Contents

Chapter 1

Principles of Numerical Calculations

1.1 Introduction

[1] Although mathematics has been used for centuries in one form or another within many areas of science and industry, modern scientific computing using electronic computers has its origin in research and developments during the second world war. In the late forties and early fifties the foundation of numerical analysis was laid as a separate discipline of mathematics. The new capabilities of performing millions of operations led to new classes of algorithms, which needed a careful analysis to ensure their accuracy and stability.

Recent modern development has increased enormously the scope for using numerical methods. Not only has this been caused by the continuing advent of faster computers with larger memories. Gain in problem solving capabilities through better mathematical algorithms have in many cases played an equally important role! This has meant that today one can treat much more complex and less simplified problems through massive amounts of numerical calculations. This development has caused the always close interaction between mathematics on the one hand and science and technology on the other to increase tremendously during the last decades. Advanced mathematical models and methods are now used more and more also in areas like medicine, economics and social sciences. It is fair to say that today experiment and theory, the two classical elements of scientific method, in many fields of science and engineering are supplemented in many areas by computations as an equally important component.

As a rule, applications lead to mathematical problems which in their complete form cannot be conveniently solved with exact formulas unless one restricts oneself to special cases or simplified models which can be exactly analyzed. In many cases, one thereby reduces the problem to a linear problem—for example, a linear system of equations or a linear differential equation. Such an approach can be can quite often lead to concepts and points of view which can, at least qualitatively, be used even in the unreduced problems.

[1]This section last revised by Ake Bjorck 030115.

1.2 Common Ideas and Concepts

In most numerical methods one applies a small number of general and relatively simple ideas. These are then combined in an inventive way with one another and with such knowledge of the given problem as one can obtain in other ways—for example, with the methods of mathematical analysis. Some knowledge of the background of the problem is also of value; among other things, one should take into account the order of magnitude of certain numerical data of the problem.

In this chapter we shall illustrate the use of some general ideas behind numerical methods on some simple problems which may occur as subproblems or computational details of larger problems, though as a rule they occur in a less pure form and on a larger scale than they do here. When we present and analyze numerical methods, we use to some degree the same approach which was described first above: we study in detail special cases and simplified situations, with the aim of uncovering more generally applicable concepts and points of view which can guide in more difficult problems.

It is important to have in mind that the success of the methods presented depends on the smoothness properties of the functions involved. In this first survey we shall tacitly assume that the functions have as many well-behaved derivatives as is needed.

1.2.1 Iteration

One of the most frequently recurring ideas in many contexts is **iteration** (from the Latin *iteratio*, "repetition") or **successive approximation**. Taken generally, iteration means the repetition of a pattern of action or process. Iteration in this sense occurs, for example, in the repeated application of a numerical process—perhaps very complicated and itself containing many instances of the use of iteration in the somewhat narrower sense to be described below—in order to improve previous results. To illustrate a more specific use of the idea of iteration, we consider the problem of solving a nonlinear equation of the form

$$x = F(x), \tag{1.2.1}$$

where F is assumed to be a differentiable function whose value can be computed for any given value of a real variable x, within a certain interval. Using the method of iteration, one starts with an initial approximation x_0, and computes the sequence

$$x_1 = F(x_0), \qquad x_2 = F(x_1), \qquad x_3 = F(x_2), \dots \tag{1.2.2}$$

Each computation of the type $x_{n+1} = F(x_n)$ is called an iteration. If the sequence $\{x_n\}$ converges to a limiting value α then we have

$$\alpha = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} F(x_n) = F(\alpha),$$

so $x = \alpha$ satisfies the equation $x = F(x)$. As n grows, we would like the numbers x_n to be better and better estimates of the desired root. One then stops the iterations when sufficient accuracy has been attained.

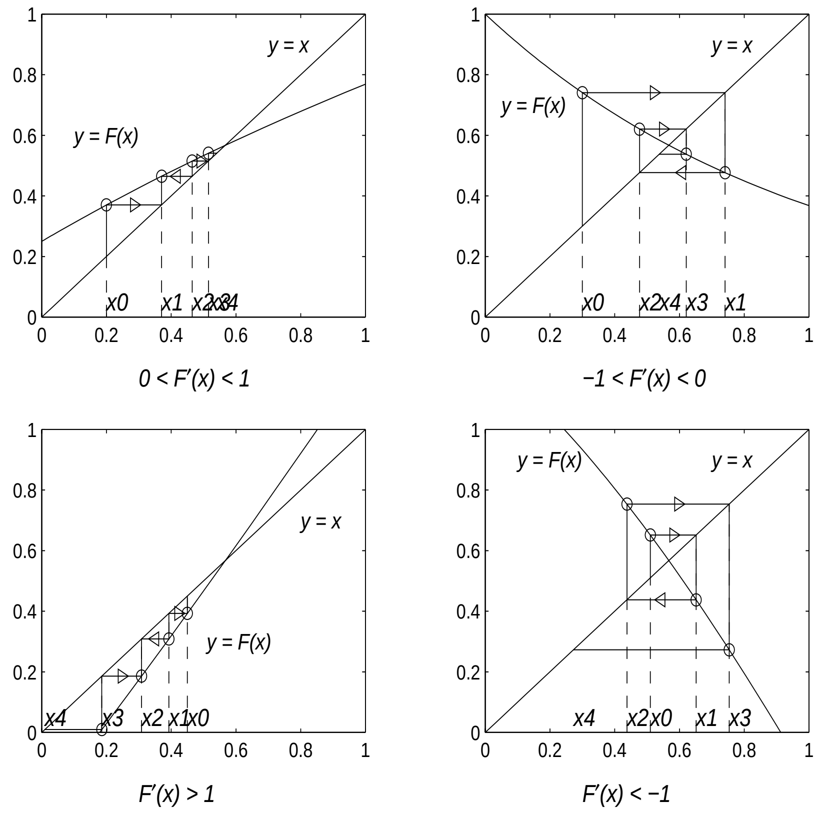


Figure 1.2.1. (a)–(d) *Geometric interpretation of iteration* $x_{n+1} = F(x_n)$.

A geometric interpretation is shown in Fig. 1.2.1. A root of Equation (1.2.1) is given by the abscissa (and ordinate) of an intersecting point of the curve $y = F(x)$ and the line $y = x$. Using iteration and starting from x_0 we have $x_1 = F(x_0)$. The point x_1 on the x-axis is obtained by first drawing a horizontal line from the point $(x_0, F(x_0)) = (x_0, x_1)$ until it intersects the line $y = x$ in the point (x_1, x_1) and from there drawing a vertical line to $(x_1, F(x_1)) = (x_1, x_2)$ and so on in a "staircase" pattern. In Fig. 1.2.1a it is obvious that $\{x_n\}$ converges monotonically to α. Fig. 1.2.1b shows a case where F is a decreasing function. There we also have convergence but not monotone convergence; the successive iterates x_n are alternately to the right and to the left of the root α.

But there are also divergent cases, exemplified by Figs. 1.2.1c and 1.2.1d. One can see geometrically that the quantity which determines the rate of convergence (or divergence) is the slope of the curve $y = F(x)$ in the neighborhood of the root. Indeed, from the mean value theorem we have

$$\frac{x_{n+1} - \alpha}{x_n - \alpha} = \frac{F(x_n) - F(\alpha)}{x_n - \alpha} = F'(\xi_n),$$

where ξ_n lies between x_n and α. We see that, if x_0 is chosen sufficiently close to the root, (yet $x_0 \neq \alpha$), the iteration will diverge if $|F'(\alpha)| > 1$ and converge if $|F'(\alpha)| < 1$. In these cases the root is called, respectively, repulsive and attractive. We also see that the convergence is faster the smaller $|F'(\alpha)|$ is.

Example 1.2.1. A CLASSICAL FAST METHOD FOR CALCULATING SQUARE ROOTS:
The equation $x^2 = c$ $(c > 0)$ can be written in the form $x = F(x)$, where $F(x) = \frac{1}{2}(x + c/x)$. If we set

$$x_{n+1} = \tfrac{1}{2}(x_n + c/x_n),$$

then the $\alpha = \lim_{n\to\infty} x_n = \sqrt{c}$ (see Fig. 1.2.2)

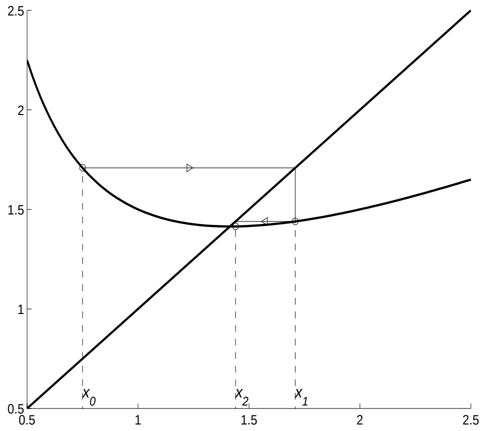


Figure 1.2.2. *The fix-point iteration $x_n = (x_n + c/x_n)/2$, $c = 2$, $x_0 = 0.75$.*

For $c = 2$, and $x_0 = 1.5$, we get $x_1 = \frac{1}{2}(1.5 + 2/1.5) = 1\frac{5}{12} = 1.4166666\ldots$, and

$$x_2 = 1.414215\,686274, \qquad x_3 = 1.414213\,562375,$$

which can be compared with $\sqrt{2} = 1.414213\,562373\ldots$ (correct to digits shown). As can be seen from Fig. 1.2.2 a rough value for x_0 suffices. The rapid convergence is due to the fact that for $\alpha = \sqrt{c}$ we have

$$F'(\alpha) = (1 - c/\alpha^2)/2 = 0.$$

One can in fact show that if x_n has t correct digits, then x_{n+1} will have at least $2t - 1$ correct digits; see Example 6.3.3 and the following exercise. The above iteration method is used quite generally on both pocket calculators and computers for calculating square roots. The computation converges for any $x_0 > 0$.

Iteration is one of the most important aids for the practical as well as theoretical treatment of both linear and nonlinear problems. One very common application of iteration is to the solution of *systems of equations*. In this case $\{x_n\}$ is a sequence of vectors, and F is a vector-valued function. When iteration is applied to *differential equations* $\{x_n\}$ means a sequence of functions, and $F(x)$ means an expression in which integration or other operations of functions may be involved. A number of other variations on the very general idea of iteration will be given in later chapters.

The form of equation (1.2.1) is frequently called the **fixed point form**, since the root α is a fixed point of the mapping F. An equation may not be given originally in this form. One has a certain amount of choice in the rewriting of equation $f(x) = 0$ in fixed point form, and the rate of convergence depends very much on this choice. The equation $x^2 = c$ can also be written, for example, as $x = c/x$. The iteration formula $x_{n+1} = c/x_n$, however, gives a sequence which alternates between x_0 (for even n) and c/x_0 (for odd n)—the sequence does not even converge!

Let an equation be given in the form $f(x) = 0$, and for any $k \neq 0$, set

$$F(x) = x + kf(x).$$

Then the equation $x = F(x)$ is equivalent to the equation $f(x) = 0$. Since $F'(\alpha) = 1 + kf'(\alpha)$, we obtain the fastest convergence for $k = -1/f'(\alpha)$. Because α is not known, this cannot be applied literally. However, if we use x_n as an approximation this leads to the choice $F(x) = x - f(x)/f'(x)$, or the iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{1.2.3}$$

This is the celebrated **Newton's method**.[2] (Occasionally this method is referred to as the Newton–Raphson method.) We shall derive it in another way below.

Example 1.2.2. The equation $x^2 = c$ can be written in the form $f(x) = x^2 - c = 0$. Newton's method for this equation becomes

$$x_{n+1} = x_n - \frac{x_n^2 - c}{2x_n} = \frac{1}{2}\Big(x_n + \frac{c}{x_n}\Big),$$

which is the fast method in Example 1.2.1.

1.2.2 Linearization and Extrapolation

Another often recurring idea is that one *locally*, that is, in a small neighborhood of a point, *approximates a more complicated function with a linear function.* We shall first illustrate the use of this idea in the solution of the equation $f(x) = 0$. Geometrically, this means that we are seeking the intersection point between the x-axis and the curve $y = f(x)$, see Fig. 1.2.3. Assume that we have an approximating

[2]Isac Newton (1642–1727), English mathematician, astronomer and physicist, invented, independently of the German mathematician and philosopher Gottfried W. von Leibniz (1646–1716), the infinitesimal calculus. Newton, the Greek mathematician Archimedes (287–212 B.C. and the German mathematician Carl Friedrich Gauss (1777–1883) gave pioneering contributions to numerical mathematics and to other sciences.

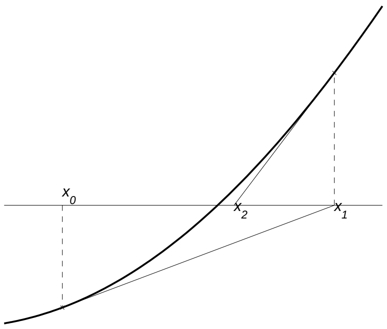


Figure 1.2.3. *Newton's method.*

value x_0 to the root. We then approximate the curve with its *tangent* at the point $(x_0, f(x_0))$. Let x_1 be the abscissa of the point of intersection between the x-axis and the tangent. Since the equation for the tangent reads

$$y - f(x_0) = f'(x_0)(x - x_0),$$

we obtain by setting $y = 0$, the approximation

$$x_1 = x_0 - f(x_0)/f'(x_0).$$

In many cases x_1 will have about twice as many correct digits as x_0. However, if x_0 is a poor approximation and $f(x)$ far from linear, then it is possible that x_1 will be a worse approximation than x_0.

If we combine the ideas of iteration and local linear approximation, that is, we substitute x_n for x_0 and x_{n+1} for x_1, we rediscover Newton's method mentioned earlier. If x_0 is close enough to α the iterations will converge rapidly, see Fig. 1.2.3, but there are also cases of divergence.

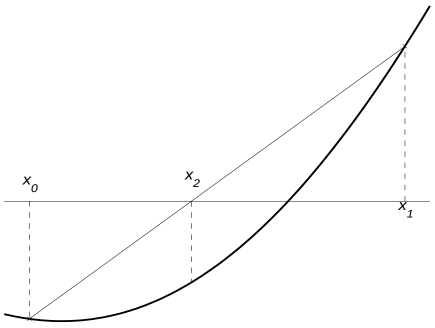


Figure 1.2.4. *The secant method.*

Another way, instead of drawing the tangent, to approximate a curve locally with a linear function is to choose two neighboring points on the curve and to approximate the curve with the *secant* which joins the two points, see Fig. 1.2.4. The **secant method** for the solution of nonlinear equations is based on this approximation. This method, which preceded Newton's method, is discussed more closely in Sec. 6.4.1.

