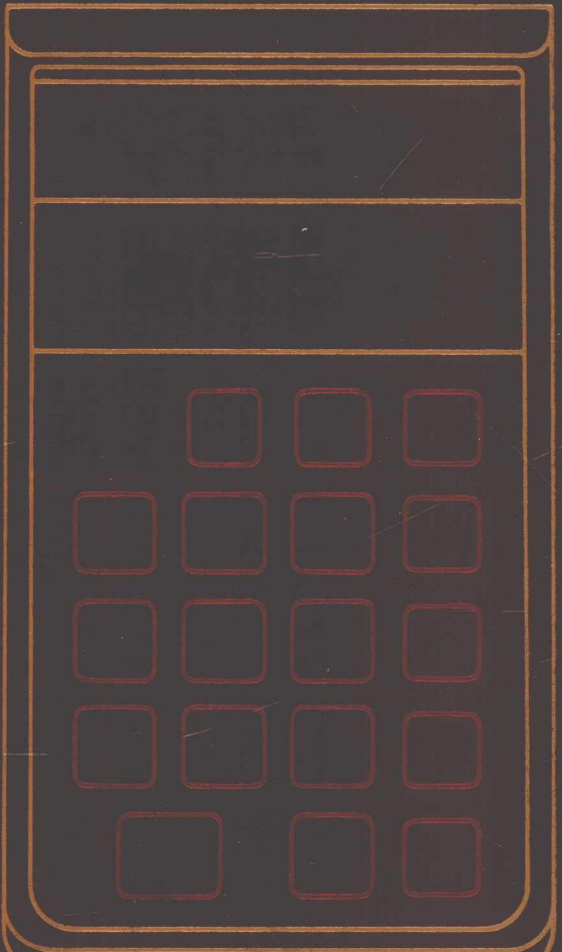
Peter Henrici



# ESSENTIALS OF NUMERICAL ANALYSIS

## With Pocket Calculator Demonstrations

# Essentials of Numerical Analysis with Pocket Calculator Demonstrations

Peter Henrici

1807 1982

John Wiley & Sons

New York Chichester Brisbane Toronto Singapore

*To Garrett Birkhoff*

**Other Books by Peter Henrici**

*Elements of Numerical Analysis* (1964)
*Discrete Variable Methods in Ordinary Differential Equations* (1962)
*Computational Analysis with the HP-25 Pocket Calculator* (1977)
*Applied and Computational Complex Analysis Vol. 1* (1974)
*Applied and Computational Complex Analysis Vol. 2* (1977)



**Library of Congress Cataloging in Publication Data:**

Henrici, Peter, 1923–
Essentials of numerical analysis, with pocket calculator demonstrations.

Bibliography: p.
Includes index.
1. Numerical analysis—Data processing. 2. Programmable calculators. I. Title.
QA297.H42 519.4 81-10468
ISBN 0-471-05904-8 AACR2

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

# Preface

This text grew out of an attempt to modernize my *Elements of Numerical Analysis* which first appeared in 1964. However, a new book has resulted. The treatment of the difference calculus has been eliminated. There now are substantial chapters on numerical linear algebra and on numerical Fourier analysis. The chapter on approximation has been enriched by sections on Hermite interpolation and on spline functions. A chapter on discrete computation, emphasizing such concepts as the numerical stability of an algorithm and the condition of a problem, now forms the basis of the whole work.

In addition, the number of numerical examples has been greatly increased. All examples, or *demonstrations* as they are called here, have been calculated on programmable pocket calculators. The numerous computational problems also are designed to be solved by means of such calculators. The majority of the demonstrations and problems can be performed on a calculator of the size of the HP-33E; a few (marked by the symbol $^{\oplus}$) require the capacity of the HP-34C. Specific calculators are rarely mentioned in the text; however, in order to make the demonstrations more easily reproducible for the student and to encourage further experimentation, companion booklets will be available for some current calculators which contain fully documented programs for virtually all demonstrations and for selected computational problems. In this pragmatic manner, a student working with this book will not only acquire firsthand experience with the joys and pitfalls of numerical computation, but will also learn some rudiments of scientific computer programming.

