Walter Gautschi

Numerical Analysis
An Introduction

1997

Birkhäuser
Boston • Basel • Berlin

Walter Gautschi
Department of Computer Science
Purdue University
West Lafayette, IN 47907-1398
USA

Library of Congress Cataloging-in-Publication Data

Gautschi, Walter.
Numerical analysis : an introduction / Walter Gautschi.
p. cm.
Includes bibliographical references (p. 451- 481).
ISBN 0-8176-3895-4 (alk. paper). -- ISBN 3-7643-3895-4 (Basel : alk. paper)
1. Numerical analysis. I. Title.
QA297.G35 1997
519.4--dc21 97-186
CIP

Printed on acid-free paper

ISBN 0-8176-3895-4
ISBN 3-7643-3895-4
Typeset by the Author in LaTeX.
Cover design by Dutton & Sherman Design, New Haven, CT.
Printed and bound by Quinn-Woodbine, Woodbine, NJ.
Printed in the U.S.A.
9 8 7 6 5 4 3 2 1

CONTENTS

PREFACE

The book is designed for use in a graduate program in Numerical Analysis that is structured so as to include a basic introductory course and subsequent more specialized courses. The latter are envisaged to cover such topics as numerical linear algebra, the numerical solution of ordinary and partial differential equations, and perhaps additional topics related to complex analysis, to multidimensional analysis, in particular optimization, and to functional analysis and related functional equations. Viewed in this context, the first four chapters of our book could serve as a text for the basic introductory course, and the remaining three chapters (which indeed are at a distinctly higher level) could provide a text for an advanced course on the numerical solution of ordinary differential equations. In a sense, therefore, the book breaks with tradition in that it does no longer attempt to deal with all major topics of numerical mathematics. It is felt by the author that some of the current subdisciplines, particularly those dealing with linear algebra and partial differential equations, have developed into major fields of study that have attained a degree of autonomy and identity that justifies their treatment in separate books and separate courses on the graduate level. The term "Numerical Analysis" as used in this book, therefore, is to be taken in the narrow sense of the numerical analogue of Mathematical Analysis, comprising such topics as machine arithmetic, the approximation of functions, approximate differentiation and integration, and the approximate solution of nonlinear equations and of ordinary differential equations.

What is being covered, on the other hand, is done so with a view toward stressing basic principles and maintaining simplicity and student-friendliness as far as possible. In this sense, the book is "An Introduction". Topics that, even though important and of current interest, require a level of technicality that transcends the bounds of simplicity striven for, are referenced in detailed bibliographic notes at the end of each chapter. It is hoped, in this way, to place the material treated in proper context and to help, indeed encourage, the reader to pursue advanced modern topics in more depth.

A significant feature of the book is the large collection of exercises that are designed to help the student develop problem-solving skills, and to provide interesting extensions of topics treated in the text. Particular attention is given to machine assignments, where the student is encouraged to implement numerical techniques on the computer and to make use of modern software packages.

The author has taught the basic introductory course, and the advanced course on ordinary differential equations regularly at Purdue University for the last 30 years or so. The former, typically, was offered both in the fall and spring semesters, to a mixed audience consisting of graduate (and some good undergraduate) students in mathematics, computer science, and engineering, while the latter was taught only in the fall, to a smaller but also mixed audience. Written notes began to materialize in the 1970s, when the author taught the basic course repeatedly in summer courses on Mathematics held in Perugia, Italy. Indeed, for some time, these notes existed only in the Italian language. Over the years, they were progressively expanded, updated, and transposed into English, and along with that, notes for the advanced course were developed. This, briefly, is how the present book evolved.

A long gestation period such as this, of course, is not without dangers, the most notable one being a tendency for the material to become dated. The author tried to counteract this by constantly updating and revising the notes, adding newer developments when deemed appropriate. There are, however, benefits as well: over time, one develops a sense for what is likely to stand the test of time and what may only be of temporary interest, and one selects and deletes accordingly. Another benefit is the steady accumulation of exercises and the opportunity to have them tested on a large and diverse student population.

The purpose of academic teaching, in the author's view, is twofold: to transmit knowledge, and, perhaps more important, to kindle interest and even enthusiasm in the student. Accordingly, the author did not strive for comprehensiveness — even within the boundaries delineated — but rather tried to concentrate on what is essential, interesting and intellectually pleasing, and teachable. In line with this, an attempt has been made to keep the text uncluttered with numerical examples and other illustrative material. Being well aware, however, that mastery of a subject does not come from studying alone, but from active participation, the author provided many exercises, including machine projects. Attributions of results to specific authors and citations to the literature have been deliberately omitted from the body of the text. Each chapter, as already mentioned, has a set of appended notes that help the reader to pursue related topics in more depth and to consult the specialized literature. It is here where attributions and historical remarks are made, and where citations to the literature — both textbook and research — appear.

The main text is preceded by a Prologue (Chapter 0), which is intended to place the book in proper perspective. In addition to other textbooks on

the subject, and information on software, it gives a detailed list of topics not treated in this book, but definitely belonging to the vast area of computational mathematics, and it provides ample references to relevant texts. A list of numerical analysis journals is also included.

The reader is expected to have a good background in calculus and advanced calculus. Some passages of the text require a modest degree of acquaintance with linear algebra, complex analysis, or differential equations. These passages, however, can easily be skipped, without loss of continuity, by a student who is not familiar with these subjects.

It is a pleasure to thank the publisher for his interest in this book and his cooperation in producing it. The author is also grateful to Soren Jensen and Manil Suri, who taught from this text, and to an anonymous reader; they all made many helpful suggestions on improving the presentation. He is particularly indebted to Professor Jensen for substantially helping in preparing the exercises to Chapter 7. The author further acknowledges assistance from Carl de Boor in preparing the notes to Chapter 2, and from Werner C. Rheinboldt for helping with the notes to Chapter 4. Last but not least, he owes a measure of gratitude to Connie Wilson for typing a preliminary version of the text, and to Adam Hammer for assisting the author with the more intricate aspects of LaTeX.

January, 1997 *Walter Gautschi*

CHAPTER 0

PROLOGUE

§0.1. **Overview.** *Numerical Analysis* is the branch of mathematics that provides tools and methods for solving mathematical problems in numerical form. The objective is to develop detailed computational procedures, capable of being implemented on electronic computers, and to study their performance characteristics. Related fields are *Scientific Computation*, which explores the application of numerical techniques and computer architectures to concrete problems arising in the sciences and engineering; *Complexity Theory*, which analyzes the number of "operations" and the amount of computer memory required to solve a problem; and *Parallel Computation*, which is concerned with organizing computational procedures in a manner that allows running various parts of the procedures simultaneously on different processors.

The problems dealt with in computational mathematics come from virtually all branches of pure and applied mathematics. There are computational aspects in number theory, combinatorics, abstract algebra, linear algebra, approximation theory, geometry, statistics, optimization, complex analysis, nonlinear equations, differential and other functional equations, and so on. It is clearly impossible to deal with all these topics in a single text of reasonable size. Indeed, the tendency today is to develop specialized texts dealing with one or the other of these topics. In the present text we concentrate on subject matters that are basic to problems in approximation theory, nonlinear equations, and differential equations. Accordingly, we have chapters on machine arithmetic, approximation and interpolation, numerical differentiation and integration, nonlinear equations, one-step and multistep methods for ordinary differential equations, and boundary value problems in ordinary differential equations. Important topics not covered in this text are computational number theory, algebra, and geometry; constructive methods in optimization and complex analysis; numerical linear algebra; and the numerical solution of problems involving partial differential equations and integral equations. Selected texts for these areas are listed in §0.3.

We now describe briefly the topics treated in this text. Chapter 1 deals with the basic facts of life regarding machine computation. It recognizes that, although present-day computers are extremely powerful in terms of computational speed, reliability, and amount of memory available, they are

less than ideal — unless supplemented by appropriate software — when it comes to the precision available, and accuracy attainable, in the execution of elementary arithmetic operations. This raises serious questions as to how arithmetic errors, either present in the input data of a problem or committed during the execution of a solution algorithm, affect the accuracy of the desired results. Concepts and tools required to answer such questions are the topic of this introductory chapter. In Chapter 2, the central theme is the approximation of functions by simpler functions, typically, polynomials and piecewise polynomial functions. Approximation in the sense of least squares provides an opportunity to introduce orthogonal polynomials, which are relevant also in connection with problems of numerical integration treated in Chapter 3. A large part of the chapter, however, deals with polynomial interpolation and associated error estimates, which are basic to many numerical procedures for integrating functions and differential equations. Also discussed briefly is inverse interpolation, an idea useful in solving equations.

First applications of interpolation theory are given in Chapter 3, where the tasks presented are the computation of derivatives and definite integrals. Although the formulae developed for derivatives are subject to the detrimental effects of machine arithmetic, they are useful, nevertheless, for purposes of discretizing differential operators. The treatment of numerical integration includes routine procedures, such as the trapezoidal and Simpson's rules, appropriate for well-behaved integrands, as well as the more sophisticated procedures based on Gaussian quadrature to deal with singularities. It is here where orthogonal polynomials reappear. The method of undetermined coefficients is another technique for developing integration formulae. It is applied in order to approximate general linear functionals, the Peano representation of linear functionals providing an important tool for estimating the error. The chapter ends with a discussion of extrapolation techniques; although applicable to more general problems, they are inserted here since the composite trapezoidal rule together with the Euler-Maclaurin formula provides the best-known application — Romberg integration.

Chapter 4 deals with iterative methods for solving nonlinear equations and systems thereof, the pièce de résistance being Newton's method. The emphasis here lies in the study of, and the tools necessary to analyze, convergence. The special case of algebraic equations is also briefly given attention.

Chapter 5 is the first of three chapters devoted to the numerical solution of ordinary differential equations. It concerns itself with one-step methods for solving initial value problems, such as the Runge-Kutta method, and gives a detailed analysis of local and global errors. Also included is a brief

introduction to stiff equations and special methods to deal with them. Multistep methods and, in particular, Dahlquist's theory of stability and its applications, is the subject of Chapter 6. The final Chapter 7 is devoted to boundary value problems and their solution by shooting methods, finite difference techniques, and variational methods.

§0.2. **Numerical analysis software**. There are many software packages available, both in the public domain and distributed commercially, that deal with numerical analysis algorithms. A widely used source of numerical software is Netlib, which can be accessed either on the World Wide Web at the URL `http://www.netlib.org`, or by e-mail at the address (in the US) `netlib@netlib.org`.

If you wish to see an index of all packages contained in the Netlib repository, you may, on the Web, open up the highlighted entry `Browse the Netlib repository`. This will display on the screen a list of all software packages, which may then be examined more closely by clicking on `A more descriptive version of the list`. Instructions as to how files are retrieved can be found under `Frequently Asked Questions about Netlib (FAQ)` on the home page of Netlib. Via e-mail, you can get an index, as well as instructions regarding retrieval of an individual package or file(s) from a package, by sending the message `send index` to the e-mail address given here.

Large collections of general-purpose numerical algorithms are contained in sources such as Slatec and TOMS (ACM Transactions on Mathematical Software). Specialized packages relevant to the topics in the chapters ahead are identified in the "Notes" to each chapter. Likewise, specific files needed to do some of the machine assignments in the Exercises are identified as part of the exercise.

Among the commercial software packages we mention the visual numerics (formerly IMSL) package, the NAG library, and MLAB (Modeling LABoratory). Interactive systems include HiQ, MACSYMA, MAPLE, Mathcad, Mathematica, and MATLAB. Many of these packages, in addition to numerical computation, have symbolic computation and graphics capabilities. Further information is available in the Netlib file `commercial`. For more libraries, and for the interactive systems, also see Lozier and Olver [1994, §3].

§0.3. **Textbooks and monographs**. We provide here an annotated list (ordered alphabetically with respect to authors) of other textbooks on nu-

merical analysis, written at about the same, or higher, level as the present one. Following this we also mention books and monographs dealing with topics in computational mathematics not covered in our (and many other) books on numerical analysis. Additional books dealing with specialized subject areas, as well as other literature, are referenced in the "Notes" to the individual chapters. We generally restrict ourselves to books written in English and, with a few exceptions, published within the last 15 years or so. Even so, we have had to be selective. (No value judgment is to be implied by our selections or omissions.) A reader with access to the AMS (American Mathematical Society) MathSci CD-ROM will have no difficulty in retrieving a more complete list of relevant items, including older texts.

Selected Textbooks on Numerical Analysis

Atkinson [1989]

A comprehensive in-depth treatment of standard topics short of partial differential equations; includes an appendix describing some of the better-known software packages.

Bruce, Giblin, and Rippon [1990]

A collection of interesting mathematical problems, ranging from number theory and computer-aided design to differential equations, that require the use of computers for their solution.

Cheney and Kincaid [1994]

Although an undergraduate text, it covers a broad area, has many examples from science and engineering as well as computer programs; there are many exercises, including machine assignments.

Conte and de Boor [1980]

A widely used text for upper-division undergraduate students; written for a broad audience, with algorithmic concerns in the foreground; has FORTRAN subroutines for many algorithms discussed in the text.

Deuflhard and Hohmann [1995]

An introductory text with emphasis on machine computation and algorithms; includes a discussion of three-term recurrence relations (not usually found in textbooks), but no differential equations.

Fröberg [1985]

A thorough and exceptionally lucid exposition of all major topics of numerical analysis exclusive of algorithms and computer programs.

Hämmerlin and Hoffmann [1991]

Similar to Stoer and Bulirsch [1993] in its emphasis on mathematical theory; has more on approximation theory and multivariate interpolation and integration, but nothing on differential equations.

Isaacson and Keller [1994]

One of the older but still eminently readable texts, stressing the mathematical analysis of numerical methods.

Kincaid and Cheney [1996]

Related to Cheney and Kincaid [1985] but more mathematically oriented and unusually rich in exercises and bibliographic items.

Rutishauser [1990]

An annotated translation from the German of an older text based on posthumous notes by one of the pioneers of numerical analysis; although the subject matter reflects the state of the art in the early 1970s, the treatment is highly original and is supplemented by translator's notes to each chapter pointing to more recent developments.

Schwarz [1989]

A mathematically oriented treatment of all major areas of numerical analysis, including ordinary and partial differential equations.

Stoer and Bulirsch [1993]

Fairly comprehensive in coverage; written in a style appealing more to mathematicians than engineers and computer scientists; has many exercises and bibliographic references; serves not only as a textbook, but also as a reference work.

Todd [1980, 1977]

Rather unique books, emphasizing problem solving in areas often not covered in other books on numerical analysis.

A collection of outstanding survey papers on specialized topics in numerical analysis is being assembled by Ciarlet and Lions [1990,1991,1994] in handbooks of numerical analysis; three volumes have appeared so far. Another source of surveys on a variety of topics is *Acta numerica*, an annual series of books edited by Iserles [1992–1996], of which five volumes have so far been published. For an authoritative account of the history of numerical analysis, the reader is referred to the book by Goldstine [1977].

