NEERAJ®

NUMERICAL AND STATISTICAL COMPUTING

Reference Book Including Question Papers

By:

Ranveer Kumar

M.Sc.

New Edition

NEERAJ PUBLICATIONS

(Publishers of Educational Books)

(An ISO 9001 : 2008 Certified Company)

1507, Ist Floor, NAI SARAK, DELHI - 110 006

Ph. 23260329, 69414105 Fax 011-23285501

E-mail: info@neerajbooks.com *Website:* www.neerajbooks.com

Price: Rs. 120.00

Published by:

NEERAJ PUBLICATIONS

Admn. Office : **Delhi-110 007**
Sales Office : **1507, 1st Floor, Nai Sarak, Delhi-110 006**
E-mail: **info@neerajbooks.com** *Website*: **www.neerajbooks.com**

Typesetting by: ***Competent Computers*** *Printed at:*

Note:

1. *For the best & upto-date study & results, please prefer the recommended textbooks/study material only.*
2. *This book is just a Guide Book/Reference Book published by NEERAJ PUBLICATIONS based on the suggested syllabus by a particular Board /University.*
3. *The information and data etc given in this Book are from the best of the data arranged by the Author, but for the complete and upto-date information and data etc see the Govt of India Publications/textbooks recommended by the Board/University.*
4. *Publisher is not responsible for any omission or error though every care has been taken while preparing, printing, composing and proof reading of the Book. As all the Composing, Printing, Publishing and Proof Reading etc. are done by Human only and chances of Human Error could not be denied. If any reader is not satisfied, then he is requested not to buy this book.*
5. *In case of any dispute whatsoever the maximum anybody can claim against NEERAJ PUBLICATIONS is just for the price of the Book.*
6. *If anyone finds any mistake or error in this Book, he is requested to inform the Publisher, so that the same could be rectified and he would be provided the rectified Book free of cost.*
7. *The number of questions in NEERAJ study materials are indicative of general scope and design of the question paper. However, the actual question paper might somewhat vary in its contents, distribution of questions and their level of difficulty.*
8. *Sample Question Paper given in this Book provides you just the approximate pattern of the actual paper and is prepared based on the memory only.*
9. *Subject to Delhi Jurisdiction only.*

How to get Books by V.P.P.?

If you want to buy **NEERAJ® BOOKS** through V.P.P. then you can order the same at the Online Order Form of **www.neerajbooks.com** or send us an E-mail at **info@neerajbooks.com** or you may send us a Fax at **011-23285501** or write us a letter alongwith Rs. 100/- as advance by Bank Draft mentioning your complete requirement, class, course, subject code, medium, name, address and telephone no. which will help us in sending your required **NEERAJ® BOOKS** to you at the earliest.

23260329
69414105
Fax: 011-23285501

NEERAJ PUBLICATIONS

(Publishers of Educational Books)

(An ISO 9001 : 2008 Certified Company)

1507, 1st Floor, NAI SARAK, DELHI - 110 006

E-mail: **info@neerajbooks.com** *Website:* **www.neerajbooks.com**

CONTENTS

Numerical and Statistical Computing

Sample Preview
of the
Solved
Sample Question
Papers

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

QUESTION PAPER

(June – 2010)

(Unsolved)

NUMERICAL AND STATISTICAL COMPUTING

Time: 3 hours] *[Maximum Marks: 100*

Note: *Question number **1** is **compulsory.** Attempt **any three** questions from the rest. Use of calculator is allowed.*

1. *(a)* Estimate the relative error in $z = x - y$ when $x = 0.1234 \times 10^4$ and $y = 0.1232 \times 10^4$ as stored in a system with four-digit mantissa.

(b) Show that the series $e^x = 1 + x + \frac{x^2}{2!} + ...$ becomes unstable when $x = -10$.

(c) Find the root of the equation $x^x + x - 4 = 0$ using the Newton-Raphson method correct to four decimal places.

(d) The observed values of a function are respectively 168, 120, 72 and 63 at the four positions 3, 7, 9 and 10 of the independent variable. What is the best estimate you can give of the value of the function at the position 6 of the independent variable. Apply Lagrange's formula.

