828 - (-7.8510)}{-54.8828} \right| \times 100 = 85.69\%$$

$$|\varepsilon_a|_3 = \left| \frac{-798.3568 - (-155.36)}{-798.3568} \right| \times 100 = 80.54\%$$

At the end of second iteration the estimate of the solution is

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} 12.0566 \\ -54.8828 \\ -798.3568 \end{bmatrix}$$

and the maximum absolute relative approximate error is 85.69%.

Conducting more iterations gives the following values for the solution vector and the corresponding absolute relative approximate errors (Table 4.1).

**Table 4.1 Summary of Gauss-Seidel Computation**

Iteration	$x_1^{(k)}$	$ \varepsilon_a _1$ %	$x_2^{(k)}$	$ \varepsilon_a _2$ %	$x_3^{(k)}$	$ \varepsilon_a _3$ %
1	3.672	72.76	-7.8510	125.47	-155.36	103.22
2	12.056	67.64	-54.882	85.69	-798.34	80.54
3	47.182	74.45	-255.51	78.52	-3448.9	76.85
4	193.33	75.60	1093.4	76.63	-14440	76.12
5	800.53	75.85	-4577.2	76.11	-60072	75.96
6	3322.6	75.91	-19049	75.97	-249580	75.93

As seen in the Table 4.1, the solution is not converging to the true solution of

$$x_1 = 0.29048, \quad x_2 = 19.690, \quad x_3 = 1.0858$$

#### 4.4.1 Weakness of Gauss-Seidel Iterative Method

A pitfall of most iterative methods is that they may or may not converge. However, certain classes of systems of simultaneous equations do always converge to a solution using Gauss-Seidel method. This class of system of equations is where the coefficient matrix  $A$  in  $Ax = b$  is **diagonally dominant**, that is

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \quad \text{for all } i$$

and

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \quad \text{for at least one } i$$

That is, a matrix is diagonally dominant if the numerical value of the leading diagonal element in each row is greater than or equal to the sum of the numerical values of the other elements in that row.

For the Gauss-Seidel method to converge quickly, the coefficient matrix must be diagonally dominant. If it is not so, we have to rearrange the equations in such a way that the coefficient matrix is diagonally dominant and then only we can apply Gauss-Seidel method.

If a system of equations has a coefficient matrix that is not diagonally dominant, it may or may not converge. Fortunately, many physical systems that result in simultaneous

linear equations have diagonally dominant coefficient matrix, which then assures convergence for iterative methods such as Gauss-Seidel method of solving simultaneous linear equations.

### Example 4.3 Gauss-Seidel Method

Use Gauss-Seidel Method to solve where

$$12x_1 + 3x_2 - 5x_3 = 1$$

$$x_1 + 5x_2 + 3x_3 = 28$$

$$3x_1 + 7x_2 + 13x_3 = 76$$

find the solution. Given  $[x_1 \ x_2 \ x_3]^T = [1 \ 0 \ 1]^T$  as the initial guess.

#### Solution

The coefficient matrix

$$A = \begin{bmatrix} 12 & 3 & -5 \\ 1 & 5 & 3 \\ 3 & 7 & 13 \end{bmatrix}$$

is diagonally dominant as

$$|a_{11}| \geq |a_{12}| + |a_{13}| \Rightarrow |12| \geq |3| + |-5| = 8$$

$$|a_{22}| \geq |a_{21}| + |a_{23}| \Rightarrow |5| \geq |1| + |3| = 4$$

$$|a_{33}| \geq |a_{31}| + |a_{32}| \Rightarrow |13| \geq |3| + |7| = 10$$

and the inequality is strictly greater than for at least one row. Hence the solution should converge using Gauss-Seidel method.

Rewriting the equations, we get

$$x_1^{(k)} = \frac{1 - 3x_2^{(k-1)} + 5x_3^{(k-1)}}{12}$$

$$x_2^{(k)} = \frac{28 - x_1^{(k)} - 3x_3^{(k-1)}}{5}$$

$$x_3^{(k)} = \frac{76 - 3x_1^{(k)} - 7x_2^{(k)}}{13}$$

Assuming an initial guess of

$$\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

### 1<sup>st</sup> Iteration

$$x_1^{(1)} = \frac{1 - 3(0) + 5(1)}{12} = 0.5000$$

$$x_2^{(1)} = \frac{28 - (0.5) - 3(1)}{5} = 4.9000$$

$$x_3^{(1)} = \frac{76 - 3(0.5) - 7(4.9)}{13} = 3.0923$$

The absolute relative approximate error at the end of first iteration is

$$|\varepsilon_a|_1 = \left| \frac{0.5000 - 1.0000}{0.5000} \right| \times 100 = 100.0000\%$$

$$|\varepsilon_a|_2 = \left| \frac{4.9000 - 0}{4.9000} \right| \times 100 = 100.0000\%$$

$$|\varepsilon_a|_3 = \left| \frac{3.0923 - 1.0000}{3.0923} \right| \times 100 = 67.6616\%$$

The maximum absolute relative approximate error is 100.00%

### 2<sup>nd</sup> Iteration

The estimate of the solution vector at the end of iteration 1 is

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0.5000 \\ 4.9000 \\ 3.0923 \end{bmatrix}$$

We get

$$x_1^{(2)} = \frac{1 - 3(4.9000) + 5(3.0923)}{12} = 0.1468$$

$$x_2^{(2)} = \frac{28 - (0.1468) - 3(3.0923)}{5} = 3.7153$$

$$x_3^{(2)} = \frac{76 - 3(0.1468) - 7(4.9000)}{13} = 3.8118$$

The absolute relative approximate error at the end of second iteration is

$$|\varepsilon_a|_1 = \left| \frac{0.1467 - 0.5000}{0.1467} \right| \times 100 = 240.8316\%$$

$$|\varepsilon_a|_2 = \left| \frac{3.7153 - 4.9000}{3.7153} \right| \times 100 = 31.8871\%$$

$$|\varepsilon_a|_3 = \left| \frac{3.8118 - 3.0923}{3.8118} \right| \times 100 = 18.8756\%$$

From Table 4.2, the maximum absolute relative approximate error is 240.60%. This is greater than the value of 67.612% we obtained in the first iteration. Is the solution diverging? No, as you conduct more iterations, the solution converges as follows.

**Table 4.2 Summary of Gauss-Seidel (Showing Weakness)**

Iteration	$x_1$	$ \varepsilon_{\alpha 1} (\%)$	$x_2$	$ \varepsilon_{\alpha 2} (\%)$	$x_3$	$ \varepsilon_{\alpha 3} (\%)$
1	0.5000	100.0000	4.9000	100.0000	3.0923	67.6616
2	0.1468	240.8316	3.7153	31.8871	3.8118	18.8756
3	0.7428	80.2300	3.1644	17.4090	3.9708	4.0042
4	0.9468	21.5470	3.0281	4.5012	3.9971	0.6580
5	0.9918	4.5394	3.0034	0.8224	4.0001	0.0750
6	0.9992	0.7426	3.0001	0.1100	4.0001	0.0000

This is close to the exact solution vector of

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

#### 4.4.2 MATLAB M-file: Gauss-Seidel

An M-file to implement the Gauss-Seidel method is presented in **Appendix E**.

### 4.5 Jacobi Iterative Method

As each new  $x$  value is computed for the Gauss-Seidel method, it is immediately used in the next equation to determine another  $x$  value. Thus, if the solution is converging, the best available estimates will be employed. An alternative approach, called **Jacobi** iteration, utilizes a somewhat different tactic. Rather than using the latest available  $x$ 's, this technique uses Equations (4.6), (4.7) and (4.8) to compute a set of new  $x$ 's on the basis of a set of old  $x$ 's. Thus, as new values are generated, they are not immediately used but rather are retained for the next iteration.

The difference between the Gauss-Seidel method and Jacobi iteration is depicted in Figure 4.4. Although there are certain cases where the Jacobi method is useful, Gauss-Seidel's utilization of the best available estimates usually makes it the method of preference.

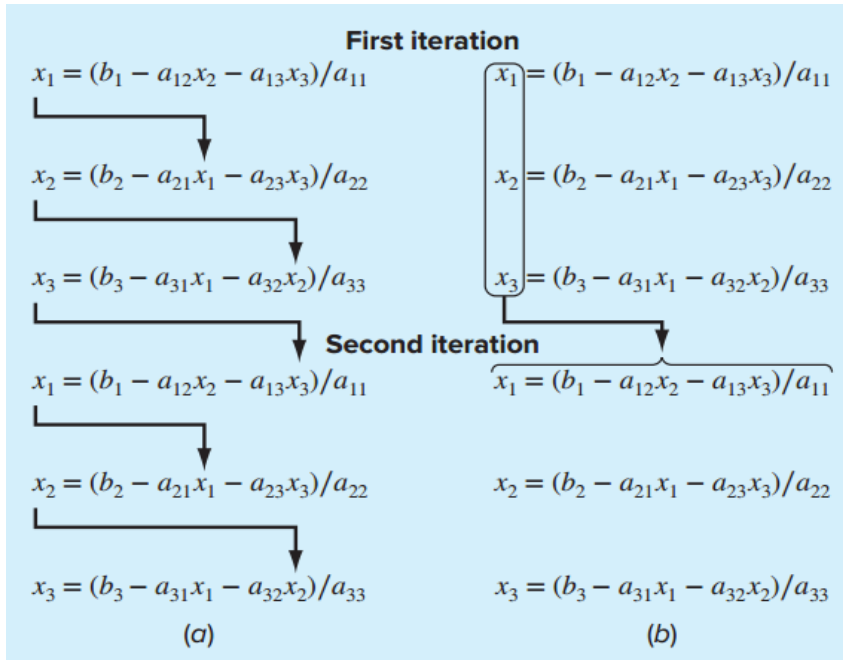


Figure 4.4 Difference b/n (a) the Gauss-Seidel and (b) the Jacobi Iterative Methods

#### Example 4.4 Jacobi's Iteration Method

Use Jacobi Method to solve where

$$\begin{aligned} 12x_1 + 3x_2 - 5x_3 &= 1 \\ x_1 + 5x_2 + 3x_3 &= 28 \\ 3x_1 + 7x_2 + 13x_3 &= 76 \end{aligned}$$

Assume an initial guess of the solution as  $[x_1 \ x_2 \ x_3]^T = [1 \ 0 \ 1]^T$ . Perform two (2) iterations.

