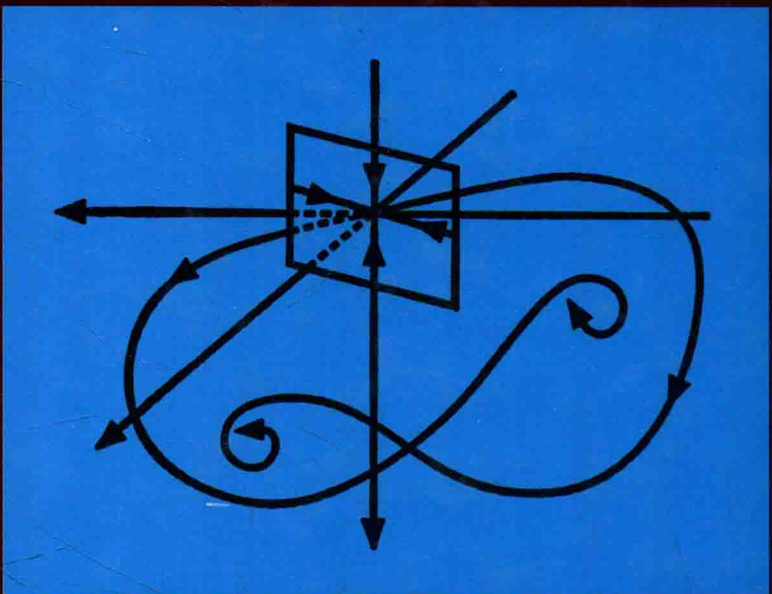
ELSEVIER

MATHEMATICS IN SCIENCE AND ENGINEERING *Volume* 201
SERIES EDITOR: C.K. CHUI

Computational Error *and* Complexity *in* Science *and* Engineering



V. Lakshmikantham and S.K. Sen

Computational Error and Complexity in Science and Engineering

V. Lakshmikantham
FLORIDA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MATHEMATICAL SCIENCES
MELBOURNE, FLORIDA

S.K. Sen
FLORIDA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MATHEMATICAL SCIENCES
MELBOURNE, FLORIDA

2005
ELSEVIER
Amsterdam – Boston – Heidelberg – London – New York – Oxford
Paris – San Diego – San Francisco – Singapore – Sydney – Tokyo

ELSEVIER B.V.	ELSEVIER Inc.	ELSEVIER Ltd	ELSEVIER Ltd
Radarweg 29	525 B Street, Suite 1900	The Boulevard, Langford Lane	84 Theobalds Road
P.O. Box 211, 1000 AE Amsterdam	San Diego, CA 92101-4495	Kidlington, Oxford OX5 1GB	London WC1X 8
The Netherlands	USA	UK	UK

First edition 2005

Library of Congress Cataloging in Publication Data
A catalog record is available from the Library of Congress.

British Library Cataloguing in Publication Data
A catalogue record is available from the British Library.

ISBN: 0-444-51860-6
ISSN (Series): 0076-5392

⊗ The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).
Printed in Great Britain.

Computational Error and Complexity in Science and Engineering

This is volume 201 in
MATHEMATICS IN SCIENCE AND ENGINEERING
Edited by C.K. Chui, *Stanford University*

A list of recent titles in this series appears at the end of this volume.

Preface

The monograph focuses on an estimation of the quality of the results/outputs produced by an algorithm in scientific and engineering computation. In addition the cost to produce such results by the algorithm is also estimated. The former estimation refers to error computation while the later estimation refers to complexity computation. It is mainly intended for the graduate in engineering, computer science, and mathematics. It can also be used for the undergraduate by selecting topics pertinent to a given curriculum. To gain practical experience, any such course should be supplemented with laboratory work.. Besides, it would be of value as a reference to anyone engaged in numerical computation with a high-speed digital computer.

If we have to compare two or more algorithms to solve a particular type of problems, we need both error and complexity estimation for each of the algorithms. Whenever we solve a problem and produce a result, we would always like to know error in the result and the amount of computation and that of storage, i.e., computational complexity and space complexity. The monograph is precisely an exposition of both error and complexity over different types of algorithms including exponential/combinatorial ones.

