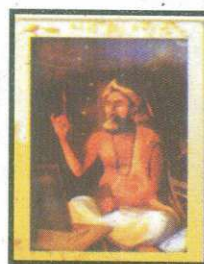




Babylonian clay tablet
YBC 7289 for value of
 $\sqrt{2}$ (c.1800-1600 BC)



Aryabhata (c. 476-550 AD)



Varahmihira (c. 505-587 AD)



Babylonian Mathematical
clay tablet Pimpton 322
(c.1800 BC)

Vedic Period: Sulbha/
Shulbha sutras composed
by **Baudhayana** (c. 800
BCE), **Manav** (c. 750 BCE),
Apastamba (c. 600 BCE)
and **Katyayana** (3rd Century,
BCE) contain, among other
mathematical contributions,
examples of Pythagorean
triples and formula for the
value of $\sqrt{2}$ giving accuracy
up to 5 places after the
decimal.



Brahmagupta
(c. 598-670 AD)



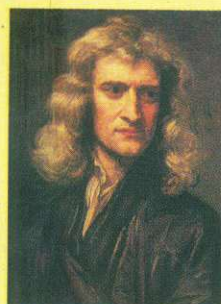
Muhammad ibn Musa
al-Khwarizmi
(c.780-c.850 AD)



A page from Hisab al-jabar
w' al Muqabala by
al-Khwarizmi (c. 820 AD)



Fragment from Euclid's
Elements (c.100 AD)



Isaac Newton
(1642-1727 AD)



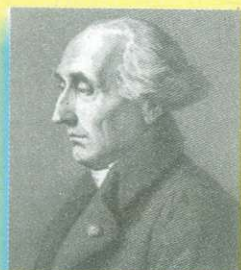
Leonhard Euler
(1707-1783 AD)



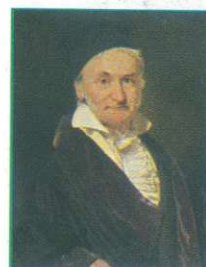
A page from Bakhshali
Manuscript (Between
2nd Cent. BC, 3rd Cent. AD)



A page from (Chinese): The Nine
Chapters on the Mathematical
Art (c. 2nd Cent. AD)



Joseph-Louis Lagrange
(1736-1813 AD)



Johann Carl Friedrich Gauss
(1777-1855 AD)



A page from Ganita-Kaumudi
(1356 AD) on Magic Squares

“शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्र की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।”

— इन्दिरा गांधी

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

- Indira Gandhi

Block

1

COMPUTER ARITHMETIC AND SOLUTION OF LINEAR AND NON-LINEAR EQUATIONS

UNIT 0

Overview of Macro-Issues in 'Numerical Analysis and Techniques'	5
--	----------

UNIT 1

Computer Arithmetic	27
----------------------------	-----------

UNIT 2

Solution of Linear Algebraic Equations	59
---	-----------

UNIT 3

Solution of Non-Linear Equations	83
---	-----------

Appendix	97
-----------------	-----------

PROGRAMME DESIGN COMMITTEE

Prof. Manohar Lal
SOCIS, IGNOU, New Delhi

Prof. H. M. Gupta
Dept. of Elect. Engg., IIT, Delhi

Prof. M. N. Doja, Dept. of CE
Jamia Millia, New Delhi

Prof. C. Pandurangan
Dept. of CSE, IIT, Chennai

Prof. I. Ramesh Babu
Dept. of CSE
Acharya Nagarjuna University
Nagarjuna Nagar (AP)

Prof. N. S. Gill
Dept. of CS, MDU, Rohtak

Prof. Arvind Kalia, Dept. of CS
HP University, Shimla

Prof. Anju Sahgal Gupta
SOH, IGNOU, New Delhi

Prof. Sujatha Varma
SOS, IGNOU, New Delhi

Prof. V. Sundarapandian
IITMK, Trivandrum

Prof. Dharamendra Kumar
Dept. of CS, GJU, Hissar

Prof. Vikram Singh
Dept. of CS, CDLU, Sirsa

Sh. Shashi Bhushan, Associate. Prof.
SOCIS, IGNOU, New Delhi

Sh. Akshay Kumar, Associate Prof.
SOCIS, IGNOU, New Delhi

Dr. P. K. Mishra, Associate Prof.
Dept. of CS, BHU, Varanasi

Sh. P. V. Suresh, Associate Prof.
SOCIS, IGNOU, New Delhi

Sh. V. V. Subrahmanyam, Associate
Prof. SOCIS, IGNOU, New Delhi

Sh. M. P. Mishra, Asst. Prof.
SOCIS, IGNOU, New Delhi

Dr. Naveen Kumar, Reader
SOCIS, IGNOU, New Delhi

Dr. Subodh Kesharwani, Asst. Prof.
SOMS, IGNOU, New Delhi

COURSE CURRICULUM DESIGN COMMITTEE

Sh. Shashi Bhushan
SOCIS, IGNOU, New Delhi

Sh. Akshay Kumar
SOCIS, IGNOU, New Delhi

Prof. Radhey S. Gupta
New Delhi

Prof. Sujatha Varma
SOS, IGNOU

Sh. Milind Mahajani
New Delhi

Dr. D. K. Loyal
SC&SS, JNU

Dr. Pravin Chandra
IIC, DU, New Delhi

Sh. P. V. Suresh
SOCIS, IGNOU, New Delhi

Sh. V. V. Subrahmanyam
SOCIS, IGNOU, New Delhi

Dr. Naveen Kumar
SOCIS, IGNOU, New Delhi

Sh. M. P. Mishra
SOCIS, IGNOU, New Delhi

SOCIS FACULTY

Sh. Shashi Bhushan, Director

Sh. Akshay Kumar
Associate Professor

Sh. M. P. Mishra
Asst. Professor

Prof. Manohar Lal

Sh. V. V. Subrahmanyam
Associate Professor

Dr. Sudhansh Sharma
Asst. Professor

Dr. P. V. Suresh
Associate Professor

Dr. Naveen Kumar
Reader

PREPARATION TEAM

Prof. Radhey S. Gupta (Content Editor)
Former Professor & Head
Department of Mathematics
I.I.T. Roorkee, Roorkee

Prof. Manohar Lal
SOCIS, IGNOU
New Delhi

Course Coordinator : Prof. Manohar Lal, SOCIS, IGNOU, New Delhi

PRINT PRODUCTION

Shri Rajiv Girdhar
A R (P) MPDD
IGNOU, New Delhi

Mr. Tilak Raj
S O (P) MPDD
IGNOU, New Delhi

March, 2015 (Reprint)

© Indira Gandhi National Open University, 2013

ISBN -978-81-266-6566-2

All rights reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from the Indira Gandhi National Open University.

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidam Garhi, New Delhi-110 068.

**Printed and published on behalf of the Indira Gandhi National Open University,
New Delhi by the Registrar, MPDD.**

Printed at : Chandra Prabhu Offset Printing Works Pvt. Ltd., C-40, Sector - 8, Noida 201301 (U.P.)

COURSE INTRODUCTION

Numerical analysis is, in essence, a branch of Mathematics concerned with development, analysis and evaluation of **constructive numerical solutions**, generally, executable by a computer for obtaining, if possible, **exact; else at least**, reasonably good approximate numerical solutions to mathematical problems.

In view of the availability of ever more powerful computer systems with ever increasing processor speeds, more precise representations for numerical quantities, and more efficient (computer-executable) algorithms; the discipline of **Numerical Analysis/ Techniques** has become essentially the discipline of **Computer-Oriented Numerical Analysis / Techniques**. However, for solving numerical problems, the use of computer as an essential tool, puts restrictions on the solution process, in view of the fact that **computers are finite machines**; in which quantities representing information, must be represented using (pre-assigned, finite) number of computer words; and which, for solving the problem under consideration, execute computer programs/ algorithms, which, by definition, are solutions involving only finite number of steps. The significance of the restrictions can be visualized from the facts that (i) a simple number like $1/3$ cannot be represented exactly in any computer system and (ii) the process of calculating a simple quantity/ function like e^x , or even, e^1 ; using Taylor's series requiring infinitely many steps, has to be (pre-maturely) truncated.

In view of these approximations, exact solutions are exceptions, rather than being the rule. Compounding effect of the various types of approximations may lead to results that hardly have any proximity to the required solution. Hence, in a **general Numerical Analysis/ Techniques** course, in addition to the numerical methods for solving problems, there has to be emphasis on careful analysis with respect to quantum of possible error. **However, this course, being elementary, will restrict to discussing some well-known methods/ algorithms for numerical solutions of the problems.** Analysis and other related matters will be discussed in an advanced course.

The course material consists of three blocks. The topics covered in the **first block** are: Computer arithmetic; solution of systems of simultaneous linear equations using both direct methods and indirect methods; and solution of non-linear equations. In **Block 2**, the following topics about interpolation are discussed: operators and inter-relations between them, interpolation methods when data is equally-spaced, and when data is not necessarily equally-spaced. **In Block 3**, the topics discussed are: numerical differentiation, numerical integration and numerical solutions of linear differential equations using Euler's method, Improved Euler's method and Runge-Kutta (order 2 and order 4) methods.

The continuous evaluation assignment shall be mainly about applying methods to solve problems or about writing programs in C/C++ and/ or in MS-Excel/ any spread-sheet. However, theory paper shall be based on theory and application of methods for numerical computation. **No algorithm/ computer program will be asked to be written in theory paper (in TEE).**

→Useful references are given at the end of Unit 0 of this Block←

BLOCK INTRODUCTION

The block consists of 5 units, namely, Unit 0, Unit 1, Unit 2, Unit 3 and Appendix.

Unit 0 and Appendix are optional, but, highly recommended readings, especially for the academic counsellors of the course. **However, there will not be any questions in Term End Examination from the material of these two units.** These units have been included in view of the fact that, the discipline of numerical methods/analysis is about attempting to solve mathematical problems, through algorithms that (i) use only computer numbers (to be defined in Unit 0) for representing information, and (ii) use only four arithmetic operations, viz., addition, subtraction, multiplication and division on these numbers, for manipulating information. However, the computer numbers constitute a special subset of real, rather rational, numbers. Hence, the number systems, both conventional and computer, form the foundation of the discipline of numerical methods. A major portion of each of Unit 0 and Appendix is about the number systems.

Also, whenever a new academic discipline is intended to be pursued, some questions, about the discipline, including the following ones arise naturally: Why, at all, we should study the discipline?, What is its domain of study or subject-matter of the discipline? , What are its distinct features, special tools and techniques? Also, we will like to know the opinions of the experts in the field in respect of major issues, including such questions.

In Unit 0, we just briefly discuss these issues.

In Unit 1, we mainly discuss representations of (real) numbers, if possible, exactly; otherwise, approximately. For this purpose, we discuss various representation schemes, viz., fixed point, scientific and floating point schemes. The discussion is mainly in terms of decimal numbers, especially, in view of our familiarity with these numbers. However, representation schemes are also discussed for binary numbers, in view of the fact that most of the computer systems use binary numbers. We also discuss how the four arithmetic operations are applied to the represented numbers, and how the application of these operations may affect the result.

In view of the fact that only four elementary arithmetic operations are available for manipulation, each of the other operations/ functions like log, exponentiation, sin etc. have to be realized through some appropriate sequence of the four elementary operations. Taylor's series expansion is a well-known technique for the purpose. However, in view of the fact that Taylor's series expansion, for most of the functions, is infinite; its application has to be truncated. Hence, Taylor's series expansion and effect of truncation on computed results are discussed next. Also, special type of algorithms, viz., unstable algorithms and special type of problems, viz., ill-defined problems, each of which may give unexpected/ undesirable results, are also discussed.

In Unit 2, which is about solving systems of linear equations; two direct methods, viz., Gauss Elimination Method, and its modification using pivotal condensation; and two iterative methods, viz., the Jacobi Method and the Gauss-Seidel Method; are discussed. Then, Direct and Iterative approaches are compared.

Finally, in Unit 3, the following well-known methods for solving equations, which may not be necessarily linear, are discussed: Fixed-point Method, Bisection Method, Regula-falsi Method, Secant Method, and Newton-Raphson Method.

UNIT 0 OVERVIEW OF MACRO-ISSUES IN 'NUMERICAL ANALYSIS AND TECHNIQUES'

The unit is optional, though highly desirable, reading. It is included for a quick run-through for the first time, and for later references, from time to time, for better understanding of the subject-matter.

Structure

- 0.0 Introduction
- 0.1 Why (Computer-Oriented) Numerical Techniques?
- 0.2 What are Numerical Techniques?
- 0.3 Numbers : Sets, Systems and Notations
 - 0.3.1 Sets of Numbers
 - 0.3.2 Algebraic Systems of Numbers
 - 0.3.3 Numerals : Notations for Numbers
 - 0.3.4 Table of Contrasting and Other Properties of Conventional Number Systems and Computer Number Systems
- 0.4 Conversion of Numbers (*in Fixed Point Format*) from Decimal to Binary Representation and Vice-versa
- 0.5 Two Different Approaches : Direct and Iterative
- 0.6 Some Similar, but Distinct Concepts, Explained : *Precision, Accuracy, Significant Digits/Figures, Machine Epsilon, Number of Digits after the Decimal*
- 0.7 Definitions and Comments by Pioneers and Leading Writers about What Numerical Analysis is
- 0.8 References

0.0 INTRODUCTION

Whenever a new academic discipline is intended to be pursued, some questions, about the discipline, including the following ones arise naturally : Why, at all, we should study the discipline? What is its domain of study or its subject-matter? What are its distinct features, special tools and techniques? Also, we will like to know the opinions of the experts in the field in respect of major issues, including such questions. In this Unit, we just briefly discuss, rather only enumerate, the questions along with our brief explanations and opinions of some experts in the field.

0.1 WHY (COMPUTER-ORIENTED) NUMERICAL TECHNIQUES?

A mathematician knows how to solve a problem – but he can't do it
W.E. Milne [¹]

The reason is that a mathematician, being a mathematician, uses all sorts of mathematical assets including mathematical concepts, notations, techniques, and at the top of all these, mathematical thinking, intuition, reasoning and habit in solving problems. Doing so,

¹ Page 1 of Introduction to Numerical Analysis (Second Edition) by Carl-Erik Froberg (Addison Wesley, 1981)

though, may be quite useful in solving many problems, yet may be quite problematic unless done carefully, while using computer as a tool for solving problems.

Because of our habitual (*mathematical*) thinking we use, without second thoughts, many mathematical identities, including the following ones :

$$a + (b + c) = (a + b) + c$$

$$\frac{a}{(b \times c)} = \frac{\left(\frac{a}{b}\right)}{c} \quad \text{and} \quad \frac{(1+a)}{2} = \frac{1}{2} + \frac{a}{2}$$

However, neither of these may be correct in many cases of *numerical* computation, and use of these identities *blindly*, may lead to completely erroneous results.

Also, many a time, we know a *mathematical* solution to the problem, but the solution cannot be useful because of various practical reasons, including the fact that the mathematical solution may involve infinitely many computational steps. For example, there is a mathematical solution to the problem of finding value of e^x , for some given value of x , by using the formula

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

where the R.H.S. converges absolutely to the function e^x for any value of x . However, apart from the fact that R.H.S. requires an infinite process, which is not realizable on a computer system; *otherwise also*, it may not help us in obtaining a *numerical* solution. The reason being that the use of the series to evaluate e^{-100} would be completely impractical, because of the fact that we would need to compute about 100 terms of the series before the size of the terms would begin to decrease. (The example is from Page 1 of *A Survey of Numerical Mathematics Volume 1* by Young & Gregory (Addison & Wesley, 1972)).

*These two examples amply illustrate that a different framework of mind and, at least, some different set of techniques (to be called **numerical techniques**) are required to successfully solve problems numerically.*

On closer examination, these problems arise because of the fact that Mathematics (especially as it is currently taught) and numerical analysis differ from each other more than is usually realized. The most obvious *differences* are that *mathematics regularly uses the infinite* both for representation and for processes, whereas *computing is necessarily done on a finite machine in a finite time*². This fact of difference is repeatedly emphasized in these lecture notes, and also, in every numerical methods' book.

In this context, it is not irrelevant to mention the too obvious fact that computer has become an indispensable tool to solve mathematical problems, specially, because of its fast speed, iterative capability and the capability to represent very large to very small quantities, much more precisely than human beings can do using pen and paper, etc.

However, as mentioned earlier, *computer is a finite machine* – a machine having pre-assigned *machine-specific finite space* (1, 2 or 4 etc., number of words in memory) for representing quantities and *finite time* to accomplish a task (what will be the utility of a solution, if it is delivered after infinite time, i.e., after eternity). And, as clarified earlier through the two examples, it has to be used with utmost care, particularly while using it for solving numerical problems based on mathematical results or solutions. While adapting a mathematical solution for execution on a computer, we have to be *perennially aware* that each (specific) computer requires the computer-specific adapting of a mathematical solution. Forgetting these facts, even momentarily, has lead to a number of *disasters, due to numerical errors*, including the following ones : *Patriot Missile*

² Page 2 Numerical Methods for Scientists and Engineers (First Edition) by R.W. Hamming (McGraw-Hill, 1962)

Failure; Explosion of the Ariane 5; EURO page : Conversion Arithmetics; The Vancouver Stock Exchange; Rounding error changes Parliament makeup, The sinking of the Sleipner, An offshore platform; Tacoma bridge failure (wrong design); 200 million dollar typing error (typing error) and What's 77.1×850 ? Don't ask Excel 2007³.

These *disasters, due to numerical errors*, further emphasize the need for *being extremely cautious* in solving *mathematical problems numerically*. Computer-oriented numerical techniques help us in doing so. More explicitly, these techniques, among other matters, help us in *adapting appropriately* mathematical solutions for execution on computer – rather help us in *specific adapting* for each (specific) computer and, hence help us in avoiding many potential disasters.

0.2 WHAT ARE NUMERICAL TECHNIQUES?

The purpose of numerical analysis is 'insight, not numbers'

R. W. Hamming in [4]

(Though numerical techniques have been used for hundreds of years, yet the advent of computer has enhanced many-folds the frequency of use and utility of numerical techniques in solving mathematical problems. Hence, by a 'numerical technique', we will invariably mean 'Computer oriented numerical technique')

The explanation in previous section, also gives an idea of the need to know the differences between, on one hand, a mathematical approach/technique, in general and, on the other hand, a numerical approach/ technique for solving mathematical problems. In this respect, the following points may be noted:

- (1) **Numerical Techniques** are about designing algorithms or constructive methods for solving *mathematical problems*, which (i.e., algorithms)
 - use only *computer-represent-able numbers* (to be defined later, and to be called only **computer numbers**) for representing data/information, and
 - use only the **numerical operations**, i.e., plus, minus, multiplication and division, on the computer numbers, for transforming data/information
Even a simple operation like square-rooting ($\sqrt{}$) is not a numeric operation, and has to be realized through some algorithm, which can use only the above-mentioned (numerical) operations.
- (2) **Rounding** : There are only finitely many computer numbers, and the number of real numbers, is infinite. Therefore, not all real numbers (even, not all natural numbers) can be expressed as computer numbers. Those real numbers (and complex numbers, expressed as pair of real numbers), which are not computer numbers, have to be approximated and represented in the computer, by computer numbers. The process is called **rounding**, and induces **rounding error**.
- (3) Further, numerical operations when applied to *computer numbers* in the usual arithmetic sense, may result in a *real number* that may not be a *computer number*. Such a *real number* has, again, to be appropriately approximated to a computer number through '**rounding**'.
- (4) **Truncation** : An algorithm or a constructive method, by definition, is *finite*. Thus, any infinite process/method, including the one mentioned above in respect of e^x , is not constructive or algorithmic An infinite mathematical process, if required to be used, has to be replaced or approximated by some appropriate finite process. The process of approximation, is called *truncation*, and induces *truncation error*.

³ In order to emphasize the point, discussion of some of the disasters is included in the Appendix of the block.

⁴ Page 3 Numerical Methods for Scientists and Engineers (Second Edition) by R.W. Hamming (McGraw-Hill, 1973)

- (5) An *analytic* function is a function which, directly or indirectly, involves the concept of limit. The set of analytical functions include: all trigonometric functions like $\sin x$ (rather, only $\sin$, though conventionally written informally as $\sin(x)$), $\log(x)$, e^x , $\frac{d}{dx}$ (i.e., derivative), and $\int f(x) dx$ (i.e. integration).

The evaluation of an analytical function, in general, is an infinite process. As mentioned above, for some value of x , e^x may be evaluated by using the (infinite)

formula : $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$ Evaluation of an analytical function

using such a formula is called *analytical solution*, which not being finite is *not* a numerical solution.

- (6) **Iteration** is an important (numerical) technique for reformulating and/or solving many mathematical problems into numerical/computational problems, specially,
- (i) when a mathematical problem does not have an algorithmic/computational solution, e.g. the problem of finding roots of a general polynomial equation of degree five or more or
 - (ii) when a mathematical problem involves infinity
 - (a) directly, in the form of an infinite series/process, as in finding the value of e^x , for some value of x , using the formula mentioned above, or
 - (b) indirectly, to approximate irrational numbers and other non-computer numbers, which may occur as either in the final answer or during the solution process. Thus, iteration may be used in finding better approximation of the root of the equation $x^2 = 2$, after starting with some reasonably appropriate initial guess.

Iteration/iterative method (as opposed to direct method) is an important numerical technique. ... specially useful when no direct method may be available. (P.13/ Young & Gregory)

- (7) **Examining and mathematically analyzing** a problem before, during and after attempting a solution, and, if required, **mathematically reformulating** the problem at any stage, in order to get a better, if not perfect, solution of the problem under consideration, *are mathematical techniques*.

For example, we may *first* attempt to evaluate $f(x) = \tan x - \sin x$ at, say $x = 0.1250$, *in the usual way*, by evaluating each of $\tan(0.1250)$ and $\sin(0.1250)$, and then subtracting the latter from the former to get the result. However, on analysis, it is found that, if we first use the trigonometric expansion of $\tan x$ and $\sin x$ as follows :

$$\tan x = x + \left(\frac{1}{3}\right)x^3 + \left(\frac{2}{15}\right)x^5 + \left(\frac{17}{315}\right)x^7 + \dots$$

and
$$\sin x = x - \left(\frac{1}{6}\right)x^3 + \left(\frac{1}{120}\right)x^5 - \left(\frac{1}{5040}\right)x^7 + \dots$$

and reformulate the function as

$$f(x) = \frac{1}{2}x^3 + \left(\frac{1}{8}\right)x^5 + \left(\frac{13}{240}\right)x^7 + \dots,$$

then with this reformulation, a better approximation of $f(x)$ is obtained.

- (8) **Discretization** : is a specialized technique of mathematically analyzing and reformulating the problem, in which the reformulation is restricted to replacement of *continuous type mathematical concepts* by (numerically) *computable objects*.

The replacement may be made in the beginning itself and then **only** the reformulated problem is attempted to be solved. The most well-known examples of discretization are in respect of replacement of (mathematical continuous concepts of) integral and differential. Using Trapezoidal rule :

$$\int_a^b f(x) \approx \sum_{k=0}^{n-1} \left(\frac{1}{2} \right) h [f(x_k) + f(x_{k+1})],$$

the mathematical continuous concept of integral on the left is replaced by the computable object of finite sum on the right. Similarly, the mathematical continuous concept of derivative $y'(x_k)$ is replaced by the computable object, viz.

$$\text{divided difference : } \frac{(y_{k+1} - y_k)}{(x_{k+1} - x_k)}.$$

Discretization, being an approximation of *mathematical continuous concepts*, also introduces error, which we call *approximation/discretization error* and this is another type of error, different from truncation and round-off types of error.

The above-mentioned techniques, particularly, truncation, iteration and discretization, are not necessarily distinct, and, may be overlapping.

- (9) **Not every mathematical technique is necessarily a numeric technique.** One of the non-constructive (or non-algorithmic) mathematical techniques of proof is proof-by-contradiction. However, the mathematical technique **proof-by-contradiction, not being constructive, is not a numerical technique.**

Some Remarks

Remark 1 : At this stage, we may note the **difference between a numerical technique and a numerical solution** of a problem. A **numerical solution** is an algorithm that involves only computer numbers and four elementary arithmetic operations.

On the other hand, a **numerical technique** may, if required, use general mathematical knowledge, tools and techniques which **may help** in solving a problem numerically. A technique, may help in solving a problem by mathematically analyzing the problem and then reformulating the problem into other problems, which may be numerically solvable. Thus, **the techniques: Truncation, Iteration and Discretization, are numerical techniques, which use mathematical tools and techniques, for appropriate numerical actions.**

In solving problems numerically, of course, the *mechanical power* of the computer is an indispensable tool in executing the algorithm. But the power of computer comes into play only when an appropriate algorithm is already designed.

However, for designing an appropriate algorithm, the choice of appropriate numerical techniques is required. As per state of art in solving problems numerically, the choice of appropriate techniques is not a mechanical task, i.e., there is no systematic method for choosing appropriate techniques, and requires (human) intelligence and practice. With practice, the process of making appropriate choices gets refined leading to *insight*, which is what **R.W. Hamming has emphasized above.**

Remark 2 : From the description of numerical techniques in (1) of *What are numerical techniques?*, it may be concluded that discipline of (Computer-Oriented) Numerical Techniques is a specialized sub-discipline of Design and Analysis of Algorithm, in which

- (i) data/information is represented only in the form of computer numbers,
- (ii) the operations for information transformation are only the four elementary arithmetic operations, and
- (iii) mathematical problems may be reformulated into some numerical problems, using some numerical approximation techniques.

0.3 NUMBERS : SETS, SYSTEMS AND NOTATIONS

(This subsection is a summary of parts of Appendix in this Block. For more detailed discussion on this topic, refer to the Appendix.)

We have earlier mentioned that the discipline of Numerical Techniques is about

- numbers, rather special type of numbers called computer numbers, and
- application of (some restricted version of) the four arithmetic operations, viz., + (plus), - (minus), $\times$ (multiplication) and $\div$ (division) on these special numbers.

Therefore, let us, first, recall some important sets of numbers, which have been introduced to us earlier, some of these even from school days. Then we will discuss computer numbers vis-à-vis these numbers.

0.3.1 Sets of Numbers

Set of **Natural numbers** denoted by **N**, where

$$N = \{0, 1, 2, 3, 4, \dots\} \text{ or } N = \{1, 2, 3, 4, \dots\}$$

Set of **Integers** denoted by **I**, or **Z**, where

$$I \text{ (or } Z) = \{\dots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \dots\}$$

Set of **Rational Numbers** denoted by **Q**, where $Q = \{a/b, \text{ where } a \text{ and } b \text{ are integers and } b \text{ is not } 0\}$

Set of **Real Numbers** denoted by **R**. There are different ways of looking at or thinking of Real Numbers. *One of the intuitive ways of thinking of real numbers* is as the numbers that correspond to the points on a straight line extended infinitely in both the directions, such that one of the points on the line is marked as 0 and another point (different from, and to the right of, the earlier point) is marked as 1. Then to each of the points on this line, a unique real number is associated. ... There is a large subset of real numbers, no member of which is a rational number. A real number which is not a rational number is called **irrational number**. For example, $\sqrt{2}$ is an irrational number.

Set of **Complex Numbers** denoted by **C**, where $C = \{a + bi \text{ or } a + ib \text{ where } a \text{ and } b \text{ are real numbers and } i \text{ is the square root of } -1\}$.

By minor notational modifications (e.g. by writing an integer, say, 4 as a rational number $4/1$; and by writing a real number, say $\sqrt{2}$ as a complex number $\sqrt{2} + 0i$), we can easily see that $N \subset I \subset Q \subset R \subset C$.

When we do not have any specific set under consideration, the set may be referred to as a **set of numbers**, and a member of the set as just **number**.

Apart from these well-known sets of numbers, there are sets of numbers that may be useful in our later discussion. Next, we discuss two such sets.

Set of algebraic Numbers (no standard notation for the set), where an *Algebraic number* is a number that is a root of a non-zero polynomial equation⁵ with rational numbers as coefficients. For example :

- Every rational numbers is algebraic (e.g., the rational number a/b , with $b \neq 0$, is a root of the polynomial equation : $bx - a = 0$). Thus, a real number, which is not algebraic, must be irrational number.
- Even, some irrational numbers are algebraic, e.g., $\sqrt{2}$ is an algebraic number, because, it satisfies the polynomial equation : $x^2 - 2 = 0$. In general, n^{th} root

⁵ We may recall that a polynomial $P(x)$ is an expression of the form: $a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_{n-1}x + a_n$, where a_i is a number and x is a variable. Then, $P(x) = 0$ represents a polynomial equation.

of a rational number a/b , with $b \neq 0$, is algebraic, because, it is a root of the polynomial equation : $b x^n - a = 0$.

Even, a complex number may be an algebraic number, as each of the complex numbers $\sqrt{2}i (= 0 + \sqrt{2}i)$ and $-\sqrt{2}i$ is algebraic, because, each satisfies the polynomial equation : $x^2 + 2 = 0$.

Set of Transcendental Numbers (again, no standard notation for the set), where, a *transcendental number* is a complex number (and, hence, also, a real number, as, $\mathbb{R} \subset \mathbb{C}$), which is not algebraic. From the above examples, it is clear that a rational number cannot be transcendental, and some, but not all, irrational numbers, may be transcendental. The most prominent examples of transcendental numbers are π and⁶ e .

0.3.2 Algebraic Systems of Numbers (To be Called, Simply, Systems of Numbers)

In order to discuss, system of numbers, to begin with, we need to understand the concept of **operation on a set**. For this purpose, recall that $\mathbb{N}$, the set of Natural numbers, is **closed under '+'** (plus). By ' **$\mathbb{N}$ is closed under +**', we mean : if we take (any) two natural numbers, say m and n , then $m + n$ is *also* a natural number.

But, $\mathbb{N}$, the set of Natural numbers, is not closed under '-' (minus). In other words, *for some* natural numbers m and n , $m - n$ may not be a natural number, for example, for 3 and 5, $3 - 5$ is not a natural number. (Of course, $3 - 5 = -2$ is an integer.)

These facts are also stated by saying : '+' is a binary operation on $\mathbb{N}$, but, '-' is not a binary operation on $\mathbb{N}$. Here, the word **binary** means that in order to apply '+', we need (exactly) two members of $\mathbb{N}$.

In the light of above illustration of binary operation, we may recall many such statements including the following :

- (i) $\times$ (multiplication) is a binary operation on $\mathbb{N}$ (or, equivalently, we can say that $\mathbb{N}$ is closed under the binary operation $\times$)
- (ii) $-$ (minus) is a binary operation on $\mathbb{I}$ (or, equivalently, we can say that $\mathbb{I}$ is closed under the binary operation $-$), etc.

However, there are operations on numbers, which may require only one number (*the number is called argument of the operation*) of the set. For example, The **square** operation on a set of numbers takes only one number and returns its square, for example, square (3) = 9.

Thus, some operations (e.g. square) on a set of numbers may take only one argument. Such operations are called **unary operations**. Other operations may take two arguments (e.g. $+$, $-$, $\times$, $\div$) from a set of numbers. Such operations are called **binary operations**. There are operations which may take three arguments and are called **ternary operations**. Even some operations may take zero number of arguments.

Definition : Algebraic System of Numbers : A set of numbers, say, S , along with a (finite) set of operations on S , is called an algebraic system of numbers. Instead of 'algebraic system', we may use the word '**system**'.

Notation for a System : If $O_1, O_2, \dots, O_n$ are some n operations on a set S , then, we denote the corresponding system as $\langle S, O_1, O_2, \dots, O_n \rangle$, or as $(S, O_1, O_2, \dots, O_n)$.

Examples and Non-examples of Systems of Numbers

(1) Examples of Number Systems

Each of following is a system of numbers : $\langle \mathbb{N}, + \rangle$, $\langle \mathbb{N}, \times \rangle$, and $\langle \mathbb{N}, +, \times \rangle$ etc.

⁶ It should be noted that it is quite complex task to show a number as transcendental. In order to show a number, say, n , to be transcendental, theoretically, it is required to ensure that for *each* polynomial equation $P(x) = 0$, n is not a root of the equation. And, there are infinitely many polynomial equations. This direct method for showing a number as transcendental, cannot be used. There are other methods for the purpose.

(2) Non-examples of Number Systems

Each of following is NOT a system of numbers : $\langle \mathbb{N}, - \rangle$, $\langle \mathbb{N}, \div \rangle$, $\langle \mathbb{N}, -, \div \rangle$, and $\langle \mathbb{I}, \div \rangle$ etc.

Remark : In the above discussion, the use of the word **number** is inaccurate. Actually, a number is a concept (a mental entity), which may be represented in some (physical) forms, so that we can experience the concept through our senses. The number, the name of which is, say, **ten** in English and दस in Hindi, and zehn in German language may be represented as 10 as decimal numeral, X as Roman numeral, 1010 as binary numeral. As, you may have already noticed, the physical representation of a number is called its **numeral**. Thus, number and numeral are two different entities, incorrectly taken to be the same. Also, a particular number is unique, but, it can have many (physical) representations, each being called a numeral, corresponding to the number.

The difference between **number** and **numeral** may be further clarified from the following explanation : We have the **concept** of the animal **that is called COW** in English, गाय in Hindi and KUH in German language. The animal, represented as cow in English, has four legs; however, its representation in English : **cow**, is a word in English and has three letters, but does not have four legs.

However, due to usage, though inaccurate, over centuries, in stead of the word **numeral**, almost, the word **number** is used. Except for the discussion in the following subsection, we will also not differentiate between Number and Numeral.

0.3.3 Numerals : Notations for Numbers

First, we recall some well-known sets used to denote numbers. These sets are called *sets of numerals* and then discuss various *number systems*, developed from these numeral sets, for representing numbers.

0.3.3.1 Sets of Numerals

We are already familiar with some of the sets of numerals. The most familiar, and frequently used, set is **Decimal Numeral Set**. It is called *Decimal*, because, it uses ten figures, or digits, viz., *digits from the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ of ten digits*⁷.

Another numeral set, familiar to computer science students, is **binary** numeral set. It is called binary, because, it uses two figures, or digits, viz., *digits from the set $\{0, 1\}$ of two digits*. In this case, 0 and 1 are called *bits*.

Also, **Roman Numeral Set**, is well-known. This set uses figures/digits/letters from the set $\{I, V, X, L, C, D, M, \dots\}$, where, I represents 1 (of decimal numeral system), V represents 5, X represents 10, L represents 50, C represents 100, D represents 500 and M represents 1000, etc.⁸

⁷ Over a number of centuries now, mainly decimal number systems have been used to represent quantities/ numbers. However, other number systems have been used and are still being used in special applications, e.g., base-12 systems (dozen = 12 and gross = 144, used still in purchase of, say, paper sheets and bananas); base-20 (score = 20); base-60 or sexagesimal (used in measure of time in terms of hour-minute-seconds).

But, in general, we have a better understanding of the measure of a quantity, if it is expressed in decimal. For example, we understand a quantity better when written as 152 in decimal than when it is written as one gross and eight in base-12; seven score and twelve in base-20; or 10011000 in binary. Generally, in order to have proper idea of the quantity, we convert the representation, if in other base, to base-10.

The decimal number system (i.e., base-10) has become intuitive, quite natural to us, the human beings.

⁸ Appropriate choice of numeral system has significant role in solving problems, particularly solving problems *efficiently*. For example, it is a child's play to get the answer for 46×37 (in decimal numeral system) as a single number. However, using Roman numerals, i.e., writing $VLII \times XXXVII$, instead of, 46×37 , it is really very difficult to get the same number, using only Roman numerals, as answer.

Apart from these sets of numerals, in context of computer systems, we also come across

- (i) **Hexadecimal numeral set**, which uses figures/digits, viz., from the set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E} of sixteen digits.
- (ii) **Octal numeral set**, which uses figures/digits, viz., from the set : {0, 1, 2, 3, 4, 5, 6, 7} of eight digits.

0.3.3.2 Number Representation using a Set of Numerals

A number is represented by a *string of digits* from the set of numerals under consideration, e.g., 3426 in decimal, IX in Roman, 10010110 in binary and 37A08 in hexadecimal.

0.3.3.3 Value of the Number Denoted by a String

Using either of numeral sets *introduced* in 0.3.3.1 above, there are **different schemes, i.e. sets of rules for interpreting a string as a number.**

For example, the string '4723', according to *usual decimal system*, represents the number : $4 \times 10^3 + 7 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$. Also, the string 'CLVII', according to the *Roman system*, represents the (decimal) number : $100 + 50 + 5 + 1 + 1 = 157$ (decimal), where C denotes 100, L denotes 50, V denotes 5 and I denotes 1. Similarly, the *binary string* 10010110 may be interpreted as the number (with value in decimal) : $1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$.