The methods of numerical analysis are, of course, largely independent of the computing equipment on hand. In this sense, the book may serve as a general introduction to numerical analysis and may be used in conjunction with any computing system. The emphasis on pocket calculators serves to free the teaching of numerical analysis from the organizational constraints of a computing center, and to enable the student to do experimental work in a relaxed atmosphere without having the next student who is waiting for a turn at the terminal breathing down his neck.

The prerequisites for this book are modest. They include calculus (including functions of several variables), plus a smattering of linear algebra and of differential equations. Courses covering most of the material of the Chapters 1 through 6 are regularly taught at the Swiss Federal Institute of Technology to first-year students of electrical engineering, geodesy, earth sciences, physics, and mathematics. I am grateful for having been allowed in the fall of 1978 to teach this course to a similar audience of junior-level students at Stanford University in California.

A glance at the table of contents will show that I am not trying to teach, or to allude to, everything that is known in numerical analysis. Something must be left to more advanced courses. Thus, the important topics of eigenvalue problems (differential and algebraic), iterative methods in linear algebra, and partial differential equations are totally absent. Even within the compass of topics that I am trying to do justice to, I prefer to treat a few methods and algorithms in reasonable depth rather than to superficially mention a large number of methods without adequate discussion. My judgement of what is most useful and pedagogically feasible has usually dictated my choice of topics. In a few instances, when such considerations did not point strongly in any particular direction, I have given preference to methods that have small memory requirements, and thus are suitable for pocket calculators.

While restricting the number of algorithms treated, I have taken pains, as I did in my *Elements*, to emphasize the connection of numerical analysis with certain topics of theoretical analysis that are in danger of being slighted in the structure-oriented treatment of mathematics which has become fashionable. Some examples are the emphasis on the asymptotic behavior of sequences throughout the book; the allusion to the calculus of variations in the treatment of splines in Chapter 5; the Euler-Maclaurin sum formula in Chapter 6; and the introduction to Fourier series from a least squares point of view in Chapter 7. In all these instances, an effort has been made to preserve an elementary and relaxed level of presentation.

In the Introduction I try to make clear that I regard numerical analysis as an essential tool of *applied* mathematics. To stress this fact I have often attempted, especially in the problem sections, to embed the numerical problem under study in an applied environment. I realize that even much more should be done in this direction. As anyone involved in the teaching of elementary courses knows, however, there is the difficulty that, at the level for which this book is intended, the acquaintance with mathematical models of applied situations is still very limited, especially considering the diversity of the backgrounds of my potential readers.

It is my pleasure to express my thanks to a number of individuals who helped me to shape this book. W. Robert Mann and David R. Kincaid have read the entire manuscript and have suggested many valuable improvements. Several of my assistants, notably Peter Geiger, have contributed to the exercises and have ironed out inconsistencies and outright errors. My wife, Marie-Louise Henrici, has examined the numerical work with great care. I have profited from conversations with Walter Gander who has generously shared his expertise in practical numerical analysis. Finally, anybody familiar with the work of Heinz Rutishauser will recognize my indebtedness to this giant of modern numerical analysis; my way of presenting the material (especially in Chapter 4) often is rooted in his teachings, as preserved in his *Vorlesungen über numerische Mathematik* (2 vols., Birkhäuser, Basel 1976).

I dedicate this volume to Garrett Birkhoff who in his lectures, in his teaching, and in his written work, has set a standard for doing justice to both dialectic and algorithmic mathematics, and for dealing with genuine applications while maintaining a high level of mathematical polish and intellectual purity. May his example prolong the traditional unity of mathematics.

Peter Henrici
*Zurich*

# Contents

# INTRODUCTION

C. G. J. Jacobi (b. 1804, d. 1851), the famous mathematician from Königsberg (the Jacobian matrix mentioned in §1.7 is named after him) once declared: "Mathematics solely and exclusively serves the honor of the human spirit." Nevertheless, mathematics time and again has been *applied*; that is, used to control man's relation to his physical environment.