Newton's method can easily be generalized to solve a *system of nonlinear equations*

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad i = 1 : n.$$

or $f(x) = 0$, where f and x now are vectors in $\mathbf{R}^n$. Then x_{n+1} is determined by the *system of linear equations*

$$f'(x_n)(x_{n+1} - x_n) = f(x_n),$$

where

$$f'(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \in \mathbf{R}^{n \times n},$$

is the matrix of partial derivatives of f with respect to x. This matrix is called the **Jacobian** of f and often denoted by $J(x)$. We shall several times, in later chapters, return to this fundamental method.

The secant approximation is useful in many other contexts. It is, for instance, generally used when one "reads between the lines" or interpolates in a table of numerical values. In this case the secant approximation is called **linear interpolation**. When the secant approximation is used in **numerical integration**, that is in the approximate calculation of a definite integral,

$$I = \int_a^b y(x)\, dx, \tag{1.2.4}$$

(see Fig. 1.2.5) it is called the **trapezoidal rule**. With this method, the area between the curve $y = y(x)$ and the x-axis is approximated with the sum $T(h)$ of the areas of a series of parallel trapezoids.

Using the notation of Fig. 1.2.5, we have

$$T(h) = h\frac{1}{2}\sum_{i=0}^{n-1}(y_i + y_{i+1}), \quad h = \frac{b-a}{n}. \tag{1.2.5}$$

(In the figure, $n = 4$.) We shall show in a later chapter that the error is very nearly proportional to h^2 when h is small. One can then, in principle, attain arbitrary high accuracy by choosing h sufficiently small. However, the computational work involved is roughly proportional to the number of points where $y(x)$ must be computed, and thus inversely proportional to h. Thus the computational work growth rapidly as one demands higher accuracy (smaller h).

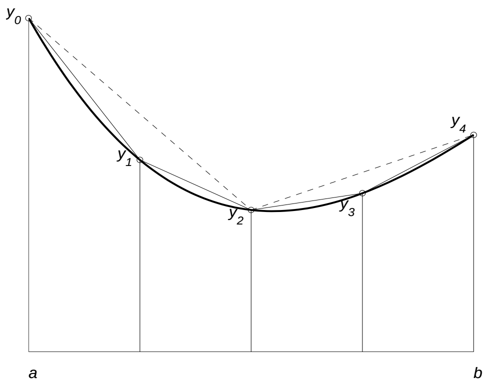


Figure 1.2.5. *Numerical integration by the trapezoidal rule ($n = 4$).*

Numerical integration is a fairly common problem because in fact it is quite seldom that the "primitive" function can be analytically calculated in a finite expression containing only elementary functions. It is not possible, for example, for such simple functions as e^{x^2} or $(\sin x)/x$. In order to obtain *higher accuracy* with significant less work than the trapezoidal rule requires, one can use one of the following two important ideas:

(a) **Local approximation** of the integrand with a polynomial of higher degree, or with a function of some other class, for which one knows the primitive function.

(b) Computation with the trapezoidal rule for several values of h and then extrapolation to $h = 0$, so-called **Richardson extrapolation** or **the deferred approach to the limit**, with the use of general results concerning the dependence of the error on h.

The technical details for the various ways of approximating a function with a polynomial, among others Taylor expansions, interpolation, and the method of least squares, are treated in later chapters.

The extrapolation to the limit can easily be applied to numerical integration with the trapezoidal rule. As was mentioned previously, the trapezoidal approximation (1.2.5) to the integral has an error approximately proportional to the square of the step size. Thus, using two step sizes, h and $2h$, one has:

$$T(h) - I \approx kh^2, \qquad T(2h) - I \approx k(2h)^2,$$

and hence $4(T(h) - I) \approx T(2h) - I$, from which it follows that

$$I \approx \tfrac{1}{3}(4T(h) - T(2h)) = T(h) + \tfrac{1}{3}(T(h) - T(2h)).$$

Thus, by adding the corrective term $\frac{1}{3}(T(h) - T(2h))$ to $T(h)$, one should get an estimate of I which typically is far more accurate than $T(h)$. In Sec. 3.6 we shall see that the improvements is in most cases quite striking. The result of the Richardson extrapolation is in this case equivalent to the classical **Simpson's rule** for numerical integration, which we shall encounter many times in this volume. It can be derived in several different ways. Sec. 3.6 also contains a further development of the extrapolation idea, **Romberg's method**.

Knowledge of the behavior of the error can, together with the idea of extrapolation, lead to a powerful method for improving results. Such a line of reasoning is useful not only for the common problem of numerical integration, but also in many other types of problems.

Example 1.2.3. The integral $\int_{10}^{12} f(x)\,dx$ is computed for $f(x) = x^3$ by the trapezoidal method. With $h = 1$ we obtain

$$T(h) = 2,695, \qquad T(2h) = 2,728,$$

and extrapolation gives $T = 2.684$, equal to the exact result. Similarly, for $f(x) = x^4$ we obtain

$$T(h) = 30,009, \qquad T(2h) = 30,736,$$

and with extrapolation $T = 29,766.7$ (exact $29,766.4$).

1.2.3 Finite Difference Approximations

The local approximation of a complicated function by a linear function leads to another frequently encountered idea in the construction of numerical methods, namely the approximation of a derivative by a difference quotient. Fig. 1.2.6 shows the graph of a function $y(x)$ in the interval $[x_{n-1}, x_{n+1}]$ where $x_{n+1} - x_n = x_n - x_{n-1} = h$; h is called the step size. If we set $y_i = y(x_i)$, $i = n-1, n, n+1$, then the derivative at x_n can be approximated by a **forward difference quotient**,

$$y'(x_n) \approx \frac{y_{n+1} - y_n}{h}, \tag{1.2.6}$$

or a similar backward difference quotient involving y_n and y_{n-1}. The error in the approximation is called a **discretization error**.

However, it is conceivable that the **centered difference approximation**

$$y'(x_n) \approx \frac{y_{n+1} - y_{n-1}}{2h} \tag{1.2.7}$$

will usually be more accurate. It is in fact easy to motivate this. By Taylor's formula,

$$y(x+h) - y(x) = y'(x)h + y''(x)h^2/2 + y'''(x)h^3/6 + \dots \tag{1.2.8}$$

$$-y(x-h) + y(x) = y'(x)h - y''(x)h^2/2 + y'''(x)h^3/6 - \dots \tag{1.2.9}$$

Set $x = x_n$. Then, by the first of these equations,

$$y'(x_n) = \frac{y_{n+1} - y_n}{h} + \frac{h}{2}y''(x_n) + \dots$$

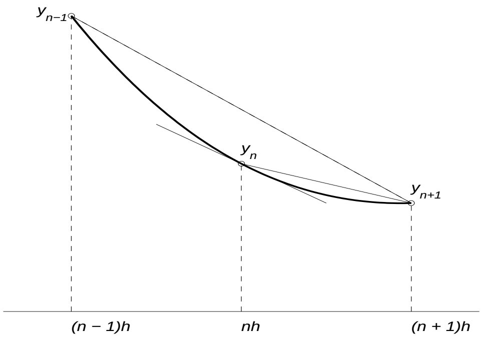


Figure 1.2.6. *Finite difference quotients.*

Next, add the two Taylor expansions and divide by $2h$. Then the first error term cancels and we have

$$y'(x_n) = \frac{y_{n+1} - y_{n-1}}{2h} + \frac{h^2}{6}y'''(x_n) + \dots$$

We shall in the sequel call a formula (or a method), where a step size parameter h is involved, **accurate of order** p, if its error is approximately proportional to h^p. Since $y''(x)$ vanishes for all x if and only if y is a linear function of x, and similarly, $y'''(x)$ vanishes for all x if and only if y is a quadratic function, we have established the following important result:

Lemma 1.2.1. *The forward difference approximation* (1.2.6) *is exact only for a linear function, and it is only first order accurate in the general case. The centered difference approximation* (1.2.7) *is exact also for a quadratic function, and is second order accurate in the general case.*

For the above reason the approximation (1.2.7) is, in most situations, preferable to (1.2.6). However, there are situations when these formulas are applied to the approximate solution of differential equations where the forward difference approximation suffices, but where the centered difference quotient is entirely unusable, for reasons which have to do with how errors are propagated to later stages in the calculation. We shall not discuss it more closely here, but mention it only to intimate some of the surprising and fascinating mathematical questions which can arise in the study of numerical methods.

Higher derivatives are approximated with **higher differences**, that is, differences of differences, another central concept in numerical calculations. We define:

$$\begin{aligned}(\Delta y)_n &= y_{n+1} - y_n;\\ (\Delta^2 y)_n = (\Delta(\Delta y))_n &= (y_{n+2} - y_{n+1}) - (y_{n+1} - y_n)\\ &= y_{n+2} - 2y_{n+1} + y_n;\\ (\Delta^3 y)_n = (\Delta(\Delta^2 y))_n &= y_{n+3} - 3y_{n+2} + 3y_{n+1} - y_n;\end{aligned}$$

etc. For simplicity one often omits the parentheses and writes, for example, $\Delta^2 y_5$ instead of $(\Delta^2 y)_5$. The coefficients that appear here in the expressions for the higher differences are, by the way, the binomial coefficients. In addition, if we denote the step length by Δx instead of by h, we get the following formulas, which are easily remembered:

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x}, \qquad \frac{d^2y}{dx^2} \approx \frac{\Delta^2 y}{(\Delta x)^2}, \tag{1.2.10}$$

etc. Each of these approximations is second order accurate for the value of the derivative at an x which equals the *mean value* of the largest and smallest x for which the corresponding value of y is used in the computation of the difference. (The formulas are only first order accurate when regarded as approximations to derivatives at other points between these bounds.) These statements can be established by arguments similar to the motivation for the formulas (1.2.6) and (1.2.7). The following second order accurate formula, which can be derived by taking the *sum* of the two Taylor expansions in (1.2.8)–(1.2.9), is an important particular case:

$$y''(x_n) = \frac{(\Delta^2 y)_{n-1}}{h^2} = \frac{(\delta^2 y)_n}{h^2} \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2}, \tag{1.2.11}$$

where δ denotes the **central difference operator**

$$\delta y_n = \frac{1}{h}\left(y\left(x_n + \tfrac{1}{2}h\right) - y\left(x_n + \tfrac{1}{2}h\right)\right).$$

The approximation of equation (1.2.7) can be interpreted as an application of (1.2.10) with $\Delta x = 2h$, or else as the mean of the estimates which one gets according to equation (1.2.10) for $y'((n + \frac{1}{2})h)$ and $y'((n - \frac{1}{2})h)$.

When the values of the function have errors, for example, when they are rounded numbers, the difference quotients become more and more uncertain the less h is. Thus if one wishes to compute the derivatives of a function given by a table, one should as a rule use a step length which is greater than the table step.

Example 1.2.4. For $y = \cos x$ one has, using function values correct to six decimal digits:

x	y	Δy	$\Delta^2 y$
0.59	0.830941		
		-5605	
0.60	0.825336		-83
		-5688	
0.61	0.819648		

This arrangement of the numbers is called a **difference scheme**. Note that the differences are expressed in units of 10^{-6}. Using (1.2.7) and (1.2.10) one gets

$$y'(0.60) \approx (0.819648 - 0.830941)/0.02 = -0.56465,$$
$$y''(0.60) \approx -83 \cdot 10^{-6}/(0.01)^2 = -0.83.$$

The correct results are, with six decimals, $y'(0.60) = -0.564642, y''(0.60) = -0.825336$. In y'' we only got two correct decimal digits. This is due to **cancellation**, which is an important cause of loss of accuracy, see further Sec. 2.2.3. Better accuracy can be achieved by *increasing* the step h, see Problem 5 at the end of this section.

Finite difference approximations are useful for partial derivatives too. Suppose that the values $u_{i,j} = u(x_i, y_j)$ of a function $u(x, y)$ are given on a square grid with grid size h, i.e. $x_i = x_0 + ih$, $y_j = y_0 + jh$, $0 \le i \le M$, $0 \le j \le N$ that covers a rectangle. By (1.2.11), the **Laplace operator**

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

can then be approximated by

$$\begin{aligned}&\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}\\ &= \frac{1}{h^2}\big(u_{i,j+1} + u_{i-1,j} + u_{i+1,j} + u_{i,j-1} - 4u_{i,j}\big).\end{aligned}$$

see the "computational molecule"

$$\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \tag{1.2.12}$$

where $f(x, y)$ is a given function.

Review Questions

1. Make lists of the concepts and ideas which have been introduced. Review their use in the various types of problems mentioned.
2. Discuss the convergence condition and the rate of convergence of the method of iteration for solving $x = F(x)$.
3. What is the trapezoidal rule? What is said about the dependence of its error on the step length?

Problems and Computer Exercises

1. Calculate $\sqrt{10}$ to seven decimal places using the method in Example 1.2.1. Begin with $x_0 = 2$.
2. Consider $f(x) = x^3 - 2x - 5$. The cubic equation $f(x) = 0$ has been a standard test problem, since Newton used it in 1669 to demonstrate his method. By computing (say) $f(x)$ for $x = 1, 2, 3$, we see that $x = 2$ probably is a rather good initial guess. Iterate then by Newton's method until you trust that the result is correct to six decimal places.

3. The equation $x^3 - x = 0$ has three roots, $-1, 0, 1$. We shall study the behaviour of Newton's method on this equation, with the notations used in §1.2.2 and Fig. 1.2.3.

(a) What happens if $x_0 = 1/\sqrt{3}$? Show that x_n converges to 1 for any $x_0 > 1/\sqrt{3}$. What is the analogous result for convergence to -1?

(b) What happens if $x_0 = 1/\sqrt{5}$? Show that x_n converges to 0 for any $x_0 \in (-1/\sqrt{5},\ 1/\sqrt{5})$.

Hint: Show first that if $x_0 \in (0,\ 1/\sqrt{5})$ then $x_1 \in (-x_0, 0)$. What can then be said about x_2?

(c) Find, by a drawing (with paper and pencil), $\lim x_n$ if x_0 is a little less than $1/\sqrt{3}$. Find by computation $\lim x_n$ if $x_0 = 0.46$.

*(d) A complete discussion of the question in (c) is rather complicated, but there is an implicit recurrence relation that produces a decreasing sequence $\{a_1 = 1/\sqrt{3}, a_2, a_3, \ldots\}$, by means of which you can easily find $\lim_{n\to\infty} x_n$ for any $x_0 \in (1/\sqrt{5},\ 1/\sqrt{3})$. Try to find this recurrence.