Most of the computer systems use only binary representation for numbers, i.e., a number is a string of only 0's and 1's. However, a particular string of bits may represent different numbers according to different schemes, i.e., sets of rules for interpreting a string as a number.

The schemes for interpreting a (binary) string as a number, at the top level may be categorized into two major classes according to (i) Fixed point representation and (ii) Floating point representation schemes.

In Fixed point binary representation, the binary point forms a part of the string representing the number and its location within the string determines the relative size of the number. Further, *within the Fixed point representation class*; there are a number of schemes for interpreting a string as a number. Different schemes may give different numbers. For example, the string : 1000 0111 (assuming binary point is at the right-most position) has the value in decimal, given within the parentheses for the corresponding interpreting scheme

- Binary unsigned magnitude ($128 + 4 + 2 + 1 = 135$)
- Binary signed magnitude ($-(4 + 2 + 1) = -7$), (the left-most bit is interpreted as sign - : '0' as '+' and '1' as '-').
- BCD (Binary Coded Decimal) (87) (the string is divided into groups of 4 contiguous bits, starting from the right-most bit, using some 0's on left, if required. Then replacing each 4-bit group by its unsigned magnitude value.)
- Excess-3 (54) (0 is represented as 0011, which is the representation of digit 3 ($= 0 + 3$) in binary unsigned magnitude.... and 9 as 1100, which is the representation of 12 ($= 9 + 3$) in binary unsigned magnitude.)
- signed 1's complement (-120), (the left-most bit is interpreted as sign as in binary signed magnitude. If sign bit is 0, then interpret the binary string as is done in binary signed magnitude. Otherwise, other bits are inverted, i.e., '0' becomes a '1' and a '1' becomes a '0', and the new string is interpreted as binary unsigned magnitude).
- signed 2's complement (-121). (for getting the value, first, do as in signed 1's complement, followed by addition of 1 to the number so obtained).

0.3.4 Table of Contrasting and Other Properties of Conventional Number Systems and Computer Number Systems

Sl. No.	Properties of Conventional Number Systems	Properties of Computer Numbers
1.	Each of the sets N, I, Q, R and C is an <i>infinite</i> set	The number of computer representable numbers, is <i>finite</i> only, though substantially large. Therefore, <i>not</i> even all natural numbers (and, hence, all integers, all rational numbers and all real numbers) can be represented in a computer system
2.	Each of these sets/ systems of numbers is unique, independent of representation scheme	The set of computer numbers is not unique. Each computer system has its unique set of computer-representable numbers, which may be different from those of another computer system. The numbers that can be represented in a computer system depend on the word size of the computer system and the scheme of representation used
3.	None of the number systems mentioned above, is bounded above, i.e., has the <i>maximum element</i>	For each computer system, the set of computer numbers has the <i>maximum element</i> , i.e., the set of computer numbers is bounded above. However, the maximum element is, again, computer dependent
4.	Except N, the set of natural numbers, none of the other number systems I, Q, R has the <i>minimum element</i>	For each computer system, the set of computer numbers has the <i>minimum element</i> , i.e., the set is bounded below. However, the minimum element is, again, computer dependent. The number is close to the number $-(max)$, where <i>max</i> denotes maximum computer representable number, and the minimum depends on scheme of representation
5.	The set of real (or rational) numbers does not have the <i>least positive real number</i> . Because, between 0 and any positive real/ rational number, say r , lies the positive real/ rational number $r/2$	Computer representable numbers <i>have minimum positive computer number</i> . However, each computer system has its own unique minimum positive computer representable number.
6.	The number 0 is a single number in conventional number systems	Computer number zero is not the same as real number Zero : If x is any real number such that $ x $, if after rounding, is less than minimum computer-representable number, say, ϵ , then x is represented by zero. Thus, computer zero represents <i>not a single real number 0</i> , but all the infinitely many real numbers of an interval contained in $]-\epsilon, \epsilon[$, and ϵ <i>varies from computer to computer</i>
7.	If r is a root of an equation $f(x) = 0$, in mathematical sense, then it is necessary that $f(r) = 0$	In view of the above statement at 6, for a <i>computed real root</i> , say r , of an equation $f(x) = 0$, it does not necessarily mean $f(r) = 0$. It may only mean that $f(r)$, as a real number, lies in the interval $]-\epsilon, \epsilon[$
8.	a. Each of the Number systems: N, I, Q, R, and C is closed under $+$ and $\times$, i.e., for two numbers a and b in any one of these sets, then the sum $a + b$ and product $a \times b$ are also in the same set b. Each of the Number systems: I, Q, R, and C is closed under $-$, i.e., for two numbers a and b in any one of these sets, then the difference $a - b$ is also in the same set c. Each of the Number systems: $R \setminus \{0\}$ and $C \setminus \{0\}$ is closed under $\div$ (division), i.e., for two numbers a and b in any one of these sets, then the quotient $a \div b$ is also in the same set	The set of computer numbers is not closed under each of the four numerical operations. For example, if M denotes the maximum representable number in a computer system, then $M + M$ and $M \times M$ are NOT computer representable numbers Thus, set of computer numbers is NOT closed under $+$ (sum) and $\times$ (product) Also, then $M - (-1) = M + 1$ is NOT computer representable number. Thus, set of computer numbers is NOT closed under $-$ (difference) Also, each of the numbers 1 and 3 is computer representable number, but, $1 \div 3 = 1/3$ is not a computer number. Thus, set of computer numbers is NOT closed under $\div$ (division)

9.	For each of the number systems: N, I, Q, R, and C, the following hold a. '+' is Commutative in numbers, i.e., $x + y = y + x$, for any numbers x and y b. 'x' is Commutative in numbers, i.e., $x \times y = y \times x$, for any numbers x and y	For computer numbers, the following hold a. '+' is Commutative in numbers, i.e., $x + y = y + x$, for any numbers x and y b. 'x' is Commutative in numbers, i.e., $x \times y = y \times x$, for any numbers x and y
10.	For each of the number systems: N, I, Q, R, and C, the following hold a. '+' is Associative in numbers, i.e., $(x + y) + z = x + (y + z)$, for any numbers x, y and z b. 'x' is Associative in numbers, i.e., $(x \times y) \times z = x \times (y \times z)$, for any numbers x, y and z ,	For computer numbers the following Hold : a. '+' is NOT Associative, i.e., $(x + y) + z \neq x + (y + z)$, for computer numbers x, y and z b. 'x' is NOT Associative i.e. $(x \times y) \times z \neq x \times (y \times z)$, for computer numbers x, y and z ,
11.	The conventional number systems are mainly for human understanding and comprehension of number size. <i>Mainly decimal number system</i> have been used to represent numbers, because of the facts that (i) decimal system have been in use over a long period and (ii) its capability to represent numbers in a uniform, easy to understand, manner	A computer number is necessarily a binary number, i.e., it is a (finite) string of only 0's and 1's (in this case, 0 and 1 are called <i>bits</i>). However, as mentioned earlier, a particular string of bits may represent different numbers according to different schemes, i.e., sets of rules for interpreting a string as a number
		Further, in context of Property 1 of computer numbers, it may be stated that no real number, which is transcendental number, like π and e , can be represented exactly in any computer system. Thus Further, no irrational number, like $\sqrt{2}$, can be represented exactly in any computer system.... Only finitely many real numbers, each of which must also be rational, can be computer represent-able. But, as mentioned above, even not all rational numbers are computer represent-able. For example, $1/3$ is a rational number, which cannot be represented as a finite binary string, and hence, is not a computer number. Further, it may be noted that some rational numbers which can be represented as a finite string of <i>decimal</i> digits, may not be written as a finite string of bits. For example, $1/5$ can be written as: 0.2, a finite decimal string, but can be written only as an infinite <i>binary</i> string: 0.00110011..... Each of the real numbers, which cannot be represented in a computer system, if required to be stored in a computer, is approximated appropriately, to a computer number (of the computer system)

0.4 CONVERSION OF NUMBERS (IN FIXED POINT FORMAT) FROM DECIMAL TO BINARY REPRESENTATION AND VICE-VERSA

The decimal numbers having been in use over centuries, have become quite natural, almost intuitive to the human beings. On the other hand, most of the computer systems use, for internal representation of quantities, binary numbers. In order to use computers for solving problems, the human beings communicate these quantities to computer in decimal form. The computer converts a decimal number to binary number for internal representation and subsequent processing. After processing, the numbers need to be converted back to decimal form for human understanding. In some cases, conversion of decimal numbers to binary, followed by processing and then conversion back to decimals, do not yield the expected and mathematically correct numbers. For example,

one computer with nine decimal digits of accuracy gave the answer 9999.99447, when it was asked to add 100,000 times the number⁹ 0.1.

This type of facts necessitates the analysis, of numerical solutions, for errors. For proper analysis, we need to understand properly the processes of conversion of numbers from decimal to binary and, from binary to decimal. We explain the processes through some examples.

In order to distinguish numbers in decimal form and numbers in binary form from each other; when ever there is need, we use suffixes: 10 for decimal and 2 for binary. **Also, the decimal digits may be denoted by d_i and binary digits by b_i .**

Procedure 1 for converting an integer from decimal to binary : Divide the magnitude of the given number by 2 and also the subsequent quotients successively by 2 and note the successive remainders as digits of binary representation, the first remainder bit taken as the least significant and the last as the most significant. The procedure is terminated when the quotient becomes 0. Output the sequence of bits, starting with the most significant bit first on the left. Prefix the sign (i.e., if negative, change the sign etc.)

Example : Convert the decimal integer – 349 to binary.

Solution : Divide the magnitude 349, and subsequent quotients successively by 2 and note the successive remainders as digits of binary representation, as follows :

$$\begin{array}{ll} 349 = 2 \times 174 + 1, & b_0 = 1 \\ 174 = 2 \times 87 + 0, & b_1 = 0 \\ 87 = 2 \times 43 + 1, & b_2 = 1 \\ 43 = 2 \times 21 + 1, & b_3 = 1 \\ 21 = 2 \times 10 + 1, & b_4 = 1 \\ 10 = 2 \times 5 + 0, & b_5 = 0 \\ 5 = 2 \times 2 + 1, & b_6 = 1 \\ 2 = 2 \times 1 + 0, & b_7 = 0 \\ 1 = 2 \times 0 + 1, & b_8 = 1 \end{array}$$

Thus, $(-349)_{10} = -(101011101)_2$ $\xrightarrow{\quad} b_0$

Procedure 2 for converting a fraction (no integral part) from decimal to binary is explained through the following example.

Example : Convert the number with decimal fraction – 0.3125 to binary fraction

$$\begin{array}{ll} 0.3125 \times 2 & ; \text{multiply only the fractional part by 2} \\ \hline 0.6250 & b_k = 0 \quad ; \text{at each stage, } b_i \text{ is assigned the digit} \\ & ; \text{before the decimal point.} \\ \times 2 & ; \text{multiply only the fractional part by 2} \\ \hline 1.250 & b_{k-1} = 1 \\ \times 2 & ; \text{multiply only the fractional part by 2} \\ \hline 0.500 & b_{k-2} = 0 \\ \times 2 & ; \text{multiply only the fractional part by 2} \\ \hline 1.00 & b_{k-3} = 1 \end{array}$$

⁹ From Page 13, Numerical Methods Using MATLAB by Mathews & Fink

At this stage, the fractional part is zero, hence, terminate the procedure for the decimal fraction – 0.3125, the corresponding binary fraction is – 0.0101.

Remark : In case the equivalent binary string is non-terminating, after reaching maximum (say 8) number of bits allowed for binary fraction, the procedure is terminated, or 9th bit is calculated for rounding the 8th bit

Example : Convert the decimal number – 349.3125 to equivalent binary number

Solution : The number – 349.3125 can be written as – 349 + (– 0.3125), the sum of its integral and fractional parts. The integral and fractional parts are then treated separately applying respectively Procedure 1 and Procedure 2, discussed above, which yield respectively – $(101011101)_2$ and – $(0.0101)_2$. Thus, the required binary number is – $(101011101.0101)_2$.

Procedure 3 for converting an integer in binary form, say, $\pm b_k b_{k-1} b_{k-2} \dots b_1 b_0$ to its decimal form :

Begin (assuming the bits are directly available to us, for arithmetic operations)

Let Temp $\leftarrow b_k$; the highest order bit is stored in Temp

For i = 1 to k **do**

Temp $\leftarrow 2 \times \text{Temp} + b_{k-i}$; the multiplication by 2 is in base 10.

If the given integer is negative

Temp $\leftarrow (-1) \times \text{Temp}$;

Output Temp as the required number

End.

Example : Convert the binary number – $(101011101)_2$ to decimal integer.

Temp $\leftarrow 1$; assign to Temp the most significant bit after the
; binary point in binary form of given number

For i = 1, Temp $\leftarrow 2 \times 1 + 0 = 2$; multiplication in decimal arithmetic, adding
; next most significant bit

For i = 2, Temp $\leftarrow 2 \times 2 + 1 = 5$

For i = 3, Temp $\leftarrow 2 \times 5 + 0 = 10$

For i = 4, Temp $\leftarrow 2 \times 10 + 1 = 21$

For i = 5, Temp $\leftarrow 2 \times 21 + 1 = 43$

For i = 6, Temp $\leftarrow 2 \times 43 + 1 = 87$

For i = 7, Temp $\leftarrow 2 \times 87 + 0 = 174$

For i = 8, Temp $\leftarrow 2 \times 174 + 1 = 349$.

Procedure 4 for converting a fraction (no integral part) $.b_1 b_2 b_3 \dots b_k$ from binary to decimal.

In respect of the contribution of a particular bit to the required decimal value, it may be noted that the bit b_1 is multiplied by 0.5. And other bits are multiplied by a factor which is half of the factor for the bit immediately to its left. In general

The factor by which b_i is to be multiplied = $(\frac{1}{2})$ the factor by which b_{i-1} is multiplied.

After multiplication of each of the successive bits, from left, by respective factor, the quantities so obtained, are added.

The above method is illustrated through the following example. In the following, Decimal-value denotes the partial values, at different stages of computation, of the required number.

Example : Convert the number given as a binary fraction – $(0.11001010)_2$ to decimal.

Let factor $\leftarrow 0.5$

Decimal-value = $b_1 \times \text{factor} = 1 \times 0.5 = 0.5$

factor $\leftarrow 0.5/2 = 0.250$ then

Decimal-value = decimal-value + $b_2 \times \text{factor} = 0.5 + 1 \times 0.250 = 0.750$

factor $\leftarrow 0.250/2 = 0.125$ then

Decimal-value = decimal-value + $b_3 \times \text{factor} = 0.750 + 0 \times 0.1250 = 0.750$

factor $\leftarrow 0.125/2 = 0.0625$

Decimal-value = decimal-value + $b_4 \times \text{factor} = 0.750 + 0 \times 0.06250 = 0.750$

factor $\leftarrow 0.0625/2 = 0.03125$

Decimal-value = decimal-value + $b_5 \times \text{factor} = 0.750 + 1 \times 0.03125 = 0.78125$

factor $\leftarrow 0.03125/2 = 0.015625$

Decimal-value = decimal-value + $b_6 \times \text{factor} = 0.78125 + 0 \times 0.015625 = 0.78125$

factor $\leftarrow 0.015625/2 = 0.0078125$

Decimal-value = decimal-value + $b_7 \times \text{factor} = 0.78125 + 0 \times 0.0078125 = 0.7890625$

Thus decimal equivalent of – $(0.11001010)_2$ is – 0.7890625.

Example : Convert the number given in binary as – $(101011101.1001010)_2$ to decimal.

Solution : The integral and fractional parts will be treated separately applying respectively Procedure 3 and Procedure 4, which yield respectively – 349 and – 0.7890625.

Thus, the required binary number is – 349.7890625.

The number – $(101011101.1001010)_2$ can be written as $(-101011101)_2 + (-0.1001010)_2$, the sum of its integral and fractional parts. The integral and fractional parts are then treated separately applying respectively Procedure 3 and Procedure 4, discussed above, which yield respectively – 349 and – 0.7890625. Thus, the required decimal number is – 349.7890625.

0.5 TWO DIFFERENT APPROACHES : DIRECT AND ITERATIVE

In Mathematics in general, and in the discipline of Numerical Techniques and Analysis in particular, all the methods and techniques may be classified under two approaches : (i) Direct and (ii) Iterative. In a method under iterative approach, the value of the (unknown) variable is repeatedly modified until some conditions of convergence, which tell whether the values in successive repetitions are coming closer to each other, are met. It may happen that instead of converging to a single value, the values in successive repetitions go away from each other, i.e. the values diverge, then the method is either modified or given up. In a method under iterative approach, the initial value of a variable is generally guessed. A method under iterative approach is called an iterative method. On the other hand, in a method, like finding the roots of a quadratic equation, under direct approach, a value of a variable is generally neither initially guessed nor is modified once it is obtained. A method under direct approach is called a direct method. The four procedures discussed above for conversion, are all direct methods.

Let us start with precision.

- (i) In the sense of an adjective modifying the noun *numbers* in single-precision and double-precision numbers etc., e.g. in the IEEE 754 standard, single-precision uses 32-bits, where as double-precision uses 64-bits for representing floating numbers.
- (ii) In the second sense, *precision* is the number of digits of the mantissa (including any leading implicit bit) used to characterize a floating point number. For example, in IEEE 754 standard, precision is 24 (including one implicit bit).
- (iii) In the third sense, Precision is defined as the smallest change (in terms of *decimal* digits, irrespective of the base *b* of the number system) that can be represented in floating point representation. For example in IEEE 754, in the single-precision,

[illegible]

the fractional part of a single precision normalized number has exactly 23 bits of resolution (24 bits with the implied bit), as shown above, filled up in one row by all 0's and in second row by all 0's except the least significant position, which is filled up with 1. This is a change with minimum magnitude, e.g., having magnitude 2^{-23} , i.e., 23 positions to the right of the *binary* point. This corresponds to $\log_{(10)}(2^{23}) = 6.924 \approx 7$ (the characteristic of logarithm) *decimal* digits, to the right of the *decimal* point, of accuracy. Similarly, in case of double precision numbers, the precision is $\log_{(10)}(2^{52}) = 15.654 \approx 16$ decimal digits.

19

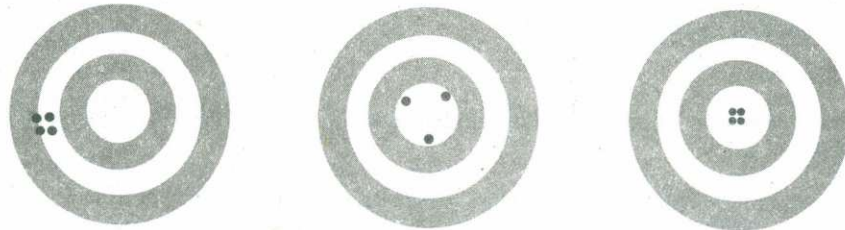
2. Accuracy vs Precision

In numerical analysis, *accuracy* is the nearness of a calculation to the *true* value; while *precision* is the resolution of the representation, typically defined by the number of decimal or binary digits, as done above.

Key Difference between the two is that while *accuracy* is the degree of closeness of a measurement of a quantity to a *true* value, *precision* is the number of times a measurement is close to a/some central value, where the central value may not be true value. **Precision** refers to how close measured or computed values agree with each other after repeated sampling.

Accuracy is the amount of times a person is close to the 'actual' or 'true' value, while precision is the amount of times a person comes up with results similar to a value, which may or may not be precise. A measurement can be accurate, precise, both or neither.

Let's take accuracy and precision in terms of a target :



In the first image, the person shooting the arrow or the 'results' are precise but not accurate. The image in the middle shows that the shooter is accurate as the points are in the bull's eye, but the person is not precise as the points are not closer to each other. The last image on the right shows that the shooter is accurate as well as precise.

In terms of language, accurate is described as truth or without any errors, while precise is described as defined, strict, neither more nor less.

A measurement system is considered *valid* if it is both *accurate* and *precise*.

3. Significant Digits

Next, we consider the concept of *Significant digits/figures* which is a measure of relative or proportional accuracy and is far more meaningful. However, the concept has a number of shades of meanings, or connotations, which, one by one, we consider below.

First Connotation : Significant Digits : A computational process is said to use m significant digits, if, whenever, a result is rounded, if required, so that after rounding, there are no more than m digits after and including the first non-zero digit. For further illustration of this connotation, we compare it with a related concept of *number of digits after the decimal* :

Value	Significant Figures	Decimal Places
7.5	2	1
7.50	3	2
0.075	2	3

Second Connotation : Significant Digits (used mostly in scientific and engineering applications) : A common convention in science and engineering is to express accuracy and/or precision implicitly by means of *significant figures*. Here, when not explicitly stated, the margin of error is understood to be one-half the value of the last significant place. For instance, a recording of 843.6 m, or 843.0 m, or 800.0 m would imply a margin of 0.05 m (the last significant place is the tenths place), while a recording of 8436 m would imply a margin of error of 0.5 m (the last significant digits are the units).

The actual value is in the range of 8435.5 m and 8436.5 m.

A reading of 8000 m, with trailing zeroes and no decimal point, is ambiguous; the trailing zeroes may or may not be intended as significant figures. To avoid this ambiguity, the number could be represented in scientific notation : 8.0×10^3 m indicates that the first zero is significant (hence a margin of $0.05 \times 10^3 = 50$ m) while 8.000×10^3 m indicates that all three zeroes are significant, giving a margin of $0.0005 \times 10^3 = 0.5$ m. Similarly, it is possible to use a multiple of the basic measurement unit : 8.0 km is equivalent to 8.0×10^3 m. In fact, it indicates a margin of 0.05 km (50 m).

Third connotation in respect representations : (abridged form of what is given in Wikipedia).

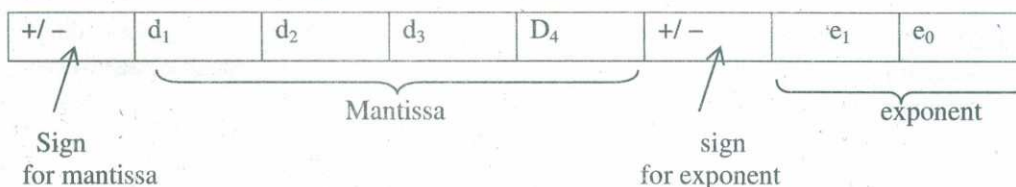
The **significant figures** (also known as **significant digits** in American English), of a number are those digits that carry meaning contributing to its precision. This includes all digits *except*:

- All *Leading zeros*;
- *Trailing zeros* when they are merely placeholders to indicate the scale of the number (exact rules are explained at *Identifying significant figures*); and
- Spurious digits introduced, for example, by calculations carried out to greater precision than that of the original data, or measurements reported to a greater precision than the equipment supports.

The rules for identifying significant figures when writing or interpreting numbers are as follows :

- All non-zero digits are considered significant. For example, 73 has two significant figures (7 and 3), while 837.45 has five significant figures (8, 3, 7, 4 and 5).
- Zeros appearing anywhere between two non-zero digits are significant. Example : 103.4205 has seven significant figures : 1, 0, 3, 4, 2, 0 and 5.
- Leading zeros are not significant. For example, 0.000793 has three significant figures : 7, 9 and 3.
- Trailing zeros in a number containing a decimal point are significant. For example, 54.2800 has six significant figures : 5, 4, 2, 8, 0 and 0. The number 0.000725300 still has only six significant figures (the zeros before the 7 are not significant). In addition, 540.00 has five significant figures since it has three trailing zeros. This convention clarifies the precision of such numbers; for example, if a measurement precise to four decimal places (0.0001) is given as 25.23 then it might be understood that only two decimal places of precision are available. Stating the result as 25.2300 makes clear that it is precise to four decimal places (in this case, six significant figures).
- The significance of trailing zeros in a number not containing a decimal point can be ambiguous. For example, it may not always be clear if a number like 3400 is precise to the nearest unit (and just happens coincidentally to be an exact multiple of a hundred) or if it is only shown to the nearest hundred due to rounding or uncertainty.

Fourth connotation in respect binary floating point representations : For the purpose of definition of significant digits in computational/ numerical context of numbers in *binary* floating point representation, for the sake of illustration, first, we discuss the involved ideas in terms of the familiar *decimal* numbers, and then come to the binary case. We assume the representation uses 4 decimal digits for mantissa and two digits for exponent, as shown in the following format for normalized floating point representation (d_1 assumed to be non-zero) :



where each of $d_1, d_2, d_3, d_4, e_1, e_0$ is a decimal digit. Further, *mantissa* part is interpreted as a decimal *fraction*, viz., the above one as $.d_1d_2d_3d_4$ and *exponent* part as a decimal whole number, viz., e_1e_0 and the mantissa m satisfies $0.1 \leq m < 1$. For example, the following

-	5	0	4	8	-	1	8
Mantissa							

represents the number $(-0.5048) \times 10^{-18}$, in which the mantissa part 5048 is treated as the fraction 0.5048.

For the purpose of estimating error, let us assume the *actual fractional part (without sign)* can be accommodated in no less than 5 digits as follows :

Say the actual fractional part is :

1	0	0	7	8	3.....
---	---	---	---	---	--------

but, which is represented by 0.1008 through rounding. If rounding is used then the error $0.000017 \leq (1/2) 0.0001 = (1/2) 10^{-4}$. Similarly if chopping is used, then the fraction is represented by 0.1007 and the error is $0.000083... \leq 0.0001 = 10^{-4}$.

In general, if t decimal digits are used in mantissa, then the chopping error $\leq 10^{-t}$ and rounding error $\leq (1/2) 10^{-t}$. In other words, the representation and the actual value, begin to differ only from t^{th} digit onward to the right. First $(t - 1)$ fractional digits of the actual and the representation agree. This is the underlying idea for the concept of *significant digits for number systems using bases, other than 10, including binary*.

Now, if, in stead of *decimal* numbers, the *binary* numbers are used, and if p bits are used in mantissa, then, in the similar way, First $(p - 1)$ fractional bits of the actual and the representation agree. The least value of the binary fraction, viz. $2^{-(p-1)} = 2^{-p+1}$, is the level of precision. This level of precision in terms of number of decimal digits is called *significant digits of the representation system*. Thus, if t were the decimal digits, then in first $(t - 1)$ decimal positions, in the fractional parts of the actual and the represented agree. Therefore, representation system has t (*decimal*) digit significance, if $10^{-(t-1)} = 10^{t-1} = 2^{p-1}$, i.e., $(t - 1) = \log_{10} (2^{p-1})$ or $t = 1 + \log_{10} (2^{p-1})$.

In respect of the IEEE 754 standards, the significant digits are given by the table below.

	Single Precision	Double Precision
Exponent bits	8	11
Mantissa bits	23	52
Significant figures	$1 + \log_{10} (2^{23})$ $\approx 7 \text{ to } 8 \text{ Decimal digits}$	$1 + \log_{10} (2^{52})$ $\approx 15 \text{ to } 16 \text{ Decimal digits}$

5. **Machine precision** (also called *Machine-epsilon*, *macheps* or *Unit round-off*) is used in backward analysis, a theory for error analysis. Machine precision is a function of

- Floating point number system under consideration and
- Rounding scheme used for approximating those real numbers which cannot be represented exactly by the number system under (i).

Definition : Machine precision/*Machine-epsilon*/*Unit round-off*/*Macheps* of a rounding scheme for a given floating number system, denoted by ϵ is the **maximum relative error** determined by the rounding for the number system.

We explain the definition through our following familiar floating point normalized representation having 4 decimal digits for mantissa (m) and 2 digits for exponent (e), and the mantissa m satisfying $0.1 \leq m < 1$.

For the purpose of estimating error, let us assume the *actual fractional part* (without sign) can be accommodated in no less than 5 digits as follows :

Further, for the sake of explanation, let us again assume the actual fractional part is :

1	0	0	7	8	3.....
---	---	---	---	---	--------

but, which is represented by 0.1008 through rounding. If *rounding* is used then the error $0.000017 \dots \leq (1/2) 0.0001 = (1/2) 10^{-4}$. Similarly if *chopping* is used, then the fraction is represented by 0.1007 and the error is $0.000083 \dots \leq 0.0001 = 10^{-4}$.

In this case :

If e is the exponent then the actual number is $0.100783 \dots \times 10^e$

If rounding is used, then

The actual error is $0.000017 \dots \times 10^e \leq (1/2) 0.0001 \times 10^e = (1/2) 10^{-4} \times 10^e$

The relative error = error/actual value

$$\begin{aligned} &= (0.000017 \dots \times 10^e) / 0.100783 \dots \times 10^e \leq ((1/2) 0.0001 \times 10^e) / 0.100783 \dots \times 10^e \\ &= ((1/2) 10^{-4} \times 10^e) / 0.100783 \dots \times 10^e = ((1/2) 10^{-4}) / 0.100783 \dots \leq [(1/2) 10^{-4} / 0.1] \\ &= (1/2) 10^{-4+1}. \end{aligned}$$

In summary, **rounding is used, then**

The relative error $\leq (1/2) 10^{-4+1}$, where 4 is the number of digits in the floating point number

Similarly, we can show, in this example that **if chopping is used, then**

The relative error $\leq 10^{-4+1}$, where 4 is the number of digits in the floating point number.

In respect of this error analysis, it may be noted that

- (i) the bounds on the relative error is free from e , the exponent,
- (ii) At one of the stages, we used $((1/2) 10^{-4}) / 0.100783 \dots \leq [(1/2) 10^{-4} / 0.1]$

But, whatever may be the actual number; in normalized form, its fractional part ≥ 0.1 . Hence, whatever may be the actual number the above inequality will be maintained.

Thus, whatever is the actual number, whenever, it is approximated by 4-digit mantissa, then,

The relative error (and hence, maximum relative error, ϵ) $\leq (1/2) 10^{-4+1}$, if **rounding is used**,

The relative error (and hence, maximum relative error, ϵ) $\leq 10^{-4+1}$, if **chopping is used**,

We can generalize the above observations about ϵ , in the following two ways :

- (i) If the fractional/mantissa is normalized having p digits, instead of 4, then
The relative error (and hence, maximum relative error, ϵ) $\leq (1/2) 10^{-p+1}$, if **rounding is used**,
The relative error (and hence, maximum relative error, ϵ) $\leq 10^{-p+1}$, if **chopping is used**.

(ii) if, instead of base 10, the base b is used then

[a] The relative error (and hence, maximum relative error, ϵ)
 $\leq (1/2) b^{-p+1}$, if **rounding is used**.

[b] The relative error (and hence, maximum relative error, ϵ)
 $\leq b^{-p+1}$, if **chopping is used**.

There are other variant definitions of machine-precision, which, however, we will not consider.

The following values of machine epsilon apply to standard floating point formats :

Formal definition

IEEE 754 – 2008	Common Name	C++ data type	Base b	Precision p	Machine epsilon [a] $\frac{1}{2} b^{-p+1}$	Machine epsilon [b] $b^{-(p-1)}$
binary32	single precision	Float	2	24 (one bit is implicit)	$2^{-24} =$ 5.96e – 08	$2^{-23} =$ 1.19e – 07
binary64	double precision	Double	2	53 (one bit is implicit)	$2^{-53} =$ 1.11e – 16	$2^{-52} =$ 2.22e – 16

0.7 DEFINITIONS AND COMMENTS BY PIONEERS AND LEADING WRITERS ABOUT WHAT NUMERICAL ANALYSIS IS

1. **K. E. Atkinson** : Numerical analysis is the area of mathematics and computer science that creates, analyzes, and implements algorithms for solving numerically the problems of continuous mathematics.

Such problems originate generally from real-world applications of algebra, geometry and calculus, and they involve variables which vary continuously; these problems occur throughout the natural sciences, social sciences, engineering, medicine, and business.

2. **Hammerlin and Hoffman** : Numerical Analysis is the mathematics of constructive methods, which can be realized numerically. Thus, one of the problems of numerical analysis is to design computer algorithms for either exactly or approximately solving problems in mathematics itself, or in its applications in natural sciences, technology, economics, and so forth.
3. **R. W. Hamming (Page 1/ Numerical Methods for Scientists and Engineers) :** *Mathematics versus Numerical Analysis.....* Mathematics and numerical analysis differ from each other more than is usually realized. The most obvious differences are that *mathematics regularly uses the infinite* both for representation of numbers and for processes, whereas *computing is necessarily done on a finite machine in a finite time*. The finite representation of numbers in the machine leads to round-off errors, whereas the finite representation leads to truncation errors.
4. **Young and Gregory (P.1/ A survey of Numerical Mathematics, Vol. I) :** Numerical analysis is concerned with the application of mathematics to the development of constructive, or algorithmic, methods which can be used to obtain numerical solutions to mathematical problems.

5. **E. K. Blum (Preface/Numerical Analysis and Computation : Theory and Practice) :** Numerical analysis, in essence, is a branch of mathematics which deals with the numerical – and therefore constructive – solutions of problems formulated and studied in other branches of mathematics.
6. **Wikipedia : Numerical Analysis** is the study of algorithms that use numerical approximation (as opposed to general symbolic manipulations) for the problems of mathematical analysis (as distinguished from discrete mathematics).
7. **From Encyclopedia of Mathematics :** The branch of mathematics concerned with finding accurate approximations to the solutions of problems whose exact solution is either impossible or infeasible to determine. In addition to the approximate solution, a realistic bound is needed for the error associated with the approximate solution.

Typically, a mathematical model for a particular problem, generally consisting of mathematical equations with constraint conditions, is constructed by specialists in the area concerned with the problem. Numerical analysis is concerned with devising methods for approximating the solution to the model, and analyzing the results for stability, speed of implementation, and appropriateness to the situation.

0.8 REFERENCES

1. *Numerical Methods Using MATLAB* (Fourth Edition) by J. H. Mathews and K. D. Fink (PHI, 2004).
2. *Elements of Numerical Analysis* by R. S. Gupta (Macmillan India Ltd., 2009).
3. *A First Course in Numerical Methods* by Ascher and Greif (PHI Learning, 2013).
4. *Computer-Oriented Numerical Methods* (Third Edition) by V. Rajaraman (PHI, 1999).
5. *Numerical Analysis and Algorithms* by Pradip Niyogi (Tata McGraw-Hill Pub, 2003).
6. *Theory and Problems on Numerical Analysis* by F. Scheid (Schaum Series, 1989).

For Advanced Learners

(A number of very good books, specially written during 1960-1990, are available; only few, which were easily accessible to us, are mentioned here.)

7. *Numerical Methods with Fortran IV Case Studies* by W. S. Dorn and D. D. McCracken (John Wiley and Sons).
8. *Introduction to Numerical Computation* (Second Edition) by James S. Vandergraft (Academic Press (1983)).
9. *Numerical Methods for Scientists and Engineers* (Second Edition) by R. W. Hamming (McGraw-Hill, 1973).
10. *Introduction to Numerical Analysis* (Second Edition) by Carl-Eric Froberg (Addison-Wesley, 1981).
11. *Basic Computational Mathematics* by V. F. D'yachnko (Mir Publishers, 1979).
12. *Elementary Numerical Analysis* (3rd Edition) by S. D. Conte and C. DeBoor (McGraw-Hill, 1981).
13. **Free NUMERICAL METHODS WITH APPLICATIONS**

UNIT 1 COMPUTER ARITHMETIC

Structure

- 1.0 Introduction
- 1.1 Objectives
- 1.2 Notation for Numbers : Fixed-point and Scientific
- 1.3 Floating-Point Representation (Only Decimal), Arithmetic and Errors
 - 1.3.1 Floating-Point Number Representation
 - 1.3.2 Floating-Point Number Representation and Errors
 - 1.3.3 Floating Number Operations of '+' and '-' and Errors
 - 1.3.4 Other Computer Arithmetic Problems : Non-associativity of '+' for Floating Numbers
 - 1.3.5 Operations of 'x' and '÷', on Floating-Point Number and Errors
- 1.4 Brief Introduction to Binary Floating Numbers : Fixed Point and Floating Point
- 1.5 Unstable Algorithms and Ill-conditioned Problems
- 1.6 Truncation Error and Taylor's Series
- 1.7 Summary
- 1.8 Solutions/Answers

1.0 INTRODUCTION

In view of the fact that numbers and arithmetic play an important role, not only in our academic matters, but in everyday life also; even children are taught these almost from the very beginning in the school. Numbers play even bigger, rather indispensable, role in our understanding of computer systems, their functioning and their applications; in view of which, the *Number systems* have been discussed in some form or other, in our earlier courses, including BCS-011, BCS-012, MCS-012, MCS-013 and MCS-021.