The related areas of *Scientific Computing* and *Parallel Computing* are rather more recent fields of study, and currently are most actively pursued in proceedings of conferences and workshops. Nevertheless, a few textbooks have also appeared, notably in the linear algebra context and in connection with ordinary and partial differential equations, for example Schendel [1984], Ortega and Voigt [1985], Ortega [1989], Golub and Ortega [1992], [1993], Van de Velde [1994], and Heath [1997], but also in optimization, Pardalos, Phillips, and Rosen [1992], computational geometry, Akl and Lyons [1993], and other miscellaneous areas, Crandall [1994], Köckler [1994], and Bellomo and Preziosi [1995]. Interesting historical essays are contained in Nash [1990]. Matters regarding the *Complexity* of numerical algorithms are discussed in an abstract framework in books by Traub and Woźniakowski [1980] and Traub, Wasilkowski, and Woźniakowski [1983], [1988], with applications to the numerical integration of functions and nonlinear equations, and similarly, applied to elliptic partial differential equations and integral equations, in the book by Werschulz [1991]. Other treatises are those by Kronsjö [1987], Ko [1991], Bini and Pan [1994], and Wang, Xu, and Gao [1994]. For an in-depth complexity analysis of Newton's method, the reader is encouraged to study Smale's [1987] lecture.

Material on *Computational Number Theory* can be found, at the undergraduate level, in the book by Rosen [1993], which also contains applications to cryptography and computer science, and in Allenby and Redfern [1989], and at a more advanced level in the books by Niven, Zuckerman, and Montgomery [1991], Cohen [1993], and Bach and Shallit [1996], the first volume of a projected two-volume set. Computational methods of factorization are dealt with in the book by Riesel [1994]. Other useful sources are the set of lecture notes by Pohst [1993] on algebraic number theory algorithms, and the proceedings volumes edited by Pomerance [1990] and Gautschi [1994a, Part II]. For algorithms in *Combinatorics*, see the books by Nijenhuis and Wilf [1978], Hu [1982], and Cormen, Leiserson, and Rivest [1990]. Various aspects of *Computer Algebra* are treated in the books by Cox, Little,

and O'Shea [1992], Geddes, Czapor, and Labahn [1992], Mignotte [1992], Davenport, Siret, and Tournier [1993], Heck [1996], and Mishra [1993].

Other relatively new disciplines are *Computational Geometry* and *Computer-Aided Design*, for which relevant texts are Preparata and Shamos [1985], Edelsbrunner [1987], Mäntylä [1988], and Taylor [1992]; and Farin [1995], [1997] and Hoschek and Lasser [1993], respectively. *Statistical Computing* is covered in general textbooks such as Kennedy and Gentle [1980], Anscombe [1981], Maindonald [1984], and Thisted [1988]. More specialized texts are Devroye [1986] on the generation of nonuniform random variables, Späth [1992] on regression analysis, Heiberger [1989] on the design of experiments, Stewart [1994] on Markov chains, and Fang and Wang [1994] on the application of number-theoretic methods. Numerical techniques in *Optimization* (including optimal control problems) are discussed in Evtushenko [1985]. An introductory book on unconstrained optimization is Wolfe [1978]; among the more advanced and broader texts on optimization techniques we mention Gill, Murray, and Wright [1981], Fletcher [1987], and Ciarlet [1989]. Linear programming is treated in Nazareth [1987] and Panik [1996], linear and quadratic problems in Sima [1996], and the application of conjugate direction methods to problems in optimization in Hestenes [1980]. The most comprehensive text on (numerical and applied) *Complex Analysis* is the three-volume treatise by Henrici [1988, 1991, 1986]. Numerical methods for conformal mapping are also treated in Schinzinger and Laura [1991]. For approximation in the complex domain, the standard text is Gaier [1987]; Stenger [1993] deals with approximation by sinc functions. The book by Iserles and Nørsett [1991] contains interesting discussions on the interface between complex rational approximation and the stability theory of discretized differential equations. The impact of high-precision computation on problems and conjectures involving complex approximation is beautifully illustrated in the set of lectures by Varga [1990].

For an in-depth treatment of many of the preceding topics, also see the three-volume work of Knuth [1975, 1981, 1973].

Perhaps the most significant topic omitted in our book is numerical linear algebra and its application to solving partial differential equations by finite difference or finite element methods. Fortunately, there are good treatises available that address these areas. For *Numerical Linear Algebra*, we refer to the book by Golub and Van Loan [1996] and the classic work of Wilkinson [1988]. Other general texts are Watkins [1991], Jennings and McKeown [1992], and Datta [1995]; Higham [1996] has a comprehensive treatment of error and stability analyses. The solution of sparse linear sys-

tems, and the special data structures and pivoting strategies required in direct methods, are treated in Østerby and Zlatev [1983], Duff, Erisman, and Reid [1989], and Zlatev [1991], whereas iterative techniques are discussed in the classic texts by Varga [1962] and Young [1971], and more recently in Hageman and Young [1981], Il'in [1992], Hackbusch [1994], Saad [1996], and Weiss [1996]. The books by Branham [1990] and Björck [1996] are devoted especially to least squares problems. For eigenvalues, see Chatelin [1983], [1993], and for a good introduction to the numerical analysis of symmetric eigenvalue problems, see Parlett [1980]. The currently very active investigation of large sparse symmetric and nonsymmetric eigenvalue problems and their solution by Lanczos-type methods has given rise to many books, for example, Cullum and Willoughby [1985], Meyer [1987], Sehmi [1989], and Saad [1992]. For readers wishing to test their algorithms on specific matrices, the collection of test matrices in Gregory and Karney [1978], and the recently established "matrix market" on the Web (`http://math.nist.gov./MatrixMarket`), are useful sources.

Even more extensive is the textbook literature on the numerical solution of *Partial Differential Equations.* The field has grown so much that there are currently only a few books that attempt to cover the subject as a whole. Among these are Birkhoff and Lynch [1984] (for elliptic problems), Sewell [1988], Hall and Porsching [1990], Ames [1992], Celia and Gray [1992], Morton and Mayers [1994], and Quarteroni and Valli [1994]. Variational and finite element methods seem to have attracted the most attention. An early and still frequently cited reference is the book by Ciarlet [1978]; among the more recent texts we mention the Texas Finite Element Series (Becker, Carey, and Oden [1981], Carey and Oden [1983], [1984], [1986], Oden [1983], Oden and Carey [1984]), Axelsson and Barker [1984], Wait and Mitchell [1985], Sewell [1985] (with a slant toward software), White [1985], Girault and Raviart [1986] (focusing on Navier-Stokes equations), Burnett [1987], Hughes [1987], Johnson [1987], Schwarz [1988], Beltzer [1990] (using symbolic computation), Křížek and Neittaanmäki [1990], Brezzi and Fortin [1991], and Brenner and Scott [1994]. Finite difference methods are treated in Godunov and Ryaben'kiĭ [1987], Strikwerda [1989], Ashyralyev and Sobolevskiĭ [1994], Gustafsson, Kreiss, and Oliger [1995], and Thomas [1995], the method of lines in Schiesser [1991], and the more refined techniques of multigrids and domain decomposition in Hackbusch [1985], Briggs [1987], McCormick [1989], [1992], Bramble [1993], Shaĭdurov [1995], and Smith, Bjørstad, and Gropp [1996]. Problems in potential theory and elasticity are often approached via boundary element methods, for which represen-

tative texts are Banerjee and Butterfield [1981], Brebbia [1984], Hartmann [1989], Chen and Zhou [1992], and Hall [1994]. A discussion of conservation laws is given in the classic monograph by Lax [1973] and more recently in LeVeque [1992] and Godlewski and Raviart [1996]. Spectral methods (i.e., expansions in (typically) orthogonal polynomials), applied to a variety of problems, were pioneered in the monograph by Gottlieb and Orszag [1977] and have received extensive treatments in more recent texts by Canuto, Hussaini, Quarteroni, and Zang [1988], Mercier [1989], and Fornberg [1996]. The numerical solution of elliptic boundary value problems nowadays is greatly facilitated thanks to the software package ELLPACK, which is described in Rice and Boisvert [1985].

Early, but still relevant, texts on the numerical solution of *Integral Equations* are Atkinson [1976] and Baker [1977]. A more recent introduction to the subject is Delves and Mohamed [1988]. Volterra integral equations are discussed in Linz [1985] and, more extensively, in Brunner and van der Houwen [1986], whereas singular integral equations are the subject of Prössdorf and Silbermann [1991].

§0.4. **Journals**. Here we list the major journals (in alphabetical order) covering the areas of numerical analysis and mathematical software.

ACM Transactions on Mathematical Software
Applied Numerical Mathematics
BIT Numerical Mathematics
Calcolo
Chinese Journal of Numerical Mathematics and Applications
Computational Mathematics and Mathematical Physics
Computing
IMA Journal on Numerical Analysis
Journal of Computational and Applied Mathematics
Mathematical Modelling and Numerical Analysis
Mathematics of Computation
Numerische Mathematik
SIAM Journal on Numerical Analysis

CHAPTER 1

MACHINE ARITHMETIC AND RELATED MATTERS

The questions addressed in this introductory chapter are fundamental in the sense that they are relevant in any situation that involves numerical machine computation, regardless of the kind of problem that gave rise to these computations. In the first place, one has to be aware of the rather primitive type of number system available on computers. It is basically a finite system of numbers of finite length, thus a far cry from the idealistic number system familiar to us from mathematical analysis. The passage from a real number to a machine number entails *rounding*, and thus small errors, called *roundoff errors*. Additional errors are introduced when the individual arithmetic operations are carried out on the computer. In themselves, these errors are harmless, but acting in concert and propagating through a lengthy computation, they can have significant — even disastrous — effects.

Most problems involve input data not representable exactly on the computer. Therefore, even before the solution process starts, simply by storing the input in computer memory, the problem is already slightly perturbed, owing to the necessity of rounding the input. It is important, then, to estimate how such small perturbations in the input affect the output, the solution of the problem. This is the question of the (numerical) *condition of a problem*: the problem is called well-conditioned if the changes in the solution of the problem are of the same order of magnitude as the perturbations in the input that caused those changes. If, on the other hand, they are much larger, the problem is called ill-conditioned. It is desirable to measure by a single number — the *condition number* of the problem — the extent to which the solution is sensitive to perturbations in the input. The larger this number, the more ill-conditioned the problem.

Once the solution process starts, additional rounding errors will be committed, which also contaminate the solution. The resulting errors, in contrast to those caused by input errors, depend on the particular solution algorithm. It makes sense, therefore, to also talk about the *condition of an algorithm*, although its analysis is usually quite a bit harder. The quality of the computed solution is then determined by both (essentially the product of) the condition of the problem and the condition of the algorithm.

§1. Real Numbers, Machine Numbers, and Rounding

We begin with the number system commonly used in mathematical analysis and confront it with the more primitive number system available to us on any particular computer. We identify the basic constant (the machine precision) that determines the level of precision attainable on such a computer.

§1.1. **Real numbers.** One can introduce real numbers in many different ways. Mathematicians favor the axiomatic approach, which leads them to define the set of real numbers as a "complete Archimedean ordered field." Here we adopt a more pedestrian attitude and consider the set of real numbers $\mathbb{R}$ to consist of positive and negative numbers represented in some appropriate number system and manipulated in the usual manner known from elementary arithmetic. We adopt here the *binary number system*, since it is the one most commonly used on computers. Thus,

$$x \in \mathbb{R} \text{ iff } x = \pm\,(b_n 2^n + b_{n-1}2^{n-1} + \cdots + b_0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots). \quad (1.1)$$

Here $n \geq 0$ is some integer, and the "binary digits" b_i are either 0 or 1,

$$b_i = 0 \text{ or } b_i = 1 \text{ for all } i. \quad (1.2)$$

It is important to note that in general we need infinitely many binary digits to represent a real number. We conveniently write such a number in the abbreviated form (familiar from the decimal number system)

$$x = \pm\,(b_n b_{n-1} \cdots b_0 \,.\, b_{-1}b_{-2}b_{-3}\cdots)_2, \quad (1.3)$$

where the subscript 2 at the end is to remind us that we are dealing with a binary number. (Without this subscript, the number could also be read as a decimal number, which would be a source of ambiguity.) The dot in (1.3) — appropriately called the binary point — separates the integer part on the left from the fractional part on the right. Note that the representation (1.3) is not unique; for example, $(.011\overline{1}\ldots)_2 = (.1)_2$. We regain uniqueness if we always insist on a finite representation, if one exists.

Examples.

(1) $(10011.01)_2 = 2^4 + 2^1 + 2^0 + 2^{-2} = 16 + 2 + 1 + \frac{1}{4} = (19.25)_{10}$

(2) $(.0101\overline{01}\ldots)_2 = \sum_{\substack{k=2 \\ (k \text{ even})}}^{\infty} 2^{-k} = \sum_{m=1}^{\infty} 2^{-2m} = \frac{1}{4}\sum_{m=0}^{\infty}\left(\frac{1}{4}\right)^m$

$= \frac{1}{4}\,\frac{1}{1-\frac{1}{4}} = \frac{1}{3} = (.33\overline{3}\ldots)_{10}$

(3) $\frac{1}{5} = (.2)_{10} = (.0011\overline{0011}\ldots)_2$

To determine the binary digits on the right, one keeps multiplying by 2 and observing the integer part in the result; if it is zero, the binary digit in question is 0, otherwise 1. In the latter case, the integral part is removed and the process repeated.

The last example is of interest insofar as it shows that to a finite decimal number there may correspond a (nontrivial) infinite binary representation. One cannot assume, therefore, that a finite decimal number is exactly representable on a binary computer. Conversely, however, to a finite binary number there always corresponds a finite decimal representation. (Why?)

§1.2. **Machine numbers.** There are two kinds of machine numbers: floating-point and fixed-point. The first corresponds to the "scientific notation" in the decimal system, whereby a number is written as a decimal fraction times an integral power of 10. The second allows only for fractions. On a binary computer, one consistently uses powers of 2 instead of 10. More important, the number of binary digits, both in the fraction and in the exponent of 2 (if any), is finite and cannot exceed certain limits that are characteristics of the particular computer at hand.

(a) *Floating-point numbers.* We denote by t the number of binary digits allowed by the computer in the fractional part, and by s the number of binary digits in the exponent. Then the set of (real) floating-point numbers on that computer will be denoted by $\mathbb{R}(t, s)$. Thus,

$$x \in \mathbb{R}(t, s) \quad \text{iff} \quad x = f \cdot 2^e, \tag{1.4}$$

where, in the notation of (1.3),

$$f = \pm\,(.\ b_{-1}b_{-2}\cdots b_{-t})_2, \quad e = \pm\,(c_{s-1}c_{s-2}\cdots c_0.)_2. \tag{1.5}$$

Here all b_i and c_j are binary digits, that is, either zero or one. The binary fraction f is usually referred to as the *mantissa* of x, and the integer e as the *exponent* of x. The number x in (1.4) is said to be *normalized* if in its fraction f we have $b_{-1} = 1$. We assume that all numbers in $\mathbb{R}(t, s)$ are normalized (with the exception of $x = 0$, which is treated as a special number). If $x \neq 0$ were not normalized, we could multiply f by an appropriate power of 2, to normalize it, and adjust the exponent accordingly. This is always possible as long as the adjusted exponent is still in the admissible range.

We can think of a floating-point number (1.4) as being accommodated in a machine register as shown in Figure 1.1.1. The figure does not quite correspond to reality, but is close enough to it for our purposes.