Chapter 1 is introductory. It discusses the distinction between science and engineering, highlights the limitation of computation, tools and types of computation, algorithms and complexity, models of computation, computer-representable numbers, and stages of problem-solving.

Chapter 2 is an exposition of all that is connected with error. Precisely what error is, why we get error, and how we estimate the error constitute the core of this chapter. Similarly, Chapter 3 explains what, why, and how of complexity of algorithms including various types of complexity.

Errors and approximations in digital computers constitute Chapter 4. The details of IEEE 754 arithmetic are also included in this chapter. Chapter 5, on the other hand, presents several numerical algorithms and the associated error and complexity.

Error in error-free computation as well as that in parallel and probabilistic computations are described in Chapter 6. The confidence level which is never 100% in probabilistic computations is stressed in this chapter.

Simple examples have been included throughout the monograph to illustrate the underlying ideas of the concerned topics. Sufficient references have been included in each chapter.

Certainly a monograph of this type cannot be written without deriving many valuable ideas from several sources. We express our indebtedness to all the authors, too numerous to acknowledge individually, from whose specialized knowledge we have benefited.

V. Lakshmikantham
S.K. Sen

Contents

Chapter 1

Introduction

1.1 Science versus engineering

The Collins Gem dictionary meaning of science is the systematic study of natural or physical phenomena while that of engineering is the profession of applying scientific principles to the design and construction of engines, cars, buildings, or machines. All the laws of physics such as the Newton's laws of motion, the first and second laws of thermodynamics, Stokes law, all the theorems in mathematics such as the binomial theorem, Pythagoras theorem, fundamental theorem of linear algebra, fundamental theorem of linear programming, all the laws, rules, and properties in chemistry as well as in biology come under science. In engineering, on the other hand, we make use or apply these rules, laws, properties of science to achieve/solve specified physical problems including real-world implementation of the solution.

To stress the difference between science and engineering, consider the problem: Compute $f(x) = (x^2 - 4)/(x - 2)$ at $x = 2$. In engineering/technology, the answer is 4. This is obtained just by taking the left-hand limit as well as the right-hand limit and observing that these are equal. A simpler numerical way to obtain the value of f(x) at x = 2 in engineering is to compute f(x) at x = 1.99, 2.01. 1.999, 2.001, 1.9999, 2.0001, and observe that these values increasingly become closer to 4. We have assumed in the previous computation sufficiently large, say 14 digit, precision. In fact, the value of f(x) at $x = 2 + 10^{-500}$ as well as at $x = 2 - 10^{-500}$ will each be extremely close to 4. By any measuring/computing device in engineering, we will get f(x) as 4 although exactly at the point x = 2, f(x) is not defined. In science/mathematics, the solution of the problem will be output as undefined (0/0 form).

The function $y(x) = |x|$ is 0 at x = 0. The left-hand limit, the right-hand limit, and the value of the function at x = 0 are all the same. Hence y(x) is

continuous at x = 0. The first derivative of y(x) at x = 0 does not exist as the right-hand derivative
$y'_r(0) = \lim_{h\to 0^+} (y(0 + h) - y(0))/h = +1$

while the left-hand derivative

$y'_l(0) = \lim_{h\to 0^-} (y(0 + h) - y(0))/h = -1$

and both are different. In engineering/technology, we would say "*y'(0) does not exist*". In science/mathematics, the most precise answer will be "*$y'_r(0)$ exists and is +1 while $y'_l(0)$ exists and is –1 and $y'_r(0) \neq y'_l(0)$*". One might say that this answer implies "the derivative y′(0) does not exist". Strictly speaking, the implication may not tell us the fact that the left-hand derivative does certainly exist as well as the right-hand derivative also does exist. For the sake of preciseness, we, however, still prefer to distinguish these answers.