(e) The table gives the distance in nautical miles of the visible horizon for the given heights in feet above the earth's surface.

x = height	100	150	200	250	300	350	400
y = distance	10.63	13.03	15.04	16.81	18.42	19.90	21.27

Find the value of y when $x = 410$ using Newton's Backward Interpolation formula.

(f) Five men in a group of 20 are graduates. If 3 men are picked out of 20 at random *(i)* what is the probability that all are graduates and *(ii)* what is the probability of at least one being graduate?

2. *(a)* Find the root of the equation $x\,e^x = \cos x$ using the Secant method correct to four decimal places.

(b) Evaluate $\int_1^2 \log x$ by Trapezoidal rule.

(c) A book contains 100 misprints distributed randomly throughout its 100 pages. What is the probability that a page observed at random contains atleast two misprints.

3. *(a)* Solve the system of equations:

$4x_1 + x_2 + x_3 = 2$

$x_1 + 5x_2 + 2x_3 = -6$

$x_1 + 2x_2 + 3x_3 = -4$

Using Jacobi iteration method.

(b) Use Euler method to solve numerically the initial value problem.

$\upsilon' = -2t\,\upsilon^2,\ \upsilon\ (0) = 1$

with $h = 0.2$ and 0.1 on the interval [0, 1].

Or

A sample of 100 dry battery cells tested to find the length of life produced the following results:

$\overline{X} = 12$ hours, $\sigma = 3$ hours

Assuming the data to be normally distributed, what percentage of battery cells are expected to have life:

(i) More than 15 hours

(ii) Between 10 and 14 hours

$$\begin{bmatrix} \text{Given} & Z: & 2.5 & 2 & 1 & 0.67 \\ & \text{Area:} & 0.4938 & 0.4772 & 0.3413 & 0.2487 \end{bmatrix}$$

4. *(a)* Show that the LU decomposition method fails to solve the system of equations:

$$\begin{bmatrix} 1 & 1 & -1 \\ 2 & 2 & 5 \\ 3 & 2 & -3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 6 \end{bmatrix}$$

Exact solution is $x_1 = 1, x_2 = 0, x_3 = -1$.

Or

Apply Runge-Kutta method to find approximate value of y for $x = 0.2$, in steps of 0.1, if $\frac{dy}{dx} = x + y^2$, given that $y = 1$ where $x = 0$.

(b) A problem in statistics is given to five students A, B, C, D and E. Their chances of solving it are $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ and $\frac{1}{6}$. What is the probability that the problem will be solved?

5. *(a)* Perform five iterations of the bisection method to obtain the smallest positive root of the equation $f(x) = x^3 - 5x + 1 = 0$.

(b) With the help of Newton's forward difference interpolation formula obtain the interpolating polynomial satisfying the data.

x	1	2	3	4
$f(x)$	26	18	4	1

If a point $x = 5$, $f(x) = 26$, is added to above data, will the interpolation polynomial change? Explain.

(c) What is a random variable? Write down the expression which define Binomial, Poisson and Normal probability distribution. Give two physical situation illustrating a poisson random variable.

■■

Sample Preview
of
The Chapter

Published by:

NEERAJ
PUBLICATIONS
www.neerajbooks.com

NUMERICAL AND STATISTICAL COMPUTING

Computer Arithmetic

1

INTRODUCTION

Computer arithmetic is the mathematical theory which underlies the way calculation machines operate on integer numbers.

Computers manipulate *integer numbers* of a finite, fixed precision, internally represented as strings of bits of fixed length. A processor's hardware [Braun, 1963] is built to perform additions, multiplications, and other standard arithmetical operations along with logical operations like "or", "and", "not", "exclusive or", and so on.

The distinguishing features of computer arithmetic are:

- Logical operations, i.e. the ability to calculate bit per bit the conjunction, disjunction and negation of *integers.* For clarity, since we deal with a logical theory, we will refer to these operations with the adjective *bitwise.*
- Fixed precision, which means every representable number lies in a fixed, well-defined range of values and every operation must signal exceptions, when unable to provide a result which fits into that range, usually by means of carry and/or overflows bits.

The main features of a computer which influence the fomulation of algorithms are:

(i) The algorithm is stored in the memory of the computer. This facilitates the repetitive execution of instructions.

(ii) Results of computation may be stored in the memory and retrieved when necessary for further computation.