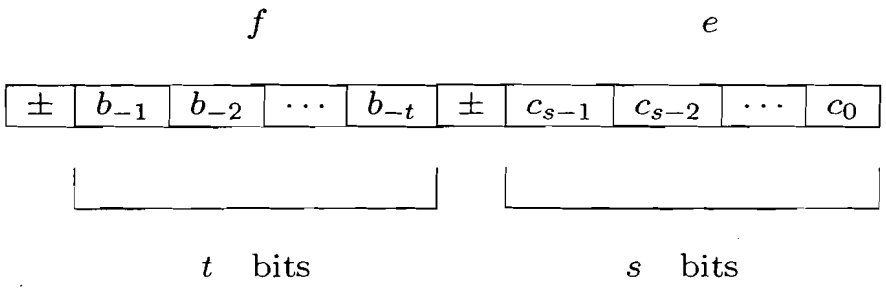


FIGURE 1.1.1. Packing of a floating-point number in a machine register

Note that the set (1.4) of normalized floating-point numbers is finite, and is thus represented by a finite set of points on the real line. What is worse, these points are not uniformly distributed (cf. Ex. 1). This, then, is all we have to work with!

It is immediately clear from (1.4) and (1.5) that the largest and smallest magnitude of a (normalized) floating-point number is given, respectively, by

$$\max_{x \in \mathbb{R}(t,s)} |x| = (1 - 2^{-t})\, 2^{2^s - 1}, \quad \min_{x \in \mathbb{R}(t,s)} |x| = 2^{-2^s}. \tag{1.6}$$

On a SUN SPARC workstation, for example, one has $t = 23$, $s = 7$, so that the maximum and minimum in (1.6) are 1.70×10^{38} and 2.94×10^{-39}, respectively. (Because of an asymmetric internal hardware representation of the exponent on these computers, the true range of floating-point numbers is slightly shifted, more like from 1.18×10^{-38} to 3.40×10^{38}.)

A real nonzero number whose modulus is not in the range determined by (1.6) cannot be represented on this particular computer. If such a number is produced during the course of a computation, one says that *overflow* has

occurred if its modulus is larger than the maximum in (1.6), and *underflow* if it is smaller than the minimum in (1.6). The occurrence of overflow is fatal, and the machine (or its operating system) usually prompts the computation to be interrupted. Underflow is less serious, and one may get away with replacing the delinquent number by zero. However, this is not foolproof. Imagine that at the next step the number that underflowed is to be multiplied by a huge number. If the replacement by zero has been made, the result will always be zero.

In order to increase the precision, one can use two machine registers to represent a machine number. In effect, one then embeds $\mathbb{R}(t,s) \subset \mathbb{R}(2t,s)$, and calls $x \in \mathbb{R}(2t,s)$ a *double-precision* number.

(b) *Fixed-point numbers.* This is the case (1.4) where $e = 0$. That is, fixed-point numbers are binary fractions, $x = f$, hence $|f| < 1$. We can therefore only deal with numbers that are in the interval (–1,1). This, in particular, requires extensive scaling and rescaling to make sure that all initial data, as well as all intermediate and final results, lie in that interval. Such a complication can only be justified in special circumstances where machine time and/or precision are at a premium. Note that on the same computer as considered before, we do not need to allocate space for the exponent in the machine register, and thus have in effect $s+t$ binary digits available for the fraction f, hence more precision; cf. Figure 1.1.2.

$\pm$	b_{-1}	b_{-2}	$\cdots$	b_{-t}	$b_{-(t+1)}$	$\cdots$	$b_{-(t+s)}$

FIGURE 1.1.2. Packing of a fixed-point number in a machine register

(c) *Other data structures for numbers.* Complex floating-point numbers consist of pairs of real floating-point numbers, the first of the pair representing the real part and the second the imaginary part. To avoid rounding errors in arithmetic operations altogether, one can employ rational arithmetic, in which each (rational) number is represented by a pair of extended-precision integers — the numerator and denominator of the rational number. The Euclidean algorithm is used to remove common factors. A device that allows keeping track of error propagation and the influence of data errors is interval arithmetic involving intervals guaranteed to contain the desired numbers. In complex arithmetic one employs rectangular or circular domains.

§1.3. **Rounding**. A machine register acts much like the infamous Procrustes bed in Greek mythology. Procrustes was the innkeeper whose inn

had only beds of one size. If a fellow came along who was too tall to fit into his beds, he cut off his feet. If the fellow was too short, he stretched him. In the same way, if a real number comes along that is too long, its tail end (not the head!) is cut off; if it is too short, it is padded by zeros at the end.

More specifically, let

$$x \in \mathbb{R}, \quad x = \pm \left(\sum_{k=1}^{\infty} b_{-k} 2^{-k} \right) 2^e \tag{1.7}$$

be the "exact" real number (in normalized floating-point form), and

$$x^* \in \mathbb{R}(t,s), \quad x^* = \pm \left(\sum_{k=1}^{t} b^*_{-k} 2^{-k} \right) 2^{e^*} \tag{1.8}$$

the rounded number. One then distinguishes between two methods of rounding, the first being Procrustes' method.

(a) *Chopping.* One takes

$$x^* = \mathrm{chop}(x), \quad e^* = e, \quad b^*_{-k} = b_{-k} \quad \text{for} \quad k = 1, 2, \ldots, t. \tag{1.9}$$

(b) *Symmetric rounding.* This corresponds to the familiar rounding up or rounding down in decimal arithmetic, based on the first discarded decimal digit: if it is larger than or equal to 5, one rounds up; if it is less than 5, one rounds down. In binary arithmetic, the procedure is somewhat simpler, since there are only two possibilities: either the first discarded binary digit is 1, in which case one rounds up, or it is 0, in which case one rounds down. We can write the procedure very simply in terms of the chop operation in (1.9):

$$x^* = \mathrm{rd}(x), \quad \mathrm{rd}(x) := \mathrm{chop}\left(x + \frac{1}{2} \cdot 2^{-t} \cdot 2^e \right). \tag{1.10}$$

There is a small error incurred in rounding, which is most easily estimated in the case of chopping. Here the *absolute error* $|x - x^*|$ is

$$\begin{aligned} |x - \mathrm{chop}(x)| &= \left| \pm \sum_{k=t+1}^{\infty} b_{-k} 2^{-k} \right| 2^e \\ &\leq \sum_{k=t+1}^{\infty} 2^{-k} \cdot 2^e = 2^{-t} \cdot 2^e. \end{aligned}$$

It depends on e (i.e., the magnitude of x), which is the reason why one prefers the *relative error* $|(x - x^*)/x|$ (if $x \neq 0$), which, for normalized x, can be estimated as

$$\left|\frac{x - \mathrm{chop}(x)}{x}\right| \leq \frac{2^{-t} \cdot 2^e}{\left|\pm\sum_{k=1}^{\infty} b_{-k}2^{-k}\right| 2^e} \leq \frac{2^{-t} \cdot 2^e}{\frac{1}{2} \cdot 2^e} = 2 \cdot 2^{-t}. \tag{1.11}$$

Similarly, in the case of symmetric rounding, one finds (cf. Ex. 6)

$$\left|\frac{x - \mathrm{rd}(x)}{x}\right| \leq 2^{-t}. \tag{1.12}$$

The number on the right is an important, machine-dependent quantity, called the *machine precision*,

$$\mathrm{eps} = 2^{-t}; \tag{1.13}$$

it determines the level of precision of any large-scale floating-point computation. On the SUN SPARC workstation, where $t = 23$, we have $\mathrm{eps} \approx 1.19 \times 10^{-7}$, corresponding to a precision of 6 to 7 significant decimal digits.

Since it is awkward to work with inequalities, one prefers writing (1.12) equivalently as an equality,

$$\mathrm{rd}(x) = x(1 + \varepsilon), \quad .|\varepsilon| \leq \mathrm{eps}, \tag{1.14}$$

and defers dealing with the inequality (for ε) to the very end.

§2. Machine Arithmetic

The arithmetic used on computers unfortunately does not respect the laws of ordinary arithmetic. Each elementary floating-point operation, in general, generates a small error that may then propagate through subsequent machine operations. As a rule, this error propagation is harmless, except in the case of subtraction, where cancellation effects may seriously compromise the accuracy of the results.

§2.1. **A model of machine arithmetic.** Any of the four basic arithmetic operations, when applied to two machine numbers, may produce a

result no longer representable on the computer. We have therefore errors also associated with arithmetic operations. Barring the occurrence of overflow or underflow, we may assume as a *model of machine arithmetic* that each arithmetic operation $\circ$ $(= +, -, \times, /)$ produces a correctly rounded result. Thus, if x, $y \in \mathbb{R}(t, s)$ are floating-point machine numbers, and $\mathrm{fl}(x \circ y)$ denotes the machine-produced result of the arithmetic operation $x \circ y$, then

$$\mathrm{fl}(x \circ y) = x \circ y\,(1 + \varepsilon), \quad |\varepsilon| \leq \mathrm{eps}. \tag{2.1}$$

This can be interpreted in a number of ways; for example, in the case of multiplication,

$$\mathrm{fl}(x \times y) = [x(1+\varepsilon)] \times y = x \times [y(1+\varepsilon)] = (x\sqrt{1+\varepsilon}) \times (y\sqrt{1+\varepsilon}) = \cdots .$$

In each equation we identify the computed result as the exact result on data that are slightly perturbed, whereby the respective relative perturbations can be estimated, for example, by $|\varepsilon| \leq \mathrm{eps}$ in the first two equations, and $\sqrt{1+\epsilon} \approx 1 + \frac{1}{2}\varepsilon$, $\left|\frac{1}{2}\varepsilon\right| \leq \frac{1}{2}\mathrm{eps}$ in the third. These are elementary examples of *backward error analysis*, a powerful tool for estimating errors in machine computation.

Even though a single arithmetic operation causes a small error that can be neglected, a succession of arithmetic operations can well result in a significant error, owing to *error propagation*. It is like the small microorganisms that we all carry in our bodies: if our defense mechanism is in good order, the microorganisms cause no harm, in spite of their large presence. If for some reason our defenses are weakened, then all of a sudden they can play havoc with our health. The same is true in machine computation: the rounding errors, although widespread, will cause little harm unless our computations contain some weak spots that allow rounding errors to take over to the point of completely invalidating the results. We learn about one such weak spot (indeed the only one) in the next subsection.[1]

[1]Rounding errors can also have significant implications in real life. One example, taken from politics, concerns the problem of apportionment: how should the representatives in an assembly, such as the US House of Representatives or the Electoral College, be constituted to fairly reflect the size of population in the various states? If the total number of representatives in the assembly is given, say, A, the total population of the US is P, and the population of State i is p_i, then State i should be allocated

$$r_i = \frac{p_i}{P} A$$

representatives. The problem is that r_i is not an integer, in general. How then should r_i

§2.2. **Error propagation in arithmetic operations; cancellation error.** We now study the extent to which the basic arithmetic operations propagate errors already present in their operands. Previously, in §2.1, we assumed the operands to be exact machine-representable numbers and discussed the errors due to imperfect execution of the arithmetic operations by the computer. We now change our viewpoint and assume that the operands themselves are contaminated by errors, but the arithmetic operations are carried out exactly. (We already know what to do, cf. (2.1), when we are dealing with machine operations.) Our interest is in the errors in the results caused by errors in the data.

(a) *Multiplication.* We consider values $x(1+\varepsilon_x)$ and $y(1+\varepsilon_y)$ of x and y contaminated by relative errors ε_x and ε_y, respectively. What is the relative error in the product? We assume ε_x, ε_y sufficiently small so that quantities of second order, ε_x^2, $\varepsilon_x\varepsilon_y$, ε_y^2 — and even more so, quantities of still higher order — can be neglected against the epsilons themselves. Then

$$x(1+\varepsilon_x)\cdot y(1+\varepsilon_y) = x\cdot y\,(1+\varepsilon_x+\varepsilon_y+\varepsilon_x\varepsilon_y) \approx x\cdot y\,(1+\varepsilon_x+\varepsilon_y).$$

Thus, the relative error $\varepsilon_{x\cdot y}$ in the product is given (at least approximately) by

$$\varepsilon_{x\cdot y} = \varepsilon_x + \varepsilon_y; \tag{2.2}$$

that is, the (relative) errors in the data are being added to produce the (relative) error in the result. We consider this to be acceptable error propagation, and in this sense, multiplication is a *benign* operation.

(b) *Division.* Here we have similarly (if $y \neq 0$)

$$\begin{aligned}\frac{x(1+\varepsilon_x)}{y(1+\varepsilon_y)} &= \frac{x}{y}(1+\varepsilon_x)(1-\varepsilon_y+\varepsilon_y^2-+\cdots)\\ &\approx \frac{x}{y}(1+\varepsilon_x-\varepsilon_y);\end{aligned}$$

---

be rounded to an integer r_i^*? One can think of three natural criteria to be imposed: (i) r_i^* should be one of the two integers closest to r_i ("quota condition"). (ii) If A is increased, all other things being the same, then r_i^* should not decrease ("house monotonicity"). (iii) If p_i is increased, the other p_j remaining constant, then r_i^* should not decrease ("population monotonicity"). Unfortunately, there is no apportionment method that satisfies all three criteria. There is indeed a case in US history when Samuel J. Tilden lost his bid for the presidency in 1876 in favor of Rutherford B. Hayes, purely on the basis of the apportionment method adopted on that occasion (which, incidentally, was not the one prescribed by law at the time).

that is,

$$\varepsilon_{x/y} = \varepsilon_x - \varepsilon_y. \tag{2.3}$$

Also division is a benign operation.

(c) *Addition and subtraction.* Since x and y can be numbers of arbitrary signs, it suffices to look at addition. We have

$$\begin{aligned} x(1+\varepsilon_x) + y(1+\varepsilon_y) &= x + y + x\varepsilon_x + y\varepsilon_y \\ &= (x+y)\left(1 + \frac{x\varepsilon_x + y\varepsilon_y}{x+y}\right), \end{aligned}$$

assuming $x + y \neq 0$. Therefore,

$$\varepsilon_{x+y} = \frac{x}{x+y}\varepsilon_x + \frac{y}{x+y}\varepsilon_y. \tag{2.4}$$

As before, the error in the result is a linear combination of the errors in the data, but now the coefficients are no longer ± 1 but can assume values that are arbitrarily large. Note first, however, that when x and y have the same sign, then both coefficients are positive and bounded by 1, so that

$$|\varepsilon_{x+y}| \leq |\varepsilon_x| + |\varepsilon_y| \quad (x \cdot y > 0); \tag{2.5}$$

addition, in this case, is again a benign operation. It is only when x and y have opposite signs that the coefficients in (2.4) can be arbitrarily large, namely, when $|x + y|$ is arbitrarily small compared to $|x|$ and $|y|$. This happens when x and y are almost equal in absolute value, but opposite in sign. The large magnification of error then occurring in (2.4) is referred to as *cancellation error.* It is the only serious weakness — the Achilles heel, as it were — of numerical computation, and it should be avoided whenever possible. In particular, one should be prepared to encounter cancellation effects not only in single devastating amounts, but also repeatedly over a long period of time involving "small doses" of cancellation. Either way, the end result can be disastrous.

We illustrate the cancellation phenomenon schematically in Figure 1.2.1, where b, b', b'' stand for binary digits that are reliable, and the gs represent binary digits contaminated by error; these are often called "garbage" digits. Note in Figure 1.2.1 that "garbage – garbage = garbage," but, more important, that the final normalization of the result moves the first garbage digit from the 12th position to the 3rd.

