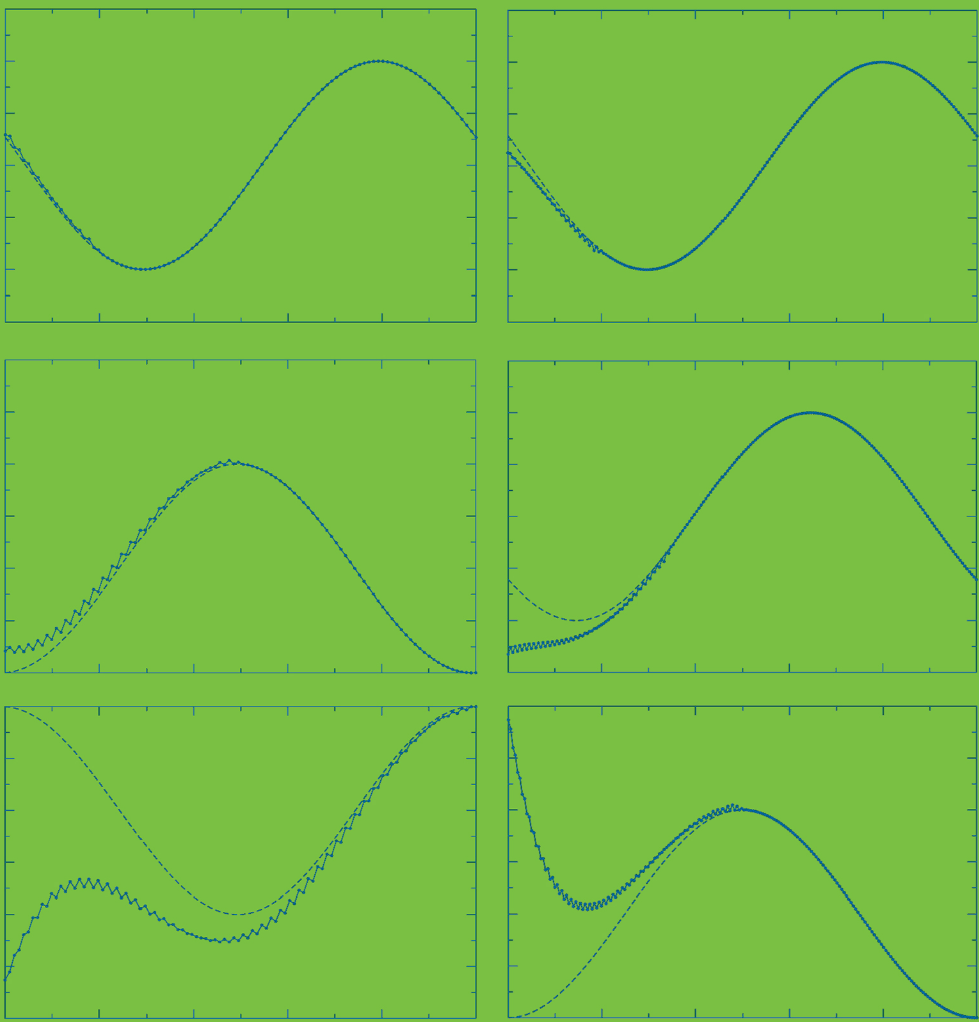
A THEORETICAL INTRODUCTION TO NUMERICAL ANALYSIS



Victor S. Ryaben'kii
Semyon V. Tsynkov

Chapman & Hall/CRC
Taylor & Francis Group

A THEORETICAL INTRODUCTION TO NUMERICAL ANALYSIS

Victor S. Ryaben'kii
Semyon V. Tsynkov

Chapman & Hall/CRC
Taylor & Francis Group
Boca Raton London New York

Chapman & Hall/CRC is an imprint of the
Taylor & Francis Group, an informa business

Chapman & Hall/CRC
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-10: 1-58488-607-2 (Hardcover)
International Standard Book Number-13: 978-1-58488-607-5 (Hardcover)

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Preface

This book introduces the key ideas and concepts of numerical analysis. The discussion focuses on how one can represent different mathematical models in a form that enables their efficient study by means of a computer. The material learned from this book can be applied in various contexts that require the use of numerical methods. The general methodology and principles of numerical analysis are illustrated by specific examples of the methods for real analysis, linear algebra, and differential equations. The reason for this particular selection of subjects is that these methods are proven, provide a number of well-known efficient algorithms, and are used for solving different applied problems that are often quite distinct from one another.

The contemplated readership of this book consists of beginning graduate and senior undergraduate students in mathematics, science and engineering. It may also be of interest to working scientists and engineers. The book offers a first mathematical course on the subject of numerical analysis. It is carefully structured and can be read in its entirety, as well as by selected parts. The portions of the text considered more difficult are clearly identified; they can be skipped during the first reading without creating any substantial gaps in the material studied otherwise. In particular, more difficult subjects are discussed in Sections 2.3.1 and 2.3.3, Sections 3.1.3 and 3.2.7, parts of Sections 4.2 and 9.7, Section 10.5, Section 12.2, and Chapter 14.

Hereafter, numerical analysis is interpreted as a mathematical discipline. The basic concepts, such as discretization, error, efficiency, complexity, numerical stability, consistency, convergence, and others, are explained and illustrated in different parts of the book with varying levels of depth using different subject material. Moreover, some ideas and views that are addressed, or at least touched upon in the text, may also draw the attention of more advanced readers. First and foremost, this applies to the key notion of the saturation of numerical methods by smoothness. A given method of approximation is said to be saturated by smoothness if, because of its design, it may stop short of reaching the intrinsic accuracy limit (unavoidable error) determined by the smoothness of the approximated solution and by the discretization parameters. If, conversely, the accuracy of approximation self-adjusts to the smoothness, then the method does not saturate. Examples include algebraic vs. trigonometric interpolation, Newton-Cotes vs. Gaussian quadratures, finite-difference vs. spectral methods for differential equations, etc.

Another advanced subject is an introduction to the method of difference potentials in Chapter 14. This is the first account of difference potentials in the educational literature. The method employs discrete analogues of modified Calderon's potentials and boundary projection operators. It has been successfully applied to solving a variety of direct and inverse problems in fluids, acoustics, and electromagnetism.

This book covers three semesters of instruction in the framework of a commonly used curriculum with three credit hours per semester. Three semester-long courses can be designed based on Parts I, II, and III of the book, respectively. Part I includes interpolation of functions and numerical evaluation of definite integrals. Part II covers direct and iterative solution of consistent linear systems, solution of overdetermined linear systems, and solution of nonlinear equations and systems. Part III discusses finite-difference methods for differential equations. The first chapter in this part, Chapter 9, is devoted to ordinary differential equations and serves an introductory purpose. Chapters 10, 11, and 12 cover different aspects of finite-difference approximation for both steady-state and evolution partial differential equations, including rigorous analysis of stability for initial boundary value problems and approximation of the weak solutions for nonlinear conservation laws. Alternatively, for the curricula that introduce numerical differentiation right after the interpolation of functions and quadratures, the material from Chapter 9 can be added to a course based predominantly on Part I of the book.

A rigorous mathematical style is maintained throughout the book, yet very little use is made of the apparatus of functional analysis. This approach makes the book accessible to a much broader audience than only mathematicians and mathematics majors, while not compromising any fundamentals in the field. A thorough explanation of the key ideas in the simplest possible setting is always prioritized over various technicalities and generalizations. All important mathematical results are accompanied by proofs. At the same time, a large number of examples are provided that illustrate how those results apply to the analysis of individual problems.

This book has no objective whatsoever of describing as many different methods and techniques as possible. On the contrary, it treats only a limited number of well-known methodologies, and only for the purpose of exemplifying the most fundamental concepts that unite different branches of the discipline. A number of important results are given as exercises for independent study. Altogether, many exercises supplement the core material; they range from elementary to quite challenging.

Some exercises require computer implementation of the corresponding techniques. However, no substantial emphasis is put on issues related to programming. In other words, any computer implementation serves only as an illustration of the relevant mathematical concepts and does not carry an independent learning objective. For example, it may be useful to have different iteration schemes implemented for a system of linear algebraic equations. By comparing how their convergence rates depend on the condition number, one can subsequently judge the efficiency from a mathematical standpoint. However, other efficiency issues, e.g., runtime efficiency determined by the software and/or computer platform, are not addressed as there is no direct relation between them and the mathematical analysis of numerical methods.

Likewise, no substantial emphasis is put on any specific applications. Indeed, the goal is to clearly and concisely present the key mathematical concepts pertinent to the analysis of numerical methods. This provides a foundation for the subsequent specialized training. Subjects such as computational fluid dynamics, computational acoustics, computational electromagnetism, etc., are very well addressed in the literature. Most corresponding books require some numerical background from the reader, the background of precisely the kind that the current text offers.

Acknowledgments

This book has a Russian language prototype [Rya00] that withstood two editions: in 1994 and in 2000. It serves as the main numerical analysis text at Moscow Institute for Physics and Technology. The authors are most grateful to the rector of the Institute at the time, Academician O. M. Belotserkovskii, who has influenced the original concept of this textbook.

Compared to [Rya00], the current book is completely rewritten. It accommodates the differences that exist between the Russian language culture and the English language culture of mathematics education. Moreover, the current textbook includes a very considerable amount of additional material.

When writing Part III of the book, we exploited the ideas and methods previously developed in [GR64] and [GR87].

When writing Chapter 14, we used the approach of [Rya02, Introduction].

We are indebted to all our colleagues and friends with whom we discussed the subject of teaching the numerical analysis. The book has greatly benefited from all those discussions. In particular, we would like to thank S. Abarbanel, K. Brushlinskii, V. Demchenko, A. Chertock, L. Choudov, L. Demkowicz, A. Ditkowski, R. Fedorenko, G. Fibich, P. Gremaud, T. Hagstrom, V. Ivanov, C. Kelley, D. Keyes, A. Kholodov, V. Kosarev, A. Kurganov, C. Meyer, N. Onofrieva, I. Petrov, V. Pirogov, L. Strygina, E. Tadmor, E. Turkel, S. Utyuzhnikov, and A. Zabrodin. We also remember the late K. Babenko, O. Lokutsievskii, and Yu. Radvogin.

We would like to specially thank Alexandre Chorin of UC Berkeley and David Gottlieb of Brown University who read the manuscript prior to publication.

A crucial and painstaking task of proofreading the manuscript was performed by the students who took classes on the subject of this book when it was in preparation. We are most grateful to L. Bilbro, A. Constantinescu, S. Ernsberger, S. Grove, A. Peterson, H. Qasimov, A. Sampat, and W. Weiselquist. All the imperfections still remaining are a sole responsibility of the authors.

It is also a pleasure for the second author to thank Arje Nachman and Richard Albanese of the US Air Force for their consistent support of the second author's research work during and beyond the period of time when the book was written.

And last, but not least, we are very grateful to the CRC Press Editor, Sunil Nair, as well as to the company staff in London and in Florida, for their advice and assistance.

Finally, our deepest thanks go to our families for their patience and understanding without which this book project would have never been completed.

V. Ryaben'kii, Moscow, Russia
S. Tsynkov, Raleigh, USA August 2006

Chapter 1

Introduction

Modern numerical mathematics provides a theoretical foundation behind the use of electronic computers for solving applied problems. A mathematical approach to any such problem typically begins with building a model for the phenomenon of interest (situation, process, object, device, laboratory/experimental setting, etc.). Classical examples of mathematical models include definite integrals, equation of a pendulum, the heat equation, equations of elasticity, equations of electromagnetic waves, and many other equations of mathematical physics. For comparison, we should also mention here a model used in formal logics — the Boolean algebra.

Analytical methods have always been considered a fundamental means for studying the mathematical models. In particular, these methods allow one to obtain closed form exact solutions for some special cases (for example, tabular integrals). There are also classes of problems for which one can obtain a solution in the form of a power series, Fourier series, or some other expansion. In addition, a certain role has always been played by approximate computations. For example, quadrature formulae are used for the evaluation of definite integrals.

The advent of computers in the middle of the twentieth century has drastically increased our capability of performing approximate computations. Computers have essentially transformed approximate computations into a dominant tool for the analysis of mathematical models. Analytical methods have not lost their importance, and have even gained some additional "functionality" as components of combined analytical/computational techniques and as verification tools. Yet sophisticated mathematical models are analyzed nowadays mostly with the help of computers. Computers have dramatically broadened the applicability range of mathematical methods in many traditional areas, such as mechanics, physics, and engineering. They have also facilitated a rapid expansion of the mathematical methods into various non-traditional fields, such as management, economics, finance, chemistry, biology, psychology, linguistics, ecology, and others.

Computers provide a capability of storing large (but still finite) arrays of numbers, and performing arithmetic operations with these numbers according to a given program that would run with a fast (but still finite) execution speed. Therefore, computers may only be appropriate for studying those particular models that are described by finite sets of numbers and require no more than finite sequences of arithmetic operations to be performed. Besides the arithmetic operations per se, a computer model can also contain comparisons between numbers that are typically needed for the automated control of subsequent computations.

In the traditional fields, one frequently employs such mathematical models as functions, derivatives, integrals, and differential equations. To enable the use of computers, these original models must therefore be (approximately) replaced by the new models that would only be based on finite arrays of numbers supplemented by finite sequences of arithmetic operations for their processing (i.e., finite algorithms). For example, a function can be replaced by a table of its numerical values; the derivative

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

can be replaced by an approximate formula, such as

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

where h is fixed (and small); a definite integral can be replaced by its integral sum; a boundary value problem for the differential equation can be replaced by the problem of finding its solution at the discrete nodes of some grid, so that by taking a suitable (i.e., sufficiently small) grid size an arbitrary desired accuracy can be achieved. In so doing, among the two methods that could seem equivalent at first glance, one may produce good results while the other may turn out completely inapplicable. The reason can be that the approximate solution it generates would not approach the exact solution as the grid size decreases, or that the approximate solution would turn out overly sensitive to the small round-off errors.