Consider yet another problem: Compute $g(x) = (\sqrt{(\sin^2 x)})/x$ at x = 0. In engineering/technology, the answer is "*g(0) does not exist at x = 0*". This is obtained by taking the left-hand limit and the right-hand limit and observing that these limist are not equal. One is –1 while the other is +1. A simpler numerical way to obtain the value of g(x) at x = 0 in engineering is to compute g(x) at x = –.001, +.001, –.0001, +.0001, –.00001, +.00001 and observe that these values will alternately tend to –1 and +1. The solution of the problem in science could be output as undefined (0/0 form). However, if we pose the problem as "Compute $g(x) = \lim_{x\to 0} \sqrt{(\sin^2 x)/x}$" then in engineering the answer will be "*the limit does not exist*". In science, the precise answer will be "*the left-hand limit exists and it is –1; the right-hand limit exists and it is +1; both are different*". In fact, the answer in engineering, viz., "the limit does not exist" may not reveal the fact that the left-hand limit exists, so does the right-hand limit. All these are essentially subtle differences. A clear conceptual understanding of these differences does help us in a given context.

From the computation point of view, we will not distinguish between science and engineering computations although we might keep in mind the context while performing computations. However, the precision of computation in science may be significantly more than that in engineering. In fact, in engineering/technology, a relative error (lack of accuracy) less than 0.005% is not, in general, required as it is not implementable in the real world situation and it is hard to find a measuring device which gives accuracy more than 0.005%. We will discuss this accuracy aspect further later in this book.

1.2 Capability and limit of computation

One common feature that pervades both science and engineering is computation. The term computation is used here in the context of a digital computer in a broader sense, viz., in the sense of data/information processing that includes arithmetic and nonarithmetic operations as well as data communication as discussed in Section 1.3. In fact, anything that is done by a computer/computing system is computation. *While mathematical quantities may not satisfy a scientist/an engineer, the numerical quantities do.* A conceptual clarity and quantitative feeling are improved through computation. Till mid-twentieth century, we had computational power next to nothing compared to to-day's (beginning of twenty-first century's) power. To-day tera-flops (10^{12} floating-point operations per second) is a reality and we are talking of peta-flops (10^{15} floating-point operations per second). In fact, the silicon technology on which the digital computers are based is still going unparallely strong. Every 18 months the processing power is doubled, every twelve months the data-communication band-width is doubled while every nine months the disk storage capacity is doubled. The other technologies which might lead to quantum computers or protein-based computers are not only in their infancy but also are not yet commercially promising. These do have some excellent theoretical properties as well as severe bottle-necks.

Capability of computation An important need for computational power is storage/memory. For higher computational power, larger memory is needed since a smaller memory could be a bottle-neck. A rough chart representing storage capacity (bits) versus computational power (bits per second) in both biological computers (living beings including animals) and non-biological (non-living) machines could be as given in Table 1.

Among living computers, the first (topmost) place goes to the whale having a huge memory capacity of 10^{16} bits and a processing speed of 10^{16} bits/sec while among nonliving computers it is the supercomputer (2003) with 10^{14} bits of storage and 10^{13} bits/sec of processing speed in the top position. The British library has 10^{15} bits of information but the processing capability is of order 1, i.e., practically nil. The supercomputing power and storage capacity is dynamic in the sense these are increasing with time while the living computer's power and storage capacity is possibly not that dynamic. It is not seriously possible to distinguish between the nineteenth century human beings and twenty-first century human beings in terms of their memory capability and processing power.

Limit of computation Can we go on doubling the processing power indefinitely? Is there a limit for this power? The answers to these questions are "no" and "yes", respectively. Our demand for higher computational speed as well as storage knows no bound. There are problems, say those in weather forecast, VLSI design, that would take over 1500 hours on today's (2003) supercomputers to be solved. A computer in early 1980s was considered the supermachine if it was capable of executing over 100 million floating point operations per second (> 100 Mflops) with word length of 64 bits and main memory capacity of over 100 million words. Today (2003) it is called a supermachine if it can execute over 1 billion flops (> 1 Gflops) with the same word-length of 64 bits and main memory capacity of over 256 million words. Thus the definition of supercomputers is time-dependent, i.e., yesterday's supercomputers are today's ordinary computers.