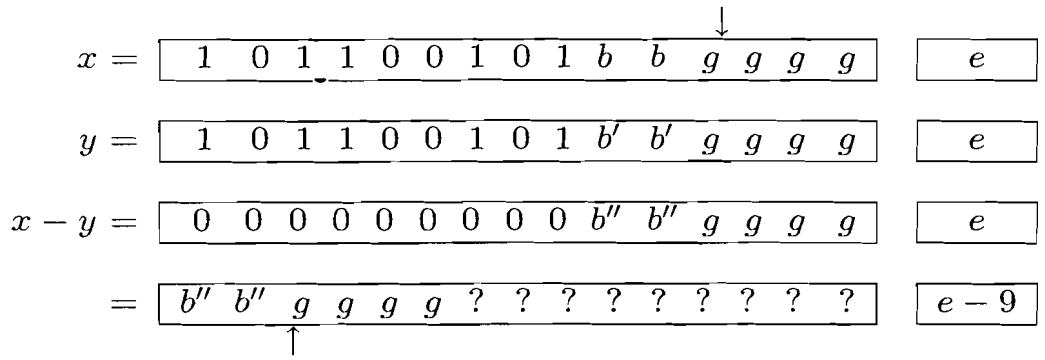


FIGURE 1.2.1. The cancellation phenomenon

Cancellation is such a serious matter that we wish to give a number of elementary examples, not only of its occurrence, but also of how it might be avoided.

Examples.

(1) An algebraic identity: $(a-b)^2 = a^2-2ab+b^2$. Although this is a valid identity in algebra, it is no longer valid in machine arithmetic. Thus, on a 2-decimal-digit computer, with $a = 1.8$, $b = 1.7$, we get, using symmetric rounding,

$$\text{fl}(a^2 - 2ab + b^2) = 3.2 - 6.2 + 2.9 = -.10$$

instead of the true result .010, which we obtain also on our 2-digit computer if we use the left-hand side of the identity. The expanded form of the square thus produces a result which is off by one order of magnitude and on top has the wrong sign!

(2) Quadratic equation: $x^2 - 56x + 1 = 0$. The usual formula for a quadratic gives, in 5-decimal arithmetic,

$$x_1 = 28 - \sqrt{783} = 28 - 27.982 = .018000,$$

$$x_2 = 28 + \sqrt{783} = 28 + 27.982 = 55.982.$$

This should be contrasted with the exact roots .0178628... and 55.982137.... As can be seen, the smaller of the two is obtained to only two correct decimal digits, owing to cancellation. An easy way out, of course, is to compute x_2 first, which involves a benign addition, and then to compute $x_1 = 1/x_2$ by Vieta's formula, which again involves a benign operation — division. In this way we obtain both roots to full machine accuracy.

(3) Compute $y = \sqrt{x+\delta} - \sqrt{x}$, where $x > 0$ and $|\delta|$ is very small. Clearly, the formula as written causes severe cancellation errors, since each square root has to be rounded. Writing instead

$$y = \frac{\delta}{\sqrt{x+\delta} + \sqrt{x}}$$

completely removes the problem.

(4) Compute $y = \cos(x+\delta) - \cos x$, where $|\delta|$ is very small. Here cancellation can be avoided by writing y in the equivalent form

$$y = -2\sin\frac{\delta}{2}\sin\left(x + \frac{\delta}{2}\right).$$

(5) Compute $y = f(x+\delta) - f(x)$, where $|\delta|$ is very small and f a given function. Special tricks, such as those used in the two preceding examples, can no longer be played, but if f is sufficiently smooth in the neighborhood of x, we can use Taylor expansion:

$$y = f'(x)\delta + \tfrac{1}{2}f''(x)\delta^2 + \cdots .$$

The terms in this series decrease rapidly when $|\delta|$ is small, so that cancellation is no longer a problem.

Addition is an example of a potentially ill-conditioned function (of two variables). It naturally leads us to study the condition of more general functions.

§3. The Condition of a Problem

A problem typically has an input and an output. The input consists of a set of data, say, the coefficients of some equation, and the output of another set of numbers uniquely determined by the input, say, all the roots of the equation in some prescribed order. If we collect the input in a vector $x \in \mathbb{R}^m$ (assuming the data consist of real numbers), and the output in the vector $y \in \mathbb{R}^n$ (also assumed real), we have the black box situation shown in Figure 1.3.1, where the box P accepts some input x and then solves the problem for this input to produce the output y.

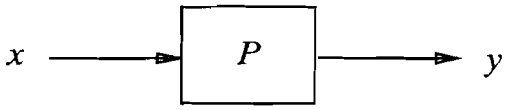


FIGURE 1.3.1. Black box representation of a problem

We may thus think of a problem as a map f, given by

$$f: \quad \mathbb{R}^m \to \mathbb{R}^n, \quad y = f(x). \tag{3.1}$$

(One or both of the spaces $\mathbb{R}^m$, $\mathbb{R}^n$ could be complex spaces without changing in any essential way the discussion that follows.) What we are interested in is the sensitivity of the map f at some given point x to a small perturbation of x, that is, how much bigger (or smaller) the perturbation in y is compared to the perturbation in x. In particular, we wish to measure the degree of sensitivity by a single number — the *condition number* of the map f at the point x. We emphasize that, as we perturb x, the function f is always assumed to be evaluated exactly, with infinite precision. The condition of f, therefore, is an inherent property of the map f and does not depend on any algorithmic considerations concerning its implementation.

This is not to say that knowledge of the condition of a problem is irrelevant to any algorithmic solution of the problem. On the contrary! The reason is that quite often the *computed* solution y^* of (3.1) (computed in floating-point machine arithmetic, using a specific algorithm) can be demonstrated to be the *exact* solution to a "nearby" problem; that is,

$$y^* = f(x^*), \tag{3.2}$$

where x^* is a vector close to the given data x,

$$x^* = x + \delta, \tag{3.3}$$

and moreover, the distance $\|\delta\|$ of x^* to x can be estimated in terms of the machine precision. Therefore, if we know how strongly (or weakly) the map f reacts to a small perturbation, such as δ in (3.3), we can say something about the error $y^* - y$ in the solution caused by this perturbation. This, indeed, is an important technique of error analysis — known as *backward error analysis* — which was pioneered in the 1950s by J. W. Givens, C. Lanczos, and, above all, J. H. Wilkinson.

Maps f between more general spaces (in particular, function spaces) have also been considered from the point of view of conditioning, but eventually, these spaces have to be reduced to finite-dimensional spaces for practical implementation.

§3.1. **Condition numbers**. We start with the simplest case of a single function of one variable.

The case $m = n = 1$: $y = f(x)$. Assuming first $x \neq 0$, $y \neq 0$, and denoting by Δx a small perturbation of x, we have for the corresponding perturbation Δy by Taylor's formula

$$\Delta y = f(x + \Delta x) - f(x) \approx f'(x)\Delta x, \tag{3.4}$$

assuming that f is differentiable at x. Since our interest is in *relative* errors, we write this in the form

$$\frac{\Delta y}{y} \approx \frac{xf'(x)}{f(x)} \cdot \frac{\Delta x}{x}. \tag{3.5}$$

The approximate equality becomes a true equality in the limit as $\Delta x \to 0$. This suggests that the condition of f at x be defined by the quantity

$$(\operatorname{cond} f)(x) := \left|\frac{xf'(x)}{f(x)}\right|. \tag{3.6}$$

This number tells us how much larger the relative perturbation in y is compared to the relative perturbation in x.

If $x = 0$ and $y \neq 0$, it is more meaningful to consider the absolute error measure for x and for y still the relative error. This leads to the condition number $|f'(x)/f(x)|$. Similarly for $y = 0$, $x \neq 0$. If $x = y = 0$, the condition number by (3.4) would then simply be $|f'(x)|$.

The case of arbitrary m, n: Here we write

$$x = [x_1, x_2, \ldots, x_m]^T \in \mathbb{R}^m, \quad y = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n$$

and exhibit the map f in component form

$$y_\nu = f_\nu(x_1, x_2, \ldots, x_m), \quad \nu = 1, 2, \ldots, n. \tag{3.7}$$

We assume again that each function f_ν has partial derivatives with respect to all m variables at the point x. Then the most detailed analysis departs

from considering each component y_ν as a function of one single variable, x_μ. In other words, we subject only one variable, x_μ, to a small change and observe the resulting change in just one component, y_ν. Then we can apply (3.6) and obtain

$$\gamma_{\nu\mu}(x) := (\mathrm{cond}_{\nu\mu} f)(x) := \left| \frac{x_\mu \frac{\partial f_\nu}{\partial x_\mu}}{f_\nu(x)} \right| . \tag{3.8}$$

This gives us a whole matrix $\Gamma(x) = [\gamma_{\nu\mu}(x)] \in \mathbb{R}_+^{n\times m}$ of condition numbers. To obtain a single condition number, we can take any convenient measure of the "magnitude" of the matrix $\Gamma(x)$ such as one of the matrix norms defined in (3.11),

$$(\mathrm{cond}\, f)(x) = \|\Gamma(x)\|, \quad \Gamma(x) = [\gamma_{\nu\mu}(x)]. \tag{3.9}$$

The condition so defined, of course, depends on the choice of norm, but the order of magnitude (and that is all that counts) should be more or less the same for any reasonable norm.

If a component of x, or of y, vanishes, one modifies (3.8) as discussed earlier.

A less refined analysis can be modeled after the one-dimensional case by defining the relative perturbation of $x \in \mathbb{R}^m$ to mean

$$\frac{\|\Delta x\|_{\mathbb{R}^m}}{\|x\|_{\mathbb{R}^m}} , \quad \Delta x = [\Delta x_1, \Delta x_2, \ldots, \Delta x_m]^T, \tag{3.10}$$

where Δx is a perturbation vector whose components Δx_μ are small compared to x_μ, and where $\|\cdot\|_{\mathbb{R}^m}$ is some vector norm in $\mathbb{R}^m$. For the perturbation Δy caused by Δx, one defines similarly the relative perturbation $\|\Delta y\|_{\mathbb{R}^n}/\|y\|_{\mathbb{R}^n}$, with a suitable vector norm $\|\cdot\|_{\mathbb{R}^n}$ in $\mathbb{R}^n$. One then tries to relate the relative perturbation in y to the one in x.

To carry this out, one needs to define a matrix norm for matrices $A \in \mathbb{R}^{n\times m}$. We choose the so-called "operator norm,"

$$\|A\|_{\mathbb{R}^{n\times m}} := \max_{\substack{x\in\mathbb{R}^m \\ x\neq 0}} \frac{\|Ax\|_{\mathbb{R}^n}}{\|x\|_{\mathbb{R}^m}} . \tag{3.11}$$

In the following we take for the vector norms the "uniform" (or infinity) norm,

$$\|x\|_{\mathbb{R}^m} = \max_{1\le\mu\le m} |x_\mu| =: \|x\|_\infty, \quad \|y\|_{\mathbb{R}^n} = \max_{1\le\nu\le n} |y_\nu| =: \|y\|_\infty . \tag{3.12}$$

It is then easy to show that (cf. Ex. 30)

$$\|A\|_{\mathbb{R}^{n\times m}} =: \|A\|_\infty = \max_{1\le\nu\le n} \sum_{\mu=1}^{m} |a_{\nu\mu}|, \quad A = [a_{\nu\mu}] \in \mathbb{R}^{n\times m}. \tag{3.13}$$

Now in analogy to (3.4), we have

$$\Delta y_\nu = f_\nu(x+\Delta x) - f_\nu(x) \approx \sum_{\mu=1}^{m} \frac{\partial f_\nu}{\partial x_\mu} \Delta x_\mu.$$

Therefore, at least approximately,

$$\begin{aligned} |\Delta y_\nu| &\le \sum_{\mu=1}^{m} \left|\frac{\partial f_\nu}{\partial x_\mu}\right| |\Delta x_\mu| \le \max_\mu |\Delta x_\mu| \cdot \sum_{\mu=1}^{m} \left|\frac{\partial f_\nu}{\partial x_\mu}\right| \\ &\le \max_\mu |\Delta x_\mu| \cdot \max_\nu \sum_{\mu=1}^{m} \left|\frac{\partial f_\nu}{\partial x_\mu}\right|. \end{aligned}$$

Since this holds for each $\nu = 1, 2, \ldots, n$, it also holds for $\max\limits_\nu |\Delta y_\nu|$, giving, in view of (3.12) and (3.13),

$$\|\Delta y\|_\infty \le \|\Delta x\|_\infty \left\|\frac{\partial f}{\partial x}\right\|_\infty. \tag{3.14}$$

Here

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \cdot & \cdot & \cdots & \cdot \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \in \mathbb{R}^{n\times m} \tag{3.15}$$

is the *Jacobian matrix* of f. (This is the analogue of the first derivative for *systems* of functions of *several* variables.) From (3.14) one now immediately obtains for the relative perturbations

$$\frac{\|\Delta y\|_\infty}{\|y\|_\infty} \le \frac{\|x\|_\infty \, \|\partial f/\partial x\|_\infty}{\|f(x)\|_\infty} \cdot \frac{\|\Delta x\|_\infty}{\|x\|_\infty}.$$

Although this is an inequality, it is sharp in the sense that equality can be achieved for a suitable perturbation Δx. We are justified, therefore, in defining a global condition number by

$$(\text{cond } f)(x) := \frac{\|x\|_\infty \|\partial f/\partial x\|_\infty}{\|f(x)\|_\infty} . \tag{3.16}$$

Clearly, in the case $m = n = 1$, the definition (3.16) reduces precisely to the definition (3.6) (as well as (3.9)) given earlier. In higher dimensions (m and/or n larger than 1), however, the condition number in (3.16) is much cruder than the one in (3.9). This is because norms tend to destroy detail: if x, for example, has components of vastly different magnitudes, then $\|x\|_\infty$ is simply equal to the largest of these components, and all the others are ignored. For this reason, some caution is required when using (3.16).

To give an example, consider

$$f(x) = \begin{bmatrix} \dfrac{1}{x_1} + \dfrac{1}{x_2} \\ \dfrac{1}{x_1} - \dfrac{1}{x_2} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

The components of the condition matrix $\Gamma(x)$ in (3.8) are then

$$\gamma_{11} = \left|\frac{x_2}{x_1 + x_2}\right|, \ \gamma_{12} = \left|\frac{x_1}{x_1 + x_2}\right|, \ \gamma_{21} = \left|\frac{x_2}{x_2 - x_1}\right|, \ \gamma_{22} = \left|\frac{x_1}{x_2 - x_1}\right|,$$

indicating ill-conditioning if either $x_1 \approx x_2$ or $x_1 \approx -x_2$ and $|x_1|$ (hence also $|x_2|$) is not small. The global condition number (3.16), on the other hand, since

$$\frac{\partial f}{\partial x}(x) = -\frac{1}{x_1^2 x_2^2} \begin{bmatrix} x_2^2 & x_1^2 \\ x_2^2 & -x_1^2 \end{bmatrix},$$

becomes, when L_1 vector and matrix norms are used (cf. Ex. 31),

$$(\text{cond } f)(x) = \frac{\|x\|_1 \cdot \dfrac{2}{x_1^2 x_2^2} \max(x_1^2,\ x_2^2)}{\dfrac{1}{|x_1 x_2|}(|x_1 + x_2| + |x_1 - x_2|)} = 2\frac{|x_1| + |x_2|}{|x_1 x_2|} \frac{\max(x_1^2,\ x_2^2)}{|x_1 + x_2| + |x_1 - x_2|}.$$

Here $x_1 \approx x_2$ or $x_1 \approx -x_2$ yield $(\text{cond } f)(x) \approx 2$, which is obviously misleading.