The subject of numerical analysis is precisely the theory of those models and algorithms that are applicable, i.e., that can be efficiently implemented on computers. This theory is intimately connected with many other branches of mathematics: Approximation theory and interpolation of functions, ordinary and partial differential equations, integral equations, complexity theory for functional classes and algorithms, etc., as well as with the theory and practice of programming languages. In general, both the exploratory capacity and the methodological advantages that computers deliver to numerous applied areas are truly unparalleled. Modern numerical methods allow, for example, the computation of the flow of fluid around a given aerodynamic configuration, e.g., an airplane, which in most cases would present an insurmountable task for analytical methods (like a non-tabular integral).

Moreover, the use of computers has enabled an entirely new scientific methodology known as computational experiment, i.e., computations aimed at verifying the hypotheses, as well as at monitoring the behavior of the model, when it is not known ahead of time what may interest the researcher. In fact, computational experiment may provide a sufficient level of feedback for the original formulation of the problem to be noticeably refined. In other words, numerical computations help accumulate the vital information that eventually allows one to identify the most interesting cases and results in a given area of study. Many remarkable observations, and even discoveries, have been made along this route that empowered the development of the theory and have found important practical applications as well.

Computers have also facilitated the application of mathematical methods to non-traditional areas, for which few or no "compact" mathematical models, such as differential equations, are readily available. However, other models can be built that lend themselves to the analysis by means of a computer. A model of this kind can often be interpreted as a direct numerical counterpart (such as encoding) of the object of interest and of the pertinent relations between its elements (e.g., a language or its abridged subset and the corresponding words and phrases). The very possibility of studying such models on a computer prompts their construction, which, in turn, requires that the rules and guiding principles that govern the original object be clearly and unambiguously identified. On the other hand, the results of computer simulations, e.g., a machine translation of the simplified text from one language to another, provide a practical criterion for assessing the adequacy of the theories that constitute the foundation of the corresponding mathematical model (e.g., linguistic theories).

Furthermore, computers have made it possible to analyze probabilistic models that require large amounts of test computations, as well as the so-called imitation models that describe the object or phenomenon of interest without simplifications (e.g., functional properties of a telephone network).

The variety of problems that can benefit from the use of computers is huge. For solving a given problem, one would obviously need to know enough specific detail. Clearly, this knowledge cannot be obtained ahead of time for all possible scenarios.

Therefore, the purpose of this book is rather to provide a systematic perspective on those fundamental ideas and concepts that span across different applied disciplines and can be considered established in the field of numerical analysis. Having mastered the material of this book, one should encounter little or no difficulties when receiving subsequent specialized training required for the successful work in a given research or industrial field. The general methodology and principles of numerical analysis are illustrated in the book by "sampling" the methods designed for mathematical analysis, linear algebra, and differential equations. The reason for this particular selection is that the aforementioned methods are most mature, lead to a number of well-known, efficient algorithms, and are extensively used for solving various applied problems that are often quite distant from one another.

Let us mention here some of the general ideas and concepts that require the most thorough attention in every particular setting. These general ideas acquire a concrete interpretation and meaning in the context of each specific problem that needs to be solved on a computer. They are the discretization of the problem, conditioning of the problem, numerical error, and computational stability of a given algorithm. In addition, comparison of the algorithms along different lines obviously plays a central role when selecting a specific method. The key criteria for comparison are accuracy, storage, and operation count requirements, as well as the efficiency of utilization of the input information. On top of that, different algorithms may vary in how amenable they are to parallelization — a technique that allows one to conduct computations simultaneously on multi-processor computer platforms.

In the rest of the Introduction, we provide a brief overview of the foregoing notions and concepts. It helps create a general perspective on the subject of numerical mathematics, and establishes a foundation for studying the subsequent material.

1.1 Discretization

Let $f(x)$ be a function of the continuous argument $x \in [0, 1]$. Assume that this function provides (some of) the required input data for a given problem that needs to be approximately solved on a computer. The value of the function f at every given x can be either measured or obtained numerically. Then, to store this function in the memory of a computer, one may need to approximately characterize it with a table of values at a finite set of points: $x_1, x_2, \dots, x_n$. This is an elementary example of discretization: The problem of storing the function defined on the interval $[0, 1]$, which is a continuum of points, is replaced by the problem of storing a table of its discrete values at the subset of points $x_1, x_2, \dots, x_n$ that all belong to this interval.

Let now $f(x)$ be sufficiently smooth, and assume that we need to calculate its derivative at a given point x. The problem of exactly evaluating the expression

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

that contains a limit can be replaced by the problem of computing an approximate value of this expression using one of the following formulae:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \tag{1.1}$$

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}, \tag{1.2}$$

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{1.3}$$

Similarly, the second derivative $f''(x)$ can be replaced by the finite formula:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{1.4}$$

One can show that all these formulae become more and more accurate as h becomes smaller; this is the subject of Exercise 1, and the details of the analysis can be found in Section 9.2.1. Moreover, for every fixed h, each formula (1.1)–(1.4) will only require a finite set of values of f and a finite number of arithmetic operations. These formulae are examples of discretization for the derivatives $f'(x)$ and $f''(x)$.

Let us now consider a boundary value problem:

$$\begin{gathered} \frac{d^2y}{dx^2} - x^2 y = \cos x, \qquad 0 \le x \le 1, \\ y(0) = 2, \qquad y(1) = 3, \end{gathered} \tag{1.5}$$

where the unknown function $y = y(x)$ is defined on the interval $0 \le x \le 1$. To construct a discrete approximation of problem (1.5), let us first partition the interval

$[0, 1]$ into N equal sub-intervals of size $h = N^{-1}$. Instead of the continuous function $y(x)$, we will be looking for a finite set of its values $y_0, y_1, \dots, y_N$ on the grid $x_k = kh$, $k = 0, 1, \dots, N$. At the interior nodes of this grid: x_k, $k = 1, 2, \dots, N-1$, we can approximately replace the second derivative $y''(x)$ by expression (1.4). After substituting into the differential equation of (1.5) this yields:

$$\frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} - (kh)^2 y_k = \cos(kh), \qquad k = 1, 2, \dots, N-1. \tag{1.6}$$

Furthermore, the boundary conditions at $x = 0$ and at $x = 1$ from (1.5) translate into:

$$y_0 = 2, \qquad y_N = 3. \tag{1.7}$$

The system of $N+1$ linear algebraic equations (1.6), (1.7) contains exactly as many unknowns $y_0, y_1, \dots, y_N$, and renders a discrete counterpart of the boundary value problem (1.5). One can, in fact, show that the finer the grid, i.e., the larger the N, the more accurate will the approximation be that the discrete solution of problem (1.6), (1.7) provides for the continuous solution of problem (1.5). Later, this fact will be formulated and proven rigorously.

Let us denote the continuous boundary value problem (1.5) by M_∞, and the discrete boundary value problem (1.6), (1.7) by M_N. By taking $N = 2, 3, \dots$, we associate an infinite sequence of discrete problems $\{M_N\}$ with the continuous problem M_∞. When computing the solution to a given problem M_N for any fixed N, we only have to work with a finite array of numbers that specify the input data, and with a finite set of unknown quantities $y_0, y_1, y_2, \dots, y_N$. It is, however, the entire infinite sequence of finite discrete models $\{M_N\}$ that plays the central role from the standpoint of numerical mathematics. Indeed, as those models happen to be more and more accurate, we can always choose a sufficiently large N that would guarantee any desired accuracy of approximation.

In general, there are many different ways of transitioning from a given continuous problem M_∞ to the sequence $\{M_N\}$ of its discrete counterparts. In other words, the approximation (1.6), (1.7) of the boundary value problem (1.5) is by no means the only one possible. Let $\{M_N\}$ and $\{M'_N\}$ be two sequences of approximations, and let us also assume that the computational costs of obtaining the discrete solutions of M_N and M'_N are the same. Then, a better method of discretization would be the one that provides the same accuracy of approximation with a smaller value of N.

Let us also note that for two seemingly equivalent discretization methods M_N and M'_N, it may happen that one will approximate the continuous solution of problem M_∞ with an increasingly high accuracy as N increases, whereas the other will yield "an approximate solution" that would bear less and less resemblance to the continuous solution of M_∞. We will encounter situations like this in Part III of the book, where we also discuss how the corresponding difficulties can be partially or fully overcome.

Exercises

1. Let $f(x)$ have as many bounded derivatives as needed. Show that the approximation error of formulae (1.1), (1.2), (1.3), and (1.4), is $\mathscr{O}(h)$, $\mathscr{O}(h)$, $\mathscr{O}(h^2)$, and $\mathscr{O}(h^2)$.

1.2 Conditioning

Speaking in most general terms, for any given problem one can basically identify the input data and the output result(s), i.e., the solution, so that the former determine the latter. In this book, we will mostly analyze problems for which the solution exists and is unique. If, in addition, the solution depends continuously on the data, i.e., if for a vanishing perturbation of the data the corresponding perturbation of the solution will also be vanishing, then the problem is said to be *well-posed.*

A somewhat more subtle characterization of the problem, on top of its well-posedness, is known as the *conditioning*. It has to do with quantifying the sensitivity of the solution, or some of its key characteristics, to perturbations of the input data. This sensitivity may vary strongly for different problems that could otherwise look very similar. If it is "low" (weak), then the problem is said to be *well conditioned;* if, conversely, the sensitivity is "high" then the problem is *ill conditioned*. The notions of low and high are, of course, problem-specific. We emphasize that the concept of conditioning pertains to both continuous and discrete problems. Typically, not only do ill conditioned problems require excessively accurate definition of the input data, but also appear more difficult for computations.

Consider, for example, the quadratic equation $x^2 - 2\alpha x + 1 = 0$ for $|\alpha| > 1$. It has two real roots that can be expressed as functions of the argument α: $x_{1,2} = \alpha \pm \sqrt{\alpha^2 - 1}$. We will interpret α as the datum in the problem, and $x_1 = x_1(\alpha)$ and $x_2 = x_2(\alpha)$ as the corresponding solution. Clearly, the sensitivity of the solution to the perturbations of α can be characterized by the magnitude of the derivatives $\frac{dx_{1,2}}{d\alpha} = 1 \pm \frac{\alpha}{\sqrt{\alpha^2-1}}$. Indeed, $\Delta x_{1,2} \approx \frac{dx_{1,2}}{d\alpha}\Delta\alpha$. We can easily see that the derivatives $\frac{dx_{1,2}}{d\alpha}$ are small for large $|\alpha|$, but they become large when α approaches 1. We can therefore conclude that the problem of finding the roots of $x^2 - 2\alpha x + 1 = 0$ is well conditioned when $|\alpha| \gg 1$, and ill conditioned when $|\alpha| = \mathscr{O}(1)$. We should also note that conditioning can be improved if, instead of the original quadratic equation, we consider its equivalent $x^2 - \frac{1+\beta^2}{\beta}x + 1 = 0$, where $\beta = \alpha + \sqrt{\alpha^2 - 1}$. In this case, $x_1 = \beta$ and $x_2 = \beta^{-1}$; the two roots coincide for $|\beta| = 1$, or equivalently, $|\alpha| = 1$. However, the problem of evaluating $\beta = \beta(\alpha)$ is still ill conditioned near $|\alpha| = 1$.

Our next example involves a simple ordinary differential equation. Let $y = y(t)$ be the concentration of some substance at the time t, and assume that it satisfies:

$$\frac{dy}{dt} - 10y = 0.$$

Let us take an arbitrary t_0, $0 \le t_0 \le 1$, and perform an approximate measurement of the actual concentration $y_0 = y(t_0)$ at this moment of time, thus obtaining:

$$y\Big|_{t=t_0} = y_0^*.$$

Our overall task will be to determine the concentration $y = y(t)$ at all other moments of time t from the interval $[0, 1]$.

If we knew the quantity $y_0 = y(t_0)$ exactly, then we could have used the exact formula available for the concentration:

$$y(t) = y_0 e^{10(t-t_0)}. \tag{1.8}$$

We, however, only know the approximate value $y_0^* \approx y_0$ of the unknown quantity y_0. Therefore, instead of (1.8), the next best thing is to employ the approximate formula:

$$y^*(t) = y_0^* e^{10(t-t_0)}. \tag{1.9}$$

Clearly, the error $y^* - y$ of the approximate formula (1.9) is given by:

$$y^*(t) - y(t) = (y_0^* - y_0)e^{10(t-t_0)}, \quad 0 \leq t \leq 1.$$

Assume now that we need to measure y_0^* to the the accuracy δ, $|y_0^* - y_0| < \delta$, that would be sufficient to guarantee an initially prescribed tolerance ε for determining $y(t)$ everywhere on the interval $0 \leq t \leq 1$, i.e., would guarantee the error estimate:

$$|y^*(t) - y(t)| < \varepsilon, \quad 0 \leq t \leq 1.$$

It is easy to see that $\max\limits_{0 \leq t \leq 1} |y^*(t) - y(t)| = |y^*(1) - y(1)| = |y_0^* - y_0| e^{10(1-t_0)}$. This yields the following constraint that the accuracy δ of measuring y_0 must satisfy:

$$\delta \leq \varepsilon e^{-10(1-t_0)}. \tag{1.10}$$

Let y_0 be measured at the moment of time $t_0 = 0$. Then, inequality (1.10) would imply that this measurement has to be e^{10} times, i.e., thousands of times, more accurate than the required guaranteed accuracy of the result ε. In other words, the answer $y(t)$ appears quite sensitive to the error in specifying the input data y_0, and the problem is ill conditioned.

On the other hand, if y_0 were to be measured at $t_0 = 1$, then $\delta = \varepsilon$, and it would be sufficient to conduct the measurement with a considerably lower accuracy than the one required in the case of $t_0 = 0$. This problem is well conditioned.

Exercises

1. Consider the problem of computing $y(x) = \frac{1+x}{1-x}$ as a function of x, for $x \in (1/2, 1)$ and also for $x \in (-1, 0)$. On which of the two intervals is this problem better conditioned with respect to the perturbations of x?
2. Let $y = \sqrt{2} - 1$. Equivalently, one can write $y = (\sqrt{2} + 1)^{-1}$. Which of the two formulae is more sensitive to the error when $\sqrt{2}$ is approximated by a finite decimal fraction?
   Hint. Compare absolute values of derivatives for the functions $(x - 1)$ and $(x + 1)^{-1}$.