Table 1 Memory capacity and computational power of computers

Computers (Living/nonliving)	**Storage capacity (number of bits)**	**Computational power (number of bits/sec)**
Abacus	10^0	10^0
Radio channel	10^0	10^3
Television channel	10^0	10^6
Viral DNA	10^3	10^0
Hand calculator	10^3	10^3
Smart missile	10^3	10^9
Bacterial DNA	10^6	10^0
Bacterial reproduction	10^6	10^3
Personal computer	10^6	10^6
Main frame computer (1980s)	10^8	10^8
Human DNA	10^9	10^0
Honey bee	10^9	10^8
Rat/mouse	10^9	10^{10}
Telephone system	10^{11}	10^{13}
English dictionary	10^{12}	10^0
Video recorder	10^{12}	10^6
Cray supercomputer (1985)	10^{12}	10^{11}
Human visual system	10^{13}	10^{13}
Supercomputer (2003)	10^{14}	10^{13}
Elephant	10^{14}	10^{16}
Human being	10^{14}	10^{13}
British library	10^{15}	10^0
Whale	10^{16}	10^{16}

To discuss about the limit of computation, we should keep the following facts (Alam and Sen 1996) in mind:

1. Classical Von Neumann architecture in which all instructions are executed sequentially has influenced programmers to think sequentially.
2. Programming is affected by both the technology and the architecture which are interrelated.
3. Physics rather than technology and architecture sets up the obstacles (barriers)/ limits to increase the computational power arbitrarily:
   (i) *Speed of light barrier*: Electrical signals (pulses) cannot propagate faster than the speed of light. A random access memory used to 10^9 cycles per second (1 GHtz) will deliver information/data at 0.1 nanosecond (0.1×10^{-9} second) speed if it has a diameter of 3 cm since in 0.1 nanosecond, light travels 3 cm.
   (ii) *Thermal efficiency barrier* The entropy of the system increases whenever there is information processing. Hence the amount of heat that is absorbed is kT $\log_e 2$ per bit, where k is the Boltzmann constant (1.38×10^{-16} erg per degree) and T is the absolute temperature (taken as room temperature, i.e., 300). It is not possible to economize any further on this. If we want to process 10^{30} bits per second, the amount of power that we require is $10^{30} \times 1.38 \times 10^{-16} \times 300 \times 0.6931 / 10^7 = 2.8697 \times 10^9$ watts, where 10^7 erg/sec = 1 watt.
   (iii) *Quantum barrier* Associated with every moving particle is a wave which is quantified such that the energy of one quantum E = hν, where ν = frequency of the wave and h = Plank's constant. The maximum frequency $\nu_{max} = mc^2/h$, where m = mass of the system and c = velocity of light. Thus the frequency band that can be used for signaling is limited to the maximum frequency ν_{max}. From Shannon's information theory, the rate of information (number of information that can be processed per second) cannot exceed ν_{max}. The mass of hydrogen atom is 1.67×10^{-24} gm. $c = 3 \times 10^{10}$ cm/sec, $h = 6 \times 10^{-27}$. Hence per mass of hydrogen atom, maximum $1.67 \times 10^{-24} \times 3^2 \times 10^{20} / (6 \times 10^{-27}) = 2.5050 \times 10^{23}$ bits/sec can be transmitted. The number of protons in the universe is estimated to be around 10^{73}. Hence if the whole universe is dedicated to information processing, i.e., if all the 10^{73} protons are employed to information processing simultaneously (parallely) then no more than 2.5050×10^{96} bits/sec or 7.8996×10^{103} bits per year can be processed. This is the theoretical limitation (for massively parallel processing) set by the laws of physics and we are nowhere near it! A single

processor (proton) can process only 2.5×10^{23} bits per second or 7.9×10^{30} bits per year and no more!