(iii) The sequence of execution of instructions may be altered based on the results of computation. The facility to test the sign of a number or test if it is zero coupled with the presence of the entire algorithm in the computer's memory enables alternate routes to be taken during the execution of the algorithm.

(iv) The computer has the capability to perform only the basic arithmetic operations of addition, subtraction, multiplication and division. In formulating algorithms all other mathematical operations should be reduced to these basic operations.

To summarise, in order to solve a mathematical problem (like say the solution of differential equations) on a computer, a step-by-step procedure utilising the above characteristics of a computer should be evolved.

In particular, it should be observed that only the elementary arithmetic operations may be used even when solving problems involving the operations of calculus (like differentiation and integration). Formulation of such algorithms is the main subject-matter of *numerical analysis.*

After going through this chapter, we will be able to define computer arithmetic includes Floating-Point Arithmetic and Errors, Floating-Point Representation of Numbers, Sources of Errors, Non-Associativity of Arithmetic, Propagated Errors, some pitfalls in computation, loss of significant digits, instability of algorithms.

CHAPTER AT A GLANCE

FLOATING-POINT ARITHMETIC AND ERRORS

Here we first start with floating-point representation of numbers.

Floating-point Representation of Numbers

In general, we are using two types of numbers in calculations

(a) Integers 1, …– 3, – 2, – 1, 0, 1, 2, 3,….. and

(b) Other real numbers, such as numbers with decimal point.

Scientists and engineers have developed a compact notation for writing very large or very small numbers. If we wrote it out, the mass of the Sun in grams would be a two followed by 33 zeroes. The speed of light in metres per second would be a three followed by eight zeroes. These same numbers, when expressed in scientific notation are 2×10^{33} and 3×10^{8}. Any number n can be expressed as $n = f \times 10^{e}$.

Where f is a fraction and e is an exponent. Both f and e may be negative. If f is negative the number n is negative. If e is negative, the number is less than one.

The essential idea of scientific notation is to separate the significant digits of a number from its magnitude. The number of significant digits is determined by the size of f and the range of magnitude is determined by the size of e.

We wrote the speed of light as 3×10^{8} metres per second. If that is not precise enough, we can write 2.997×10^{8} to express the same number with four digits of precision.

Floating-Point Numbers: Definition Floating-point number systems apply this same idea–separating the significant digits of a number from its magnitude–to represent numbers in computer systems.

Relatively small numbers for the fraction and exponent part provide a way to represent a very wide range with acceptable precision.

An n-digit floating-point number in base β (a given natural number), has the form $x = \pm (.d_1 d_2 \ldots d_n)_\beta \; \beta^e$, $0 \leq d_i < \beta$, $m \leq e \leq M$; I = 1, 2, …n, $d_1 \neq 0$; where, $(.d_1 d_2 \ldots d_n)_\beta$ is a β-fraction called mantissa and its value is given by $(.d_1 d_2 \ldots d_n)_\beta = d_1 \times 1/\beta + d_2 \times 1/\beta^2 + \;\; + d_n \times 1/\beta^n$; e is an integer called exponent.

The exponent e is also limited to range $m < e < M$, where m and M are integers varying from computer to computer. Usually, $m = -M$.

In IBM 1130, $m = -128$ (in binary), – 39 (decimal) and M = 127 (in binary), 38 (in decimal). For most of the computers β = 2 (binary), on same computers β = 16 (hexadecimal) and in pocket calculators β = 10 (decimal).

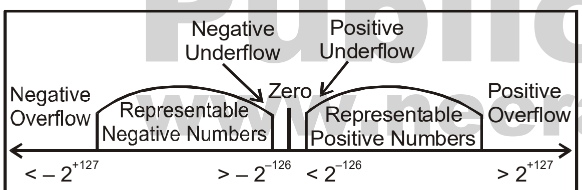
Normalized Numbers: We represented the speed of light as 2.997×10^{8}. We could also have written 0.2997×0^{9} or 0.02997×10^{10}. We can move the decimal point to the left, adding zeroes as necessary, by increasing the exponent by one for each place the decimal point is moved. Similarly, we can compensate for moving the decimal point to the right by decreasing the exponent. However, if we are dealing with a fixed-size fraction part, as in a computer implementation, leading zeroes in the fraction part cost precision. If we were limited to four digits of fraction, the last example would become 0.0299×10^{10}, a cost of one digit of precision. The same problem can occur in binary fractions. In order to preserve as many significant digits as possible, floating-point numbers are stored such that the leftmost digit of the fraction part is non-zero. If, after a calculation, the leftmost digit is not significant (i.e. it is zero), the fraction is shifted left and the exponent decreased by one until a significant digit, for binary numbers, a one, is present in the leftmost digit. A floating-point number in that form is called a normalised number. There are many possible unnormalised forms for a number, but only one normalised form.