However, as has been emphasized in Unit 0, using computers to solve numerical problems, requires still deeper understanding of the number systems, specially, *computer number systems*, particularly, in view of the fact that slight lack of understanding of the numbers or lack of attention in their use may lead to disasters involving huge loss of life and property. In view of these facts, first, we discussed conventional numbers, their sets and systems, in some detail in Unit 0, and in more details in Appendix of the Block. In this unit, we discuss the number systems from the point of view of solving mathematical problems using computer as a tool.

Notes (i) : In the discussion of scientific and floating point notation for numbers, we assume that the fixed point arithmetic operations are known to us and are available to us.

(ii) The algorithms are given for only applications, and will not be asked to be reproduced.

1.1 OBJECTIVES

After going through this unit, you should be able to

- explain and use the concepts of fixed point, scientific and floating-point numbers;
- explain and find various types of errors in floating point numbers: chopping, rounding, truncation, underflow and overflow types;

1.3.1 Floating-Point Number Representation

- (i) sign of the exponent is explicitly included in the representation, as is done in the case of mantissa or
- (ii) sign of the exponent is NOT explicitly included in the representation and realized implicitly through, what is generally called, the bias or excess.

- we use '+' and '-' for sign bit/digit, instead of respectively, '0' and '1'.
- also, we use the phrase '*floating number*' instead of '*floating point number*' in our further discussion.
- there can be any number, *but fixed number*, of digits for each of the mantissa/fractional part and exponent part. However, for the sake of uniformity and ease of explanation, *we use 4 digits for mantissa and 2 digits for exponent*.
- The number 10 is adopted in floating point *decimal* number representation, as in scientific notation, and is called ***the base*** of the floating point number system. In *scientific* notation 10 is the only base used. However, in *floating* number representation, base may be any positive integer. The base 2 and the base 16 are frequently used in computer floating number representations; and then, the floating number representations, are called respectively ***binary floating number*** system and ***hexadecimal number system***. ***But, we restrict our discussion, in this section, to only decimal (i.e., base = 10) floating number representation. To be more specific***

where each of $d_1, d_2, d_3, d_4, e_1, e_0$ is a decimal digit. Further, *mantissa* part is interpreted as a decimal fraction, viz., the above one as $.d_1d_2d_3d_4$ and *exponent* part as a decimal whole number, viz., e_1e_0 . For example, the following

29

represents the number $(-0.5048) \times 10^{-18}$, in which the mantissa part 5048 is treated as the fraction 0.5048.

Further, a floating point representation is said to be in *Normalized form*, if the mantissa m satisfies $0.1 \leq m < 1$.

Note the slight difference in the normalized scientific and normalized floating point representations: The mantissa m in the scientific notation is such that $1 \leq m < 10$, whereas in the floating point notation, we have¹ $0.1 \leq m < 1$.

Why Normalized Form?

Advantage of Normalized form of floating number representation is that we can have more numbers that can be represented exactly, instead of just approximately. And, also whenever, a number cannot be represented exactly in normalized form, then, at least, the normalized form gives better approximation of the number than can be given by any of the corresponding un-normalized form. For example, in the four-digit mantissa, if we are to represent the number 243.5, then one of the un-normalized form of representation is 0.0243×10^4 , as shown below:

+	0	2	4	3	+	0	4
---	---	---	---	---	---	---	---

And another is 0.0024×10^5 , as shown below:

+	0	0	2	4	+	0	5
---	---	---	---	---	---	---	---

In each case, the representation is approximate and, not exact. However, the normalized form of representation 0.2435×10^3 , as shown below

+	2	4	3	5	+	0	3
---	---	---	---	---	---	---	---

contains more decimal digits, and, in this case, is exact representation of the number.

Even, in cases, where the number under consideration, say 243.57, may not be exactly represented in our 4-digit mantissa, the normalized representation 0.2435×10^3 , is better approximation of the number 243.57 in floating form than any un-normalized form, say, 0.0243×10^4 or, 0.0024×10^5 .

The numbers, with the *maximum magnitude*, that can be represented by this format, as shown below, are $+(0.9999) \times 10^{99}$ and $-(0.9999) \times 10^{99}$, each of which is quite a large number, and requires 99 decimal digits in fixed point representation.

+/-	9	9	9	9	+	9	9
-----	---	---	---	---	---	---	---

The numbers, with *minimum magnitude*, apart from zero, that can be represented by this format, as shown below, are $+(0.0001) \times 10^{-99} = 10^{-103}$ and $-(0.0001) \times 10^{-99} = -10^{-103}$.

+/-	0	0	0	1	-	9	9
-----	---	---	---	---	---	---	---

However, unless we state otherwise, our floating number is assumed to be, *normalized* floating number. In that case, the numbers, with *minimum magnitude*, apart from zero, that can be represented by the *normalized* format, as shown below,

+/-	1	0	0	0	-	9	9
-----	---	---	---	---	---	---	---

are $+(0.1000) \times 10^{-99} = 10^{-100}$ and $-(0.1000) \times 10^{-99} = -10^{-100}$, each of which is so small in magnitude that, the mass of billionth part of an electron is much larger than the magnitude.

¹ In IEEE-754 standard for representation of binary floating numbers, leading bit is assumed as 1 and is not included explicitly in representation. Further, if the explicitly given part of the mantissa is $d_1 d_2 d_3 d_4 \dots$, where each d_i is a '0' or a '1', the mantissa is interpreted as $1. d_1 d_2 d_3 d_4 \dots$. Therefore, mantissa m in IEEE-754, satisfies $1.0 \leq m < (10.0)_2 = (2)_{10}$, the (decimal) number 2, or the digit 2 in number system with base 10.

Representation of Zero as Floating Point Number : The requirement for the mantissa m to satisfy the condition : $0.1 \leq m < 1$, cannot be met in the case of the number zero.

Thus, 0 is represented only as un-normalized. Then we can easily see that, in our format the number can be represented as

+/-	0	0	0	0	+/-	e_1	e_0
-----	---	---	---	---	-----	-------	-------

where e_1 and e_2 are arbitrary decimal digits. Thus, **zero has 400 possible representations in our format : 10 possible values for each of e_1 and e_0 and two values for each sign.**

Note : It will be explained later that so many possible representations of zero, create serious problems in computations. The best possible solution is to use any of two representations (either '+' or '-') with minimum exponent. **For representing zero, we use the following representation with '+' in mantissa :**

+	0	0	0	0	-	9	9
---	---	---	---	---	---	---	---

Ex. 1 : Find floating point representation, if possible normalized, in the 4-digit mantissa, two digit exponent etc. explained above, if necessary, use approximation, for each of the following Numbers : (i) 23.45, (ii) - 0.00458, (iii) - 8970065, (iv) 0, and (v) 0.0785432. The mantissa is assumed to be a fraction.

1.3.2 Floating-point Number Representation and Errors

Despite the fact that floating numbers can represent a number with very small magnitude to a number with very large magnitude, the floating numbers do not represent all the numbers between the smallest representable number to the largest representable number. For example, the number y with $10^{-100} < y = 10^{-100} + 10^{-(100+4)} < 1$ is such a number; but it cannot be represented exactly in the above format, because, the mantissa 0.100001 of the resultant number $0.100001 \times 10^{-100}$, requires space for six decimal digits.

Actually, in our representation, the number of floating points is only 4×10^6 (there are 6 digital positions, each of which can take 10 values and two positions for sign, each of which can take two values). But the number of real numbers, even between minimum and maximum computer numbers, is infinite. Thus, not every real number can be exactly represented as a floating number. For similar reason, even all integers/rational numbers cannot be represented as floating numbers.

Each of the real numbers, which cannot be represented in a floating number format, if required to be stored in a computer, is *approximated appropriately*, to some neighbouring floating number. However, the use of such an approximation **gives rise to error** in the (exact) the number approximated.

Members of each of the four categories mentioned below, cannot be represented exactly in any floating point number system, and, hence, **necessarily will have error** in representation, even if 100's (but the number is fixed in advance) of digits are used for each of mantissa and for exponent :

- (i) The set of transcendental numbers, like π and e , even when the base is not 10 or 2,
- (ii) The set of irrational number, like $\sqrt{2}$, **because any transcendental number and any irrational number** cannot be represented exactly as a finite string of decimal/binary digits and, hence, as a floating number, *even when the base is not 10 or 2.*
- (iii) **The set of those rational numbers, like $1/3 = 0.33333...$, having repeating, but non-terminating decimal representation,** cannot be represented exactly as a finite decimal/binary string, and hence, cannot be represented as a floating number.
- (iv) Also, even some rational numbers which can be represented as a finite string of **decimal digits**, may not be written as a finite string of **binary digits**. For example, $1/5$ can be written as 0.2, a finite decimal string, but can be written only as an infinite *binary* string : 0.00110011..... **Such numbers can be written exactly as decimal floating numbers, but not as binary floating numbers.**

- (v) Further, with our standard of floating number representation in which mantissa has 4 decimal digits, any decimal number which has at least 5 digits for fractional part, with each of the two outer-most digits not equal to zero, say, $60.065 = 0.60065 \times 10^2$, necessarily requires as many digits, in this case 5 digits, in the mantissa. But the mantissa has pre-assigned fixed number of digits, in our case, 4. Thus, the number 60.065 cannot be exactly represented as a floating number. Hence its representation will have an error.

In general, if, in some decimal floating number representation scheme, mantissa is allotted t decimal digits, and a number has $(t + 1)$ or more decimal digits, with each of the two outer-most digits not equal to zero then, the number cannot be represented exactly as a floating point number. Hence its representation will have an error.

In all the five types of numbers, a member of any of the types cannot be represented exactly as a floating number, and is generally approximated to some near-by floating number, through either of the two processes called : (i) **rounding**, and (ii) **chopping**.

In order to explain the two processes, we assume (for our 4-digit mantissa floating representation scheme), that a number, say, n from any of the above-mentioned classes (except class (iv)) can be represented (with pen and paper), as $n = .d_1 d_2 d_3 d_4 d_5 d_6 \dots \times 10^k$, with integer k satisfying $-99 \leq k \leq 99$, and, in which d_1 and at least one of $d_5, d_6, \dots$ is non-zero. Then, the process of approximation, called

- (i) **Chopping** applies the rule: approximate n by the floating number $.d_1 d_2 d_3 d_4 \times 10^k$, i.e. leave out all the digits d_5, d_6 , etc. Chopping introduces error, called chopping error in the representation of the given number. We can show that the **chopping² error** $\leq 10^{k-4}$... (A)

- (ii) **Rounding** (also, called symmetric rounding), applies the rule :

If $d_5 \leq 4$, then n is approximated by the floating number $.d_1 d_2 d_3 d_4 \times 10^k$ (as in chopping)..... and (otherwise), i.e.

If $d_5 \geq 5$, then n is approximated by the floating number

$$^*(.d_1 d_2 d_3 d_4 + 0.0001) \times 10^k$$

Rounding introduces error, called **Rounding error or Round-off** in the representation of the given number.

We can show that the absolute value of³ (The Round-off error) $\leq 0.5 \times 10^{k-4}$... (D)

² The chopping error $= n - .d_1 d_2 d_3 d_4 \times 10^k = (.d_1 d_2 d_3 d_4 d_5 d_6 \dots \times 10^k) - .d_1 d_2 d_3 d_4 \times 10^k$
 $= (0.0000 d_5 d_6 \dots) \times 10^k = (.d_5 d_6 \dots) \times 10^{k-4} \leq 1 \times 10^{k-4} = 10^{k-4}$

Thus, The chopping error $\leq 10^{k-4}$... (A)

^{*} Floating point addition will be introduced soon. The result may require re-normalization.

³ If $d_5 \leq 4$, the Round-off error, as in the case of chopping

$$\begin{aligned} &= (0.0000 d_5 d_6 \dots) \times 10^k, \\ \text{i.e.} &= (0.d_5 d_6 \dots) \times 10^{k-4}, \quad \text{if } d_5 \leq 4 \\ &\leq (0.4 d_6 \dots) \times 10^{k-4} \leq 0.5 \times 10^{k-4} \quad \text{if } d_5 \leq 4 \dots (B) \end{aligned}$$

If $d_5 \geq 5$, then absolute value of (The Round-off error)

$$\begin{aligned} &= \text{absolute value of } (n - (.d_1 d_2 d_3 d_4 + .0001) \times 10^k) \\ &= \text{absolute value of } ((.d_1 d_2 d_3 d_4 d_5 d_6 \dots \times 10^k) - (.d_1 d_2 d_3 d_4 + .0001) \times 10^k) \\ &= \text{absolute value of } ([.d_1 d_2 d_3 d_4 \times 10^k + (.0000 d_5 d_6 \dots) \times 10^k] - [.d_1 d_2 d_3 d_4 \times 10^k + .0001 \times 10^k]) \\ &= \text{absolute value of } ((.0000 d_5 d_6 \dots) \times 10^k - [.0001 \times 10^k]), \quad \text{if } d_5 \geq 5, \\ &= \text{absolute value of } ((.d_5 d_6 \dots) \times 10^{k-4} \times 10^k - [1 \times 10^{-4} \times 10^k]), \quad \text{if } d_5 \geq 5, \\ &= \text{absolute value of } ([.d_5 d_6 \dots \times 10^{k-4}] - [1 \times 10^{k-4}]), \quad \text{if } d_5 \geq 5, \\ &= \text{absolute value of } ([1 - .d_5 d_6 \dots] \times 10^{k-4}), \quad \text{if } d_5 \geq 5, \end{aligned}$$

Therefore, if $d_5 \geq 5$, absolute value of (The Round-off error)

$$\leq 0.5 \times 10^{k-4} \dots (C)$$

From (B) and (C), we get absolute value of (The Round-off error) $\leq 0.5 \times 10^{k-4}$... (D)

From (A) and (D), we get, $\text{Max (absolute value of (The Round-off error))} \leq (1/2)(\text{Max (absolute value of (The chopping error))})$.

Ex. 2: Find floating point representation, if possible normalized, in the 4-digit mantissa, two digit exponent etc. explained above, if necessary, approximate, *using chopping*, for each of the following Numbers : (i) 23.45, (ii) -0.0045876 , (iii) -8970565 , (iv) 0, and (v) 0.0785432. Also, find absolute error, if any, in each case.

Ex. 3: Find floating point representation, if possible normalized, in the 4-digit mantissa, two digit exponent etc. explained above, if necessary, approximate, *using rounding*, for each of the following Numbers : (i) 23.45, (ii) -0.0045876 , (iii) -8970565 , (iv) 0, and (v) 0.0785432. Also, find absolute error, if any, in each case.

1.3.3 Floating Number Operations of '+', '-', and Errors

Next, we explain how arithmetic binary operations of +, -, $\times$ and $\div$ are applied to a pair of floating numbers. For each operation, first, we explain application of an operation through *some examples and then describe the general process* of applying the operation. *We will find that in many cases, some of the numbers must be temporarily 'un-normalized' and then the result be normalized.*

Let us first explain the operations of '+' and '-' on floating numbers.

Example 2 : Let $x_1 = 0.4273 \times 10^3$ and $x_2 = 0.3400 \times 10^2$, we are to find the result of $x_1 + x_2$.

First of all, we must make the exponents equal, by re-writing the number with smaller exponent in *un-normalized* form : $x_2 = 0.03400 \times 10^3$, so that the exponents of the numbers equal. The numbers be then written so that decimal points align

$$\begin{aligned} x_1 &= 0.4273 \times 10^3, \\ x_2 &= 0.03400 \times 10^3. \end{aligned}$$

Adding the fractions, we get the un-normalized form $x_1 + x_2 = 0.46130 \times 10^3$. Using rounding/chopping, we get the normalized result $x_1 + x_2 = 0.4613 \times 10^3$. In this case, result has no error.

Ex. 4 : Find sum of the two floating numbers $x_1 = 0.4273 \times 10^3$ and $y_2 = -0.3400 \times 10^2$.

Note : The example above, also shows that subtracting x_2 from x_1 is the same as adding x_1 and negative of x_2 . Thus, the operations of adding and subtracting two floating numbers, are almost similar except that when applying the operation of subtraction, first, the sign of the number to be subtracted, is reversed. Rest of the operation for subtraction is the same as that for addition. Therefore, we need to discuss only the process of addition operation on floating numbers

The next example shows how errors are induced by arithmetic operations, in this case, by addition/subtraction of floating numbers.

Example 3 : $x_1 = 0.2473 \times 10^1$ and $x_2 = 0.8340 \times 10^{-1}$. Find $x = x_1 + x_2$.

First of all, we make the exponents equal, by re-writing the number with smaller exponent in un-normalized form : $x_2 = 0.008340 \times 10^1$, so that the exponents of the numbers equal. The numbers be then written so that decimal points align

$$\begin{aligned} x_1 &= 0.2473 \times 10^1, \\ x_2 &= 0.008340 \times 10^1. \end{aligned}$$

Adding the fractions, we get the un-normalized form $x = x_1 + x_2 = 0.255640 \times 10^1$.

However, in order to write x in the following format,

+/ -	d_1	d_2	d_3	d_4	+/ -	e_1	e_0
------	-------	-------	-------	-------	------	-------	-------

we have to use chopping/ rounding. In this case, both give the same result, viz., 2556×10^1 .

However, result when represented as a floating number, *has an error*, with absolute value, as $0.00004 \times 10^1 = 4 \times 10^{1-5} = 4 \times 10^{-4}$.

Absolute Error (Definition) : Let V_a denote the actual value and V_r , the represented value of V_a . Then absolute error in V_a = absolute value of $(V_a - V_r)$

= absolute value of (actual value – represented value)

Relative Error (Definition) : We know the phrase ‘an error of 5 in an actual value of 100’ has quite different meaning from the phrase ‘an error of 5 in an actual value of 100000’. In view of such differences in meaning of error-value, the concept of relative error is introduced as follows : The relative error in V_a = absolute value of $((V_a - V_r)/V_a)$

= absolute value of ((actual value – represented value)/actual value)

= absolute error/ actual value |.

Example 4 : In the example above, the actual value V_a of $x_1 + x_2$ is 2.55640. The represented value V_r of $x_1 + x_2$ is 2.55600. The absolute error in V_a = 0.00040.

The relative error in V_a = $(4 \times 10^{-4})/2.55640 \approx 1.56 \times 10^{-4}$,

(where the symbol ‘ $\approx$ ’ denotes the phrase ‘is approximately equal’)

In order to show how chopping and rounding may give different results, we consider the above Example 4 with minor modification.

Ex. 5 : Find the sum of the two floating numbers $x_1 = 0.2473 \times 10^1$ & $x_2 = 0.8370 \times 10^{-1}$. Further, express the result in normal form, using (i) chopping (ii) rounding. Also find the absolute error.

Ex. 6 : In the exercise above, find the relative error.

Remark : In view of such errors, there is need for utmost care while using numerical methods for solving problems, particularly, from domains of critical applications. Otherwise, it may lead to disasters mentioned in the beginning of this unit⁴.

Next, we explain the general method/algorithm of addition/subtraction of two floating numbers.

Algorithm 1 (to be refined later) for addition/subtraction of two floating numbers :

Let $x_1 = m_1 \times 10^{e_1}$ $x_2 = m_2 \times 10^{e_2}$

Step 0 : If to compute $x_1 - x_2$, Let $x_2 \leftarrow (-1 \times x_2)$; i.e., x_2 is replaced by $(-x_2)$.

Step 1 : Compare exponents : Is $e_1 \leq e_2$?

If yes/true then do

Exchange values of e_1 and e_2 and

Exchange values of m_1 and m_2

⁴ We have already mentioned that if the initial data belongs to any of the five classes mentioned in Section 1.3.2, then it cannot be represented exactly as a floating number, and hence, has to be approximately represented as a floating number. In practical problems such numbers, including π and e , and irrational numbers, like $\sqrt{2}$ occur very frequently. Therefore, there is error in their representations. Also, arithmetic operations may further induce errors. Also, we must note that in the solution of a mathematical problem, not only hundreds, but even, thousands of arithmetic operations may be involved, each possibly, propagating errors. Hence, there is need for utmost care while using numerical methods for solving problems from domains of critical applications. Otherwise, it may lead to disasters mentioned in the beginning of this unit.

; (you may use a temporary variable t for each exchange)

; (at this stage, $e_1 \geq e_2$)

Step 2 : Let $d = e_1 - e_2$; d must be ≥ 0)

Shift digits d places to the right in m_2 , by using extra space for these digits and putting 0's in places of the vacated positions. In this process, the number $m_2 \times 10^{e_2}$ becomes un-normalized.

For example, if $x_2 = 0.2431 \times 10^{e_2}$ with m_2 as 2431 and $d = 2$ then a six-digit un-normalized number $0.002431 \times 10^{2+e_2}$ is created through Step 2.

Step 3 : Add m_1 and new form of m_2 by aligning the decimal points. Let m denote the result of addition of m_1 and m_2 . Then the result $x = x_1 + x_2$, not in normalized form, is $m \times 10^{e_1}$.

Step 4 : Normalise $m \times 10^{e_1}$

end.

How some of these steps may be modified and new steps may be added, will be explained after considering some more examples.

Example 5 : Let $x_1 = 0.9572 \times 10^2$ and $x_2 = 0.8341 \times 10^2$.

To compute $x_1 + x_2 = x = m \times 10^e$.

Using the above algorithm, after Step 3, we get $m = 1.7913$ and the result $x = 1.7913 \times 10^2$. But 1.7913 is not pure fraction, it has an integral part. Hence, m is rewritten as 0.17913×10^1 and $x = 0.17913 \times 10 \times 10^2 = 0.17913 \times 10^3$. Now apply Step 4.

Example 6 : Let $x_1 = 0.9572 \times 10^2$ and $x_2 = -0.9641 \times 10^2$.

Then after Step 3, we get $m = -0.0069$. In view of the fact that m has two leading 0's, it is re-written as $m = -0.69 \times 10^{-2}$. Hence $x = -0.69 \times 10^{-2} \times 10^2 = -0.69 \times 10^0$.

Thus, the result x will be stored in our format as

-	6	9	0	0	+	0	0
---	---	---	---	---	---	---	---

The two examples, discussed just now, explain the need for modifying Step 3. We develop a procedure to be called *normalize-mantissa-n-modify-exponent* (m, e), which will be used to modify Step 3.

In this respect, from Example 5, we notice that, in this case, $1 \leq |m| < 2$ (always)

Then, the example suggests that one of the steps in the procedure should be :

If $|m| \geq 1$ then $\{e \leftarrow e + 1, m \leftarrow m/10\}$

Further, from Example 6 we notice that, in this case, $m = 0$ or $0^{-4} \leq |m| < 1$.

The example suggests that one of the steps in the procedure should be :

If $|m| < 0.1$ then $\{\text{while } |m| < 0.1 \text{ do } (e \leftarrow e - 1, m \leftarrow m \times 10)\}$.

Thus, we have

Procedure normalize-mantissa-n-modify-exponent (m, e); not necessarily having 4-digits mantissa

If $|m| \geq 1$ then $\{e \leftarrow e + 1, m \leftarrow m/10\}$;

If $|m| < 0.1$ then $\{\text{while } |m| < 0.1 \text{ do } (e \leftarrow e - 1, m \leftarrow m \times 10)\}$.

Incorporating the procedure, we get the algorithm

Algorithm 2 for addition/subtraction of two floating numbers, becomes :

Let $x_1 = m_1 \times 10^{e_1}$ $x_2 = m_2 \times 10^{e_2}$

Step 0 : If to compute $x_1 - x_2$, Let $x_2 \leftarrow (-1 \times x_2)$; i.e., x_2 is replaced by $(-x_2)$

Step1 : Compare exponents : Is $e_1 \leq e_2$?

If yes/true then do

Exchange values of e_1 and e_2 and

Exchange values of m_1 and m_2

;(you may use a temporary variable t for each exchange)

;(at this stage, $e_1 \geq e_2$)

Step 2 : Let $d = e_1 - e_2$; d must be ≥ 0)

Shift digits in m_2 , d places to the right, by using extra space for these digits and putting 0's in places of the vacated positions. In this process, the number $m_2 \times 10^{e_2}$ becomes un-normalized.

For example, if $x_2 = 0.2431 \times 10^{e_2}$ with m_2 as 2431 and $d = 2$ then a six-digit un-normalized number $0.002431 \times 10^{2+e_2}$ is created through Step 2.

Step 3 : Add m_1 and new form of m_2 by aligning the decimal points. Let m denote the result of addition of m_1 and m_2 . Then the result $x_1 + x_2$, not in normalized form, is $m \times 10^{e_1}$.

If $m = 0$ # then call procedure make-99 (e_1) else
if $(|m| < 0.1 \text{ or } |m| \geq 1)$, then

call Procedure normalize-mantissa-n-modify-exponent (m, e_1).

Step 4 : Normalise $m \times 10^{e_1}$

end.

we will explain procedure make-99(e) later sometime.

Example 7 : Let $x_1 = 0.6705 \times 10^{-99}$ and $x_2 = 0.6685 \times 10^{-99}$

Problem : To compute the value of $x_1 - x_2$

Applying Algorithm 2, during Step 3, we get **result** $x = x_1 - x_2 = 0.0020 \times 10^{-99}$
 $= 0.2000 \times 10^{-101}$.

But the exponent -101 of the result is less than the smallest exponent -99 that can be stored in our standard format.

+/-	D ₁	d ₂	d ₃	D ₄	+/-	e ₁	e ₀
-----	----------------	----------------	----------------	----------------	-----	----------------	----------------

Underflow : The error, due to the fact that the result cannot be stored, because the resultant number is too small, having its exponent less than least exponent that can be stored, is called *UNDERFLOW*. One of the strategies to handle underflow, is to represent the resultant number by zero, specially, in view of the fact that in the case of underflow, the resultant number is too small. However, an appropriate message may also be given in this respect. And as mentioned earlier, zero should be represented with smallest exponent, i.e., -99 . Thus, in case of underflow, as a strategy, for the resultant number, mantissa $\leftarrow 0$ and for exponent, call procedure make-99.

Ex. 7 : Let $x_1 = 0.6105 \times 10^{-99}$ and $x_2 = -0.6050 \times 10^{-99}$
To attempt to compute the value of $x_1 + x_2$.

Overflow : The error, due to the fact that the result cannot be stored *even approximately*, because the resultant number is too large, as its exponent is more than largest exponent that can be stored, is called **OVERFLOW**. The overflow is more serious type of errors than earlier discussed errors, because, in this case, the result cannot be stored *even approximately*.

There are strategies to handle overflow but, we will not discuss these strategies.

Example 8 : Let $x_1 = 0.6005 \times 10^{99}$ and $x_2 = 0.4150 \times 10^{99}$.

Problem : To compute the value of $x_1 + x_2$.

Applying Algorithm 2, during Step 3, we get **result** $x = x_1 + x_2 = 1.1055 \times 10^{99} = 0.1106 \times 10^{100}$ (after rounding).

But the exponent + 100 of the result is more than the largest exponent + 99 that can be stored in our standard format.

+/-	D ₁	d ₂	d ₃	d ₄	+/-	e ₁	e ₀
-----	----------------	----------------	----------------	----------------	-----	----------------	----------------

Next, we explain what is the procedure make-99 and what is its significance. One reason is that in case of $m = 0$, Step 4 cannot be applied, as 0 cannot be normalized. But, another very serious problem is the fact, which we have already mentioned that the number 0 is not unique in the sense it has 400 possible representations in our floating representation scheme. In the next example, we explain what problems this fact can create and how to solve these problems when $m = 0$.

Example 9 : Let $x_1 = 0.6705 \times 10^{12}$ and $x_2 = -0.6705 \times 10^{12}$ and $x_3 = 0.6685 \times 10^5$

Problem : To compute value of the expression $(x_1 + x_2) + x_3$.

First, we compute $(x_1 + x_2)$, through Algorithm 3,

During Step3 before applying the condition ' $m \neq 0$ ', the result for $(x_1 + x_2)$ in the floating number form is

+	0	0	0	0	+	1	2
---	---	---	---	---	---	---	---

Next, using again Algorithm 3, we add this result to the number $x_3 = 0.6685 \times 10^5$ represented as

+	6	6	8	5	+	0	5
---	---	---	---	---	---	---	---

As per algorithm, the number x_3 , having smaller exponent, will be written in un-normalized form : $0.00000006685 \times 10^{12}$. Then 0.0000×10^{12} is added. The un-normalized result is $0.00000006685 \times 10^{12}$. After rounding/chopping the result to 4-digit mantissa we result zero. But the correct result is 0.6685×10^5 .

Suggested Solution : If $m = 0$, then re-write the number in the following format

+	0	0	0	0	-	9	9
---	---	---	---	---	---	---	---

i.e., we rewrite 0 with minimum possible exponent, in our representation, with - 99.

Next, we define one-statement procedure

Procedure make-99(e) { $e \leftarrow -99$; replace the current value of e by - 99 }

[(in general, if the exponent has k digits in the exponent, then use If $m = 0$ then

$e_1 \leftarrow -99 \dots 9$)

$\underbrace{\hspace{1.5cm}}_k$

(Further, if base is b , instead of 10, then use If $m = 0$

then $e_1 \leftarrow -(b-1)(b-1) \dots b(b-1)$)

$\underbrace{\hspace{1.5cm}}_k$]

Thus, we justify inclusion of procedure make-99 in the Algorithm 3

In view of the above examples, the Step 3 of Algorithm 2 will be modified to give the following :

Algorithm 3 : for addition/subtraction of two floating numbers, becomes :

Let $x_1 = m_1 \times 10^{e_1}$ $x_2 = m_2 \times 10^{e_2}$

Step 0 : If to compute $x_1 - x_2$, Let $x_2 \leftarrow (-1 \times x_2)$; i.e., x_2 is replaced by $(-x_2)$

Step1 : Compare exponents : Is $e_1 \leq e_2$?

If yes/true then do

Exchange values of e_1 and e_2

Exchange values of m_1 and m_2

; (you may use a temporary variable t for each exchange

; (at this stage, $e_1 \geq e_2$)

Step2 : Let $d = e_1 - e_2$; d must be ≥ 0)

Shift digits in m_2 d places to the right, by using extra space for these digits and putting 0's in places of the vacated positions. In this process, the number $m_2 \times 10^{e_2}$ becomes un-normalized.

For example, if $x_2 = 0.2431 \times 10^{e_2}$ with m_2 as 2431 and $d = 2$ then a six-digit un-normalized number $002431 \times 10^{2+e_2}$ is created through Step 2.

Step 3 : Add m_1 and new form of m_2 by aligning the decimal points. Let m denote the result of addition of m_1 and m_2 . Then the result $x_1 + x_2$, not in normalized form, is $m \times 10^{e_1}$.

If $m = 0$ then call procedure make-99(e_1)

else

If ($|m| < .1$ or $|m| \geq 1$) call *Procedure normalize-mantissa-n-modify-exponent* (m, e_1)

If $e_1 \leq -99$

then

call procedure make-99 (e_1), $m \leftarrow 0$ and
write 'There is underflow, as the result is too small to
be written in floating number form; we represent the
result by 0'

else If $e_1 \geq 99$ then write 'There is overflow, as the result is too large to be
written in floating number form'

Step 4 : Normalise $m \times 10^{e_1}$

end.

Example 10 : Revisiting the previous Example 9: (This time applying Algorithm 3 for computing the sum)

Let $x_1 = 0.6705 \times 10^{12}$ and $x_2 = -0.6705 \times 10^{12}$ and $x_3 = 0.6685 \times 10^5$

Problem : To compute the value of the expression $(x_1 + x_2) + x_3$

First, we compute $(x_1 + x_2)$, through Algorithm 3,

During Step 3 before applying the condition ' $m = 0$ ', the result for $(x_1 + x_2)$ in the floating number form is

+	0	0	0	0	+	1	2
---	---	---	---	---	---	---	---

By the new step in Algorithm 3, the format for $(x_1 + x_2)$ becomes

+	0	0	0	0	-	9	9
---	---	---	---	---	---	---	---

To the number with above format, we add $x_3 = 0.6685 \times 10^5$, the format for which is

+	6	6	8	5	+	0	5
---	---	---	---	---	---	---	---

As $-99 \leq 5$, by Step 1 of the Algorithm 3, $e_1 \leftarrow 05$. Application of the rest of the algorithm gives $(x_1 + x_2) + x_3 = 0.6685 \times 10^5 = x_3$, which is the correct required answer.

Note : The proposed solution with Algorithm 3 will give correct results in such cases, because $-99 \leq e$, the exponent of any number. Hence, $e_1 \leftarrow e$.

Loss of Significant Digits : Another Type of Error/Serious Problem.

In order to explain the involved ideas, we recall.

Example 11 : Let $x_1 = 0.9572 \times 10^2$ and $x_2 = -0.9641 \times 10^2$.

Then after Step 3, we get $m = -0.0069$. In view of the fact that m has two leading 0's, it is re-written as $m = -0.69 \times 10^{-2}$. Hence $x = -0.69 \times 10^{-2} \times 10^2 = -0.69 \times 10^0$.

Thus, the result x will be stored in our format as

-	6	9			+	0	0
---	---	---	--	--	---	---	---

in which the two right-most digit space in the mantissa are unfilled, as there are no more digit in the fractional part of the result. However, as per our format for floating numbers, all the 8 space in the format be occupied -- 6 with decimal digits and two with signs. And the best solution, without changing value of the result, is to put 0's in the vacant slots of mantissa so the number in the required floating number format is

-	6	9	0	0	+	0	0
---	---	---	---	---	---	---	---

As discussed just now, filling up the two slots with 0's is just our compulsion, but these 0's do not have any contribution, significance or value in respect of the information about the result.

Loss of Significant Digits (Definition): In a calculated result, using normalized floating numbers, if mantissa has less than 4 digits (in general, less than allotted number for digits in the mantissa), then the remaining digit spaces, on the right hand in the space for the mantissa, are filled with 0's. But these additional 0's do not carry any significance/value/meaning. These 0's have been forced on us by, the otherwise very useful, format. Introduction of these 0's is called loss of significant digits.

1.3.4 Other Computer Arithmetic Problems : Non-associativity of '+' for Floating Numbers

Example 12 : Let $a = 0.1000 \times 10^5$, $b = 0.1050 \times 10^3$ and $c = -0.1000 \times 10^1$ be three floating point numbers. Then $(a + b) + c \neq a + (b + c)$, in floating point arithmetic using rounding, as shown below : For addition, we rewrite b as 0.001050×10^5 (the un-normalized form).

For $a + b = 0.1000 \times 10^5 + 0.001050 \times 10^5 = 0.101050 \times 10^5$

Normalizing using rounding, we get $a + b = 0.1011 \times 10^5$

Then $(a + b) + c = 0.1011 \times 10^5 + (-0.1000 \times 10^1)$, we re-write c in the un-normalized form as $(-0.00001000 \times 10^5)$.

Aligning the decimal points we get

$$(a + b) + c = 0.1011 \times 10^5 + (-0.00001000 \times 10^5) = 0.101090 \times 10^5$$

Normalizing, we get $(a + b) + c = 0.1011 \times 10^5$

... (A)

Similarly, we calculate $(b + c) = 0.104000 \times 10^3$ (UN-NORMALIZED) $= 0.1040 \times 10^3$

Next, similarly, we calculate $a + (b + c) = 0.1000 \times 10^5 + 0.1040 \times 10^3$

$$= 0.1000 \times 10^5 + 0.001040 \times 10^5$$

$$a + (b + c) = 0.101040 \times 10^5 \text{ (UN-NORMALIZED)} = 0.1010 \times 10^5 \quad \dots (B)$$

Hence, from (A) and (B), we get

$$(a + b) + c \neq a + (b + c).$$

Ex. 8 : Is '+' associative when : $a = 0.2134 \times 10^5$, $b = 0.2354 \times 10^3$ and $c = -0.2142 \times 10^1$ be three floating point numbers to be added, in this order.

For more General cases of choices of a , b and c^5 , so that $(a + b) + c \neq a + (b + c)$

1.3.5 Operations of '×' and '÷', on Floating-Point Number, and Errors

Multiplication and division are relatively simpler operations on floating numbers. Therefore, we give the algorithm and then consider some suitable examples. We consider the operations on only normalized numbers. If any of the involved numbers, is not normalized, we first normalize the number.

Note : We have already mentioned that for arithmetic operations on floating numbers, arithmetic operations, including multiplication and division, on fixed point numbers is available to us.

Algorithm : Multiplication of two (normalized) Floating Numbers.

Let $a = m_1 \times 10^{e_1}$ and $b = m_2 \times 10^{e_2}$ be two given floating normalized number.