#### Solution

Rewriting the equations gives

$$\begin{aligned} x_1^{(k)} &= \frac{1 - 3x_2^{(k-1)} + 5x_3^{(k-1)}}{12} \\ x_2^{(k)} &= \frac{28 - x_1^{(k-1)} - 3x_3^{(k-1)}}{5} \\ x_3^{(k)} &= \frac{76 - 3x_1^{(k-1)} - 7x_2^{(k-1)}}{13} \end{aligned}$$

Assuming an initial guess of

$$\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

### 1<sup>st</sup> Iteration

$$x_1^{(1)} = \frac{1 - 3(0) + 5(1)}{12} = 0.5000$$

$$x_2^{(1)} = \frac{28 - (1) - 3(1)}{5} = 4.8000$$

$$x_3^{(1)} = \frac{76 - 3(1) - 7(0)}{13} = 5.6154$$

The absolute relative approximate error at the end of first iteration is

$$|\varepsilon_a|_1 = \left| \frac{0.5000 - 1.0000}{0.5000} \right| \times 100 = 67.66\%$$

$$|\varepsilon_a|_2 = \left| \frac{4.8000 - 0}{4.8000} \right| \times 100 = 100\%$$

$$|\varepsilon_a|_3 = \left| \frac{5.6154 + 1.0000}{5.6154} \right| \times 100 = 82.19\%$$

The maximum absolute relative approximate error is 100.00%

### 2<sup>nd</sup> Iteration

The estimate of the solution vector at the end of iteration 1 is

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0.5000 \\ 4.8000 \\ 5.6154 \end{bmatrix}$$

We get

$$x_1^{(2)} = \frac{1 - 3(4.8000) + 5(5.6154)}{12} = 1.2231$$

$$x_2^{(2)} = \frac{28 - (0.5000) - 3(5.6154)}{5} = 2.1308$$

$$x_3^{(2)} = \frac{76 - 3(0.5000) - 7(4.8000)}{13} = 3.1462$$

The absolute relative approximate error at the end of second iteration is

$$|\varepsilon_a|_1 = \left| \frac{1.2231 - 0.5000}{1.2231} \right| \times 100 = 59.12\%$$

$$|\varepsilon_a|_2 = \left| \frac{2.1308 - 4.8000}{2.1308} \right| \times 100 = 125.27\%$$

$$|\varepsilon_a|_3 = \left| \frac{3.1462 - 5.6154}{3.1462} \right| \times 100 = 78.48\%$$

The maximum absolute relative approximate error is 125.27%.



### Example 4.5 Jacobi's Iteration Method

Use Jacobi Method to solve where

$$A = \begin{bmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 106.8 \\ 177.2 \\ 279.2 \end{bmatrix}$$

Assume an initial guess of the solution as  $[x_1 \ x_2 \ x_3]^T = [1 \ 2 \ 5]^T$ . Perform two (2) iterations in 4 dp.

#### Solution

Rewriting the equations gives

$$\begin{aligned} x_1^{(k)} &= \frac{106.8 - 5x_2^{(k-1)} - x_3^{(k-1)}}{25} \\ x_2^{(k)} &= \frac{177.2 - 64x_1^{(k-1)} - x_3^{(k-1)}}{8} \\ x_3^{(k)} &= \frac{279.2 - 144x_1^{(k-1)} - 12x_2^{(k-1)}}{1} \end{aligned}$$

#### 1<sup>st</sup> Iteration

Given the initial guess of the solution vector as

$$\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ x_3^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}$$

we get

$$\begin{aligned} x_1^{(1)} &= \frac{106.8 - 5(2) - (5)}{25} = 3.6720 \\ x_2^{(1)} &= \frac{177.2 - 64(1) - (5)}{8} = 13.5250 \\ x_3^{(1)} &= \frac{279.2 - 144(2) - 12(5)}{1} = 111.2000 \end{aligned}$$

The absolute relative approximate error for each  $x_i$  then is

$$\begin{aligned} |\varepsilon_a|_1 &= \left| \frac{3.6720 - 1.0000}{3.6720} \right| \times 100 = 72.7669\% \\ |\varepsilon_a|_2 &= \left| \frac{13.5250 - 2.0000}{13.5250} \right| \times 100 = 85.2126\% \\ |\varepsilon_a|_3 &= \left| \frac{111.2000 - 5.0000}{111.2000} \right| \times 100 = 108.5036\% \end{aligned}$$

At the end of the first iteration, the guess of the solution vector is

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 3.6720 \\ 13.5250 \\ 111.2000 \end{bmatrix}$$

and the maximum absolute relative approximate error is 108.53%.

## 2<sup>nd</sup> Iteration

The estimate of the solution vector at the end of iteration 1 is

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 3.6720 \\ 13.5250 \\ 111.2000 \end{bmatrix}$$

Now we get

$$\begin{aligned} x_1^{(2)} &= \frac{106.8 - 5(13.5250) - (111.2000)}{25} = -2.8810 \\ x_2^{(2)} &= \frac{177.2 - 64(3.6720) - (111.2000)}{8} = -21.1260 \\ x_3^{(2)} &= \frac{279.2 - 144(3.6720) - 12(13.5250)}{1} = -411.8680 \end{aligned}$$

The absolute relative approximate error for each  $x_i$  then is

$$\begin{aligned} |\varepsilon_a|_1 &= \left| \frac{-2.8810 - 3.6720}{-2.8810} \right| \times 100 = 227.4557\% \\ |\varepsilon_a|_2 &= \left| \frac{-21.1260 - 13.525}{-21.1260} \right| \times 100 = 164.0206\% \\ |\varepsilon_a|_3 &= \left| \frac{-411.8680 - 111.2000}{-411.8680} \right| \times 100 = 126.9989\% \end{aligned}$$

At the end of second iteration the estimate of the solution is

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} -2.8810 \\ -21.1260 \\ -411.8680 \end{bmatrix}$$

and the maximum absolute relative approximate error is 227.4557%.

#### 4.6 Try Exercise

1. Apply Gauss Seidel iteration method to solve:

$$10x + 2y + z = 9$$

$$2x + 20y - 2z = -44$$

$$-2x + 3y + 10z = 22$$

2. Apply Gauss Seidel iteration method to solve:

$$1.2x + 2.1y + 4.2z = 9.9$$

$$5.3x + 6.1y + 4.7z = 21.6$$

$$9.2x + 8.3y + z = 15.2$$

3. Apply Jacobi's iteration method to solve:

$$5x + 2y + z = 12$$

$$x + 4y + 2z = 15$$

$$x + 2y + 5z = 20$$

4. The following system of equations is designed to determine concentrations (the  $c$ 's in  $\text{g/m}^3$ ) in a series of coupled reactors as a function of the amount of mass input to each reactor (the right-hand sides in  $\text{g/day}$ ):

$$15c_1 - 3c_2 - c_3 = 3800$$

$$-3c_1 + 18c_2 - 6c_3 = 1200$$

$$-4c_1 - c_2 + 12c_3 = 2350$$

Solve this problem with the Gauss-Seidel and Jacobi's method to  $Tol = 5\%$

## CHAPTER 5

### NUMERICAL INTERPOLATION (POLYNOMIALS)

#### 5.1 Objectives

The primary objective of this chapter is to introduce you to polynomial interpolation. Specific objectives and topics covered are

- Recognizing that evaluating polynomial coefficients with simultaneous equations is an ill-conditioned problem.
- Knowing how to evaluate polynomial coefficients.
- Knowing how to perform an interpolation with Newton's polynomial.
- Knowing how to perform an interpolation with a Lagrange polynomial.

#### 5.2 You've Got a Problem

If we want to improve the velocity prediction for the free-falling bungee jumper, we might expand our model to account for other factors beyond mass and the drag coefficient. As was previously mentioned in Section 1.3, the drag coefficient can itself be formulated as a function of other factors such as the area of the jumper and characteristics such as the air's density and viscosity.

Air density and viscosity are commonly presented in tabular form as a function of temperature. For example, Table 5.1 is reprinted from a popular fluid mechanics textbook (White, 1999).

**Table 5.1 Fluid Mechanics Parameters**

Temperature (°C)	Density (kg/m <sup>3</sup> )	Dynamic Viscosity (Ns/m <sup>2</sup> )	Kinematic Viscosity (m <sup>2</sup> /s)
−40	1.52	$1.51 \times 10^{-5}$	$0.99 \times 10^{-5}$
0	1.29	$1.71 \times 10^{-5}$	$1.33 \times 10^{-5}$
20	1.20	$1.80 \times 10^{-5}$	$1.50 \times 10^{-5}$
50	1.09	$1.95 \times 10^{-5}$	$1.79 \times 10^{-5}$
100	0.946	$2.17 \times 10^{-5}$	$2.30 \times 10^{-5}$
150	0.835	$2.38 \times 10^{-5}$	$2.85 \times 10^{-5}$
200	0.746	$2.57 \times 10^{-5}$	$3.45 \times 10^{-5}$
250	0.675	$2.75 \times 10^{-5}$	$4.08 \times 10^{-5}$
300	0.616	$2.93 \times 10^{-5}$	$4.75 \times 10^{-5}$
400	0.525	$3.25 \times 10^{-5}$	$6.20 \times 10^{-5}$
500	0.457	$3.55 \times 10^{-5}$	$7.77 \times 10^{-5}$

(Source: White, 1999)

Suppose that you desired the density at a temperature not included in Table 5.1. In such a case, you would have to interpolate. That is, you would have to estimate the

value at the desired temperature based on the densities that bracket it. The simplest approach is to determine the equation for the straight line connecting the two adjacent values and use this equation to estimate the density at the desired intermediate temperature. Although such **linear interpolation** is perfectly adequate in many cases, error can be introduced when the data exhibit significant curvature. In this chapter, we will explore several different approaches for obtaining adequate estimates for such situations.

### 5.3 Introduction to Interpolation

You will frequently have occasion to estimate intermediate values between precise data points. The most common method used for this purpose is polynomial interpolation. The general formula for an  $(n-1)^{th}$ -order polynomial can be written as

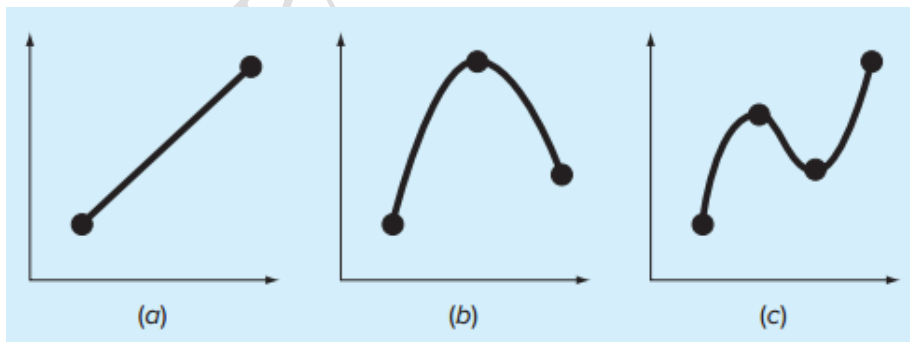
$$f(x) = a_1 + a_2x + a_3x^2 + \dots + a_nx^{n-1} \quad (5.1)$$

For  $n$  data points, there is one and only one polynomial of order  $(n-1)$  that passes through all the points. For example, there is only one straight line (i.e., a first-order polynomial) that connects two points (Figure 5.1a). Similarly, only one quadratic or parabola connects a set of three points (Figure 5.1b). Polynomial interpolation consists of determining the unique  $(n-1)^{th}$ -order polynomial that fits  $n$  data points. This polynomial then provides a formula to compute intermediate values.