The old Babylonians used special cases of the theorem of Pythagoras to generate right angles. Eratosthenes of Kyrene (b. −290, d. −214) knew that the planet earth is a sphere and used simple trigonometry to estimate its radius. Archimedes (b. −285, d. −215), in addition to numerous mathematical achievements, discovered the basic laws of statics and used them to construct engines of war which enabled Syracuse, his home, to withstand the Roman siege for two years. Johannes Kepler (b. 1571, d. 1630) explained the motions of the planets by means of conic sections. Isaac Newton (b. 1643, d. 1727) formulated the basic laws of dynamics and reduced all of celestial mechanics to a single mathematically formulated law, the law of gravitation, which even today serves to compute the trajectories of satellites and space ships. The Swiss mathematician Leonhard Euler (b. 1707, d. 1783), while still a student, won a Paris Academy prize with a paper on where to place masts on sailing vessels. His collected works abound with numerical applications of mathematics to all branches of physics that were then known. Carl Friedrich Gauss (b. 1777, d. 1855) was active as a surveyor and astronomer and created the method of least squares, a basic tool of applied mathematics. Bernhard Riemann (b. 1826, d. 1866), considered by many the greatest mathematician of the nineteenth century, laid the groundstone for what was to become a standard textbook of mathematical physics. Henri Poincaré (b. 1854, d. 1912), the founder of modern topology, wrote a *Méchanique Céleste* in three volumes.

Since World War II, the application of mathematics has been facilitated by the electronic computer with its enormous speed of calculation and its automatically executed programs. The atom bomb would have been impossible without large-scale computations, and the same holds for the conquest of space or for the peaceful uses of atomic energy. Today, mathematical methods are even used in fields, such as medicine and economics, that formerly seemed rather removed from quantification.

Most applications of mathematics exhibit a common pattern. First, an intellectual *model* is constructed of the phenomenon to be investigated. Usually one will try to keep this model as simple as possible; however, all relevant effects should be taken into account in an unambiguous manner. To keep the model simple it is quite customary to make simplifying assumptions. In studies of planetary motion, the planets are assumed to be points of mass. In models of economic science, functional dependencies are frequently assumed to be *linear*.

If complex phenomena are modelled mathematically, one first will have to ask whether the model used is *meaningful*. For instance, in the modelling of phenomena in classical physics the model frequently consists of a *differential equation* or of a system of such equations. Here, one will have to ask whether a solution of the equation exists, whether it depends continuously on the data (which usually are not precisely known), and whether the solution (as in nature) is determined uniquely by the data. Similar questions will be asked where the model consists of a system of algebraic equations. Questions of this kind, which relate to the meaningfulness of a mathematical model, are frequently resolved in a completely satisfactory manner by theoretical, "dialectic" mathematics, worked solely with pencil and paper.

The methods of dialectic mathematics may fail, however, if the model is to be used to draw quantitative conclusions. Consider space travel. Theoretical mathematics is very well able to assure the *existence* of a solution of the system of differential equations describing the trajectory of a space vehicle, but if the problem is to land on the moon, a knowledge of the mere existence of the correct trajectory is not sufficient. This trajectory must also be *constructed*; that is, the differential equations have to be solved.

This is the point where numerical mathematics springs into action. Its task is to develop methods for extracting quantitative answers from mathematical models. In this sense, numerical mathematics is a secondary science. Its points of reference are not the axioms of pure mathematics, but models and concrete problems which usually originate elsewhere. If one considers the ideal mathematical theory to be one that is purely abstract, axiomatic, and independent of extraneous considerations, then the preoccupation of numerical mathematics with *ad hoc* problems is clearly unsatisfactory. The inflexible concern with a given problem, which cannot be changed at will, somehow seems to stifle the free flight of imagination and at times gives numerical mathematics an aspect more akin to engineering than to pure mathematics. Whatever feelings of constraint may result from this should be compensated for by an awareness of the indispensable role which numerical mathematics has played time and again in solving man's problems in a modern world.

As far as certain standardized models of applied mathematics are concerned (ordinary differential equations, linear equations, approximation), numerical mathematics has created standardized methods of solution, called **algorithms**, which solve the problem in a routine fashion if the data do not exhibit pathological properties. Such standard algorithms are incorporated in the program libraries of most large computing centers and may be used without a detailed knowledge of numerical mathematics.

