Texts in Applied Mathematics 12

J.Stoer
R.Bulirsch

Introduction to Numerical Analysis

Second Edition

数值分析导论

第 2 版

Springer-Verlag
世界图书出版公司

J. Stoer R. Bulirsch

Introduction to Numerical Analysis

Second Edition

Translated by R. Bartels, W. Gautschi, and C. Witzgall

With 35 Illustrations

Springer-Verlag
世界图书出版公司
北京・广州・上海・西安

书　　名：Introduction to Numerical Analysis　2nd ed.
作　　者：J.Stoer　R. Bulirsch
中 译 名：数值分析导论 第 2 版
出 版 者：世界图书出版公司北京公司
印 刷 者：北京世图印刷厂
发　　行：世界图书出版公司北京公司 (北京朝内大街 137 号　100010)
联系电话：010-64015659, 64038347
电子信箱：kjsk@vip.sina.com
开　　本：24 开　　印　张：28.5
出版年代：1998 年 3 月第 1 版　　2004 年 5 月第 2 次印刷
书　　号：7-5062-3389-4
版权登记：图字:01-9[illegible]
定　　价：98.00 元

Preface to the Second Edition

On the occasion of this new edition, the text was enlarged by several new sections. Two sections on B-splines and their computation were added to the chapter on spline functions: Due to their special properties, their flexibility, and the availability of well-tested programs for their computation, B-splines play an important role in many applications.

Also, the authors followed suggestions by many readers to supplement the chapter on elimination methods with a section dealing with the solution of large sparse systems of linear equations. Even though such systems are usually solved by iterative methods, the realm of elimination methods has been widely extended due to powerful techniques for handling sparse matrices. We will explain some of these techniques in connection with the Cholesky algorithm for solving positive definite linear systems.

The chapter on eigenvalue problems was enlarged by a section on the Lanczos algorithm; the sections on the LR and QR algorithm were rewritten and now contain a description of implicit shift techniques.

In order to some extent take into account the progress in the area of ordinary differential equations, a new section on implicit differential equations and differential-algebraic systems was added, and the section on stiff differential equations was updated by describing further methods to solve such equations.

The last chapter on the iterative solution of linear equations was also improved. The modern view of the conjugate gradient algorithm as an iterative method was stressed by adding an analysis of its convergence rate and a description of some preconditioning techniques. Finally, a new section on multigrid methods was incorporated: It contains a description of their basic ideas in the context of a simple boundary value problem for ordinary differential equations.

Many of the changes were suggested by several colleagues and readers. In particular, we would like to thank R. Seydel, P. Rentrop, and A. Neumaier for detailed proposals and our translators R. Bartels, W. Gautschi, and C. Witzgall for their valuable work and critical commentaries. The original German version was handled by F. Jarre, and I. Brugger was responsible for the expert typing of the many versions of the manuscript.

Finally we thank Springer-Verlag for the encouragement, patience, and close cooperation leading to this new edition.

Würzburg, München
May 1991

J. Stoer
R. Bulirsch

Preface to the First Edition

This book is based on a one-year introductory course on numerical analysis given by the authors at several universities in Germany and the United States. The authors concentrate on methods which can be worked out on a digital computer. For important topics, algorithmic descriptions (given more or less formally in ALGOL 60), as well as thorough but concise treatments of their theoretical foundations, are provided. Where several methods for solving a problem are presented, comparisons of their applicability and limitations are offered. Each comparison is based on operation counts, theoretical properties such as convergence rates, and, more importantly, the intrinsic numerical properties that account for the reliability or unreliability of an algorithm. Within this context, the introductory chapter on error analysis plays a special role because it precisely describes basic concepts, such as the numerical stability of algorithms, that are indispensable in the thorough treatment of numerical questions.