However, rather than using increasing powers of  $x$  as shown in Equation (5.1), softwares such as MATLAB it uses decreasing powers as in

$$f(x) = p_1x^{n-1} + p_2x^{n-2} + \dots + p_{n-1}x + p_n \quad (5.2)$$

To ensure consistency, Equation (5.2) we is adopted throughout the remaining sections.



**Figure 5.1 Interpolating Polynomials**

#### 5.3.1 Determining Polynomial Coefficients

A straightforward way for computing the coefficients of Equation (5.2) is based on the fact that  $n$  data points are required to determine the  $n$  coefficients. As in the following

example, this allows us to generate  $n$  linear algebraic equations that we can solve simultaneously for the coefficients.

### Example 5.1 Determining Polynomial Coefficients with Simultaneous equations

Suppose that we want to determine the coefficients of the parabola,  $f(x) = p_1x^2 + p_2x + p_3$ , that passes through the last three density values from Table 5.1.

#### Solution

$$x_1 = 300 \quad f(x_1) = 0.616 \quad \Rightarrow f(300) = p_1(300)^2 + p_2(300) + p_3 = 0.616$$

$$x_2 = 400 \quad f(x_2) = 0.525 \quad \Rightarrow f(400) = p_1(400)^2 + p_2(400) + p_3 = 0.525$$

$$x_3 = 500 \quad f(x_3) = 0.457 \quad \Rightarrow f(500) = p_1(500)^2 + p_2(500) + p_3 = 0.457$$

The system of three equations is written in a matrix form as

$$\begin{bmatrix} 90000 & 300 & 1 \\ 160000 & 400 & 1 \\ 250000 & 500 & 1 \end{bmatrix} \begin{Bmatrix} p_1 \\ p_2 \\ p_3 \end{Bmatrix} = \begin{Bmatrix} 0.616 \\ 0.545 \\ 0.457 \end{Bmatrix} \quad (5.3)$$

Thus, the problem reduces to solving three simultaneous linear algebraic equations for the three unknown coefficients. A simple Gaussian elimination can be used to find the solution for the problem as

$$p_1 = 0.00000115$$

$$p_2 = -0.001715$$

$$p_3 = 1.027$$

Thus, the parabola that passes exactly through the three points is

$$f(x) = 0.00000115x^2 - 0.001715x + 1.027$$

This polynomial then provides a means to determine intermediate points. For example, the value of density at a temperature of 350 °C can be calculated as

$$f(350) = 0.00000115(350)^2 - 0.001715(350) + 1.027 = 0.567625$$

Although the approach in Example 5.1 provides an easy way to perform interpolation, it has a serious deficiency. Coefficient matrices of the form shown in Equation (5.3) are referred to as **Vandermonde matrices**. Such matrices are very ill-conditioned. The ill-conditioning becomes even worse as the number of simultaneous equations becomes larger.

As a remedy, there are alternative approaches that do not manifest this shortcoming. In subsequent sections, we will describe two alternatives that are well-suited for computer implementation: the **Newton** and the **Lagrange polynomials**.

## 5.4 Newton Interpolating Polynomial

There are a variety of alternative forms for expressing an interpolating polynomial beyond the familiar format of Equation (5.2). Newton's interpolating polynomial is among the most popular and useful forms. Before presenting the general equation, we will introduce the first- and second-order versions because of their simple visual interpretation.

### 5.4.1 Linear Interpolation

The simplest form of interpolation is to connect two data points with a straight line. This technique, called **linear interpolation**. Using similar triangles,

$$\frac{f_1(x) - f(x_1)}{x - x_1} = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad (5.4)$$

which can be rearranged to yield

$$f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1} (x - x_1) \quad (5.5)$$

which is the Newton linear-interpolation formula. The notation  $f_1(x)$  designates that this is a first-order interpolating polynomial. Notice that besides representing the slope of the line connecting the points, the term  $\frac{f(x_2) - f(x_1)}{x_2 - x_1}$  is a finite-difference

approximation of the first derivative [recall Equation (2.18)]. In general, the smaller the interval between the data points, the better the approximation. This is due to the fact that, as the interval decreases, a continuous function will be better approximated by a straight line. This characteristic is demonstrated in the following Example 5.2.

#### Example 5.2 Linear Interpolation

Estimate the natural logarithm of 2 using linear interpolation. First, perform the computation by interpolating between  $\ln 1 = 0$  and  $\ln 6 = 1.791759$ . Then, repeat the procedure, but use a smaller interval from  $\ln 1$  to  $\ln 4$  (1.386294). Note that the true value of  $\ln 2$  is 0.6931472.

#### Solution

Using Equation (5.5) from  $x_1 = 1$  to  $x_2 = 6$  to give

$$f_1(2) = 0 + \frac{1.791759 - 0}{6 - 1} (2 - 1) = 0.3583519$$

which represents an error of  $\varepsilon_t = 48.3\%$ .

Using the smaller interval from  $x_1 = 1$  to  $x_2 = 4$  yields

$$f_1(2) = 0 + \frac{1.386294 - 0}{4 - 1} (2 - 1) = 0.4620981$$

Thus, using the shorter interval reduces the percent relative error to  $\varepsilon_t = 33.3\%$ . Both interpolations are shown in Figure 5.2, along with the true function.

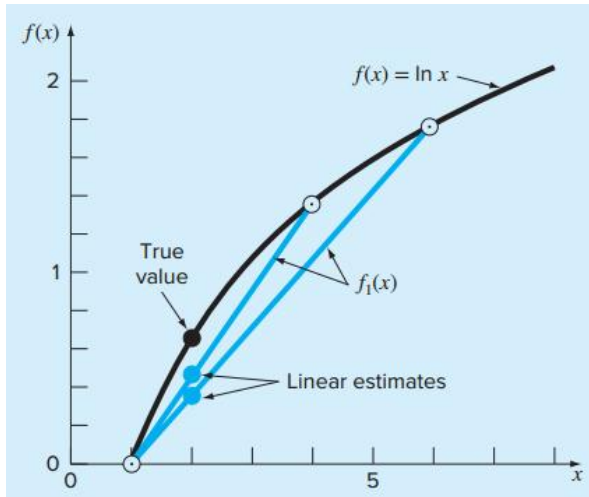


Figure 5.2 Two Linear Interpolations to Estimate  $\ln 2$

Note how the smaller interval provides a better estimate.

### Example 5.3 Linear Interpolation

Evaluate  $\ln 9.2$ , given that  $\ln 9.0 = 2.197$  and  $\ln 9.5 = 2.251$ .

#### Solution

Here  $x_1 = 9.0$ ,  $x_2 = 9.5$

Then,  $f_1 = f(x_1) = \ln 9.0 = 2.197$  and  $f_2 = f(x_2) = \ln 9.5 = 2.251$

Now to calculate  $\ln 9.2 = f(9.2)$ , take  $x = 9.2$ , so that

$$\ln 9.2 \Rightarrow f(9.2) = 2.197 + \frac{2.251 - 2.197}{9.5 - 9.0}(9.2 - 9.0) = 2.219$$

### Example 5.4 Linear Interpolation

Evaluate  $f(15)$ , given that  $f(10) = 46$  and  $f(20) = 66$ .

#### Solution

Here  $x_1 = 10$ ,  $x_2 = 20$

Then,  $f_1 = f(x_1) = 46$  and  $f_2 = f(x_2) = 66$

Now to calculate  $f(15)$ , take  $x = 15$ , so that

$$f(15) = 46 + \frac{66 - 46}{20 - 10}(15 - 10) = 56$$

### Example 5.5 Linear Interpolation

Evaluate  $e^{1.24}$ , given that  $e^{1.1} = 3.0042$  and  $e^{1.4} = 4.0552$ .



### Solution

Here  $x_1 = 1.1$ ,  $x_2 = 1.4$ ,  $f(x_1) = 3.0042$  and  $f(x_2) = 4.0552$

Now to calculate  $e^{1.24} = f(1.24)$ , take  $x = 1.24$ , so that

$$e^{1.24} \Rightarrow f(1.24) = 3.0042 + \frac{4.0552 - 3.0042}{1.4 - 1.1}(1.24 - 1.1) = 3.4933$$

while the exact value of  $e^{1.24}$  is 3.4947.

#### 5.4.2 Quadratic Interpolation

The error in Example 5.2 resulted from approximating a curve with a straight line. Consequently, a strategy for improving the estimate is to introduce some curvature into the line connecting the points. If three data points are available, this can be accomplished with a second-order polynomial (also called a quadratic polynomial or a parabola). A particularly convenient form for this purpose is

$$f(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2) \quad (5.6)$$

A simple procedure can be used to determine the values of the coefficients. For  $b_1$ , Equation (5.6) with  $x = x_1$  can be used to compute

$$b_1 = f(x_1) \quad (5.7)$$

Equation (5.7) can be substituted into Equation (5.6), which can be evaluated at  $x = x_2$  for

$$b_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad (5.8)$$

Finally, Equations (5.7) and (5.8) can be substituted into Equation (5.6), which can be evaluated at  $x = x_3$  and solved (after some algebraic manipulations) for

$$b_3 = \frac{\frac{f(x_3) - f(x_2)}{x_3 - x_2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_3 - x_1} \quad (5.9)$$

Notice that, as was the case with linear interpolation,  $b_2$  still represents the slope of the line connecting points  $x_1$  and  $x_2$ . Thus, the first two terms of Equation (5.6) are equivalent to linear interpolation between  $x_1$  and  $x_2$ , as specified previously in Equation (5.5). The last term,  $b_3(x - x_1)(x - x_2)$ , introduces the second-order curvature into the formula.

### Example 5.6 Quadratic Interpolation

Employ a second-order Newton polynomial to estimate  $\ln 2$  with the same three points used in Example 5.2:

$$x_1 = 1 \quad f(1) = \ln(1) = 0$$

$$x_2 = 4 \quad f(4) = \ln(4) = 1.386294$$

$$x_3 = 6 \quad f(6) = \ln(6) = 1.791759$$

#### Solution

Applying Equation (5.7) yields

$$b_1 = 0$$

Equation (5.8) gives

$$b_2 = \frac{1.386294 - 0}{4 - 1} = 0.4620981$$

and Equation (5.9) yields

$$b_3 = \frac{\frac{1.791759 - 1.386294}{6 - 4} - 0.4620981}{6 - 1} = -0.0518731$$

Substituting these values into Equation (5.6) yields the quadratic formula

$$f_2(x) = 0 + 0.4620981(x - 1) - 0.0518731(x - 1)(x - 4)$$

which can be evaluated at  $x = 2$  for  $f_2(2) = 0.5658444$ , which represents a relative error of  $\varepsilon_r = 18.4\%$ . Thus, the curvature introduced by the quadratic formula (Figure 5.3) improves the interpolation compared with the result obtained using straight lines in Example 5.2 and Figure 5.2.