In addition to these standard models, in practice, models of a non-standard type frequently occur which require the development of special methods of computation, taking into account the pecularities of the model. In such instances, a knowledge of some principles of numerical mathematics is usually helpful. The art of *computer programming*, in addition to developing useful methodologies, serves to codify solution algorithms so that by means of a compiler they can be translated directly into a machine program.

In Chapters 2 through 7 of this book, solution algorithms for some typical problems of applied mathematics will be presented and, to the extent made possible by the modest prerequisites required, studied mathematically. In Chapter 1, we address the basic problem of *stability*, which must always be faced when the mathematical model involves a continuum, regardless of the special problem on hand or the algorithm used for solving it.

# CHAPTER 1

# Computation

In this chapter, we call attention to the fact that the number system of any computer is *discrete*, which implies that the results of even the simplest arithmetic operations and function evaluations are inaccurate. Unless appropriate measures are taken, these inaccuracies through such effects as cancellation and smearing can significantly diminish, and sometimes destroy, the accuracy of the result of an extended computation.

Using examples we then proceed to a discussion of the notions of *numerical* and *mathematical* stability. The former is a property of an algorithm; the latter (also called *condition*) is a property of the problem to be solved. These notions are basic to all of numerical analysis.

## §1.1 THE DISCRETE NUMBER SYSTEM OF THE COMPUTER

In principle, numerical mathematics is concerned with all kinds of computation, including computations involving Boolean expressions, algebraic formulas, or formulas from the predicate calculus. In practice, however, numerical mathematics is concerned mainly with computations involving *numbers* or systems of numbers, such as vectors or matrices.

Frequently, the models of applied mathematics are based on the idea of a *continuum* in space or time. The variables of the models then are (systems of) *real numbers*. Many models in electrical engineering and in mathematical physics work with *complex numbers*. The models of economics frequently involve numbers of units, that is, *integers*. In mechanical engineering, for instance in the design of gearboxes, it is sometimes appropriate to assume that the variables of the model are *rational numbers*.

In principle, the representation of *integers* in a computer does not present any difficulties. Any computer internally works with a fixed **base** $b$. Here $b$ is an integer $\geq 2$; frequently $b$ is chosen as a power of 2; less frequently $b = 10$. (However, the input and output of numbers is nearly always performed in the base $b = 10$.) As is well known, a given integer $n \neq 0$ possesses a unique representation,

$$n = \pm(n_0 + n_{-1}b + n_{-2}b^2 + \cdots + n_{-k}b^k),$$

where the $n_i$ are integers satisfying $0 \leq n_i < b$ ($i = 0, -1, \ldots, -k$), and where the representation is made unique by requiring that $n_{-k} \neq 0$. In ordinary notation, $n$ would be shown as

$$n_{-k}n_{-k+1} \cdots n_{-2}n_{-1}n_0 .$$

(The digits $n_i$ are numbered so that the indices increase from left to right.) Any integer $n$ thus is completely specified by its sign and by the sequence of its digits. The number 0 requires a special representation; among other things, it has no sign. (Regrettably, however, $-0 \neq +0$ on some calculators.) A difficulty arises from the fact that $k + 1$, the number of digits, is bounded. If, for instance, $b = 2$, and if 48 digits can be accommodated, then the largest integer that can be represented in the computer equals

$$2^{48} - 1 = 281,474,976,710,655.$$

Some applications, such as certain investigations in number theory, require the accommodation of larger integers. These would be represented in the "super" base $b = 2^{48}$, with digits which themselves are 48-digit integers in the base $b = 2$.

*Rational numbers* are represented in the computer as they are conceived in pure mathematics, namely, as pairs of integers.

The problems of numerical computation begin with the representation of *real numbers.* (Complex numbers, as in pure mathematics, are treated as ordered pairs of real numbers, subject to certain rules of computation.) It is clear that a computer cannot represent every real number. As is well known, the set of real numbers is not denumerable, whereas in the computer, a number must be characterized by denumerably many (in fact by finitely many) yes-or-no states. Two methods of representing real numbers are in use:

**(A) Fixed point representation.** This was the system used by the very early computers. The computer here works exclusively with numbers of the form,

$$x = \pm \sum_{i=m}^{n} x_i b^{-i},$$

where $m$ and $n$ are fixed integers (depending only on the computer) satisfying $m < n$ and (usually) $m \leq 0$, $n > 0$, and where the $x_i$ are integers satisfying $0 \leq x_i < b$. Every number that can be represented in the computer thus is determined by its sign and by the sequence of its digits,

$$(x_m, x_{m+1}, \ldots, x_n).$$

In ordinary notation this would be read as,

$$x_m x_{m+1} \cdots x_{-1} x_0 \cdot x_1 x_2 \cdots x_n.$$

In fixed point representation, the decimal (or binary, or "$b$-ary") point is always at the same place. For instance, the SEAC, the pioneering computer completed in 1951 at the National Bureau of Standards, worked with $b = 2$, $m = -1$, $n = 48$, which enabled it at least to accommodate approximations to such mathematically important numbers as $e$ and $\pi$.

If fixed point representation is used, the largest number that can be represented in the computer is $b^{1-m} - b^{-n}$; thus, for instance, on SEAC $4 - 2^{48}$. Before running a problem on a fixed point computer, it had to be prepared so that no final or intermediate result of the computation exceeded the maximum number that could be handled. This advance preparation, called **scaling**, could be extremely tedious. It had the advantage, however, that the details of the computation had to be looked into in advance with much greater care than is customary today.

**(B) Floating point representation.** This representation, which is much more flexible than fixed point representation, is almost universally used today. The computer works with numbers of the form

$$x = \pm y \cdot b^z. \tag{1}$$

Here,

$b$ is the **base** of the number system used,
$y$ is the **mantissa**,
$z$ is the **exponent**

of the number $x$. The *mantissa* is a fixed point number,

$$y = \sum_{i=m}^{n} y_i b^{-i},$$

made unique (if $x \neq 0$) by the condition that $y_m \neq 0$. Frequently, in large computers $m = 1$, so that (if $x \neq 0$), the $b$-ary point is to the extreme left, and

$$b^{-1} \leq y < 1.$$

The *number of digits* of the mantissa in any case is $n - m + 1$. The *exponent* $z$ is an integer, usually also represented in the base $b$.

**EXAMPLE 1.1-1:** On the CDC computer in use at the ETH Zurich, $y$ is a 48-digit binary number with $m = 1$. The range of the exponent $z$ is approximately indicated by

$$10^{-293} \leq 2^z \leq 10^{322}.$$

**EXAMPLE 1.1-2:** On the HP-33E pocket calculator, the mantissa is externally shown as a 7-digit number with $m = 0$. Thus, the mantissa shown satisfies

$$1 \leq y \leq 9.999999.$$

Internally, however, this calculator works with a 10-digit mantissa. The whole mantissa can be made visible by a special instruction. The range of the exponent is

$$-99 \leq z \leq 99.$$

We see that rather small and rather large numbers can be handled in floating point representation. This does not change the fundamental fact that the set of numbers which the computer has at its disposal is finite.

A number of interesting problems pose themselves to the engineer concerned with the design of digital computers, especially if the floating point representation is used. How should the basic arithmetic operations be performed? How should input and output be handled? How should the number 0 be treated? When should one round up, when down? We do not concern ourselves with these questions here. We simply recognize that the results of almost all operations have to be *rounded.*

Somewhat formalizing this fact, let $\mathbb{M} = \mathbb{M}_C$ be the set of numbers representable in the computer $C$. (This set generally depends on the computer; however, the index $C$ is omitted in the following.) If

$$x \in \mathbb{M}, \qquad y \in \mathbb{M},$$

the numbers

$$x + y, \qquad x - y, \qquad xy, \qquad \frac{x}{y}$$

are in general not in $\mathbb{M}$. The results of the basic arithmetical operations can be represented only approximately in the computer. Thus, for instance, the exact value of the product $xy$ must be approximated by some number $(xy)^*$ which is in $\mathbb{M}$. Assuming the engineers have worked well, the quantity

$$|(xy)^* - xy|$$

will be as small as possible; that is, the exact value $xy$ is represented by the (or a) machine number $(xy)^*$ closest to $xy$.

## PROBLEM

1 Find out the base, the number of digits in the mantissa, and the exponent range of your calculator. What is the smallest positive number representable in your calculator? What does your calculator do in the case of exponent underflow or overflow?