§3.2. **Examples.** We illustrate the idea of numerical condition in a number of examples, some of which are of considerable interest in applications.

(1) Compute $I_n = \int_0^1 \frac{t^n}{t+5} dt$ for some fixed integer $n \geq 1$. As it stands, the example here deals with a map from the integers to reals, and therefore does not fit our concept of "problem" in (3.1). However, we propose to compute I_n recursively by relating I_k to I_{k-1} and noting that

$$I_0 = \int_0^1 \frac{dt}{t+5} = \ln(t+5)\Big|_0^1 = \ln\frac{6}{5} . \tag{3.17}$$

To find the recursion, observe that

$$\frac{t}{t+5} = 1 - \frac{5}{t+5} .$$

Thus, multiplying both sides by t^{k-1} and integrating from 0 to 1 yields

$$I_k = -5I_{k-1} + \frac{1}{k}, \quad k = 1, 2, \ldots, n. \tag{3.18}$$

We see that I_k is a solution of the (linear, inhomogeneous, first-order) difference equation

$$y_k = -5y_{k-1} + \frac{1}{k}, \quad k = 1, 2, 3, \ldots . \tag{3.19}$$

We now have what appears to be a practical scheme to compute I_n: start with $y_0 = I_0$ given by (3.17), and then apply in succession (3.19) for $k = 1, 2, \ldots, n$; then $y_n = I_n$. The recursion (3.19), for any starting value y_0, defines a function,

$$y_n = f_n(y_0). \tag{3.20}$$

We have the black box in Figure 1.3.2 and thus a problem $f_n : \mathbb{R} \to \mathbb{R}$.

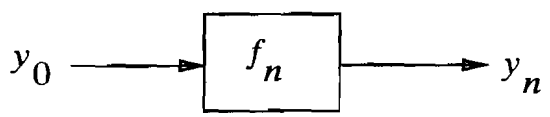


FIGURE 1.3.2. Black box for the recursion (3.19)

(Here n is a parameter.) We are interested in the condition of f_n at the point $y_0 = I_0$ given by (3.17). Indeed, I_0 in (3.17) is not machine-representable, and must be rounded to I_0^* before the recursion (3.19) can be employed. Even if no further errors are introduced during the recursion, the final result will not be exactly I_n, but some approximation $I_n^* = f_n(I_0^*)$, and we have, at least approximately (actually exactly; see the remark after (3.27)),

$$\left|\frac{I_n^* - I_n}{I_n}\right| = (\text{cond } f_n)(I_0)\left|\frac{I_0^* - I_0}{I_0}\right| . \tag{3.21}$$

To compute the condition number, note that f_n is a linear function of y_0. Indeed, if $n = 1$, then

$$y_1 = f_1(y_0) = -5y_0 + 1.$$

If $n = 2$, then

$$y_2 = f_2(y_0) = -5y_1 + \frac{1}{2} = (-5)^2 y_0 - 5 + \frac{1}{2} ,$$

and so on. In general,

$$y_n = f_n(y_0) = (-5)^n y_0 + p_n,$$

where p_n is some number (independent of y_0). There follows

$$(\text{cond } f_n)(y_0) = \left|\frac{y_0 f_n'(y_0)}{y_n}\right| = \left|\frac{y_0(-5)^n}{y_n}\right| . \tag{3.22}$$

Now, if $y_0 = I_0$, then $y_n = I_n$, and from the definition of I_n as an integral it is clear that I_n decreases monotonically in n (and indeed converges monotonically to zero as $n \to \infty$). Therefore,

$$(\text{cond } f_n)(I_0) = \frac{I_0 \cdot 5^n}{I_n} > \frac{I_0 \cdot 5^n}{I_0} = 5^n. \tag{3.23}$$

We see that $f_n(y_0)$ is severely ill-conditioned at $y_0 = I_0$, the more so the larger n.

We could have anticipated this result by just looking at the recursion (3.19): we keep multiplying by (–5), which tends to make things bigger, whereas they should get smaller! Thus, there will be continuous cancellation occurring throughout the recursion.

How can we avoid this ill-conditioning? The clue comes from the remark just made: instead of multiplying by a large number, we would prefer dividing by a large number, especially if the results get bigger at the same time. This is accomplished by reversing the recurrence (3.19), that is, by choosing an $\nu > n$ and computing

$$y_{k-1} = \frac{1}{5}\left(\frac{1}{k} - y_k\right), \quad k = \nu, \nu-1, \ldots, n+1. \tag{3.24}$$

The problem then, of course, is how to compute the starting value y_ν. Before we deal with this, let us observe that we now have a new black box, as shown in Figure 1.3.3.

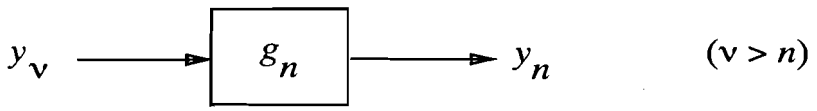


FIGURE 1.3.3. Black box for the recursion (3.24)

As before, the function involved, g_n, is a linear function of y_ν, and an argument similar to the one leading to (3.22) then gives

$$(\operatorname{cond} g_n)(y_\nu) = \left|\frac{y_\nu\left(-\frac{1}{5}\right)^{\nu-n}}{y_n}\right|, \quad \nu > n. \tag{3.25}$$

For $y_\nu = I_\nu$, we get, again by the monotonicity of I_n,

$$(\operatorname{cond} g_n)(I_\nu) < \left(\frac{1}{5}\right)^{\nu-n}, \quad \nu > n. \tag{3.26}$$

In analogy to (3.21), we now have

$$\left|\frac{I_n^* - I_n}{I_n}\right| = (\operatorname{cond} g_n)(I_\nu)\left|\frac{I_\nu^* - I_\nu}{I_\nu}\right| < \left(\frac{1}{5}\right)^{\nu-n}\left|\frac{I_\nu^* - I_\nu}{I_\nu}\right|, \tag{3.27}$$

where I_ν^* is some approximation of I_ν. Actually, I_ν^* does not even have to be close to I_ν for (3.27) to hold, since the function g_n is linear. Thus, we may take $I_\nu^* = 0$, committing a 100% error in the starting value, yet obtaining I_n^* with a relative error

$$\left|\frac{I_n^* - I_n}{I_n}\right| < \left(\frac{1}{5}\right)^{\nu-n}, \quad \nu > n. \tag{3.28}$$

The bound on the right can be made arbitrarily small, say, $\leq \varepsilon$, if we choose ν large enough; for example,

$$\nu \geq n + \frac{\ln \frac{1}{\varepsilon}}{\ln 5}. \tag{3.29}$$

The final procedure, therefore, is: given the desired relative accuracy ε, choose ν to be the smallest integer satisfying (3.29), and then compute

$$\begin{aligned} I_\nu^* &= 0, \\ I_{k-1}^* &= \frac{1}{5}\left(\frac{1}{k} - I_k^*\right), \quad k = \nu, \nu-1, \ldots, n+1. \end{aligned} \tag{3.30}$$

This will produce a sufficiently accurate $I_n^* \approx I_n$, even in the presence of rounding errors committed in (3.30): they, too, will be consistently attenuated.

Similar ideas can be applied to the more important problem of computing solutions to second-order linear recurrence relations such as those satisfied by Bessel functions and many other special functions of mathematical physics. The procedure of backward recurrence is then closely tied up with the theory of continued fractions.

(2) *Algebraic equations*: these are equations involving a polynomial of given degree n,

$$p(x) = 0, \; p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, \; a_0 \neq 0. \tag{3.31}$$

Let ξ be some fixed root of the equation, which we assume to be simple,

$$p(\xi) = 0, \quad p'(\xi) \neq 0. \tag{3.32}$$

The problem then is to find ξ, given p. The data vector $a = [a_0, a_1, \ldots, a_{n-1}]^T \in \mathbb{R}^n$ consists of the coefficients of the polynomial p, and the result is ξ, a real or complex number. Thus, we have

$$\xi: \quad \mathbb{R}^n \to \mathbb{C}, \quad \xi = \xi(a_0, a_1, \ldots, a_{n-1}). \tag{3.33}$$

What is the condition of ξ? We adopt the detailed approach of (3.8) and first define

$$\gamma_\nu = (\mathrm{cond}_\nu\, \xi)(a) = \left| \frac{a_\nu \frac{\partial \xi}{\partial a_\nu}}{\xi} \right|, \quad \nu = 0, 1, \ldots, n-1. \tag{3.34}$$

Then we take a convenient norm, say, the L_1 norm $\|\gamma\|_1 := \sum_{\nu=0}^{n-1} |\gamma_\nu|$ of the vector $\gamma = [\gamma_0, \ldots, \gamma_{n-1}]^T$, to define

$$(\operatorname{cond} \xi)(a) = \sum_{\nu=0}^{n-1} (\operatorname{cond}_\nu \xi)(a). \tag{3.35}$$

To determine the partial derivative of ξ with respect to a_ν, observe that we have the identity

$$[\xi(a_0, a_1, \ldots, a_n)]^n + a_{n-1}[\xi(\cdots)]^{n-1} + \cdots + a_\nu[\xi(\cdots)]^\nu + \cdots + a_0 \equiv 0.$$

Differentiating this with respect to a_ν, we get

$$n[\xi(a_0, a_1, \ldots, a_n)]^{n-1} \frac{\partial \xi}{\partial a_\nu} + a_{n-1}(n-1)[\xi(\cdots)]^{n-2} \frac{\partial \xi}{\partial a_\nu} + \cdots$$

$$+ a_\nu \nu [\xi(\cdots)]^{\nu-1} \frac{\partial \xi}{\partial a_\nu} + \cdots + a_1 \frac{\partial \xi}{\partial a_\nu} + [\xi(\cdots)]^\nu \equiv 0,$$

where the last term comes from differentiating the first factor in the product $a_\nu \xi^\nu$. The last identity can be written as

$$p'(\xi) \frac{\partial \xi}{\partial a_\nu} + \xi^\nu = 0.$$

Since $p'(\xi) \neq 0$, we can solve for $\partial \xi / \partial a_\nu$ and insert the result in (3.34) and (3.35) to obtain

$$(\operatorname{cond} \xi)(a) = \frac{1}{|\xi p'(\xi)|} \sum_{\nu=0}^{n-1} |a_\nu| \, |\xi|^\nu. \tag{3.36}$$

We illustrate (3.36) by considering the polynomial p of degree n that has the zeros $1, 2, \ldots, n$,

$$p(x) = \prod_{\nu=1}^{n} (x - \nu) = x^n + a_{n-1}x^{n-1} + \cdots + a_0. \tag{3.37}$$

This is a famous example due to J. H. Wilkinson, who discovered the ill-conditioning of some of the zeros almost by accident. If we let $\xi_\mu = \mu$, $\mu = 1, 2, \ldots, n$, it can be shown that

$$\min_\mu \operatorname{cond} \xi_\mu = \operatorname{cond} \xi_1 \sim n^2 \quad \text{as} \quad n \to \infty,$$

$$\max_{\mu} \operatorname{cond} \xi_\mu \sim \frac{1}{\left(2-\sqrt{2}\right)\pi n} \left(\frac{\sqrt{2}+1}{\sqrt{2}-1}\right)^n \quad \text{as} \quad n \to \infty.$$

The worst-conditioned root is ξ_{μ_0} with μ_0 the integer closest to $n/\sqrt{2}$, when n is large. Its condition number grows like $(5.828\ldots)^n$, thus exponentially fast in n. For example, when $n = 20$, then $\operatorname{cond} \xi_{\mu_0} = .540 \times 10^{14}$.

The example teaches us that the roots of an algebraic equation written in the form (3.31) can be extremely sensitive to small changes in the coefficients a_ν. It would, therefore, be ill-advised to express every polynomial in terms of powers, as in (3.37) and (3.31). This is particularly true for characteristic polynomials of matrices. It is much better here to work with the matrices themselves and try to reduce them (by similarity transformations) to a form that allows the eigenvalues — the roots of the characteristic equation — to be read off relatively easily.

(3) *Systems of linear algebraic equations*: given a nonsingular square matrix $A \in \mathbb{R}^{n\times n}$, and a vector $b \in \mathbb{R}^n$, the problem now discussed is solving the system

$$Ax = b. \tag{3.38}$$

Here the data are the elements of A and b, and the result the vector x. The map in question is thus $\mathbb{R}^{n^2+n} \to \mathbb{R}^n$. To simplify matters, let us assume that A is a fixed matrix not subject to change, and only the vector b is undergoing perturbations. We then have a map $f: \mathbb{R}^n \to \mathbb{R}^n$ given by

$$x = f(b) := A^{-1}b.$$

It is in fact a linear map. Therefore, $\partial f/\partial b = A^{-1}$, and we get, using (3.16),

$$(\operatorname{cond} f)(b) = \frac{\|b\| \, \|A^{-1}\|}{\|A^{-1}b\|}, \tag{3.39}$$

where we may take any vector norm in $\mathbb{R}^n$ and associated matrix norm (cf. (3.11)). We can write (3.39) alternatively in the form

$$(\operatorname{cond} f)(b) = \frac{\|Ax\| \, \|A^{-1}\|}{\|x\|} \quad (\text{where } Ax = b),$$

and since there is a one-to-one correspondence between x and b, we find for the worst condition number

$$\max_{\substack{b\in\mathbb{R}^n \\ b\neq 0}} (\operatorname{cond} f)(b) = \max_{\substack{x\in\mathbb{R}^n \\ x\neq 0}} \frac{\|Ax\|}{\|x\|} \cdot \|A^{-1}\| = \|A\| \cdot \|A^{-1}\|,$$

by definition of the norm of A. The number on the far right no longer depends on the particular system (i.e., on b) and is called the *condition number* of the matrix A. We denote it by

$$\text{cond } A := \|A\| \cdot \|A^{-1}\| \ . \tag{3.40}$$

It should be clearly understood, though, that it measures the condition of a linear system with coefficient matrix A, and not the condition of other quantities that may depend on A, such as eigenvalues.

Although we have considered only perturbations in the right-hand vector b, it turns out that the condition number in (3.40) is also relevant when perturbations in the matrix A are allowed, provided they are sufficiently small (so small, for example, that $\|\Delta A\| \cdot \|A^{-1}\| < 1$).

We illustrate (3.40) by several examples.

(i) Hilbert[2] matrix:

$$H_n = \begin{bmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n-1} \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{3.41}$$

This is clearly a symmetric matrix, and it is also positive definite. Some numerical values for the condition number of H_n, computed with the Euclidean norm,[3] are shown in Table 1.3.1. Their rapid growth is devastating.

[2]David Hilbert (1862–1943) was the most prominent member of the Göttingen school of mathematics. Hilbert's fundamental contributions to almost all parts of mathematics — algebra, number theory, geometry, integral equations, calculus of variations, and foundations — and in particular the 23 now famous problems he proposed in 1900 at the International Congress of Mathematicians in Paris, gave a new impetus, and new directions, to 20th-century mathematics. Hilbert is also known for his work in mathematical physics, where among other things he formulated a variational principle for Einstein's equations in the theory of relativity.