Let $c = a \times b = m \times 10^e$

Begin

Step1 if either $m_1 = 0$ or $m_2 = 0$,

then $m \leftarrow 0$, $e \leftarrow -99$; least exponent, the result is 0.

else

⁵ In more general case, For our 4-decimal mantissa format,

If $a = .x_1x_2x_3x_4 \times 10^{n+4}$ $b = .y_1y_2y_3y_4 \times 10^{n+2}$ and $c = -.z_1z_2z_3z_4 \times 10^n$

Then $a + b = [.x_1x_2(x_3 + y_1)(x_4 + y_2)y_3y_4] \times 10^{n+4}$; un-normalized 6-digit mantissa
 $= [.x_1x_2(x_3 + y_1)(x_4 + y_2 + 1)] \times 10^{n+4}$; if $5 \leq y_3$,

Therefore

$$(a+b)+c = [.x_1x_2(x_3 + y_1)(x_4 + y_2 + 1)] \times 10^{n+4} + [-z_1z_2z_3z_4] \times 10^n; \text{ if } 5 \leq y_3$$

$$= \{ [.x_1x_2(x_3 + y_1)(x_4 + y_2 + 1)] + (-.0000z_1z_2z_3z_4) \} \times 10^{n+4}$$

$$= \{ [.x_1x_2(x_3 + y_1)(x_4 + y_2 + 1)] \} \times 10^{n+4}$$

Further if we take digit $z_1 = 1, 2, 3, 4$, so that $0 \leq y_3 - z_1 \leq 4$ let us assume $z_1 = 2$, Then

$(b+c) = (.y_1y_2(y_3 - z_1)(y_4 - z_2)z_3z_4) \times 10^{n+2}$; un-normalized 6-digit mantissa

Therefore, $a+(b+c) = .x_1x_2x_3x_4 \times 10^{n+4} + (.y_1y_2(y_3 - z_1)(y_4 - z_2)z_3z_4) \times 10^{n+2}$

$= [.x_1x_2(x_3 + y_1)(x_4 + y_2)(y_3 - z_1)(y_4 - z_2)] \times 10^{n+4}$; un-normalized 6-digit mantissa

$= [.x_1x_2(x_3 + y_1)(x_4 + y_2)] \times 10^{n+4}$; if $0 \leq y_3 - z_1 \leq 4$

Thus, in more general case, If $a = .x_1x_2x_3x_4 \times 10^{n+4}$ $b = .y_1y_2y_3y_4 \times 10^{n+2}$ and

$c = -.z_1z_2z_3z_4 \times 10^n$ and $5 \leq y_3$, $0 \leq y_3 - z_1 \leq 4$

then $(a + b) + c \neq a + (b + c)$.

Step 2 : [Let $m \leftarrow m_1 \times m_2$ (using temporarily 8-digit registers and then rounding or chopping to 4 digits),

(at this stage, $0.01 \leq |m| < 1$)

$e \leftarrow e_1 + e_2$.

If $|m| < 0.1$ Then (at most, one-step normalization)

$[m \leftarrow 10 \times m ; e \leftarrow e - 1]$

]

End

Example 13 : To find the product of the two numbers : $a = 0.4031 \times 10^3$ and $b = -0.3100 \times 10^{-4}$.

Then as $a \neq 0 \neq b$, $m \leftarrow (0.4031) \times (-0.3100) = -0.12496100$

(un-normalized, using 8-digit registers)

$e \leftarrow (3 + (-4)) = -1$

On normalizing m through, say rounding, we get

$m \leftarrow (0.4031) \times (-0.3100) = -0.1250$ and $e \leftarrow -1$.

The required number in floating form is $c = -0.1250 \times 10^{-1}$.

Ex. 9 : To find the product of the two numbers : $a = -0.4031 \times 10^3$ and $b = -0.1101 \times 10^{-4}$.

Algorithm : Division of a floating number by another floating number (both normalized)

Let $a = m_1 \times 10^{e_1}$ and $b = m_2 \times 10^{e_2}$ be two given floating normalized numbers. And a is to be divided by b .

Let $c = a \div b = m \times 10^e$

Begin

Step1 if

$m_2 = 0$,

then write 'The division by zero is not possible'

else if

$m_1 = 0$ then $m \leftarrow 0$, $e \leftarrow -99$; least exponent, the

else

[Step2 : Let $m \leftarrow m_1 \div m_2$ (using temporarily 8-digit registers and then rounding or chopping to 4 digits),

(at this stage, $0.1 \leq |m| < 10$)

$e \leftarrow e_1 - e_2$.

If $1 \leq |m|$ then (at most, one-step normalization)

$[m \leftarrow m \div 10 ; e \leftarrow e + 1]$

]

End

Example 14 : Let $a = 0.1101 \times 10^3$ and $b = -0.3326 \times 10^{-4}$

To find the value of $c = a \div b$.

Solution : As $a \neq 0 \neq b$, therefore

$$m \leftarrow (0.1101) \div (-0.3326) = -0.36106\dots \text{ (un-normalized 8 digit register)}$$

$$\text{and } e \leftarrow 3 - (-4) = +7$$

as $m < 1$, no adjustments of m and e are required

On normalization $m \leftarrow -0.3611$ (using rounding) and $e = 7$,

we get $m = 0.3021$. Therefore, $c = -0.3021 \times 10^7$.

Ex. 10 : Let $a = -0.3326 \times 10^{-4}$ and $b = 0.1101 \times 10^3$ and to find the value of $c = a \div b$.

Problems in Floating number multiplication and division – overflow, underflow

Overflow : We know that in our 4-digit mantissa and 2-digit exponent, overflow occurs when the exponent, which is an integer for floating numbers, exceeds 99.

Example 15 : Of overflow due to multiplication :

Let $a = 0.2003 \times 10^{53}$, $b = -0.5200 \times 10^{49}$ and

Let $c = a \times b = m \times 10^e$, where m is in normalized floating form, then

$$\text{If } 0.1 \leq |m| \text{ then } m = m_1 \times m_2, e \leftarrow e_1 + e_2 = 53 + 49 = 103$$

$$\text{else } m = m_1 \times m_2 \times 10 \text{ and } e \leftarrow e_1 + e_2 - 1 = 53 + 49 - 1 = 102$$

In both cases e cannot be stored in the 2-decimal digit space allotted to e

or, alternatively, we can show the overflow through the following argument

[through explicit calculations, we have : As $0.1 < |0.2003 \times (-0.5200)| < 1$ therefore $m = 0.2003 \times (-0.5200)$ and $e = 53 + 49 = 103 \geq 99$ cannot be stored in the 2 decimal digit space allotted to e]. Hence, overflow. **In general**⁶.

Example 16 : Of overflow due to Division : $a = 0.2003 \times 10^{53}$, $b = -0.5200 \times 10^{-49}$

If $c = a \div b = m \times 10^e$, and m is in normalized floating form, then

$$\text{If } 1 \leq |m| \text{ then } m = m_1 \div m_2, e \leftarrow e_1 - e_2 + 1 = 53 - (-49) + 1 = 103$$

$$\text{else } m = m_1 \div m_2, e \leftarrow 53 - (-49) = 102$$

In both cases e cannot be stored in the 2 decimal digit space allotted to e .

or, alternatively, we can show the overflow through the following argument

[through explicit calculations, we have : As $0.1 < |0.2003 \div (-0.5200)| = 0.385\dots < 1$, therefore $m = 0.2003 \div (-0.5200)$ and $e = 53 - (-49) = 103 \geq 99$, cannot be stored in the

⁶ General conditions for overflow due to multiplication

$$\text{Let } a = m_1 \times 10^{e_1}, b = m_2 \times 10^{e_2}.$$

If, $c = a \times b = m \times 10^e$, in floating number form, then

Either $m = m_1 \times m_2$ and $e = (e_1 + e_2)$

Or $m = (m_1 \times m_2) \times 10$ and $e = (e_1 + e_2) - 1$.

Thus in multiplication, overflow shall occur, if $(e_1 + e_2) - 1 \geq 99$, in other words

in multiplication, overflow shall occur, if $(e_1 + e_2) \geq 100$,

where $-99 \leq e_1 \leq 99$

2 decimal digit space allotted to e]. For overflow to occur, more General cases of choices of a, b ⁷.

Underflow : We know that in our 4-digit mantissa and 2-digit exponent, underflow occurs when the exponent, which is an integer for floating numbers, is (strictly) less than 99.

Ex. 11 : Check whether there is underflow due to multiplication :

$$a = 0.2003 \times 10^{-53}, b = -0.5200 \times 10^{-49}?$$

Ex. 12 : Check whether there is underflow due to Division of a by b :

$$a = 0.2003 \times 10^{-53}, b = -0.5200 \times 10^{-49}.$$

Non-associativity of Multiplication of Floating Numbers:

Example 17 : Let $a = 0.1000 \times 10^{80}$, $b = 0.1000 \times 10^{30}$, $c = 0.1000 \times 10^{-50}$

Then $(a \times b) = 0.1000 \times 10^{80+30-1} = 0.1000 \times 10^{109}$

There is an overflow, hence, $(a \times b)$ is not a floating number.

Therefore, $(a \times b) \times c$ is not defined.

On the other hand $(b \times c) = 0.1000 \times 10^{30+(-50)-1} = 0.1000 \times 10^{-21}$

Hence, $a \times (b \times c) = 0.1000 \times 10^{80-21-1} = 0.1000 \times 10^{58}$

Thus, $(a \times b) \times c \neq a \times (b \times c)$.

Ex. 13 : $a = 0.1000 \times 10^{-80}$, $b = 0.1000 \times 10^{-30}$, $c = 0.1000 \times 10^{50}$

Check whether ' $\times$ ' is associative for these three numbers, in this order.

Non-distributivity of ' $\times$ ' over ' $+$ ' : To show, for some floating numbers a, b and c , we can have $a \times (b + c) \neq a \times b + a \times c$

Example : Let $a = 0.2222 \times 10^2$, $b = -0.1001 \times 10^3$, $c = 0.1002 \times 10^3$

Then $b + c = 0.0001 \times 10^3 = 0.1000 \times 10^0$

Therefore, $a \times (b + c) = 0.2222 \times 10^2 \times (0.1000 \times 10^0) = 0.2222 \times 10^1$... (A)

$$a \times b = 0.2222 \times 10^2 \times (-0.1001 \times 10^3) = -0.2224222 \times 10^4$$

(un-normalized, using temporarily 8-digits in mantissa)

$$= -0.2224 \times 10^4 \text{ (normalized, after rounding)}$$

$$a \times c = 0.2222 \times 10^2 \times (0.1002 \times 10^3) = 0.2226444 \times 10^4$$

(un-normalized, using temporarily 8-digits in mantissa)

$$= 0.2226 \times 10^4 \text{ (normalized, after rounding)}$$

$$a \times b + a \times c = 0.0001 \times 10^3 = -0.2224 \times 10^4 + 0.2226 \times 10^4$$

$$= 0.0002 \times 10^4 = 0.2000 \times 10^1$$
 ... (B)

$$a \times (b + c) \neq a \times b + a \times c.$$

⁷ General conditions for overflow due to division

$$\text{Let } a = m_1 \times 10^{e_1}, b = m_2 \times 10^{e_2}.$$

If, $c = a \div b = m \times 10^e$, in floating number form, then

Either $m = m_1 \div m_2$ and $e = (e_1 - e_2)$

Or $m = (m_1 \div m_2)/10$ and $e = (e_1 - e_2) + 1$.

Thus in division, overflow shall occur, if $(e_1 - e_2) + 1 \geq 99$, in other words

in division, overflow shall occur, if $(e_1 - e_2) \geq 98$,

where $-99 \leq e_1 \leq 99$

Example 18 : Let $a = 0.1223 \times 10^1$, $b = -0.1021 \times 10^3$, $c = 0.1022 \times 10^3$

Then $b + c = 0.0001 \times 10^3 = 0.1000 \times 10^0$

Therefore, $a \times (b + c) = 0.1223 \times 10^1 \times (0.1000 \times 10^0) = 0.1223 \times 10^0$

$$a \times b = 0.1223 \times 10^1 \times (-0.1021 \times 10^3) = -0.01248683 \times 10^4$$

(un-normalized, using temporarily 8-digits in mantissa)

$$= -0.1249 \times 10^3 \text{ (normalized, after rounding)}$$

$$a \times c = 0.1223 \times 0.1022 \times 10^3 = 0.01249906 \times 10^4$$

(un-normalized, using temporarily 8-digits in mantissa)

$$= 0.1250 \times 10^3 \text{ (normalized, after rounding)}$$

$$a \times b + a \times c = 0.0001 \times 10^3 = 0.1000 \times 10^0$$

$$a \times (b + c) \neq a \times b + a \times c$$

Therefore, $a \times b + a \times c = -0.1248683 \times 10^4 + 0.1249906 \times 10^4$.

Ex. 14 : Check distributivity of $\times$ over $+$, where $a = 0.2706 \times 10^1$, $b = -0.7425 \times 10^2$, $c = 0.7445 \times 10^2$.

For more General cases of choices of a , b and c ⁸, so that $a \times (b + c) \neq a \times b + a \times c$.

1.4 BRIEF INTRODUCTION TO BINARY NUMBER REPRESENTATIONS : FIXED POINT AND FLOATING POINT

The decimal numbers having been in use over centuries, have become quite natural, almost intuitive to the human beings. This is why, we discussed the decimal numbers, both in fixed and floating formats. Also, for the same reason, the essential ideas, of floating number representation, which is not very frequently used in day-to-day business, have first been explained for decimal numbers.

Having understood the essential ideas behind floating representation format (as well as behind fixed) through the well-understood decimal numbers, we can now discuss these formats for binary numbers. The understanding of these formats for binary is essential in view of the fact that the alphabet of most of the computer systems is binary, i.e., $\{0, 1\}$. In other words, every entity within most of the computer systems, is necessarily a string of 0's and 1's. Hence, a number in any numeral system (decimal, Latin etc.) is also represented within such a computer as a binary string, say, *10010110*. A binary string when used for representing a number may represent different numbers according to different schemes of interpreting the same string as a number.

The schemes for interpreting a string as a number, at the top level may be categorized into two classes : (i) *Fixed point* and (ii) *Floating point* of schemes of representation.

Fixed point binary representation is similar to the Fixed point *decimal* point representation, except now only two digits, viz., 0 and 1, are used for representing numbers and in place of decimal point, we use binary point.

⁸ if the following conditions are satisfied:

- (i) Let a be an arbitrary normalized floating number having exponent, say e .
- (ii) floating numbers b and c are of opposite signs
- (iii) b and c have equal decimal digits in $t = 1, 2, 3$ or 4 corresponding leading positions in the mantissa,

then $a \times (b + c)$ and $a \times b + a \times c$ differ by a magnitude of order $10^{e-(4-t)}$

However, Fixed-point binary schemes, can be obtained from the decimal fixed point representation, by minor modifications, including (i) change of the base from 10 to 2 and (ii) restricting of the digits to 0 and 1. Hence these have same problems as decimal fixed point scheme.

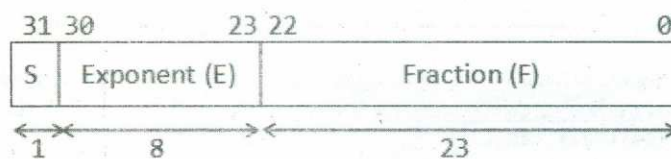
Floating-Point binary Representation schemes are obtained by modification of the floating point *decimal notation* by restricting the digits to be used to only 0 and 1 and by change of the base from 10 to 2. The various concepts, issues and problems in respect of binary floating numbers can, without difficulty, be translated/modified/transformed from the corresponding ones for decimal floating numbers. Therefore, except for a brief discussion of the **IEEE Standard 754 Floating Point Numbers**, we will not discuss these topics any more.

IEEE Standard 754 floating point representation is the most common representation today for real numbers on computers, including Intel-based PC's, Macintoshes, and most Unix platforms. Here, we have a brief overview of IEEE floating point and its representation. According to the Standard, there are two types of floating numbers – single precision and double precision. For single precision, 32-bits are used, whereas for double precision 64-bits are used.

Below we briefly explain only 32-bit format. The bits are indexed from right to left – the right-most bit has the index 0. In the 32-bit (single precision) representation, the left-most bit is indexed as 31.

In 32-bit single-precision floating-point representation:

- The most significant bit (with index 31) is the *sign bit* (S), with 1 for negative numbers and 0 for positive numbers.
- The following 8 bits represent *exponent* (E).
- The remaining 23 bits represents *fraction* (F).



32-bit Single-Precision Floating-point Number

The Base : (Implicit) is 2

The Exponent : Next 8 bits (with index 30-23) to the right of the sign bit are used for exponent. No sign is used in the exponent. Thus, *binary value* of the exponent is between 0 and 255 (both included). But, in order to represent even negative powers of 2 in the floating representation, 127 is subtracted from the binary value, so that the binary value from 0 to 255 contribute – 127 (represented by 0000 0000) to 128 (represented by 1111 1111) to the value of the power of 2. For example, if the sequence of 8 bits in the exponent, is 1010 1010 with binary value $128 + 32 + 8 + 2 = 170$, but contributes $170 - 127 = 43$, i.e., the value contributed by the mantissa is multiplied 2^{43} . The number 127 is called the Bias.

The Mantissa in IEEE standard 754, is interpreted slightly differently. The sequence of 23 bits, say, $a_{22} a_{21} a_{20} \dots a_2 a_1 a_0$ is interpreted as the binary number $1 . a_{22} a_{21} a_{20} \dots a_2 a_1 a_0$ or the binary number, which is the result of $(1 + . a_{22} a_{21} a_{20} \dots a_2 a_1 a_0)$. This 1 is (implicitly) assumed without its explicit inclusion in the representation.

Summary of single precision (32-bit) IEEE standard 754 is :

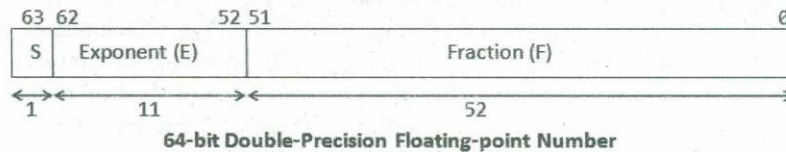
	Sign bit	Exponent	Fraction	Bias
Single Precision	1 [31]	8 [30-23]	23 [22-00]	127

Range of real numbers represent-able in 32-bit normalized format is

$$\pm 2^{-126} \text{ to } (2-2^{-23}) \times 2^{127} \quad \pm \sim 10^{-44.85} \text{ to } \sim 10^{38.53}$$

The representation scheme for 64-bit double-precision is similar to the 32-bit single-precision:

- The most significant bit is the *sign bit* (S), with 0 for negative numbers and 1 for positive numbers.
- The following 11 bits represent *exponent* (E).
- The remaining 52 bits represents *fraction* (F).



Summary of double precision (64-bit) IEEE standard 754 is :

	Sign bit	Exponent	Fraction	Bias
Double Precision	1 [63]	11 [62-52]	52 [51-00]	1023

Range of real numbers represent-able in 64-bit normalized format is

$$\pm 2^{-1022} \text{ to } (2-2^{-52}) \times 2^{1023} \quad \pm \sim 10^{-323.3} \text{ to } \sim 10^{308.3}$$

Example 19 : Suppose that we are to find the value of the representation of the 32-bit pattern is 1 1000 0001 011 0000 0000 0000 0000 0000, with S = 1, E = 1000 0001, F = 011 0000 0000 0000 0000.

In the *normalized form*, the actual fraction is normalized with an implicit leading 1 in the form of 1.F. In this example, the actual fraction is 1.011 0000 0000 0000 0000 0000 = $1 + 1 \times 2^{-2} + 1 \times 2^{-3} = 1.375$ (decimal).

The sign bit represents the sign of the number, with S = 0 for positive and S = 1 for negative number. In this example with S = 1, this is a negative number, i.e., -1.375 (decimal).

In normalized form, the actual exponent is **E-127 (so-called excess-127 or bias-127)**. This is, because, we need to represent both positive and negative exponent. With an 8-bit E, ranging from 0 to 255, the excess-127 scheme could provide actual exponent of -127 to 128. In this example, E-127 = 129. The actual = $129 - 127 = 2$ (decimal).

Hence, the number represented is $-1.375 \times 2^2 = -5.5$ (decimal).

Example 20 : Suppose that IEEE-754 32-bit floating-point representation pattern is 1 10000000 111 0000 0000 0000 0000 0000. To find its value in decimal form.

Sign bit S = 1 $\Rightarrow$ negative number

E = 1000 0000(binary) = 128(decimal) (in normalized form)

Fraction is 1.111(binary) (with an implicit leading 1) = $1 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 1.875$ (Decimal).

The number is $-1.875 \times 2^{(128-127)} = -3.75$ (Decimal).

Example 21 : Suppose that IEEE-754 32-bit floating-point representation pattern is 0 01111110 110 0000 0000 0000 0000 0000.

Sign bit $S = 0 \Rightarrow$ positive number

$E = 0111\ 1110$ (binary) = 126 DECIMAL (in normalized form)

Fraction is 1.11 (binary) (with an implicit leading 1) = $1 + 2^{-1} + 2^{-2} = 1.75$ (DECIMAL).

The number is $1.75 \times 2^{(126-127)} = 0.75$ (DECIMAL).

Ex. 15 : To find the value in decimal of the IEEE-754 32-bit floating-point representation pattern is 0 01111110 100 0000 0000 0000 0000 0001.

Ex. 16 : To find the value in decimal of the IEEE-754 32-bit floating-point representation pattern in IEEE-754 32-bit floating-point representation pattern is 0 00000000 000 0000 0000 0000 0000 0011.

1.5 UNSTABLE ALGORITHMS AND III-CONDITIONED PROBLEMS

Next, we discuss two quite similar, but distinct, concepts, viz.

- (i) Unstable Algorithm
- (ii) Ill-conditioned/ Unstable Problem

First, we consider **Unstable Algorithm**. Consider the problem of evaluating the function $f(x) = \sqrt{(x^2 + 1)} - 1$. Let us assume that in our calculations, we are using 3 significant digits. Then

$$f(0.25) = \sqrt{((0.25)^2 + 1)} - 1 = \sqrt{(1.06)} - 1 = 1.03 - 1 = 0.03$$

But, the answer, correct to three significant digits, is 0.0308. In the process two significant digits are lost and the error in computation is about 3%. The reason, in this particular case, is that nearly equal numbers, viz., $\sqrt{(1.06)}$ and 1 are involved in subtraction.

Subtractive Cancellation : The process that involves subtraction of nearly equal numbers which, at the same time, leads to large relative deviation from the correct result (including loss of significant digits), is called subtractive cancellation.

The problem of subtractive cancellation can, in some cases, be avoided, by using some other equivalent form, instead of the form that leads to subtractive cancellation.

In this case, $f(x) = \sqrt{(x^2 + 1)} - 1$, can be re-written as $f(x) = x^2 / (\sqrt{(x^2 + 1)} + 1)$, (using

$$a - b = \frac{a^2 - b^2}{a + b}, \text{ if } a \neq -b, \text{ which does not involve subtraction of nearly two equal numbers.}$$

$$f(0.25) = (0.25)^2 / (\sqrt{(0.25)^2 + 1} + 1) = 0.0625 / (1.03 + 1) \approx 0.0308.$$

Thus, just by **re-formulation of the function, hence, of the algorithm**, we get much better approximation of the correct result. The source of error was the form/algorithm for the function. **In such cases, the earlier algorithm is called unstable algorithm.**

Definition : One of the characteristics of **Unstable Algorithm** is that the unstable algorithm does not give good result, but there is an alternative algorithm, which gives good result.

⁹ Page 9, Introduction to Numerical Analysis (II Edition) by C-E Froberg

⁹
Round numbers are
always false.
Samuel Johnson

One of the reasons for an algorithm to be unstable is that it generally involves (i) subtraction of nearly two equal numbers, (ii) division by a number of very small magnitude, or (iii) multiplication by a number of a large magnitude. In an alternative algorithm, in which these operations are replaced through some reformulation, as was done in the example above, the (new algorithm) may give desirably better approximation.

Example 22 : Consider the problem of finding the roots of a quadratic equation

$$x^2 + a x + e = 0,$$

where we use 4-digit mantissa, when $a^2 \gg |e|$, i.e. a^2 is much larger than $|e|$.

The two roots of the equation are $x_1 = (-a + \sqrt{a^2 - 4e})/2$ and $x_2 = (-a - \sqrt{a^2 - 4e})/2$.

In view of the fact that e is much smaller than a^2 , depending upon sign of a , one of x_1 or x_2 may involve subtractive cancellation.

For example, for a particular case of the equation

$$x^2 - 1000 x + 25 = 0,$$

the two roots are $x_1 = (1000 + \sqrt{(10^3)^2 - 4 \times 25})/2 = (10^3 + \sqrt{10^6 - (10^2)^2})/2$ and $x_2 = (10^3 - \sqrt{10^6 - (10^2)^2})/2$.

Using 4-digit mantissa floating point arithmetic,

$$10^6 = 0.1000 \times 10^7 \text{ and } 10^2 = 0.1000 \times 10^3$$

Thus, $10^6 - (10^2)^2 = 0.1000 \times 10^7$. Thus $\sqrt{10^6 - (10^2)^2} = 0.1000 \times 10^4$.

Therefore, the two roots are (in floating point representation)

$$x_1 = (0.1000 \times 10^4 + 0.1000 \times 10^4)/2 \text{ and } x_2 = (0.1000 \times 10^4 - 0.1000 \times 10^4)/2,$$

giving $x_1 = 0.1000 \times 10^4 = 1000$ and $x_2 = 0.0000 \times 10^4 = 0$.

To avoid subtractive cancellation, we may use the fact that product of the roots = coefficient of x^0 in the quadratic equation = 25 and computed value of one of the roots $x_1 = 0.1000 \times 10^4$.

Therefore, the other root $x_2 = 25/(0.1000 \times 10^4) = 0.2500 \times 10^{-1}$.

However, in some cases, as discussed below, there cannot be an alternative algorithm to remedy the underlying difficulty in solving properly the problem under consideration.

Example 23 : Consider the system of two linear equations

$$x + 2y = 3$$

$$0.499 x + 1.001 y = 1.5,$$

whose solution is $x = 1$ and $y = 1$.

Next consider the set of equations

$$x + 2 y = 3$$

$$0.5 x + 1.001 y = 1.5,$$

obtained by changing coefficient of x from 0.499 to 0.5 in second equation. The unique solution of this set of equations is $x = 3$ and $y = 0$. Thus, the small change of coefficient from 0.499 to 0.5 has caused the solution to change drastically.

The underlying difficulty in such cases is that just through slight changes in (input) data set, the nature of solution is completely changed. In respect of such difficulty, we note that

- (i) There cannot be any algorithmic change/modification, that can remedy the problem, i.e., difficulty is independent of the algorithm. The difficulty is inherent in the problem itself,

- (ii) Such minor changes cannot be avoided in practical problems, because of various factors, including error of judgement, limitation of measurements etc.

Such problems are called ill-conditioned. Formally, a problem is said to be **ill-conditioned**, if slight change in the input data leads to substantial change in the result. In other words, a problem is said to be ill-conditioned in which the solution is very sensitive to changes in the data. Some authors use the term **Unstable Problem**, (*different from Unstable Algorithm/Solution*) in place of ill-conditioned problems.

A problem is said to be **well-conditioned or Stable Problem**, if small changes in the input data always lead to small changes in the results.

Ex. 17 : Discuss, whether in arithmetic, with 4 significant digits, the problem of solving the following system of linear equations

$$(2.000) x + (0.6667) y = 2.000$$

$$(1.000) x + (0.3333) y = 1.000,$$

is ill-conditioned or not.

1.6 TRUNCATION ERRORS AND TAYLOR'S SERIES¹⁰

In mathematics, a **Taylor series** is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point¹¹. If the Taylor series is centered at zero, then that series is also called a **Maclaurin series**. A sum of *finite number of initial terms* of the Taylor series of a function is called a **Taylor polynomial**.

In view of the fact that a computer can calculate the sum of *only a finite number of terms* of a Taylor's series of a function, Taylor polynomials are used to approximate a function. As mentioned earlier also, the process of approximation of an infinite series/process by a finite sub-series/appropriate finite process is called **Truncation**. Thus, an approximation of a function by Taylor's **polynomial** from the Taylor's **series** expansion of the function is also truncation. As truncation is achieved from a finite part of an infinite entity, it gives rise to an error called **Truncation Error**.

Next, we illustrate these concepts through suitable examples:

The Taylor series of a real valued function $f(x)$ that is infinitely differentiable at a real number a is the power series (*called power series, because, powers square, cube, and higher powers occur in the series*) :

$$f(a) + \frac{f'(a)}{1!} (x-a) + \frac{f''(a)}{2!} (x-a)^2 + \frac{f^{(3)}(a)}{3!} (x-a)^3 + \dots \quad \dots (A)$$

which can be written in the more compact sigma notation as

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

¹⁰The following material in this subunit is adapted version of excerpts from Wikipedia, the free encyclopedia.

¹¹ The Taylor series of a function is the limit of that function's Taylor polynomials, provided that the limit exists. **A function may not be equal to its Taylor series, even if its Taylor series converges at every point.** There are certain conditions, including existence of n^{th} derivatives of the function, for all n , for the Taylor's series of the function to exist, and then to be equal to the value of the function at some point.

where $n!$ denotes the factorial of n and $f^{(n)}(a)$ denotes the n th derivative of f evaluated at the point a . The derivative of order zero of f is defined to be f itself and $(x-a)^0$ and $0!$ are both defined to be 1.

In the case that $a = 0$, the series is also called a **Maclaurin series**. Thus, the Maclaurin series of a function $f(x)$, obtained by replacing a by 0 in the above, is given as

$$f(0) + \left(\frac{x}{1!}\right)f'(0) + \left(\frac{x^2}{2!}\right)f''(0) + \left(\frac{x^3}{3!}\right)f'''(0) + \dots \quad \dots (B)$$

Example 24 : Find the Taylor/Maclaurin series for $(1-x)^{-1}$ at $x = 0$.

Solution : As the n^{th} derivative of $(1-x)^{-1}$, i.e.,

$$d^n ((1-x)^{-1})/dx^n = n!/(1-x)^{-(n+1)}$$

Therefore, the value of $d^n ((1-x)^{-1})/dx^n$ at $x = 0$ is $n!$

Using in (B) above, we get

$$1 + \left(\frac{x}{1!}\right)1! + \left(\frac{x^2}{2!}\right)2! + \left(\frac{x^3}{3!}\right)3! + \dots$$

giving the geometric series. Thus, the value of $(1-x)^{-1}$ around $x = 0$, is given by the infinite series

$$1 + x + x^2 + x^3 + \dots$$

The value of $(1-x)^{-1}$, say, at $x = 0.1$ is exactly given by infinite series

$$1 + (0.1) + (0.1)^2 + (0.1)^3 + \dots$$

Example 25 : Find truncation error in approximating $(1-x)^{-1}$, say, at $x = 0.1$, by taking first three terms.

Solution : Now suppose, we approximate the value of $(1-x)^{-1}$, say, at $x = 0.1$ by first three terms, viz., $1 + (0.1) + (0.1)^2 + (0.1)^3 = 1.111$.

Thus truncation error in the value of $(1-x)^{-1}$, say, at $x = 0.1$ is 0.000111....

Ex. 18 : Find the Maclaurin series for $\log(1-x)$, where $\log$ denotes the *natural logarithm*.

Ex. 19 : (of truncation error) Using the Maclaurin series for $\log(1-x)$, where $\log$ denotes the *natural logarithm*, find approximate values of (i) $\log(2)$ and (ii) $\log(1)$, by taking 4 terms and find respective truncation errors.

Example 26 : Find Maclaurin's series of $f(x) = e^x$, around $x = 0$.

Solution : In order to find Maclaurin's series of $f(x) = e^x$, around $x = 0$, we know that

$$f^{(n)}(x) = \frac{d^n(e^x)}{dx^n} = e^x. \text{ Hence } \left(\frac{d^n(e^x)}{dx^n}\right)_{x=0} = (e^x)_{x=0} = 1.$$

Therefore, Maclaurin's expansion of e^x around $x = 0$, is obtained by replacing $f^{(n)}(x)$ by 1, in the following :

$$f(0) + \left(\frac{x}{1!}\right)f'(0) + \left(\frac{x^2}{2!}\right)f''(0) + \left(\frac{x^3}{3!}\right)f'''(0) + \dots$$

We get
$$e^x = 1 + \left(\frac{x}{1!}\right) \times 1 + \left(\frac{x^2}{2!}\right) \times 1 + \left(\frac{x^3}{3!}\right) \times 1 + \dots$$

$$= 1 + \left(\frac{x}{1!}\right) + \left(\frac{x^2}{2!}\right) + \left(\frac{x^3}{3!}\right) + \dots$$

Example 27 : From previous example, the Taylor/Maclaurin series for e^x around $x = 0$ is given by

$$1 + \left(\frac{x}{1!}\right) + \left(\frac{x^2}{2!}\right) + \left(\frac{x^3}{3!}\right) + \dots$$

Solution : For $x = 1$, we get

$$\begin{aligned} e = e^1 &= 1 + \left(\frac{1}{1!}\right) + \left(\frac{1^2}{2!}\right) + \left(\frac{1^3}{3!}\right) + \dots \\ &= 1 + \left(\frac{1}{1!}\right) + \left(\frac{1}{2!}\right) + \left(\frac{1}{3!}\right) + \dots \end{aligned}$$

Example 28 : (of truncation error) Find the approximate value of e by taking first four terms of Maclaurin's series and also, find the truncation error.

Solution : We know that

$$e = 1 + \left(\frac{1}{1!}\right) + \left(\frac{1}{2!}\right) + \left(\frac{1}{3!}\right) + \dots$$

Therefore approximation of the value of e by first four terms is $1 + 1 + 1/2 + 1/6 = 2.666$ (4 significant digits).

The truncation error is $\left(\frac{1}{4!}\right) + \left(\frac{1}{5!}\right) + \left(\frac{1}{6!}\right) + \dots$

Ex. 20 : Find the Taylor series for x^{-1} at $a = 1$.

Ex. 21 : Approximate the value of $(1.5)^{-1}$, using first three terms of Taylor's series expansion.

Ex. 22 : Find the Taylor series for $\log(x)$ at $a = 1$.

1.7 SUMMARY

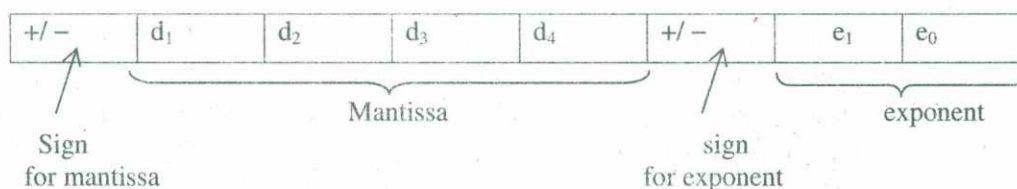
Section 1.2

In the *fixed point decimal representation*, the decimal point forms a part of the string representing the number, and its location within the string determines the relative size of the number. The number of numbers that can be represented with fixed point representation is very limited.

The scientific notation, explained below provides an efficient solution to the problem. In scientific notation all numbers are written in the form of $m \times 10^e$, m is multiplied by ten raised to the power of e , where the exponent e is an integer, and the coefficient m is a real number, which is generally a fraction, and is called the *significant* or *mantissa*. If the number is negative then a minus sign precedes a . The exponent is also called characteristic.

Section 1.3

Floating point *decimal* number systems are adaptation of *scientific* notation for computer representation of numbers. WE USE ONLY THE FOLLOWING FORMAT FOR FLOATING POINT REPRESENTATION



where each of $d_1, d_2, d_3, d_4, e_1, e_0$ is a decimal digit. Further, *mantissa* part is interpreted as a decimal *fraction*, viz., the above one as $.d_1d_2d_3d_4$ and *exponent* part as a decimal whole number, viz., e_1e_0 .

Further, a floating point representation is said to be in *Normalized form*, if the mantissa m satisfies $0.1 \leq m < 1$.

Advantage of Normalized form of floating number representation is that we can have more numbers that can be represented exactly, instead of just approximately.

One of the major problems in floating representation is that there may be very large number of representations for the number 0. In our representation, zero has 400 possible representations in our format. This fact may create serious problems. The best remedy is to replace each representation of zero, whenever it arises, by the representation in which exponent is the least possible, in our case, -99 .

However, even for floating point representation also, the number of floating points is only 4×10^6 (there are 6 digital positions, each of which can take 10 values and two positions for sign, each of which can take two values). But the number of real numbers is infinite. Thus not every real number can be exactly represented as floating numbers. For similar reason, even all integers/rational numbers cannot be represented as floating numbers.