Representation of Real Numbers in the Computers: Although the range of a single-precision floating-point number is $\pm 10^{-38}$ to $\pm 10^{38}$, it is important to remember that there are still only 2^{32} distinct values.

The floating-point system cannot represent every possible real number. Instead, it approximates the real numbers by a series of points. If the result of a calculation is not one of the numbers that can be represented exactly, what is stored is the nearest number that can be represented. This process is called *rounding,* and it introduces error in floating-point calculations. Since rounding down is as likely as rounding up, the cumulative effect of rounding error is generally negligible.

The spacing between floating-point numbers is not constant. Clearly, the difference between 0.10×2^1 and 0.11×2^1 is far less than the difference between 0.10×2^{127} and 0.11×2^{127}. If the difference between numbers is expressed as a percentage of the number, the distances are similar throughout the range, and the relative error due to rounding is about the same for small numbers as for large numbers.

Not only cannot all real numbers be expressed exactly, there are whole ranges of numbers that cannot be represented. Consider the real number line as shown in the Figure. The number zero can be represented exactly because it is defined by the standard. The positive numbers that can be represented fall approximately in the range 2^{-126} to 2^{+127}.



Zones of floating-point numbers along the real number line

Numbers greater than 2^{+127} cannot be represented; this is called *positive overflow.* A similar range of negative numbers can be represented. Numbers to the left of that range cannot be represented; this is negative overflow. There is also a range of numbers near zero that cannot be represented.

The smallest positive number that can be represented in normalised form is 1.0×2^{-126}. The condition of trying to represent smaller numbers is called *positive underflow.* The same condition on the negative side of zero is called *negative underflow.*

Rounding: Definition: A number x is rounded to t digits when x is replaced by a t digit number that approximates x with minimum error.

Example: $t = 5$, $x = 2.5873892874$ then rounding will be 2.5874.

Chopping: Definition: A number is chopped to t digits and all the digits past t are discarded.

Example: $t = 5$, $x = 2.5873892874$ then chopping will be 2.5874.

Overflow occurs when the result of a floating-point operation is larger than the largest floating-point number in the given floating-point number system.

When this occurs, almost all computers will signal an error message.

Underflow occurs when the result of a computation is smaller than the smallest quantity the computer can store.

Some computers don't see this error because the machine sets the number to zero.

Relative Error: Let $fl(x)$ be floating-point representation of real number x. Then $e_x = |x - fl(x)|$ is called *round-off* (absolute error).

$r_x = \dfrac{x - fl(x)}{x}$ is called the relative error.

Theorem: If $fl(x)$ is the n-digit floating-point representation in base β of a real number x, then r_x the relative error in x satisfies the following:

(i) $|r_x| < 1/2\ \beta^{1-n}$ if rounding is used

(ii) $0 \leq |r_x| \leq \beta^{1-n}$ if chopping is used

Proof: Case 1. $d_{n+1} < ½\ \beta$, then

$$fl(x) = \pm(.d_1\, d_2 \ldots d_n)\ \beta^e$$

$$|x - fl(\mathrm{x})| = d_{n+1}, d_{n+2} \ldots\ldots \beta^{e-n-1}$$

$$\leq ½\ \beta.\beta^{e-n-1} = ½\ \beta^{e-n}$$

Case 2. $d_{n+1} < ½\ \beta$, then

$$fl(x) = \pm\{(.d_1\, d_2 \ldots. d_n)\ \beta^e + \beta^{e-n}\}$$

$$|x - fl(x)| = .|- d_{n+1}, d_{n+2}.\ \beta^{e-n-1} + \beta^{e-n}|$$

$$= \beta^{e-n-1}\,|\,d_{+1}.\ d_{n+2} - \beta|$$

$$\leq \beta^{e-n-1} \times 1/2\ \beta = ½\ \beta^{e-n}.$$

Sources of Errors

In scientific computing, we never expect to get the exact answer. Inexactness is practically the definition of scientific computing. Getting the exact answer, generally with integers or rational numbers, is symbolic computing, an interesting but distinct subject. Suppose we are trying to compute the number A. The computer will produce an approximation, which we call Â. This Â may agree with A to 16 decimal places, but the identity