[3]We have $\text{cond}_2 H_n = \lambda_{\max}(H_n) \cdot \lambda_{\max}(H_n^{-1})$, where $\lambda_{\max}(A)$ denotes the largest eigenvalue of the (symmetric, positive definite) matrix A. We computed all eigenvalues of H_n and H_n^{-1}, using the appropriate Eispack routine. The inverse of H_n was computed from its well-known explicit form (not by inversion!). The total computing time (on a CDC 6500 computer in the 1980s) was 45 sec, at a cost of $1.04

n	$\mathrm{cond}_2 H_n$
10	1.60×10^{13}
20	2.45×10^{28}
40	7.65×10^{58}

TABLE 1.3.1. The condition of Hilbert matrices

A system of order $n = 10$, for example, cannot be solved with any reliability in single precision on a 14-decimal computer. Double precision will be "exhausted" by the time we reach $n = 20$. The Hilbert matrix thus is a prototype of an ill-conditioned matrix. From a result of G. Szegő it can be seen that

$$\mathrm{cond}_2 H_n \sim \frac{\left(\sqrt{2}+1\right)^{4n+4}}{2^{15/4}\sqrt{\pi n}} \quad \text{as} \quad n \to \infty.$$

(ii) Vandermonde[4] matrices: these are matrices of the form

$$V_n = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ t_1 & t_2 & \cdots & t_n \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ t_1^{n-1} & t_2^{n-1} & \cdots & t_n^{n-1} \end{bmatrix} \in \mathbb{R}^{n\times n}, \tag{3.42}$$

where $t_1, t_2, \ldots, t_n$ are parameters, here assumed real. The condition number of these matrices, in the ∞-norm, has been studied at length. Here are some sample results: if the parameters are equally spaced in [–1,1], that is,

$$t_\nu = 1 - \frac{2(\nu-1)}{n-1}, \quad \nu = 1, 2, \ldots, n,$$

then

$$\mathrm{cond}_\infty V_n \sim \frac{1}{\pi} e^{-\pi/4} e^{n\left(\frac{\pi}{4}+\frac{1}{2}\ln 2\right)}, \quad n \to \infty.$$

Numerical values are shown in Table 1.3.2.

[4] Alexandre Théophile Vandermonde (1735–1796), the author of only four mathematical papers, was elected to the French Academy of Sciences before he even wrote his first paper, apparently as a result of influential acquaintances. Nevertheless, his papers, especially the first, made important contributions to the then emerging theory of equations. By virtue of his fourth paper, he is regarded as the founder of the theory of determinants. What today is referred to as the "Vandermonde determinant," however, does not seem to appear anywhere in his writings. As a member of the Academy, he was appointed to the committee that in 1799 was to define the unit of length — the meter.

n	$\mathrm{cond}_\infty V_n$
10	1.36×10^4
20	1.05×10^9
40	6.93×10^{18}
80	3.15×10^{38}

TABLE 1.3.2. The condition of Vandermonde matrices

They are not growing quite as fast as those for the Hilbert matrix, but still exponentially fast. Worse than exponential growth is observed if one takes harmonic numbers as parameters,

$$t_\nu = \frac{1}{\nu}, \quad \nu = 1, 2, \ldots, n.$$

Then indeed

$$\mathrm{cond}_\infty V_n > n^{n+1}.$$

Fortunately, there are not many matrices occurring naturally in applications that are *that* ill-conditioned, but moderately to severely ill-conditioned matrices are no rarity in real-life applications.

§4. The Condition of an Algorithm

We again assume that we are dealing with a problem f given by

$$f: \quad \mathbb{R}^m \to \mathbb{R}^n, \quad y = f(x). \tag{4.1}$$

Along with the problem f, we are also given an algorithm A that "solves" the problem. That is, given a machine vector $x \in \mathbb{R}^m(t, s)$, the algorithm A produces a vector y_A (in machine arithmetic) that is supposed to approximate $y = f(x)$. Thus, we have another map f_A describing how the problem f is solved by the algorithm A,

$$f_A: \quad \mathbb{R}^m(t, s) \to \mathbb{R}^n(t, s), \quad y_A = f_A(x). \tag{4.2}$$

In order to be able to analyze f_A in these general terms, we must make a basic assumption, namely, that

$$\begin{aligned} &\text{for every } \ x \in \mathbb{R}^m(t, s), \quad \text{there holds} \\ &f_A(x) = f(x_A) \ \text{ for some } \ x_A \in \mathbb{R}^m. \end{aligned} \tag{4.3}$$

That is, the computed solution corresponding to some input x is the exact solution for some different input x_A (not necessarily a machine vector and not necessarily uniquely determined) that we hope is close to x. The closer we can find an x_A to x, the more confidence we should place in the algorithm A. We therefore define the condition of A in terms of the x_A closest to x (if there is more than one), by comparing its relative error with the machine precision eps:

$$(\text{cond } A)(x) = \inf_{x_A} \frac{\|x_A - x\|}{\|x\|}/\text{eps}. \tag{4.4}$$

Here the infimum is over all x_A satisfying $y_A = f(x_A)$. In practice one can take any such x_A and then obtain an upper bound for the condition number:

$$(\text{cond } A)(x) \leq \frac{\|x_A - x\|}{\|x\|}/\text{eps}. \tag{4.5}$$

The vector norm in (4.4), respectively, (4.5), can be chosen as seems convenient.

Here are some very elementary examples.

(1) Suppose a library routine for the logarithm function furnishes $y = \ln x$, for any positive machine number x, by producing a y_A satisfying $y_A = [\ln x](1 + \varepsilon)$, $|\varepsilon| \leq 5\,\text{eps}$. What can we say about the condition of the underlying algorithm A? We clearly have

$$y_A = \ln x_A \quad \text{where,} \quad x_A = x^{1+\varepsilon} \quad \text{(uniquely)}.$$

Consequently,

$$\left|\frac{x_A - x}{x}\right| = \left|\frac{x^{1+\varepsilon} - x}{x}\right| = |x^{\varepsilon} - 1| \approx |\varepsilon \ln x| \leq 5\,|\ln x| \cdot \text{eps},$$

and, therefore, $(\text{cond } A)(x) \leq 5\,|\ln x|$. The algorithm A is well-conditioned, except in the immediate right-hand vicinity of $x = 0$ and for x very large. (In the latter case, however, x is likely to overflow before A becomes seriously ill-conditioned.)

(2) Consider the problem

$$f: \quad \mathbb{R}^n \to \mathbb{R}, \quad y = x_1 x_2 \cdots x_n.$$

We solve the problem by the obvious algorithm

$$A: \quad \begin{aligned} p_1 &= x_1, \\ p_k &= \mathrm{fl}(x_k p_{k-1}), \quad k = 2, 3, \ldots, n, \\ y_A &= p_n. \end{aligned}$$

Note that x_1 is machine-representable, since for the algorithm A we assume $x \in \mathbb{R}^n(t, s)$.

Now using the basic law of machine arithmetic (cf. (2.1)), we get

$$\begin{aligned} p_1 &= x_1, \\ p_k &= x_k p_{k-1}(1 + \varepsilon_k), \quad k = 2, 3, \ldots, n, \quad |\varepsilon_k| \leq \mathrm{eps}, \end{aligned}$$

from which

$$p_n = x_1 x_2 \cdots x_n (1 + \varepsilon_2)(1 + \varepsilon_3) \cdots (1 + \varepsilon_n).$$

Therefore, we can take, for example (there is no uniqueness),

$$x_A = [x_1, x_2(1 + \varepsilon_2), \ldots, x_n(1 + \varepsilon_n)]^T.$$

This gives, using the ∞-norm,

$$\frac{\|x_A - x\|_\infty}{\|x\|_\infty \mathrm{eps}} = \frac{\|[0, x_2\varepsilon_2, \ldots, x_n\varepsilon_n]^T\|_\infty}{\|x\|_\infty \mathrm{eps}} \leq \frac{\|x\|_\infty \mathrm{eps}}{\|x\|_\infty \mathrm{eps}} = 1,$$

and so, by (4.5), $(\mathrm{cond}\, A)(x) \leq 1$ for any $x \in \mathbb{R}^n(t, s)$. Our algorithm, to nobody's surprise, is perfectly well-conditioned.

§5. Computer Solution of a Problem; Overall Error

The problem to be solved is again

$$f: \quad \mathbb{R}^m \to \mathbb{R}^n, \quad y = f(x). \tag{5.1}$$

This is the mathematical (idealized) problem, where the data are exact real numbers, and the solution is the mathematically exact solution.

When solving such a problem on a computer, in floating-point arithmetic with precision eps, and using some algorithm A, one first of all rounds the data, and then applies to these rounded data not f, but f_A:

$$\begin{aligned} x^* &= \text{rounded data}, \quad \frac{\|x^* - x\|}{\|x\|} = \varepsilon, \\ y_A^* &= f_A(x^*). \end{aligned} \tag{5.2}$$

Here ε represents the rounding error in the data. (The error ε could also be due to sources other than rounding, e.g., measurement.) The total error that we wish to estimate is then

$$\frac{\|y_A^* - y\|}{\|y\|} . \tag{5.3}$$

By the basic assumption (4.3) made on the algorithm A, and choosing x_A^* optimally, we have

$$f_A(x^*) = f(x_A^*), \quad \frac{\|x_A^* - x^*\|}{\|x^*\|} = (\operatorname{cond} A)(x^*) \cdot \text{eps}. \tag{5.4}$$

Let $y^* = f(x^*)$. Then, using the triangle inequality, we have

$$\frac{\|y_A^* - y\|}{\|y\|} \le \frac{\|y_A^* - y^*\|}{\|y\|} + \frac{\|y^* - y\|}{\|y\|} \approx \frac{\|y_A^* - y^*\|}{\|y^*\|} + \frac{\|y^* - y\|}{\|y\|} ,$$

where we have used the (harmless) approximation $\|y\| \approx \|y^*\|$. By virtue of (5.4), we now have for the first term on the right,

$$\begin{aligned} \frac{\|y_A^* - y^*\|}{\|y^*\|} = \frac{\|f_A(x^*) - f(x^*)\|}{\|f(x^*)\|} &= \frac{\|f(x_A^*) - f(x^*)\|}{\|f(x^*)\|} \\ &\le (\operatorname{cond} f)(x^*) \cdot \frac{\|x_A^* - x^*\|}{\|x^*\|} \\ &= (\operatorname{cond} f)(x^*) \cdot (\operatorname{cond} A)(x^*) \cdot \text{eps}. \end{aligned}$$

For the second term we have

$$\frac{\|y^* - y\|}{\|y\|} = \frac{\|f(x^*) - f(x)\|}{\|f(x)\|} \le (\operatorname{cond} f)(x) \cdot \frac{\|x^* - x\|}{\|x\|} = (\operatorname{cond} f)(x) \cdot \varepsilon.$$

Assuming finally that $(\text{cond } f)(x^*) \approx (\text{cond } f)(x)$, we get

$$\frac{\|y_A^* - y\|}{\|y\|} \leq (\text{cond } f)(x)\{\varepsilon + (\text{cond } A)(x^*) \cdot \text{eps}\}. \tag{5.5}$$

This shows how the data error and machine precision contribute toward the total error: both are amplified by the condition of the problem, but the latter is further amplified by the condition of the algorithm.

NOTES TO CHAPTER 1

In addition to rounding errors in the data and those committed during the execution of arithmetic operations, there may be other sources of errors not considered in this introductory chapter. One such source of error, which is not entirely dismissible, is a faulty design of the computer chip that executes arithmetic operations. This was brought home in a recent incident when it was discovered in 1994 (by Thomas Nicely in the course of number-theoretic computations involving reciprocals of twin primes) that the Pentium floating-point divide chip manufactured by Intel can produce erroneous results for certain (extremely rare) bit patterns in the divisor. The incident — rightly so — has stirred up considerable concern, and prompted not only remedial actions, but also careful analysis of the phenomenon; some relevant articles are those by Coe, Mathisen, Moler, and Pratt [1995] and Edelman [preprint].

Neither should the occurrence of overflow and proper handling thereof be taken lightly, especially not in real-time applications. Again, a case in point is the failure of the French rocket Ariane 5, which on June 4, 1996, less than a minute into its flight, self-destructed. The failure was eventually traced to an overflow in a floating-point to integer conversion and lack of protection against this occurrence in the rocket's on-board software (cf. Anonymous [1996]).

§1.1. The abstract notion of the real number system is discussed in most texts on real analysis, for example, Hewitt and Stromberg [1975, Ch.1, §5] or Rudin [1976, Ch.1]. The development of the concept of real (and complex) numbers has had a long and lively history, extending from pre-Hellenic times to the recent past. Many of the leading thinkers over time contributed to this development. A reader interested in a detailed historical account (and who knows German) is referred to the monograph by Gericke [1970].

§1.2 (a) The notion of the floating-point number system and associated arithmetic, including interval arithmetic, can also be phrased in abstract algebraic terms; for this, see, for example, Kulisch and Miranker [1981]. A more elementary, but detailed, discussion of floating-point numbers and arithmetic is given in Sterbenz [1974]. There the reader will learn, for example, that computing the average of two floating-point numbers, or solving a quadratic equation, can be fairly intricate

tasks if they are to be made foolproof. The quadratic equation problem is also considered at some length in Young and Gregory [1988, §3.4], where further references are given to earlier work of W. Kahan and G. E. Forsythe.

The basic standard for binary floating-point arithmetic, used on all contemporary computers, is the ANSI/IEEE Standard 754 established in IEEE [1985]. It provides for $t = 23$ bits in the mantissa and $s = 7$ bits in the exponent, in single-precision arithmetic, and has $t = 52$, $s = 11$ in double precision. There is also an "extended precision" for which $t = 63$, $s = 14$, allowing for a number range of approx. 10^{-4964} to 10^{+4964}.

(c) Rational arithmetic is available in all major symbolic computation packages such as Mathematica and MACSYMA.

Interval arithmetic has evolved to become an important tool in computations that strive at obtaining guaranteed and sharp inclusion regions for the results of mathematical problems. The basic texts on interval analysis are Moore [1966], [1979] and Alefeld and Herzberger [1983]. Specific applications such as computing inclusions of the range of functions, of global extrema of functions of one and several variables, and of solutions to systems of linear and nonlinear equations are studied, respectively, in Ratschek and Rokne [1984], [1988], Hansen [1992], and Neumaier [1990]. Concrete algorithms and codes (in Pascal and C^{++}) for "verified computing" are contained in Hammer, Hocks, Kulisch, and Ratz [1993], [1995]. Interval arithmetic has been most widely used in processes involving finite-dimensional spaces; for applications to infinite-dimensional problems, notably differential equations, see, however, Eijgenraam [1981] and Kaucher and Miranker [1984].

§2. The fact that thoughtless use of mathematical formulae and numerical methods, or inherent sensitivities in a problem, can lead to disastrous results, has been known since the early days of computers; see, for example, the old but still relevant papers by Stegun and Abramowitz [1956] and Forsythe [1970]. Nearby singularities can also cause the accuracy to deteriorate unless corrective measures are taken; Forsythe [1958] has an interesting discussion of this.

§2.1. For the implications of rounding in the problem of apportionment, mentioned in Footnote 1, a good reference is Garfunkel and Steen [1988, Ch.12, pp.230–249].

§3.1. An early but basic reference for ideas of conditioning and error analysis in algebraic processes is Wilkinson [1963]. An impressive continuation of this work, containing copious references to the literature, is Higham [1996]. It analyzes the behavior in floating-point arithmetic of virtually all the algebraic processes in current use. Problems of conditioning specifically involving polynomials are discussed in Gautschi [1984]. The condition of general (differentiable) maps has been studied as early as 1966 in Rice [1966].