1.3 Error

In any computational problem, one needs to find the solution given some appropriate input data. If the solution can be obtained with an ideal accuracy, then there is no error. Typically, however, there is a certain error content in every feasible numerical solution. This error may be attributed to (at least) three different mechanisms.

First, the input data are often specified with some degree of uncertainty that, in turn, will generate uncertainty in the corresponding output. Then, the solution to the problem of interest may only be obtained with an error called *unavoidable error.*

Second, even if we eliminate the foregoing uncertainty by fixing the input data, and subsequently compute the solution using an approximate method, then we still won't find the solution that would exactly correspond to the specified data. There will be *error due to the choice of an approximate computational procedure*.

Third, the chosen approximate method is not implemented exactly either, because of *round-off errors* that arise when performing computations on a real machine.

Therefore, the overall error in the solution consists of unavoidable error, the error of the method, and round-off error. We will now illustrate these concepts.

1.3.1 Unavoidable Error

Assume that we need to find the value y of some function $y = f(x)$ for a given $x = x_0$. The quantity x_0, as well as the relation f itself that associates the value of the function with every given value of its argument, are considered the input data of the problem, whereas the quantity $y = y(x_0)$ will be its solution.

Now let the function $f(x)$ be known approximately rather than exactly, say, $f(x) \approx \sin x$, and suppose that $f(x)$ may differ from $\sin x$ by no more than a specified $\varepsilon > 0$:

$$\sin x - \varepsilon \leq f(x) \leq \sin x + \varepsilon. \tag{1.11}$$

Let the value of the argument $x = x_0$ be also measured approximately: $x = x_0^*$, so that regarding the actual x_0 we can only say that

$$x_0^* - \delta \leq x_0 \leq x_0^* + \delta, \tag{1.12}$$

where $\delta > 0$ characterizes the accuracy of the measurement.

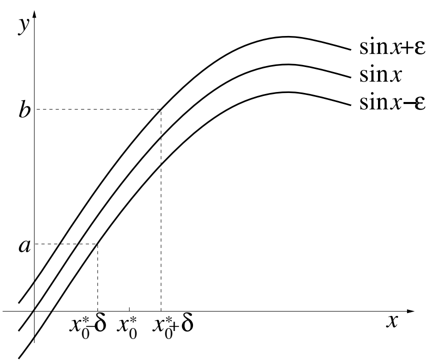


FIGURE 1.1: Unavoidable error.

One can easily see from Figure 1.1 that any point on the interval $[a, b]$ of variable y, where $a = \sin(x_0^* - \delta) - \varepsilon$ and $b = \sin(x_0^* + \delta) + \varepsilon$, can serve in the capacity of $y = f(x_0)$. Clearly, by taking an arbitrary $y^* \in [a, b]$ as the approximate value of $y = f(x_0)$, one can always guarantee the error estimate:

$$|y - y^*| \leq |b - a|. \tag{1.13}$$

For the given uncertainty in the input data, see formulae (1.11) and (1.12), this estimate cannot be considerably improved. In fact, the best error estimate that one can guarantee is obtained by choosing y^* exactly in the middle of the interval $[a,b]$:

$$y^* = y^*_{\text{opt}} = (a+b)/2.$$

From Figure 1.1 we then conclude that

$$|y - y^*| \leq |b-a|/2. \tag{1.14}$$

This inequality transforms into an exact equality when $y(x_0) = a$ or when $y(x_0) = b$.

As such, the quantity $|b-a|/2$ is precisely the unavoidable (or irreducible) error, i.e., the minimum error content that will always be present in the solution and that cannot be "dodged" no matter how the approximation y^* is actually chosen, simply because of the uncertainty that exists in the input data. For the optimal choice of the approximate solution y^*_{opt} the smallest error (1.14) can be guaranteed; otherwise, the appropriate error estimate is (1.13).

We see, however, that the optimal error estimate (1.14) is not that much better than the general estimate (1.13). We will therefore stay within reason if we interpret any arbitrary point $y^* \in [a,b]$, rather than only y^*_{opt}, as an approximate solution for $y(x_0)$ obtained within the limits of the unavoidable error. In so doing, the quantity $|b-a|$ shall replace $|b-a|/2$ of (1.14) as the estimate of the unavoidable error.

Along with the simplest illustrative example of Figure 1.1, let us consider another example that would be a little more realistic and would involve one of the most common problem formulations in numerical analysis, namely, that of reconstructing a function of continuous argument given its tabulated values at some discrete set of points. More precisely, let the values $f(x_k)$ of the function $f = f(x)$ be known at the equidistant grid nodes $x_k = kh$, $h > 0$, $k = 0, \pm 1, \pm 2, \ldots$. Let us also assume that the first derivative of $f(x)$ is bounded everywhere: $|f'(x)| \leq 1$, and that together with $f(x_k)$, this is basically all the information that we have about $f(x)$. We need to be able to obtain the (approximate) value of $f(x)$ at an arbitrary "intermediate" point x that does not necessarily coincide with any of the nodes x_k.

A large variety of methods have been developed in the literature for solving this problem. Later, we will consider interpolation by means of algebraic (Chapter 2) and trigonometric (Chapter 3) polynomials. There are other ways of building the approximating polynomials, e.g., the least squares fit, and there are other types of functions that can be used as a basis for the approximation, e.g., wavelets. Each specific method will obviously have its own accuracy. We, however, are going to show that *irrespective of any particular technique* used for reconstructing $f(x)$, there will always be error due to incomplete specification of the input data. This error merely reflects the uncertainty in the formulation; it is unavoidable and cannot be suppressed by any "smart" choice of the reconstruction procedure.

Consider the simplest case $f(x_k) = 0$ for all $k = 0, \pm 1, \pm 2, \ldots$. Clearly, the function $f_1(x) \equiv 0$ has the required trivial table of values, and also $|f_1'(x)| \leq 1$. Along with $f_1(x)$, it is easy to find another function that would satisfy the same constraints, e.g., $f_2(x) = \frac{h}{\pi} \sin\left(\frac{\pi x}{h}\right)$. Indeed, $f_2(x_k) = 0$, and $|f_2'(x)| = \left|\cos\left(\frac{\pi x}{h}\right)\right| \leq 1$. We therefore see that there are at least two different functions that cannot be told apart based on the available information. Consequently, the error $\max\limits_x |f_1(x) - f_2(x)| = \mathcal{O}(h)$ is unavoidable when reconstructing the function $f(x)$, given its tabulated values $f(x_k)$ and the fact that its first derivative is bounded, no matter what specific reconstruction methodology may be employed.

For more on the notion of the unavoidable error in the context of reconstructing continuous functions from their discrete values see Section 2.2.4 of Chapter 2.

1.3.2 Error of the Method

Let $y^* = \sin x_0^*$. The number y^* belongs to the interval $[a, b]$; it can be considered a non-improvable approximate solution of the first problem analyzed in Section 1.3.1. For this solution, the error satisfies estimate (1.13) and is unavoidable. The point $y^* = \sin x_0^*$ has been selected among other points of the interval $[a, b]$ only because it is given by the formula convenient for subsequent analysis.

To evaluate the quantity $y^* = \sin x_0^*$ on a computer, let us use Taylor's expansion for the function $\sin x$:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots.$$

Thus, for computing y^* one can take one of the following approximate expressions:

$$\begin{aligned}
y^* &\approx y_1^* = x_0^*, \\
y^* &\approx y_2^* = x_0^* - \frac{(x_0^*)^3}{3!}, \\
&\dots\dots \quad \dots \quad \dots \quad \dots \\
y^* &\approx y_n^* = \sum_{k=1}^{n} (-1)^{k-1} \frac{(x_0^*)^{2k-1}}{(2k-1)!}.
\end{aligned} \tag{1.15}$$

By choosing a specific formulae (1.15) for the approximate evaluation of y^*, we select our method of computation. The quantity $|y^* - y_n^*|$ is then known as *the error of the computational method*. In fact, we are considering a family of methods parameterized by the integer n. The larger the n the smaller the error, see (1.15); and by taking a sufficiently large n we can always make sure that the associated error will be smaller than any initially prescribed threshold.

It, however, does not make sense to drive the computational error much further down than the level of the unavoidable error. Therefore, the number n does not need to be taken excessively large. On the other hand, if n is taken too small so that the error of the method appears much larger than the unavoidable error, then one can say that the chosen method does not fully utilize the information about the solution that is contained in the input data, or equivalently, loses a part of this information.

1.3.3 Round-off Error

Assume that we have fixed the computational method by selecting a particular n in (1.15), i.e., by setting $y^* \approx y_n^*$. When calculating this y_n^* on an actual computer, we will, generally speaking, obtain a different value $\tilde{y}_n^*$ due to rounding. Rounding is an intrinsic feature of the floating-point arithmetic on computers, as they only operate with numbers that can be represented as finite binary fractions of a given fixed length. As such, all other real numbers (e.g., infinite fractions) may only be stored approximately in the computer memory, and the corresponding approximation procedure is known as rounding. The error $|y_n^* - \tilde{y}_n^*|$ is called *the round-off error.*

This error shall not noticeably exceed the error of the computational method. Otherwise, a loss of the overall accuracy will be incurred due to the round-off error.

Exercises[1]

1. Assume that we need to calculate the value $y = f(x)$ of some function $f(x)$, while there is an uncertainty in the input data x^*: $x^* - \delta \le x \le x^* + \delta$.

   How does the corresponding unavoidable error depend on x^* and on δ for the following functions:

   a) $f(x) = \sin x$;

   b) $f(x) = \ln x$, where $x > 0$?

   For what values of x^*, obtained by approximately measuring the "loose" quantity x with the accuracy δ, can one guarantee only a one-sided upper bound for $\ln x$ in problem b)? Find this upper bound.

2. Let the function $f(x)$ be defined by its values sampled on the grid $x_k = kh$, where $h = 1/N$ and $k = 0, \pm 1, \pm 2, \ldots$. In addition to these discrete values, assume that $\max\limits_x |f''(x)| \le 1$.

   Prove that as the available input data are incomplete, they do not, generally speaking, allow one to reconstruct the function at an arbitrary given point x with accuracy better than the unavoidable error $\varepsilon(h) = h^2/\pi^2$.

   Hint. Show that along with the function $f(x) \equiv 0$, which obviously has all its grid values equal to zero, another function, $\varphi(x) = \left(h^2/\pi^2\right)\sin(N\pi x)$, also has all its grid values equal to zero, and satisfies the condition $\max\limits_x |\varphi''(x)| \le 1$, while $\max\limits_x |f(x) - \varphi(x)| = h^2/\pi^2$.

3. Let $f = f(x)$ be a function, such that the absolute value of its second derivative does not exceed 1. Show that the approximation error for the formula:

   $$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

   will not exceed h.

4. Let $f = f(x)$ be a function that has bounded second derivative: $\forall x: \ |f''(x)| \le 1$. For any x, the value of the function $f(x)$ is measured and comes out to be equal to some $f^*(x)$; in so doing we assume that the accuracy of the measurement guarantees the following estimate:

   $$|f(x) - f^*(x)| \le \varepsilon, \qquad \varepsilon > 0.$$

   Suppose now that we need to approximately evaluate the first derivative $f'(x)$.

[1] Hereafter, we will be using the symbol * to indicate the increased level of difficulty for a given problem.

a) How shall one choose the parameter h so that to minimize the guaranteed error estimate of the approximate formula:

$$f'(x) \approx \frac{f^*(x+h) - f^*(x)}{h}.$$

b) Show that given the existing uncertainty in the input data, the unavoidable error of evaluating $f'(x)$ is at least $\mathscr{O}\left(\sqrt{\varepsilon}\right)$, no matter what specific method is used.

Hint. Consider two functions, $f(x) \equiv 0$ and $f^*(x) = \varepsilon \sin\left(x/\sqrt{\varepsilon}\right)$. Clearly, the absolute value of the second derivative for either of these two functions does not exceed 1. Moreover, $\max_x |f(x) - f^*(x)| \le \varepsilon$. At the same time,

$$\left|\frac{df^*}{dx} - \frac{df}{dx}\right| = \left|\sqrt{\varepsilon}\cos\frac{x}{\sqrt{\varepsilon}}\right| = \mathscr{O}\left(\sqrt{\varepsilon}\right).$$

By comparing the solutions of sub-problems a) and b), verify that the specific approximate formula for $f'(x)$ given in a) yields the error of the irreducible order $\mathscr{O}\left(\sqrt{\varepsilon}\right)$; and also show that the unavoidable error is, in fact, exactly of order $\mathscr{O}\left(\sqrt{\varepsilon}\right)$.

5. For storing the information about a linear function $f(x) = kx + b$, $\alpha \le x \le \beta$, that satisfies the inequalities: $0 \le f(x) \le 1$, we use a table with six available cells, such that one of the ten digits: $0, 1, 2, \dots, 9$, can be written into each cell.

What is the unavoidable error of reconstructing the function, if the foregoing six cells of the table are filled according to one of the following recipes?

a) The first three cells contain the first three digits that appear right after the decimal point when the number $f(\alpha)$ is represented as a normalized decimal fraction; and the remaining three cells contain the first three digits after the decimal point in the normalized decimal fraction for $f(\beta)$.

b) Let $\alpha = 0$ and $\beta = 10^{-2}$. The first three cells contain the first three digits in the normalized decimal fraction for k, the fourth cell contains either 0 or 1 depending on the sign of k, and the remaining two cells contain the first two digits after the decimal point in the normalized decimal fraction for b.

c)* Show that irrespective of any specific strategy for filling out the aforementioned six-cell table, the unavoidable error of reconstructing the linear function $f(x) = kx + b$ is always at least 10^{-3}.

Hint. Build 10^6 different functions from the foregoing class, such that the maximum modulus of the difference between any two of them will be at least 10^{-3}.

1.4 On Methods of Computation

Suppose that a mathematical model is constructed for studying a given object or phenomenon, and subsequently this model is analyzed using mathematical and computational means. For example, under certain assumptions the following problem:

$$\begin{aligned} &\frac{d^2y}{dt^2} + y = 0, \quad t \geq 0, \\ &y(0) = 0, \quad \left.\frac{dy}{dt}\right|_{t=0} = 1, \end{aligned} \tag{1.16}$$

can provide an adequate mathematical model for small oscillations of a pendulum, where $y(t)$ is the pendulum displacement from its equilibrium at the time t.