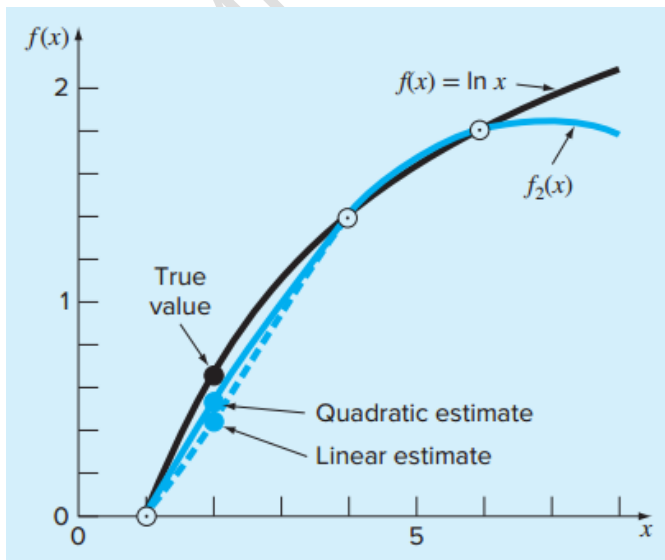


Figure 5.3 Comparison of Quadratic and Linear Interpolation of  $\ln 2$

### 5.4.3 General Form of Newton's Interpolating Polynomials

The preceding analysis can be generalized to fit an  $(n - 1)^{th}$ -order polynomial to  $n$  data points. The  $(n - 1)^{th}$ -order polynomial is

$$f_{n-1}(x) = b_1 + b_2(x - x_1) + \dots + b_n(x - x_1)(x - x_2) \dots (x - x_{n-1}) \quad (5.10)$$

As was done previously with linear and quadratic interpolation, data points can be used to evaluate the coefficients  $b_1, b_2, \dots, b_n$ . For an  $(n - 1)^{th}$ -order polynomial,  $n$  data points are required so that we can estimate:  $[x_1, f(x_1)], [x_2, f(x_2)], \dots, [x_n, f(x_n)]$ . We use these data points and the following equations to evaluate the coefficients:

$$b_1 = f(x_1) \quad (5.11)$$

$$b_2 = f[x_1, x_2] \quad (5.12)$$

$$b_3 = f[x_1, x_2, x_3] \quad (5.13)$$

$$\vdots \quad \vdots$$

$$b_n = f[x_n, x_{n-1}, \dots, x_2, x_1] \quad (5.14)$$

where the bracketed function evaluations are finite divided differences. For example, the first finite divided difference is represented generally as

$$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j} \quad (5.15)$$

The second finite divided difference, which represents the difference of two first divided differences, is expressed generally as

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k} \quad (5.16)$$

Similarly, the  $n^{th}$  finite divided difference is

$$f[x_n, x_{n-1}, \dots, x_2, x_1] = \frac{f[x_n, x_{n-1}, \dots, x_2] - f[x_{n-1}, x_{n-2}, \dots, x_1]}{x_n - x_1} \quad (5.17)$$

These differences can be used to evaluate the coefficients in Equation (5.11) through to Equation (5.14), which can then be substituted into Equation (5.10) to yield the general form of Newton's interpolating polynomial:

$$f_{n-1}(x) = f(x_1) + (x - x_1)f[x_2, x_1] + (x - x_1)(x - x_2)f[x_3, x_2, x_1] + \dots + (x - x_1)(x - x_2) \dots (x - x_{n-1})f[x_n, x_{n-1}, \dots, x_2, x_1] \quad (5.18)$$

We should note that it is not necessary that the data points used in Equation (5.18) be equally spaced. However, the points should be ordered so that they are centered around and as close as possible to the unknown. Also, notice how Equation (5.15)

through to Equation (5.17) are recursive, that is, higher-order differences are computed by taking differences of lower-order differences (Figure 5.4). This representation is referred to as a divided difference table.

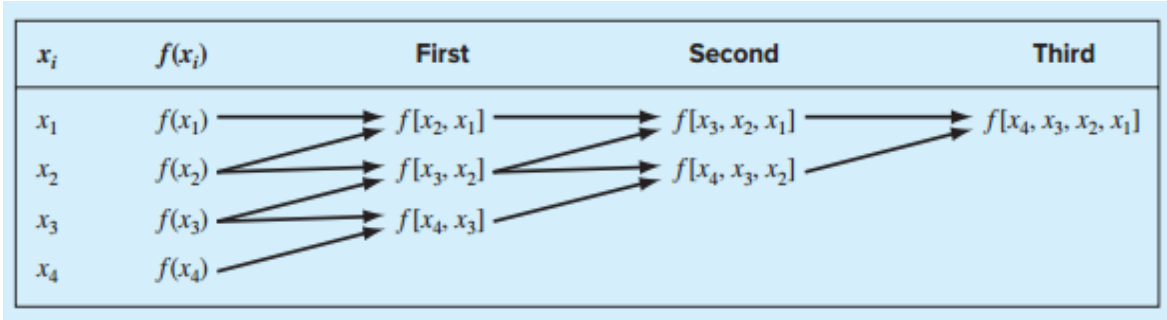


Figure 5.4 Recursive Nature of Finite Divided Differences

### Example 5.7 Newton Interpolating Polynomial

In Example 5.6, data points at  $x_1 = 1$ ,  $x_2 = 4$ , and  $x_3 = 6$  were used to estimate  $\ln 2$  with a parabola. Now, adding a fourth point  $[x_4 = 5; f(x_4) = 1.609438]$ , estimate  $\ln 2$  with a third-order Newton's interpolating polynomial.

#### Solution

The third-order polynomial, Equation (5.10) with  $n = 4$ , is

$$f_3(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2) + b_4(x - x_1)(x - x_2)(x - x_3)$$

The first divided differences for the problem is [Equation (5.15)]

$$f[x_2, x_1] = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \Rightarrow f[4, 1] = \frac{1.386294 - 0}{4 - 1} = 0.4620981$$

$$f[x_3, x_2] = \frac{f(x_3) - f(x_2)}{x_3 - x_2} \Rightarrow f[6, 4] = \frac{1.791759 - 1.386294}{6 - 4} = 0.2027326$$

$$f[x_4, x_3] = \frac{f(x_4) - f(x_3)}{x_4 - x_3} \Rightarrow f[5, 6] = \frac{1.609438 - 1.791759}{5 - 6} = 0.1823216$$

The second divided differences are [Equation (5.17)]

$$f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1} \Rightarrow f[6, 4, 1] = \frac{0.2027326 - 0.4620981}{6 - 1} = -0.05187311$$

$$f[x_4, x_3, x_2] = \frac{f[x_4, x_3] - f[x_3, x_2]}{x_4 - x_2} \Rightarrow f[5, 6, 4] = \frac{0.1823216 - 1.791759}{5 - 4} = -0.02041100$$

The third divided difference is [Equation (5.17) with  $n = 4$ ]

$$f[x_4, x_3, x_2, x_1] = \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{x_4 - x_1} \Rightarrow f[5, 6, 4, 1] = \frac{-0.02041100 - (-0.05187311)}{5 - 1} = 0.007865529$$

Thus, the divided difference table is shown as Table 5.2.

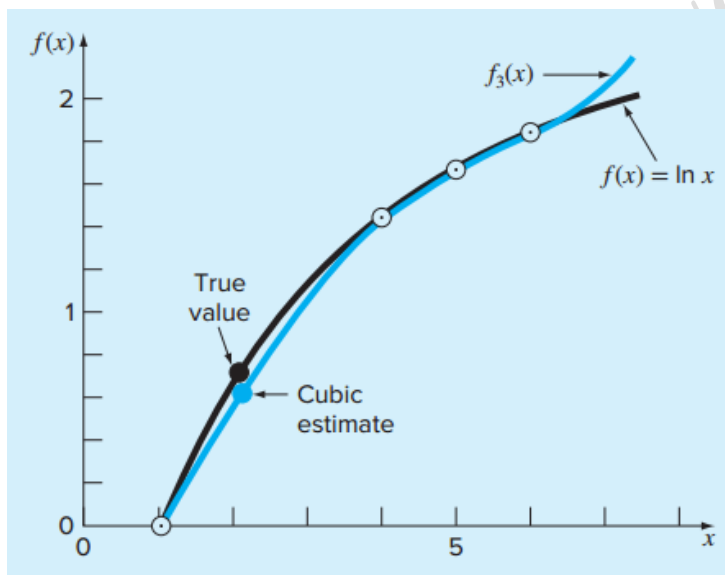
**Table 5.2 Divided Difference Table**

$x_i$	$f(x_i)$	First	Second	Third
1	0	0.4620981	-0.05187311	0.007865529
4	1.386294	0.2027326	-0.02041100	
6	1.791759	0.1823216		
5	1.609438			

The results for  $f(x_1)$ ,  $f[x_2, x_1]$ ,  $f[x_3, x_2, x_1]$  and  $f[x_4, x_3, x_2, x_1]$  represent the coefficients  $b_1$ ,  $b_2$ ,  $b_3$  and  $b_4$ , respectively, of Equation (5.10). Thus, the interpolating cubic is

$$f_3(x) = 0 + 0.4620981(x-1) - 0.05187311(x-1)(x-4) + 0.007865529(x-1)(x-4)(x-6)$$

which can be used to evaluate  $f_3(2) = 0.6287686$ , which represents a relative error of  $\varepsilon_t = 9.3\%$ . The complete cubic polynomial is shown in Figure 5.5.



**Figure 5.5 Cubic Interpolation to Estimate  $\ln 2$**

## 5.5 Lagrange Interpolating Polynomial

Suppose we formulate a linear interpolating polynomial as the weighted average of the two values that we are connecting by a straight line:

$$f(x) = L_1 f(x_1) + L_2 f(x_2) \quad (5.19)$$

where the  $L$ 's are the weighting coefficients. It is logical that the first weighting coefficient is the straight line that is equal to 1 at  $x_1$  and 0 at  $x_2$ :

$$L_1 = \frac{x - x_2}{x_1 - x_2} \quad (5.20)$$

Similarly, the second coefficient is the straight line that is equal to 1 at  $x_2$  and 0 at  $x_1$ :

$$L_2 = \frac{x - x_1}{x_2 - x_1} \quad (5.21)$$

Substituting these coefficients (Equations (5.20) and (5.21)) into Equation (5.19) yields the straight line that connects the points (Figure 5.6):

$$f_1(x) = \frac{x - x_2}{x_1 - x_2} f(x_1) + \frac{x - x_1}{x_2 - x_1} f(x_2) \quad (5.22)$$

where the nomenclature  $f_1(x)$  designates that this is a first-order polynomial. Equation (5.22) is referred to as the **linear Lagrange interpolating polynomial**. Each of the two terms of Equation (5.22) passes through one of the points and is zero at the other. The summation of the two terms must, therefore, be the unique straight line that connects the two points.