Members of each of the four categories mentioned below, cannot be represented exactly in any floating point representation system, and, hence, **necessarily will have error** in representation, even if 100's (but the number is fixed in advance) of digits are used for each of mantissa and for exponent:

- (i) The set of transcendental numbers, like π and e , even when the base is not 10 or 2.
- (ii) The set of irrational number, like $\sqrt{2}$, **because any transcendental number and any irrational number** cannot be represented exactly as a finite string of decimal/binary digits and, hence, as a floating number, even when the base is not 10 or 2.
- (iii) **The set of those rational numbers, like $1/3 = 0.33333\dots$, having repeating, but non-terminating decimal representation**, cannot be represented exactly as a finite decimal/binary string, and hence, cannot be represented as a floating number.
- (iv) Also, even some rational numbers which can be represented as a finite string of **decimal digits**, may not be written as a finite string of **binary digits**. For example, $1/5$ can be written as 0.2 , a finite decimal string, but can be written only as an infinite *binary* string : $0.00110011\dots$

In all the four types of number representation, as a member of the type cannot be represented exactly as a floating number, it is generally approximated to some near-by floating number, through either of the two processes called : (i) **rounding** (ii) **chopping**.

- (i) **Chopping** applies the rule: approximate n by the floating number $.d_1d_2d_3d_4 \times 10^k$, i.e., leave out all the digits d_5, d_6 etc. Chopping introduces error, called chopping error in the representation of the given number. We can show that the **chopping error** $\leq 10^{k-4}$... (A)
- (ii) **Rounding** (also, called symmetric rounding), applies the rule : If $d_5 \leq 4$, then n is approximated by the floating number $.d_1d_2d_3d_4 \times 10^k$ (as in chopping)..... and (otherwise, i.e.,)

If $d_5 \geq 5$, then n is approximated by the floating number $(.d_1d_2d_3d_4 + 0.0001) \times 10^k$

Rounding introduces error, called *Rounding error* or *Round-off* in the representation of the given number.

We can show that the absolute value of (The Round-off error) $\leq 0.5 \times 10^{-4}$.

There are two popular types of errors :

Absolute Error (Definition) : Let V_a denote the actual value and V_r the represented value of V_a . Then absolute error in V_a = absolute value of $((V_a - V_r))$

= absolute value of (actual value – represented value)

Relative Error (Definition) : We know the phrase ‘an error of 5 in an actual value of 100’ has quite different meaning from the phrase ‘an error of 5 in an actual value of 100000’. In view of such differences in meaning of error-value, the concept of relative error is introduced as follows : Then relative error in V_a = absolute value of $((V_a - V_r)/V_a)$

= absolute value of ((actual value – represented value)/actual value)

= absolute error/ actual value l.

The arithmetic operations on floating numbers may further propagate errors in the required results. Some more types of errors :

Underflow : The error, due to the fact that the result cannot be stored, because the resultant number is too small, having its exponent less than least exponent that can be stored, is called *UNDERFLOW*.

Overflow : The error, due to the fact that the result cannot be stored *even approximately*, because the resultant number is too large, as its exponent is more than largest exponent that can be stored, is called *OVERFLOW*. **Loss of Significant Digits (Definition):** In a calculated result, using normalized floating numbers, if, mantissa has less than 4 digits (*in general, less than allotted number for digits in the mantissa*), then the remaining digit spaces, on the right hand in the space for the mantissa, are filled with 0's. But these additional 0's do not carry any significance/value/meaning. These 0's have been forced on us by, the otherwise very useful, format. Introduction of these 0's is called loss of significant digits.

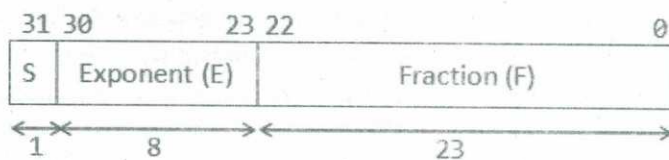
Some more problems due to arithmetic operations :

- (i) **Non-associativity of ‘+’ for Floating Numbers.**
- (ii) **Problems in Floating number multiplication and division – overflow, underflow,**
- (iii) **Non-distributivity of ‘x’ over ‘+’ :** To show, for some floating numbers a, b and c , we can have $a \times (b + c) \neq a \times b + a \times c$.

Section 1.4 : Floating-Point binary Representation schemes are obtained by modification of the floating point *decimal notation* by restricting the digits to be used to only 0 and 1 and by change of the base from 10 to 2.

For representing floating numbers, IEEE Standard 754 floating point representation is the most common. In IEEE Standard 754, 32-bit single-precision floating-point representation :

- The most significant bit (with index 31) is the *sign bit (S)*, with 1 for negative numbers and 0 for positive numbers.
- The following 8 bits represent *exponent (E)*.
- The remaining 23 bits represents *fraction (F)*.



32-bit Single-Precision Floating-point Number

The Mantissa in IEEE standard 754, is interpreted slightly differently. The sequence of 23 bits, say, $a_{22} a_{21} a_{20} \dots a_2 a_1 a_0$ is interpreted as the binary number $1 . a_{22} a_{21} a_{20} \dots a_2 a_1 a_0$ or the binary number, which is the result of $(1 + . a_{22} a_{21} a_{20} \dots a_2 a_1 a_0)$. This 1 is (implicitly) assumed without its explicit inclusion in the representation.

Summary of single precision (32-bit) IEEE standard 754 is :

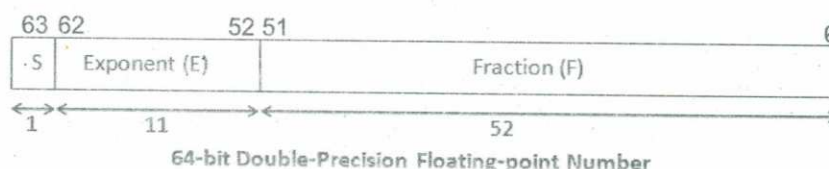
	Sign bit	Exponent	Fraction	Bias
Single Precision	1 [31]	8 [30-23]	23 [22-00]	127

Range of real numbers represent-able in 32-bit normalized format is

$$\pm 2^{-126} \text{ to } (2-2^{-23}) \times 2^{127} \quad \pm \sim 10^{-44.85} \text{ to } \sim 10^{38.53}$$

The representation scheme for 64-bit double-precision is similar to the 32-bit single-precision :

- The most significant bit is the *sign bit* (S), with 0 for negative numbers and 1 for positive numbers.
- The following 11 bits represent *exponent* (E).
- The remaining 52 bits represents *fraction* (F).



Summary of double precision (64-bit) IEEE standard 754 is :

	Sign bit	Exponent	Fraction	Bias
Double Precision	1 [63]	11 [62-52]	52 [51-00]	1023

Range of real numbers represent-able in 64-bit normalized format is

$$\pm 2^{-1022} \text{ to } (2-2^{-52}) \times 2^{1023} \quad \pm \sim 10^{-323.3} \text{ to } \sim 10^{308.3}$$

Section 1.5 : In this section, we discuss the following two quite similar, but distinct, concepts, viz.,

- Unstable Algorithm
- Ill-conditioned/Unstable Problem

Definition : One of the characteristics of *Unstable Algorithm* is that the unstable algorithm does not give good result, but there is an alternative algorithm, which gives good result.

Subtractive Cancellation : The process that involves subtraction of nearly equal numbers which, at the same time, leads to large relative deviation from the correct result (including loss of significant digits), is called subtractive cancellation.

However, in some cases, as discussed below, there cannot be an alternative algorithm to remedy the underlying difficulty in solving properly the problem under consideration.

Example

Consider the system of two linear equations : $x + 2y = 3$, and $0.499x + 1.001y = 1.5$, whose solution is $x = 1$ and $y = 1$.

Next consider the set of equations : $x + 2y = 3$, and $0.5x + 1.001y = 1.5$, obtained by changing coefficient of x from 0.499 to 0.5 in second equation. The unique solution of this set of equations is $x = 3$ and $y = 0$. Thus, the small change of coefficient from 0.499 to 0.5 has caused the solution to change drastically.

Such problems are called ill-conditioned. Formally, a problem is said to be **ill-conditioned**, if slight change in the input data leads to substantial change in the result. In other words, a problem is said to be ill-conditioned in which the solution is very sensitive to changes in the data. Some authors use the term **Unstable Problem**, (different from **Unstable Algorithm/Solution**) in place of ill-conditioned problems.

A problem is said to be **well-conditioned or Stable Problem**, if small changes in the input data always lead to small changes in the results.

Section 1.6

In mathematics, a **Taylor series** is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point. If the Taylor series is centered at zero, then that series is also called a **Maclaurin series**. Any finite number of initial terms of the Taylor series of a function is called a **Taylor polynomial**.

In view of the fact that a computer can calculate the sum of *only a finite number of terms* of a Taylor's series of a function, Taylor polynomials are used to approximate a function. As mentioned earlier also, the process of approximation of an infinite series/process by a finite sub-series/appropriate finite process is called **Truncation**. Thus, an approximation of a function by Taylor's *polynomial* from the Taylor's *series* expansion of the function is also truncation. As truncation is achieved from a finite part of an infinite entity, it gives rise to an error called **Truncation Error**.

1.8 SOLUTIONS/ANSWERS

Ex. 1 : (i) + 2345 + 02, (ii) - 4580 - 02, (iii) - 8970 + 7, (iv) many representations possible, none normal, one is + 0000 - 99, and (v) + 7854 - 01.

Ex. 2 : (i) exactly same as in Ex. 1 above (ii) - 4587 - 02, (iii) - 8970 + 7, (iv) many representations possible, none normal, one is + 0000 - 99, and (v) + 7854 - 01.

Error, if any (i) None (ii) 6×10^{-7} , (iii) 565, (iv) none, and (v) 32×10^{-7} .

Ex. 3 : (i) exactly same as in Ex. 1 above (ii) - 45878 - 02, (iii) - 8971 + 7, (iv) many representations possible, none normal, one is + 0000 - 99, and (v) + 7854 - 01.

Error, if any (i) None (ii) 4×10^{-7} , (iii) 435, (iv) none, and (v) 32×10^{-7} .

Ex. 4 : First, we make the exponents equal, by re-writing the number with smaller exponent in un-normalized form : $y_2 = -0.03400 \times 10^3$, so that the exponents of the numbers equal. The numbers be then written so that decimal points align

$$x_1 = 0.4273 \times 10^3,$$

$$y_2 = -0.03400 \times 10^3,$$

Then, adding the fractions, taking into consideration the sign, we get

$$x_1 + y_2 = +0.39330 \times 10^3$$

Finally, normalizing the result, we get $x_1 + y_2 = +0.3933 \times 10^3$ in the normalized form. Also, there is no error in the result.

Ex. 5 : First of all, we make the exponents equal, by re-writing the number with smaller exponent in un-normalized form : $x_2 = 0.008370 \times 10^1$, so that the exponents of the numbers equal. The numbers be then written so that decimal points align

$$x_1 = 0.2473 \times 10^1,$$

$$x_2 = 0.008370 \times 10^1,$$

Adding the fractions, we get the un-normalized form $x = x_1 + x_2 = 0.255670 \times 10^1$.

However, in order to write x in the following format,

+/-	d_1	d_2	d_3	d_4	+/-	e_1	e_0
-----	-------	-------	-------	-------	-----	-------	-------

we have to use chopping/rounding. In this case,

Rounding gives the result $x \approx 0.2557 \times 10^1$, and

Chopping gives the result $x \approx 0.2556 \times 10^1$.

However, the result when represented as a floating number, through rounding or chopping, **has an absolute error** :

Mod of $(0.255670 \times 10^1 - 0.2557 \times 10^1) = 0.00003 \times 10^1 = 3 \times 10^{1-5} = 3 \times 10^{-4}$, when rounding is used,

$0.255670 \times 10^1 - 0.2556 \times 10^1 = 0.00007 \times 10^1 = 7 \times 10^{1-5} = 7 \times 10^{-4}$, when chopping is used.

Ex. 6 : As the actual value V_a of $x_1 + x_2$ is 2.55670.

The relative error in $V_a = (3 \times 10^{-4})/2.55670$, when rounding is used

The relative error in $V_a = (7 \times 10^{-4})/2.55670$, when chopping is used

Ex. 7 : Applying Algorithm 2, during Step 3, we get result

$$x = x_1 + x_2 = 0.0055 \times 10^{-99} = 0.5500 \times 10^{-101}$$

But the exponent -101 of the result is less than the least exponent -99 that can be stored in our standard format

+/-	d_1	d_2	d_3	d_4	+/-	e_1	e_0
-----	-------	-------	-------	-------	-----	-------	-------

The result cannot be stored in our format. It is a case of underflow.

Ex. 8 : For the values of a , b and c ; $(a + b) + c \neq a + (b + c)$, in floating point arithmetic using rounding, as shown below : For $a + b = 0.2134 \times 10^5 + 0.2354 \times 10^3$, re-write b as 0.002354×10^5 (the un-normalized form), so that

$$\begin{aligned} a + b &= 0.2134 \times 10^5 + 0.002354 \times 10^5 \text{ (we get un-normalized form)} \\ &= 0.215754 \times 10^5 \end{aligned}$$

Normalizing using rounding, we get $a + b = 0.2158 \times 10^5$.

Then to get $(a + b) + c = 0.2158 \times 10^5 + (-0.2142 \times 10^1)$, we re-write c in the un-normalized form as $(-0.00002142 \times 10^5)$. Aligning the decimal points we get $(a + b) + c = 0.2158 \times 10^5 + (-0.00002142 \times 10^5) = 0.2158 \times 10^5$.

Next, similarly, we calculate $(b + c) = 0.2333 \times 10^3$.

$$\begin{aligned} \text{Next, similarly, we calculate } a + (b + c) &= 0.2134 \times 10^5 + 0.2333 \times 10^3 \\ &= 0.2134 \times 10^5 + 0.002333 \times 10^5 = 0.2157 \times 10^5 \end{aligned}$$

Hence $(a + b) + c \neq a + (b + c)$.

Ex. 9 : As $a \neq 0 \neq b$, $m \leftarrow (-0.4031) \times (-0.1101) = +0.04438131$

(un-normalized, using 8-digit registers)

$$e \leftarrow (3 + (-4)) = -1$$

On normalizing m and then using, say, rounding, we get

$$m \leftarrow +0.4438131 \text{ (still un-normalized) and } e \leftarrow -1 + 1.$$

The required number in floating form is $c = +0.4438 \times 10^0$.

Ex. 10 : As $a \neq 0 \neq b$, therefore

$$m \leftarrow (-0.3326) \div (0.1101) = -3.02089 \dots \text{ (un-normalized 8 digit register)}$$

$$\text{and } e \leftarrow (-4) - 3 = -7$$

$$\text{as } 1 < m$$

$$m \leftarrow -0.302089 \dots \text{ and } e \leftarrow -7 + 1 = -6$$

Normalizing, we get $m = 0.3021$. Therefore, $c = -0.3021 \times 10^{-6}$.

Ex. 11 : Let $c = a \times b = m \times 10^e$, and m is in normalized floating form, then

If $0.1 \leq |m|$, then $m = m_1 \times m_2$ $e \leftarrow e_1 + e_2 = -53 + (-49) = -103$

Else $m = m_1 \times m_2 \times 10$ and $e \leftarrow e_1 + e_2 - 1 = -53 + (-49) - 1 = -104$

In both cases e cannot be stored in the 2 decimal digit space allotted to e or, alternatively, we can show the underflow through the following argument.

[through explicit calculations, we have : As $0.1 < |0.2003 \times (-0.5200)| < 1$ therefore $m = 0.2003 \times (-0.5200)$ and $e = -(53 + 49) = -103 \leq -99$ cannot be stored in the 2 decimal digit space allotted to e].

Ex. 12 : If $c = a \div b = m \times 10^e$, and m is in normalized floating form, then

If $1 \leq |m|$ then $m = m_1 \div m_2$ $e \leftarrow e_1 - e_2 + 1 = -53 - (49) + 1 = -101$

Else $m = m_1 \div m_2$ $e \leftarrow 53 - (-49) = -102$

In both cases e cannot be stored in the 2 decimal digit space allotted to e or, alternatively, we can show the underflow through the following argument.

[through explicit calculations, we have : As $0.1 < |0.2003 \div (-0.5200)| = 0.385 \dots < 1$ therefore $m = 0.2003 \times (-0.5200)$ and $e = -53 - (49) = -103 \leq -99$, cannot be stored in the 2 decimal digit space allotted to e].

Ex. 13 : There is an underflow, According to our rule/strategy, the result of $a \times b$ in this case is assigned 0 and is represented as 0.0000×10^{-99} .

Therefore, $(a \times b) \times c = (0000 \times 10^{-99}) \times c = 0000 \times 10^{-99}$, representing zero.

On the other hand $(b \times c) = 0.1000 \times 10^{-30+(50)-1} = 0.1000 \times 10^{19}$

Hence, $a \times (b \times c) = 0.1000 \times 10^{-80+19-1} = 0.1000 \times 10^{-62}$

Thus, $(a \times b) \times c \neq a \times (b \times c)$

Ex. 14 : Then $b + c = 0.0020 \times 10^2 = 0.2000 \times 10^0 = 0.2$.

Therefore, $a \times (b + c) = (0.2706 \times 10^1) \times 0.2 = 0.5412$... (A)

Next, we consider

$$a \times b = 0.2706 \times 10^1 \times (-0.7445 \times 10^2) = -0.20092050 \times 10^3$$

(un-normalized, using temporarily 8-digits in mantissa)

$$= -0.2009 \times 10^3$$

(normalized, after rounding)

$$a \times c = 0.2706 \times 10^1 \times 0.7445 \times 10^2 = 0.20146170 \times 10^3$$

(un-normalized, using temporarily 8-digits in mantissa)

$$= 0.2015 \times 10^3$$

(normalized, after rounding)

Therefore, $a \times b + a \times c = -0.2009 \times 10^3 + 0.2015 \times 10^3 = 0.0006 \times 10^3 = 0.6$... (B)

Comparing (A) and (B), we get the required result

$$a \times (b + c) \neq a \times b + a \times c$$

Ex. 15 : Sign bit $S = 0 \Rightarrow$ positive number

$$E = 0111\ 1110 \text{ (binary)} = 126 \text{ DECIMAL (in normalized form)}$$

Fraction is 1.100 0000 0000 0000 0000 0001 (Binary) (with an implicit leading 1)
 $= 1 + 2^{-1} + 2^{-23}$

The number is $(1 + 2^{-1} + 2^{-23}) \times 2^{(126-127)} = 0.7500000059604644775390625$ (may not be exact in decimal!).

Ex. 16 : Sign bit $S_s = 0 \Rightarrow$ positive number

$$E = 0 \text{ (in de-normalized form)}$$

Fraction is 0.000 0000 0000 0000 0000 0011 (binary) (with an implicit leading 1)
 $= 1 + 1 \times 2^{-22} + 1 \times 2^{-23}$

The number is $(1 + 1 \times 2^{-22} + 1 \times 2^{-23}) \times 2^{(-126)} = (1 + 3 \times 2^{-23}) \times 2^{(-126)}$

Ex. 17 : The above system of equations has a *unique* solution :

$$x = 1.000 \text{ and } y = 0.000$$

Next, we consider another system of linear equations, obtained by making very small change in one of the coefficients, viz., coefficient of y is changed from 0.6667 to 0.6666 to get :

$$(2.000) x + (0.6666) y = 2.000$$

$$(1.000) x + (0.3333) y = 1.000$$

This new system of equations, surprisingly, has the *infinitely* many solutions:

$$x = 1.000 - (0.3333) k \text{ and } y = k$$

where k may be any real number. Therefore, ill-conditioned.

Ex. 18 : By integrating the following Maclaurin series for $f(x) = (1-x)^{-1}$:

$$1 + x + x^2 + x^3 + \dots$$

We find the Maclaurin series for $\log(1-x)$ is

$$-x - \frac{1}{2}x^2 - \frac{1}{3}x^3 - \frac{1}{4}x^4 - \dots \text{ for } -1 \leq x < 1.$$

Ex. 19 : (i) for approximating $\log(2)$

In view of the fact that $\log(1-x) = -\int (1-x)^{-1}$

From the Maclaurin series for $\log(1-x)$,

$$-x - \frac{1}{2}x^2 - \frac{1}{3}x^3 - \frac{1}{4}x^4 - \dots \text{ for } -1 \leq x < 1.$$

Taking $x = -1$, and first 4 terms, we get

$$\log(2) \approx -(-1) - (-1)^2/2 - (-1)^3/3 - (-1)^4/4 = 1 - 1/2 + 1/3 - 1/4 = 7/12 = 0.5833 \dots$$

Better approximation of $\log(2) \approx 0.693147 \dots$

$$\text{The truncation error} = (1/5 - 1/6) + (1/7 - 1/8) + (1/9 - 1/10) + \dots$$

$$\approx (0.693147 \dots) - (0.5833 \dots) \approx 0.1098 \dots$$

(ii) for approximating $\log(1)$, by taking 4 terms, Take $x = 0$, we get

$\log(1) = 0$, which is the correct value.

Ex. 20 : As $f^{(n)}(x) = d^n(x^{-1})/dx^n = (-1)^n n!/(x)^{-(n+1)}$

Using $a = 1$ and the value of the n^{th} derivative $f^{(n)}(1)$ at $x = a = 1$ which is $(-1)^n n!$ in (A),

We get, the Taylor's series expansion of x^{-1} around $x = 1$

$$1 - (x-1) + (x-1)^2 - (x-1)^3 + \dots$$

Ex. 21 : Approximate the value of $(1.5)^{-1}$, using first three terms of Taylor's series expansion. From the above example, x^{-1} , in the neighbourhood of $x = 1$ is given by

$$1 - (x-1) + (x-1)^2 - (x-1)^3 + (x-1)^4 - \dots$$

Replacing x by 1.5, and taking first three terms, we get

$$1 - (0.5) + (0.5)^2 - (0.5)^3 = 0.625$$

The actual value of $1/1.5$ is 0.66666...

The truncation error is $(0.5)^4 - (0.5)^5 + \dots = 0.04166 \dots$

Ex. 22 : The Taylor's series expansion for x^{-1} at $a = 1$ is

$$0 + (1 - (x-1) + (x-1)^2 - (x-1)^3 + \dots)$$

(0 is added, because its integral is any constant including -1).

Integrating both sides, we get the corresponding Taylor series for $\log(x)$ at $a = 1$ is

$$(x-1) - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 - \frac{1}{4}(x-1)^4 + \dots$$

UNIT 2 SOLUTION OF LINEAR ALGEBRAIC EQUATIONS

Structure

- 2.0 Introduction
- 2.1 Objectives
- 2.2 Preliminaries
- 2.3 Direct Methods
 - 2.3.1 Gauss Elimination Method
 - 2.3.2 Row Interchanges/Pivoted Condensation Method
- 2.4 Iterative Methods
 - 2.4.1 Gauss-Jacobi Iterative Method
 - 2.4.2 Gauss-Seidel Iterative Method
 - 2.4.3 Comparison of Direct and Iterative Methods
- 2.5 Summary
- 2.6 Solutions/Answers

(In the first reading, you may go directly to Example 4, and read the other examples after that and may come back and start reading the unit from the beginning)

2.0 INTRODUCTION

In Unit 0 and Unit 1, we discussed many issues including: Why we use numerical techniques? What are numerical techniques like rounding, truncation and discretisation etc.? What are the potential problems arising out of these techniques? However, no part of the discussion in these units can be used to solve any mathematical problem. The discussion was about general issues and nature of the discipline of numerical analysis and method. In this unit, we will discuss methods that can and are used for solving a system of linear equations.

2.1 OBJECTIVES

After going through this unit, you should be able to

- explain and use the concepts of a type of equations, system of equations (and linear equations), equivalence of systems of equations, types of system of equations, like consistent/inconsistent, homogeneous/non-homogeneous etc., solution of a system of equations; some types of matrices;
- explain what is matrix representation of a system of linear equations and how to obtain such a representation;
- explain and apply method of transforming a system of equations into an equivalent one, which is easier to solve;
- explain and apply Gaussian Elimination Method to solve a system of equations;
- explain and apply Gaussian Elimination Method with row interchange/pivotal condensation to solve a system of equations;
- tell how many solutions a system of linear equations, may have, and why;

- explain and apply Gaussian Jordan and Gauss Seidel iterative methods to solve a system of equations; and
- compare the direct and iterative methods for solving a system of equations.

2.2 PRELIMINARIES

We explain some relevant elementary concepts through examples. For example, each of the following *expressions* is an *equation*:

Example 1 : (i) $3x + 5 = 14$, (ii) $5x - 17y = 45$, (iii) $ay - bz + 31 = 49$,
(iv) $ax^2 + 5y - b = c$, (v) $4y x^2 z + a z^3 y - 9 = 0$, (vi) $a e^z - bz + 31 = 49$,
(vii) $y^{-3}x + zx^2 + d = 42$, (viii) $27 \log(x) + ax^2 + 5y - b = c$, (ix) $(3x + 5)/(5x - 15y) = 45$;
in which a, b, c and d etc., are assumed to be constants and x, y and z etc. as variables.

A *constant* is a quantity/entity, like 5.7 or the word 'cow', which is already known; or, it is an unknown quantity/entity, like, say, a or b etc. which is, of course, unknown, *but is to be read as part of input data, before some processing takes place*. On the other hand, a *variable* is an unknown quantity/entity, the value which *is to be determined by an algorithm/procedure* or, is used in the algorithm/procedure; and the value so determined or used may be given as *output* of the procedure.

Each of the following *expression* is *not* an *equation*, *because*, it does not involve the sign of equality (*viz.*, $=$):

Example 2 : (i) $3x + 5$, (ii) $a e^z - bz + 31$, and (iii) $y^{-3}x + zx^2 + d - 42$.

We frequently use the concept of *term* in an equation. Each part of any of the equations, given above, in which the arithmetic signs of $+$ and $-$ (and, of course, the sign $=$) are *not* involved, is called a term of the equation. For example, the equation (i) $3x + 5 = 14$, has three terms, namely, $3x$, 5 and 14 . The equation (vi) $a e^z - bz + 31 = 49$ has 4 terms, namely, $a e^z$, bz , 31 and 49 . *However, '-' may be involved as a unary operation as in each of $-c$, -56 , $-x^2yz$ etc., but not as a binary operation as in $5x - ay$ etc.*

Equations may be further *classified as*, *polynomial, linear, non-polynomial, non-linear* etc.

In *Example 1*, each of the equations (i) to (v) is a *polynomial* equation. Each of Equations (vi) to (viii) is *not* a polynomial equation, because, in (i) to (v), the powers of the variables x, y, z etc., in each are *only positive integers*. But in each of the equations (vi) to (viii), in addition, some other functions of the variables like $\log$, e (exponential), and negative integers are also involved, and, hence, equations (vi) to (viii) are *not* polynomial equations.

Equation (ix) $(3x + 5)/(5x - 15y) = 45$ of *Example 1*, is a bit different, in the sense that, in the given form, it is not polynomial equation; but, it is *conditional* polynomial equation, under the condition on x and y that $(5x - 15y) \neq 0$. Because then (ix) can be re-written as $(3x + 5) = 45(5x - 15y)$, which is a polynomial equation.

Further, out of polynomial equations (i) to (v), each of the equations (i) to (iii) is *linear equation*, because, power of each of the terms is at most 1 (and 0 also, because, a constant, say b can be thought as bx^0 , or bz^0 etc.). The polynomial equation (iv) $ax^2 + 5y - b = c$, is not linear, but *quadratic*, because, the maximum power of a term is 2. Also, (v) $4y x^2 z + a z^3 y - 9 = 0$, is *not* a linear, not even a quadratic equation, but is a *bi-quadratic* equation, because, sum of the powers in at least one term, namely, $4y x^2 z = 4y^1 x^2 z^1$ is 4.

Next, we consider the concept of *(simultaneous) system of equations*. For example, if we have two equations, *viz.*, $3x + 4y = 5$ and $8x - 3y = -14$. If, the purpose is to determine a value of x and a value of y so that by substituting the value of x and the value of y , both the equations are satisfied, then *the two equations form a single system (in this case) of two equations*. When we speak of a system of equations, generally, we mean two or more

than two equations; though even a single equation also may be considered as a system of equations. Also, the involved equations in a system may be linear, polynomial or neither. However, in this unit, we consider systems of only linear equations. Before we proceed to describe methods of solving a system of linear equations, we define some concepts and make some observations, which will be useful throughout the discussion of the topic.

Definition : A **solution** to a set of equations, consists of a set of values assigned to variables, one for each variable, occurring in the equations, say a to variable x , b to the variable y ,... etc., s. t. all the equations are satisfied simultaneously, when each of the assigned values is substituted for the respective variable. For example, $x = 2$ and $y = 3$, together, constitute a solution for the set of equations in the following definition.

Definition : Two systems of linear equations are said to be **equivalent**, if they have same solution/solutions. For example, one system of equations, in two variables, is

$$2x + 5y = 19$$

$$6x + 11y = 45$$

And another system of equations, in two variables, is

$$x + 3y = 11 \quad \text{and}$$

$$3x + 2y = 12$$

Then the two systems of equations are equivalent, because, for each of the two systems, $x = 2$, and $y = 3$ is the only solution.

More Definitions : A matrix A in which all the off-diagonal elements are zero, i.e. $a_{ij} = 0$

for $i \neq j$ is called a **diagonal matrix**; e.g., $A = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$ is a 3×3 diagonal matrix.

A matrix U is said to be **upper triangular** if all its elements below the diagonal are zero, e.g.

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

Similarly, a matrix L is said to be **lower triangular** if all its elements above the diagonal are zero, e.g.

$$L = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}$$

Definition : A system of linear equations is said to be **consistent**, if there exists a solution of the system of equations, and, a system is said to be **inconsistent** if no solution exists.

Next, we make some useful observations.

Observation (I): is discussed through the following example.

Example 3 : If we are given a system of equations, say, the following one of 2 linear equations :

$$2x + 5y = 19 \quad \dots (\text{Eq. 1})$$

$$6x + 11y = 45 \quad \dots (\text{Eq. 2})$$

then for any real number $k \neq 0$, if we obtain a new equation say, Eq. 3 such that the L.H.S. of Eq. 3 = (L.H.S. of Eq. 2) - $k \times$ (L.H.S. of Eq. 1), and the (R.H.S. of Eq. 3) = (R.H.S. of Eq. 2) - $k \times$ (R.H.S. of Eq. 1), then the system of equations having Eq. 1 and Eq. 3, is equivalent to the earlier system consisting of Eq. 1 and Eq. 2. In other words, the system of equations having equations Eq. 1 and Eq. 2 has same solution as system of equations Eq. 1 and Eq. 3. (Similarly, interchanging the roles of Eq. 1 and Eq. 2, the above observation is again valid).

For example, if we take $k = 3$, then we get Eq. 3 as

$$6x + 11y - 3 \times (2x + 5y) = 45 - 3 \times 19$$

$$\text{i.e.,} \quad 11y - 15y = -12 \quad \dots (\text{Eq. 3})$$

$$\text{i.e.,} \quad -4y = -12 \quad \dots (\text{Eq. 3})$$

In view of the above observation, instead of system of equations Eq. 1 and Eq. 2, we may solve system of equations Eq. 1 and Eq. 3 or we may solve system of equations Eq. 2 and Eq. 3. From the above example, we can see the advantage that, in this case, from Eq. 3, we get the value of one of the variables, namely, $y = 3$. Using $y = 3$, in either Eq. 1 or Eq. 2, we get the value of the other variable. Using $y = 3$ in either Eq. 1 or Eq. 2, we get $x = 2$.

Observation (II) : A system of linear equations can be equivalently written as an equation of matrices.

Let us consider the system of equations

$$\left. \begin{array}{l} 3x + 5y = 11 \\ 7x + 3y = 17 \end{array} \right\} \quad \dots (\text{A})$$

By the rules of multiplication of two matrices, we can write L.H.S of the given equations, i.e.,

$$\begin{aligned} & \left. \begin{array}{l} 3x + 5y \\ 7x + 3y \end{array} \right\} \\ &= \begin{bmatrix} 3 & 5 \\ 7 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \end{aligned}$$

Hence, by the definition of equality of two matrices, the equations in (A) can be re-written in the matrix form as

$$\begin{bmatrix} 3 & 5 \\ 7 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 11 \\ 17 \end{bmatrix}$$

If we write $A = \begin{bmatrix} 3 & 5 \\ 7 & 3 \end{bmatrix}$ and $\underline{X} = [x \ y]$ and $\underline{b} = [11 \ 17]$ then the given equation can be written in the $A \underline{X}^T = \underline{b}^T$.

More generally, a system of (say) 4 linear equations,

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3$$

$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4$$

can be written in matrix form as

$$A \underline{X} = \underline{b}$$

where the coefficient matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

and $\underline{x}^T = (x_1 \ x_2 \ x_3 \ x_4)$ and $\underline{b}^T = (b_1 \ b_2 \ b_3 \ b_4)$, where $\underline{x}^T$ etc. denote transpose of matrix $\underline{x}$. The elements of matrix A and that of $\underline{b}$ are known while the solution $\underline{x}$ has to be determined.

Matrix A is called coefficient matrix

Definition : A system of linear equations $A\underline{x} = \underline{b}$ is said to be homogeneous, if $\underline{b} = 0$, i.e. $b_i = 0$ for all i . Otherwise, system is called non-homogeneous.

The advantage of matrix notation is that we can view all the equations as a single entity – a matrix equation, and many available matrix operations can be used to operate on all the equations simultaneously.

Observation (III) In respect of system of 2 equations in 2 variables (and in general, n equations in n variables), regarding number of solutions, there are three and only three of the following possibilities:

- (i) The system of equations has one solution, and that is unique. Example 3 above, under Observation (1), involves a system of equations, has one solution, and it can be easily verified that the solution is unique.

In this context, we state the following useful result on the unique solvability of a system of linear equations.

Theorem : A non-homogeneous system $A\underline{x} = \underline{b}$ (i.e., $\underline{b} \neq 0$) of n linear equations in n unknown has a unique solution if and only if the coefficient matrix A is non singular ($\det A \neq 0$) and then the solution can be expressed as $\underline{x} = A^{-1}\underline{b}$.

- (ii) The system of equations has no solution. The system in this case, is called inconsistent. For example, $3x + 4y = 15$ and $9x + 12y = 35$. Attempting to solve this system gives $35 = 9x + 12y = 3(3x + 4y) = 3(15)$ (from the first equation) $= 45$.

Thus, the system of equations gives $35 = 45$. The possibility arises when L.H.S. of one of the equations (say, first), is some $k \neq 0$ of the L.H.S. of the other, but, R.H.S. of first is not k times of the R.H.S. of the second.

- (iii) The system of equations has infinite number of solutions. An example of such a possibility is obtained by slightly modifying system of equations in (ii) above by replacing 35 by 45 in second equation. The modified system of equations is : $3x + 4y = 15$ and $9x + 12y = 45$. Attempting to solve this system as we did in (ii), we get $45 = 45$, which is no additional information. The problem is that the second equation is essentially the same as the first equation, written in a different form. Thus, in essence, in this case, we have only one equation involving two variables.

We may conclude from the three (and only three) possibilities that a system of equations, if has two or more solutions, then it must have infinite solutions.

Observation (IV) : The discussion in Observation (III) above is from mathematical consideration, i.e., from the point of view of perfect calculations. However, we have discussed in earlier units that numerical computations are not perfect, e.g. a near-zero number may be represented by zero etc.

The problems of unstable algorithm and ill-conditioned problems as discussed in Section 1.5, are generic, which may arise in any domain of application of numerical methods. Hence these difficulties may also arise in attempting to solve system of linear equations.

2.3 DIRECT METHODS

Under Observation (II) above, we mentioned that a system of linear equations, may be equivalently considered as a matrix equation $A \underline{x} = \underline{b}$, where matrices A and $\underline{b}$ are given as input data, and $\underline{x}$ is to be determined. Assuming, given system of linear equations is a non-homogeneous (i.e., $\underline{b} \neq 0$) system of n linear equations in n unknowns and the coefficient matrix A is non singular ($\det A \neq 0$), then, we can express $\underline{x}$ as,

$$\underline{x} = A^{-1} \underline{b}$$

But to compute A^{-1} by conventional method using adjoint A is most uneconomical computationally. The other method, known as Cramer's rule expresses the solution as,

$$x_j = \frac{|A_j|}{|A|}, \quad j = 1(1)n,$$

where $1(1)n$ means the initial value of j is the integer 1, and then the value of j , for subsequent computation, is incremented by 1. Further, final/last value of j is to be n , and where $|A|$ denotes determinant of matrix A and $|A_j|$ denotes the determinant of the matrix when j th column of A is replaced by the right side vector $\underline{b}$. This again is very uneconomical computation-wise since it also involves evaluation of determinants. Here we shall discuss a few methods which are dependent on elementary row operations on a matrix, like changing of rows, multiplying a row by some factor and adding to another row etc. In the discussion that follows, we use the concept of upper triangle matrix (and, even sometime, lower diagonal matrix).