A study of harmonic oscillations based on this mathematical model, i.e., on the Cauchy problem (1.16), can benefit from a priori knowledge about the physical nature of the object of study. In particular, one can predict, based on physical reasoning, that the motion of the pendulum will be periodic. However, once the mathematical model (1.16) has been built, it becomes a separate and independent object that can be investigated using any available mathematical tools, including those that have little or no relation to the physical origins of the problem. For example, the numerical value of the solution $y = \sin t$ to problem (1.16) at any given moment of time $t = z$ can be obtained by expanding $\sin z$ into the Taylor series:

$$\sin z = z - \frac{z^3}{3!} + \frac{z^5}{5!} - \dots,$$

and subsequently taking its appropriate partial sum. In so doing, representation of the function $\sin t$ as a power series hardly admits any tangible physical interpretation.

In general, when solving a given problem on the computer, many different methods, or different algorithms, can be used. Some of them may prove far superior to others. In subsequent parts of the book, we are going to describe a number of established, robust and efficient, algorithms for frequently encountered classes of problems in numerical analysis. In the meantime, let us briefly explain how the algorithms may differ.

Assume that for computing the solution y to a given problem we can employ two algorithms, A_1 and A_2, that yield the approximate solutions $y_1^* = A_1(X)$ and $y_2^* = A_2(X)$, respectively, where X denotes the entire required set of the input data. In so doing, a variety of situations may occur.

1.4.1 Accuracy

The algorithm A_2 may be more accurate than the algorithm A_1, that is:

$$|y - y_1^*| \gg |y - y_2^*|.$$

For example, let us approximately evaluate $y = \sin x\big|_{x=0.1}$ using the expansion:

$$y_n^* = \sum_{k=1}^{n} (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!}. \tag{1.17}$$

The algorithm A_1 will correspond to taking $n=1$ in formula (1.17), and the algorithm A_2 will correspond to taking $n=2$ in formula (1.17). Then, obviously,

$$|\sin 0.1 - y_1^*| \gg |\sin 0.1 - y_2^*|.$$

1.4.2 Operation Count

Both algorithms may provide the same accuracy, but the computation of $y_1^* = A_1(X)$ may require many more arithmetic operations than the computation of $y_2^* = A_2(X)$. Suppose, for example, that we need to find the value of

$$y = 1 + x + x^2 + \ldots + x^{1023} \qquad \left(\text{clearly, } y = \frac{1-x^{1024}}{1-x}\right)$$

for $x = 0.99$. Let A_1 be the algorithm that would perform the computations directly using the given formula, i.e., by raising 0.99 to the powers $1, 2, \ldots, 1023$ one after another, and subsequently adding the results. Let A_2 be the algorithm that would perform the computations according to the formula:

$$y = \frac{1-0.99^{1024}}{1-0.99}.$$

The accuracy of these two algorithms is the same — both are absolutely accurate provided that there are no round-off errors. However, the first algorithm requires considerably more arithmetic operations, i.e., it is computationally more expensive. Namely, for successively computing

$$x, \quad x^2 = x \cdot x, \quad \ldots, \quad x^{1023} = x^{1022} \cdot x,$$

one will have to perform 1022 multiplications. On the other hand, to compute 0.99^{1024} one only needs 10 multiplications:

$$0.99^2 = 0.99 \cdot 0.99, \quad 0.99^4 = 0.99^2 \cdot 0.99^2, \quad \ldots, \quad 0.99^{1024} = 0.99^{512} \cdot 0.99^{512}.$$

1.4.3 Stability

The algorithms, again, may yield the same accuracy, but $A_1(X)$ may be computationally unstable, whereas $A_2(X)$ may be stable. For example, to evaluate $y = \sin x$ with the prescribed tolerance $\varepsilon = 10^{-3}$, i.e., to guarantee $|y - y^*| \le 10^{-3}$, let us employ the same finite Taylor expansion as in formula (1.17):

$$y_1^* = y_1^*(x) = \sum_{k=1}^{n} (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!}, \tag{1.18}$$

where $n = n(\varepsilon)$ is to be chosen to ensure that the inequality

$$|y - y_1^*| \le 10^{-3}$$

will hold. The first algorithm A_1 will compute the result directly according to (1.18). If $|x| \leq \pi/2$, then by noticing that the following inequality holds already for $n = 5$:

$$\frac{1}{(2n-1)!}\left(\frac{\pi}{2}\right)^{2n-1} \leq 10^{-3},$$

we can reduce the sum (1.18) to

$$y_1^* = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}.$$

Clearly, the computations by this formula will only be weakly sensitive to round-off errors when evaluating each term on the right-hand side. Moreover, as for $|x| \leq \pi/2$, those terms rapidly decay when the power grows, there is no room for the cancellation of significant digits, and the algorithm A_1 will be computationally stable.

Consider now $|x| \gg 1$; for example, $x = 100$. Then, for achieving the prescribed accuracy of $\varepsilon = 10^{-3}$, the number n should satisfy the inequality:

$$\frac{100^{2n-1}}{(2n-1)!} \leq 10^{-3},$$

which yields an obvious conservative lower bound for n: $n > 49$. This implies that the terms in sum (1.18) become small only for sufficiently large n. At the same time, the first few leading terms in this sum will be very large. A small relative error committed when computing those terms will result in a large absolute error; and since taking a difference of large quantities to evaluate a small quantity $\sin x$ ($|\sin x| \leq 1$) is prone to the loss of significant digits (see Section 1.4.4), the algorithm A_1 in this case will be computationally unstable.

On the other hand, in the case of large x a stable algorithm A_2 for evaluating $\sin x$ is also easy to build. Let us represent a given x in the form $x = l\pi + z$, where $|z| \leq \pi/2$ and l is integer. Then,

$$\sin x = (-1)^l \sin z,$$
$$y_2^* = A_2(x) = (-1)^l \left(z - \frac{z^3}{3!} + \frac{z^5}{5!} - \frac{z^7}{7!}\right).$$

This algorithm has the same stability properties as the algorithm A_1 for $|x| \leq \pi/2$.

1.4.4 Loss of Significant Digits

Most typically, numerical instability manifests itself through a strong amplification of the small round-off errors in the course of computations. A key mechanism for the amplification is the loss of significant digits, which is a purely computer-related phenomenon that only occurs because the numbers inside a computer are represented as finite (binary) fractions (see Section 1.3.3). If computers could operate with infinite fractions (no rounding), then this phenomenon would not take place.

Consider two real numbers a and b represented in a computer by finite fractions with m significant digits after the decimal point:

$$a = 0.a_1a_2a_3\dots a_m,$$
$$b = 0.b_1b_2b_3\dots b_m.$$

We are assuming that both numbers are normalized and that they have the same exponent that we are leaving out for simplicity. Suppose that these two numbers are close to one another, i.e., that the first k out of the total of m digits coincide:

$$a_1 = b_1,\ a_2 = b_2,\ \dots,\ a_k = b_k.$$

Then the difference $a-b$ will only have $m-k<m$ significant digits (provided that $a_{k+1} > b_{k+1}$, which we, again, assume for simplicity):

$$a-b = 0.\underbrace{0\dots0}_{k}c_{k+1}\dots c_m.$$

The reason for this reduction from m to $m-k$, which is called *the loss of significant digits,* is obvious. Even though the actual numbers a and b may be represented by the fractions much longer than m digits, or by infinite fractions, the computer simply has no information about anything beyond digit number m. Even if the result $a-b$ is subsequently normalized:

$$a-b = 0.c_{k+1}\dots c_m\underbrace{c_{m+1}\dots c_{m+k}}_{\text{artifacts}}\cdot\beta^{-k},$$

where β is the radix, or base ($\beta = 2$ for all computers), then the digits from c_{m+1} through c_{m+k} will still be completely artificial and will have nothing to do with the true representation of $a-b$.

It is clear that the loss of significant digits may lead to a very considerable degradation of the overall accuracy. The error once committed at an intermediate stage of the computation will not disappear and will rather "propagate" further and contaminate the subsequent results. Therefore, when organizing the computations, it is not advisable to compute small numbers as differences of large numbers. For example, suppose that we need to evaluate the function $f(x) = 1-\cos x$ for x which is close to 1. Then $\cos x$ will also be close to 1, and significant digits could be lost when computing $A_1(x) = 1-\cos x$. Of course, there is an easy fix for this difficulty. Instead of the original formula we should use $f(x) = A_2(x) = 2\sin^2\frac{x}{2}$.

The loss of significant digits may cause an instability even if the original continuous problem is well conditioned. Indeed, assume that we need to compute the value of the function $f(x) = \sqrt{x}-\sqrt{x-1}$. Conditioning of this problem can be judged by evaluating the maximum ratio of the resulting relative error in the solution over the eliciting relative error in the input data:

$$\sup_{\Delta x}\frac{|\Delta f|/|f|}{|\Delta x|/|x|} \approx |f'(x)|\frac{|x|}{|f|} = \frac{1}{2}\left|\frac{1}{\sqrt{x}}-\frac{1}{\sqrt{x-1}}\right|\frac{|x|}{\left|\sqrt{x}-\sqrt{x-1}\right|} = \frac{|x|}{2\sqrt{x}\sqrt{x-1}}.$$

For large x the previous quantity is approximately equal to $\frac{1}{2}$, which means that the problem is perfectly well conditioned. Yet we can expect to incur a loss of significant digits when $x \gg 1$. Consider, for example, $x = 12345$, and assume that we are operating in a six-digit decimal arithmetic. Then:

$$\begin{aligned}\sqrt{x-1} &= 111.10355529865\ldots \approx 111.104,\\ \sqrt{x} &= 111.10805551354\ldots \approx 111.108,\end{aligned}$$

and consequently, $A_1(x) = \sqrt{x} - \sqrt{x-1} \approx 111.108 - 111.104 = 0.004$. At the same time, the true answer is $f(x) = 0.004500214891\ldots$, which implies that our approximate computation carries an error of roughly 11%. To understand where this error is coming from, consider f as a function of two arguments: $f = f(t_1,t_2) = t_1 - t_2$, where $t_1 = \sqrt{x}$ and $t_2 = \sqrt{x-1}$. Conditioning with respect to the second argument t_2 can be estimated as follows:

$$\left|\frac{\partial f}{\partial t_2}\right| \cdot \frac{|t_2|}{|f|} = \frac{t_2}{|t_1 - t_2|},$$

and we conclude that this number is large when t_2 is close to t_1, which is precisely the case for large x. In other words, although the entire function is well conditioned, there is an intermediate stage that is ill conditioned, and it gives rise to large errors in the course of computation. This example illustrates why it is practically impossible to design a stable numerical procedure for an ill conditioned continuous problem.

A remedy to overcome the previous hurdle is quite easy to find:

$$f(x) = A_2(x) = \frac{1}{\sqrt{x} + \sqrt{x-1}} \approx \frac{1}{111.104 + 111.108} = 0.00450020701\ldots.$$

This is a considerably more accurate answer.

Yet another example is given by the same quadratic equation $x^2 - 2\alpha x + 1 = 0$ as we considered in Section 1.2. The roots $x_{1,2}(\alpha) = \alpha \pm \sqrt{\alpha^2 - 1}$ have been found to be ill conditioned for α close to 1. However, for $\alpha \gg 1$ both roots are clearly well conditioned. In particular, for $x_2 = \alpha - \sqrt{\alpha^2 - 1}$ we have:

$$\left|\frac{dx_2(\alpha)}{d\alpha}\right| \cdot \frac{|\alpha|}{|x_2|} = \frac{\alpha}{\sqrt{\alpha^2 - 1}} \longrightarrow 1, \quad \text{as} \quad \alpha \longrightarrow +\infty.$$

Nevertheless, the computation by the formula $x_2(\alpha) = \alpha - \sqrt{\alpha^2 - 1}$ will obviously be prone to the loss of significant digits for large α. A cure may be to compute $x_1(\alpha) = \alpha + \sqrt{\alpha^2 - 1}$ and then $x_2 = 1/x_1$. Note that even for the equation $x^2 - 2\alpha x - 1 = 0$, for which both roots $x_{1,2}(\alpha) = \alpha \pm \sqrt{\alpha^2 + 1}$ are well conditioned for all α, the computation of $x_2(\alpha) = \alpha - \sqrt{\alpha^2 + 1}$ is still prone to the loss of significant digits and as such, to instability.

1.4.5 Convergence

Finally, the algorithm may be either convergent or divergent. Suppose we need to compute the value of $y = \ln(1+x)$. Let us employ the power series:

$$y = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots \tag{1.19}$$

and set

$$y^*(x) \approx y_n^* = \sum_{k=1}^{n} (-1)^{k+1} \frac{x^k}{k}. \tag{1.20}$$

In doing so, we will obtain a method of approximately evaluating $y = \ln(1+x)$ that will depend on n as a parameter.

If $|x| = q < 1$, then $\lim\limits_{n\to\infty} y_n^*(x) = y(x)$, i.e., the error committed when computing $y(x)$ according to formula (1.20) will be vanishing as n increases. If, however, $x > 1$, then $\lim\limits_{n\to\infty} y_n^*(x) = \infty$, because the convergence radius for the series (1.19) is $r = 1$. In this case the algorithm based on formula (1.20) diverges, and cannot be used for computations.

1.4.6 General Comments

Basically, the properties of continuous well-posedness and numerical stability, as well as those of ill and well conditioning, are independent. There are, however, certain relations between these concepts.

- First of all, it is clear that no numerical method can ever fix a continuous ill-posedness.[2]
- For a well-posed continuous problem there may be stable and unstable discretizations.
- Even for a well conditioned continuous problem one can still obtain both stable and unstable discretizations.
- For an ill-conditioned continuous problem a discretization will typically be unstable.

Altogether, we can say that numerical methods cannot improve things in the perspective of well-posedness and conditioning.

In the book, we are going to discuss some other characteristics of numerical algorithms as well. We will see the algorithms that admit easy parallelization, and those that are limited to sequential computations; algorithms that automatically adapt to specific characteristics of the input data, such as their smoothness, and those that only partially take it into account; algorithms that have a straightforward logical structure, as well as the more elaborate ones.

[2] The opposite of well-posedness, when there is no continuous dependence of the solution on the data.