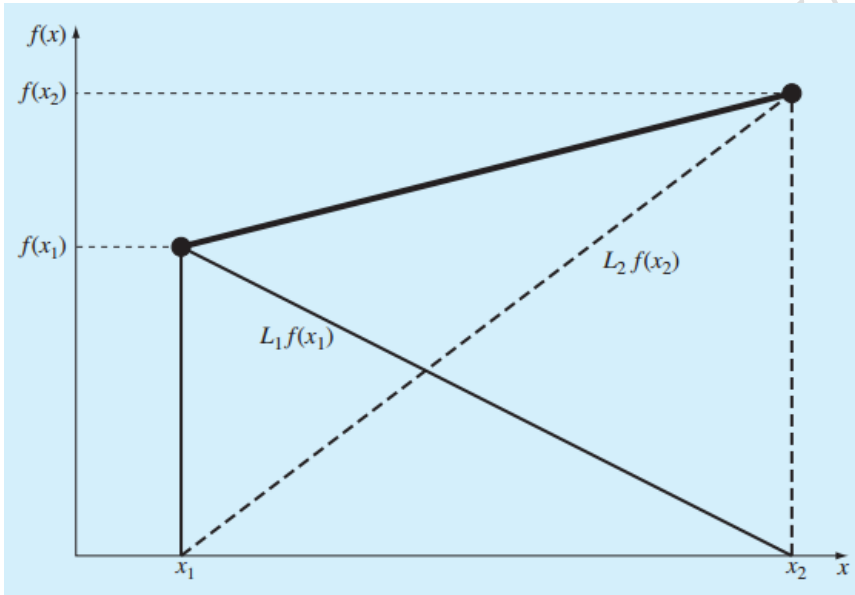


Figure 5.6 Rationale behind Lagrange Interpolating Polynomials (First-Order)

The same strategy can be employed to fit a parabola through three points. For this case three parabolas would be used with each one passing through one of the points and equaling zero at the other two. Their sum would then represent the unique parabola that connects the three points. Such a second-order Lagrange interpolating polynomial can be written as

$$f_2(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} f(x_1) + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} f(x_2) + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} f(x_3) \quad (5.23)$$

Notice how the first term is equal to  $f(x_1)$  at  $x_1$  and is equal to zero at  $x_2$  and  $x_3$ . The other terms work in a similar fashion.

Both the first- and second-order versions as well as higher-order Lagrange polynomials can be represented concisely as

$$f_{n-1}(x) = \sum_{i=1}^n L_i(x) f(x_i) \quad (5.24)$$

where

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}$$

where  $n$  is the number of data points and  $\prod$  designates the “product of”.

### Example 5.8 Lagrange Interpolating Polynomial

Use a Lagrange interpolating polynomial of the first and second order to evaluate the density of unused motor oil at  $T = 15^\circ\text{C}$  based on the following data:

$$\begin{aligned} x_1 = 0 & \quad f(x_1) = 3.85 \\ x_2 = 20 & \quad f(x_2) = 0.800 \\ x_3 = 40 & \quad f(x_3) = 0.212 \end{aligned}$$

#### Solution

The first-order polynomial [Equation (5.22)] can be used to obtain the estimate at  $x = 15$

$$f_1(x) = \frac{15-20}{0-20} \times 3.85 + \frac{15-0}{20-0} \times 0.800 = 1.5625$$

In a similar fashion, the second-order polynomial is developed as [Equation (5.23)]

$$\begin{aligned} f_2(x) &= \frac{(15-20)(15-40)}{(0-20)(0-40)} \times 3.85 + \frac{(15-0)(15-40)}{(20-0)(20-40)} \times 0.800 + \frac{(15-0)(15-20)}{(40-0)(40-20)} \times 0.212 \\ &= 1.3316875 \end{aligned}$$

#### 5.5.1 MATLAB M-file: Lagrange Interpolation

An M-file to implement the Lagrange Interpolation method is presented in **Appendix F**.

### 5.6 Try Exercises

1. Given the data

$x$	1	2	2.5	3	4	5
$f(x)$	0	5	7	6.5	2	0

- (a) Calculate  $f(3.4)$  using Newton's interpolating polynomials of order 1 through 3. Choose the sequence of the points for your estimates to attain the best

possible accuracy. That is, the points should be centered around and as close as possible to the unknown.

(b) Repeat (a) but use the Lagrange polynomial.

2. Given the data

$x$	1	2	3	5	6
$f(x)$	4.75	4	5.25	19.75	36

Calculate  $f(4)$  using Newton's interpolating polynomials of order 1 through 4. Choose your base points to attain good accuracy. That is, the points should be centered around and as close as possible to the unknown. What do your results indicate regarding the order of the polynomial used to generate the data in the table?

3. Repeat Question 2 using the Lagrange polynomial of order 1 through 3.

4. Table 5.3 lists values for dissolved oxygen concentration in water as a function of temperature and chloride concentration.

(a) Use quadratic and cubic interpolation to determine the oxygen concentration for  $T = 12^\circ\text{C}$  and  $c = 10\text{ g/L}$ .

(b) Use linear interpolation to determine the oxygen concentration for  $T = 12^\circ\text{C}$  and  $c = 15\text{ g/L}$ .

(c) Repeat (b) but use quadratic interpolation.

**Table 5.3 Dissolved Oxygen (mg/L)**

$T, ^\circ\text{C}$	$c = 0\text{ g/L}$	$c = 10\text{ g/L}$	$c = 20\text{ g/L}$
0	14.6	12.9	11.4
5	12.8	11.3	10.3
10	11.3	10.1	8.96
15	10.1	9.03	8.08
20	9.09	8.17	7.35
25	8.26	7.46	6.73
30	7.56	6.85	6.20

5. You measure the voltage drop  $V$  across a resistor for a number of different values of current  $i$ . The results are

$i$	0.5	1.5	2.5	3.0	4.0
$V$	-0.45	-0.6	0.70	1.88	6.0

Use first- through fourth-order polynomial interpolation to estimate the voltage drop for  $i = 2.3$ . Interpret your results.



6. Find the cubic polynomial which takes the following values:

$x$	0	1	2	3
$f(x)$	1	2	1	10

7. The sales in a particular shop for the last ten years is given in the table:

Year	1996	1998	2000	2002	2004
Sales (in cedis)	40	43	48	52	57

Use first- through third-order polynomial interpolation to estimate the sales for 2001. Interpret your results.

## CHAPTER 6

### NUMERICAL SOLUTION OF ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

#### 6.1 Objectives

The primary objective of this chapter is to introduce you to solving initial-value problems for Differential Equations (DEs). Specific objectives and topics covered are

- Understanding the meaning of local and global truncation errors and their relationship to step size for one-step methods for solving Ordinary DEs (ODEs).
- Knowing how to implement the following Runge-Kutta (RK) methods for a single ODE:
  - ✓ Euler
  - ✓ Fourth-order RK

#### 6.2 You've Got a Problem

This lecture material started with the problem of simulating the velocity of a free-falling bungee jumper. This problem amounted to formulating and solving an ordinary differential equation, the topic of this chapter. Now let's return to this problem and make it more interesting by computing what happens when the jumper reaches the end of the bungee cord.

To do this, we should recognize that the jumper will experience different forces depending on whether the cord is slack or stretched. If it is slack, the situation is that of free fall where the only forces are gravity and drag. However, in the case of stretched where the jumper can now move up as well as down, the sign of the drag force must be modified so that it always tends to retard velocity,

$$\frac{dv}{dt} = g - \text{sign}(v) \frac{c_d}{m} v^2 \quad (6.1)$$

where  $v$  is velocity (m/s),  $t$  is time (s),  $g$  is the acceleration due to gravity ( $9.81 \text{ m/s}^2$ ),  $c_d$  is the drag coefficient (kg/m), and  $m$  is mass (kg). The **signum function**,  $\text{sign}$ , returns a -1 or a +1 depending on whether its argument is negative or positive, respectively. Thus, when the jumper is falling downward (positive velocity,  $\text{sign} = +1$ ), the drag force will be negative and hence will act to reduce velocity. In contrast, when the jumper is moving upward (negative velocity,  $\text{sign} = -1$ ), the drag force will be positive so that it again reduces the velocity.

Once the cord begins to stretch, it obviously exerts an upward force on the jumper. The Hooke's law can be used as a first approximation of this force. In addition, a dampening force should also be included to account for frictional effects as the cord stretches and

contracts. These factors can be incorporated along with gravity and drag into a second force balance that applies when the cord is stretched. The result is the following differential equation:

$$\frac{dv}{dt} = g - \text{sign}(v) \frac{c_d}{m} v^2 - \frac{k}{m} (x - L) - \frac{\gamma}{m} v \quad (6.2)$$

where  $k$  is the cord's spring constant (N/m),  $x$  is vertical distance measured downward from the bungee jump platform (m),  $L$  is the length of the unstretched cord (m), and  $\gamma$  is a dampening coefficient (N.s/m).

Because Equation (6.2) only holds when the cord is stretched ( $x > L$ ), the spring force will always be negative. That is, it will always act to pull the jumper back up. The dampening force increases in magnitude as the jumper's velocity increases and always acts to slow the jumper down.

If we want to simulate the jumper's velocity, we would initially solve Equation (6.1) until the cord was fully extended. Then, we could switch to Equation (6.2) for periods that the cord is stretched. Although this is fairly straightforward, it means that knowledge of the jumper's position is required. This can be done by formulating another differential equation for distance:

$$\frac{dx}{dt} = v \quad (6.3)$$

Thus, solving for the bungee jumper's velocity amounts to solving two ordinary differential equations where one of the equations takes different forms depending on the value of one of the dependent variables. Such equations may be solved analytically to obtain an exact solution. However, many of such equations are either too difficult to solve or impossible to handle analytically. In such cases we employ numerical methods to obtain approximate solutions. Thus, this chapter explores methods for solving this and similar problems involving ODEs.

### 6.3 Introduction

A linear or nonlinear first order differential equation can always be solved numerically. A first order initial value problem has the form:

$$\frac{dy}{dt} = y' = f(t, y) \quad (6.4)$$

where  $f(t_i, y_i)$  is the differential equation evaluated at  $t_i$  and  $y_i$ , with initial values  $y(t_0) = y_0$ . The solution to the Equation (6.4) is plotted in Figure 6.1.