We start our discussion with the first of these methods, called Gaussian Elimination Method (Basic).

2.3.1 Gauss Elimination Method

→ (if you feel that the discussion of the general method that immediately follows, is too abstract and difficult to understand, you may read Example 4 first, and then come back to the general case discussed below)

Gaussian Elimination Method (Basic) : We shall illustrate the method by taking a system of three equations,

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

Since we shall be required to make elementary operations on coefficients let us write the above system in the form of an augmented matrix as follows :

$$[A/b] = \begin{array}{cccc|c} x_1 & x_2 & x_3 & & \underline{b} \\ a_{11} & a_{12} & a_{13} & | & b_1 \\ a_{21} & a_{22} & a_{23} & | & b_2 \\ a_{31} & a_{32} & a_{33} & | & b_3 \end{array}$$

Let us choose a multiplier $-\frac{a_{21}}{a_{11}}$; multiply first row by it and add to the second. If a_{11}

happens to be zero or very small then change the first row by a row of which first element is not zero or very small.

Example 4 : Solve the following linear system of equations

$$x_1 + x_2 + x_3 = 3,$$

$$4x_1 + 3x_2 + 4x_3 = 8,$$

$$9x_1 + 3x_2 + 4x_3 = 7$$

using the Gauss elimination method.

Solution : In augmented form, we write the system as

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 4 & 3 & 4 & 8 \\ 9 & 3 & 4 & 7 \end{array} \right]$$

Subtracting 4 times the first row from the second row gives

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 0 & -1 & 0 & -4 \\ 9 & 3 & 4 & 7 \end{array} \right]$$

Subtracting 9 times the first row from the third row, we get

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 0 & -1 & 0 & -4 \\ 0 & -6 & -5 & -20 \end{array} \right]$$

Subtracting 6 times the second row from the third row gives

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 0 & -1 & 0 & -4 \\ 0 & 0 & -5 & 4 \end{array} \right]$$

Restoring the transformed matrix equation gives

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 0 & -1 & 0 & -4 \\ 0 & 0 & -5 & 4 \end{array} \right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \\ 4 \end{bmatrix}$$

Solving the last equation, we get $x_3 = \frac{-4}{5}$. Solving the second equation, we get $x_2 = 4$

and the first equation gives $x_1 = 3 - x_2 - x_3 = 3 - 4 + \frac{4}{5} = \frac{-1}{5}$.

Example 5 : Solve the following system of equations by Gaussian elimination method :

$$2x_1 + 5x_2 + x_3 + 8x_4 = 5$$

$$x_1 + 6x_2 + 3x_3 + 5x_4 = 11$$

$$7x_1 + 2x_2 + 6x_3 + 3x_4 = 14$$

$$4x_1 + 8x_2 + x_3 + 2x_4 = 19$$

Compute upto 2 places of decimal only.

Solution

Let us write the above system as follows :

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & b \\ \hline 2 & 5 & 1 & 8 & 5 \\ 1 & 6 & 3 & 5 & 11 \\ 7 & 2 & 6 & 3 & 14 \\ 4 & 8 & 1 & 2 & 19 \end{array}$$

Multiplying 1st row by $-\frac{1}{2} = -0.5$ and adding to the 2nd, i.e.

$$R2 \leftarrow R2 - 0.5 \times R1$$

Next, we perform operation as

$$R3 \leftarrow R3 - \frac{7}{2} \times R1 = R3 - 3.5 \times R1$$

Further, choose factor $-\frac{4}{2} = -2$ and perform operation as

$$R4 \leftarrow R4 - 2 \times R1$$

After above operations we get following system of equations.

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & b \\ \hline 2 & 5 & 1 & 8 & 5 \\ 0 & 3.5 & 2.5 & 1 & 8.5 \\ 0 & -15.5 & 2.5 & -25 & -3.5 \\ 0 & -2 & -1 & -14 & 9 \end{array}$$

We make following operations

$$R3 \leftarrow R3 + \frac{15.5}{3.5} \times R2 = R3 + 4.43 \times R2$$

$$R4 \leftarrow R4 + \frac{2}{3.5} \times R2 = R4 + 0.57 \times R2$$

We get

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & b \\ \hline 2 & 5 & 1 & 8 & 5 \\ 0 & 3.5 & 2.5 & 1 & 8.5 \\ 0 & 0 & 13.58 & -20.57 & 34.16 \\ 0 & 0 & 0.43 & -13.43 & 13.85 \end{array}$$

We do following operation

$$R4 \leftarrow R4 - \frac{0.43}{13.58} \times R3 = R4 - 0.03 \times R3$$

We get

$$\begin{array}{ccccc|c} x_1 & x_2 & x_3 & x_4 & & b \\ \hline 2 & 5 & 1 & 8 & & 5 \\ & & & & & | \\ 0 & 3.5 & 2.5 & 1 & & 8.5 \\ & & & & & | \\ 0 & 0 & 13.58 & -20.57 & & 34.16 \\ & & & & & | \\ 0 & 0 & 0 & -12.8 & & 12.83 \\ & & & & & | \end{array}$$

By back substitution we get

$$x_4 = \frac{12.83}{-12.8} = -1.00 ; \quad x_3 = \frac{(34.16 - 20.57 \times 1.00)}{13.58} = 1.00$$

$$x_2 = \frac{(8.5 - 2.5 \times 1 + 1)}{3.5} = 2.00$$

$$x_1 = \frac{(5 - 5 \times 2 - 1 + 8)}{2} = 1.00$$

Essential idea/steps in Gaussian Elimination Method (Basic) and why the method works? In the above example, in which 4 linear equations in 4 variables are given, the following big/mega steps are taken :

- (i) through the first three row operations, the values in column of x_1 in 2nd, 3rd, and 4th rows, are made 0, so that the coefficient of x_1 in the last three equations becomes 0, and, hence, these three equations do not involve x_1 ;
- (ii) through the next two row operations, the values in column of x_2 in 3rd, and 4th rows, are made 0, so that the coefficient of x_2 in the last two equations becomes 0, and, hence, these two equations do not involve x_1 and x_2 also;
- (iii) finally, through the next one row operation, the values in column of x_3 in 4th row, is made 0, so that the coefficient of x_3 in the last equation becomes 0, and, hence, the last equation do not involve x_1 , x_2 and x_3 also. Thus, last equation, now, involves only x_4 , the value of which can be directly calculated from this equation;
- (iv) then the process of back-substitution is started. The value of x_4 found at step (iii), is substituted back in the earlier three equations;
- (v) However, as third equation, after substitution of value of x_4 , involves only one variable, namely, x_3 , the value of which is now directly calculated;
- (vi) The process of back substitution is repeated until values of all the variables are determined.

In the case of n equations in n variables, the above procedure is suitably modified. For example, in step (i) above, the values in column of x_1 in 2nd, 3rd, ..., n th rows, are made 0, so that the coefficient of x_1 in the last $(n - 1)$ equations becomes 0, and, hence, these $(n - 1)$ equations do not involve x_1 .

Complexity of Gaussian Elimination Method : We may just know the fact that in

Gauss elimination procedure and back substitution $\left[\frac{(2n^3 + 3n^2 - 5n)}{6} \right] + \left(\frac{n^2 + n}{2} \right)$

multiplications/divisions and $\frac{n^3 - n}{3} + \frac{n^2 - n}{2}$ additions/subtractions are performed

respectively. The total arithmetic operations involved in this method of solving an $n \times n$

linear system is $\frac{n^3 + 3n^2 - n}{3}$ multiplication/divisions and $\frac{2n^3 + 3n^2 - 5n}{6}$

additions/subtractions.

It may also be remembered that special cases of system of linear equations may lead to (i) there being no solution for the system and (ii) there being infinitely many solutions.

To discover these cases solve the following exercises. In case of difficulty, refer to the solutions at the end of the unit

Ex. 1 : Solve the following system of equations consisting of four equations.

$$E_1: \quad x_1 + x_2 + 0.x_3 + 3x_4 = 1$$

$$E_2: \quad 2x_1 + x_2 - x_3 + x_4 = -2$$

$$E_3: \quad 3x_1 - x_2 - x_3 + 2x_4 = 0$$

$$E_4: \quad -x_1 + 2x_2 + 3x_3 - x_4 = -2.$$

Ex. 2 : Solve the system of equations

$$3x_1 + 2x_2 + x_3 = 3$$

$$2x_1 + x_2 + x_3 = 0$$

$$6x_1 + 2x_2 + 4x_3 = 6$$

using Gauss elimination method. Does the solution exist? If yes, how many?

Ex. 3 : Solve the system of equations

$$16x_1 + 22x_2 + 4x_3 = -2$$

$$4x_1 - 3x_2 + 2x_3 = 9$$

$$12x_1 + 25x_2 + 2x_3 = -11$$

using Gauss elimination method and comment on the nature of the solution.

Ex. 4 : Use Gauss Elimination to solve the system of linear equations

$$10x_1 - 7x_2 = 7$$

$$-3x_1 + 2.099x_2 + 6x_3 = 3.901$$

$$5x_1 - x_2 + 5x_3 = 6$$

correct to six places of significant digits.

Ex. 5 : Solve the system of equations by Gauss elimination.

$$x_1 - x_2 + 2x_3 - x_4 = -8$$

$$2x_1 - 2x_2 + 3x_3 - 3x_4 = -20$$

$$x_1 + x_2 + x_3 + 0.x_4 = -2$$

$$x_1 - x_2 + 4x_3 + 3x_4 = 4$$

Ex. 6 : Solve the system of equations by Gauss elimination.

$$x_1 + x_2 + x_3 + x_4 = 3$$

$$x_1 + x_2 + 0.x_3 + 2x_4 = 0$$

$$2x_1 + 2x_2 + 3x_3 + 0.x_4 = 10$$

$$-x_1 - x_2 - 2x_3 + 2x_4 = -8$$

2.3.2 Row Interchanges/Pivotal Condensation Method

Gaussian Elimination with Row Interchanges/pivotal condensation method

The Gaussian elimination method with row interchanges is also known as pivotal condensation method. We illustrate the method by the following example.

Example 6

Problem : Solve the following equations by Gaussian elimination method with row Interchanges/pivotal condensation method :

$$3x_2 + 4x_3 = 2$$

$$4x_1 - 2x_2 + x_3 = 18$$

$$3x_1 + 4x_2 + 5x_3 = 11$$

Compute upto two decimals only.

Solution

Let us express the above system in the following form,

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & & \underline{b} \\ 0 & 3 & 4 & | & 2 \\ 4 & -2 & 1 & | & 18 \\ 3 & 4 & 5 & | & 11 \end{array}$$

In the pivotal condensation method, we look for the numerically largest (i.e. irrespective of + or - sign) element in the first column. In this case 4 is the largest element in the second row (and, in column first). We interchange the first row by the second, getting

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & & \underline{b} \\ 4 & -2 & 1 & | & 18 \\ 0 & 3 & 4 & | & 2 \\ 3 & 4 & 5 & | & 11 \end{array}$$

Now perform row operations as before i.e.

$$R2 \leftarrow R2 - \frac{0}{4} \times R1 = R2$$

$$R3 \leftarrow R3 - \frac{3}{4} \times R1 = R3 - 0.75 \times R1$$

We get the following system

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & & \underline{b} \\ 4 & -2 & 1 & | & 18 \\ 0 & 3 & 4 & | & 2 \\ 0 & 5.5 & 4.25 & | & -2.5 \end{array}$$

Now we consider only last two equations, omitting first equation from consideration and look for the numerically largest element in the second column i.e. between 3 and 5.5. Here 5.5 in the third row is largest. Hence we interchange 2nd row by 3rd giving,

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & & b \\ \hline 4 & -2 & 1 & | & 18 \\ & & & | & \\ 0 & 5.5 & 4.25 & | & -2.5 \\ & & & | & \\ 0 & 3 & 4 & | & 2 \\ & & & | & \end{array}$$

Now perform operation on 3rd row as follows,

$$R3 \leftarrow R3 - \frac{3}{5.5} \times R2 = R3 - 0.55 \times R2$$

We get,

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & & b \\ \hline 4 & -2 & 1 & | & 18 \\ & & & | & \\ 0 & 5.5 & 4.25 & | & -2.5 \\ & & & | & \\ 0 & 0 & 1.66 & | & 3.38 \\ & & & | & \end{array}$$

Back substitution gives

$$x_3 = 2.04 ; x_2 = -2.03 ; x_1 = 2.98.$$

Ex. 7 : Solve the following system of linear equations with partial pivoting

$$x_1 - x_2 + 3x_3 = 3$$

$$2x_1 + x_2 + 4x_3 = 7$$

$$3x_1 + 5x_2 - 2x_3 = 6$$

Ex. 8 : Solve the following system of linear equations with row interchanges

$$0.3x_1 + 2.6x_2 + 1.3x_3 = 7.65$$

$$8.3x_1 + 8.2x_2 + 5.6x_3 = 43.17$$

$$12.7x_1 + 3.5x_2 + 7.4x_3 = 49.68$$

Compute up to two significant figures after the decimal.

2.4 ITERATIVE METHODS

The methods we have discussed above are called **direct methods**. There is another approach also for solving linear simultaneous equations known as **iterative methods**. In an iterative method we start from an approximate solution and improve it by repeated use of the method, called 'iterations'. When two successive solutions agree within a desired accuracy we stop the process and take it as final solution. But these methods do not guarantee stopping of process or a solution for a general system of linear equations. However, these methods guarantee solution for some particular classes of equations. For example, solution of $A\mathbf{x} = \mathbf{b}$ is guaranteed, if the coefficient matrix A is diagonally dominant, where :

(Definition) A matrix A is said to be **Diagonally Dominant Matrix** if the modulus of the diagonal element in any row is greater than the sum of the moduli of other elements in that row. Mathematically,

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|, \quad i = 1, 2, \dots$$

It is important to note that sometimes the matrix A may not look diagonally dominant but by changing the order of equations the coefficient matrix may be converted to diagonally dominant.

For other forms of $A \underline{x} = \underline{b}$ also, the methods may solve the system, yet there is no guarantee.

Two iterative methods will be discussed :

(i) Gauss-Jacobi Iterative Method

(ii) Gauss-Seidel Iterative Method

We shall illustrate these methods with a 3×3 system of equations $A \underline{x} = \underline{b}$, say

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

Let us assume that the equations are arranged so that matrix A is diagonally dominant. We solve first equation for x_1 , second for x_2 and third for x_3 as follows :

$$x_1 = \frac{(b_1 - a_{12}x_2 - a_{13}x_3)}{a_{11}}$$

$$x_2 = \frac{(b_2 - a_{21}x_1 - a_{23}x_3)}{a_{22}}$$

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

As stated above we start from some approximate values of x_1, x_2 and x_3 . But they are not known generally and we start by taking $x_1 = x_2 = x_3 = 0$. These values are improved in successive iterations. Let us suppose that we have computed up to n th iteration i.e. values of $x_1^{(n)}, x_2^{(n)}$ and $x_3^{(n)}$ have been computed where superfix n denotes iteration. The initial guess for x_1, x_2 and x_3 may be considered as $x_1^{(0)}, x_2^{(0)}$ and $x_3^{(0)}$ respectively. The two methods may be expressed as follows.

2.4.1 Gauss-Jacobi Iterative Method

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

$$x_2^{(n+1)} = \frac{[b_2 - a_{21}x_1^{(n)} - a_{23}x_3^{(n)}]}{a_{22}}$$

$$x_3^{(n+1)} = \frac{[b_3 - a_{31}x_1^{(n)} - a_{32}x_2^{(n)}]}{a_{33}}$$

2.4.2 Gauss-Seidal Iterative Method

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

$$x_2^{(n+1)} = \frac{[b_2 - a_{21}x_1^{(n+1)} - a_{23}x_3^{(n)}]}{a_{22}}$$

$$x_3^{(n+1)} = \frac{[b_3 - a_{31}x_1^{(n+1)} - a_{32}x_2^{(n+1)}]}{a_{33}}$$

It may be noted that in the *Gauss-Seidel method*, the most recent values which have been computed even in the $(n+1)^{\text{th}}$ iteration are used when modifying value of a particular x_i while in *Gauss-Jacobi method* the values at n^{th} iteration only are used for computing all the values at the $(n+1)^{\text{th}}$ iteration. After having computed the value of the last x , the new values computed during the $(n+1)^{\text{th}}$ iteration are used in $(n+2)^{\text{th}}$ iteration. In general, Gauss-Seidel method converges faster than Gauss-Jacobi method.

Example 7

Solve the following system of equations by Gauss-Jacobi and Gauss-Seidel methods, correct up to two places of decimal only.

$$-4x_1 + x_2 + 10x_3 = 21$$

$$5x_1 - x_2 + x_3 = 14$$

$$2x_1 + 8x_2 - x_3 = -7$$

First rearrange the equations so that coefficient matrix is diagonal dominant.

Solution

Rearranging equations,

$$5x_1 - x_2 + x_3 = 14$$

$$2x_1 + 8x_2 - x_3 = -7$$

$$-4x_1 + x_2 + 10x_3 = 21$$

Gauss-Jacobi scheme is as follows :

$$x_1^{(n+1)} = \frac{(14 + x_2^{(n)} - x_3^{(n)})}{5}$$

$$x_2^{(n+1)} = \frac{(-7 - 2x_1^{(n)} + x_3^{(n)})}{8}$$

$$x_3^{(n+1)} = \frac{(21 + 4x_1^{(n)} - x_2^{(n)})}{10}$$

Taking $x_1 = x_2 = x_3 = 0$

$$x_1^{(1)} = \frac{14}{5} = 2.80 ; x_2^{(1)} = -\frac{7}{8} = -0.88 ; x_3^{(1)} = \frac{21}{10} = 2.10$$

$$x_1^{(2)} = \frac{(14 - 0.88 - 2.10)}{5} = 2.20$$

$$x_2^{(2)} = \frac{(-7 - 2 \times 2.80 + 2.10)}{8} = -1.31$$

$$x_3^{(2)} = \frac{(21 + 4 \times 2.80 + 0.88)}{10} = 3.31$$

We show other computations in the form of a table as given below :

$x \backslash N$	0	1	2	3	4	5	6	7	8
$x_1^{(n)}$	0	2.80	2.20	1.88	1.98	2.02	2.00	2.00	2.00
$x_2^{(n)}$	0	-0.88	-1.31	-1.01	-0.96	-1.00	-1.00	-1.00	-1.00
$x_3^{(n)}$	0	2.10	3.31	3.11	2.95	2.99	3.01	3.00	3.00

Gauss-Seidel scheme may be written as follows :

$$x_1^{(n+1)} = \frac{(14 + x_2^{(n)} - x_3^{(n)})}{5}$$

$$x_2^{(n+1)} = \frac{(-7 - 2x_1^{(n+1)} + x_3^{(n)})}{8}$$

$$x_3^{(n+1)} = \frac{(21 + 4x_1^{(n+1)} - x_2^{(n+1)})}{10}$$

Taking $x_1 = x_2 = x_3 = 0$, the first iteration gives,

$$x_1^{(1)} = \frac{14}{5} = 2.80$$

$$x_2^{(1)} = \frac{(-7 - 2 \times 2.80 + 0)}{8} = -1.58$$

$$x_3^{(1)} = \frac{(21 + 4 \times 2.80 + 1.58)}{10} = 3.38$$

Second iteration gives

$$x_1^{(2)} = \frac{(14 - 1.58 - 3.38)}{5} = 1.81$$

$$x_2^{(2)} = \frac{(-7 - 2 \times 1.81 + 3.38)}{8} = -0.90$$

$$x_3^{(2)} = \frac{(21 + 4 \times 1.81 + 0.90)}{10} = 2.91$$

Let us write down iterations in tabular form as follows :

$x \backslash n$	0	1	2	3	4	5	6
$x_1^{(n)}$	0	2.80	1.81	2.04	1.99	2.02	2.00
$x_2^{(n)}$	0	-1.58	-0.90	-1.02	-1.00	-1.00	-1.00
$x_3^{(n)}$	0	3.38	2.91	3.02	3.00	3.00	3.00

Example 8

Solve the following linear system $Ax = b$ by Jacobi method rounded to four decimal places.

$$10x_1 - x_2 + 2x_3 = 6$$

$$-x_1 + 11x_2 - x_3 + 3x_4 = 25$$

$$2x_1 - x_2 + 10x_3 - x_4 = -11$$

$$3x_2 - x_3 + 8x_4 = 15$$

Solution

Letting $x^{(0)} = (0, 0, 0, 0)^T$, we get

$$x^{(1)} = (0.6000, 2.2727, -1.1000, 1.8750)^T$$

$$x^{(2)} = (1.0473, 1.7159, -0.8052, 0.8852)^T$$

and $x^{(3)} = (0.9326, 2.0533, -1.0493, 1.1309)^T$

Proceeding similarly one can obtain

$$\mathbf{x}^{(5)} = (0.9890, 2.0114, -1.0103, 1.0214)^T \text{ and}$$

$$\mathbf{x}^{(10)} = (1.0001, 1.9998, -0.9998, 0.9998)^T.$$

The solution is $\mathbf{x} = (1, 2, -1, 1)^T$. You may note that $\mathbf{x}^{(10)}$ is a good approximation to the exact solution compared to $\mathbf{x}^{(5)}$.

You also observe that A is strictly diagonally dominant (since $10 > 1 + 2$, $11 > 1 + 1 + 3$, $10 > 2 + 1 + 1$ and $8 > 3 + 1$).

Example 9

Solve the linear system $A\mathbf{x} = \mathbf{b}$ given below (same as in the above example) by Gauss-Seidel method rounded to four decimal places.

$$10x_1 - x_2 + 2x_3 = 6$$

$$-x_1 + 11x_2 - x_3 + 3x_4 = 25$$

$$2x_1 - x_2 + 10x_3 - x_4 = -11$$

$$3x_2 - x_3 + 8x_4 = 15$$

The equations can be written as follows :

$$x_1^{(k+1)} = \frac{1}{10} x_2^{(k)} - \left(\frac{1}{5}\right) x_3^{(k)} + \frac{3}{5}$$

$$x_2^{(k+1)} = \frac{1}{11} x_1^{(k+1)} + \frac{1}{11} x_3^{(k)} - \frac{3}{11} x_4^{(k)} + \frac{25}{11}$$

$$x_3^{(k+1)} = -\left(\frac{1}{5}\right) x_1^{(k+1)} + \frac{1}{10} x_2^{(k+1)} + \frac{1}{10} x_4^{(k)} - \frac{11}{10}$$

$$x_4^{(k+1)} = -\frac{3}{8} x_2^{(k+1)} + \frac{1}{8} x_3^{(k+1)} + \frac{15}{8}$$

Letting $\mathbf{x}^{(0)} = (0, 0, 0, 0)^T$ we have from first equation

$$x_1^{(1)} = 0.6000$$

$$x_2^{(1)} = \frac{0.6000}{3} + \frac{25}{11} = 2.3273$$

$$x_3^{(1)} = -\frac{0.6000}{5} + \frac{1}{10} (2.3273) - \frac{11}{10} = -0.1200 + 0.2327 - 1.1000 = -0.9873$$

$$\begin{aligned} x_4^{(1)} &= -\frac{3}{8} (2.3273) + \frac{1}{8} (-0.9873) + \frac{15}{8} \\ &= -0.8727 - 0.1234 + 1.8750 \\ &= 0.8789 \end{aligned}$$

Using $\mathbf{x}^{(1)}$ we get

$$\mathbf{x}^{(2)} = (1.0300, 2.037, -1.014, 0.9844)^T$$

and we can check that

$$\mathbf{x}^{(5)} = (1.0001, 2.0000, -1.0000, 1.0000)^T$$

Note that $\mathbf{x}^{(5)}$ is a good approximation to the exact solution.

Ex. 9 : Perform five iterations of the Jacobi method for solving the system of equations.

$$\begin{bmatrix} 5 & -1 & -1 & -1 \\ -1 & 10 & -1 & -1 \\ -1 & -1 & 5 & -1 \\ -1 & -1 & -1 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -4 \\ 12 \\ 8 \\ 34 \end{bmatrix}$$

Starting with $\mathbf{x}^{(0)} = (0, 0, 0, 0)^T$. The exact solution is $\mathbf{x} = (1, 2, 3, 4)^T$. How good $\mathbf{x}^{(5)}$ as an approximation to $\mathbf{x}$?

Ex. 10 : Perform four iterations of the Jacobi method for solving the following system of equations.

$$\begin{bmatrix} 2 & -1 & -0 & -0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

With $\mathbf{x}^{(0)} = (0.5, 0.5, 0.5, 0.5)^T$. Here $\mathbf{x} = (1, 1, 1, 1)^T$. How good $\mathbf{x}^{(5)}$ as an approximation to $\mathbf{x}$?

Ex. 11 : Perform four iterations (rounded to four decimal places) using Jacobi Method and Gauss-Seidel method for the following system of equations.

$$\begin{bmatrix} -8 & 1 & 1 \\ 1 & -5 & -1 \\ 1 & 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 16 \\ 7 \end{bmatrix}$$

With $\mathbf{x}^{(0)} = (0, 0, 0)^T$. The exact solution is $(-1, -4, -3)^T$. Which method gives better approximation to the exact solution?

Ex. 12 : For linear system given in Ex. 10 above, use the Gauss Seidel method for solving the system starting with $\mathbf{x}^{(0)} = (0.5, 0.5, 0.5, 0.5)^T$ obtain $\mathbf{x}^{(4)}$ by Gauss-Seidel method and compare this with $\mathbf{x}^{(4)}$ obtained by Jacobi method in Ex. 10.

2.4.3 Comparison of Direct and Iterative Methods

Each of the direct approach and iterative approach has its relative merits and demerits or strengths and weaknesses. The choice of the approach depends on the type of problem to be solved. We mention below, for each of the two approaches, the type of problems for which the approach is appropriate:

Direct Method

1. The direct methods are generally used when the matrix A is dense or filled, that is, there are few zero elements, and the order of the matrix is not very large, say $n < 50$.
2. The rounding errors may become quite large for ill conditioned equations (If at any stage during the application of pivoting strategy, it is found that all values of $\{ |a_{mk}| \text{ for } m = k + 1 \text{ to } n \}$ are less than a pre-assigned small quantity ϵ , then the equations are ill-conditioned and no useful solution is obtained). Ill-conditioned matrices are not discussed in this unit.

1. These methods are generally used when the matrix A is sparse and the order of the matrix A is very large say $n > 50$. Sparse matrices have very few non-zero elements.
2. An important advantage of the iterative methods is the small rounding error. Thus, these methods are good choice for ill-conditioned systems.
3. However, convergence may be guaranteed only under special conditions. But when convergence is assured, this method is better than direct.

With this we conclude this unit. Let us now recollect the main points discussed in this unit.

2.5 SUMMARY

In this unit we have dealt with the following:

1. We have discussed the direct methods and the iterative techniques for solving linear system of equations $Ax = b$ where A is an $n \times n$ non-singular matrix.
2. The direct methods produce the exact solution in a finite number of steps provided there are no round off errors. Direct method is used for linear system $Ax = b$ where the matrix A is dense and order of the matrix is less than 50.
3. In direct methods, we have discussed Gauss elimination, and Gauss elimination with partial (maximal column) pivoting.
4. We have discussed two iterative methods, Jacobi method and Gauss-Seidel method and stated the convergence criterion for the iteration scheme. The iterative methods are suitable for solving linear systems when the matrix is sparse and the order of the matrix is greater than 50.

2.6 SOLUTIONS/ANSWERS

Ex. 1 : The first step is to use first equation to eliminate the unknown x_1 from second, third and fourth equation. This is accomplished by performing $E_2 - 2E_1$, $E_3 - 3E_1$ and $E_4 + E_1$. This gives the derived system as :

$$E'_1: \quad x_1 + x_2 + 0.x_3 + 3x_4 = 1$$

$$E'_2: \quad -x_2 - x_3 - 5x_4 = -4$$

$$E'_3: \quad -4x_2 - x_3 - 7x_4 = -3$$

$$E'_4: \quad 3x_2 + 3x_3 + 2x_4 = -1$$

In this new system, E'_2 is used to eliminate x_2 from E'_3 and E'_4 by performing the operations $E'_3 - 4E'_2$ and $E'_4 + 3E'_2$. The resulting system is

$$E''_1: \quad x_1 + x_2 + 0.x_3 + 3x_4 = 1$$

$$E''_2: \quad -x_2 - x_3 - 5x_4 = -4$$

$$E''_3: \quad 3x_3 + 13x_4 = 3$$

$$E''_4: \quad -13x_4 = -13$$

This system of equation is now in triangular form and can be solved by back substitution.

E''_4 gives $x_4 = 1$, E''_3 gives

$$x_3 = \frac{1}{3}(13 - 13x_4) = \frac{1}{3}(13 - 13 \times 1) = 0.$$

E''_2 gives $x_2 = -x_3 - 5x_4 + 4 = 0 - 5 + 4 = -1$

and E''_1 gives $x_1 = 4 - 3x_4 - x_2 = 4 - 3 \times 1 - 2 = -1$

The above procedure can be carried out conveniently in matrix form as shown below:

We consider the Augmented matrix $[A|b]$ and perform the elementary row operations on the augmented matrix.

$$[A|b] = \left[\begin{array}{cccc|c} 1 & -1 & -0 & 3 & -1 \\ 2 & 1 & -1 & 1 & -2 \\ 3 & -1 & -1 & 2 & 0 \\ -1 & 2 & 3 & -1 & -2 \end{array} \right]$$

$R_2 - 2R_1$, $R_3 - 3R_1$ and $R_4 + R_1$ gives

$$= \left[\begin{array}{cccc|c} 1 & 1 & 0 & 3 & 1 \\ 0 & -1 & -1 & -5 & -4 \\ 0 & -4 & -1 & -7 & -3 \\ 0 & 3 & 3 & 2 & -1 \end{array} \right]$$

$R_3 - 4R_2$, $R_4 + 3R_2$ gives

$$= \left[\begin{array}{cccc|c} 1 & 1 & 0 & 3 & 1 \\ 1 & -1 & -1 & -5 & -4 \\ 0 & 0 & 3 & 13 & 13 \\ 0 & 0 & 0 & -13 & -13 \end{array} \right]$$

This is the final equivalent system :

$$x_1 + x_2 + 0x_3 + 3x_4 = 1$$

$$-x_2 - x_3 - 5x_4 = -4$$

$$3x_3 + 13x_4 = 13$$

$-13x_4 = -13$. Last equation gives $x_4 = 1$ and so on.

Ex. 2 :

$$[A|b] = \left[\begin{array}{ccc|c} 3 & 2 & 1 & 3 \\ 2 & 1 & 1 & 0 \\ 6 & 2 & 4 & 6 \end{array} \right]$$

$$\xrightarrow{a_{11} \neq 0} \left[\begin{array}{ccc|c} 3 & 2 & 1 & 3 \\ 0 & -\frac{1}{3} & \frac{1}{3} & -2 \\ 0 & -2 & 2 & 0 \end{array} \right]$$

$$\xrightarrow{a_{22}^{(1)} \neq 0} \left[\begin{array}{ccc|c} 3 & 2 & 1 & 3 \\ 0 & -\frac{1}{3} & \frac{1}{3} & -2 \\ 0 & 0 & 0 & 12 \end{array} \right]$$

This system has no solution since x_3 cannot be determined from the last equation. This system is said to be inconsistent. Also note that $\det(A) = 0$.

$$[A|b] = \left[\begin{array}{ccc|c} 16 & 22 & 4 & -2 \\ 4 & -3 & 2 & 9 \\ 12 & 25 & -2 & -11 \end{array} \right]$$

$$\xrightarrow{a_{11} \neq 0} \left[\begin{array}{ccc|c} 16 & 22 & 4 & 2 \\ 0 & -\frac{17}{2} & 1 & \frac{19}{2} \\ 0 & \frac{17}{2} & -1 & -\frac{19}{2} \end{array} \right] \begin{array}{l} R_2 \leftarrow R_2 - \left(\frac{1}{4}\right) R_1 \\ R_3 \leftarrow R_3 + R_1 \end{array}$$

$$\xrightarrow{a_{22}^{(1)} \neq 0} \left[\begin{array}{ccc|c} 16 & 22 & 4 & 2 \\ 0 & -\frac{17}{2} & 1 & \frac{19}{2} \\ 0 & 0 & 0 & 0 \end{array} \right] R_3 \leftarrow R_3 + R_2$$

$\Rightarrow x_3 = \text{arbitrary value and}$

$$x_2 = \frac{-2}{17} \left(\frac{19}{2} - x_3 \right) \text{ and } x_1 = \left(\frac{1}{16} \right) (-2 - 22x_2 - 4x_3).$$

This system has infinitely many solutions. Also you may check that $\det(A) = 0$.

Ex.4 : We write the given equations in matrix form as follows :

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2.099 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 3.901 \\ 6 \end{bmatrix}$$

Multiply the first row by 3/10 and add to the second equation, we get

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.001 \\ 6 \end{bmatrix}$$

Multiply the first row by 5/10 and subtract from the third equation, we get

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.001 \\ 2.5 \end{bmatrix}$$

This completes the first step of forward elimination.

Multiply the second equation by $\frac{2.5}{(-0.005)} = -2500$ and subtract from the third equation, we obtain

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 0 & 15005 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.001 \\ 15005 \end{bmatrix}$$

We can now solve the above equations by back substitution. From the third equation, we get

$$15005x_3 = 15005, \text{ or } x_3 = 1.$$

Substituting the value of x_3 in the second equation, we get

$$-0.001x_2 + 6x_2 = 6.001, \text{ or } -0.001x_2 = 6.001 - 6 = 0.001, \text{ or } x_2 = -1$$

Substituting the values of x_3 and x_2 in the first equation, we get

$$10x_1 - 7x_2 = 7, \text{ or } 10x_1 = 7 + 7x_2 = 0, \text{ or } x_1 = 0.$$

Hence, the solution is $[0 \ -1 \ 1]^T$.

Ex. 5 : Final derived system is :

$$\left[\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 0 & 2 & 4 \end{array} \right]$$

and the solution is $x_4 = 2, x_3 = 2, x_2 = 3, x_1 = -7$.

Ex. 6 : The extended matrix is :

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 3 \\ 1 & 1 & 0 & 2 & 0 \\ 2 & 2 & 3 & 0 & 10 \\ -1 & -1 & -2 & 2 & -8 \end{array} \right]$$

Using $R_2 \leftarrow R_2 - R_1, R_3 \leftarrow R_3 - 2R_1, R_4 \leftarrow R_4 + R_1$, we get

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 3 \\ 0 & 0 & -1 & 1 & -3 \\ 0 & 0 & 1 & -2 & 4 \\ 0 & 0 & -1 & 3 & -5 \end{array} \right]$$

Using $R_3 \leftarrow R_3 + R_2, R_4 \leftarrow R_4 - R_2$, we get

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 3 \\ 0 & 0 & -1 & 1 & -3 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{array} \right]$$

and the solutions are $x_4 = -1, x_3 = 2, x_2$ is arbitrary and $x_1 = 2 - x_2$.

Thus this linear system has infinite number of solutions.

Ex. 7 : The given system of equations can be written in matrix equation form as

$$[A | b] = \left(\begin{array}{ccc|c} 1 & -1 & 3 & 3 \\ 2 & 1 & 4 & 7 \\ 3 & 5 & -2 & 6 \end{array} \right) \quad \begin{array}{l} \text{Largest element in Col. 1 is 3, therefore} \\ R_1 \leftarrow R_1 - (1/3) R_3; \quad R_2 \leftarrow R_2 - (2/3) R_3 \end{array}$$

$$= \left(\begin{array}{ccc|c} 0 & -\frac{8}{3} & \frac{11}{3} & 1 \\ 0 & -\frac{7}{3} & \frac{16}{3} & 3 \\ 3 & 5 & -2 & 6 \end{array} \right) \quad \begin{array}{l} \text{Largest magnitude element in Col. 2 is } -8/3, \\ \text{therefore} \\ R_2 \leftarrow R_2 - (-7/3) (-3/8) R_3 \end{array}$$

$$= \left(\begin{array}{ccc|c} 0 & -\frac{8}{3} & \frac{11}{3} & 1 \\ 0 & 0 & \frac{51}{24} & \frac{17}{8} \\ 3 & 5 & -2 & 6 \end{array} \right)$$

Re-arranging the equations (3rd equation becomes the first equation and first equation becomes the second equation in the derived system), we have

$$3x_1 + 5x_2 - 2x_3 = 6$$

$$-\frac{8}{3}x_2 + \frac{11}{3}x_3 = 1$$

$$\frac{51}{24}x_3 = \frac{17}{8}$$

Using back substitution we have $x_1 = 1$, $x_2 = 1$ and $x_3 = 1$.