Exercises

1. Propose an algorithm for evaluating $y = \ln(1+x)$ that would also apply to $x > 1$.
2. Show that the intermediate stages of the algorithm A_2 from page 17 are well conditioned, and there is no danger of losing significant digits when computing:

$$f(x) = A_2(x) = \frac{1}{\sqrt{x} + \sqrt{x-1}}.$$

3. Consider the problem of evaluating the sequence of numbers $x_0, x_1, \ldots, x_N$ that satisfy the difference equations:

$$2x_n - x_{n+1} = 1 + n^2/N^2, \quad n = 0, 1, \ldots, N-1,$$

and the additional condition:

$$x_0 + x_N = 1. \tag{1.21}$$

We introduce two algorithms for computing x_n. First, let

$$x_n = u_n + cv_n, \quad n = 0, 1, \ldots, N. \tag{1.22}$$

Then, in the algorithm A_1 we define u_n, $n = 0, 1, \ldots, N$, as solution of the system:

$$2u_n - u_{n+1} = 1 + n^2/N^2, \quad n = 0, 1, \ldots, N-1, \tag{1.23}$$

subject to the initial condition:

$$u_0 = 0. \tag{1.24}$$

Consequently, the sequence v_n, $n = 0, 1, \ldots, N$, is defined by the equalities:

$$2v_n - v_{n+1} = 0, \quad n = 0, 1, \ldots, N-1, \tag{1.25}$$

$$v_0 = 1, \tag{1.26}$$

and the constant c of (1.22) is obtained from the condition (1.21). In so doing, the actual values of u_n and v_n are computed consecutively using the formulae:

$$\begin{aligned} u_{n+1} &= 2u_n - (1 + n^2/N^2), \quad n = 0, 1, \ldots, \\ v_{n+1} &= 2^{n+1}, \quad n = 0, 1, \ldots. \end{aligned}$$

In the algorithm A_2, u_n, $n = 0, 1, \ldots, N$, is still defined as solution to system (1.23), but instead of the condition (1.24) an alternative condition $u_N = 0$ is employed. The sequence v_n, $n = 0, 1, \ldots, N$, is again defined as a solution to system (1.25), but instead of the condition (1.26) we use $v_N = 1$.

   a) Verify that the second algorithm, A_2, is stable while the first one, A_1, is ("violently") unstable.

   b) Implement both algorithms on the computer and try to compare their performance for $N = 10$ and for $N = 100$.

Part I

Interpolation of Functions. Quadratures

One of the key concepts in mathematics is that of a function. In the simplest case, the function $y = f(x)$, $a \leq x \leq b$, can be specified in the closed form, i.e., defined by means of a finite formula, say, $y = x^2$. This formula can subsequently be transformed into a computer code that will calculate the value of $y = x^2$ for every given x. In real-life settings, however, the functions of interest are rarely available in the closed form. Instead, a finite array of numbers, commonly referred to as the table, would often be associated with the function $y = f(x)$. By processing the numbers from the table in a particular prescribed way, one should be able to obtain an approximate value of the function $f(x)$ at any point x. For instance, a table can contain several leading coefficients of a power series for $f(x)$. In this case, processing the table would mean calculating the corresponding partial sum of the series.

Let us, for example, take the function

$$y = e^x, \quad 0 \leq x \leq 1, \quad e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots,$$

for which the power series converges for all x, and consider the table

$$1, \frac{1}{1!}, \frac{1}{2!}, \ldots, \frac{1}{n!}$$

of its $n+1$ leading Taylor coefficients, where $n > 0$ is given. The larger the n, the more accurately can one reconstruct the function $f(x) = e^x$ from this table. In so doing, the formula

$$e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!}$$

is used for processing the table.

In most cases, however, the table that is supposed to characterize the function $y = f(x)$ would not contain its Taylor coefficients, and would rather be obtained by sampling the values of this function at some finite set of points $x_0, x_1, \ldots, x_n \in [a, b]$. In practice, sampling can be rendered by either measurements or computations. This naturally gives rise to the problem of reconstructing (e.g., interpolating) the function $f(x)$ at the "intermediate" locations x that do not necessarily coincide with any of the nodes $x_0, x_1, \ldots, x_n$.

The two most widely used and most efficient interpolation techniques are algebraic interpolation and trigonometric interpolation. We are going to analyze both of them. In addition, in the current Part I of the book we will also consider the problem of evaluating definite integrals of a given function when the latter, again, is specified by a finite table of its numerical values. The motivation behind considering this problem along with interpolation is that the main approaches to approximate evaluation of definite integrals, i.e., to obtaining the so-called quadrature formulae, are very closely related to the interpolation techniques.

Before proceeding further, let us also mention several books for additional reading on the subject: [Hen64, IK66, CdB80, Atk89, PT96, QSS00, Sch02, DB03].

Chapter 2

Algebraic Interpolation

Let $x_0, x_1, \ldots, x_n$ be a given set of points, and let $f(x_0), f(x_1), \ldots, f(x_n)$ be values of the function $f(x)$ at these points (assumed known). The one-to-one correspondence

x_0	x_1	$\ldots$	x_n
$f(x_0)$	$f(x_1)$	$\ldots$	$f(x_n)$

will be called *a table of values of the function* $f(x)$ *at the nodes* $x_0, x_1, \ldots, x_n$. We need to realize, of course, that for actual computer implementations one may only use the numbers that can be represented as finite binary fractions (Section 1.3.3 of the Introduction), whereas the values $f(x_j)$ do not necessarily have to belong to this class (e.g., $\sqrt{3}$). Therefore, the foregoing table may, in fact, contain rounded rather than true values of the function $f(x)$.

A polynomial $P_n(x) \equiv P_n(x, f, x_0, x_1, \ldots, x_n)$ of degree no greater than n that has the form

$$P_n(x) = c_0 + c_1 x + \ldots + c_n x^n$$

and coincides with $f(x_0), f(x_1), \ldots, f(x_n)$ at the nodes $x_0, x_1, \ldots, x_n$, respectively, is called *the algebraic interpolating polynomial.*

2.1 Existence and Uniqueness of Interpolating Polynomial

2.1.1 The Lagrange Form of Interpolating Polynomial

THEOREM 2.1

Let $x_0, x_1, \ldots, x_n$ be a given set of distinct interpolation nodes, and let the values $f(x_0), f(x_1), \ldots, f(x_n)$ of the function $f(x)$ be known at these nodes. There is one and only one algebraic polynomial $P_n(x) \equiv P_n(x, f, x_0, x_1, \ldots, x_n)$ of degree no greater than n that would coincide with the given $f(x_k)$ at the nodes x_k, $k = 0, 1, \ldots, n$.

PROOF We will first show that there may be no more than one interpolating polynomial, and will subsequently construct it explicitly.

Assume that there are two algebraic interpolating polynomials, $P_n^{(1)}(x)$ and $P_n^{(2)}(x)$. Then, the difference between these two polynomials, $R_n(x) = P_n^{(1)}(x) - P_n^{(2)}(x)$, is also a polynomial of degree no greater than n that vanishes at the $n+1$ points $x_0, x_1, \dots, x_n$. However, for any polynomial that is not identically equal to zero, the number of roots (counting their multiplicities) is equal to the degree. Therefore, $R_n(x) \equiv 0$, i.e., $P_n^{(1)}(x) \equiv P_n^{(2)}(x)$, which proves uniqueness.

Let us now introduce the auxiliary polynomials

$$l_k(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}.$$

It is clear that each $l_k(x)$ is a polynomial of degree no greater than n, and that the following equalities hold:

$$l_k(x_j) = \begin{cases} 1, & x_j = x_k, \\ 0, & x_j \neq x_k, \end{cases} \qquad j = 0, 1, \dots, n.$$

Then, the polynomial $P_n(x)$ given by the equality

$$\begin{aligned} P_n(x) &\stackrel{\text{def}}{=} P_n(x, f, x_0, x_1, \dots, x_n) \\ &= f(x_0)l_0(x) + f(x_1)l_1(x) + \dots + f(x_n)l_n(x) \end{aligned} \tag{2.1}$$

is precisely the interpolating polynomial that we are seeking. Indeed, its degree is no greater than n, because each term $f(x_j)l_j(x)$ is a polynomial of degree no greater than n. Moreover, it is clear that this polynomial satisfies the equalities $P_n(x_j) = f(x_j)$ for all $j = 0, 1, \dots, n$. $\square$

Let us emphasize that not only have we proven Theorem 2.1, but we have also written the interpolating polynomial explicitly using formula (2.1). This formula is known as *the Lagrange form of the interpolating polynomial.* There are other convenient forms of the unique interpolating polynomial $P_n(x, f, x_0, x_1, \dots, x_n)$. The Newton form is used particularly often.

2.1.2 The Newton Form of Interpolating Polynomial. Divided Differences

Let $f(x_a)$, $f(x_b)$, $f(x_c)$, $f(x_d)$, etc., be values of the function $f(x)$ at the given nodes x_a, x_b, x_c, x_d, etc. A Newton's divided difference of order zero $f(x_k)$ of the function $f(x)$ at the point x_k is defined as simply the value of the function at this point:

$$f(x_k) = f(x_k), \quad k = a, b, c, d, \dots.$$

A divided difference of order one $f(x_k, x_m)$ of the function $f(x)$ is defined for an arbitrary pair of points x_k, x_m (x_k and x_m do not have to be neighbors, and we allow

$x_k \gtrless x_m$) through the previously introduced divided differences of order zero:

$$f(x_k, x_m) = \frac{f(x_m) - f(x_k)}{x_m - x_k}.$$

In general, a divided difference of order n: $f(x_0, x_1, \dots, x_n)$ for the function $f(x)$ is defined through the preceding divided differences of order $n-1$ as follows:

$$f(x_0, x_1, \dots, x_n) = \frac{f(x_1, x_2, \dots, x_n) - f(x_0, x_1, \dots, x_{n-1})}{x_n - x_0}. \tag{2.2}$$

Note that all the points $x_0, x_1, \dots, x_n$ in formula (2.2) have to be distinct, but they do not have to be arranged in any particular way, say, from the smallest to the largest value of x_j or vice versa.

Having defined the Newton divided differences[1] according to (2.2), we can now represent the interpolating polynomial $P_n(x, f, x_0, x_1, \dots, x_n)$ in the following *Newton form:*

$$\begin{aligned} P_n(x, f, x_0, x_1, \dots, x_n) = {} & f(x_0) + (x - x_0) f(x_0, x_1) + \dots \\ & + (x - x_0)(x - x_1) \dots (x - x_{n-1}) f(x_0, x_1, \dots, x_n). \end{aligned} \tag{2.3}$$

Formula (2.3) itself will be proven later. In the meantime, we will rather establish several useful corollaries that it implies.

COROLLARY 2.1
The following equality holds:

$$\begin{aligned} P_n(x, f, x_0, x_1, \dots, x_n) = {} & P_{n-1}(x, f, x_0, x_1, \dots, x_{n-1}) \\ & + (x - x_0)(x - x_1) \dots (x - x_{n-1}) f(x_0, x_1, \dots, x_n). \end{aligned} \tag{2.4}$$

PROOF Immediately follows from formula (2.3). ∎

COROLLARY 2.2
The divided difference $f(x_0, x_1, \dots, x_n)$ of order n is equal to the coefficient c_n in front of the term x^n in the interpolating polynomial

$$P_n(x, f, x_0, x_1, \dots, x_n) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_0.$$

In other words, the following equality holds:

$$f(x_0, x_1, \dots, x_n) = c_n. \tag{2.5}$$

[1] A more detailed account of divided differences and their role in building the interpolating polynomials can be found, e.g., in [CdB80].

PROOF It is clear that the monomial x^n on the right-hand side of expression (2.3) is multiplied by the coefficient $f(x_0,x_1,\dots,x_n)$. ∎

COROLLARY 2.3
The divided difference $f(x_0,x_1,\dots,x_n)$ may be equal to zero if and only if the quantities $f(x_0), f(x_1),\dots,f(x_n)$ are nodal values of some polynomial $Q_m(x)$ of degree m that is strictly less than n ($m<n$).

PROOF If $f(x_0,x_1,\dots,x_n)=0$, then formula (2.3) implies that the degree of the interpolating polynomial $P_n(x,f,x_0,x_1,\dots,x_n)$ is less than n, because according to equality (2.5) the coefficient c_n in front of x^n is equal to zero. As the nodal values of this interpolating polynomial are equal to $f(x_j)$, $j=0,1,\dots,n$, we can simply set $Q_m(x)=P_n(x)$. Conversely, as the interpolating polynomial of degree no greater than n is unique (Theorem 2.1), the polynomial $Q_m(x)$ with nodal values $f(x_0), f(x_1),\dots,f(x_n)$ must coincide with the interpolating polynomial $P_n(x,f,x_0,x_1,\dots,x_n)=c_nx^n+c_{n-1}x^{n-1}+\dots+c_0$. As $m<n$, equality $Q_m(x)=P_n(x)$ implies that $c_n=0$. Then, according to formula (2.5), $f(x_0,x_1,\dots,x_n)=0$. ∎

COROLLARY 2.4
The divided difference $f(x_0,x_1,\dots,x_n)$ remains unchanged under any arbitrary permutation of its arguments $x_0,x_1,\dots,x_n$.

PROOF Due to its uniqueness, the interpolating polynomial $P_n(x)$ will not be affected by the order of the interpolation nodes. Let $x'_0,x'_1,\dots,x'_n$ be a permutation of $x_0,x_1,\dots,x_n$; then, $\forall x:\ P_n(x,f,x_0,x_1,\dots,x_n)=P_n(x,f,x'_0,x'_1,\dots,x'_n)$. Consequently, along with formula (2.3) one can write

$$\begin{aligned}P_n(x,f,x_0,x_1,\dots,x_n)&=f(x'_0)+(x-x'_0)f(x'_0,x'_1)+\dots\\&\quad+(x-x'_0)(x-x'_1)\dots(x-x'_{n-1})f(x'_0,x'_1,\dots,x'_n).\end{aligned}$$

According to Corollary 2.2, one can therefore conclude that

$$f(x'_0,x'_1,\dots,x'_n)=c_n. \tag{2.6}$$

By comparing formulae (2.5) and (2.6), one can see that $f(x_0,x_1,\dots,x_n)=f(x'_0,x'_1,\dots,x'_n)$. ∎

COROLLARY 2.5
The following equality holds:

$$f(x_0,x_1,\dots,x_n)=\frac{f(x_n)-P_{n-1}(x_n,f,x_0,x_1,\dots,x_{n-1})}{(x_n-x_0)(x_n-x_1)\dots(x_n-x_{n-1})}. \tag{2.7}$$

PROOF Let us set $x = x_n$ in equality (2.4); then its left-hand side becomes equal to $f(x_n)$, and formula (2.7) follows. ∎

THEOREM 2.2
The interpolating polynomial $P_n(x, f, x_0, x_1, \dots, x_n)$ can be represented in the Newton form, i.e., equality (2.3) does hold.