Answer: $a_i - f(a_i)/f'(a_i) = -a_{i-1}$; $\lim_{n\to\infty} x_n = (-1)^i$ if $x_0 \in (a_i, a_{i+1})$;
$a_1 = 0.577$, $a_2 = 0.462$, $a_3 = 0.450$, $a_4 \approx \lim_{i\to\infty} a_i = 1/\sqrt{5} = 0.447$.

4. Calculate $\int_0^{1/2} e^x\,dx$

(a) to six decimals using the primitive function.

(b) with the trapezoidal rule, using step length $h = 1/4$.

(c) using Richardson extrapolation to $h = 0$ on the results using step length $h = 1/2$, and $h = 1/4$.

(d) Compute the ratio between the error in the result in (c) to that of (b).

5. In Example 1.2.4 we computed $y''(0.6)$ for $y = \cos(x)$, with step length $h = 0.01$. Make similar calculations using $h = 0.1$, $h = 0.05$ and $h = 0.001$. Which value of h gives the best result, using values of y to six decimal places? Discuss qualitatively the influences
of both the rounding errors in the table values and the error in the approximation of a derivative with a difference quotient on the result for various values of h.

1.3 Numerical Algorithms

For a given numerical problem one can consider many different algorithms. These can differ in efficiency and reliability and give approximate answers sometimes with widely varying accuracy. In the following we give a few examples of how algorithms can be developed to solve some typical numerical problems.

1.3.1 Recurrence Relations

One of the most important and interesting parts of the preparation of a problem for a computer is to *find a recursive description of the task.* Sometimes an enormous amount of computation can be described by a small set of recurrence relations.

A common computational task is the evaluation of a polynomial, at a given point z where, say,

$$\begin{aligned} p(z) &= a_0 z^3 + a_1 z^2 + a_2 z + a_3 \\ &= ((a_0 z + a_1)z + a_2)z + a_3. \end{aligned}$$

We set $b_0 = a_0$, and compute

$$b_1 = b_0 z + a_1, \quad b_2 = b_1 z + a_2, \quad p(z) = b_3 = b_2 z + a_3.$$

This illustrates, for $n = 3$, **Horner's rule** for evaluating a polynomial of degree n,

$$p(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n,$$

at a point z. This algorithm can be described by the recurrence relation:

$$b_0 = a_0, \quad b_i = b_{i-1} z + a_i, \quad i = 1 : n, \tag{1.3.1}$$

where $p(z) = b_n$. Clearly this algorithm requires n additions and multiplications for evaluating $p(z)$. An algorithm where the powers are calculated recursively by $x^i = z \cdot z^{i-1}$ and subsequently multiplied by a_{n-i} requires twice as many multiplications.

The quantities b_i in (1.3.1) are of intrinsic interest because of the following result, often called **synthetic division**:

$$\frac{p(x) - p(z)}{x - z} = \sum_{i=0}^{n-1} b_i x^{n-1-i}, \tag{1.3.2}$$

where the b_i are defined by (1.3.1). The proof of this result is left as an exercise. Synthetic division is used, for instance, in the solution of algebraic equations, when already computed roots are successively eliminated. After each elimination, one can deal with an equation of lower degree. This process is called **deflation**. (As shown in Sec. 6.6.4, some care is necessary in the numerical application of this idea.)

The proof of the following useful relation is left as an exercise to the reader:

Lemma 1.3.1.

Let the b_i be defined by (1.3.1) and

$$c_0 = b_0, \qquad c_i = b_i + z c_{i-1}, \quad i = 1 : n - 1. \tag{1.3.3}$$

Then $p'(z) = c_{n-1}$.

Recurrence relations are among the most valuable aids in numerical calculation. Very extensive calculations can specified in relatively short computer programs with the help of such formulas. However, unless used in the right way errors can grow exponentially and completely ruin the results.

Example 1.3.1.

To compute the integrals $I_n = \displaystyle\int_0^1 \frac{x^n}{x+5}\,dx$, $i = 1 : N$ one can use the recurrence relation

$$I_n + 5 I_{n-1} = 1/n, \tag{1.3.4}$$

which follows from

$$I_n + 5 I_{n-1} = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5}\,dx = \int_0^1 x^{n-1}\,dx = \frac{1}{n}.$$

Below we use this formula to compute I_8, using six decimals throughout. For $n = 0$ we have

$$I_0 = [\ln(x+5)]_0^1 = \ln 6 - \ln 5 = 0.182322.$$

Using the recurrence relation we get

$$\begin{aligned}
I_1 &= 1 - 5I_0 = 1 - 0.911610 = 0.088390,\\
I_2 &= 1/2 - 5I_1 = 0.500000 - 0.441950 = 0.058050,\\
I_3 &= 1/3 - 5I_2 = 0.333333 - 0.290250 = 0.043083,\\
I_4 &= 1/4 - 5I_3 = 0.250000 - 0.215415 = 0.034585,\\
I_5 &= 1/5 - 5I_4 = 0.200000 - 0.172925 = 0.027075,\\
I_6 &= 1/6 - 5I_5 = 0.166667 - 0.135375 = 0.031292,\\
I_7 &= 1/7 - 5I_5 = 0.142857 - 0.156460 = -0.013603.
\end{aligned}$$

It is strange that $I_6 > I_5$, and obviously absurd that $I_7 < 0$! The reason for the absurd result is that the round-off error ϵ in $I_0 = 0.18232156\ldots$, whose magnitude is about $0.44 \cdot 10^{-6}$ is *multiplied* by (-5) in the calculation of I_1, which then has an error of -5ϵ. That error produces an error in I_2 of $5^2\epsilon$, etc. Thus the magnitude of the error in I_7 is $5^7\epsilon = 0.0391$, which is larger than the true value of I_7. On top of this comes the round-off errors committed in the various steps of the calculation. These can be shown in this case to be relatively unimportant.

If one uses higher precision, the absurd result will show up at a later stage. For example, a computer that works with a precision corresponding to about 16 decimal places, gave a negative value to I_{22} although I_0 had full accuracy. The above algorithm is an example of a disagreeable phenomenon, called **numerical instability**.

We now show how, in this case, one can avoid numerical instability by choosing a more suitable algorithm.

Example 1.3.2.

We shall here use the recurrence relation in the other direction,

$$I_{n-1} = (1/n - I_n)/5. \tag{1.3.5}$$

Now the errors will be *divided* by -5 in each step. But we need a starting value. We can directly see from the definition that I_n decreases as n increases. One can also surmise that I_n decreases slowly when n is large (the reader is recommended to motivate this). Thus we try setting $I_{12} = I_{11}$. It then follows that

$$I_{11} + 5I_{11} \approx 1/12, \qquad I_{11} \approx 1/72 \approx 0.013889.$$

(show that $0 < I_{12} < 1/72 < I_{11}$). Using the recurrence relation we get

$$I_{10} = (1/11 - 0.013889)/5 = 0.015404, \qquad I_9 = (1/10 - 0.015404)/5 = 0.016919,$$

and further

$$\begin{array}{llll} I_8 = 0.018838, & I_7 = 0.021232, & I_6 = 0.024325, & I_5 = 0.028468, \\ I_4 = 0.034306, & I_3 = 0.043139, & I_2 = 0.058039, & I_1 = 0.088392, \end{array}$$

and finally $I_0 = 0.182322$. Correct!

If we instead simply take as starting value $I_{12} = 0$, one gets $I_{11} = 0.016667$, $I_{10} = 0.018889$, $I_9 = 0,016222$, $I_8 = 0.018978$, $I_7 = 0.021204$, $I_6 = 0.024331$, and $I_5, \dots, I_0$ have the same values as above. The difference in the values for I_{11} is 0.002778. The subsequent values of $I_{10}, I_9, \dots, I_0$ are quite close because the error is divided by -5 in each step. The results for I_n obtained above have errors which are less than 10^{-3} for $n \le 8$.

The reader is warned, however, not to draw erroneous conclusions from the above example. The use of a recurrence relation "backwards" is not a universal recipe as will seen later on! Compare also Problems 6 and 7 at the end of this section.

1.3.2 Divide and Conquer Strategy

A powerful strategy for solving large scale problems is the **divide and conquer** strategy. The idea is to split a high dimensional problem into problems of lower dimension. Each of these are then again split into smaller subproblems, etc., until a number of sufficiently small problems are obtained. The solution of the initial problem is then obtained by combining the solution of the subproblems working backwards in the hierarchy.

We illustrate the idea on the computation of the sum $s = \sum_{i=1}^{n} a_i$. The usual way to proceed is to to use the recursion

$$s_0 = 0, \qquad s_i = s_{i-1} + a_i, \quad i = 1 : n.$$

Another order of summation is as illustrated below for $n = 2^3 = 8$:

$$\begin{array}{cccccccccccccccc}
a_1 & & a_2 & & a_3 & & a_4 & & a_5 & & a_6 & & a_7 & & a_8 \\
& s_{1,2} & & & & s_{3,4} & & & & s_{5,6} & & & & s_{7,8} & \\
& & & s_{1,4} & & & & & & & & s_{5,8} & & & \\
& & & & & & & s_{1,8} & & & & & & &
\end{array}$$

where $s_{i,j} = a_i + \cdots + a_j$. In this table each new entry is obtained by adding its two neighbors in the row above. Clearly this can be generalized to compute an arbitrary sum of $n = 2^k$ terms in k steps. In the first step we perform $n/2$ sums of two terms, then $n/4$ partial sums each of 4 terms, etc., until in the kth step we compute the final sum.

This summation algorithm uses the same number of additions as the first one. However, it has the advantage that it splits the task in *several subtasks that can be* *performed in parallel.* For large values of n this summation order can also be much more accurate than the conventional order (see Problem 2.3.1).

The algorithm can also be described in another way. Consider the following definition of a summation algorithm for computing the $s(i,j) = a_i + \cdots + a_j$, $j > i$:

```
$sum = s(i,j)$;
**if** $j = i + 1$ **then** $sum = a_i + a_j$;
    **else** $k = \lfloor (i+j)/2 \rfloor$;  $sum = s(i,k) + s(k+1,j)$;
**end**
```

This function defines $s(i,j)$ in a recursive way; if the sum consists of only two terms then we add them and return with the answer. Otherwise we split the sum in two and use the function again to evaluate the corresponding two partial sums. This approach is aptly called the **divide and conquer** strategy. The function above is an example of a **recursive algorithm**—it calls itself. Many computer languages (e.g., Matlab) allow the definition of such recursive algorithms. The divide and conquer is a **top down** description of the algorithm in contrast to the **bottom up** description we gave first.

There are many other less trivial examples of the power of the divide and conquer approach. It underlies the Fast Fourier Transform and leads to efficient implementations of, for example, matrix multiplication, Cholesky factorization, and other matrix factorizations. Interest in such implementations have increased lately since it has be realized that they achieve very efficient automatic parallelization of many tasks.

1.3.3 Approximation of Functions

One fundamental problem which occurs in many variants, is to approximate a function f by a member f^* of a class of functions which is easy to work with mathematically (for example, polynomials, rational functions, or trigonometric polynomials), where each particular function in the class is specified by the numerical values of a number of parameters.

There are two types of shortcomings to take into account: errors in the input data, and shortcomings in the particular model (class of functions, form) which one intends to adopt to the input data. For ease in discussion we shall call these the **measurement error** and the **error in the model**, respectively.

Example 1.3.3. The points in Fig. 1.3.1 show, for $n = 1,2,3,4,5$, the time t for the nth passage of a swinging pendulum through its point of equilibrium. The condition of the experiment were such that a relation of the form $t = a + bn$ can be assumed to be valid to very high accuracy. Random errors in measurement are the dominant cause of the deviation from linearity shown in Figure 1.3.1. This deviation causes the values of the parameters a and b to be uncertain. We have five points and only two parameters to determine; the problem is said to be **overdetermined**. Such overdetermined problems can be treated by the method of least squares, i.e. by minimizing the sum of squares of the deviations; see Section 1.6.5.

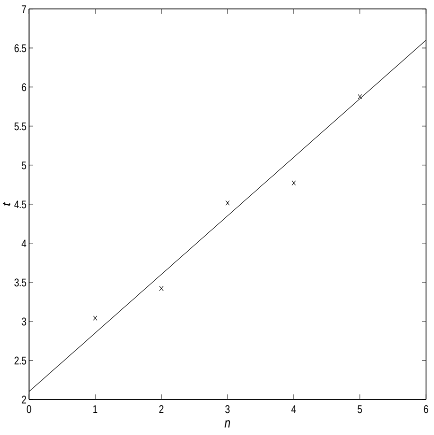


Figure 1.3.1. *Fitting a linear relation to observations.*

Example 1.3.4. The Error function $\operatorname{erf}(x)$ is defined by the integral of the Gaussian distribution function from 0 to x

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

In order to compute $\operatorname{erf}(x)$ for arbitrary $x \in [0, 1]$ with a relative error less than 10^{-8} with a small number of arithmetic operations, one can approximate the function by a polynomial. Setting $x = -t^2$ in the well known Maclaurin series for e^x, truncating after seven terms and integrating term by term we obtain the polynomial approximation of degree $2n + 1$

$$f^*(x) = \frac{2}{\sqrt{\pi}} \int_0^x \sum_{j=0}^{n} (-1)^j \frac{t^{2j}}{j!}\, dt = \frac{2}{\sqrt{\pi}} \sum_{j=0}^{n} a_j x^{2j+1}, \quad a_j = \frac{(-1)^j}{j!(2j+1)}.$$

This series converges for all x, but is suitable for numerical computations only for values of x which are not too large. To evaluate the series we note that the coefficients a_j satisfies the recursion.

$$a_j = -a_{j-1} \frac{(2j-1)}{j(2j+1)}.$$

This recursion shows that for $x \in [0, 1]$ the absolute values of the terms $t_j = a_j x^{2j+1}$ decrease monotonically. This implies that the absolute error in a partial sum is bounded by the absolute value of the first neglected term. (Why? For an answer see Theorem 3.1.5.) A possible algorithm then is:

Set $t_0 = x$, $s_0 = a_0$; for $j = 1, 2, \ldots$ compute

$$t_j = -t_{j-1} \frac{(2j-1)}{j(2j+1)} x^2, \quad s_j = s_{j-1} + t_j, \quad \text{until} \quad |t_j| \le \cdot 10^{-8} s_j.$$

Here we have estimated the error by the last term added in the series. Since we have to compute this term for the error estimate we might as well use it! Note also that in this case, where the number of terms is fixed in advance, Horner's scheme is not suitable for the evaluation.