Ex. 8 :

$$[A|b] = \begin{array}{cccc|c} x_1 & x_2 & x_3 & & b \\ \hline 0.3 & 2.6 & 1.3 & & 7.65 \\ 8.3 & 8.2 & 5.6 & & 43.17 \\ 12.7 & 3.5 & 7.4 & & 49.68 \end{array}$$

Since, $|12.7|$ is the largest in absolute value in the first column, we interchange first row with third row

$$[A|b] = \begin{array}{cccc|c} x_1 & x_2 & x_3 & & b \\ \hline 12.7 & 3.5 & 7.4 & & 49.68 \\ 8.3 & 8.2 & 5.6 & & 43.17 \\ 0.3 & 2.6 & 1.3 & & 7.65 \end{array}$$

$$m_{21} = -8.3/12.7 = -0.65, R_2 \leftarrow R_2 - 0.65 R_1$$

$$m_{31} = -0.3/12.7 = -0.024, \text{ (two significant figures after decimal)}$$

$$R_3 \leftarrow R_3 - 0.024 R_1$$

$$[A|b] = \begin{array}{cccc|c} x_1 & x_2 & x_3 & & b \\ \hline 12.7 & 3.5 & 7.4 & & 49.68 \\ 0 & 5.92 & 0.79 & & 10.88 \\ 0 & 2.52 & 1.12 & & 6.46 \end{array}$$

Since, $|5.92|$ is larger in absolute value in the second column (now, only from row 2 and row 3), rows are not interchanged

$$m_{32} = -2.52/5.92 = -0.42; R_3 \leftarrow R_3 - 0.42 R_2$$

$$[A|b] = \begin{array}{cccc|c} x_1 & x_2 & x_3 & & b \\ \hline 12.7 & 3.5 & 7.4 & & 49.68 \\ 0 & 5.92 & 0.79 & & 10.88 \\ 0 & 0 & 0.79 & & 1.89 \end{array}$$

By back-substitution, we get

$$x_3 = 2.39, x_2 = 1.52, x_1 = 2.10$$

Ex. 9 : Using $\mathbf{x}_{(0)} = [0, 0, 0, 0]^T$ we have

$$\mathbf{x}^{(1)} = [-0.8, 1.2, 1.6, 3.4]^T$$

$$\mathbf{x}^{(2)} = [0.44, 1.62, 2.36, 3.6]^T$$

$$\mathbf{x}_{(3)} = [0.716, 1.84, 2.732, 3.842]^T$$

$$\mathbf{x}_{(4)} = [0.8823, 1.9290, 2.8796, 3.9288]^T$$

Ex. 10 : Using $\mathbf{x}^{(0)} = [0.5, 0.5, 0.5, 0.5]^T$, we have

$$\mathbf{x}^{(1)} = [0.75, 0.5, 0.5, 0.75]^T$$

$$\mathbf{x}^{(2)} = [0.75, 0.625, 0.625, 0.75]^T$$

$$\mathbf{x}^{(3)} = [0.8125, 0.6875, 0.6875, 0.8125]^T$$

$$\mathbf{x}^{(4)} = [0.8438, 0.75, 0.75, 0.8438]^T$$

Ex. 11 : By Jacobi method we have

$$\mathbf{x}^{(1)} = [-0.125, -3.2, -1.75]^T$$

$$\mathbf{x}^{(2)} = [-0.7438, -3.5750, -2.5813]^T$$

$$\mathbf{x}^{(3)} = [-0.8945, -3.8650, -2.8297]^T$$

$$\mathbf{x}^{(4)} = [-0.9618, -3.9448, -2.9399]^T$$

whereas by Gauss-Seidel method, we have

$$\mathbf{x}^{(1)} = [-0.125, -3.225, -2.5875]^T$$

$$\mathbf{x}^{(2)} = [-0.8516, -3.8878, -2.9349]^T$$

$$\mathbf{x}^{(3)} = [-0.9778, -3.9825, -2.9901]^T$$

$$\mathbf{x}^{(4)} = [-0.9966, -3.9973, -2.9985]^T$$

Ex. 12 : Starting with the initial approximation

$\mathbf{x}^{(0)} = [0.5, 0.5, 0.5, 0.5]^T$, we have the following iterates:

$$\mathbf{x}^{(1)} = [0.75, 0.625, 0.5625, 0.7813]^T$$

$$\mathbf{x}^{(2)} = [0.8125, 0.6875, 0.7344, 0.8672]^T$$

$$\mathbf{x}^{(3)} = [0.8438, 0.7891, 0.8282, 0.9141]^T$$

$$\mathbf{x}^{(4)} = [0.8946, 0.8614, 0.8878, 0.9439]^T$$

Since the exact solution is $\mathbf{x} = [1, 1, 1, 1]^T$, the Gauss-Seidel method gives better approximation than the Jacobi method at fourth iteration.

UNIT 3 SOLUTION OF NON-LINEAR EQUATIONS

Structure

- 3.0 Introduction
- 3.1 Objectives
- 3.2 Fixed-Point Method (Successive Substitution)
- 3.3 Bisection Method
- 3.4 Regula-Falsi Method
- 3.5 Secant Method
- 3.6 Newton-Raphson Method
- 3.7 Summary
- 3.8 Solutions/Answers

3.0 INTRODUCTION

In this unit, we discuss one of the most basic problems in numerical analysis, namely, that of finding the zeroes of a real-valued function of x defined over a finite interval. We assume it is continuous and differentiable. If $f(x)$ becomes zero for some value $x = \alpha$, say, i.e. $f(\alpha) = 0$, then $x = \alpha$ is called a ZERO of the FUNCTION or a ROOT of the EQUATION $f(x) = 0$. We shall discuss methods for finding the roots of an equation $f(x) = 0$ where $f(x)$ may contain algebraic or transcendental expressions.

An equation of the type $f(x) = 0$ is algebraic if it contains only non-negative integral powers of x , that is, $f(x)$ is a polynomial. The equation is called transcendental, if it may contain powers of x , but also at least some other functions like exponential functions, logarithm functions etc. Through the following examples, we further illustrate these concepts

Examples of algebraic equations : $2x = 5$, $x^2 + x = 1$, $x^7 = x(1 + 2x)$

Examples of transcendental equations $x + \sin x = 0$, $e^{\sqrt{x}} = x$, $\tan x = x$

Also, we mention some Properties of polynomial equations, which may be useful in their solutions :

- (i) The total number of roots of an algebraic equation is the same as its degree.
- (ii) An algebraic equation can have at most as many positive roots as the number of changes of sign in the coefficients of $f(x)$.
- (iii) An algebraic equation can have at most as many negative roots as the number of changes of sign in the coefficient of $f(-x)$.
- (iv) If $f(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-1}x + a_n$ have roots $\alpha_1, \alpha_2, \dots, \alpha_n$, then the following hold good :

$$\sum_i \alpha_i = -\frac{a_1}{a_0}, \quad \sum_{i < j} \alpha_i \alpha_j = \frac{a_2}{a_0}, \quad \prod_i \alpha_i = (-1)^n \frac{a_n}{a_0}$$

We shall be interested in real roots only. It is also assumed that the roots are simple (non-repeated) and isolated and well-separated i.e. there is a finite neighbourhood about the root in which no other root exists. All the methods discussed will be iterative type, i.e. we start from an approximate value of the root and improve it by applying the method successively until two values agree within desired accuracy. It is important to note that approximate root is not chosen arbitrarily. Instead, we look for an interval in which only

one root lies and choose the initial value suitably in that interval. In this respect, the following theorem proves to be quite useful.

Theorem : If $f(x)$ is continuous in the closed interval $[a, b]$ and $f(a)$ and $f(b)$ are of opposite signs, then there is at least one real root α of the equation $f(x) = 0$ such that $a < \alpha < b$.

Usually we have to compute the function values at several points but sometimes we have to get the approximate value graphically close to the exact root.

In this unit, we discuss the following five methods of finding the zeros of a real-valued function, or, equivalently, the roots of an equation $f(x) = 0$. The methods to be discussed, in order, are Fixed-point Method, Bisection Method, Regula-falsi Method, Secant Method and Newton-Raphson Method.

3.1 OBJECTIVES

After going through this unit, you should be able to find an approximate real root of the equation $f(x) = 0$ by any one of the following methods : Bisection Method, Fixed-point Method, Regula-falsi Method, Newton-Raphson Method and Secant Method.

3.2 FIXED-POINT METHOD (SUCCESSIVE SUBSTITUTION)

Suppose we have to find the roots of the equation $f(x) = 0$. We express $f(x) = 0$ in the form¹ $x = \phi(x)$ and the iterative scheme is given as

¹ Some related concepts: (mathematical) expression, equation, inequality and function etc. : We explain the concepts and relations between these concepts through some examples.

Any (legal, mathematically acceptable) sequence of symbols is an expression including

	15	... (0)
and	$19x^6 - 17x^2 - 5$	... (i)
	$19x^{-6} - 17x^2 - 5$	... (ii)
	$5x = 6$	... (iii)
	$19x^6 - 17x^2 - 5 = 8x$	... (iv)
and	$19x^6 - 17x^2 - 5 < 8x$	... (v)
	$3x^2 + 5 \log(x) - e^x$	... (vi)
And	$3x^2 + 5 \log(x) - e^x = 5 \cos(x)$	... (vii)

But each of the following is not (mathematical) expression

$$19x^6 - 17x^2 - \div 5; 19x^6 - '17x^2 + - \leq 5; 3x^2 + 5 \log(x) - < e^x = 5 \cos(x)$$

Out of the legal sequences, (i) is a polynomial, but not an equation; (ii) is not a polynomial, because power of x cannot be negative; (iii) is a linear equation, and also, polynomial equation; (iv) is a polynomial equation, but not linear equation; (v) is an inequality; (vi) is neither a polynomial, nor an equation; (vii) is an equation, but not polynomial equation. Also, (0) is a polynomial of degree 0.

Any expression, which does not include any relational symbol like $=, <, \leq$ etc. is frequently used in the sense of a function, e.g., each of x^2 , or e^{2x} is used in the sense of a function. For example, x^2 is used to denote the rule/ function that maps 1 to 1; 2 to 4 and so on. In case of such usage of x^2 , or any other such expression, we usually write $f(x) = x^2$. Even expression under (0) may be treated as a function, which associates 15 with every value of the variable.

In order to differentiate the role of an expression as a function, the expression $3x^2 + 5 \log(x) - e^x$ is written as $f(x) = 3x^2 + 5 \log(x) - e^x$, for some symbol f . But for different expressions as functions, different symbols should be used.

(Footnote continues on next page)

$$x_{n+1} = \phi(x_n)$$

where x_n denotes the n^{th} iterated value which is known and x_{n+1} denotes $(n+1)^{\text{th}}$ approximated value which is to be computed. However, $f(x) = 0$ can be expressed in the form $x = \phi(x)$ in many ways but the corresponding iterative scheme may not converge in all cases to the true value, rather it may diverge and start giving absurd values. It can be proved that necessary and sufficient condition for convergence of the scheme is that the modulus of the first derivative of $\phi(x)$ i.e. $\phi'(x)$ at the exact root should be less than 1 i.e. if α is the exact root then $|\phi'(\alpha)| < 1$. But since we do not know the exact root which is to be computed we test the condition for convergence at the initial approximation i.e. $|\phi'(x_0)| < 1$. Hence, it is necessary that the initial approximation should be taken quite close to the exact root and test the condition before starting the iteration. This method is also known as 'fixed point' method since the mapping $x = \phi(x)$ maps the root α to itself since $\alpha = \phi(\alpha)$ i.e. α remains unchanged (fixed) under the mapping $x = \phi(x)$.

Example 1

Find the positive root of $x^3 - 2x - 8 = 0$ by method of successive substitution correct up to two places of decimal.

Solution

$$f(x) = x^3 - 2x - 8$$

To find the approximate location of the root (+ive), we try to evaluate the function values at different x and tabulate as follows :

x	0	1	2	3	$x > 3$
$f(x)$	-8	-9	-4	13	+ive
Sign of $f(x)$	-	-	-	+	+

The root lies between 2 and 3. Let us choose the initial approximation as $x_0 = 2.5$.

Let us express $f(x) = 0$ as $x = \phi(x)$ in the following forms and check for each, whether $|\phi'(\alpha)| < 1$ for $x = 2.5$.

$$(i) \quad x = x^3 - x - 8$$

$$(ii) \quad x = \frac{1}{2}(x^3 - 8)$$

$$(iii) \quad x = (2x + 8)^{\frac{1}{3}}$$

Expressions may be involved in equations and inequalities also. Here, each of the expressions may again be treated as a function also.

An equation may be expressed with different functions/relevant expressions. For example, the equation $x^2 + 5x + 6 = 0$ (two functions involved are $x^2 + 5x + 6$ and 0) ... (A)

can also be re-written as, say, $x^2 = -5x - 6$... (B)

or even $x = -5 - 6/x$, for $x \neq 0$... (C)

Let us denote the various expressions as functions, as follows:

Let us denote $f(x) = x^2 + 5x + 6$ and $g(x) = 0$; $\psi(x) = x^2$, $h(x) = -5x - 6$; $\xi(x) = x$, and $\phi(x) = -5 - 6/x$, then the same equation is represented by the following functional equations

$$f(x) = g(x); \psi(x) = h(x); \text{ and } \xi(x) = x, \text{ and } \xi(x) = \phi(x), \text{ for if } x \neq 0$$

If 'x' occurs on L.H.S., then instead of function notation, we just keep 'x' itself. Thus, instead of $\xi(x) = \phi(x)$, we write $x = \phi(x)$. Also if R.H.S. is a constant, say 0 or 15.87 etc. then again the constant, instead of functional notation is used. Thus, instead of $f(x) = g(x)$, we write $f(x) = 0$. Thus, we have explained how from $f(x) = 0$ we get $x = \phi(x)$.

We see that in cases (i) and (ii) $|\phi'(x)| > 1$, hence we should discard these representations. As the third case satisfies the condition,

$$|\phi'(x)| = \left| \frac{1}{3(2x+8)^{\frac{2}{3}}} \right| < 1 \text{ for } x = 2.5 \text{ we have the iteration scheme as,}$$

$$x_{n+1} = (2x_n + 8)^{\frac{1}{3}}$$

Starting from $x_0 = 2.5$, we get the successive iterates as shown in the table below :

n	0	1	2	3
x_n	2.5	2.35	2.33	2.33

Ex. 1 : A fixed point iteration to find a root of $3x^3 + 2x^2 + 3x + 2 = 0$ close to $x_0 = -0.5$

$$\text{is written as } x_{k+1} = -\frac{(2 + 3x_k + 2x_k^2)}{3x_k^2}.$$

Does this iteration converge? If so, iterate twice. If not, write a suitable form of the iteration, show that it converges and iterate twice to find the root.

Ex. 2 : Do three iterations of fixed point iteration method to find the smallest positive roots of $x^2 - 3x + 1 = 0$, by choosing a suitable iteration function, that converges. Start with $x_0 = 0.5$.

3.3 BISECTION METHOD

Bisection Method (Method of Halving)

In this method we find an interval in which the root lies and that there is no other root in that interval. Then we keep on narrowing down the interval to half at each successive iteration. We proceed as follows :

- (i) Find interval $I = (x_1, x_2)$ in which the root of $f(x) = 0$ lies and that there is no other root in I .
- (ii) Bisect the interval at $x = \frac{x_1 + x_2}{2}$ and compute $f(x)$. If $|f(x)|$ is less than the desired accuracy then it is the root of $f(x) = 0$.
- (iii) Otherwise check sign of $f(x)$. If $\text{sign}\{f(x)\} = \text{sign}\{f(x_2)\}$ then root lies in the interval $[x_1, x]$ and if they are of opposite signs then the root lies in the interval $[x, x_2]$. Change x to x_2 or x_1 accordingly. We may test sign of $f(x) \times f(x_2)$ for same sign or opposite signs.
- (iv) Check the length of interval $|x_1 - x_2|$. If an accuracy of say, two decimal places is required then stop the process when the length of the interval is 0.005 or less. We may take the midvalue $x = \frac{x_1 + x_2}{2}$ as the root of $f(x) = 0$. Otherwise, repeat step (ii), the convergence of this method is very slow in the beginning.

Example 2

Find the positive root of the equation $x^3 + 4x^2 - 10 = 0$ by bisection method correct upto two places of decimal.

Solution

$$f(x) \equiv x^3 + 4x^2 - 10 = 0$$

Let us find location of the +ive roots.

X	0	1	2	> 2
f(x)	-10	-5	14	
Sign f(x)	-	-	+	+

There is only one +ive root and it lies between 1 and 2. Let $x_1 = 1$ and $x_2 = 2$; at $x = 1$, $f(x)$ is -ive and at $x = 2$, $f(x)$ is +ive. We examine the sign of $f(x)$ at

$$x = \frac{x_1 + x_2}{2} = 1.5 \text{ and check whether the root lies in the interval } (1, 1.5) \text{ or}$$

$(1.5, 2)$. Let us show the computations in the table below :

Iteration No.	$x = \frac{x_1 + x_2}{2}$	Sign f(x)	Sign f(x) $\times$ f(x ₂)	x ₁	x ₂
1	1.5	+ 2.375	+	1	1.5
2	1.25	- 1.797	-	1.25	1.5
3	1.375	+ 0.162	+	1.25	1.375
4	1.3125	- 0.8484	-	1.3125	1.375
5	1.3438	- 0.3502	-	1.3438	1.375
6	1.3594	- 0.0960	-	1.3594	1.375
7	1.367	- 0.0471	-	1.367	1.375
8	1.371	+ 0.0956	+	1.367	1.371

We see that $|x_1 - x_2| = 0.004$.

We can choose the root as $x = \frac{1.367 + 1.371}{2} = 1.369$.

Ex. 3 : Obtain the smallest positive root of the equation of $x^3 - 5x + 1 = 0$ by using 3 iterations of the bisection method.

Ex. 4 : Apply bisection method to find an approximation to the positive root of the equation, $2x - 3 \sin x - 5 = 0$ rounded off to three decimal places.

3.4 REGULA-FALSI METHOD

In this method also, we find two values of x say x_1 and x_2 where function $f(x)$ has opposite signs and there is only one root in the interval (x_1, x_2) . Let us express the function as $y = f(x)$ and we are interested in finding the value of x where curve $y = f(x)$ intersects x -axis i.e. $y = 0$. We identify two points (x_1, y_1) and (x_2, y_2) on the curve. Then we approximate the curve by a straight line joining these two points. We find the point on the x -axis where this line cuts the x -axis. The equation of the straight line passing through (x_1, y_1) and (x_2, y_2) is given by

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

The point on x -axis where $y = 0$ is given by

$$x = \frac{x_1 y_2 - x_2 y_1}{y_2 - y_1} \quad \dots (A)$$

Now we check the sign of $f(x)$ and proceed like as we did in the bisection method. That is, if $f(x)$ has same sign as $f(x_2)$ then root lies in the interval (x_1, x) and if they have opposite signs, then it lies in the interval (x, x_2) . See Figure 3.1.

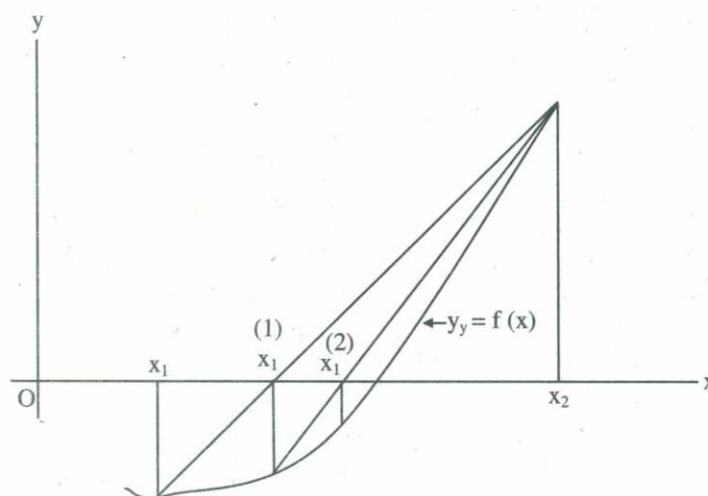


Figure 3.1 : Regula-Falsi Method. Superscript Shows Iteration Number

The main difference between Bisection Method and Regula Falsi is that for finding the next interval from the current interval $[x_1, x_2]$, in Bisection we use $x = \frac{x_1 + x_2}{2}$ whereas in Regula Falsi, we use (A) above.

Example 3

Find positive root of $x^3 + 4x^2 - 10 = 0$ by Regula-Falsi method. Compute upto the two decimal places only.

Solution

It is the same problem as given in the previous example. We start by taking $x_1 = 1$ and $x_2 = 2$. We have $y = x^3 + 4x^2 - 10$; $y_1 = -5$ and $y_2 = 14$. The point on the curve are $(1, -5)$ and $(2, 14)$. The points on the x-axis where the line joining these two points cuts it, is given by

I-Iteration

$$x = \frac{1 \times 14 - 2 \times (-5)}{14 + 5} = \frac{24}{19} = 1.26$$

$$y = f(x) = -1.65$$

II-Iteration

Take points $(1.26, -1.65)$ and $(2, 14)$

$$x = \frac{1.26 \times 14 - 2 \times (-1.65)}{14 + 1.65} = 1.34$$

$$y = f(x) = -0.41$$

III-Iteration

Take two points $(1.34, -0.41)$ and $(2, 14)$

$$x = \frac{1.34 \times 14 - 2 \times (-0.41)}{14 + 0.41} = 1.36$$

$$y = f(x) = -0.086$$

IV-Iteration

Take two points $(1.36, -0.086)$ and $(2, 14)$

$$x = \frac{1.36 \times 14 - 2 \times (-0.086)}{14 + 0.086} = 1.36$$

Since value of x repeats we take the root as $x = 1.36$.

Ex. 5 : It is known that the equation $x^3 + 7x^2 + 9 = 0$ has a root between -8 and -7 . Use the regula-falsi method to obtain the root rounded off to 3 decimal places. Stop the iteration when $|x_{i+1} - x_i| < 10^{-4}$.

3.5 SECANT METHOD

Like Regula-Falsi method, in this method also two values of x , say, x_1 and x_2 are chosen in the neighbourhood of the actual root but they may be on the same side or on the opposite sides of the root. Then a straight line is drawn through (x_1, y_1) and (x_2, y_2) and position of x is found where it intersects the x -axis. Then we take the points (x, y) and (x_1, y_1) or (x_2, y_2) and draw straight line and find point of intersection with x -axis and so on. See Figure 3.2.

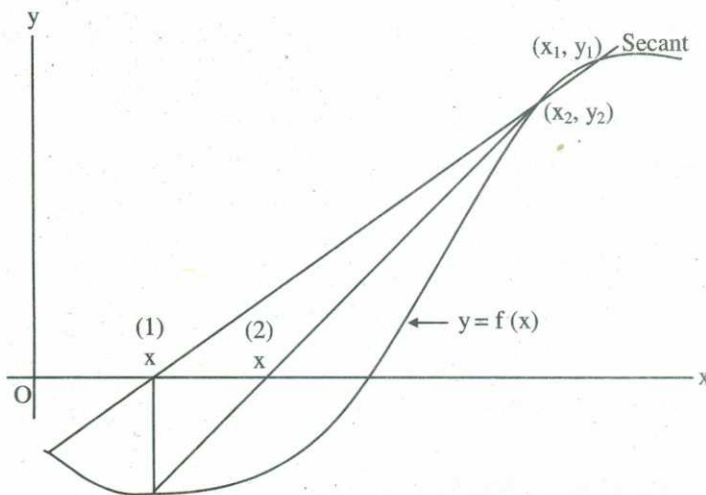


Figure 3.2 : Secant Method – Superscript Denotes Iteration Number

Example 4

Show four iterations of Secant method for finding the root of the equation $x^3 + 4x^2 - 10 = 0$ near $x = 0$ and $x = 1$. Compute upto two decimal places only.

Solution

$$f(x) \equiv x^3 + 4x^2 - 10 = 0; \quad y = f(x) = x^3 + 4x^2 - 10$$

$$y_1 = f(0) = -10; \quad y_2 = f(1) = -5$$

We have two points on the curve $(0, -10)$ and $(1, -5)$ and can draw a secant passing through these points. The point where it cuts x -axis is given by,

I-Iteration

$$x = \frac{0 \times (-5) - 1 \times (-10)}{-5 - (-10)} = 2.0$$

$$y = f(x) = 14$$

II-Iteration

Take two points $(1, -5)$ and $(2, 14)$

$$x = \frac{1 \times 14 - 2 \times (-5)}{14 + 5} = 1.26$$

$$y = f(x) = -1.65$$

III-Iteration

Take two points (1, -5) and (1.26, -1.65)

$$x = \frac{1 \times (-1.65) - 1.26 \times (-5)}{-1.65 + 5} = 1.39$$

$$y = f(x) = 0.41$$

IV-Iteration

Take (1.26, -1.65) and (1.39, 0.41)

$$x = \frac{1.26 \times 0.41 - 1.39 \times (-1.65)}{0.41 + 1.65} = 1.36$$

$$y = f(x) = -0.086.$$

Ex. 6 : Do three iterations of Secant method to find an approximate root of the equation.

$$3x^3 - 4x^2 + 3x - 4 = 0$$

Starting with initial approximations $x_0 = 0$ and $x_1 = 1$.

Ex. 7 : Do three iterations of the Secant method to solve the equation

$$x^3 + x - 6 = 0,$$

starting with $x_0 = 1$ and $x_1 = 2$.

Ex. 8 : Determine an approximate root of the equation

$$\cos x - xe^x = 0$$

using Secant method with the two initial approximations as $x_0 = 0$ and $x_1 = 1$. Do two iterations.

3.6 NEWTON-RAPHSON METHOD

The Newton-Raphson's method or commonly known as N-R method is most popular for finding the roots of an equation. Its approach is different from all the methods discussed earlier in the sense that it uses only one value of x in the neighbourhood of the root instead of two. We can explain the method geometrically as follows :

Let us suppose we want to find out the root of an equation $f(x) = 0$ while $y = f(x)$ represents a curve and we are interested in finding the point where it cuts the x -axis. Let $x = x_0$ be an initial approximate value of the root close to the actual root. We evaluate $y(x_0) = f(x_0) = y_0$ (say). Then point (x_0, y_0) lies on the curve $y = f(x)$. We find

$\frac{dy}{dx} = f'(x)$ for $x = x_0$, say $f'(x_0)$. Then we may draw a tangent at (x_0, y_0) given as,

$$y - y_0 = f'(x_0)(x - x_0).$$

The point where the tangent cuts the x -axis ($y = 0$) is taken as the next estimate $x = x_1$ for the root, i.e.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

In general $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ (see Figure 3.3)

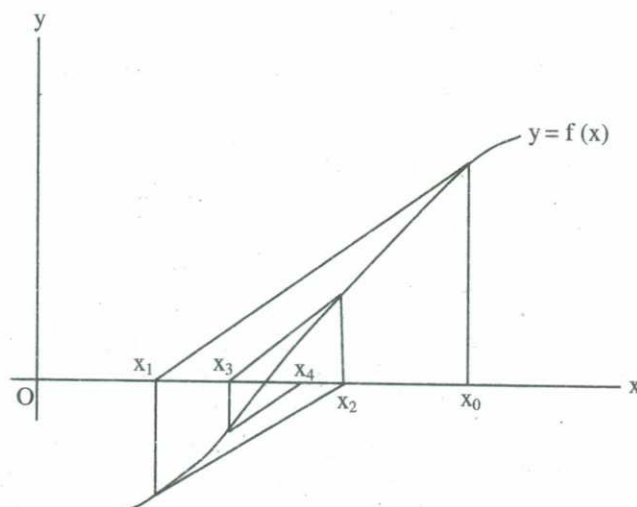


Figure 3.3 : Newton-Raphson Method

Theoretically, the N-R method may be explained as follows :

Let α be the exact root of $f(x) = 0$ and let $\alpha = x_0 + h$ where h is a small number to be determined. From Taylor's series as have,

$$f(\alpha) = f(x_0 + h) = f(x_0) + h f'(x_0) + \frac{h^2}{2} f''(x_0) + \dots = 0$$

Neglecting h^2 and higher powers we get an approximate value of h , as $h = -\frac{f(x_0)}{f'(x_0)}$.

Hence, an approximation for the exact root α may be written as,

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}$$

In general the N-R formula may be written as,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \dots$$

It is same as derived above geometrically. It may be stated that the convergence of N-R method is faster as compared to other methods. Further, comparing the N-R method with method of successive substitution, it can be seen as iterative scheme for

$$x = \phi(x)$$

where
$$\phi(x) = x - \frac{f(x)}{f'(x)}$$

The condition for convergence $|\phi'(\alpha)| < 1$ in this case would be

$$\phi'(x) = 1 - \frac{\{f'(x)\}^2 - f(x) f''(x)}{\{f'(x)\}^2} = \frac{f(x) f''(x)}{\{f'(x)\}^2}, \text{ at } x = \alpha.$$

This implies that $f'(\alpha) \neq 0$.

Example 5

Write N-R iterative scheme to find inverse of an integer number N . Hence, find inverse of 17 correct upto 4 places of decimal starting with 0.05.

Solution

Let inverse of N be x , so that we have the equation to solve as,

$$x = N^{-1} \quad \text{or} \quad x^{-1} - N = 0$$

$$f(x) = x^{-1} - N ; \quad f'(x) = -\frac{1}{x^2}$$

N-R scheme is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} = x_n + x_n^2 \left(\frac{1}{x_n} - N \right) \\ &= x_n (2 - Nx_n) = x_n (2 - 17x_n) \quad \because N = 17. \end{aligned}$$

We take $x_0 = 0.05$.

Substituting in the formula, we get

$$x_1 = 0.0575 ; \quad x_2 = 0.0588 ; \quad x_3 = 0.0588$$

$$\text{Hence, } \frac{1}{17} = 0.0588.$$

Example 6

Write down N-R iterative scheme for finding q^{th} root of a positive number N .

Hence, find cube root of 10 correct upto 3 places of decimal taking initial estimate as 2.0.

Solution

We have to solve $x = N^{\frac{1}{q}}$ or $x^q - N = 0$

$$f(x) = x^q - N ; \quad f'(x) = qx^{q-1}$$

The N-R iterative scheme may be written

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

For cube root of 10 we have $N = 10$, $q = 3$.

$$\text{Hence, } x_{n+1} = \frac{2x_n^3 + 10}{3x_n^2}$$

Taking $x_0 = 2.0$ we get the following iterated values

$$x_1 = \frac{16}{2} = 2.167 ; \quad x_2 = \frac{30.3520}{14.0877} = 2.154 ; \quad x_3 = \frac{29.9879}{13.9191} = 2.154$$

Hence, we get $10^{\frac{1}{3}} = 2.154$.

Example 7

Using N-R method find the root of the equation $x - \cos x = 0$ correct upto two places of decimal only. Take the starting value as $\frac{\pi}{4}$ ($\pi = 3.1416$, π radian = 180°).

Solution

$$f(x) = x - \cos x ; \quad f'(x) = 1 + \sin x$$

N-R scheme is given by

$$x_{n+1} = x_n - \frac{x_n - \cos x_n}{1 + \sin x_n} = \frac{x_n \sin x_n + \cos x_n}{1 + \sin x_n}$$

Taking $x_0 = \frac{\pi}{4}$

$$\begin{aligned}
 x_1 &= \frac{\frac{\pi}{4} \sin \frac{\pi}{4} + \cos \frac{\pi}{4}}{1 + \sin \frac{\pi}{4}} = \frac{\frac{\pi}{4} \times \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}}{1 + \frac{1}{\sqrt{2}}} \\
 &= \frac{\frac{\pi}{4} + 1}{\sqrt{2} + 1} = \frac{1.7854}{2.4142} = 0.7395 \\
 x_2 &= \frac{0.7395 \sin (0.7395) + \cos (0.7395)}{1 + \sin (0.7395)} \\
 &= \frac{0.7395 \times 0.6724 + 0.7449}{1 + 0.6724} = 0.7427
 \end{aligned}$$

Up to two places of decimal the root is 0.74.

Note : If starting value is not given, we can plot graphs of $y = x$ and $y = \cos x$ and locate their point of intersection which will be root of $x - \cos x = 0$. See Figure 3.4.

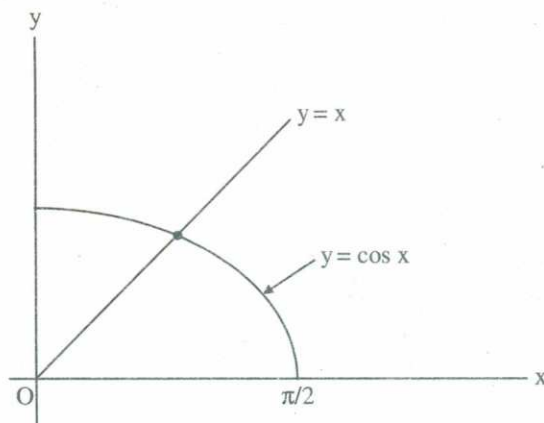


Figure 3.4 : Intersection of $y = x$ and $y = \cos x$

Ex. 9 : Using the Newton-Raphson method, find the square root of 10 with initial approximation $x_0 = 3$.

Ex. 10 : Starting with $x_0 = 0$, perform two iterations to find an approximate root of the equation $x^3 - 4x + 1 = 0$, using Newton-Raphson method.

3.7 SUMMARY

In this unit we have covered the following points :

The methods for finding an approximate solution of equation in one variable involve two steps :

- (i) Find an initial approximation to a root.
- (ii) Improve the initial approximation to get more accurate value of the root.

The following iterative methods have been discussed :

- (i) Bisection method
- (ii) Fixed point iteration method

- (iii) Regula-falsi method
- (iv) Secant method
- (v) Newton-Raphson method.

3.8 SOLUTIONS/ANSWERS

Ex. 1 : Here $\phi(x) = -\frac{1}{3x^2} (2 + 3x + 2x^2)$

$$|\phi'(x)| = \left| \frac{1}{3} \frac{(4 + 3x)}{x^3} \right| > 1 \text{ at } x_0 = -0.5$$

Hence iteration does not converge.

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

then $|\phi'(x)| = \left| -\frac{1}{3} (4x + 9x^2) \right| < 1 \text{ at } x_0 = -0.5$

Hence in this case iteration converges

First iteration $x_1 = -0.708$

Second iteration $x_2 = -0.646$

Ex. 2 : Root lies in $[0, 1]$. We take.

$$x = \frac{x^2 + 1}{3} = g(x)$$

$$g'(x) = \frac{2x}{3} \Rightarrow |g'(x)| < 1 \text{ in } [0, 1]$$

Starting with $x_0 = 0.5$, we have

$$x_1 = \frac{5}{12} = 0.417, x_2 = \frac{169}{432} = 0.391 \text{ and } x_3 = 0.384$$

Ex. 3 : $f(0) > 0$ and $f(1) < 0$. The smallest positive root lies in $]0, 1[$.

No. of Bisection	Bisected Value x_i	$f(x_i)$	Improved Interval
1	0.5	-1.375	$]0, 0.5[$
2	0.25	-0.09375	$]0, 0.25[$
3	0.125	0.37895	$]0.125, 0.25[$

It is enough to check the sign of $f(x_0)$ – the value need not be calculated.

The approximate value of the desired root is $0.1875 = (0.25 + 0.125)/2$.

Ex. 4 : Here $f(x) = 2x - 3 \sin x - 5$

x	0	1	2	2.5	2.8	2.9
f(x)	-5.0	-5.51224	-3.7278	-1.7954	-0.4049	0.0822

Thus a positive root lies in the interval $[2.8, 2.9]$.

No. of bisection	Bisected value x_0	$f(x_0)$	Improved Interval
1	2.85	-0.1624	[2.85, 2.9]
2	2.875	-0.0403	[2.875, 2.9]
3	2.8875	-0.02089	[2.875, 2.8875]
4	2.88125		

Ex. 5: $x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)} = \frac{(-8)9 - (-7)(-55)}{9 + 55} = 7.1406$

Similarly $x_3 = -7.168174$.

The iterated values are presented in tabular form below :

No. of Intersections	Interval	Bisected Value x_0	The function Value $f(x_i)$
1	$] -8, -7[$	-7.1406	1.862856
2	$] -8, -7.1406[$	-7.168174	0.358767
3			
4			
5			
6			

Complete the above table. You can find that the difference between the 5th and 6th iterated values is $|7.1748226 - 7.1747855| = 0.0000371$ signaling a stop to the iteration. We conclude that -7.175 is an approximate root rounded to the decimal places.