PROOF We will use induction with respect to n. For $n = 0$ (and $n = 1$) formula (2.3) obviously holds. Assume now that it has already been justified for $n = 1, 2, \dots, k$, and let us show that it will also hold for $n = k+1$. In other words, let us prove the following equality:

$$\begin{aligned} P_{k+1}(x, f, x_0, x_1, \dots, x_k, x_{k+1}) &= P_k(x, f, x_0, x_1, \dots, x_k) \\ &+ f(x_0, x_1, \dots, x_k, x_{k+1})(x - x_0)(x - x_1)\dots(x - x_k). \end{aligned} \tag{2.8}$$

Notice that due to the assumption of the induction, formula (2.3) is valid for $n \le k$. Consequently, the proofs of Corollaries 2.1 through 2.5 that we have carried out on the basis of formula (2.3) will also remain valid for $n \le k$.

To prove equality (2.8), we will first demonstrate that the polynomial $P_{k+1}(x, f, x_0, x_1, \dots, x_k, x_{k+1})$ can be represented in the form:

$$\begin{aligned} P_{k+1}(x, f, x_0, x_1, \dots, x_k, x_{k+1}) &= P_k(x, f, x_0, x_1, \dots, x_k) \\ &+ \frac{f(x_{k+1}) - P_k(x_{k+1}, f, x_0, x_1, \dots, x_k)}{(x_{k+1} - x_0)(x_{k+1} - x_1)\dots(x_{k+1} - x_k)}(x - x_0)(x - x_1)\dots(x - x_k). \end{aligned} \tag{2.9}$$

Indeed, it is clear that on the right-hand side of formula (2.9) we have a polynomial of degree no greater than $k+1$ that is equal to $f(x_j)$ at all nodes x_j, $j = 0, 1, \dots, k+1$. Therefore, the expression on the the right-hand side of (2.9) is actually the interpolating polynomial

$$P_{k+1}(x, f, x_0, x_1, \dots, x_k, x_{k+1}),$$

which proves that (2.9) is a true equality. Next, by comparing formulae (2.8) and (2.9) we see that in order to justify (2.8) we need to establish the equality:

$$f(x_0, x_1, \dots, x_k, x_{k+1}) = \frac{f(x_{k+1}) - P_k(x_{k+1}, f, x_0, x_1, \dots, x_k)}{(x_{k+1} - x_0)(x_{k+1} - x_1)\dots(x_{k+1} - x_k)}. \tag{2.10}$$

Using the same argument as in the proof of Corollary 2.4, and also employing Corollary 2.1, we can write:

$$\begin{aligned} P_k(x, f, x_0, x_1, \dots, x_k) &= P_k(x, f, x_1, x_2, \dots, x_k, x_0) \\ &= P_{k-1}(x, f, x_1, x_2, \dots, x_k) \\ &+ f(x_1, x_2, \dots, x_k, x_0)(x - x_1)(x - x_2)\dots(x - x_k). \end{aligned} \tag{2.11}$$

Then, by substituting $x = x_{k+1}$ into (2.11), we can transform the right-hand side of equality (2.10) into:

$$\begin{aligned}&\frac{f(x_{k+1}) - P_k(x_{k+1}, f, x_0, x_1, \dots, x_k)}{(x_{k+1}-x_0)(x_{k+1}-x_1)\dots(x_{k+1}-x_k)} \\ &= \frac{1}{x_{k+1}-x_0}\frac{f(x_{k+1}) - P_{k-1}(x_{k+1}, f, x_1, \dots, x_k)}{(x_{k+1}-x_1)\dots(x_{k+1}-x_k)} - \frac{f(x_1, x_2, \dots, x_k, x_0)}{x_{k+1}-x_0}.\end{aligned} \tag{2.12}$$

By virtue of Corollary 2.5, the minuend on the right-hand side of equality (2.12) is equal to:

$$\frac{1}{x_{k+1}-x_0} f(x_1, x_2, \dots, x_k, x_{k+1}),$$

whereas in the subtrahend, according to Corollary 2.4, one can change the order of the arguments so that it would coincide with

$$\frac{f(x_0, x_1, \dots, x_k)}{x_{k+1}-x_0}.$$

Consequently, the right-hand side of equality (2.12) is equal to

$$\frac{f(x_1, x_2, \dots, x_{k+1}) - f(x_0, x_1, \dots, x_k)}{x_{k+1}-x_0} \stackrel{\text{def}}{=} f(x_0, x_1, \dots, x_{k+1}).$$

In other words, equality (2.12) coincides with equality (2.10) that we need to establish in order to justify formula (2.8). This completes the proof. ∎

THEOREM 2.3

Let $x_0 < x_1 < \dots < x_n$; assume also that the function $f(x)$ is defined on the interval $x_0 \le x \le x_n$, and is n times differentiable on this interval. Then,

$$n! f(x_0, x_1, \dots, x_n) = \left.\frac{d^n f}{dx^n}\right|_{x=\xi} \equiv f^{(n)}(\xi), \tag{2.13}$$

where ξ is some point from the interval $[x_0, x_n]$.

PROOF Consider an auxiliary function

$$\varphi(x) \stackrel{\text{def}}{=} f(x) - P_n(x, f, x_0, x_1, \dots, x_n) \tag{2.14}$$

defined on $[x_0, x_n]$; it obviously has a minimum of $n+1$ zeros on this interval located at the nodes $x_0, x_1, \dots, x_n$. Then, according to the Rolle (mean value) theorem, its first derivative vanishes at least at one point in between every two neighboring zeros of $\varphi(x)$. Therefore, the function $\varphi'(x)$ will have a minimum of n zeros on the interval $[x_0, x_n]$. Similarly, the function $\varphi''(x)$ vanishes at least at one point in between every two neighboring zeros of $\varphi'(x)$, and will therefore have a minimum of $n-1$ zeros on $[x_0, x_n]$.

By continuing this line of argument, we conclude that the n-th derivative $\varphi^{(n)}(x)$ will have at least one zero on the interval $[x_0, x_n]$. Let us denote this zero by ξ, so that $\varphi^{(n)}(\xi) = 0$. Next, we differentiate identity (2.14) exactly n times and subsequently substitute $x = \xi$, which yields:

$$0 = \varphi^{(n)}(\xi) = f^{(n)}(\xi) - \frac{d^n}{dx^n} P_n(x, f, x_0, x_1, \ldots, x_n)\Big|_{x=\xi}. \tag{2.15}$$

On the other hand, according to Corollary 2.2, the divided difference $f(x_0, x_1, \ldots, x_n)$ is equal to the leading coefficient of the interpolating polynomial P_n, i.e., $P_n(x, f, x_0, x_1, \ldots, x_n) = f(x_0, x_1, \ldots, x_n)x^n + c_{n-1}x^{n-1} + \ldots + c_0$. Consequently, $\frac{d^n}{dx^n} P_n(x, f, x_0, x_1, \ldots, x_n) = n! f(x_0, x_1, \ldots, x_n)$, and therefore, equality (2.15) implies (2.13). ∎

THEOREM 2.4
The values $f(x_0), f(x_1), \ldots, f(x_n)$ of the function $f(x)$ are expressed through the divided differences $f(x_0)$, $f(x_0, x_1), \ldots,$ $f(x_0, x_1, \ldots, x_n)$ by the formulae:

$$\begin{aligned} f(x_j) = f(x_0) &+ (x_j - x_0) f(x_0, x_1) + (x_j - x_0)(x_j - x_1) f(x_0, x_1, x_2) \\ &+ (x_j - x_0)(x_j - x_1) \ldots (x_j - x_{n-1}) f(x_0, x_1, \ldots, x_n), \qquad j = 0, 1, \ldots, n, \end{aligned}$$

i.e., by linear combinations of the type:

$$f(x_j) = a_{j0} f(x_0) + a_{j1} f(x_0, x_1) + \ldots + a_{jn} f(x_0, x_1, \ldots, x_n), \qquad j = 0, 1, \ldots, n. \tag{2.16}$$

PROOF The result follows immediately from formula (2.3) and equalities $f(x_j) = P(x, f, x_0, x_1, \ldots, x_n)\big|_{x=x_j}$ for $j = 0, 1, \ldots, n$. ∎

2.1.3 Comparison of the Lagrange and Newton Forms

To evaluate the function $f(x)$ at a point x that is not one of the interpolation nodes, one can approximately set: $f(x) \approx P_n(x, f, x_0, x_1, \ldots, x_n)$.

Assume that the polynomial $P_n(x, f, x_0, x_1, \ldots, x_n)$ has already been built, but in order to try and improve the accuracy we incorporate an additional interpolation node x_{n+1} and the corresponding function value $f(x_{n+1})$. Then, to construct the interpolating polynomial $P_{n+1}(x, f, x_0, x_1, \ldots, x_{n+1})$ using the Lagrange formula (2.1) one basically needs to start from the scratch. At the same time, to use the Newton formula (2.3), see also Corollary 2.1:

$$\begin{aligned} P_{n+1}(x, f, x_0, x_1, \ldots, x_{n+1}) = &P_n(x, f, x_0, x_1, \ldots, x_n) \\ &+ (x - x_0)(x - x_1) \ldots (x - x_n) f(x_0, x_1, \ldots, x_{n+1}) \end{aligned}$$

one only needs to obtain the correction

$$(x - x_0)(x - x_1) \ldots (x - x_n) f(x_0, x_1, \ldots, x_{n+1}).$$

Moreover, one will immediately be able to see how large this correction is.

2.1.4 Conditioning of the Interpolating Polynomial

Let all the interpolation nodes $x_0, x_1, \ldots, x_n$ belong to some interval $a \leq x \leq b$. Let also the values $f(x_0)$, $f(x_1), \ldots$, $f(x_n)$ of the function $f(x)$ at these nodes be given. Hereafter, we will be using a shortened notation $P_n(x, f)$ for the interpolating polynomial $P_n(x) = P_n(x, f, x_0, x_1, \ldots, x_n)$.

Let us now perturb the values $f(x_j)$ by some quantities $\delta f(x_j)$, $j = 0, 1, \ldots, n$. Then, the interpolating polynomial $P_n(x, f)$ will change and become $P_n(x, f + \delta f)$. One can clearly see from the Lagrange formula (2.1) that $P_n(x, f + \delta f) = P_n(x, f) + P_n(x, \delta f)$. Therefore, the corresponding perturbation of the interpolating polynomial, i.e., its response to δf, will be $P_n(x, \delta f)$. For a given fixed set of $x_0, x_1, \ldots, x_n$, this perturbation depends only on δf and not on f itself. As such, one can introduce *the minimum number* L_n such that the following inequality would hold for any δf:

$$\max_{a \leq x \leq b} |P_n(x, \delta f)| \leq L_n \max_j |\delta f(x_j)|. \tag{2.17}$$

The numbers $L_n = L_n(x_0, x_1, \ldots, x_n, a, b)$ are called *the Lebesgue constants.*[2] They provide a natural measure for the sensitivity of the interpolating polynomial to the perturbations $\delta f(x_j)$ of the interpolated function $f(x)$ at the nodes x_j. The Lebesgue constants are known to grow as n increases. Their specific behavior strongly depends on how the interpolation nodes x_j, $j = 0, 1, \ldots, n$, are located on the interval $[a, b]$.

If, for example, $n = 1$, $x_0 = a$, $x_1 = b$, then $L_1 = 1$. If, however, $x_0 \neq a$ and/or $x_1 \neq b$, then $L_1 \geq \frac{b-a}{2|x_1 - x_0|}$, i.e., if x_1 and x_0 are sufficiently close to one another, then the interpolation may appear arbitrarily sensitive to the perturbations of $f(x)$. The reader can easily verify the foregoing statements regarding L_1.

In the case of equally spaced interpolation nodes:

$$x_j = a + j \cdot h, \quad j = 0, 1, \ldots, n, \quad h = \frac{b-a}{n},$$

one can show that

$$2^n > L_n > 2^{n-2} \frac{1}{\sqrt{n}} \cdot \frac{1}{n - 1/2}. \tag{2.18}$$

In other words, the sensitivity of the interpolant to any errors committed when specifying the values of $f(x_j)$ will grow rapidly (exponentially) as n increases. Note that in practice it is impossible to specify the values of $f(x_j)$ without any error, no matter how these values are actually obtained, i.e., whether they are measured (with inevitable experimental inaccuracies) or computed (subject to rounding errors).

For a rigorous proof of inequalities (2.18) we refer the reader to the literature on the theory of approximation, in particular, the monographs and texts cited in Section 3.2.7 of Chapter 3. However, an elementary treatment can also be given, and one can easily provide a qualitative argument of why the Lebesgue constants for equidistant nodes grow exponentially as the grid dimension n increases. From the

[2] Note that the Lebesgue constant L_n corresponds to interpolation on $n+1$ nodes: $x_0, \ldots, x_n$.