§3.2. (1) For a treatment of stability aspects of more general difference equations, and systems thereof, including nonlinear ones, the reader is referred to the

monograph by Wimp [1984]. This also contains many applications to special functions. Another relevant text is Lakshmikantham and Trigiante [1988].

(2) The condition of algebraic equations, although considered already in Wilkinson's book [1963], has been further analyzed by Gautschi [1973]. The circumstances that led to Wilkinson's example (3.37), which he himself describes as "the most traumatic experience in [his] career as a numerical analyst," are related in the essay by Wilkinson [1984, §2]. This reference also deals with errors committed in the evaluation and deflation of polynomials. For the latter, also see Cohen [1994]. The asymptotic estimates for the best- and worst-conditioned roots in Wilkinson's example are from Gautschi [1973]. For the computation of eigenvalues of matrices, the classic treatment is Wilkinson [1988]; more recent accounts are Parlett [1980] for symmetric matrices, and Golub and Van Loan [1996, Ch. 7–9] for general matrices.

(3) A more complete analysis of the condition of linear systems, that also allows for perturbations of the matrix, can be found, for example, in the very readable books by Forsythe and Moler [1967, Ch.8] and Stewart [1973, Ch.4, §3]. The asymptotic result of Szegő cited in connection with the Euclidean condition number of the Hilbert matrix is taken from Szegő [1936]. For the explicit inverse of the Hilbert matrix, referred to in Footnote 3, see Todd [1954]. The condition of Vandermonde and Vandermonde-like matrices has been studied in a series of papers by the author; for a summary, see Gautschi [1990].

§§4 and 5. The treatment of the condition of algorithms and of the overall error in computer solutions of problems, as given in these sections, seems to be more or less original. Similar ideas, however, can be found in the book by Dahlquist and Björck [1974, Ch.2, §4].

EXERCISES AND MACHINE ASSIGNMENTS TO CHAPTER 1

EXERCISES

1. Represent all elements of $\mathbb{R}_+(3,2) = \{x \in \mathbb{R}(3,2) : x > 0,\ x \text{ normalized}\}$ as dots on the real axis. For clarity, draw two axes, one from 0 to 8, the other from 0 to $\frac{1}{2}$.

2. (a) What is the distance $d(x)$ of a positive normalized floating-point number $x \in \mathbb{R}(t,s)$ to its next larger floating-point number:

$$d(x) = \min_{\substack{y \in \mathbb{R}(t,s) \\ y > x}} (y - x)\,?$$

   (b) Determine the relative distance $r(x) = d(x)/x$, with x as in (a), and give upper and lower bounds for it.

3. The identity $\mathrm{fl}(1+x) = 1$, $x \geq 0$, is true for $x = 0$ and for x sufficiently small. What is the largest machine number x for which the identity still holds?

4. Consider a miniature binary computer whose floating-point words consist of 4 binary digits for the mantissa and 3 binary digits for the exponent (plus sign bits). Let

$$x = (.1011)_2 \times 2^0, \qquad y = (.1100)_2 \times 2^0.$$

   Mark in the following table whether the machine operation indicated (with the result z assumed normalized) is exact, rounded (i.e., subject to a nonzero rounding error), overflows, or underflows.

operation	exact	rounded	overflow	underflow
$z = \mathrm{fl}(x - y)$				
$z = \mathrm{fl}((y - x)^{10})$				
$z = \mathrm{fl}(x + y)$				
$z = \mathrm{fl}(y + (x/4))$				
$z = \mathrm{fl}(x + (y/4))$				

5. The following algorithm (attributed to CLEVE MOLER) estimates eps:

```
a=4./3.
b=a-1.
c=b+b+b
eps=abs(c-1.).
```

Run the program with the corresponding double-precision statements appended to it and print the single- and double-precision eps.

6. Prove (1.12).

7. A set S of elements, or pairs of elements, is said to possess a metric if there is defined a distance function $d(x, y)$ for any two elements $x, y \in S$ that has the following properties:

   (i) $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$ (positive definiteness);

   (ii) $d(x, y) = d(y, x)$ (symmetry);

   (iii) $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality).

   Discuss which of the following error measures is a distance function on what set S (of real numbers, or pairs of real numbers):

   (a) absolute error: $\text{ae}(x, y) = |x - y|$;

   (b) relative error: $\text{re}(x, y) = \left|\frac{x-y}{x}\right|$;

   (c) relative precision (F.W.J. OLVER, 1978): $\text{rp}(x, y) = |\ \ln|x| - \ln|y|\ |$;

   If $y = x(1 + \varepsilon)$, show that $\text{rp}(x, y) = O(\varepsilon)$ as $\varepsilon \to 0$.

8. Assume that x_1^*, x_2^* are approximations to x_1, x_2 with relative errors E_1 and E_2, respectively, and that $|E_i| \leq E$, $i = 1, 2$. Assume further that $x_1 \neq x_2$.

   (a) How small must E be in order to ensure that $x_1^* \neq x_2^*$?

   (b) Taking $\dfrac{1}{x_1^* - x_2^*}$ to approximate $\dfrac{1}{x_1 - x_2}$, obtain a bound on the relative error committed, assuming (i) exact arithmetic; (ii) machine arithmetic with machine precision eps. (Neglect higher-order terms in E_1, E_2, eps.)

9. Consider the quadratic equation $x^2 + px + q = 0$ with roots x_1, x_2. As seen in Example (2) of §2.2, the absolutely larger root must be computed first, whereupon the other can be accurately obtained from $x_1 x_2 = q$. Suppose one incorporates this idea in a program such as

```
x1=abs(p/2)+sqrt(p*p/4-q)
if(p.gt.0.) x1=-x1
x2=q/x1.
```

Find three serious faults with this program as a "general-purpose quadratic equation solver." Take into consideration that the program will be executed in floating-point machine arithmetic. Be specific and support your arguments by examples, if necessary.

10. Let $f(x) = \sqrt{1+x^2} - 1$.

    (a) Explain the difficulty of computing $f(x)$ for a small value of $|x|$ and show how it can be circumvented.

    (b) Compute $(\text{cond}\, f)(x)$ and discuss the conditioning of $f(x)$ for small $|x|$.

    (c) How can the answers to (a) and (b) be reconciled?

11. The nth power of some positive (machine) number x can be computed

    (i) either by repeated multiplication by x, or

    (ii) as $x^n = e^{n \ln x}$.

    In each case, derive bounds for the relative error due to machine arithmetic, neglecting higher powers of the machine precision against the first power. Based on these bounds, state a criterion (involving x and n) for (i) to be better than (ii).

12. Let $f(x) = (1 - \cos x)/x$, $x \neq 0$.

    (a) Show that direct evaluation of f is inaccurate if $|x|$ is small; assume $\text{fl}(f(x)) = \text{fl}((1 - \text{fl}(\cos x))/x)$, where $\text{fl}(\cos x) = (1 + \varepsilon_c)\cos x$, and estimate the relative error of $\text{fl}(f(x))$ as $x \to 0$.

    (b) A mathematically equivalent form of f is $f(x) = \sin^2 x/(x(1 + \cos x))$. Carry out a similar analysis as in (a), based on $\text{fl}(f(x)) = \text{fl}([\text{fl}(\sin x)]^2/(x(1 + \text{fl}(\cos x))))$, assuming $\text{fl}(\cos x) = (1 + \varepsilon_c)\cos x$, $\text{fl}(\sin x) = (1 + \varepsilon_s)\sin x$ and retaining only first-order terms in ε_s and ε_c. Discuss the result.

    (c) Determine the condition of $f(x)$. Indicate for what values of x (if any) $f(x)$ is ill-conditioned. ($|x|$ is no longer small, necessarily.)

13. If $z = x + iy$, then $\sqrt{z} = \left(\frac{r+x}{2}\right)^{1/2} + i\left(\frac{r-x}{2}\right)^{1/2}$, where $r = (x^2 + y^2)^{1/2}$. Alternatively, $\sqrt{z} = u + iv$, $u = \left(\frac{r+x}{2}\right)^{1/2}$, $v = y/2u$. Discuss the computational merits of these two (mathematically equivalent) expressions when $x > 0$. Illustrate with $z = 4.5 + .025i$, using 8 significant decimal places. How would you deal with $x < 0$?

14. Consider the numerical evaluation of

$$f(t) = \sum_{n=0}^{\infty} \frac{1}{1 + n^4(t-n)^2(t-n-1)^2},$$

say, for $t = 20$, and 7-digit accuracy. Discuss the danger involved.

15. Let X_+ be the largest positive machine-representable number, and X_- the absolute value of the smallest negative one (so that $-X_- \leq x \leq X_+$ for any machine number x). Determine, approximately, all intervals on $\mathbb{R}$ on which the tangent function overflows.

16. Consider a decimal computer with 3 (decimal) digits in the floating-point mantissa.

    (a) Estimate the relative error committed in symmetric rounding.

    (b) Let $x_1 = .982$, $x_2 = .984$ be two machine numbers. Calculate in machine arithmetic the mean $m = \frac{1}{2}(x_1 + x_2)$. Is the computed number between x_1 and x_2?

    (c) Derive sufficient conditions for $x_1 < \mathrm{fl}(m) < x_2$ to hold, where x_1, x_2 are two machine numbers with $0 < x_1 < x_2$.

17. For this problem, assume a binary computer with 12 bits in the floating-point mantissa.

    (a) What is the machine precision eps?

    (b) Let $x = 6/7$ and x^* be the correctly rounded machine approximation to x (symmetric rounding). Exhibit x and x^* as binary numbers.

    (c) Determine (exactly!) the relative error ε of x^* as an approximation to x, and calculate the ratio $|\varepsilon|/\mathrm{eps}$.

18. The associative law of algebra states that

$$(a + b)c = ac + bc.$$

Discuss to what extent this is violated in machine arithmetic. Assume a computer with machine precision eps and assume that a, b, c are machine-representable numbers.

    (a) Let y_1 be the floating-point number obtained by evaluating $(a + b)c$ (as written) in floating-point arithmetic, and let $y_1 = (a + b)c(1 + e_1)$. Estimate $|e_1|$ in terms of eps (neglecting second-order terms in eps).

    (b) Let y_2 be the floating-point number obtained by evaluating $ac + bc$ (as written) in floating-point arithmetic, and let $y_2 = (a + b)c(1 + e_2)$. Estimate $|e_2|$ (neglecting second-order terms in eps) in terms of eps (and a, b, and c).

(c) Identify conditions (if any) under which one of the two ys is significantly less accurate than the other.

19. Let $x_1, x_2, \ldots, x_n$ be machine numbers. Their product can be computed by the algorithm

$$p_1 = x_1,$$
$$p_k = \text{fl}(x_k p_{k-1}), \qquad k = 2, 3, \ldots, n.$$

(a) Find an upper bound for the relative error $(p_n - x_1 x_2 \cdots x_n)/(x_1 x_2 \cdots x_n)$ in terms of the machine precision eps and n.

(b) For any integer $r \geq 1$ small enough to satisfy $r \cdot \text{eps} < \frac{1}{10}$, show that

$$(1 + \text{eps})^r - 1 < (1.06) r \cdot \text{eps}.$$

Hence simplify the answer given in (a). {*Hint*: Use the binomial theorem.}

20. Analyze the error propagation in exponentiation, x^α $(x > 0)$:

(a) assuming x exact and α subject to a small relative error ε_α;

(b) assuming α exact and x subject to a small relative error ε_x.

Discuss the possibility of any serious loss of accuracy.

21. Indicate how you would accurately compute

$$(x + y)^{1/4} - y^{1/4}, \qquad x > 0, \qquad y > 0.$$

22. (a) Let $a = .23371258 \times 10^{-4}$, $b = .33678429 \times 10^2$, $c = -.33677811 \times 10^2$. Assuming an 8-decimal-digit computer, determine the sum $s = a + b + c$ either as (i) $\text{fl}(s) = \text{fl}(\text{fl}(a + b) + c)$ or as (ii) $\text{fl}(s) = \text{fl}(a + \text{fl}(b + c))$. Explain the discrepancy between the two answers.

(b) For arbitrary machine numbers a, b, c, on a computer with machine precision eps, find a criterion on a, b, c for the result of (ii) in (a) to be more accurate than the result of (i). {*Hint*: Compare bounds on the relative errors, neglecting higher-order terms in eps and assuming $a + b + c \neq 0$.}

23. Write the expression $a^2 - 2ab\cos\gamma + b^2$ $(a > 0, b > 0)$ as the sum of two positive terms in order to avoid cancellation errors. Illustrate the advantage gained in the case $a = 16.5$, $b = 15.7$, $\gamma = 5°$, using 3-decimal-digit arithmetic. Is the method foolproof?

24. Determine the condition number for the following functions.

(a) $f(x) = \ln x, \quad x > 0;$ (b) $f(x) = \cos x, \quad |x| < \frac{1}{2}\pi;$

(c) $f(x) = \sin^{-1} x, \quad |x| < 1;$ (d) $f(x) = \sin^{-1} \dfrac{x}{\sqrt{1+x^2}}$.

Indicate the possibility of ill-conditioning.

25. Compute the condition number of the following functions, and discuss any possible ill-conditioning.

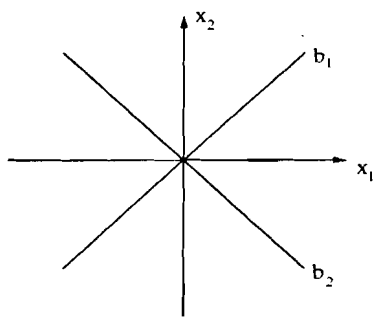
(a) $f(x) = x^{1/n} \quad (x > 0,\ n > 0$ an integer)
(b) $f(x) = x - \sqrt{x^2 - 1} \quad (x > 1)$
(c) $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$
(d) $f(x_1, x_2) = x_1 + x_2$

26. (a) Consider the composite function $h(t) = g(f(t))$. Express the condition of h in terms of the condition of g and f. Be careful to state at which points the various condition numbers are to be evaluated.
(b) Illustrate (a) with $h(t) = \frac{1+\sin t}{1-\sin t}$, $t = \frac{1}{4}\pi$.

27. Show that $(\operatorname{cond} f \cdot g)(x) \leq (\operatorname{cond} f)(x) + (\operatorname{cond} g)(x)$.

28. Let $f : \mathbb{R}^2 \to \mathbb{R}$ be given by $y = x_1 + x_2$. Define $(\operatorname{cond} f)(x) = (\operatorname{cond}_1 f)(x) + (\operatorname{cond}_2 f)(x)$, where $\operatorname{cond}_i f$ is the condition number of f considered a function of x_i only $(i = 1, 2)$.

(a) Derive a formula for $\kappa(x_1, x_2) = (\operatorname{cond} f)(x)$.
(b) Show that $\kappa(x_1, x_2)$ as a function of x_1, x_2 is symmetric with respect to both bisectors b_1 and b_2 (see figure).



(c) Determine the lines (or domains) in $\mathbb{R}^2$ on which $\kappa(x_1, x_2) = c$, $c \geq 1$ a constant. (Simplify the analysis by using symmetry; cf. part (b).)

29. Let $\|\cdot\|$ be a vector norm in $\mathbb{R}^n$ and denote by the same symbol the associated matrix norm. Show for arbitrary matrices $A, B \in \mathbb{R}^{n\times n}$ that

    (a) $\|AB\| \le \|A\|\,\|B\|$;

    (b) $\operatorname{cond}(AB) \le \operatorname{cond} A \cdot \operatorname{cond} B$.