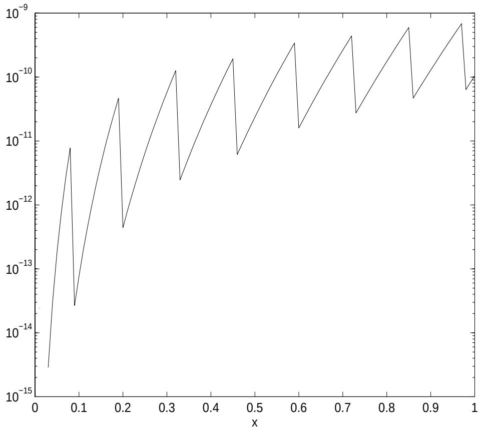


Figure 1.3.2. *Relative error* $e(x) = |f^*(x) - \mathrm{erf}(x)|/erf(x)$.

Fig. 1.3.2 shows the graph of the relative error in the computed approximation $f^*(x)$. At most twelve terms in the series were needed.

In this example there are no errors in measurement, but the "model" of approximating the error function with a polynomial is not exact, since the function demonstrably is not a polynomial. There is a **truncation error**[3] from truncating the series, which can in this case be made as small as one wants by choosing the degree of the polynomial sufficiently large (e.g., by taking more terms in the Maclaurin series).

The use of power series will be studied in depth in Chapter 3, where also other more efficient methods than the Maclaurin series for approximation by polynomials will be treated.

The above examples showed two isolated situations. In practice, both the input data and the model are as a rule insufficient. One can consider approximation as a special case of a more general and quite important problem: to fit a mathematical *model* to given data and other known facts. One can also see approximation problems as analogous to the task of a communication engineer, to filter away **noise** from the **signal**. These questions are connected with both **mathematical statistics** and the mathematical discipline **approximation theory**.

[3]In general the error due to replacing an infinite process by a finite is referred to as a truncation error.

1.3.4 Solving Linear System of Equations

A fundamental problem in scientific computing is computing the solution (if any) of a system of linear equations

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_n \end{aligned}\right\}, \tag{1.3.6}$$

where a_{ij} and b_i, $1 \le i \le m$, $1 \le j \le n$ are the known input data and x_j, $1 \le j \le n$, are the unknowns variables to be determined.

This system can be written

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \tag{1.3.7}$$

or much more compact in matrix-vector form as $Ax = b$, where $A \in \mathbf{R}^{m\times n}$ is a matrix and $x \in \mathbf{R}^n$ and $b \in \mathbf{R}^m$ are column vectors. There are three possibilities, namely, the system may have no solution, a unique solution, or an infinite set of solutions.

Linear systems which (possibly after a permutation of rows and columns of A) are of triangular form are particularly simple to solve. Consider a square **upper triangular** linear system ($m = n$)

$$\begin{pmatrix} u_{11} & \cdots & u_{1,n-1} & u_{1n} \\ & \ddots & \vdots & \vdots \\ & & u_{n-1,n-1} & u_{n-1,n} \\ & & & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}.$$

The matrix U is nonsingular if and only if

$$\det(U) = u_{11} \cdots u_{n-1,n-1} u_{nn} \neq 0.$$

If this is the case the unknowns can be computed by the following recursion

$$x_n = b_n/u_{nn}, \qquad x_i = \Big(b_i - \sum_{k=i+1}^{n} u_{ik}x_k\Big)/u_{ii}, \quad i = n-1, \ldots, 1. \tag{1.3.8}$$

Since the unknowns are solved for in *backward* order, this is called **back-substitution**. Similarly, a square linear system of **lower triangular** form $Lx = b$,

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \ldots & l_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

where L is nonsingular, can be solved by **forward-substitution**

$$x_1 = b_1/l_{11}, \qquad x_i = \Big(b_i - \sum_{k=1}^{i-1} l_{ik}x_k\Big)/l_{ii}, \quad i = 2 : n. \tag{1.3.9}$$

(Note that by reversing the order of the rows and columns an upper triangular system is transformed into a lower triangular and vice versa.)

From the formulas above it easily follows that the solution of a triangular system of order n can be computed in n divisions and $\frac{1}{2}n(n-1)$ additions and multiplications. Note that this is almost exactly the same amount of work as required for *multiplying* a vector by a triangular matrix.

It is important to know roughly how much work is required by different matrix algorithms. It should be noted, however, that counting the number of arithmetic operations is not sufficient for the estimation of running times on a computer, since subscript computation and the time to retrieve data and store the result must also be taken into account. However, a count of the number of arithmetic operations in a matrix algorithm still provides useful information, and can serve as an initial basis of comparison of different algorithms. The operation count for the solution of a triangular system tells us that the running time on a computer roughly will increase quadratically with the dimension n. Thus, doubling n will approximately increase the work in solving the system by a factor of four.

Usually in matrix computations the number of multiplicative operations $(\times, /)$ is about the same as the number of additive operations $(+, -)$. Therefore we will here use the concept of a **flop** to mean roughly the amount of work associated with the computation

$$s := s + a_{ik}b_{kj},$$

i.e., one addition and multiplication and some related subscript computation.[4]

Flop counts like these are meant only as a rough appraisal of the work and one should not assign too much meaning to their precise value. On modern computer architectures the rate of transfer of data between different levels of memory often limits the actual performance. Also ignored here is the fact that on current computers division usually is 5–10 times slower than a multiply.

If in Gaussian elimination a zero pivotal element is encountered, $a_{kk}^{(k)} = 0$, When implementing a matrix algorithm on a computer, the *order of operations* in matrix algorithms may be important. One reason for this is the economizing of storage, since even matrices of moderate dimensions have a large number of elements. When the initial data is not needed for future use, computed quantities may overwrite data. To resolve such ambiguities in the description of matrix algorithms it is important to be able to describe computations like those in equations (1.3.8) in a more precise form. For this purpose we will use an informal programming language, which is sufficiently precise for our purpose but allows the suppression of cumbersome details. We illustrate these concepts on the back-substitution algorithm given above. In the following back-substitution algorithm the solution x overwrites the data b.

[4]Not that in some textbooks (e.g., Higham [7, 2002]) a flop is instead defined as an add *or* multiply, doubling all the flop counts in this book.

Algorithm 1.3.1 Back-substitution

Given a nonsingular upper triangular matrix $U \in \mathbf{R}^{n\times n}$ and a vector $b \in \mathbf{R}^n$, the following algorithm computes $x \in \mathbf{R}^n$ such that $Ux = b$:

```
for $i = n : (-1) : 1$
    $s := \sum_{j=i+1}^{n} u_{ik} b_k;$
    $b_i := (b_i - s)/u_{ii};$
end
```

Here $x := y$ means that the value of y is evaluated and assigned to x. We use the convention that when the upper limit in a sum is smaller than the lower limit the sum is set to zero.

Another possible sequencing of the operations in Algorithm 1.3.4:

```
for $k = n : (-1) : 1$
    $b_k := b_k / u_{kk};$
    for $i = k - 1 : (-1) : 1$
        $b_i := b_i - u_{ik} b_k;$
    end
end
```

Here the elements in U are accessed column-wise instead of row-wise as in the previous algorithm. Such differences can influence the efficiency of the implementation depending on how the elements in the matrix U is stored.

Clearly the following elementary operation can be performed on the system without changing the set of solutions:

- Interchange two equations
- Multiply an equation by a nonzero scalar α.
- Adding a multiple α of the ith equation to the jth equation.

These operations correspond in an obvious way to row operations carried out on the augmented matrix (A, b). By performing a sequence of such elementary operations one can always transform the system $Ax = b$ into a simpler system, which can be solved in an elementary way. The most important direct method is **Gaussian elimination**.[5] This is still the method of choice when the matrix A is square and of full rank.

[5]Named after Carl Friedrich Gauss (1777–1855), but known already in China as early as in the first century BC.

In **Gaussian elimination** the unknowns are eliminated in a systematic way, so that at the end an equivalent triangular system is produced, which can be solved by substitution. Consider the square system

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

and assume that $a_{11} \neq 0$. Then we can eliminate x_1 from the last $(n-1)$ equations as follows. Subtracting from the ith equation the multiple

$$l_{i1} = a_{i1}/a_{11}, \quad i = 2:n,$$

of the first equation, the last $(n-1)$ equations become

$$\begin{pmatrix} a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \vdots \\ a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix},$$

where the new elements are given by

$$a_{ij}^{(2)} = a_{ij} - l_{i1}a_{1j}, \qquad b_i^{(2)} = b_i - l_{i1}b_1, \quad i = 2:n.$$

This is a system of $(n-1)$ equations in the $(n-1)$ unknowns $x_2, \dots, x_n$. If $a_{22}^{(2)} \neq 0$, we can proceed and in the next step eliminate x_2 from the last $(n-2)$ of these equations. This gives a system of equations containing only the unknowns $x_3, \dots, x_n$. We take

$$l_{i2} = a_{i2}^{(2)}/a_{22}^{(2)}, \quad i = 3:n,$$

and the elements of the new system are given by

$$a_{ij}^{(3)} = a_{ij}^{(2)} - l_{i2}a_{2j}^{(2)}, \qquad b_i^{(3)} = b_i^{(2)} - l_{i2}b_2^{(2)}, \quad i = 3:n.$$

The diagonal elements $a_{11}, a_{22}^{(2)}, a_{33}^{(3)}, \dots$, which appear during the elimination are called **pivotal elements**. As long as these are nonzero, the elimination can be continued. After $(n-1)$ steps we get the single equation

$$a_{nn}^{(n)}x_n = b_n^{(n)}.$$

Collecting the first equation from each step we get

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}, \tag{1.3.10}$$

where we have introduced the notations $a_{ij}^{(1)} = a_{ij}$, $b_i^{(1)} = b_i$ for the coefficients in the original system. Thus, we have reduced (1.3.7) to an equivalent nonsingular, upper triangular system (1.3.10), which can be solved by back-substitution.

In passing we remark that since the determinant of a matrix does not change under row operations we have from (1.3.10) $\det(A) = a_{11}^{(1)}a_{22}^{(2)}\cdots a_{nn}^{(n)}$ *Gaussian elimination is indeed in general the most efficient method for* **computing determinants**!

Algorithm 1.3.2 Gaussian Elimination (without row interchanges)

Given a matrix $A = A^{(1)} \in \mathbf{R}^{n\times n}$ and a vector $b = b^{(1)} \in \mathbf{R}^n$, the following algorithm computes the elements of the reduced system of upper triangular form (1.3.10). It is assumed that $a_{kk}^{(k)} \neq 0$, $k = 1 : n$:

```
for $k = 1 : n-1$
    for $i = k+1 : n$
        $l_{ik} := a_{ik}^{(k)}/a_{kk}^{(k)};\quad a_{ik}^{(k+1)} := 0;$
        for $j = k+1 : n$
            $a_{ij}^{(k+1)} := a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)};$
        end
        $b_i^{(k+1)} := b_i^{(k)} - l_{ik}b_k^{(k)};$
    end
end
```

We remark that no extra memory space is needed to store the multipliers. When $l_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$ is computed the element $a_{ik}^{(k+1)}$ becomes equal to zero, so the multipliers can be stored in the lower triangular part of the matrix. Note also that if the multipliers l_{ik} are saved, then the operations on the right hand side b can be carried out at a later stage. This observation is important in that it shows that *when solving a sequence of linear systems*

$$Ax_i = b_i, \quad i = 1 : p,$$

with the same matrix A but different right hand sides the operations on A only have to be carried out once.

If we form the matrices

$$L = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix} \tag{1.3.11}$$

then it can be shown that we have $A = LU$. Hence Gaussian elimination provides *a factorization of the matrix A into a lower triangular matrix L and an upper* *triangular matrix U*. This interpretation of Gaussian elimination has turned out to be extremely fruitful. For example, it immediately follows that the inverse of A (if it exists) has the factorization

$$A^{-1} = (LU)^{-1} = U^{-1}L^{-1}.$$

This shows that the solution of linear system $Ax = b$,

$$x = A^{-1}b = U^{-1}(L^{-1}b),$$

can be computed by solving the two triangular systems $Ly = b$, $Ux = y$. Indeed "*almost anything you can do with A^{-1} can be done without it*" (G. E. Forsythe and Moler)! Many other important matrix factorizations will be studied at length in Volume 2.

The Gaussian elimination algorithm consists of three nested loops, which can be ordered in $3 \cdot 2 \cdot 1 = 6$ ways. Disregarding the right hand side b, each version does the operations

$$a_{ij}^{(k+1)} := a_{ij}^{(k)} - a_{kj}^{(k)} a_{ik}^{(k)} / a_{kk}^{(k)},$$

and only the ordering in which they are done differs. The version given above uses row operations and may be called the "*kij*" variant, where k refers to step number, i to row index, and j to column index. This version is not suitable for Fortran 77, and other languages in which matrix elements are stored sequentially by columns. In such a language the form "*kji*" should be preferred, which is the column oriented backsubsitution rather than Algorithm 1.3.4 might be preferred.

From Algorithm 1.3.4 it follows that $(n-k)$ divisions and $(n-k)^2$ multiplications and additions are used in step k to transform the elements of A. A further $(n-k)$ multiplications and additions are used to transform the elements of b. Summing over k and neglecting low order terms we find that the total number of flops required for the reduction of $Ax = b$ to a triangular system by Gaussian elimination is

$$\textstyle\sum_{k=1}^{n-1}(n-k)^2 \approx n^3/3, \qquad \sum_{k=1}^{n-1}(n-k) \approx n^2/2$$

for A and each right hand side b, respectively. Comparing this with the approximately $\frac{1}{2}n^2$ flops needed to solve a triangular system we conclude that, except for very small values of n, *the reduction of A to triangular form dominates the work.*

If in Gaussian elimination a zero pivotal element is encountered, i.e. $a_{kk}^{(k)}$ for some k, then one cannot proceed and the method breaks down. A simple example is obtained by taking $\epsilon = 0$ in the system

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

However, this system is nonsingular for any $\epsilon \neq 1$ and has a unique solution $x_1 = -x_2 = -1/(1-\epsilon)$. However, when $a_{11} = \epsilon = 0$ the first step in Gaussian elimination cannot be carried through. The remedy here is obviously to interchange the two equations, which directly gives an upper triangular system.

To ensure the numerical stability in Gaussian elimination it will, except for special classes of linear systems, be necessary to perform row interchanges *not only* *when a pivotal element is exactly zero, but also when it is small.* Suppose that in the system above $\epsilon = 10^{-4}$. Then the exact solution, rounded to four decimals equals $x = (-1.0001, 1.0001)^T$. However, if Gaussian elimination is carried through without interchanges we obtain $l_{21} = 10^4$ and the triangular system

$$\begin{aligned} 0.0001x_1 + x_2 &= 1 \\ (1 - 10^4)x_2 &= -10^4. \end{aligned}$$

Suppose that the computation is performed using arithmetic with three decimal digits. Then in the last equation the coefficient $a_{22}^{(2)}$ will be rounded to 10^4 and the solution computed by back-substitution is

$$\bar{x}_2 = 1.000, \qquad \bar{x}_1 = 0,$$

which is a catastrophic result! If we instead interchange the two equations before performing Gaussian elimination then we get $l_{21} = 10^{-4}$ and the reduced system is

$$\begin{aligned} x_1 + x_2 &= 0 \\ (1 - 10^{-4})x_2 &= 1. \end{aligned}$$

The coefficient $a_{22}^{(2)}$ is now rounded to 1, and the computed solution becomes

$$\bar{x}_2 = 1.000, \qquad \bar{x}_1 = -1.000,$$

which is correct to the precision carried.