## §1.2 $\mathbb{M}$ IS NOT A FIELD

As a consequence of the discreteness of the number system of the computer, the simplest laws of algebra are no longer valid on the computer. We substantiate this claim, which at first sight may seem astonishing, by some simple examples. These examples were calculated on a programmable calculator of type HP-33E; however, similar examples could be found for any computer, large or small.

**DEMONSTRATION 1.2-1:** In any mathematical system known as a group (for instance, in the additive group of real numbers), there holds the associative law

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

where the parentheses indicate the operation that is to be executed first. However, on our calculator, the data

$$a = 1, \qquad b = 3 * 10^{-10}, \qquad c = 3 * 10^{-10}$$

yield the different results:

$$(a + b) + c = 1.000000000,$$
$$a + (b + c) = 1.000000001.$$

**DEMONSTRATION 1.2-2:** This is a more complicated example of the same kind. We consider the sum

$$s_n := 1 + \sum_{k=1}^{n} \frac{1}{k^2 + k}.$$

By writing this in the form,

$$s_n = 1 + \sum_{k=1}^{n} \left(\frac{1}{k} - \frac{1}{k+1}\right)$$

$$= 1 + (1 - \tfrac{1}{2}) + (\tfrac{1}{2} - \tfrac{1}{3}) + \cdots + \left(\frac{1}{n} - \frac{1}{n+1}\right)$$

it is clear that, mathematically,

$$s_n = 2 - \frac{1}{n+1},$$

so that, for instance,

$$s_{9999} = 1.9999.$$

Computing the sum numerically by the usual summation algorithm,

$$s_0 := 1, \qquad s_k := s_{k-1} + \frac{1}{k(k+1)}, \qquad k = 1, 2, \ldots,$$

we get

| $n$ | $s_n$ |
|---|---|
| 9 | 1.900000000 |
| 99 | 1.990000003 |
| 999 | 1.999000003 |
| 9999 | 1.999899972 |

However, if the summation is started at the tail end,

$$s_n = \frac{1}{(n+1)n} + \frac{1}{n(n+1)} + \cdots + \frac{1}{3 \cdot 2} + \frac{1}{2 \cdot 1} + 1,$$

then the results are as follows:

| $n$ | $s_n$ |
|---|---|
| 9 | 1.900000000 |
| 99 | 1.990000000 |
| 999 | 1.999000000 |
| 9999 | 1.999900000 |

The values obtained in the two modes of summation are different. Thus, the associative law also is violated here. Why does summing backwards yield the more accurate values? When the small terms are summed first, the exponents of the partial sums remain small until almost the very end. Accordingly, the rounding errors in the individual additions remain small until the exponent of the running sum reaches its final value, which in our example happens only at the very end. We conclude that, in forming a sum with a large number of terms, the terms with a small absolute value should be summed first.

**DEMONSTRATION 1.2-3:** In an additive group, an equation of the form $a + x = b$ always has a uniquely determined solution $x$. On our calculator the equation,

$$1 + x = 1$$

in addition to the mathematical solution $x = 0$ also has the solution $x = 10^{-10}$. More generally, every machine number $x$, satisfying $|x| < 5 * 10^{-10}$, is a solution of the equation. We conclude that on a computer equations need not have uniquely determined solutions.

**DEMONSTRATION 1.2-4:** It is well known that for arbitrary non-negative numbers $x$ and $y$, there holds the inequality of the arithmetic and the geometric mean,

$$\tfrac{1}{2}(x + y) \geq \sqrt{xy}$$

with equality holding only for $x = y$. On our calculator, however, the values

$$x = 5.000000001, \qquad y = 5.000000002$$

yield

$$\tfrac{1}{2}(x + y) = 5.000000000, \qquad \sqrt{xy} = 5.000000002.$$

Not only the basic laws of arithmetic, but also some fundamental facts of analysis lose their validity on the computer.

**DEMONSTRATION 1.2-5:** Let $a < b$, let $f$ be a real function continuous on the interval $[a, b]$, and let $f(a) < 0, f(b) > 0$. Then, according to the intermediate value theorem, there exists $x$ so that

$$f(x) = 0.$$

Consider however

$$f(x) := x^3 - 3.$$

We find

$$f(1.442249570) = -0.000000002,$$