The remaining seven chapters are devoted to describing numerical methods in various contexts. In addition to covering standard topics, these chapters encompass some special subjects not usually found in introductions to numerical analysis. Chapter 2, which discusses interpolation, gives an account of modern fast Fourier transform methods. In Chapter 3, extrapolation techniques for speeding up the convergence of discretization methods in connection with Romberg integration are explained at length.

The following chapter on solving linear equations contains a description of a numerically stable realization of the simplex method for solving linear programming problems. Further minimization algorithms for solving unconstrained minimization problems are treated in Chapter 5, which is devoted to solving nonlinear equations.

After a long chapter on eigenvalue problems for matrices, Chapter 7 is

devoted to methods for solving ordinary differential equations. This chapter contains a broad discussion of modern multiple shooting techniques for solving two-point boundary-value problems. In contrast, methods for partial differential equations are not treated systematically. The aim is only to point out analogies to certain methods for solving ordinary differential equations, e.g., difference methods and variational techniques. The final chapter is devoted to discussing special methods for solving large sparse systems of linear equations resulting primarily from the application of difference or finite element techniques to partial differential equations. In addition to iteration methods, the conjugate gradient algorithm of Hestenes and Stiefel and the Buneman algorithm (which provides an example of a modern direct method for solving the discretized Poisson problem) are described.

Within each chapter numerous examples and exercises illustrate the numerical and theoretical properties of the various methods. Each chapter concludes with an extensive list of references.

The authors are indebted to many who have contributed to this introduction into numerical analysis. Above all, we gratefully acknowledge the deep influence of the early lectures of F.L. Bauer on our presentation. Many colleagues have helped us with their careful reading of manuscripts and many useful suggestions. Among others we would like to thank are C. Reinsch, M.B. Spijker, and, in particular, our indefatigable team of translators, R. Bartels, W. Gautschi, and C. Witzgall. Our co-workers K. Butendeich, G. Schuller, J. Zowe, and I. Brugger helped us to prepare the original German edition. Last but not least we express our sincerest thanks to Springer-Verlag for their good cooperation during the past years.

Würzburg, München J. Stoer
August 1979 R. Bulirsch

Contents

Error Analysis 1

Assessing the accuracy of the results of calculations is a paramount goal in numerical analysis. One distinguishes several kinds of errors which may limit this accuracy:

(1) *errors in the input data,*
(2) *roundoff errors,*
(3) *approximation errors.*

Input or data errors are beyond the control of the calculation. They may be due, for instance, to the inherent imperfections of physical measurements. Roundoff errors arise if one calculates with numbers whose representation is restricted to a finite number of digits, as is usually the case.

As for the third kind of error, many methods will not yield the exact solution of the given problem P, even if the calculations are carried out without rounding, but rather the solution of another simpler problem $\tilde{P}$ which approximates P. For instance, the problem P of summing an infinite series, e.g.,

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots,$$

may be replaced by the simpler problem $\tilde{P}$ of summing only up to a finite number of terms of the series. The resulting approximation error is commonly called a *truncation error* (however, this term is also used for the roundoff related error committed by deleting any last digit of a number representation). Many approximating problems P are obtained by "discretizing" the original problem P: definite integrals are approximated by finite sums, differential quotients by a difference quotients, etc. In such cases, the approximation error is often referred to as *discretization error.*

Some authors extend the term "truncation error" to cover discretization errors.

In this chapter, we will examine the general effect of input and roundoff errors on the result of a calculation. Approximation errors will be discussed in later chapters as we deal with individual methods. For a comprehensive treatment of roundoff errors in floating-point computation see Sterbenz (1974).

1.1 Representation of Numbers

Based on their fundamentally different ways of representing numbers, two categories of computing machinery can be distinguished:

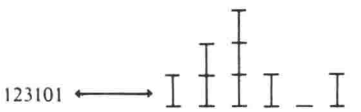
(1) *analog computers*,
(2) *digital computers*.