In this simple example it is easy to see what went wrong in the elimination without interchanges. The problem is that *the choice of a small pivotal element gives rise to large elements in the reduced matrix* and the coefficient a_{22} in the original system is lost through rounding. Rounding errors which are small when compared to the large elements in the reduced matrix are unacceptable in terms of the original elements! When the equations are interchanged the multiplier is small and the elements of the reduced matrix of the same size as in the original matrix.

Now, consider the general case when in step k we have $a_{kk}^{(k)} = 0$. (The equations may have been reordered in previous steps, but we assume that the notations have been changed accordingly.) If A is nonsingular, then in particular its first k columns are linearly independent. This must also be true for the first k columns of the reduced matrix and hence some element $a_{ik}^{(k)}$, $i = k : n$ must be nonzero, say $a_{rk}^{(k)} \neq 0$. By interchanging rows k and r this element can be taken as pivot and it is possible to proceed with the elimination. The important conclusion is that *any nonsingular system of equations can be reduced to triangular form by Gaussian elimination, if appropriate row interchanges are used.*

Note that when rows are interchanged in A the same interchanges must be made in the elements of the right hand side, b. Note also that the determinant formula (**??**) must be modified to

$$\det(A) = (-1)^s a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}, \tag{1.3.12}$$

where s denotes the total number of row and columns interchanges performed.

It will be shown in Sec. 7.5 that numerical stability can be ensured in Gaussian elimination by choosing the pivotal element in step k by **partial pivoting**, i.e., by taking as pivot the largest element in magnitude in the unreduced part of the kth column.

We now consider the general case when $A \in \mathbf{R}^{m\times n}$ and $\operatorname{rank}(A) \le \min\{m,n\}$. Then it is possible that at some step k we have $a_{ik}^{(k)} = 0$, $i = k : m$. If the entire submatrix $a_{ij}^{(k)}$, $i,j > k$, is zero, then $\operatorname{rank}(A) = k$ and we stop. Otherwise there is a nonzero element, say $a_{pq}^{(k)} \ne 0$, which can be brought into pivoting position by interchanging rows k and p and columns k and q. (Note that when columns are interchanged in A the same interchanges must be made in the elements of the solution vector x.)

Proceeding in this way any matrix A and right hand side b can always be reduced by Gaussian elimination to

$$A^{(r)} = \left(\begin{array}{ccc|ccc} a_{11}^{(1)} & \cdots & a_{1r}^{(1)} & a_{1,r+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & \ddots & \vdots & & \vdots & \vdots \\ \vdots & & a_{rr}^{(r)} & a_{r,r+1}^{(r)} & \cdots & a_{rn}^{(r)} \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right), \qquad b^{(r)} = \left(\begin{array}{c} b_1^{(1)} \\ \vdots \\ b_r^{(r)} \\ \hline b_{r+1}^{(r+1)} \\ \vdots \\ b_m^{(r+1)} \end{array}\right). \tag{1.3.13}$$

where $A^{(r)}$ has upper **trapezoidal form**, and $r = \operatorname{rank}(A)$. In the reduced form (1.3.13) the two rectangular zero blocks have dimensions $(m-r)\times r$ and $(m-r)\times(n-r)$, respectively. We can now deduce the following:

1. The system $Ax = b$ has a unique solution if and only if $r = m = n$.

2. If $b_k^{(r+1)} = 0$, $k = r+1 : m$, then the system $Ax = b$ is consistent and has an infinite number of solutions. We can assign arbitrary values to the last $n - r$ components of (the possibly permuted) solution vector x. The first r components are then uniquely determined and obtained using back-substitution with the nonsingular triangular matrix in the upper left corner.

3. If $b_k^{(r+1)} \ne 0$, for some $k > r$, the the system $Ax = b$ is inconsistent and has no solution. Then we have to be content with finding x such that the residual vector $r = b - Ax$ is small in some sense.

Performing Gaussian elimination in practice is sometimes complicated by the fact that it may be difficult to decide what rank to assign to A just by inspecting the elements in the computed upper triangular matrix. To solve ill-conditioned and rank deficient linear systems reliably requires other tools that will be treated in Chapter 8.

Disregarding rounding errors Gaussian elimination gives the exact solution of a linear system after a finite number of arithmetic operations. **Iterative methods** instead compute a sequence of approximate solutions, which in the limit converges to the exact solution x. Basic iterative methods work directly with the original matrix A and only need extra storage for a few vectors.

In a classical iterative method due to L. F. Richardson [1910], a sequence of approximate solutions $x^{(k)}$ is defined by $x^{(0)=0}$,

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots, \tag{1.3.14}$$

where $\omega > 0$ is a parameters to be chosen. It follows easily from (1.3.14) that the error in $x^{(k)}$ satisfies $x^{(k+1)} - x = (I - \omega A)(x^{(k)} - x)$, and hence

$$x^{(k)} - x = (I - \omega A)^k (x^{(0)} - x).$$

The convergence of Richardson's method will be studied in Section c10.1.4.

Iterative methods are used most often for the solution of very large linear systems, which typically arise in the solution of boundary value problems of partial differential equations by finite difference or finite element methods. The matrices involved can be huge, sometimes involving several million unknowns. The LU factors of matrices arising in such applications typically contain order of magnitudes more nonzero elements than A itself. Hence, because of the storage and number of arithmetic operations required, Gaussian elimination may be far too costly to use.

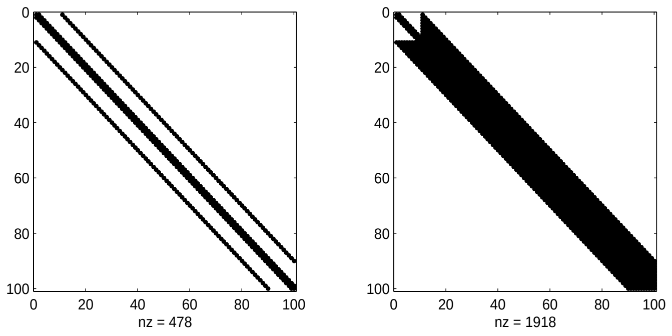


Figure 1.3.3. *Structure of A (left) and $L + U$ (right) for the Poisson problem, $N = 10$. (Rowwise ordering of the unknowns)*

Example 1.3.5.

In a typical problem for Poisson's equation (1.2.12) the function is to be determined in a plane domain D, when the values of u are given on the boundary ∂D. Such **boundary value problems** occur in the study of steady states in most branches of Physics, such as electricity, elasticity, heat flow, fluid mechanics (including meteorology). Let D be the a square grid with grid size h, i.e. $x_i = x_0 + ih$, $y_j = y_0 + jh$, $0 \le i \le N + 1$, $0 \le j \le N + 1$. Then the difference approximation yields

$$u_{i,j+1} + u_{i-1,j} + u_{i+1,j} + u_{i,j-1} - 4u_{i,j} = h^2 f(x_i, y_j),$$

$(1 \le i \le M,\ 1 \le j \le N)$. This is a huge system of linear algebraic equations; one equation for each interior gridpoint, altogether N^2 unknown and equations. (Note that $u_{i,0}$, $u_{i,N+1}$, $u_{0,j}$, $u_{N+1,j}$ are known boundary values.) To write the equations in matrix-vector form we order the unknowns in a vector

$$u = (u_{1,1}, \dots, u_{1,N}, u_{2,1}, \dots, u_{2,N-1}, u_{N,1}, \dots, u_{N,N}).$$

If the equations are ordered in the same order we get a system $Au = b$ where A is symmetric with all nonzero elements located in five diagonals; see Figure 1.3.3..

In principle Gaussian elimination can be used to solve such systems. However, even taking symmetry and the banded structure into account this would require $\frac{1}{2}N^4$ multiplications, since in the LU factors the zero elements inside the outer diagonals will fill-in during the elimination as shown in Figure 1.3.3.

The linear system arising from Poisson's equation has several features common to boundary value problems for all linear partial differential equations One of these is that there are at most 5 nonzero elements in each row of A, i.e. only a tiny fraction of the elements are nonzero. Such matrices are called **sparse**. Therefore One iteration in Richardson's method requires only about $5! \cdot N^2$ multiplications or equivalently five multiplications per unknown. Using iterative methods which take advantage of the sparsity and other features does allow the efficient solution of such systems. This becomes even more essential for three-dimensional problems!

Review Questions

1. Describe Horner's rule and synthetic division.

2. Give a concise explanation why the algorithm in Example 1.3.1 did not work and why that in Example 1.3.2 did work.

3. How many operations are needed (approximately) for

(a) The LU factorization of a square matrix?

(b) The solution of $Ax = b$, when the triangular factorization of A is known?

4. Show that if the kth diagonal entry of an upper triangular matrix is zero, then its first k columns are linearly dependent.

5. What is the LU-decomposition of an n by n matrix A, and how is it related to Gaussian elimination? Does it always exist? If not, give sufficient conditions for its existence.

Problems and Computer Exercises

1. (a) Use Horner's scheme to compute $p(2)$ where

$$p(x) = 2 - 3x^2 + 2x^3 + x^4.$$

(b) Count the number of multiplications and additions required for the evaluation of a polynomial $p(z)$ of degree n by Horner's rule. Compare with the work needed when the powers are calculated recursively by $x^i = x \cdot x^{i-1}$ and subsequently multiplied by a_{n-i}.

2. Show how repeated synthetic division can be used to move the origin of a polynomial, i.e., given $a_1,\ a_2,\dots,a_n$ and z, find $c_1,\ c_2,\dots,c_n$ so that $\sum_{j=1}^n a_j x^{j-1} \equiv \sum_{j=1}^n c_j (x-z)^{j-1}$. Write a program for synthetic division (with this ordering of the coefficients), and apply it to this algorithm.

Hint: Set $p_n(x) = \sum_{j=1}^n a_j x^{j-1}$. Apply synthetic division to $p_n(x)$, $p_{n-1}(x) = (p_n(x) - p_n(z))/(x-z)$, etc..

3. (a) Show that the transformation made in Problem 2 can also be expressed by means of the matrix-vector equation,

$$c = \operatorname{diag}(z^{1-i}) P \operatorname{diag}(z^{j-1}) a,$$

where $a = [a_1, a_2, \dots a_n]^T$, $c = [c_1, c_2, \dots c_n]^T$, and $\operatorname{diag}(z^{j-1})$ is a diagonal matrix with the elements c^{j-1}, $j = 1 : n$. The matrix P is of size $[n,n]$; its elements are $p_{i,j} = \binom{j-1}{i-1}$, if $j \ge i$, else $p_{i,j} = 0$. By convention, $\binom{0}{0} = 1$ here.

(b) Note the relation of P to the Pascal triangle, and show how P can be generated by a simple recursion formula. Also show how each element of P^{-1} can be expressed in terms of the corresponding element of P. How is the origin of the polynomial $p_n(x)$ moved, if you replace P by P^{-1} in the matrix-vector equation that defines c?

(c) If you reverse the order of the elements of the vectors a, c—this may sometimes be a more convenient ordering—how is the matrix P changed?

Comment: With a terminology to be used much in this book, we can look upon a and c as different coordinate vectors for the same element in the n-dimensional linear space $\mathcal{P}_n$ of polynomials of degree *less than n*. The matrix P gives the coordinate transformation.

4. Derive recurrence relations and write a program for computing the coefficients of the *product* r of two polynomials p and q,

$$r(x) = p(x)q(x) = \left(\sum_{i=1}^m a_i x^{i-1}\right)\left(\sum_{j=1}^n b_j x^{j-1}\right) = \sum_{k=1}^{m+n-1} c_k x^{k-1}.$$

5. Let x, y be nonnegative integers, with $y \neq 0$. The division x/y yields the quotient q and the remainder r. Show that if x and y have a common factor, then that number is a divisor of r as well. Use this remark to design an algorithm for the determination of the greatest common divisor of x and y (*Euclid's algorithm*).

6. Derive a forward and a backward recurrence relation for calculating the integrals

$$I_n = \int_0^1 \frac{x^n}{4x+1}\, dx.$$

Why is in this case the forward recurrence stable and the backward recurrence unstable?

7. (a) Solve Example 1.3.1, with the following changes: Start the recursion (1.3.4) with $I_0 = \ln 1.2$, and compute and print the sequence $\{I_n\}$ until I_n for the first time becomes negative.

(b) Start the recursion (1.3.5) first with the condition $I_{19} = I_{20}$, then $I_{29} = I_{30}$. Compare the results you obtain and assess their approximate accuracy. Compare also with the results of 7 (a).

***8.** (a) Write a program (or study some library program) for finding the quotient $Q(x)$ and the remainder $R(x)$ of two polynomials $A(x), B(x)$, i.e., $A(x) = Q(x)B(x) + R(x)$, $\deg R(x) < \deg B(x)$.

(b) Write a program (or study some library program) for finding the coefficients of a polynomial with given roots.

***9.** (a) Write a program (or study some library program) for finding the greatest common divisor of two *polynomials.* Test it on a number of polynomials of your own choice. Choose also some polynomials of a rather high degree, and do not only choose polynomials with small integer coefficients. Even if you have constructed the polynomials so that they should have a common divisor, rounding errors may disturb this, and some tolerance is needed in the decision whether a remainder is zero or not. One way of finding a suitable size of the tolerance is to make one or several runs where the coefficients are subject to some small random perturbations, and find out how much the results are changed.

(b) Apply the programs mentioned in the last two problems for finding and eliminating multiple roots of a polynomial.

Hint: A multiple root of a polynomial is a common root of the polynomial and its derivative.

9. Determine the solution x to the triangular system $U_n(a)x = e_n$, where

$$U_n(a) = \begin{pmatrix} 1 & a & a & \cdots & a \\ & 1 & a & \cdots & a \\ & & 1 & \cdots & a \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix},$$

and e_n is the nth unit vector.

10. (a) Compute the LU factorization of A and $\det(A)$, where

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{pmatrix}.$$

(b) Solve the linear system $Ax = b$, where $b = (2, 10, 44, 190)^T$.

11. Compute the inverse matrix A^{-1}, where

$$A = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 4 & 1 & 2 \end{pmatrix},$$

by LU factorization and using $A^{-1} = U^{-1}L^{-1}$.