30. Prove (3.13). {*Hint*: Let $m_\infty = \max_\nu \sum_\mu |a_{\nu\mu}|$. Show that $\|A\|_\infty \le m_\infty$ as well as $\|A\|_\infty \ge m_\infty$, the latter by taking a special vector x in (3.11).}

31. Let the L_1 norm of a vector $y = [y_\lambda]$ be defined by $\|y\|_1 = \sum_\lambda |y_\lambda|$. For a matrix $A \in \mathbb{R}^{n\times m}$, show that

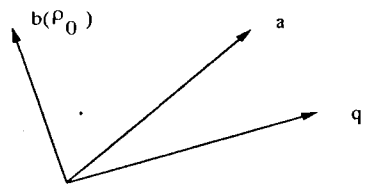
$$\|A\|_1 := \max_{\substack{x\in\mathbb{R}^m \\ x\neq 0}} \frac{\|Ax\|_1}{\|x\|_1} = \max_\mu \sum_\nu |a_{\nu\mu}|;$$

    that is, $\|A\|_1$ is the "maximum column sum." {*Hint*: Let $m_1 = \max_\mu \sum_\nu |a_{\nu\mu}|$. Show that $\|A\|_1 \le m_1$ as well as $\|A\|_1 \ge m_1$, the latter by taking for x in (3.11) an appropriate coordinate vector.}

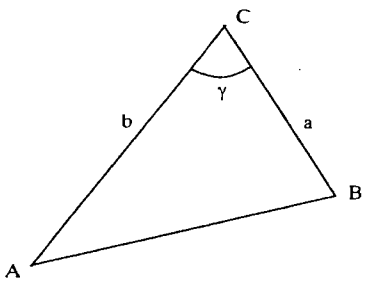
32. Let a, q be linearly independent vectors in $\mathbb{R}^n$ of (Euclidean) length 1. Define $b(\rho) \in \mathbb{R}^n$ as follows:

$$b(\rho) = a - \rho q, \quad \rho \in \mathbb{R}.$$

    Compute the condition of the angle $\alpha(\rho)$ between $b(\rho)$ and q at the value $\rho = \rho_0 = q^T a$. (Then $b(\rho_0) \perp q$; see figure.) Discuss the answer.



33. The area Δ of a triangle ABC is given by $\Delta = \frac{1}{2}ab\sin\gamma$ (see figure). Discuss the condition of Δ.



34. Define, for $x \neq 0$,

$$f_n = f_n(x) = (-1)^n \frac{d^n}{dx^n}\left(\frac{e^{-x}}{x}\right), \quad n = 0, 1, 2, \ldots .$$

(a) Show that $\{f_n\}$ satisfies the recursion

$$y_k = \frac{k}{x}\, y_{k-1} + \frac{e^{-x}}{x}, \quad k = 1, 2, 3, \ldots ; \quad y_0 = \frac{e^{-x}}{x} .$$

{*Hint*: Differentiate k times the identity $e^{-x} = x \cdot (e^{-x}/x)$.}

(b) Why do you expect the recursion in (a), without doing any analysis, to be numerically stable if $x > 0$? How about $x < 0$?

(c) Support and discuss your answer to (b) by showing

$$(\operatorname{cond} y_n)(f_0) = \frac{1}{|e_n(x)|} ,$$

where $e_n(x) = 1 + x + x^2/2! + \cdots + x^n/n!$ is the nth partial sum of the exponential series. {*Hint*: Use Leibniz's formula to evaluate f_n.}

35. Consider the algebraic equation

$$x^n + ax - 1 = 0, \quad a > 0, \quad n \geq 2.$$

(a) Show that the equation has exactly one positive root $\xi(a)$.

(b) Obtain a formula for $(\operatorname{cond} \xi)(a)$.

(c) Obtain (good) upper and lower bounds for $(\operatorname{cond} \xi)(a)$.

36. Consider the algebraic equation

$$x^n + x^{n-1} - a = 0, \quad a > 0, \quad n \geq 2.$$

(a) Show that there is exactly one positive root $\xi(a)$.

(b) Show that $\xi(a)$ is well-conditioned as a function of a. Indeed, prove

$$(\operatorname{cond} \xi)(a) < \frac{1}{n-1} .$$

37. Consider the equation

$$xe^x = a$$

for real values of x and a.

(a) Show graphically that the equation has exactly one root $\xi(a) \geq 0$ if $a \geq 0$, exactly two roots $\xi_2(a) < \xi_1(a) < 0$ if $-1/e < a < 0$, and none if $a < -1/e$.

(b) Discuss the condition of $\xi(a)$, $\xi_1(a)$, $\xi_2(a)$ as a varies in the respective intervals.

38. Given the natural number n, let $\xi = \xi(a)$ be the unique positive root of the equation $x^n = ae^{-x}$ $(a > 0)$. Determine the condition of ξ as a function of the parameter a; simplify the answer as much as possible. In particular, show that $(\text{cond } \xi)(a) < 1/n$.

39. Let $f(x_1, x_2) = x_1 + x_2$ and consider the algorithm A given as follows,

$$f_A : \mathbb{R}^2(t,s) \to \mathbb{R}(t,s) \qquad y_A = \text{fl}(x_1 + x_2).$$

Estimate $\gamma(x_1, x_2) = (\text{cond } A)(x)$, using any of the norms

$$\|x\|_1 = |x_1| + |x_2|, \quad \|x\|_2 = \sqrt{x_1^2 + x_2^2}, \quad \|x\|_\infty = \max(|x_1|, |x_2|).$$

Discuss the answer in the light of the conditioning of f.

40. This problem deals with the function $f(x) = \sqrt{1-x} - 1$, $-\infty < x < 1$.

(a) Compute the condition number $(\text{cond } f)(x)$.

(b) Let A be the algorithm that evaluates $f(x)$ in floating-point arithmetic on a computer with machine precision eps, given an (error-free) floating-point number x. Let ε_1, ε_2, ε_3 be the relative errors due, respectively, to the subtraction in $1 - x$, to taking the square root, and to the final subtraction of 1. Assume $|\varepsilon_i| \leq \text{eps}$ $(i = 1, 2, 3)$. Letting $f_A(x)$ be the value of $f(x)$ so computed, write $f_A(x) = f(x_A)$ and $x_A = x(1 + \varepsilon_A)$. Express ε_A in terms of x, ε_1, ε_2, ε_3 (neglecting terms of higher order in the ε_i). Then determine an upper bound for $|\varepsilon_A|$ in terms of x and eps, and finally an estimate of $(\text{cond } A)(x)$.

(c) Sketch a graph of $(\text{cond } f)(x)$ (found in (a)) and a graph of the estimate of $(\text{cond } A)(x)$ (found in (b)) as functions of x on $(-\infty, 1)$. Discuss your results.

41. Consider the function $f(x) = 1 - e^{-x}$ on the interval $0 \leq x \leq 1$.

(a) Show that $(\text{cond } f)(x) \leq 1$ on [0,1].

(b) Let A be the algorithm that evaluates $f(x)$ for the machine number x in floating-point arithmetic (with machine precision eps). Assume that the exponential routine returns a correctly rounded answer. Estimate $(\text{cond } A)(x)$ for $0 \leq x \leq 1$, neglecting terms of $O(\text{eps}^2)$. {*Point of information*: $\ln(1 + \varepsilon) = \varepsilon + O(\varepsilon^2)$, $\varepsilon \to 0$.}

(c) Plot $(\text{cond } f)(x)$ and your estimate of $(\text{cond } A)(x)$ as functions of x on [0,1]. Comment on the results.

42. (a) Suppose A is an algorithm that computes the (smooth) function $f(x)$ for a given machine number x, producing $f_A(x) = f(x)(1+\varepsilon_f)$, where $|\varepsilon_f| \leq \varphi(x)$eps (eps = machine precision). Show that

$$(\text{cond}\, A)(x) \leq \frac{\varphi(x)}{(\text{cond}\, f)(x)}$$

if second-order terms in eps are neglected. {*Hint*: Set $f_A(x) = f(x_A)$, $x_A = x(1+\varepsilon_A)$, and expand in powers of ε_A, keeping only the first.}

(b) Apply the result of (a) to $f(x) = \frac{1-\cos x}{\sin x}$, $0 < x < \frac{1}{2}\pi$, when evaluated as shown. (You may assume that $\cos x$ and $\sin x$ are computed within a relative error of eps.) Discuss the answer.

(c) Do the same as (b), but for the (mathematically equivalent) function $f(x) = \frac{\sin x}{1+\cos x}$, $0 < x < \frac{1}{2}\pi$.

MACHINE ASSIGNMENTS

1. Let $x = 1 + \pi/10^6$ and compute x^n for $n = 1, 2, \ldots, 10^6$, both in single precision and double precision. Use the double-precision results to observe the rounding errors ρ_n in the single-precision results (relative errors). Print (in e-format with 5 decimal digits after the decimal point, except for the integer n): n, x^n, ρ_n, $\rho_n/(n \times \text{eps})$ for $n = k \times 10^5$, $k = 1, 2, \ldots, 10$, where eps is the machine precision. What should x^n be, approximately, when $n = 10^6$? Comment on the results. (Compute the number π in a machine-independent manner, using an elementary function routine.)

2. Compute the derivative dy/dx of the exponential function $y = e^x$ at $x = 0$ from difference quotients $(e^{x+h} - e^x)/h$ with decreasing h. Use

   (a) $h = 2^{-i}$, $i = 1, 2, \ldots, 20$;

   (b) $h = (2.2)^{-i}$, $i = 1, 2, \ldots, 20$,

   and print 8 decimal digits after the decimal point. Explain what you observe.

3. Consider the following procedure for determining the limit $\lim_{h \to 0} (e^h - 1)/h$ on a computer. Let

$$d_n = \text{fl}\left(\frac{e^{2^{-n}} - 1}{2^{-n}}\right) \quad \text{for } n = 0, 1, 2, \ldots$$

and accept as the machine limit the first value satisfying $d_n = d_{n-1}$ $(n \geq 1)$.

   (a) Run the procedure on a computer.

(b) In floating-point arithmetic for $\mathbb{R}(t, s)$, with rounding by chopping, for what value of n will the correct limit be reached, assuming no underflow (of 2^{-n}) occurs? {*Hint*: Use $e^h = 1 + h + \frac{1}{2}h^2 + \cdots$.} Compare with the experiment made in (a).

(c) On what kind of computer (i.e., under what conditions on s and t) will underflow occur before the limit is reached?

4. Euler's constant $\gamma = .57721566490153286\ldots$ is defined as the limit

$$\gamma = \lim_{n\to\infty} \gamma_n, \quad \text{where} \quad \gamma_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} - \ln n.$$

Assuming that $\gamma - \gamma_n \sim cn^{-d}$, $n \to \infty$, for some constants c and $d > 0$, try to determine c and d experimentally on the computer.

5. Letting $\Delta u_n = u_{n+1} - u_n$, one has the easy formula

$$\sum_{n=1}^{N} \Delta u_n = u_{N+1} - u_1.$$

With $u_n = \ln(1 + n)$, compute each side (as it stands) for $N = 1000(1000)$ 10,000, the right-hand side in double precision. Print the relative discrepancy of the two sides. Repeat with $\sum_{n=1}^{N} u_n$, computed in single and double precision. Explain the results.

6. (a) Write a program that computes

$$S_N = \sum_{n=1}^{N} \left[\frac{1}{n} - \frac{1}{n+1}\right],$$

where the summation is carried out as written.

(a1) What is the exact sum S_N?

(a2) Run your program for $N = 10, 100, 1000, \ldots, 1,000,000$ and print out the error $|\mathrm{fl}(S_N) - S_N|$. Comment on the answers obtained.

(b) Do the same as in part (a), but for the product

$$p_N = \prod_{n=1}^{N} \frac{n}{n+1}.$$

Comment on the results.

7. (a) Suppose x, y, z are floating-point numbers with $x > y > z > 0$. Estimate the relative errors in

$$\mathrm{fl}(\mathrm{fl}(x + y) + z) \quad \text{and} \quad \mathrm{fl}(\mathrm{fl}(z + y) + x).$$

Which result is more accurate? Explain. Adapt your discussion so it applies to the sum s_n in Part (b).

(b) It is known that

$$\pi = 4 \lim_{n\to\infty} s_n, \qquad s_n = \sum_{k=0}^{n} \frac{(-1)^k}{2k+1} .$$

Write a program computing s_n for $n = 10^7$, using forward as well as backward summation. Compute the respective errors (in double precision as `err=abs(sngl(4.d0*dble(sn)-dpi))`, where `dpi` is the double-precision value of π and `sn` the single-precision value of s_n). Interpret the results.

8. In the theory of Fourier series the numbers

$$\lambda_n = \frac{1}{2n+1} + \frac{2}{\pi} \sum_{k=1}^{n} \frac{1}{k} \tan \frac{k\pi}{2n+1}, \quad n = 1, 2, 3, \ldots ,$$

known as *Lebesgue constants*, are of some importance.

(a) Show that the terms in the sum increase monotonically with k. How do the terms for k near n behave when n is large?

(b) Compute λ_n for $n = 1, 10, 10^2, \ldots, 10^5$ in single and double precision and compare the results. Explain what you observe.

9. Sum the series

$$\text{(a)} \quad \sum_{n=0}^{\infty} (-1)^n / n!^2 \qquad \text{(b)} \quad \sum_{n=0}^{\infty} 1/n!^2$$

until there is no more change in the partial sums to within the machine precision. Generate the terms recursively. Print the number of terms required and the value of the sum. (Answers in terms of Bessel functions: (a) $J_0(2)$; (b) $I_0(2)$.)

10. (P.J. DAVIS, 1993) Consider the series $\displaystyle\sum_{k=1}^{\infty} \frac{1}{k^{3/2} + k^{1/2}}$. Try to compute the sum to three correct decimal digits.

11. We know from calculus that

$$\lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n = e.$$

What is the "machine limit"? Explain.

12. Let $f(x) = (n+1)x - 1$. The iteration

$$x_k = f(x_{k-1}), \quad k = 1, 2, \ldots, K; \quad x_0 = \frac{1}{n}$$

in exact arithmetic converges to the fixed point $1/n$ in one step (why?). What happens in machine arithmetic? Run a program with $n = 1(1)5$ and $K = 10(10)50$ and explain quantitatively what you observe.

13. Compute the integral $\int_0^1 e^x dx$ from Riemann sums with n equal subintervals, evaluating the integrand at the midpoint of each. Print the Riemann sums for $n = 10, 20, 30, \ldots, 200$ (showing 6 decimal digits after the decimal point), together with the exact answers. Comment on the results.

14. Let $y_n = \int_0^1 t^n e^{-t} dt$, $n = 0, 1, 2, \ldots$.

    (a) Use integration by parts to obtain a recurrence formula relating y_k to y_{k-1} for $k = 1, 2, 3, \ldots$, and determine the starting value y_0.

    (b) Write and run in single precision a program that generates $y_0, y_1, \ldots, y_{12}$, using the recurrence of (a), and prints the results in `e15.7` format. Explain in detail (quantitatively, using mathematical analysis) what is happening.

    (c) Use the recursion in (a) in reverse order, starting (arbitrarily) with $y_N = 0$. Print in four consecutive columns (in `e15.7` format) the values $y_0^{(N)}, y_1^{(N)}, \ldots, y_{12}^{(N)}$ thus obtained for $N = 14, 16, 18, 20$. Explain in detail (quantitatively) what you observe.