Examples of analog computers are slide rules and mechanical integrators as well as electronic analog computers. When using these devices one replaces numbers by physical quantities, e.g., the length of a bar or the intensity of a voltage, and simulates the mathematical problem by a physical one, which is solved through measurement, yielding a solution for the original mathematical problem as well. The scales of a slide rule, for instance, represent numbers x by line segments of length $k \ln x$. Multiplication is simulated by positioning line segments contiguously and measuring the combined length for the result.

It is clear that the accuracy of analog devices is directly limited by the physical measurements they employ.

Digital computers express the digits of a number representation by a sequence of discrete physical quantities. Typical instances are desk calculators and electronic digital computers.

EXAMPLE



Each digit is represented by a specific physical quantity. Since only a small finite number of different digits have to be encoded—in the decimal number system, for instance, there are only 10 digits—the representation of digits in digital computers need not be quite as precise as the representation of numbers in analog computers. Thus one might tolerate voltages between, say, 7.8 and 8.2 when aiming at a representation of the digit 8 by 8 volts.

Consequently, the accuracy of digital computers is not directly limited by the precision of physical measurements.

For technical reasons, most modern electronic digital computers represent numbers internally in *binary* rather than decimal form. Here the coefficients or *bits* α_i of a decomposition by powers of 2 play the role of digits in the representation of a number x:

$$x = \pm(\alpha_n 2^n + \alpha_{n-1}2^{n-1} + \cdots + \alpha_0 2^0 + \alpha_{-1}2^{-1} + \alpha_{-2}2^{-2} + \cdots),$$
$$\alpha_i = 0 \quad \text{or} \quad 1.$$

In order not to confuse decimal and binary representations of numbers, we denote the bits of a binary number representation by **O** and **L**, respectively.

EXAMPLE. The number $x = 18.5$ admits the decomposition

$$18.5 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$$

and has therefore the binary representation

LOOLO.L.

We will use mainly the decimal system, pointing out differences between the two systems whenever it is pertinent to the examination at hand.

As the example $3.999\ldots = 4$ shows, the decimal representation of a number may not be unique. The same holds for binary representations. To exclude such ambiguities, we will always refer to the finite representation unless otherwise stated.

In general, digital computers must make do with a fixed finite number of places, the *word length*, when internally representing a number. This number n is determined by the make of the machine, although some machines have built-in extensions to integer multiples $2n$, $3n$, ... (double word length, triple word length, ...) of n to offer greater precision if needed. A word length of n places can be used in several different fashions to represent a number.

Fixed-point representation specifies a fixed number n_1 of places before and a fixed number n_2 after the decimal (binary) point, so that $n = n_1 + n_2$ (usually $n_1 = 0$ or $n_1 = n$).

EXAMPLE. For $n = 10$, $n_1 = 4$, $n_2 = 6$

30.421 →	0030	421000
0.0437 →	0000	043700
	n_1	n_2

In this representation, the position of the decimal (binary) point is fixed. A few simple digital devices, mainly for accounting purposes, are still re-

stricted to fixed-point representation. Much more important, in particular for scientific calculations, are digital computers featuring *floating-point representation* of numbers. Here the decimal (binary) point is not fixed at the outset; rather its position with respect to the first digit is indicated for each number separately. This is done by specifying a so-called *exponent*. In other words, each real number can be represented in the form

$$x = a \times 10^b (x = a \times 2^b) \quad \text{with} \quad |a| < 1,\ b \text{ integer} \tag{1.1.1}$$

(say, 30.421 by 0.30421×10^2), where the exponent b indicates the position of the decimal point with respect to the *mantissa* a. Rutishauser proposed the following "semilogarithmic" notation, which displays the basis of the number system at the subscript level and moves the exponent down to the level of the mantissa:

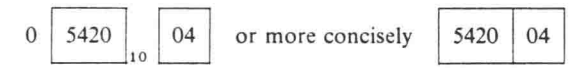
$$0.30421_{10}2$$

Analogously,

$$\mathbf{O.LOOLOL_2LOL}$$

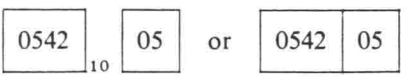
denotes the number 18.5 in the binary system. On any digital computer there are, of course, only fixed finite numbers t and e, $n = t + e$, of places available for the representation of mantissa and exponent, respectively.