Ex. 6: $f(x) = 3x^3 - 4x^2 + 3x - 4$, $x_0 = 0$, $x_1 = 1$.

$$x_{n+1} = \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}, \quad n = 1, 2, 3$$

This gives $x_2 = 2$, $x_3 = 1.167$, $x_4 = 1.255$

Ex. 7: $f(x) = x^3 + x = 6$, $x_0 = 1$, $x_1 = 2$

$$x_{n+1} = \frac{x_{n-1} f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}, \quad n = 1, 2, 3, \dots$$

This gives $x_2 = 1.5$, $x_3 = 1.609 \approx 1.61$, $x_4 = 1.64$.

Ex. 8: Here $f(x) = \cos x - xe^x$, $x_0 = 0$ and $x_1 = 1$

$$x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)} = 0.3146653378$$

$$x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} = 0.4467281466$$

Ex. 9: $x = \sqrt{10}$, i.e. $x^2 = 10$. $f(x) = x^2 - 10$, $f'(x) = 2x$

$$x_{n+1} = x_n - \frac{x_n^2 - 10}{2x_n} = \frac{x_n^2 + 10}{2x_n}, \quad n = 0, 1, 2$$

$$x_0 = 3, \quad x_1 = \frac{19}{6} = 3.167, \quad x_2 = \frac{(3.167)^2 + 10}{6.334} = 3.162$$

Ex. 10 : Here $f(x) = x^3 - 4x + 1$, $x_0 = 0$.

Differentiating, we get $f'(x) = 3x^2 - 4$

The iteration formula is $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

i.e.
$$x_{i+1} = \frac{2x_i^3 - 1}{3x_i^2 - 4}$$

This gives $x_1 = 0.25$, $x_1 = 0.254095 \approx 0.2541$.

APPENDIX MORE DETAILED DISCUSSION OF NUMBERS AND DISASTERS, ETC.

The unit is an optional, though highly, desirable, reading. The unit is included for better understanding of number systems, which play significant role in understanding of the discipline of Numerical Techniques

Structure

- A.0 Introduction
- A.1 Sets of Numbers
- A.2 Algebraic Systems of Numbers
- A.3 Some Properties of Sets and Systems of Numbers
- A.4 Numerals : Notations for Numbers
- A.5 Essential Features of Computer Represented Numbers
- A.6 Disasters Due to Numerical Errors

A.0 INTRODUCTION

We have earlier mentioned that the discipline of Numerical Techniques is about

- numbers, rather special type of numbers called computer numbers, and
- application of (some restricted version of) the four arithmetic operations, viz., + (plus), - (minus), $\times$ (multiplication) and $\div$ (division) on these special numbers.

Therefore, let us, first, recall some important sets of numbers, which have been introduced to us earlier, from school days. Then we will discuss algebraic systems (**to be called simply systems**) of numbers, and finally notations for numbers:

Also, we enumerated a number of disasters caused by subtle numerical errors. Here, we briefly describe each of these.

A.1 SETS OF NUMBERS

Set of **Natural numbers** is denoted by **N**, where $N = \{0, 1, 2, 3, 4, \dots\}$ or $N = \{1, 2, 3, 4, \dots\}$.

Set of **Integers** is denoted by **I**, or **Z**, where $I = \{\dots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \dots\}$

Justification of the Kronecker's statement can be seen through the following explanation: An integer can be considered as an ordered pair (m, n) of natural numbers. For example, -3 may be considered as the pair $(2, 5)$, or $(4, 7)$ of natural numbers and integer 3 as $(5, 2)$, or $(7, 4)$. Further, operations on integers, in this representation, can be realized as :

$$-(m_1, n_1) = (n_1, m_1)$$

$$(m_1, n_1) + (m_2, n_2) = (m_1 + m_2, n_1 + n_2), (m_1, n_1) - (m_2, n_2) = (m_1, n_1) + (n_2, m_2) \\ = (m_1 + n_2, n_1 + m_2)$$

and $(m_1, n_1) \times (m_2, n_2) = (m_1 m_2 + n_1 n_2, n_1 m_2 + m_1 n_2)$ etc.

Similarly, members of sets like **Q** etc., discussed below can be structured directly or indirectly from **N** Natural numbers.

Set of **Rational Numbers** denoted by **Q**, where

$$Q = \{a/b, \text{ where } a \text{ and } b \text{ are integers and } b \text{ is not } 0\}$$

God made the
natural numbers,
rest made the man
Kronecker

Set of **Real Numbers** denoted by **R**. There are different ways of looking at or thinking of Real Numbers. *One of the intuitive ways of thinking of real numbers* is as the numbers that correspond to the points on a straight line extended infinitely in both the directions, such that one of the points on the line is marked as 0 and another point (different from, and to the right of, the earlier point) is marked as 1. Then to each of the points on this line, a unique real number is associated.

A more formal way is to consider the set of real numbers as extension of the rational numbers, where a real number is the limit of a convergent sequence of rational numbers. There is a large subset of real numbers, no member of which is a rational number. A real number which is not a rational number is called **irrational number**. For example, $\sqrt{2}$ is an irrational number.

Set of **Complex Numbers** denoted by **C**, where $C = \{a + bi \text{ or } a + ib \text{ where } a \text{ and } b \text{ are real numbers and } i \text{ is the square root of } -1\}$.

By minor notational modifications (e.g., by writing an integer, say, 4 as a rational number $4/1$; and by writing a real number, say $\sqrt{2}$ as a complex number $\sqrt{2} + 0i$), we can easily see that $N \subset I \subset Q \subset R \subset C$.

When we do not have any specific set under consideration, the set may be referred to as a *set of numbers*, and a member of the set as just *number*.

Apart from these well-known sets of numbers, there are sets of numbers that may be useful in our later discussion. Next, we discuss two such sets.

Set of algebraic Numbers (*no standard notation for the set*), where an *Algebraic number* is a number that is a root of a non-zero polynomial equation¹ with rational numbers as coefficients. For example,

- Every rational number is algebraic (e.g., the rational number a/b , with $b \neq 0$, is a root of the polynomial equation: $bx - a = 0$). Thus, a real number, which is not algebraic, must be irrational number.
- Even, some irrational numbers are algebraic, e.g., $\sqrt{2}$ is an algebraic number, because, it satisfies the polynomial equation: $x^2 - 2 = 0$. In general, n^{th} root of a rational number a/b , with $b \neq 0$, is algebraic, because, it is a root of the polynomial equation: $b \times x^n - a = 0$.
- Even, a complex number may be an algebraic number, as each of the complex numbers $\sqrt{2}i (= 0 + \sqrt{2}i)$ and $-\sqrt{2}i$ is algebraic, because, each satisfies the polynomial equation: $x^2 + 2 = 0$.

Set of Transcendental Numbers (*again, no standard notation for the set*), where, a *transcendental number* is a real/complex number which is not algebraic. From the above examples, it is clear that a rational number cannot be transcendental, and some, but not all, irrational numbers, may be transcendental. The most prominent examples of transcendental numbers are π and² e .

A.2 ALGEBRAIC SYSTEMS OF NUMBERS (TO BE CALLED, SIMPLY, SYSTEMS OF NUMBERS)

In order to discuss, system of numbers, to begin with, we need to understand the concept of *operation on a set* For this purpose, recall that **N**, the set of Natural numbers, is

¹ We may recall that a polynomial $P(x)$ is an expression of the form: $a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_{n-1} x + a_n$, where a_i is a number and x is a variable. Then, $P(x) = 0$ represents a polynomial equation.

² It should be noted that it is quite complex task to show a number as transcendental. In order to show a number, say, n , to be transcendental, theoretically, it is required to ensure that for *each* polynomial equation $P(x) = 0$, n is not a root of the equation. And, there are infinitely many polynomial equations. This direct method for showing a number as transcendental, cannot be used. There are other methods for the purpose.

closed under '+' (plus). By '*N is closed under +*', we mean : if we take (any) two natural numbers, say m and n , then $m + n$ is *also* a natural number.

But, N , the set of Natural numbers, is not closed under '-' (minus). In other words, *for some* natural numbers m and n , $m - n$ may not be a natural number, for example, for 3 and 5, $3 - 5$ is not a natural number. (Of course, $3 - 5 = -2$ is an integer).

These facts are also stated by saying : '+' is a binary operation on N , but, '-' is not a binary operation on N . Here, the word **binary** means that in order to apply '+', we need (exactly) two members of N .

In the light of above illustration of binary operation, we may recall many such statements including the following :

- (i) $\times$ (multiplication) is a binary operation on N (or, equivalently, we can say that N is closed under the binary operation $\times$)
- (ii) $-$ (minus) is a binary operation on I (or, equivalently, we can say that I is closed under the binary operation $-$) etc.³

However, there are operations on numbers, which may require only one number (*the number is called argument of the operation*) of the set. For example, The *square* operation on a set of numbers takes only one number and returns its square, for example, square (3) = 9.

Thus, some operations (e.g., square) on a set of numbers may take only one argument. Such operations are called **unary operations**. Other operations may take two arguments (e.g., $+$, $-$, $\times$, $\div$) from a set of numbers. Such operations are called **binary operations**.

There are operations which may take three arguments and are called **ternary operations**. There can be operations requiring no argument (e.g., multiplicative identity of the set of numbers) and there can be operations requiring 4 or more arguments. **But one of the defining characteristics of an operation on a set is that the result of the operation must be a unique member of the same set.**

Definition: Algebraic System of Numbers: A set of numbers, say, S , along with a (finite) set of operations on S , is called an algebraic system of numbers. Instead of 'algebraic system', we may use the word '**system**'.

Notation for a System : If $O_1, O_2, \dots, O_n$ are some n operations on a set S , then, we denote the corresponding system as $\langle S, O_1, O_2, \dots, O_n \rangle$, or as $(S, O_1, O_2, \dots, O_n)$.

Examples AND Non-examples of Systems of Numbers

1. Examples of number systems

Each of following is a system of numbers : $\langle N, + \rangle$, $\langle N, \times \rangle$, and $\langle N, +, \times \rangle$ etc.⁴

2. Non-examples of number systems

Each of following is NOT a system of numbers : $\langle N, - \rangle$, $\langle N, \div \rangle$, $\langle N, -, \div \rangle$ $\langle I, \div \rangle$ etc.⁵

³ (iii) $\times$ (multiplication) is a binary operation on I (or, equivalently, we can say that I is closed under the binary operation $\times$)

(iv) $\div$ (division) is neither a binary operation on N nor on I .

(v) $\div$ (division) is a binary operation on $Q - \{0\}$, and (vi) on $R - \{0\}$, and also (vii) on $C - \{0\}$.

⁴ Each of following is also a system of numbers: $\langle I, + \rangle$, $\langle I, - \rangle$, $\langle I, \times \rangle$, $\langle I, +, \times \rangle$, $\langle I, +, - \rangle$, $\langle I, +, -, \times \rangle$ and $\langle Q, + \rangle$, $\langle Q, - \rangle$, $\langle Q, \times \rangle$, $\langle Q, +, \times \rangle$, $\langle Q, +, - \rangle$, $\langle Q, +, -, \times \rangle$, $\langle Q - \{0\}, \div \rangle$, $\langle Q - \{0\}, +, \div \rangle$, $\langle Q - \{0\}, +, -, \times, \div \rangle$ and $\langle R, + \rangle$, $\langle R, - \rangle$, $\langle R, \times \rangle$, $\langle Q, +, \times \rangle$, $\langle R, +, - \rangle$, $\langle R, +, -, \times \rangle$, $\langle R - \{0\}, \div \rangle$, $\langle R - \{0\}, +, \div \rangle$, and $\langle R - \{0\}, +, -, \times, \div \rangle$ and $\langle C, + \rangle$, $\langle C, - \rangle$, $\langle C, \times \rangle$, $\langle C, +, \times \rangle$, $\langle C, +, - \rangle$, $\langle C, +, -, \times \rangle$, $\langle C - \{0\}, \div \rangle$, $\langle C - \{0\}, +, \div \rangle$, and $\langle C - \{0\}, +, -, \times, \div \rangle$.

⁵ Each of following is also NOT a system of numbers : $\langle Q, +, -, \times, \div \rangle$ and $\langle R, \div \rangle$, $\langle R, +, \div \rangle$, $\langle R, +, -, \times, \div \rangle$ and $\langle C, \div \rangle$, $\langle C, +, \div \rangle$, $\langle C, +, -, \times, \div \rangle$ because division by 0 is not defined.

Some more operations on Sets of numbers

Zero-ary Operations :

- (i) The numeral 1 is multiplicative identity for every number (i.e., $1 \times n = n$, for every number n). Thus '1 as multiplicative identity' may be treated as an operation. It is a zero-ary operation on each of N , I , Q , R and C . (because, we are not required to supply any number to know its multiplicative identity. On the other hand, to know the result on application of $+$, we must supply two numbers).
- (ii) The numeral 0 is additive identity for every number (i.e., $0 + n = n$, for every number n). Thus '0 as additive identity' may be treated as an operation. It is a zero-ary operation on each of N , I , Q , R and C .

Unary Operations :

- (i) We know *square* (let us denote it by Sq) of a natural number is also a natural Number. Thus, Sq is an operation on N . (As it requires only one number to return the answer, it is a unary operation.) Similarly, Sq is a unary operation on each of I , Q , R and C .
- (ii) **The square-rooting ($\sqrt{}$) is *not* an operation on N (because, $\sqrt{2}$ is not in N , whereas 2 is in N). Similarly, square-rooting is not an operation on I , Q , R (because, $\sqrt{-2}$ is not in I , also not in Q , and also not in R , therefore, it is not an operation on I , not an operation on Q , and it is not an operation on R).**

*But, square-rooting is an operation on C . Also, as it requires only one (complex) number to return the answer, it is a **unary** operation on C .*

- (iii) For any natural number $n \geq 2$, taking n^{th} root of a number ($^n\sqrt{}$) is *not* an operation on N (because, $^n\sqrt{2}$ is not in N , for 2 in N). Similarly, it is not an operation on I , Q , R (because, $^n\sqrt{-2}$ is not in I , also not in Q , and also not in R , therefore, it is not an operation on I , not an operation on Q , and it is not an operation on R).

*But, taking n^{th} root of a number ($^n\sqrt{}$) is an operation on C . Also, as it requires only one (complex) number to return the answer, it is a **unary** operation on C .*

3. Some more Examples of Number Systems

By adding any of one, two or more of the operations, viz., Sq , 1, 0, $\sqrt{}$, and $^n\sqrt{}$ to some of the number systems mentioned above, we may get a new number system.

For example, $\langle N, +, \times, 1, Sq \rangle$ is a number system....

Similarly, $\langle C \sim \{0\}, +, -, \times, \div, ^n\sqrt{} \rangle$ is a number system.

4. Some more Non-examples of Number Systems

However, each of following is **NOT** a system of numbers :

$$\langle N, +, ^n\sqrt{} \rangle, \langle N, \times, ^n\sqrt{} \rangle, \text{ and } \langle N, +, \times, ^n\sqrt{} \rangle \text{ etc.}^6$$

Apart from *operations*, various number systems have *relations*⁷, which also may be unary, binary etc. For, example, ' $<$ ' is a binary relation on $\langle N, +, \times \rangle$. Actually,

⁶ Each of following is also **NOT** a system of numbers :

$$\langle I, +, ^n\sqrt{} \rangle, \langle I, -, ^n\sqrt{} \rangle, \langle I, \times, ^n\sqrt{} \rangle, \langle I, +, \times, ^n\sqrt{} \rangle, \langle I, +, -, \times, ^n\sqrt{} \rangle \text{ and } \langle Q, +, ^n\sqrt{} \rangle, \langle Q, -, ^n\sqrt{} \rangle, \langle Q, \times, ^n\sqrt{} \rangle, \langle Q, -, \times, ^n\sqrt{} \rangle, \langle Q, -, -, ^n\sqrt{} \rangle.$$

⁷ An *operation*, say $+$, on a set of numbers, again say, N , takes two numbers and *returns* (i.e., gives an answer as) a number, in this case, an element of N . Similarly, a *relation*, which also may be unary, binary, ternary, etc, takes appropriate number of numbers (in the case of binary relation, it takes two) **but** returns

'<' is a binary relation on each of the number system discussed above, except systems on C, the set of complex numbers. Then, we can define 'the minimum element' or simply, 'the minimum' and 'the maximum element' or simply, 'the maximum' of a number system, in the usual sense of these terms.

More Details Discussion
of Numbers and
Disasters, etc.

A.3 SOME PROPERTIES OF SETS AND SYSTEMS OF NUMBERS

1. Each of the set N, I, Q, R and C is an *infinite* set.
2. None of the number systems discussed above, is bounded above, i.e., has the *maximum element*.
3. N has the *minimum element* (0, if, N is taken as {0, 1, 2, 3,} and 1, if N is taken as {1, 2, 3,}). But, *none* of the other number systems mentioned above has the minimum element.
4. The set of real numbers does not have the *least positive real number*. Because, between 0 and any positive real number, say r, lies the positive real number $r/2$ The same is true of rational numbers.
5. For each of the relevant number systems, mentioned above, on the sets N, I, Q, R, and C, each of the following holds
 - (i) '+' is Commutative in numbers, i.e., $x + y = y + x$, for any numbers x and y
 - (ii) '×' is Commutative in numbers, i.e., $x \times y = y \times x$, for any numbers x and y
 - (iii) '+' is Associative in numbers, i.e., $(x + y) + z = x + (y + z)$, for any numbers x, y and z
 - (iv) '×' is Associative in numbers, i.e., $(x \times y) \times z = x \times (y \times z)$, for any numbers x, y and z,

However,

- (v) '-' is NOT Associative in numbers, i.e., $(x - y) - z \neq x - (y - z)$, for some numbers x, y and z.

For example, $10 - (4 - 6) = 8 \neq 0 = (10 - 4) - 6$ and

- (vi) '÷' is NOT Associative in numbers, i.e., $(x \div y) \div z \neq x \div (y \div z)$, for some numbers x, y and z.

For example, $128 \div (8 \div 4) = 64 \neq 4 = (128 \div 8) \div 4$

- (vii) '×' is Left and Right Distributive over '+', i.e.,

(a) $x \times (y + z) = (x \times y) + (x \times z)$, for numbers x, y and z (*left*)

(b) $(y + z) \times x = (y \times x) + (z \times x)$, for numbers x, y and z (*right*)

(as, '×' is both left and right distributive over '+', we just say '×' is distributive over '+')

- (viii) '×' is Distributive over '-', i.e.,

(a) $x \times (y - z) = (x \times y) - (x \times z)$, for numbers x, y and z (*left*)

(b) $(y - z) \times x = (y \times x) - (z \times x)$, for numbers x, y and z (*right*)

(i.e., gives an answer as) 'True' or 'False'. For example, the relation of '<' takes two integers, say 3 and 5 and returns 'True', because, $3 < 5$ is True. However, if 7 and 5 are given as arguments, then it returns 'False', because, $7 < 5$ is false.

(ix) ' $\div$ ' is (only) right distributive over the operation '+', i.e.,

$$(y + z) \div x = (y \div x) + (z \div x), \text{ for some numbers } x, y \text{ and } z$$

(x) ' $\div$ ' is (only) right distributive over the operation '-', i.e.,

$$(y - z) \div x = (y \div x) - (z \div x), \text{ for numbers } x, y \text{ and } z.$$

However,

(xi) the operation ' $\div$ ' is NOT Left distributive over the operation '+', i.e.,

$$x \div (y + z) \neq (x \div y) + (x \div z), \text{ for some numbers } x, y \text{ and } z \text{ in } Q, R \text{ or } C$$

For example, $120 \div (4 + 6) = 12 \neq 50 = (120 \div 4) + (120 \div 6)$ and

(xii) the operation ' $\div$ ' is NOT Left distributive over the operation '-' i.e.,

$$x \div (y - z) \neq (x \div y) - (x \div z), \text{ for numbers } x, y \text{ and } z \text{ in } Q, R \text{ or } C$$

For example, $120 \div (4 - 6) = 60 \neq 10 = (120 \div 4) - (120 \div 6)$

Remark 1 : The above discussion in this subsection, of the properties of numbers is significant in the light of the fact that many of the above mentioned properties of numbers may not hold in the set of numbers that can be stored in a computer system.

Remark 2 : In the above discussion, the use of the word **number** is inaccurate. Actually, a number is a concept (a mental entity), which may be represented in some (physical) forms, so that we can experience the concept through our senses. The number, the name of which is, say, **ten** in English and **दस** in Hindi, and **zehn** in German language may be represented as 10 as decimal numeral, X as Roman numeral, 1010 as binary numeral. As, you may have already noticed, the physical representation of a number is called its **numeral**. Thus, number and numeral are two different entities, incorrectly taken to be the same. Also, a particular number is unique, but, it can have many (physical) representations, each being called a numeral, corresponding to the number.

The difference between **number** and **numeral** may be further clarified from the following explanation: We have the **concept** of the animal that is called **COW** in English, **गाय** in Hindi and **KUH** in German language. The animal, represented as cow in English, has four legs; however, its representation in English : **cow**, is a word in English and has three letters, but does not have four legs.

However, due to usage, though inaccurate, over centuries, instead of the word **numeral**, almost, the word **number** is used. Except for the discussion of Subsection A.4, we will also not differentiate between Number and Numeral.

A.4 NUMERALS : NOTATIONS FOR NUMBERS

First, we recall some well-known sets used to denote numbers. These sets are called *sets of numerals* and then discuss various *numeral systems*, developed on these numeral sets, for representing numbers.

A.4.0 Sets of Numerals : We are already familiar with some of the sets of numerals. The most familiar, and frequently used, set is **Decimal Numeral Set**. It is called *Decimal*, because, it uses ten figures, or digits, viz., *digits from the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} of ten digits*⁸.

⁸ Over a number of centuries now, mainly decimal number systems have been used to represent quantities/numbers. However, other number systems have been used and are still being used in special applications, e.g., base-12 systems (dozen = 12 and gross = 144, used still in purchase of, say, paper sheets and bananas); base-20 (score = 20); base-60 or sexagesimal (used in measure of time in terms of hour-minute-seconds). But, in general, we have a better understanding of the measure of a quantity, if it is expressed in decimal. For example, we understand a quantity better when written as 152 in decimal than when written as one gross and eight in base-12; seven score

By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.

.A.N. Whitehead

The quantity of meaning compressed into small space by algebraic signs, is another circumstance that facilitates the reasonings we are accustomed to carry on by their aid.

Charles Babbage

Another numeral set, familiar to computer science students, is **binary** numeral set. It is called binary, because, it uses two figures, or digits, viz., *digits from the set {0, 1} of two digits*. In this case, 0 and 1 are called *bits*.

Also, **Roman Numeral Set**, is well-known. This set uses figures/digits/letters from the set { I, V, X, L, C, D, M, ... }, where, I represents 1 (of decimal numeral system), V represents 5, X represents 10, L represents 50, C represents 100, D represents 500 and M represents 1000. etc.⁹

Apart from these sets of numerals, in context of computer systems, we also come across

- (i) **Hexadecimal numeral set**, which uses figures/digits, viz., from the set : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E} of sixteen digits.
- (ii) **Octal numeral set**, which uses figures/digits, viz., from the set : {0, 1, 2, 3, 4, 5, 6, 7} of eight digits.
- (iii) **Ternary numeral set**, which uses figures/digits, viz., from the set : {0, 1, 2} of three digits.
- (iv) **Base/radix r numeral set**, where r is a natural number, which uses figures/digits, viz., from the set : {0, 1, 2, ..., r-1} of r digits.

Except for set of Roman numerals, all other sets of numerals may be considered as set of radix r numerals, with r = 2 for binary, r = 8 for octal, r = 10 for decimal and r = 16 for hexadecimal.

A.4.1 Number Representation using a Set of Numerals : a number is represented by a *string of digits* from the set of numerals under consideration, e.g., 3426 in decimal, IX in Roman 10010110 in binary and 37A08 in hexadecimal.

A.4.2 Value of the Number Denoted by a String¹⁰ : Using either of numeral sets introduced in 1.0.4.0, there are **different schemes, i.e., sets of rules for interpreting a string as a number**.

For example, the string '4723', according to *usual decimal system*, represents the number : $4 \times 10^3 + 7 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$. Also, the string 'CLVII', according to the *Roman system*, represents the (decimal) number : $100 + 50 + 5 + 1 + 1 = 157$ (decimal), where C

and twelve in base-20; or 10011000 in binary. Generally, in order to have proper idea of the quantity, we convert the representation, if in other base, to base-10.

The decimal number systems (i.e., base-10) have become intuitive, quite natural to us, the human beings.

⁹Appropriate choice of numeral system has significant role in solving problems, particularly solving problems *efficiently*. For example, it is a child's play to get the answer for 46×37 (in decimal numeral system) as a single number. However, using Roman numerals, i.e., writing VLI $\times$ XXXVII, instead of , 46×37 , it is really very difficult to get the same number, using only Roman numerals, as answer.

¹⁰ For finding the value of a string of digits, at the top level, the numeral systems may be divided into two classes :

- *positional numeral systems and*
- *non-positional numeral systems.*

Most of the numeral systems are positional. In the **positional system**, the position of a digit determines the value contributed to the number by the digit. For example, in the decimal numeral representation 4723, the digit 4, because of its position, contributes 4000, the digit 7 contributes 700 and so on.

The Roman numeral system is a well-known non-positional numeral system. In a non-positional numeral system, the contribution of a digit to the value of the number does not depend on its position in the string of the digits. For example, in the Roman numeral, I always contributes 1, V contributes 5, X contributes 10, L contributes 50 etc. to the value of the number represented. Thus, LXI, represents 61, with L contributing 50, X contributing 10 and I contributing 1. **However, even the set of Roman numerals is not pure non-positional system**: If x and y are two Roman digits such that the digit x represents a number less than a number represented by the digit y, then the numbers represented by the strings xy and the strings yx are different. We know IX represents (decimal) 9 and XI represents (decimal) 11.

denotes 100, L denotes 50, V denotes 5 and I denotes 1. Similarly, the *binary string* 10010110 may be interpreted as the number (with value in decimal) :
 $1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$.

A.4.3 Notation for Base/Radix in Number Representation : From a string of digits, by itself, it may not be clear whether it is string of binary, octal, decimal, or hexadecimal digits. For example, the string 10010110 may be equally considered as a string of binary, octal, decimal, or hexadecimal digits. Similarly, the string 4607542 may be equally considered as a string of octal, decimal, or hexadecimal digits.

Thus the same string **10010110 may determine the decimal number**

- $1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$,
if the string is assumed to be **binary**,
- $1 \times 10^7 + 0 \times 10^6 + 0 \times 10^5 + 1 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 1 \times 10^1 + 0 \times 2^0$,
if the string is assumed to be **decimal** etc.¹¹.

Therefore, in order to avoid confusion about which numeral set a string belongs to, a suffix in the following manner is used

- (string)₂ for binary, e.g., string 10010110, if treated as binary will be denoted as (10010110)₂.
- (string)₁₀ for decimal, e.g., string 10010110, if treated as decimal will be denoted as (10010110)₁₀ etc.¹².

However, if there is no possibility of confusion, the suffix is not used.

A.4.4 A set of numerals along with a scheme for interpreting a sequence of digits as number, is called a *numeral system (or slightly incorrectly, a number system)*. You may notice that the decimal system and Roman system use different ways/ schemes for getting a number from a string of digits.

It may be noted that even for the same set of numerals, there may be different schemes/rules of interpretation, with different schemes giving different values for a given string. For example, for interpreting a binary string as a number, there are two well-known schemes :

- Fixed-point, and
- Floating-point.

These schemes have been discussed in detail in Unit 1.

The number systems, that we will discuss, are positional number systems based **on only decimal and binary**. However, with minor modifications, the discussion can be generalized to r-radix set of numerals. These numeral systems will be called, though slightly incorrectly, as number systems

¹¹ The same string **10010110 may determine the decimal number**

$$1 \times 8^7 + 0 \times 8^6 + 0 \times 8^5 + 1 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 1 \times 8^1 + 0 \times 8^0,$$

if the string is assumed to be **octal**, and as

$$1 \times 16^7 + 0 \times 16^6 + 0 \times 16^5 + 1 \times 16^4 + 0 \times 16^3 + 1 \times 16^2 + 1 \times 16^1 + 0 \times 16^0,$$

if the string is assumed to be **hexadecimal**.

¹² To indicate numeral system used for a given string, a suffix in the following manner is used : (string)₈ for octal, e.g., string 10010110, if treated as octal will be denoted as (10010110)₈ and as(string)₁₆ for hexadecimal, e.g., string 10010110, if treated as hexadecimal will be denoted as (10010110)₁₆.

A.5 ESSENTIAL FEATURES OF COMPUTER REPRESENTED NUMBERS

More Details Discussion
of Numbers and
Disasters, etc.

As mentioned earlier, not all real numbers can be represented in a computer system. The numbers that can be represented in a computer system will be called *computer numbers*, computer-represented numbers or sometimes, as *computer represent-able numbers* also.

Here, we mention some of the essential features of these numbers, specially, with respect to and in comparison with the properties of the number systems discussed in the subsection above.

- (1) A computer number is necessarily a binary number, i.e., it is a (finite) string of only 0's and 1's. (in this case, 0 and 1 are called *bits*). However, a particular string of bits may represent different numbers according to different schemes, i.e., sets of rules for interpreting a string as a number. (*to be discussed in more detail later*).
- (2) There is no unique set of computer represent-able numbers. The numbers that can be represented in a computer depend on the computer system under consideration.... The numbers that can be represented in a computer system depend on the word size of the computer system and the scheme of representation used.
- (3) No real number, which is irrational number, can be represented exactly in any computer system.... Only finitely many real numbers, each of which must also be rational, can be computer represent-able¹³.
- (4) Each of the other real numbers, when required to be stored in a computer, is approximated appropriately, to a computer number (of the computer system).
- (5) Whatever may be the computer system under consideration, the number of computer represent-able numbers, though substantially very large, is *finite* only. Therefore, *not* even all natural numbers (and, hence, all integers, all rational numbers and all real numbers) can be represented in a computer system.
- (6) Computer represent-able numbers have minimum positive computer number.... However, each computer system has its own unique minimum positive computer represent-able number. This number is generally called *machine-epsilon* of the computer system.
- (7) **Computer number zero is not the same as real number Zero :** If x is any real number such that $|x|$, if after rounding, is less than machine epsilon, say, ϵ , then x is represented by zero. Thus, computer zero represents not a single real number 0, but all the infinitely many real numbers of an interval contained in $] - \epsilon, \epsilon [$, and as ϵ , and hence, the computer number zero also varies, from computer to computer.
- (8) Each computer system has its own unique maximum *positive* computer represent-able number. The number depends on the word size and the scheme of representation used.
- (9) We elaborate further, the statement under point 1 above: *A computer number is necessarily a binary number, i.e., it is a string of only 0's and 1's. However, a particular string of bits may represent different numbers according to different schemes, i.e., sets of rules for interpreting a string as a number.*

¹³ But, as mentioned above, even not all rational numbers are computer represent-able. For example, $1/3$ is a rational number, which cannot be represented as a finite binary string, and hence, is not a computer number. Further, it may be noted that some rational numbers which can be represented as a finite string of *decimal* digits, may not be written as a finite string of bits. For example, $1/5$ can be written as: 0.2, a finite decimal string, but can be written only as an infinite *binary* string: 0.00110011.....

The schemes for interpreting a string as a number, at the top may be categorized into two major classes : (i) Fixed point representation and (ii) Floating point representation schemes.

(The Floating point representation scheme has already been discussed in detail in Unit 1.)

Further, *the Fixed point representation class has a number of schemes including :* (a) binary (b) BCD (Binary Coded Decimal) (c) Excess-3 (d) Gray code (e) signed magnitude, (f) signed 1's complement and (g) signed 2's complement representation schemes, and some combinations of these.

Similarly, *the Floating point representation scheme may associate different numbers to a particular string of bits, according to* (i) how the string is considered as composed of two parts, viz., mantissa and exponent, and (ii) the choice of the base and the choice of the bias or characteristic.

A.6 DISASTERS DUE TO NUMERICAL ERRORS¹⁴

In Section 0.1, we have already explained through examples that while adapting a mathematical solution for execution on a computer, we have to be *perennially aware* that each (specific) computer requires the computer-specific adapting of a mathematical solution. Forgetting these facts, even momentarily, have led to a number of *disasters*. *In order to emphasize the need for utmost care in this respect, we briefly discuss below some such well known disasters due to numerical errors.*

1. Patriot Missile Failure

On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Saudi Arabia failed to intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks and killed 28 soldiers. A report of the General Accounting office, GAO/IMTEC-92-26, entitled *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia* reported on the cause of the failure. It turns out that the cause was an inaccurate calculation of the time since boot due to computer arithmetic errors. Specifically, the time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds. **This calculation was performed using a 24 bit fixed point register. In particular, the value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits after the radix point. The small chopping error, when multiplied by the large number giving the time in tenths of a second, lead to a significant error.**

In other words, the binary expansion of 1/10 is .000110011001100110011001100.... Now the 24 bit register in the Patriot stored instead 0.0001100110011001100 introducing an error of 0.0000000000000000000000011001100... in binary, or about 0.000000095 in decimal. Multiplying by the number of tenths of a second in 100 hours gives $0.000000095 \times 100 \times 60 \times 60 \times 10 = 0.34$.) A Scud travels at about 1,676 meters per second, and so travels more than half a kilometer in this time. This was far enough that the incoming Scud was outside the "range gate" that the Patriot tracked. Ironically, the fact that the bad time calculation had been improved in some parts of the code, but not all, contributed to the problem, since it meant that the inaccuracies did not cancel.

2. Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after lift-off. The rocket was on its first voyage, after a decade of development costing \$7 billion. The destroyed rocket and its cargo were valued at \$500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that **the cause of the failure was a software error in**

¹⁴ The instances have been taken on Aug, 22, 2013 from the site :
<http://ta.twi.tude.nl/users/vuik/wi211/disasters.html>

the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,768, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

3. Rounding Error changes Parliament Makeup

A shattering computer error was experienced during a German election. The elections to the parliament for the state of Schleswig-Holstein were affected. German elections are quite complicated to calculate. First, there is the 5% clause : no party with less than 5% of the vote may be seated in parliament. All the votes for this party are lost. Seats are distributed by direct vote and by list. All persons winning a precinct vote (i.e. having more votes than any other candidate in the precinct) are seated. Then a complicated system (often D'Hondt, now they have newer systems) is invoked that seats persons from the party lists according to the proportion of the votes for each party. Often quite a number of extra seats (and office space and salaries) are necessary so that the seat distribution reflects the vote percentages each party got.

When the votes were being counted, initially it looked like the Green party was hanging on by their teeth to a vote percentage of exactly 5%. This meant that the Social Democrats (SPD) could not have anyone from their list seated, which was most unfortunate, as the candidate for minister president was number one on the list, and the SPD won all precincts: no extra seats needed.

After midnight (and after the election results were published) someone discovered that the Greens actually only had 4.97% of the vote. *The program that prints out the percentages only uses one place after the decimal, and had rounded the count up to 5%!* This software had been used for years, and no one had thought to turn off the rounding at this very critical region!

So 4.97% of the votes were thrown away, the seats were re-calculated, the SPD got to seat one person from the list, and now have a one seat majority in the parliament.

Reported by Debora Weber-Wulff, 7 Apr 1992.

4. The Sinking of the Sleipner A : An Offshore Platform

The Sleipner A platform produces oil and gas in the North Sea and is supported on the seabed at a water depth of 82 m. It is a Condeep type platform with a concrete gravity base structure consisting of 24 cells and with a total base area of 16 000 m². The concrete base structure for Sleipner A sprang a leak and sank under a controlled ballasting operation during preparation for deck mating in Gandsfjorden outside Stavanger, Norway on 23 August 1991.

A committee for investigation into the accident was constituted. The conclusion of the investigation was that the loss was caused by a failure in a cell wall, resulting in a serious crack and a leakage that the pumps were not able to cope with.

When the first model sank in August 1991, the crash caused a seismic event registering 3.0 on the Richter scale, and left nothing but a pile of debris at 220m of depth. The failure involved a total economic loss of about \$700 million.

The post accident investigation *traced the error to inaccurate finite element approximation of the linear elastic model* of the tricell (using the popular finite element program NASTRAN). The shear stresses were underestimated by 47%, leading to insufficient design. In particular, certain concrete walls were not thick enough. More careful finite element analysis, made after the accident, predicted that failure would occur with this design at a depth of 62m, which matches well with the actual occurrence at 65m. This description is adapted from *The sinking of the Sleipner A offshore platform* by Douglas N. Arnold.

MPDD-IGNOU/P.O.2.5T/MAR., 2015 (Reprint)

ISBN: 978-81-266-6566-2