Lagrange form of the interpolating polynomial (2.1) and definition (2.17) it is clear that:

$$L_n = \mathscr{O}\left(\max_{a\le x\le b}\sum_{k=0}^{n}|l_k(x)|\right) \tag{2.19}$$

(later, see Section 3.2.7 of Chapter 3, we will prove an even more precise statement). Take $k\approx n/2$ and x very close to one of the edges a or b, say, $x-a=\eta\ll h$. Then,

$$\begin{aligned}
|l_k(x)| &= \left|\frac{(x-x_0)(x-x_1)\dots(x-x_{k-1})(x-x_{k+1})\dots(x-x_n)}{(x_k-x_0)(x_k-x_1)\dots(x_k-x_{k-1})(x_k-x_{k+1})\dots(x_k-x_n)}\right| \\
&\approx \frac{\eta\cdot h^{2k-1}\cdot(2k)!/k}{(h^k k!)^2} = \eta\cdot h\cdot\frac{(2k)!}{k(k!)^2} \\
&= \eta\cdot h\cdot\frac{(2\cdot4\cdot6\cdot\ldots\cdot2k)(1\cdot3\cdot5\cdot\ldots\cdot(2k-1))}{k(k!)^2} \\
&\approx \eta\cdot h\cdot\frac{(2\cdot4\cdot6\cdot\ldots\cdot2k)^2}{(k!)^2} = \eta\cdot h\cdot\frac{2^{2k}(k!)^2}{(k!)^2}\approx\eta\cdot h\cdot2^n.
\end{aligned}$$

The foregoing estimate for $|l_k(x)|$, along with the previous formula (2.19), do imply the exponential growth of the Lebesgue constants on uniform (equally spaced) interpolation grids. Let now $a=-1$, $b=1$, and let the interpolation nodes on $[a,b]$ be rather given by the formula:

$$x_j = -\cos\frac{(2j+1)\pi}{2(n+1)}, \quad j=0,1,\dots,n. \tag{2.20}$$

It is possible to show that placing the nodes according to (2.20) guarantees a much better estimate for the Lebesgue constants (again, see Section 3.2.7):

$$L_n \le \frac{2}{\pi}\ln(n+1)+1. \tag{2.21}$$

We therefore conclude that in contradistinction to the previous case (2.18), the Lebesgue constants may, in fact, grow slowly rather than rapidly, as they do on the non-equally spaced nodes (2.20). As such, even the high-degree interpolating polynomials in this case will not be overly sensitive to perturbations of the input data. Interpolation nodes (2.20) are known as the Chebyshev nodes. They will be discussed in detail in Chapter 3.

2.1.5 On Poor Convergence of Interpolation with Equidistant Nodes

One should not think that for any continuous function $f(x)$, $x\in[a,b]$, the algebraic interpolating polynomials $P_n(x,f)$ built on the equidistant nodes $x_j=a+j\cdot h$, $x_0=a$, $x_n=b$, will converge to $f(x)$ as n increases, i.e., that the deviation of $P_n(x,f)$ from $f(x)$ will decrease. For example, as has been shown by Bernstein, the sequence of interpolating polynomials obtained for the function $f(x) = |x|$ on equally spaced nodes diverges at every point of the interval $[a, b] = [-1, 1]$ except at $\{-1, 0, 1\}$.

The next example is attributed to Runge. Consider the function $f(x) = \frac{1}{x^2+1/4}$ on the same interval $[a, b] = [-1, 1]$; not only is this function continuous, but also has continuous derivatives of all orders. It is, however, possible to show that for the sequence of interpolating polynomials with equally spaced nodes the maximum difference $\max_{-1\le x\le 1} |f(x) - P_n(x, f)|$ will not approach zero as n increases.

Moreover, by working on Exercise 4 below, one will be able to see that the areas of no convergence for this function are located next to the endpoints of the interval $[-1, 1]$. For larger intervals the situation may even deteriorate and the sequence of interpolating polynomials $P_n(x, f)$ may diverge. In other words, the quantity $\max_{a\le x\le b} |f(x) - P_n(x, f)|$ may become arbitrarily large for large n's (see, e.g., [IK66]). Altogether, these convergence difficulties can be accounted for by the fact that on the complex plane the function $f(z) = \frac{1}{z^2+1/4}$ is not an entire function of its argument z, and has singularities at $z = \pm i/2$.

On the other hand, if, instead of the equidistant nodes, we use Chebyshev nodes (2.20) to interpolate either the Bernstein function $f(x) = |x|$ or the Runge function $f(x) = \frac{1}{x^2+1/4}$, then in both cases the sequence of interpolating polynomials $P_n(x, f)$ converges to $f(x)$ uniformly as n increases (see Exercise 5).

Exercises

1. Evaluate $f(1.14)$ by means of linear, quadratic, and cubic interpolation using the following table of values:

x	1.08	1.13	1.20	1.27	1.31
$f(x)$	1.302	1.386	1.509	1.217	1.284

Implement the interpolating polynomials in both the Lagrange and Newton form.

2. Let $x_j = j \cdot h$, $j = 0, \pm 1, \pm 2, \ldots$, be equidistant nodes with spacing h. Verify that the following equality holds:

$$f(x_{k-1}, x_k, x_{k+1}) = \frac{f(x_{k+1}) - 2f(x_k) + f(x_{k-1})}{2!h^2}.$$

3. Let $a = x_0$, $a < x_1 < b$, $x_2 = b$. Find the value of the Lebesgue constant L_2 when x_1 is the midpoint of $[a, b]$: $x_1 = (a+b)/2$. Show that if, conversely, $x_1 \to a$ or $x_1 \to b$, then the Lebesgue constant $L_2 = L_2(x_0, x_1, x_2, a, b)$ grows with no bound.

4. Plot the graphs of $f(x) = \frac{1}{x^2+1/4}$ and $P_n(x, f)$ from Section 2.1.5 (Runge example) on the computer and thus corroborate experimentally that there is no convergence of the interpolating polynomial on equally spaced nodes when n increases.

5. Use Chebyshev nodes (2.20) to interpolate $f(x) = |x|$ and $f(x) = \frac{1}{x^2+1/4}$ on the interval $[-1, 1]$, plot the graphs of each $f(x)$ and the corresponding $P_n(x, f)$ for $n = 10, 20, 40$, and 80, evaluate numerically the error $\max_{-1\le x\le 1} |f(x) - P_n(x, f)|$, and show that it decreases as n increases.

2.2 Classical Piecewise Polynomial Interpolation

High sensitivity of algebraic interpolating polynomials to the errors in the tabulated values of $f(x)$, as well as the "iffy" convergence of the sequence $P_n(x, f)$ on uniform grids, prompt the use of piecewise polynomial interpolation.

2.2.1 Definition of Piecewise Polynomial Interpolation

Let the function $f(x)$, $x \in [a, b]$, be defined by the table $\{f(x_0), f(x_1), \dots, f(x_n)\}$ of its numerical values at the nodes $\{a = x_0 < x_1 < x_2 < \dots < x_n = b\}$. To reconstruct this function in between the nodes $x_0, x_1, \dots, x_n$, one can use an auxiliary function that would coincide with a polynomial of a given low degree (say, the first, the second, the third, etc.) between every two neighboring nodes of the interpolation grid. This approach is known as *piecewise polynomial interpolation;* in particular, it may be *piecewise linear, piecewise quadratic, piecewise cubic, etc.*

In the case of piecewise linear interpolation on the interval $x_k \le x \le x_{k+1}$, one uses the linear interpolating polynomial $P_1(x, f, x_k, x_{k+1})$ to approximate the function $f(x)$. In the case of piecewise quadratic interpolation on the interval $x_k \le x \le x_{k+1}$, one can use either of the two polynomials: $P_2(x, f, x_k, x_{k+1}, x_{k+2})$ or $P_2(x, f, x_{k-1}, x_k, x_{k+1})$.

Piecewise polynomial interpolation of an arbitrary degree s is obtained similarly. There is always some flexibility in constructing the interpolant, and to approximate the function $f(x)$ on the interval $x_k \le x \le x_{k+1}$ one can basically use any of the polynomials $P_s(x, f, x_{k-j}, x_{k-j+1}, \dots, x_{k-j+s})$, where j is one of the integers $0, 1, \dots, s-1$. It is, however, desirable that the smaller interval $[x_k, x_{k+1}]$ be located maximally close to the middle of the larger interval $[x_{k-j}, x_{k-j+s}]$ (see Section 2.1.4). For equidistant nodes, the latter requirement translates into choosing j maximally close to $s/2$. In general, once the strategy for selecting j has been adopted, one can reconstruct $f(x)$ on $[a, b]$ in the form of a piecewise polynomial of degree s. It will be composed of the individual interpolating polynomials that correspond to different intervals $[x_k, x_{k+1}]$, $k = 0, 1, \dots, n-1$. For simplicity, we will hereafter denote the piecewise polynomial as follows:

$$P_s(x, f, x_{k-j}, x_{k-j+1}, \dots, x_{k-j+s}) = P_s(x, f_{kj}).$$

2.2.2 Formula for the Interpolation Error

Let us estimate the error

$$R_s(x) \stackrel{\text{def}}{=} f(x) - P_s(x, f_{kj}), \qquad x_k \le x \le x_{k+1}, \tag{2.22}$$

that arises when the function $f(x)$ is approximately replaced by the polynomial $P_s(x, f_{kj})$. To do so, we will need to exploit the following general theorem:

THEOREM 2.5
Let the function $f = f(t)$ be defined on $\alpha \le t \le \beta$, with a continuous derivative of order $s+1$ on this interval. Let $t_0, t_1, \dots, t_s$ be an arbitrary set of distinct points that all belong to $[\alpha, \beta]$, and let $f(t_0), f(t_1), \dots, f(t_s)$ be the values of the function $f(t)$ at these points. Finally, let $P_s(t) \equiv P_s(t, f, t_0, t_1, \dots, t_s)$ be the algebraic interpolating polynomial of degree no greater than s built for these given points and function values. Then, the interpolation error $R_s(t) = f(t) - P_s(t)$ can be represented on $[\alpha, \beta]$ as follows:

$$R_s(t) = \frac{f^{(s+1)}(\xi)}{(s+1)!}(t-t_0)(t-t_1)\dots(t-t_s), \tag{2.23}$$

where $\xi = \xi(t)$ is some point from the interval (α, β).

PROOF We first notice that formula (2.23) does hold for all nodes t_j, $j = 0, 1, \dots, s$, themselves, because on one hand $\forall t_j: \ f(t_j) - P_s(t_j) = 0$, and on the other hand, $R_s(t_j) = 0$, where $R_s(t)$ is defined by formula (2.23). Let us now take an arbitrary $\bar{t} \in [\alpha, \beta]$ that does not coincide with any of $t_0, t_1, \dots, t_s$. To prove formula (2.23) for $t = \bar{t}$, we introduce an auxiliary function:

$$\varphi(t) = f(t) - P_s(t) - k(t-t_0)(t-t_1)\dots(t-t_s) \tag{2.24}$$

and choose the parameter k so that $\varphi(\bar{t}) = 0$, which obviously implies

$$k = \frac{f(\bar{t}) - P_s(\bar{t})}{(\bar{t}-t_0)(\bar{t}-t_1)\dots(\bar{t}-t_s)}. \tag{2.25}$$

The numerator in formula (2.25) coincides with the value of the error $R_s(\bar{t})$, therefore, this formula yields:

$$R_s(\bar{t}) = k(\bar{t}-t_0)(\bar{t}-t_1)\dots(\bar{t}-t_s). \tag{2.26}$$

The auxiliary function φ of (2.24) clearly has a minimum of $s+2$ zeros on the interval $[\alpha, \beta]$ located at the points $\bar{t}, t_0, t_1, \dots, t_s$. Then, its first derivative $\varphi'(t)$ will have a minimum of $s+1$ zeros on the (open) interval (α, β), because according to the Rolle (mean value) theorem, the derivative $\varphi'(t)$ has to vanish at least once in between every two neighboring points where $\varphi(t)$ itself vanishes. Similarly, $\varphi''(t)$ will have at least s zeros on (α, β), $\varphi^{(3)}(t)$ will have at least $s-1$ zeros, etc., so that finally the derivative $\varphi^{(s+1)}(t)$ will have to have a minimum of one zero on the interval (α, β). Let us denote this zero by $\xi \in (\alpha, \beta)$, so that $\varphi^{(s+1)}(\xi) = 0$.

Next, we note that

$$\frac{d^{s+1}}{dt^{s+1}} t^{s+1} = (s+1)!$$

and that $(t-t_0)(t-t_1)\dots(t-t_s) = t^{s+1} + Q_s(t)$, where $Q_s(t)$ is a polynomial of degree no greater than s. We also note that

$$\frac{d^{s+1}}{dt^{s+1}} P_s(t) \equiv \frac{d^{s+1}}{dt^{s+1}} Q_s(t) \equiv 0.$$

Using the previous two expressions, we differentiate the function $\varphi(t)$ defined by formula (2.24) $s+1$ times and obtain:

$$\varphi^{(s+1)}(t) = f^{(s+1)}(t) - k(s+1)!.$$

Substituting $t = \xi$ into the last equality, and recalling that $\varphi^{(s+1)}(\xi) = 0$, we arrive at the following expression for k:

$$k = \frac{f^{(s+1)}(t)}{(s+1)!}.$$

Finally, by substituting k into equality (2.26) we obtain a formula for $R_s(\bar{t})$ that would actually coincide with formula (2.23) because $\bar{t} \in [\alpha, \beta]$ has been chosen arbitrarily. □

THEOREM 2.6

Under the assumptions of the previous theorem, the following estimate holds:

$$\max_{\alpha \le t \le \beta} |R_s(t)| \le \frac{1}{(s+1)!} \max_{\alpha \le t \le \beta} |f^{(s+1)}(t)| (\beta - \alpha)^{s+1}. \tag{2.27}$$

PROOF We first note that $\forall t \in [\alpha, \beta]$ the absolute value of each expression $t - t_0$, $t - t_1$, ..., $t - t_s$ will not exceed $\beta - \alpha$. Then, we use formula (2.23):

$$\begin{aligned} |R_s(t)| &= \frac{1}{(s+1)!} |f^{(s+1)}(\xi)(t - t_0)(t - t_1) \dots (t - t_s)| \\ &\le \frac{1}{(s+1)!} \max_{\alpha \le t \le \beta} |f^{(s+1)}(t)| (\beta - \alpha)^{s+1}. \end{aligned} \tag{2.28}$$

As $t \in [\alpha, \beta]$ on the left-hand side of formula (2.28) is arbitrary, the required estimate (2.27) follows. □