A = Â (almost) never is true in the mathematical sense, if only because the computer does not have an exact representation for A. For example, if we need to find x that satisfies the equation $x^2 - 175 = 0$, we might get 13 or 13.22876, depending on the computational method, but $\sqrt{175}$ cannot be represented exactly as a floating point number.

Four primary sources of error are:

(i) Round off error,

(ii) Truncation error,

(iii) Termination of iterations, and

(iv) Statistical error in Monte Carlo.

We will estimate the sizes of these errors, either a priori from what we know in advance about the solution, or a posteriori from the computed (approximate) solutions themselves. Software development requires distinguishing these errors from those caused by outright bugs. In fact, the bug may not be that a formula is wrong in a mathematical sense, but that an approximation is not accurate enough.

Scientific computing is shaped by the fact that nothing is exact. A mathematical formula that would give the exact answer with exact inputs might not be robust enough to give an approximate answer with (inevitably) approximate inputs. Individual errors that were small at the source might combine and grow in the steps of a long computation. Such a method is unstable. A problem is ill conditioned if any computational method for it is unstable. Stability theory, which is modelling and analysis of error growth, is an important part of scientific computing.

Generated Errors: Generated error reflects inaccuracies due to necessity of rounding or otherwise truncating the numeric results of arithmetic operations, inherent error reflects inaccuracies in initially given arguments and parameters, and analytic error reflects inaccuracies due to the use of a computing procedure which calculates only an approximation to the theoretical result desired.

During an arithmetic operation on two floating-point numbers of same length n, we obtain a floating-point number of different length m (usually $m > n$). Computer cannot store the resulting number exactly since it can represent numbers a length n. So only n digits are stored. This gives rise to error.

Example: Let $a = 0.75632 \times 10^2$

and $b = 0.235472 \times 10^{-1}$

$a + b = 75.632 + 0.023 = 75.655472$ in ammulator

$a + b = 0.756555 \times 10$ if 6 decimal digit arithmetic is used.

We denote the corresponding machine operation by superscript * i.e.

$a +^* b = 0.756555 \times 10^2$ (0.756555E2)

Due to generated error, the associative and the distributive laws of arithmetic are not satisfied in some case as shown below:

In a computer 3×1/3 would be represented as 0.999999 (in case of six significant digit) but by hand computation it is one. This simple illustration suggested that everything does not go well on computers. More precisely $0.333333 + 0.333333 + 0.333333 = 0.999999$.

Non-Associativity of Arithmetic

As we have seen that numbers have to be truncated to fit into the 4 mantissa digits allowed in our hypothetical computer for each number. This truncation leads to a number of apparently surprising results (namely, results which are not used to in our experience with arithmetic). For instance 6 × 2/3 = 4 as well as know. However, when the arithmetic is performed with floating-point numbers .6667 added 6 times gives .3998E1 whereas .6667 × 6 gives .4000E1 (check this using floating-point arithmetic).

Another consequences of the floating-point representation is that the associativity and the distributive laws of arithmetic are not always valid.

In other words,

$$(a + b) - c \neq (a - c) + b$$
$$a(b - c) \neq (ab - ac).$$

Example:

Let $a = 0.5665$ E1

$b = 0.5556$E–1

$c = 0.5644$E1

Therefore,

$$\begin{aligned}(a + b) &= 0.5665\text{E}1 + 0.5556\text{E} - 1\\ &= 0.5665\text{E}1 + 0.0055\text{E}1\\ &= 0.5720\text{E}1\\ (a + b) - c &= 0.5720\text{E}1 - 0.5644\text{E}1\\ &= 0.0076\text{E}1 = 0.7600\text{E}1\\ &= 0.7600\text{E}{-1}\end{aligned}$$