1.3 What is computation in science and engineering

The word computation conjures up an image of performing arithmetic operations such as add, subtract, multiply, and divide operations on numbers. Undoubtedly these four basic arithmetic operations as well as other operations such as a square-rooting (which involves an infinite sequence of four basic arithmetic operations, in general) do come under computation. Here by computation we would imply a much more general (broader) activity, viz., data (or information or knowledge) processing including data/control signal communication using a digital computer — conventional as well as intelligent. Each and every machine language instruction that a hardware computer executes constitutes a unit of computation. The add, the subtract, the multiply, and the divide instructions, the unconditional and conditional jump/branch instructions, the *for, while, repeat-until* instructions, *read, write/print* instructions form the building blocks of computation. Each block consists of an ordered sequence of machine instructions. The other instructions such as square-rooting (*sqrt*), sine (*sin*), cosine (*cos*), tangent (*tan*), cotangent (*cot*) computation can be considered higher (macro) level building blocks of computation. One may develop still higher level building blocks such as *inverting a matrix, drawing a least-squares curve* which can be found in MATLAB commands/instructions. Basically, a computer accepts only two distinct symbols, viz., 0 and 1 and operates/manipulates on these symbols in fixed or variable sequences and produces only sequences consisting of these two symbols, viz., 0 and 1 as outputs interpreted according to an appropriate format and context.

Although we talked about computation in a broader sense, *we will limit ourselves with the order of dominant operations for the sake of error and computational complexity*[1].

---

[1]A complex problem or a complex computation implies that the problem/computation is made up of parts; also it implies that it is complicated. The larger the number of parts of the problem/computation is, the more complex it is. For a human being — a living computer — the later implication is often more understood than the former one although psychologically the former implication also would accentuate the difficulty in terms of grasping/remembering. For a nonliving machine — a digital computer — there is absolutely no problem of grasping/remembering or that of ease or difficulty; an analogue of former implication, viz., the number of instructions, measured as amount of computation, is the complexity.

1.4 Tools for Computation

During the pre-computer days, we have been using (i) some kind of writing media such as palm leaves, slates, stone walls/slabs, mud/earth, appropriate materials (plaster of paris), paper and (ii) some kind of writing tools such as ink-pot (ink made of a mixture of powered charcoal and water or liquid colour extracted from plants and/or trees) and pen (pen made of 5 to 6 inches long sharpened bamboo branch or peacock feather or some other stick/brush or some sharp hard object or ball-point pen or fountain pen) combination for doing arithmetic computation as well as graphs, drawings, images/statues — both two dimensional and three dimensional.

During the modern computer days (late twentieth and twenty-first centuries), we use computers as another aid like paper-and-pencil but with much more revolutionary impact on us. If we are asked to compute the positive square-root of a number, say, 30 we could do it over a longer time with or without mistake using paper and pencil provided we know the deterministic arithmetic algorithm for square-rooting. The alternative way is to take out the pocket calculator — programmable or non-programmable — and instantly find the square-root of the number mistakeless (not errorless, in general) by pressing the two number keys 3 and 0 and the square-root operation key. It may be seen that the probability of modern computers committing mistakes in executing an instruction is practically nil while that of any living being — superhuman being or animal or common human being — committing mistake is certainly not nil[2]. However, computers during 1950's and early 1960's did produce wrong output due to circuit failure/malfunction without giving any warning/indication of mistake to us. For example URAL, a Russian computer, that worked with a number system in base 8 during late 1950s/early 1960s did produce occasionally wrong results. A British computer HEC 2M that we had used during late 1950s and early 1960s was relatively good but still not 100% mistakeless. Thus our

---

[2] During one afternoon in early 1970s, we have seen in Indian Institute of Science, Bangalore, India Ms. Sakuntala Devi, a human computer, telling too fast the numerical value of a very long arithmetic expression consisting of terms like $(-72.345)^{6.8911}$ and running from the top corner of a black board to the bottom corner (writing took more than 5 minutes). She took mainly the time to read the expression sequentially and told the answer which was incorrect but within another couple of seconds she told the answer correctly (as computed by the then IBM 360/44 computer! This is not a magic as she has demonstrated this superhuman faculty time and again in various environments/forum. Even a person like her could commit mistakes. How she does this is not known to us nor is it possibly known to her or she is not able to communicate all that goes in her brain.