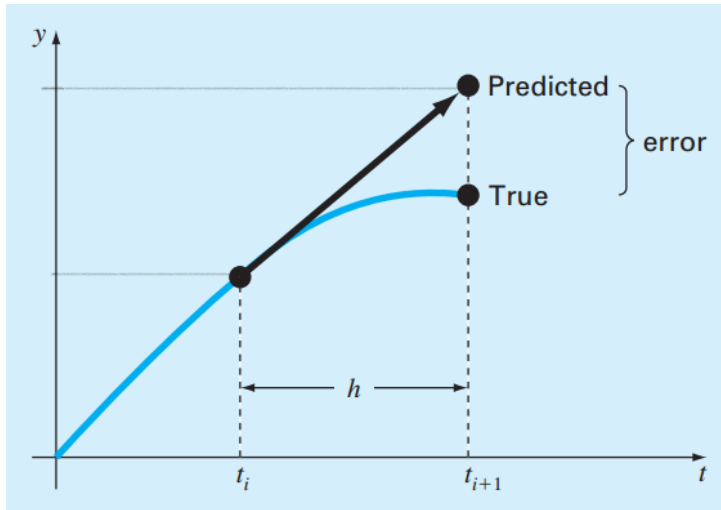
In Chapter 1, we developed a numerical method to solve such an equation for the velocity of the free-falling bungee jumper. Recall that the method was of the general form (refer to Equation (1.9))

$$\text{New value} = \text{old value} + \text{slope} \times \text{step size}$$

or, in mathematical terms,

$$y_{i+1} = y_i + y' h \quad (6.5)$$

where the slope  $y'$  is called an **increment function**. According to this equation, the slope estimate of  $y'$  is used to extrapolate from an old value  $y_i$  to a new value  $y_{i+1}$  over a distance  $h$ .



**Figure 6.1** Numerical evaluation of  $\frac{dy}{dt} = y' = f(t, y)$

Most of such numerical methods (all one-step methods) can be expressed in the general form of Equation (6.5), with the only difference being the manner in which the slope  $y'$  is estimated. The simplest approach is to use the differential equation to estimate the slope in the form of the first derivative at  $t_i$ . In other words, the slope at the beginning of the interval is taken as an approximation of the average slope over the whole interval. In subsequent sections, some numerical methods for solving Equation (6.4) are discussed.

## 6.4 Euler Method



Euler's method, although is seldom used in practice, the simplicity of its derivation can be used to illustrate the techniques involved in the construction of some of the more advanced techniques, without the cumbersome algebra that accompanies these constructions.

Euler's method is not very accurate. This is due to the fact that only the first two terms of the Taylor's series are considered, leading to the exclusion of many terms. The method, like many numerical methods, has its greatest error when trying to obtain solutions that are rapidly changing. The accuracy of the Euler method can be improved by using a smaller step size. However, very small step-size require longer run times and

can result in large accumulated error because of round-off effects. Euler's method may be derived in several ways (**Assignment**).

The first derivative provides a direct estimate of the slope at  $t_i$  (Figure 6.1):

$$\frac{dy}{dt} = y' = f(t_i, y_i) \quad (6.6)$$

where  $f(t_i, y_i)$  is the differential equation evaluated at  $t_i$  and  $y_i$ , with initial values  $y(t_0) = y_0$ . This estimate from Equation (6.6) can be substituted into Equation (6.5):

$$y_{i+1} = y_i + f(t_i, y_i)h \quad (6.7)$$

Equation (6.7) is referred to as Euler's method (or the Euler-Cauchy or point-slope method). A new value of  $y$  is predicted using the slope (equal to the first derivative at the original value of  $t$ ) to extrapolate linearly over the step size  $h$  (Figure 6.1).

### Example 6.1 Euler's Method

Use Euler's method to obtain the approximate value of  $y(1.5)$  for the solution of the initial value problem  $y' = 2xy$ ,  $y(1) = 1$ . Use a step-size of 0.1.

#### Solution

$$y_{n+1} = y_n + h(2x_n y_n) \quad \text{for } x_0 = 1, y_0 = 1$$

#### 1<sup>st</sup> Iteration

$$y_1 = y_0 + h(2x_0 y_0)$$

$$y_1 = 1 + 0.1(2(1)(1))$$

$$\therefore y_1 = 1.2$$

#### 2<sup>nd</sup> Iteration

$$y_2 = y_1 + h(2x_1 y_1)$$

$$y_2 = 1.2 + 0.1(2(1.1)(1.2))$$

$$\therefore y_2 = 1.4640$$

#### 3<sup>rd</sup> Iteration

$$y_3 = y_2 + h(2x_2 y_2)$$

$$y_3 = 1.4640 + 0.1(2(1.2)(1.4640))$$

$$\therefore y_3 = 1.8154$$

#### 4<sup>th</sup> Iteration

$$y_4 = y_3 + h(2x_3 y_3)$$

$$y_4 = 1.8154 + 0.1(2(1.3)(1.8154))$$

$$\therefore y_4 = 2.2874$$

#### 5<sup>th</sup> Iteration

$$y_5 = y_4 + h(2x_4y_4)$$

$$y_5 = 2.2874 + 0.1(2(1.4)(2.2874))$$

$$\therefore y_5 = 2.9278$$

The result is summarized in Table 6.1.

**Table 6.1 Summary of Euler's Method Computation**

Iteration	$x_n$	$y_n$	$y_{n+1}$
1	1.0	1.0000	1.2000
2	1.1	1.2000	1.4640
3	1.2	1.4640	1.8154
4	1.3	1.8154	2.2874
5	1.4	2.2874	2.9278

#### Example 6.2 Euler's Method

Solve the equation  $y' = (t^2 - y^2) \sin y$ ,  $y(0) = -1$ , for  $0 \leq t \leq 0.4$ , in steps of 0.05 by Euler's method.

#### Solution

$$y' = (t^2 - y^2) \sin y, \quad y(0) = -1$$

$$y_{n+1} = y_n + h(t_n^2 - y_n^2) \sin y_n$$

#### 1<sup>st</sup> Iteration

$$y_1 = y_0 + h(t_0^2 - y_0^2) \sin y_0$$

$$y(0.05) = -1 + 0.05(0 - (-1)^2) \sin(-1)$$

$$\therefore y(0.05) = -0.957926$$

#### 2<sup>nd</sup> Iteration

$$y_2 = y_1 + h(t_1^2 - y_1^2) \sin y_1$$

$$y(0.10) = -0.957926 + 0.05((0.05)^2 - (-0.957926)^2) \sin(-0.957926)$$

$$\therefore y(0.10) = -0.920497$$

**3<sup>rd</sup> Iteration**

$$y_3 = y_2 + h(t_2^2 - t_2^2) \sin y_2$$

$$y(0.15) = -0.920497 + 0.05((0.1)^2 - (-0.920497)^2) \sin(-0.920497)$$

$$\therefore y(0.15) = -0.887176$$

**4<sup>th</sup> Iteration**

$$y_4 = y_3 + h(t_3^2 - t_3^2) \sin y_3$$

$$y(0.2) = -0.887176 + 0.05((0.15)^2 - (-0.887176)^2) \sin(-0.887176)$$

$$\therefore y(0.2) = -0.857538$$

**5<sup>th</sup> Iteration**

$$y_5 = y_4 + h(t_4^2 - t_4^2) \sin y_4$$

$$y(0.25) = -0.857538 + 0.05((0.2)^2 - (-0.857538)^2) \sin(-0.857538)$$

$$\therefore y(0.25) = -0.831245$$

**6<sup>th</sup> Iteration**

$$y_6 = y_5 + h(t_5^2 - t_5^2) \sin y_5$$

$$y(0.3) = -0.831245 + 0.05((0.25)^2 - (-0.831245)^2) \sin(-0.831245)$$

$$\therefore y(0.3) = -0.808030$$

**7<sup>th</sup> Iteration**

$$y_7 = y_6 + h(t_6^2 - t_6^2) \sin y_6$$

$$y(0.35) = -0.808030 + 0.05((0.3)^2 - (-0.808030)^2) \sin(-0.808030)$$

$$\therefore y(0.35) = -0.787683$$

**8<sup>th</sup> Iteration**

$$y_8 = y_7 + h(t_7^2 - t_7^2) \sin y_7$$

$$y(0.4) = -0.787683 + 0.05((0.35)^2 - (-0.787683)^2) \sin(-0.787683)$$

$$\therefore y(0.4) = -0.770038$$

The results are summarised in Table 6.2.

**Table 6.2 Summary of Euler's Method Computation**

Iteration	$t$	$y$
	0	-1.000000
1	0.05	-0.957926
2	0.10	-0.920497
3	0.15	-0.887176
4	0.20	-0.857538
5	0.25	-0.831245
6	0.30	-0.808030
7	0.35	-0.787683
8	0.40	-0.770038

#### 6.4.1 MATLAB M-file: Euler's Method

An M-file to implement the Euler's method is presented in **Appendix G**.

### 6.5 The Runge-Kutta (RK) Methods



We have so far considered the Euler. However, Euler's is still not accurate enough for serious numerical work.

Methods that are both relatively simple and also sufficiently accurate to be widely useful are the Runge-Kutta methods. The most popular RK methods are fourth order. The following is the most commonly used form, and we therefore call it the **classical fourth order RK method**.

#### 6.5.1 Classical Fourth Order Runge-Kutta (RK4) Method

The Runge-Kutta formula involves a weighted average value of  $f(t, y)$  taken at different points in the interval  $t_i \leq t \leq t_{i+1}$ . It is given by

$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \quad (6.8)$$

where

$$k_1 = f(t_i, y_i) \quad (6.9)$$

$$k_2 = f\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1h\right) \quad (6.10)$$

$$k_3 = f\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2h\right) \quad (6.11)$$

$$k_4 = f(t_i + h, y_i + k_3h) \quad (6.12)$$

In addition, the fourth order RK method estimates the slope to come up with an improved average slope for the interval. As depicted in Figure 6.2, each of the  $k$ 's



represents a slope. Equation (6.8) then represents a weighted average of these to arrive at the improved slope.

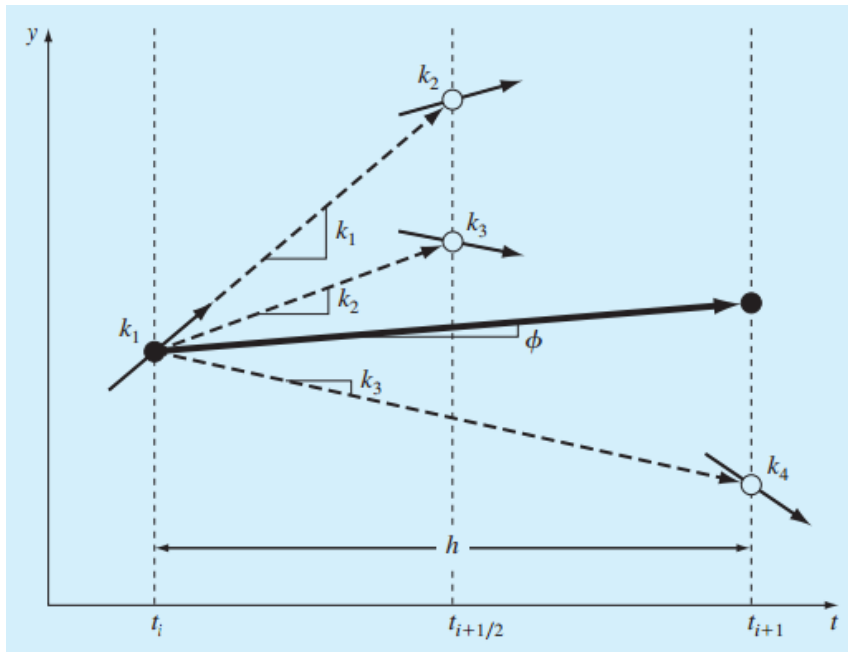


Figure 6.2 Slope Estimates of RK4 Method

### Example 6.3 4<sup>th</sup> Order Runge-Kutta (RK4) Method

Use the 4<sup>th</sup> order Runge-Kutta method to obtain the approximate value of  $y(1.5)$  for the solution of the initial value problem  $y' = 2xy$ ,  $y(1) = 1$ . Use a step-size of 0.1.