Let us emphasize that we have proven inequality (2.27) for an arbitrary distribution of the (distinct) interpolation nodes $t_0, t_1, \dots, t_s$ on the interval $[\alpha, \beta]$. For a given fixed distribution of nodes, estimate (2.27) can often be improved. For example, consider a piecewise linear interpolation and assume that the nodes t_0 and t_1 coincide with the endpoints α and β, respectively, of the interval $\alpha \le t \le \beta$. Then,

$$\begin{aligned} |R_1(t)| &= \left| \frac{f''(\xi)}{(s+1)!} (t - \alpha)(t - \beta) \right| \\ &\le \frac{1}{2} \max_{\alpha \le t \le \beta} |f''(t)| \max_{\alpha \le t \le \beta} |(t - \alpha)(t - \beta)| = \frac{1}{8} \max_{\alpha \le t \le \beta} |f''(t)| (\beta - \alpha)^2, \end{aligned}$$

which yields

$$\max_{\alpha \le t \le \beta} |R_1(t)| \le \frac{1}{8} \max_{\alpha \le t \le \beta} |f''(t)| (\beta - \alpha)^2, \tag{2.29}$$

whereas estimate (2.27) for $s=1$ transforms into

$$\max_{\alpha\le t\le\beta}|R_1(t)|\le\frac{1}{2}\max_{\alpha\le t\le\beta}|f''(t)|(\beta-\alpha)^2.$$

We will now use Theorems 2.5 and 2.6 to estimate the error (2.22) of piecewise polynomial interpolation of the function $f(x)$ on the interval $x_k\le x\le x_{k+1}$. First, let

$$\alpha=x_{k-j},\quad \beta=x_{k-j+s},\quad t_0=\alpha=x_{k-j},\quad t_1=x_{k-j+1},\ \dots,\ t_s=\beta=x_{k-j+s}.$$

Then, it is clear that

$$\max_{x_k\le x\le x_{k+1}}|R_s(x,f_{kj})|\le\max_{\alpha\le x\le\beta}|R_s(x,f_{kj})|,$$

and according to (2.27) we obtain

$$\max_{x_k\le x\le x_{k+1}}|R_s(x,f_{kj})|\le\frac{1}{(s+1)!}\max_{x_{k-j}\le x\le x_{k-j+s}}|f^{(s+1)}(x)|(x_{k-j+s}-x_{k-j})^{s+1}. \quad (2.30)$$

If the quantity $|f^{(s+1)}(x)|$ undergoes strong variations on the interval $[a,b]$, then, in order for the estimate (2.30) to guarantee some prescribed accuracy, it will be advantageous to have the grid size (distance between the neighboring nodes) and the value of $x_{k-j+s}-x_{k-j}$ smaller in those parts of $[a,b]$ where $|f^{(s+1)}(x)|$ is larger.

In the case of equidistant nodes $x_0,x_1,\dots,x_n$, estimate (2.30) implies

$$\max_{x_k\le x\le x_{k+1}}|R_s(x,f_{kj})|\le\frac{s^{s+1}}{(s+1)!}\max_{x_{k-j}\le x\le x_{k-j+s}}|f^{(s+1)}(x)|h^{s+1}, \quad (2.31)$$

where $h=(b-a)/n=x_{k+1}-x_k$ is the size of the interpolation grid. Inequality (2.31) can be recast as

$$\max_{x_k\le x\le x_{k+1}}|R_s(x,f_{kj})|\le\text{const}\cdot\max_{x_{k-j}\le x\le x_{k-j+s}}|f^{(s+1)}(x)|h^{s+1}, \quad (2.32)$$

where the key consideration is that the constant on the right-hand side of (2.32) *does not depend* on the grid size h.

To conclude this section, let us specifically mention the case of piecewise linear interpolation: $s=1$, $\alpha=x_k$, and $\beta=x_{k+1}$. Then, according to estimate (2.29), we have:

$$\max_{x_k\le x\le x_{k+1}}|R_1(x)|\le\frac{1}{8}\max_{x_k\le x\le x_{k+1}}|f''(x)|(x_{k+1}-x_k)^2=\frac{h^2}{8}\max_{x_k\le x\le x_{k+1}}|f''(x)|. \quad (2.33)$$

2.2.3 Approximation of Derivatives for a Grid Function

THEOREM 2.7

Let the function $f=f(x)$ be defined on the interval $[\alpha,\beta]$, and let it have a continuous derivative of order $s+1$ on this interval. Let $x_{k-j},x_{k-j+1},\dots,x_{k-j+s}$

be a set of interpolation nodes, such that $\alpha = x_{k-j} < x_{k-j+1} < \ldots < x_{k-j+s} = \beta$. *Then, to approximately evaluate the derivatives*

$$\frac{d^q f(x)}{dx^q}, \qquad q = 1, 2, \ldots, s,$$

of the function $f(x)$ *on the interval* $x_k \le x \le x_{k+1}$, *one can employ the interpolating polynomial* $P_s(x, f_{kj})$ *and set*

$$\frac{d^q f(x)}{dx^q} \approx \frac{d^q}{dx^q} P_s(x, f_{kj}), \qquad x_k \le x \le x_{k+1}. \tag{2.34}$$

In so doing, the approximation error will satisfy the estimate:

$$\begin{aligned} \max_{x_k \le x \le x_{k+1}} & \left| \frac{d^q f(x)}{dx^q} - \frac{d^q}{dx^q} P_s(x, f_{kj}) \right| \\ & \le \frac{1}{(s-q+1)!} \max_{x_{k-j} \le x \le x_{k-j+s}} |f^{(s+1)}(x)| (x_{k-j+s} - x_{k-j})^{s-q+1}. \end{aligned} \tag{2.35}$$

PROOF Consider an auxiliary function $\varphi(x) \stackrel{\text{def}}{=} f(x) - P_s(x, f_{kj})$; it obviously vanishes at all $s+1$ interpolation nodes $x_{k-j}, x_{k-j+1}, \ldots, x_{k-j+s}$. Therefore, its first derivative $\varphi'(x)$ will have a minimum of s zeros on the interval $x_{k-j} \le x \le x_{k-j+s}$, because according to the Rolle (mean value) theorem, there is a zero of the function $\varphi'(x)$ in between any two neighboring zeros of $\varphi(x)$. Similarly, the function $\frac{d^q \varphi(x)}{dx^q}$ will have at least $s-q+1$ zeros on the interval $x_{k-j} \le x \le x_{k-j+s}$. This implies that the derivative $\frac{d^q f(x)}{dx^q}$ and the polynomial $\frac{d^q}{dx^q} P_s(x, f_{kj})$ of degree no greater than $s-q$ coincide at $s-q+1$ distinct points. In other words, the polynomial $P_s^{(q)}(x, f_{kj})$ can be interpreted as an interpolating polynomial of degree no greater than $s-q$ for the function $f^{(q)}(x)$ on the interval $x_{k-j} \le x \le x_{k-j+s}$, built on some set of $s-q+1$ interpolation nodes.

Moreover, the function $f^{(q)}(x)$ has a continuous derivative of order $s-q+1$ on $[\alpha, \beta]$:

$$\frac{d^{s-q+1}}{dx^{s-q+1}} f^{(q)}(x) = \frac{d^{s+1}}{dx^{s+1}} f(x).$$

Consequently, one can use Theorem 2.6 and, by setting $\alpha = x_{k-j}$, $\beta = x_{k-j+s}$, obtain the following estimate [cf. formula (2.27)]:

$$\begin{aligned} \max_{x_{k-j} \le x \le x_{k-j+s}} & \left| f^{(q)}(x) - P_s^{(q)}(x, f_{kj}) \right| \\ & \le \frac{1}{(s-q+1)!} \max_{x_{k-j} \le x \le x_{k-j+s}} |f^{(s+1)}(x)| (x_{k-j+s} - x_{k-j})^{s-q+1}. \end{aligned}$$

As $\alpha = x_{k-j} \le x_k < x_{k+1} \le x_{k-j+s} = \beta$, it immediately yields (2.35). $\square$

2.2.4 Estimate of the Unavoidable Error and the Choice of Degree for Piecewise Polynomial Interpolation

Let the function $f = f(x)$ be defined on the interval $[0,\pi]$, and let its values be known at the nodes of the uniform grid: $x_k = k\pi/n \equiv kh$, $k = 0,1,\dots,n$. Using only the tabulated values of the function $f(x_0)$, $f(x_1),\dots,f(x_n)$, one cannot, even in principle, obtain an exact reconstruction of $f(x)$ in between the nodes, because different functions may have identical tables, i.e., may coincide at the nodes x_k, $k = 0,1,\dots,n$, and at the same time be different elsewhere. If, for example, in addition to the table of values nothing is known about the function $f(x)$ except that it is simply continuous, then one cannot guarantee any accuracy at all when reconstructing $f(x)$ at $x \neq x_k$, $k = 0,1,\dots,n$.

Assume now that $f(x)$ has a bounded derivative of the maximum order $s+1$:

$$\max_x |f^{(s+1)}(x)| \leq M_s = \text{const.} \tag{2.36}$$

It is easy to find two different functions from the class characterized by $M_s = 1$:

$$f_1(x) = \frac{\sin nx}{n^{s+1}} \quad \text{and} \quad f_2(x) = -\frac{\sin nx}{n^{s+1}},$$

that would deviate from one another by the value of order h^{s+1}:

$$\max_{0\leq x\leq\pi} |f_1(x) - f_2(x)| = \max_{0\leq x\leq\pi} 2\left|\frac{\sin nx}{n^{s+1}}\right| = \frac{2}{\pi^{s+1}}h^{s+1}, \tag{2.37}$$

and for which the tables would nonetheless fully coincide (both will be trivial):

$$f_1(x_k) = f_2(x_k) = 0, \qquad k = 0,1,\dots,n.$$

We therefore conclude that given the tabulated values of the function $f(x)$, and only estimate (2.36) in addition to that, one cannot, even in theory, reconstruct the function $f(x)$ on the interval $0 \leq x \leq \pi$ with the accuracy better than $\mathscr{O}(h^{s+1})$. In other words, the error $\mathscr{O}(h^{s+1})$ is unavoidable when reconstructing the function $f(x)$, $0 \leq x \leq \pi$, using its table of values on a uniform grid with size h.

It is also clear that

$$\max_{0\leq x\leq\pi}\left|\frac{d^q f_1(x)}{dx^q} - \frac{d^q f_2(x)}{dx^q}\right| = 2\frac{1}{n^{s-q+1}} = \frac{2}{\pi^{s-q+1}}h^{s-q+1}, \tag{2.38}$$

which means that the unavoidable error when reconstructing the derivative $\frac{d^q f(x)}{dx^q}$ is at least $\mathscr{O}(h^{s-q+1})$.

By comparing equalities (2.37) and (2.38) with estimates of the error obtained in Sections 2.2.2 and 2.2.3 for the piecewise polynomial interpolation of the function $f(x)$ and its derivatives, we conclude that the interpolation error and the unavoidable error have the same asymptotic order (of smallness) with respect to the grid size h. If, under the condition (2.36), one still chooses to use interpolating polynomials of degree $r < s$, then the interpolation error (for the function itself) will be $\mathscr{O}(h^{r+1})$. In other words, there will be an additional loss of the order of accuracy, on top of the uncertainty-based unavoidable error $\mathscr{O}(h^{s+1})$ that is due to the specification of $f(x)$ through its discrete table of values.

On the other hand, the use of interpolation of a higher degree $r > s$ cannot increase the order of accuracy beyond the threshold set by the unavoidable error $\mathscr{O}(h^{s+1})$, and therefore cannot speed up the convergence as $h \longrightarrow 0$. As such, the degree s of piecewise polynomial interpolation is optimal for the functions that satisfy (2.36).

REMARK 2.1 The considerations of the current section pertain primarily to the asymptotic behavior of the error as $h \longrightarrow 0$. For a given fixed $h > 0$, interpolation of some degree $r < s$ may, in fact, appear more accurate than the interpolation of degree s. Besides, in practice the tabulated values $f(x_k)$, $k = 0, 1, \dots, n$, may only be specified approximately, rather than exactly, with a finite fixed number of decimal (or binary) digits. In this case, the loss of interpolation accuracy due to rounding is going to increase as s increases, because of the growth of the Lebesgue constants (defined by formula (2.18) of Section 2.1.4). Therefore, the piecewise polynomial interpolation of high degree (higher than the third) is not used routinely. $\square$

REMARK 2.2 Error estimate (2.32) does, in fact, imply *uniform convergence* of the interpolant $P_s(x, f_{kj})$ (a piecewise polynomial) to the interpolated function $f(x)$ with the rate $\mathscr{O}(h^{s+1})$ as the grid is refined, i.e., as $h \longrightarrow 0$. Estimate (2.33), in particular, indicates that piecewise linear interpolation converges uniformly with the rate $\mathscr{O}(h^2)$. Likewise, estimate (2.35) in the case of a uniform grid with size h will imply uniform convergence of the q-th derivative of the interpolant $P_s^{(q)}(x, f_{kj})$ to the q-th derivative of the interpolated function $f^{(q)}(x)$ with the rate $\mathscr{O}(h^{s-q+1})$ as $h \longrightarrow 0$. $\square$

REMARK 2.3 The notion of unavoidable error as presented in this section (see also Section 1.3) illustrates the concept of *Kolmogorov diameters* for compact sets of functions (see Section 12.2.5 for more detail). Let W be a linear normed space, and let $U \subset W$. Introduce also an N-dimensional linear manifold $W^{(N)} \subset W$, for example, $W^{(N)} = \operatorname{span}\{w_1^{(N)}, w_2^{(N)}, \dots, w_N^{(N)}\}$, where the functions $w_n^{(N)} \in W$, $n = 1, 2, \dots, N$, are given. The N-dimensional Kolmogorov diameter of the set U with respect to the space W is defined as:

$$\kappa_N(U, W) = \inf_{W^{(N)} \subset W} \sup_{u \in U} \inf_{w \in W^{(N)}} \|w - u\|_W. \tag{2.39}$$

This quantity tells us how accurately we can approximate an arbitrary u from a given set $U \subset W$ by selecting the optimal approximating subspace $W^{(N)}$ whose dimension N is fixed. The Kolmogorov diameter and related concepts play a fundamental role in the modern theory of approximation; in particular, for the analysis of the so-called best approximations, for the analysis of saturation of numerical methods by smoothness (Section 2.2.5), as well as in the theory of ε-entropy and related theory of transmission and processing of information. $\square$