12. Show that there cannot exist a factorization

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}.$$

Hint: Equate the $(1,1)$-elements and deduce that either the first row or the first column in LU must be zero.

1.4 Numerical Solution of Differential Equations

1.4.1 Euler's Method

Approximate solution of *differential equations* is a very important task in scientific computing. Nearly all the areas of science and technology contain mathematical models which leads to systems of ordinary (or partial) differential equations. An **initial value problem** for an ordinary differential equation is to find $y(x)$ such that

$$\frac{dy}{dt} = f(t, y), \quad y(0) = c.$$

The differential equation gives, at each point (t, y), the direction of the tangent to the solution curve which passes through the point in question. The direction of the tangent changes continuously from point to point, but the simplest approximation (which was proposed as early as the 18th century by Euler) is that one studies the solution for only certain values of $t = t_n = nh$, $n = 0, 1, 2, \ldots$ (h is called the "step" or "step length") and assumes that dy/dt is constant between the points. In this way the solution is approximated by a polygon segment (Fig. 1.5.1) which joins the points (t_n, y_n), $0, 1, 2, \ldots$ where

$$y_0 = c, \quad \frac{y_{n+1} - y_n}{h} = f(t_n, y_n). \tag{1.4.1}$$

Thus we have a simple *recursion formula*, **Euler's method**:

$$y_0 = c, \quad y_{n+1} = y_n + hf(t_n, y_n), \quad n = 0, 1, 2 \ldots \tag{1.4.2}$$

During the computation, each y_n occurs first on the left-hand side, then *recurs*

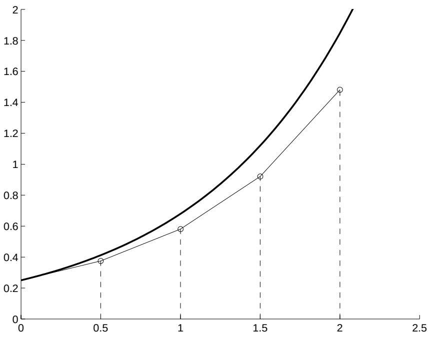


Figure 1.4.1. *Approximate solution of $dy/dx = y$, $y_0 = 0.25$, by Euler's method with $h = 0.5$.*

later on the right-hand side of an equation: hence the name **recursion formula**. (One could also call equation (1.4.2) an iteration formula, but one usually reserves the word "iteration" for the special case where a recursion formula is used solely as a means of calculating a limiting value.)

1.4.2 An Introductory Example

One of the most important techniques in computer applications to science and technology is the **step by step simulation** of a process or the time development of a system. A **mathematical model** is first set up, i.e., **state variables** which describe the essential features of the state of the system are set up. Then the laws are formulated, which govern *the rate of change of the state variables*, and other *mathematical relations* between these variables. Finally, these equations are programmed for a computer to calculate approximately, step by step, the development in time of the system.

The reliability of the results depends primarily on the goodness of the mathematical model and on the size of the time step. The choice of the time step is partly a question of economics. Small time steps may give you good accuracy, but also long computing time. More accurate numerical methods are often a good alternative to the use of small time steps. Such questions will be discussed in depth in Chapter 13.

The construction of a mathematical model is not trivial. Knowledge of numerical methods and programming helps also in that phase of the job, but more important is a good understanding of the fundamental processes in the system, and that is beyond the scope of this text. It is, however, important to realize that if the mathematical model is bad, no sophisticated numerical techniques or powerful computers can stop the results from being unreliable, or even harmful.

A mathematical model can be studied by analytic or computational techniques. Analytic methods do not belong to this text. We want, though, to emphasize that the comparison with results obtained by analytic methods, in the special cases when they can be applied, can be very useful when numerical methods and computer programs are tested. We shall now illustrate these general comments on a particular example.

Example 1.4.1. Consider the motion of a ball (or a shot) under the influence of gravity and air resistance. It is well known that the trajectory is a parabola, when the air resistance is neglected and the force of gravity is assumed to be constant. We shall still neglect the variation of the force of gravity and the curvature and the rotation of the earth. This means that we forsake serious applications to satellites, etc. We shall, however, take the air resistance into account. We neglect the rotation of the shot around its own axis. Therefore we can treat the problem a a motion in a plane, but we have to forsake the application to, for example, table tennis or a rotating projectile. Now we have introduced a number of assumptions, which define our **model** of reality.

The state of the ball is described by its position (x,y) and velocity (u,v), each of which has two Cartesian coordinates in the plane of motion. The x-axis is horizontal, and the y-axis is directed upwards. Assume that the air resistance is a force P, such that the direction is opposite to the velocity, and the strength is proportional to the square of the speed and to the square of the radius R of the shot. We can then write,

$$P_x = -mzu, \quad P_y = -mzv, \quad z = \frac{cR^2}{m}\sqrt{u^2+v^2}, \tag{1.4.3}$$

where m is the mass of the ball.

For the sake of simplicity we assume that c is a constant. It actually depends on the density and the viscosity of the air. Therefore, we have to forsake the application to cannon shots, where the variation of the density with height is important. If one has access to a good model of the atmosphere, the variation of c would not make the numerical simulation much more difficult. This contrasts to analytic methods, where such a modification is likely to mean a considerable complication. In fact, even with a constant c, a purely analytic treatment offers great difficulties.

Newton's law of motion tells us that,

$$m\,du/dt = P_x, \qquad m\,dv/dt = -mg + P_y, \tag{1.4.4}$$

where the term $-mg$ is the force of gravity. Inserting (1.4.3) into (1.4.4) and dividing by m we get

$$du/dt = -zu, \qquad dv/dt = -g - zv, \tag{1.4.5}$$

By the definition of velocity,

$$dx/dt = u, \qquad dy/dt = v, \tag{1.4.6}$$

Equations (1.4.5) and (1.4.6) constitute a system of four differential equations for the four variables x, y, u, v. The initial state x_0, y_0, and u_0, v_0 at time $t_0 = 0$ is assumed to be given. A fundamental proposition in the theory of differential equations tells that, if initial values of the state variables u, v, x, y are given at some initial time $t = t_0$, then they will be uniquely determined for all $t > t_0$.

The simulation in Example 1.4.1 means that, at a sequence of times, t_n, $n = 0, 1, 2, \ldots$, we determine the approximate values, u_n, v_n, x_n, y_n. We first look at the simplest technique, using Euler's method with a constant time step h. Set therefore $t_n = nh$. We replace the derivative du/dt by the forward difference quotient $(u_{n+1} - u_n)/h$, and similarly for the other variables. Hence after multiplication by h, the differential equations are replaced by the following system of **difference equations**:

$$\begin{aligned} u_{n+1} - u_n &= -hz_n u_n, \\ v_{n+1} - v_n &= -h(g + z_n v_n), \\ x_{n+1} - x_n &= hu_n, \quad y_{n+1} - y_n = hv_n, \end{aligned} \tag{1.4.7}$$

from which u_{n+1}, v_{n+1}, x_{n+1}, y_{n+1}, etc. are solved, step by step, for $n = 0, 1, 2, \ldots$, using the provided initial values u_0, v_0, x_0, y_0. Here z_n is obtained by insertion of $u = u_n$, $v = v_n$ into (1.4.3).

We performed these computations until y_{n+1} became negative for the first time, with $g = 9.81$, $\phi = 60^o$, and the initial values

$$x_0 = 0, \quad y_0 = 0, \quad u_0 = 100\cos\phi, \quad v_0 = 100\sin\phi.$$

In Fig. 1.4.2 are shown curves obtained for $h = 0.01$, and $cR^2/m = 0.25i \cdot 10^{-3}$, $i = 0, 1, 2, 3, 4$. There is, in this graphical representation, also an error due to the limited resolution of the plotting device.

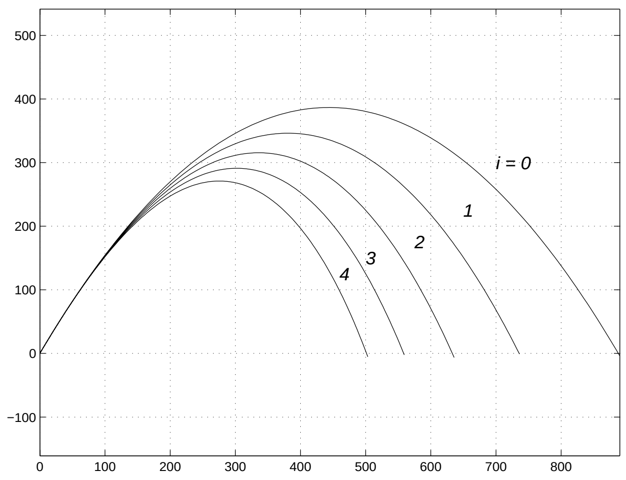


Figure 1.4.2. *Approximate trajectories computed with Euler's method for* $cR^2/m = 0.25i \cdot 10^{-3}$, $i = 0 : 4$, *and* $h = 0.01$.

In Euler's method the state variables are *locally approximated by linear functions* of time, one of the often recurrent ideas in numerical computation. We can use the same idea for computing the coordinate x^* of the point, where the shot hits the ground. Suppose that y_{n+1} becomes negative for the first time when $n = N$. For $x_N \le x \le x_{N+1}$ we then approximate y by a linear function of x, represented by the secant through the points (x_N, y_N) and(x_{N+1}, y_{N+1}) , i.e.,

$$y = y_N + (x - x_N)\frac{y_{N+1} - y_N}{x_{N+1} - x_N}.$$

By setting $y = 0$ we obtain

$$x^* = x_N - y_N\frac{x_{N+1} - x_N}{y_{N+1} - y_N}. \tag{1.4.8}$$

The error from the linear approximation in (1.4.8) used for the computation of x^* is proportional to h^2. It is thus approximately equal to the error committed in *one single step* with Euler's method, and hence of less importance than the other error.

The case without air resistance ($i = 0$) can be solved exactly. In fact it can be shown that $x^* = 2u_0v_0/9.81 = 5000 \cdot \sqrt{3}/9.81 = 882.7986$. The computer produced $x^* = 883.2985$ for $h = 0.01$, and $x^* = 883.7984$ for $h = 0.02$. The error for $h = 0.01$ is therefore 0.4999, and for $h = 0.02$ it is 0.9998. The approximate proportionality to h is thus verified, actually more strikingly than could be expected!

It can be shown that *the error in the results obtained with Euler's method is also proportional to h (not h^2).* Hence a disadvantage of the above method is that the step length h must be chosen quite short if reasonable accuracy is desired. In order to improve the method we can apply another idea mentioned in the previously, namely Richardson extrapolation. The application differs a little from the one you saw there, because now the error is approximately proportional to h, while for the trapezoidal rule it was approximately proportional to h^2. For $i = 4$, the computer produced $x^* = 500.2646$ and $x^* = 500.3845$ for, respectively, $h = 0.01$ and $h = 0.02$. Now let x^* denote the *exact* coordinate of the landing point. Then

$$x^* - 500.2646 \approx 0.01k, \qquad x^* - 500.3845 \approx 0.02k.$$

By elimination of k we obtain

$$x^* \approx 2 \cdot 500.2646 - 500.3845 = 500.1447,$$

which should be a more accurate estimate of the landing point. By a more accurate integration method we obtained 500.1440. So in this case, we gained more than two decimal digits by the use of Richardson extrapolation.

The simulations shown in Fig. 1.5.2 required about 1500 time steps for each curve. This may seem satisfactory, but we must not forget that this is a very small task, compared to most serious applications. So we would like to have a method that allows *much larger time steps* than Euler's method to provide, e.g., an accuracy that fits well to that of the plotting of the orbit on a screen.

1.4.3 A Second Order Method

In step by step computations we have to distinguish between the **local error**, i.e., the error that is committed at a single step, and the **global error**, i.e., the error of the final results. Recall that we say that a method is accurate of order p, if its global error is approximately proportional to h^p. Euler's method is only first order accurate; we shall below present a method that is second order accurate. To achieve the same accuracy as with Euler's method the number of steps can then be reduced to about the square root of the number of steps in Euler's method, e.g., in the above ball problem to $\sqrt{1500} \approx 40$ steps. Since the amount of work is closely proportional to the number of steps this is an enormous saving!

Another question is how the step size h is to be chosen. It can be shown that even for rather simple examples (see below) it is adequate to use very *different step size* in different parts of the computation. Hence the automatic control of the step size (also called *adaptive* control) is an important issue.

Both requests can be met by an improvement of the Euler method (due to Runge) obtained by the applying the Richardson extrapolation in every second step. This is different from our previous application of the Richardson idea. We first introduce a better notation by writing a **system of differential equations** and the initial conditions in vector form

$$d\mathbf{y}/dt = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(a) = \mathbf{c}, \tag{1.4.9}$$

where $\mathbf{y}$ is a column vector that contains all the state variables.[6] With this notation

[6]The boldface notation is temporarily used for vectors *in this section*, not in the rest of the book.

methods for large systems of differential equations can be described as easily as methods for a single equation. The change of a system with time can then be thought of as a motion of the state vector in a multidimensional space, where the differential equation defines the **velocity field**. This is our first example of the central role of vectors and matrices in modern computing. We temporarily use *superscripts* for the vector components, because we need subscripts for the same purpose as in the above description of Euler's method.

For the ball example, we have by (1.4.5) and (1.4.6)

$$\mathbf{y} = \begin{pmatrix} y^1 \\ y^2 \\ y^3 \\ y^4 \end{pmatrix} \equiv \begin{pmatrix} x \\ y \\ u \\ v \end{pmatrix}, \quad \mathbf{f}(t,\mathbf{y}) = \begin{pmatrix} y^3 \\ y^4 \\ -zy^3 \\ -g - zy^4 \end{pmatrix}, \quad \mathbf{c} = 10^2 \begin{pmatrix} 0 \\ 0 \\ \cos\phi \\ \sin\phi \end{pmatrix},$$

where

$$z = \frac{cR^2}{m}\sqrt{(y^3)^2 + (y^4)^2}.$$

The computations in the step which leads from t_n to t_{n+1} are then as follows:

i. One Euler step of length h yields the estimate:

$$\mathbf{y}^*_{n+1} = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n).$$

ii. Two Euler steps of length $\frac{1}{2}h$ yield another estimate:

$$\mathbf{y}_{n+\frac{1}{2}} = \mathbf{y}_n + \frac{1}{2}h\mathbf{f}(t_n, y_n); \qquad \mathbf{y}^{**}_{n+1} = \mathbf{y}_{n+\frac{1}{2}} + \frac{1}{2}h\mathbf{f}(t_{n+1/2}, \mathbf{y}_{n+1/2}),$$

where $t_{n+1/2} = t_n + h/2$.

iii. Then $\mathbf{y}_{n+1}$ is obtained by Richardson extrapolation:

$$\mathbf{y}_{n+1} = \mathbf{y}^{**}_{n+1} + (\mathbf{y}^{**}_{n+1} - \mathbf{y}^*_{n+1}).$$

It is conceivable that this yields a 2nd order accurate method. It is left as an exercise (Problem 2) to verify that *this scheme is identical to the following somewhat simpler scheme* known as **Runge's 2nd order method**:

$$\begin{aligned} \mathbf{k}_1 &= h_n\mathbf{f}(t_n, \mathbf{y}_n); \\ \mathbf{k}_2 &= h_n\mathbf{f}(t_n + h_n/2, \mathbf{y}_n + \mathbf{k}_1/2); \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \mathbf{k}_2, \end{aligned} \tag{1.4.10}$$

where we have replaced h by h_n in order to include the use of variable step size. Another explanation of the 2nd order accuracy of this method is that the displacement $\mathbf{k}_2$ equals the product of the step size and a sufficiently accurate estimate of the velocity at the *midstep* of the time step. A more detailed analysis of this method comes in Sec. 13.3.2. Sometimes this method is called the improved Euler method or Heun's method, but these names are also used to denote other 2nd order accurate methods.