#### Solution

$$y' = 2xy, \quad y(1) = 1$$

This implies that  $f(x, y) = 2xy$

#### 1<sup>st</sup> Iteration

$$k_1 = f(x_n, y_n) = 2x_n y_n = 2(1)(1) = 2.000000$$

$$\begin{aligned} k_2 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\ &= f\left(1 + 0.05, 1 + 0.05(2)\right) = f(1.05, 1.1) = 2(1.05)(1.1) = 2.310000 \end{aligned}$$

$$\begin{aligned} k_3 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \\ &= f\left(1 + 0.05, 1 + 0.05(2.31)\right) = f(1.05, 1.1155) = 2(1.05)(1.1155) = 2.342550 \end{aligned}$$

$$\begin{aligned} k_4 &= f(x_n + h, y_n + hk_3) \\ &= f(1 + 0.1, 1 + 0.1(2.342550)) = f(1.1, 1.234255) = 2(1.1)(1.234255) = 2.715361 \end{aligned}$$

$$\therefore y_1 = y_0 + \frac{0.1}{6} [k_1 + 2k_2 + 2k_3 + k_4] = 1 + \frac{0.1}{6} [2 + 2(2.31) + 2(2.34255) + 2.715361]$$

$$\therefore y(1.1) = 1.233674$$

### 2<sup>nd</sup> Iteration

Now  $x_n = 1.1$ ;  $y_n = 1.233674$

$$k_1 = f(x_n, y_n) = 2(1.1)(1.233674) = 2.714083$$

$$k_2 = f\left(1.1 + \frac{1}{2}(0.1), 1.233674 + \frac{1}{2}(0.1)(2.714083)\right)$$

$$= f(1.15, 1.369378) = 2(1.15)(1.369378) = 3.149569$$

$$k_3 = f\left(1.1 + \frac{1}{2}(0.1), 1.233674 + \frac{1}{2}(0.1)(3.149569)\right)$$

$$= f(1.15, 1.391152) = 2(1.15)(1.391152) = 3.199650$$

$$k_4 = f(1.1 + 0.1, 1.233674 + (0.1)(3.199650))$$

$$= f(1.2, 1.553639) = 2(1.2)(1.553639) = 3.728734$$

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

$$= 1.233674 + \frac{0.1}{6} [2.714083 + 2(3.149569) + 2(3.199650) + 3.728734]$$

$$\therefore y(1.2) = 1.552695$$

### 3<sup>rd</sup> Iteration

Now  $x_n = 1.2$ ,  $y_n = 1.552695$

$$k_1 = f(x_n, y_n) = 2(1.2)(1.552695) = 3.726468$$

$$k_2 = f\left(1.2 + \frac{1}{2}(0.1), 1.552695 + \frac{1}{2}(0.1)(3.726468)\right)$$

$$= f(1.25, 1.739018) = 2(1.25)(1.739018) = 4.347545$$

$$k_3 = f\left(1.2 + \frac{1}{2}(0.1), 1.552695 + \frac{1}{2}(0.1)(4.347545)\right)$$

$$= f(1.25, 1.770072) = 2(1.25)(1.770072) = 4.425180$$

$$k_4 = f(1.2 + 0.1, 1.552695 + (0.1)(4.425180))$$

$$= f(1.3, 1.995213) = 2(1.3)(1.995213) = 5.187554$$

$$\therefore y_3 = y_2 + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$= 1.552695 + \frac{0.1}{6}[3.726468 + 2(4.347545) + 2(4.42518) + 5.187554]$$

$$\therefore y(1.3) = 1.993686$$

#### 4<sup>th</sup> Iteration

Now  $x_n = 1.3$ ,  $y_n = 1.993686$

$$k_1 = f(x_n, y_n) = 2(1.3)(1.993686) = 5.183584$$

$$k_2 = f\left(1.3 + \frac{1}{2}(0.1), 1.993686 + \frac{1}{2}(0.1)(5.183584)\right)$$

$$= f(1.35, 2.252865) = 2(1.35)(2.252865) = 6.082736$$

$$k_3 = f\left(1.3 + \frac{1}{2}(0.1), 1.993686 + \frac{1}{2}(0.1)(6.082736)\right)$$

$$= f(1.35, 2.297823) = 2(1.35)(2.297823) = 6.204122$$

$$k_4 = f(1.3 + 0.1, 1.993686 + (0.1)(6.204122))$$

$$= f(1.4, 2.614098) = 2(1.4)(2.614098) = 7.319474$$

$$\therefore y_4 = y_3 + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$= 1.993686 + \frac{0.1}{6}[5.183584 + 2(6.082736) + 2(6.204122) + 7.319474]$$

$$\therefore y(1.4) = 2.611632$$

#### 5<sup>th</sup> Iteration

Now  $x_n = 1.4$ ,  $y_n = 2.611632$

$$k_1 = f(x_n, y_n) = 2(1.4)(2.611632) = 7.312570$$

$$k_2 = f\left(1.4 + \frac{1}{2}(0.1), 2.611632 + \frac{1}{2}(0.1)(7.312570)\right)$$

$$= f(1.45, 2.977261) = 2(1.45)(2.977261) = 8.634057$$

$$k_3 = f\left(1.4 + \frac{1}{2}(0.1), 2.611632 + \frac{1}{2}(0.1)(8.634057)\right)$$

$$\begin{aligned}
&= f(1.45, 3.043335) = 2(1.45)(3.043335) = 8.825672 \\
k_4 &= f(1.4 + 0.1, 2.611632 + (0.1)(8.825672)) \\
&= f(1.5, 3.494199) = 2(1.5)(3.494199) = 10.482597 \\
\therefore y_5 &= y_4 + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4] \\
&= 2.611632 + \frac{0.1}{6}[7.312570 + 2(8.634057) + 2(8.825672) + 10.482597] \\
\therefore y(1.5) &= 3.490209
\end{aligned}$$

#### Example 6.4 4<sup>th</sup> Order Runge-Kutta (RK4) Method

Solve the equation  $y' = x^2 + y^3$ ,  $y(1) = 1$ , for  $y(1.2)$  in steps of 0.1.

#### Solution

$$y' = x^2 + y^3$$

#### 1<sup>st</sup> Iteration

$$k_1 = f(x_0, y_0) = 1^2 + 1^3 = 2$$

$$k_2 = f(1.05, 1.1) = (1.05)^2 + (1.1)^3 = 2.433500$$

$$k_3 = f(1.05, 1.121675) = (1.05)^2 + (1.121675)^3 = 2.513741$$

$$k_4 = f(1.1, 1.251374) = (1.1)^2 + (1.251374)^3 = 3.169573$$

$$\therefore y(1.1) = 1 + \frac{0.1}{6}[2.3 + 2(2.4335) + 2(2.513741) + 3.169573] = 1.251068$$

#### 2<sup>nd</sup> Iteration

Now  $x_n = 1.1$ ,  $y_n = 1.251068$

$$k_1 = f(1.1, 1.251068) = (1.1)^2 + (1.251068)^3 = 3.168136$$

$$k_2 = f(1.15, 1.409475) = (1.15)^2 + (1.409475)^3 = 4.122591$$

$$k_3 = f(1.15, 1.457198) = (1.15)^2 + (1.457198)^3 = 4.416752$$

$$k_4 = f(1.2, 1.692745) = (1.2)^2 + (1.692745)^3 = 6.290427$$

$$\begin{aligned}
\therefore y(1.2) &= 1.251068 + \frac{0.1}{6}[3.168136 + 2(4.122591) + 2(4.416752) + 6.290427] \\
&= 1.693355
\end{aligned}$$

### 6.5.2 MATLAB M-file: 4th Order Runge-Kutta

An M-file to implement the RK4 method is presented in **Appendix H**.

### 6.6 Try Exercise

Solve each of the following initial value problems by using Euler's method:

1.  $x' = t^{-2}(tx - x^2)$ ,  $x(1) = 2$ , on the interval  $[1, 3]$  using  $h = \frac{1}{128}$ .
2. Solve,  $y' = 2t + e^{-ty}$ ,  $y(0) = 1$  on the interval  $[0, 0.4]$ , using  $h = 0.1$ .
3. Solve,  $y' = 1 - t + 4y$ ,  $y(0) = 1$  on the interval  $0 \leq t \leq 1$ , in steps of 0.1
4. Solve  $y' = y + \cos x$ ,  $y(0) = 1$ , using  $h = 0.1$  on the interval  $[0, 1]$ .

Solve each of the following initial value problems by using 4<sup>th</sup> order Range-Kutta method:

5.  $y' = (t^2 - y^2) \sin y$ ,  $y(0) = -1$ , for  $0 \leq t \leq 0.4$ , in steps of  $h = 0.5$ .
6.  $y' = 1 - t + 4y$ ,  $y(0) = 1$ , for  $0 \leq t \leq 1$  in steps of
  - i. 0.1
  - ii. 0.05
7.  $x' = t^{-2}(tx - x^2)$ ,  $x(1) = 2$ , on the interval  $[1, 3]$  using  $h = \frac{1}{128}$ .
8.  $y' = \frac{2xy}{x^2 - y^2}$ ,  $y(1) = 3$ ,  $h = 0.05$ , on the interval  $1 \leq x \leq 1.5$ .
9.  $y' = \frac{y^2 + 2ty}{3 + t^2}$ ,  $y(1) = 2$ ,  $h = 0.1$ ,  $h = 0.1$  on the interval  $1 \leq x \leq 1.6$

## REFERENCES

- Burden, R. L. and Faires, J. D. (2005), *Numerical Analysis*, 8<sup>th</sup> edition, Bob Pirtle, USA, 867 pp.
- Burden, R. L., Faires, J. D. and Burden, A. M. (2005), *Numerical Analysis*, 10<sup>th</sup> edition, Cengage Learning, Boston, USA, 918 pp.
- Chapra, S. C. (2018), *Applied Numerical Methods with MATLAB for Engineers and Scientists*, 4<sup>th</sup> edition, McGraw-Hill Education, New York, 714 pp.
- Sauer, T. (2012), *Numerical Analysis*, 2<sup>nd</sup> edition, Pearson, USA, 665 pp.
- Wiah, E. N. (2018), *Numerical Analysis*, BSc Lecture Note, University of Mines and Technology, Tarkwa, Ghana, 52 pp.