EXAMPLE. For $t = 4$, $e = 2$ one would have the floating-point representation



for the number 5420 in the decimal system.

The floating-point representation of a number need not be unique. Since $5420 = 0.542_{10}4 = 0.0542_{10}5$, one could also have the floating-point representation



instead of the one given in the above example.

A floating-point representation is *normalized* if the first digit (bit) of the mantissa is different from 0 (**O**). Then $|a| \geqslant 10^{-1}$ ($|a| \geqslant 2^{-1}$) holds in (1.1.1). The *significant digits* (*bits*) of a number are the digits of the mantissa not counting leading zeros.

In what follows, we will only consider normalized floating-point representations and the corresponding floating-point arithmetic. The numbers t and e determine—together with the basis $B = 10$ or $B = 2$ of the number representation—the set $A \subseteq \mathbb{R}$ of real numbers which can be represented exactly within a given machine. The elements of A are called the *machine numbers*.

While normalized floating-point arithmetic is prevalent on current electronic digital computers, unnormalized arithmetic has been proposed to ensure that only truly significant digits are carried [Ashenhurst and Metropolis, (1959)].

1.2 Roundoff Errors and Floating-Point Arithmetic

The set A of numbers which are representable in a given machine is only finite. The question therefore arises of how to approximate a number $x \notin A$ which is not a machine number by a number $g \in A$ which is. This problem is encountered not only when reading data into a computer, but also when representing intermediate results within the computer during the course of a calculation. Indeed, straightforward examples show that the results of elementary arithmetic operations $x \pm y$, $x \times y$, x/y need not belong to A, even if both operands $x, y \in A$ are machine numbers.

It is natural to postulate that the approximation of any number $x \notin A$ by a machine number $\mathrm{rd}(x) \in A$ should satisfy

$$|x - \mathrm{rd}(x)| \leqslant |x - g| \quad \text{for all } g \in A. \tag{1.2.1}$$

Such a machine-number approximation $\mathrm{rd}(x)$ can be obtained in most cases by *rounding*.

EXAMPLE 1 ($t = 4$)

$$\begin{aligned} \mathrm{rd}(0.14285_{10}0) &= 0.1429_{10}0, \\ \mathrm{rd}(3.14159_{10}0) &= 0.3142_{10}1, \\ \mathrm{rd}(0.142842_{10}2) &= 0.1428_{10}2. \end{aligned}$$

In general, one can proceed as follows in order to find $\mathrm{rd}(x)$ for a t-digit computer: $x \notin A$ is first represented in normalized form $x = a \times 10^b$, so that $|a| \geqslant 10^{-1}$. Suppose the decimal representation of $|a|$ is given by

$$|a| = 0.\alpha_1\alpha_2 \ldots \alpha_i\alpha_{i+1} \ldots, \qquad 0 \leqslant \alpha_i \leqslant 9, \quad \alpha_1 \neq 0.$$

Then one forms

$$a' := \begin{cases} |0.\alpha_1\alpha_2 \ldots \alpha_t & \text{if } 0 \leqslant \alpha_{t+1} \leqslant 4, \\ |0.\alpha_1\alpha_2 \ldots \alpha_t + 10^{-t} & \text{if } \alpha_{t+1} \geqslant 5, \end{cases}$$

that is, one increases α_t by 1 if the $(t+1)$st digit $\alpha_{t+1} \geqslant 5$, and deletes all digits after the tth one. Finally one puts

$$\widetilde{\mathrm{rd}}(x) := \mathrm{sign}(x) \cdot a' \times 10^b.$$