We shall now describe how the step size can be **adaptively** (or automatically) controlled by means of a tolerance TOL, by which the user tells the program how large error he tolerates in values of variables (relative to the values themselves).[7] Compute

$$\delta = \max_i |k_2^i - k_1^i|/|3y^i|,$$

where δ is related to the *relative* errors of the components of the vector $\mathbf{y}$, see below.

A step size is *accepted* if $\delta \leq \text{TOL}$, and the next step should be

$$h_{next} = h \min\{1.5, \sqrt{\text{TOL}/(1.2\delta)}\},$$

where 1.2 is a safety factor, since the future is never exactly like the past ....[8]

A step is *rejected*, if $\delta > \text{TOL}$, and *recomputed* with the step size

$$h_{next} = h \max\{0.1, \sqrt{\text{TOL}/(1.2\delta)}\},$$

The program needs a suggestion for the size of the first step. This can be be a very rough guess, because the step size control described above, will improve it automatically, so that an adequate step size is found after a few steps (or recomputations, if the suggested step was too big). In our experience, a program of this sort can efficiently handle guesses that are wrong by several powers of 10. If $y(a) \neq 0$ and $y'(a) = 0$, you may try the initial step size

$$h = \frac{1}{4} \sum_i |y^i| \Big/ \sum_i |dy^i/dt|$$

evaluated at the initial point $t = a$. When you encounter the cases $y(a) = 0$ or $y'(a) = 0$ for the first time, you are likely to have gained enough experience to suggest something that the program can handle. More professional programs take care of this detail automatically.

The request for a certain *relative* accuracy may cause trouble when some components of y are close to zero. So, already in the first version of your program, you had better *replace y^i in the above definition of δ by $\bar{y}^i$* $= \max\{|y^i|, 0.001\}$. A more detailed discussion of such matters follows in Sections 13.1 and 13.2, see in particular Computer Exercise 13.1.1. (You may sometimes have to replace the default value 0.001 by something else.)

It is a good habit to make a second run with a predetermined sequence of times (if your program allows this) instead of adaptive control. Suppose that the sequence of times used in the first run is t_0, t_1, t_2, ... Divide each subinterval

[7]With the terminology that will be introduced in the next chapter, TOL is, with the step size control described here, related to the *global relative errors* . At the time of writing, this contrasts to most codes for the solution of ordinary differential equations, in which the *local* errors per step are controlled by the tolerance.

[8]The square root occurring here is due to the fact that this method is 2nd order accurate, i.e., the global error is almost proportional to the square of the step size and δ is approximately proportional to h^2.

$[t_n, t_{n+1}]$ into two steps of equal length. So, the second run still has variable step size and twice as many steps as the first run. The errors are therefore expected to be approximately $\frac{1}{4}$ of the errors of the first run. The first run can therefore use a tolerance that is 4 times as large than the error you can tolerate in the final result. Denote the results of the two runs by $y_I(t)$ and $y_{II}(t)$. You can plot $\frac{1}{3}(y_{II}(t) - y_I(t))$ versus t; this is an error curve for $y_{II}(t)$ Alternatively you can add $\frac{1}{3}(y_{II}(t) - y_I(t))$ to $y_{II}(t)$. This is another application of the Richardson extrapolation idea. The cost is only 50% more work than the plain result without an error curve.

If there are no singularities in the differential equation, $\frac{1}{3}(y_{II}(t) - y_I(t))$ *strongly overestimates the error of the extrapolated values*—typically by a factor like $\text{TOL}^{-1/2}$. It is, however, a non-trivial matter to find an error curve that strictly and realistically tells how good the extrapolated results are. There will be more comments about these matters in Sec. 3.3.4 and Example 13.2.1. The reader is advised to test experimentally how this works on examples where the exact results are known.

An easier, though inferior, alternative is to run a problem with two different tolerances. One reason why it is inferior is that the two runs do not "keep in step". For example, Richardson extrapolation cannot be easily applied.

If you request very high accuracy in your results, or if you are going to simulate a system over a very long time, you will need a method with a higher order of accuracy than two. The reduction of computing time if you replace this method by a higher order method can be large, but the improvements are seldom as drastic as when you replace Euler's method by a second order accurate scheme like this. Runge's 2nd order method is, however, no universal recipe. There are special classes of problems, notably the problems which are called “stiff”, which need special methods. These matters are treated in Chapter 13.

One advantage of a second order accurate scheme when requests for accuracy are modest, is that the quality of the computed results is normally not ruined by the use of *linear interpolation* at the graphical output, or at the post-processing of numerical results. (After you have used a more than second order accurate integration method, it may be necessary to use a more sophisticated interpolation at the graphical or numerical treatment of the results.)

Example 1.4.2. The differential equation $y' = -\frac{1}{2}y^3$, with initial condition $y(1) = 1$, was treated by a program, essentially constructed as described above, with $\text{TOL} = 10^{-4}$ until $t = 10^4$.

In this example we can compare with the exact solution, $y(t) = t^{-1/2}$. It was found that the actual relative error stayed a little less than $1.5\,\text{TOL}$ all the time when $t > 10$. The step size increased almost linearly with t from $h = 0.025$ to $h = 260$. The number of steps increased almost proportionally to $\log t$; the total number of steps was 374. Only one step had to be recomputed (except for the first step, where the program had to find an appropriate step size).

The computation was repeated with $\text{TOL} = 4 \cdot 10^{-4}$. The experience was the same, except that the steps were about twice as long all the time. This is what can be expected, since the step sizes should be approximately proportional to $\sqrt{\text{TOL}}$, for a second order accurate method. The total number of steps was 194.

Example 1.4.3. The example of the motion of a ball was treated by Runge's 2nd order method with the constant step size $h = 0.9$. The coordinate of the landing point became $x^* = 500.194$, which is more than twice as accurate than the result obtained by Euler's method (without Richardson extrapolation) with $h = 0.01$, which uses about 90 times as many steps.

We have now seen a variety of ideas and concepts which can be used in the development of numerical methods. A small warning is perhaps warranted here: it is not certain that the methods will work as well in practice as one might expect. This is because approximations and the restriction of numbers to a certain number of digits introduce errors which are propagated to later stages of a calculation. The manner in which errors are propagated is decisive for the practical usefulness of a numerical method. We shall examine such questions in Chapter 2. Later chapters will treat **propagation of errors** in connection with various typical problems.

The risk that error propagation may up-stage the desired result of a numerical process should, however, not dissuade one from the use of numerical methods. It is often wise, though, to experiment with a proposed method on a simplified problem before using it in a larger context. The development of hardware as well as software has created a far better environment for such work than we had a decade ago. In this area too, the famous phrase of the Belgian-American chemist Baekeland holds:

"*Commit your blunders on a small scale and make your profits on a large scale.*"

Review Questions

1. Explain the difference between the local and global error of a numerical method for solving a differential equation. What is meant by a method the order accuracy for a method?

2. Describe how Richardson extrapolation can be used to increase the order of accuracy of Euler's method.

Problems

1. Integrate numerically using Euler's method the differential equation $dy/dx = y$, with initial conditions $y(0) = 1$, to $x = 0.4$:

(a) with step length $h = 0.2$ and $h = 0.1$.

(b) Extrapolate to $h = 0$, using the fact that the error is approximately proportional to the step length. Compare the result with the exact solution of the differential equation and determine the ratio of the errors in the results in (a) and (b).

(c) How many steps would have been needed in order to attain, without using extrapolation, the same accuracy as was obtained in (b)?

Computer Exercises

1. (a) Write a program for the simulation of the motion of the ball, using Euler's method and the same initial values and parameter values as above. Print only x, y at integer values of t and at the last two points (i.e. for $n = N$ and $n = N + 1$) as well as the coordinate of the landing point. Take $h = 0.05$ and $h = 0.1$. As post-processing, improve the estimates of x^* by Richardson extrapolation, and estimate the error by comparison with the results given in the text above.

(b) In Equation (1.4.8) replace in the equations for x_{n+1} and y_{n+1} the right hand sides u_n and v_n by, respectively, u_{n+1} and v_{n+1}. Then proceed as in (a) and compare the accuracy obtained with that obtained in (a).

(c) Choose initial values which correspond to what you think is reasonable for shot put. Make experiments with several values of u_0, v_0 for $c = 0$. How much is x^* influenced by the parameter cR^2/m?

2. Verify that Runge's 2nd order method, as described by equation (1.4.10), is equivalent to the scheme described a few lines earlier (with Euler steps and Richardson extrapolation).

3. Write a program for Runge's 2nd order method with automatic step size control that can be applied to a system of differential equations, *or* use the Matlab program on the diskette. Store the results so that they can be processed afterwards, e.g., for making table of the results, and/or curves an be drawn showing $y(t)$ versus t, or (for a system) y^2 versus y^1, or some other interesting curves.
Apply the program to Examples 1.4.2 and 1.4.3, and to the circle test, i.e.

$$y_1' = -y_2, \qquad y_2' = y_1,$$

with initial conditions $y_1(0) = 1$, $y_2(0) = 0$. Verify that the exact solution is a uniform motion along the unit circle in the (y_1, y_2)-plane. Stop the computations after 10 revolutions ($t = 20\pi$). Make experiments with different tolerances, and determine how small the tolerance has to be in order that the circle on the screen should not become "thick".

1.5 Monte Carlo Methods

1.5.1 Origin of Monte Carlo Methods

In most of the applications of probability theory one makes a mathematical formulation of a stochastic problem (i.e., a problem where chance plays some part), and then solves the problem by using analytical or numerical methods. In the **Monte Carlo method**, one does the opposite; a mathematical or physical problem is given, and one constructs **numerical game of chance**, the mathematical analysis of which leads to the same equations as the given problem, e.g., for the probability of some event, or for the mean of some random variable in the game. One plays it N times and estimates the relevant quantities by traditional statistical methods. Here N is a large number, because the standard deviation of a statistical estimate typically *decreases only inversely proportional to* $\sqrt{N}$.

The idea behind the Monte Carlo method was used by the Italian physicist Enrico Fermi to study the neutron diffusion in the early 1930s. Fermi used a small mechanical adding machine for this purpose. With the development of computers larger problems could be tackled. At Los Alamos in the late 1940s the use of the method was pioneered by John von Neumann, Stanislaw Ulam and others for many problems in mathematical physics including approximating complicated multidimensional integrals. The picturesque name of the method was coined by Nicholas Metropolis.

The Monte Carlo method is now so popular that the definition is too narrow. For instance, in many of the problem where the Monte Carlo method is successful, there is already an element of chance in the system or process which one wants to study. Thus such games of chance can be considered to be a numerical simulation of the most important aspects. In a wider sense Monte Carlo methods were used as early as in the 1800s, under different names, e.g., experimental sampling.

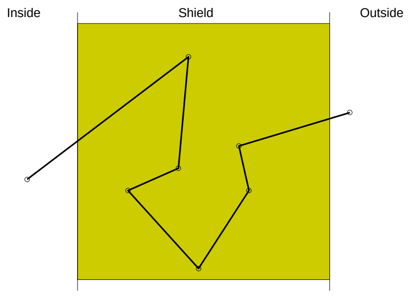


Figure 1.5.1. *Neutron scattering.*

Monte Carlo methods may be used, when the changes in the system are described with a much more complicated type of equation than a system of ordinary differential equations. The following are some areas where the method has been applied:

(a) Problems in reactor physics; for example, a neutron, because it collides with other particles, is forced to make a random journey. In infrequent but important cases the neutron can go through a layer of (say) shielding material (see Fig. 1.6.1).

(b) Technical problems concerning traffic (telecommunication, railway networks, regulation of traffic lights and other problems concerning automobile traffic, etc.).

(c) Queuing problems.

(d) Models of conflict.

(e) Approximate computation of multiple integrals.

(f) Stochastic models in financial mathematics.

In a simulation, one can study the result of various actions more cheaply, more quickly, and with less risk of organizational problems than if one were to take the corresponding actions on the actual system. In particular, for problems in applied operations research, it is quite common to take a shortcut from the actual system to a computer program for the game of chance, without formulating any mathematical equations. The game is then a model of the system. In order for the term Monte Carlo method to be correctly applied, however, **random choices** should occur in the calculations. This is achieved by using so-called **random numbers**; the values of certain variables are determined by a process comparable to dice throwing.

Simulation is so important that several special programming languages have been developed exclusively for its use.

1.5.2 Random and Pseudo-Random Numbers

The sequence of twenty digits

$$11100 \;\; 01001 \;\; 10011 \;\; 01100$$

is a record of twenty tosses of a coin where "heads" are denoted by 1 and "tails" by 0. Such digits are sometimes called (binary) **random digits**, assuming that we have a perfect coin—i.e., that heads and tails have the same probability of occurring. We also assume that the tosses of the coin are made in a statistically independent way. (Of course, these assumptions cannot be obtained in practice.)

Similarly, decimal random digits could in principle be obtained by using an icosahedral (twenty-sided) dice, and assigning each decimal digit to two of its sides. In the early 1950s the Rand Corporation constructed a million-digit table of random numbers using an electrical "roulette wheel" ([2, 1955]). The wheel had 32 slots, of which 12 were ignored; the others were numbered from 0 to 9 twice. The following row of decimal digits is taken from such a table:

$$55693 \;\; 02945 \;\; 81723 \;\; 43588 \;\; 81350 \;\; 76302 \;\; \dots \tag{1.5.1}$$

Random digits can be packed together to give sequence of equi-distributed integers. The sequence in (1.5.1) can be considered as five-digit random numbers, where each element in the sequence has probability of 10^{-5} of taking on the value, 0,1,2,...,99,999. From the same digits one can also construct the sequence

$$0.556935, 0.029455, 0.817235, 0.435885, 0.813505, 0.763025, \dots, \tag{1.5.2}$$

which can be considered a good approximation to a sequence of independent observations of a variable which is **uniformly distributed** (or rectangularly distributed) on the interval $[0, 1]$. The 5 in the sixth decimal place is added in order to get the correct mean (without this the mean would be 0.499995 instead of 0.5).