## APPENDICES

### Appendix A An M-file to implement the Bisection Method

```
% Bisection Method in MATLAB
a=input('Enter function with right hand side zero:', 's');
f=inline(a);

xl=input('Enter the first value of guess interval:') ;
xu=input('Enter the end value of guess interval:');
tol=input('Enter the allowed error:');

if f(xu)*f(xl)<0
else
    fprintf('The guess is incorrect! Enter new guesses\n');
    xl=input('Enter the first value of guess interval:\n') ;
    xu=input('Enter the end value of guess interval:\n');
end

for i=2:1000
    xr=(xu+xl)/2;
    if f(xu)*f(xr)<0
        xl=xr;
    else
        xu=xr;
    end

    if f(xl)*f(xr)<0
        xu=xr;
    else
        xl=xr;
    end

    xnew(1)=0;
    xnew(i)=xr;
    if abs((xnew(i)-xnew(i-1))/xnew(i))<tol, break, end
end
str = ['The required root of the equation is: ',
num2str(xr), '']
```

## Appendix B An M-file to implement the Regula Falsi Method

```
function a = regula_falsi( f )
%for Regula Falsi method
% f is a function for which root to be found.
% asking for the range
f = @(x) x^3 -2*x - 5;
R= input ( 'You are looking for the roots in [ x_min ,
x_max] :\n');
% check for range
[ nr , mr ] = size( R);
if nr ~= 1 || mr~= 2
    disp( 'Input error ..The matrix should be 1x2 matrix')
    return
end
% if root lies in the boundary
if feval( f , R( 1,1) )* feval( f , R(1,2)) == 0
    if feval( f , R( 1,1) ) == 0
        R(1,1)
        return
    else
        feval( f , R(1,2)) == 0
        R(1,2)
        return
    end
end
% condition for convergence
if feval( f , R( 1,1) )* feval( f , R(1,2)) > 0
    disp ( 'Either NO root lies in the given range or EVEN
no of roots')
    disp( 'lie in the given range and hence this method
cannot be applied.');
```

return

end

%error allowed in final answer

tol = abs(input(' Enter the error allowed in the final answer :'));

% no of iterations to be performed

n = input('Enter the maximum number of iterations to be performed :');

%initializing the value of k and matrix X

k=1;

X= zeros(n+1,3);

%initial display of table & initializing values for look

disp(sprintf('\t iterate \t value of x \t error'));

x0= R(1,1); x1= R(1,2); x\_disp= x0; err = x1-x0;

disp(sprintf ('\t %3d \t %11.5f \t %11.5f ', 0, x\_disp,err));

% iteration loop starts

while k <=n && abs(err) > tol



```

    x = x1 - (x1-x0)/( feval(f,x1)-feval(f,x0) )
*feval(f,x1);%REGULA FALSI formula
    if feval(f , x0) * feval(f , x) == 0
        x
        return
    else
        if feval(f,x0) * feval(f,x) <0
            err = x - x1;
            x1 = x;
            x_disp=x1;
            X(k,2) = x1;
        else
            err = x-x0;
            x0 = x;
            x_disp = x0;
            X(k,2) = x0;
        end
    end
    % storing values in the form of matrix
    X(k,1) = k;
    X(k,3) = abs(err);
    disp(sprintf ('\t %3d \t %11.5f \t %11.5f ', k,
x_disp,err));
    k = k + 1;
end
if abs(err) > tol
    disp(sprintf ('The final answer obtained after %3d
iterations is %10.10f with an error %10.10f \n' , n ,
X(n,2),X(n,3)))
    disp('Try more no of iterations for desired accuracy')
else
    disp(sprintf ('The final answer obtained after %3d
iterations is %10.10f with an error %10.10f \n' , (k-1) ,
X((k-1),2),X((k-1),3)))
end

```

## Appendix C An M-file to implement the Newton Rapson Method

% Program Code of Newton-Raphson Method in MATLAB

```
a=input('Enter the function in the form of variable  
x:', 's');  
x(1)=input('Enter Initial Guess:');  
error=input('Enter allowed Error:');  
f=inline(a)  
dif=diff(str2sym(a));  
d=inline(dif);  
  
for i=1:100  
    x(i+1)=x(i)-((f(x(i))/d(x(i))));  
    err(i)=abs((x(i+1)-x(i))/x(i));  
    if err(i)<error  
        break  
    end  
end  
root=x(i)
```

## Appendix D An M-file to implement the Secant Method

```
% Secant Method in MATLAB

a=input('Enter function:', 's');
f=inline(a)

x(1)=input('Enter first point of guess interval: ');
x(2)=input('Enter second point of guess interval: ');
n=input('Enter allowed Error in calculation: ');
iteration=0;

for i=3:1000
    x(i) = x(i-1) - (f(x(i-1)))*((x(i-1) - x(i-2))/(f(x(i-1)) - f(x(i-2))));
    iteration=iteration+1;
    if abs((x(i)-x(i-1))/x(i))*100<n
        root=x(i)
        iteration=iteration
        break
    end
end
```

## Appendix E An M-file to implement the Gauss-Seidel Method

```
% Gauss-Seidel Method in MATLAB

function x = gauss_siedel( A ,B )
disp ( 'Enter the system of linear equations in the form of
AX=B' )

%Inputting matrix A

A = input ( 'Enter matrix A :  \n' )
% check if the entered matrix is a square matrix

[na , ma ] = size (A);
if na ~= ma
    disp('ERROR: Matrix A must be a square matrix')
    return
end

% Inputting matrix B

B = input ( 'Enter matrix B :  ' )
% check if B is a column matrix

[nb , mb ] = size (B);
if nb ~= na || mb~=1
    disp( 'ERROR: Matrix B must be a column matrix')
    return
end

% Separation of matrix A into lower triangular and upper
triangular matrices

%  $A = D + L + U$ 

D = diag(diag(A));
L = tril(A) - D;
U = triu(A) - D

% check for convergence condition for Gauss-Seidel method

e= max(eig(-inv(D+L)*(U)));
if abs(e) >= 1
    disp ( 'Since the modulus of the largest Eigen value of
iterative matrix is not less than 1')
    disp ( 'this process is not convergent.')
    return
end

% initial guess for X..?
```

```

% default guess is [ 1 1 .... 1]

r = input ( 'Any initial guess for X? (y/n):  ','s');
switch r
    case 'y'
        % asking for initial guess
        X0 = input('Enter initial guess for X :\n')
        % check for initial guess
        [nx, mx] = size(X0);
        if nx ~= na || mx ~= 1
            disp( 'ERROR: Check input')
            return
        end
    otherwise
        X0 = ones(na,1);
end

% allowable error in final answer

t = input ( 'Enter the error allowed in final answer:  ');
tol = t*ones(na,1);
k= 1;

X( : , 1 ) = X0;
err= 1000000000*rand(na,1);% initial error assumption for
looping
while sum(abs(err) >= tol) ~= zeros(na,1)
    X ( : ,k+ 1 ) = -inv(D+L)*(U)*X( : ,k) + inv(D+L)*B;%
Gauss-Seidel formula
    err = X( :,k+1) - X( :, k);% finding error
    k = k + 1;

end

fprintf ('The final answer obtained after %g iterations is
\n', k)
X( : ,k)

```

## Appendix F An M-file to implement the Lagrange Interpolation

```
% Lagrange Interpolation MATLAB Program
function [P,R,S] = lagrangepoly(X,Y,XX)
X = [1 2 3 4 5 6 7 8]; % inputting the values of given x
Y = [0 1 0 1 0 1 0 1]; % inputting the values of given y
%[P,R,S] = lagrangepoly(X,Y);
xx = 0.5 : 0.01 : 8.5;
%plot(xx,polyval(P,xx),X,Y,'or',R,S,'.b',xx,spline(X,Y,xx),'
--g')
%grid
%axis([0.5 8.5 -5 5])
if size(X,1) > 1; X = X'; end % checking for parameters
if size(Y,1) > 1; Y = Y'; end
if size(X,1) > 1 || size(Y,1) > 1 || size(X,2) ~= size(Y,2)
    error('both inputs must be equal-length vectors') %
displaying error
end % end of scope of if
N = length(X);
pvals = zeros(N,N);
% for evaluating the polynomial weights for each order
for i = 1:N
    % the polynomial with roots may be values of X other
    than this one
    pp = poly(X( (1:N) ~= i));
    pvals(i,:) = pp ./ polyval(pp, X(i));
end
P = Y*pvals;
if nargin==3
    YY = polyval(P,XX); % output is YY with given XX
    P = YY; % assigning to output
end
% end of scope of if
if nargin > 1 % checking for conditions
    R = roots( ((N-1):-1:1) .* P(1:(N-1)) );
    if nargin > 2
        % the evaluation of actual value at the points of
        zero derivative
        S = polyval(P,R);
    end
end
```

## Appendix G An M-file to implement the Euler's Method

```
%function t=t(n,t0,t1,y0)

function y=y(n,t0,t1,y0)
h=(t1-t0)/n;
t(1)=t0;
y(1)=y0;
for i=1:n
    t(i+1)=t(i)+h;
    y(i+1)=y(i)+h*ex(t(i),y(i));
end;
V=[t',y']
plot(t,y)
title(' Euler Method')
```

## Appendix H An M-file to implement the Runge-Kutta Method

```
function a = runge_kutta(df)
% asking initial conditions
x0 = input('Enter initial value of x : ');
y0 = input('Enter initial value of y : ');
x1 = input('Enter value of x at which y is to be calculated
: ');
tol = input('Enter desired level of accuracy in the final
result : ');

%choose the order of Runge-Kutta method
% calculating the value of h
n =ceil( (x1-x0)/nthroot(tol,3));
h = ( x1 - x0)/n;
for i = 1 : n
    X(1,1) = x0; Y (1,1) = y0;
    k1 = h*feval( df , X(1,i), Y(1,i));
    k2 = h*feval( df , X(1,i) + h/2, Y(1,i) + k1);
    k3 = h*feval( df , X(1,i) + h/2, Y(1,i) + k2);
    k4 = h*feval( df , X(1,i) + h, Y(1,i) + k3);
    k = 1/6* ( k1+ 2*k2 + 2*k3 + k4);
    X( 1, i+1) = X(1,i) + h;
    Y( 1 ,i+1) = Y(1,i) + k;
end
%displaying results
fprintf( 'for x = %g \n y = %g \n' , x1,Y(1,n+1))

%displaying graph
x = 1:n+1;
y = Y(1,n+1)*ones(1,n+1) - Y(1,:);
plot(x,y,'r')
xlabel = (' no of interval ');
ylabel = (' Error ');
```