By far the most common method of generating random numbers is to use an algorithm on line with the computer program, which generates a sequence of "random numbers" by a constructive process. Sequences obtained in this way are uniquely determined by one or more starting values (the **seeds**) and are reproducible. If by a "random" sequence we mean a sequence that satisfies no predictable rule such sequences are not truly random and they are called **pseudo-random** sequences. It is difficult to give a satisfactory definition of a psedo-random sequence. In general the sequence should possess similar statistical properties as random sequences.[9]

An important characteristic of a random number generator is its **period**, which is the maximum length of the sequence before it begins to repeat. Note that if the algorithm for computing x_{n+1} only depends on x_n, then the entire sequence repeats one x_0 is duplicated. Other requirements are that the generator should be fast and use little memory.

The most widely used generators for producing pseudo-random numbers are the **multiple recursive generator** based on linear recurrences of order k

$$x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k} c \mod m, \tag{1.5.3}$$

i.e., x_n is the remainder obtained when the right hand side is divided by the modulus m. Here m is a positive integer and the coefficients $a_1, \dots, a_k$ belong to the set $\{0, 1, \dots, m-1\}$. The state at step i is $s_i = (x_{n-k+1}, \dots, x_n)$ and the generator is started from a seed $s_{k-1} = (x_0, \dots, x_{k-1})$.

If m is a prime number and and if the coefficients a_i satisfy certain conditions, then the generated sequence has the maximal period length $m^k - 1$; see Knuth [12]. When $k = 1$, we obtain the classical **linear congruential generator**.

Example 1.5.1. A linear congruential generator defined by

$$x_n = 16807 x_{n-1} \mod (2^{31} - 1), \tag{1.5.4}$$

with period length $(2^{31} - 2)$, was proposed originally by Lewis, Goodman, and Miller (1969). It has been used in many software libraries for statistics and optimization. This generator but with the multiplier $7^7 = 823543$ was used in early versions of MATLAB. From MATLAB 5 it has been replaced with a new generator, which can generate all the floating point numbers in the closed interval $[2^{-53}, 1 - 2^{-53}]$. Theoretically it can generate 2^{1492} values before repeating itself.

The pseudo-random integers generated by the above method can be converted into random numbers in the interval $[0, 1]$ as exemplified above. The least significant digits of these numbers turns out not to be very random. To use digits generated in this way to form more numbers with fewer digits can thus be risky. There are other (more time-consuming) methods of producing pseudo-random numbers where the above operation is risk-free.

Example 1.5.2. Buffon's needle problem (Buffon 1707–1778): Suppose a board is ruled with equidistant parallel lines and that a needle fine enough to be considered a segment of length l not longer than the distance d between consecutive lines is thrown on the board. The probability is then $2l/(\pi d)$ that it will hit one of the lines.

[9] "Anyone who considers arithmetic methods of producing random numbers is, of course, in a state of sin"; John von Neumann (1951).

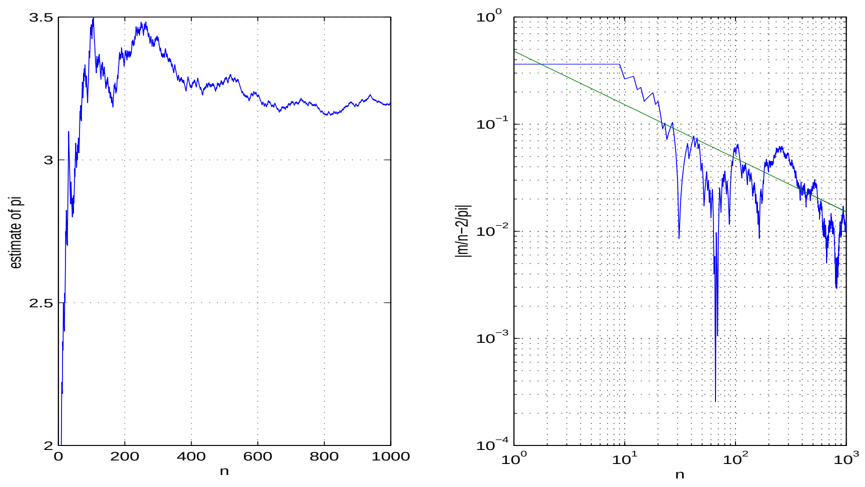


Figure 1.5.2. *The left part shows how the estimate of π varies with the number of throws. The right part compares $|m/n-2/\pi|$ with the standard deviation of m/n. The latter is inversely proportional to $n^{1/2}$, and is therefore a straight line in the figure.*

The Monte Carlo method and this game can be used to approximate the value of π. Take the distance δ between the center of the needle and the lines and the angle ϕ between the needle and the lines to be random numbers. By symmetry we can choose these to be rectangularly distributed on $[0, d/2]$ and $[0, \pi/2]$, respectively. Then the needle hits the line if $\delta < (l/2)\sin\phi$.

We took $l = d$. Let m be the number of hits in the first n throws in a Monte Carlo simulation with 1000 throws. The expected value of m/n is therefore $2/\pi$. So, $2n/m$ is an estimate of π after n throws. In the left part of Fig. 1.6.2 we see, how $2n/m$ varies with n in one simulation. The right part compares $|m/n - 2/\pi|$ with the standard deviation of m/n, which equals $\sqrt{(}2/\pi(1-2/\pi)/n)$ and is, in the log-log-diagram, represented by a straight line, the slope of which is $-1/2$. (The spikes, directed downwards in the figure, typically indicate where $m/n - 2/\pi$ changes sign.)

By arithmetic operations on numbers R uniformly distributed on $[0, 1]$ one can form random numbers with other distributions. For example, $S = a + (b-a)R$ will be uniformly distributed on $[a, b]$. The variable $R' = 1 - R$ is also uniformly distributed on $[0, 1]$. For example, from the sequence in (1.5.2) we get the following sequence for R':

$$0.443065, 0.970545, 0.182765, 0.564115, 0.186495, 0.236975, \ldots, \tag{1.5.5}$$

The sequence (1.5.5) is said to be the **antithetic sequence** derived from (1.5.2). Such sequence are an important means of reducing the variance of estimates made using the Monte Carlo method, see next section.

We now give a general method for generating random numbers with a given continuous distribution function $F(x)$. Let R be a random number uniformly distributed on $[0,1]$, and compute X by solving the equation $F(X)=R$; see Fig. 1.6.3. Then, since $F(x)$ is a nondecreasing function,

$$P[X \le x] = P[F(X) \le F(x)] = P[R \le F(x)],$$

but the last expression is equal to $F(X)$, since by definition of a uniform distribution we have $P[R \le r] = r$ for any r in $[0,1]$. Hence, X has the desired distribution.

The following example shows how one can obtain **normally distributed** random numbers in an easier way than the general method:

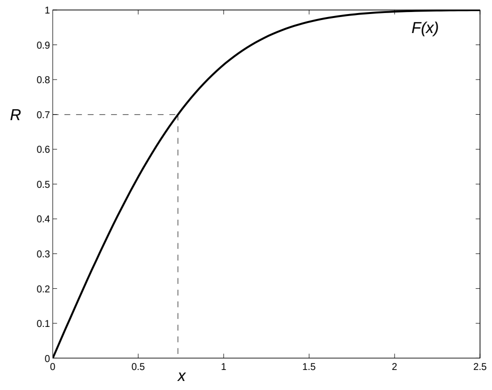


Figure 1.5.3. *Random number with distribution $F(x)$.*

Example 1.5.3. Let R_1, R_2 be two independent, uniformly distributed random numbers in $[0,1]$. Then two independent, normally distributed random numbers N_1, N_2, with mean zero and standard deviation 1 can be obtained using the following **Box–Muller's transformation**:

$$N_1 = \sqrt{-2\ln R_1}\cos(2\pi R_2), \quad N_2 = \sqrt{-2\ln R_1}\sin(2\pi R_2).$$

We shall not derive the rule here, but point out that N_1, N_2 can be considered to be rectangular coordinates of a oint whose polar coordinates (r,ϕ) are determined by the equations

$$r^2 = N_1^2 + N_2^2 = -2\ln R_1, \quad \phi = 2\pi R_2.$$

Thus the problem is to show that the distribution function for a pair of independent, normally distributed random variables is rotationally symmetric (uniformly distributed angle) and that their sum of squares is exponentially distributed with mean 2.

Notice that we can get normally distributed random variables with mean m and standard deviation σ by forming $m+\sigma N_1, m+\sigma N_2$. An antithetic sequence of normally distributed numbers can be obtained simply by reversing the sign of the original sequence. For a related method, which avoids the trigonometric functions in Box–Muller's transformation, see [4, p. 247].

Example 1.5.4. Exponentially distributed random numbers with mean $1/\lambda$ have the distribution function

$$F(x) = 1 - e^{-\lambda x} \tag{1.5.6}$$

They can be generated as follows. Let in $[0, 1]$. Then from the general rule given above we see that X can be obtained by solving the equation $1 - e^{-\lambda X} = 1 - R$, or hence by forming

$$X = -\lambda^{-1} \ln R.$$

One important use of exponentially distributed random numbers is in the generation of so-called **Poisson processes**. Such processes are often fundamental in models of telecommunications systems and other service systems. A Poisson process with frequency parameter λ is a sequence of events characterized by the property that the probability of occurrence of an event in a short time interval $(t, t + \Delta t)$ is equal to $\lambda \cdot \Delta t + o(\Delta t)$, independent of the sequence of events previous to time t. In applications and "event" can mean a call on a telephone line, the arrival of a customer in a store, etc. For simulating a Poisson process one can use the important property that the intervals of time between two successive events are independent exponentially distributed random numbers. Thus Poisson processes can be generated using the rule given in Example 1.5.4.

1.5.3 Reduction of Variance.

From statistics, we know that if one makes n independent observations of a quantity whose standard deviation is σ, then the standard deviation of the mean is $\sigma/\sqrt{n}$. In the Monte Carlo method, one can influence the value of σ by designing the experiment in various ways. One important technique is to make the experiments in antithetic pairs and then computing the mean for each pair of experiments (see Example 1.5.5 and Exercise 4).

Assume that one has two ways (which require the same amount of work) of carrying out an experiment, and these experiments have standard deviations σ_1 and σ_2 associated with them. If one repeats the experiments n_1 and n_2 times (respectively), the same precision will be obtained if $\sigma_1/\sqrt{n_1} = \sigma_2/\sqrt{n_2}$, or

$$n_1/n_2 = \sigma_1^2/\sigma_2^2. \tag{1.5.7}$$

Thus if a variance reduction by a factor k can be achieved, then the number of experiments needed is also reduced by the same factor k.

Example 1.5.5. In ten simulation and their antithetic experiments of a service system the following two sets of values were obtained for the treatment time using a sequence:

$$\begin{array}{cccccccccc}
685 & 1,045 & 718 & 615 & 1,021 & 735 & 675 & 635 & 616 & 889 \\
731 & 521 & 585 & 710 & 527 & 574 & 607 & 698 & 761 & 532^{\cdot}
\end{array}$$

Table 1.5.1. *Simulation of patients at a polyclinic.*

	$k=1$					$k=2$	
P_{no}	P_{arr}	T_{beg}	R	T_{time}	T_{end}	P_{arr}	T_{end}
1	0*	0	211	106	106	0*	106
2	50	106	3	2	108	0	108
3	100	108	53	26	134	50	134
4	150*	150	159	80	230	100	214
5	200	230	24	12	242	150	226
6	250*	250	35	18	268	200	244
7	300*	300	54	27	327	250*	277
8	350*	350	39	20	370	300*	320
9	400*	400	44	22	422	350*	372
10	450*	450	13	6	456	400*	406
Σ	2,250			319	2,663	1,800	2,407

From the first experiment the mean for the treatment time is estimated as 763 ± 52. Using the sequence means

$$708 \quad 783 \quad 652 \quad 662 \quad 774 \quad 654 \quad 641 \quad 666 \quad 688 \quad 710,$$

one gets the estimate 694 ± 16.

When one instead supplemented the first sequence with ten values using independent random numbers, the estimate 704 ± 36 using all twenty values was obtained. These results indicate that, in this example, using antithetical sequence produces the desired accuracy with $(16/36)^2 \approx 1/5$ of the work.

Roughly speaking, since the influence of chance has opposing effects in the two antithetic experiments, one can presume that the effect of chance on the *means* is much less than the effect of chance in the original experiments In Exercise 5 we give examples of how to make a quantitative estimate of the reduction of variance accomplished with the use of antithetic experiments.

Example 1.5.6. Monte Carlo methods have been successfully used to study queuing problems. A well known example is a study by Bailey [1] to determine how to give appointment times to patients at a polyclinic, see Exercise 4. The aim is to find a suitable balance between the mean waiting times of both patients and doctors. This problem was in fact solved analytically—much later—after Bailey already had gotten the results that he wanted; this situation is not uncommon when numerically methods (and especially Monte Carlo methods) have been used.

Suppose that k patients have been booked at the time $t = 0$ (when the clinic opens), and that the rest of the patients (altogether 10) are booked at intervals of 50 time units thereafter. The time of treatment is assumed to be exponentially distributed with mean 50. (Bailey used a distribution function which was based on empirical data.) Three alternatives, $k = 1, 2, 3$, are to be simulated. *By using the same random numbers for each k (hence the same treatment times) one gets a* **reduced variance** *in the estimate of the change in waiting times as k varies.*

The computations are shown in the Table 1.5.4. The following abbreviations are used: P = patient, D = doctor, T = treatment. An asterisk indicates that the patient did not need to wait. In the table P_{arr} follows from the rule for booking patients given previously. The treatment time T_{time} equals $50R/100$ where R are exponentially distributed numbers with mean 100 taken from a table. T_{beg} equals the larger of the number P_{arr} (on the same row) and T_{end} (in the row just above), where $T_{end} = T_{beg} + T_{treat}$.

From the table we find that for $k = 1$ the doctor waited the time $D = 456 - 319 = 137$; the total waiting time for patients was $P = 2,663 - 2,250 - 319 = 94$. For $k = 2$ the corresponding waiting times were $D = 406 - 319 = 87$ and $P = 2,407 - 1,800 - 319 = 288$. Similar calculations for $k = 3$ gave $D = 28$ and $P = 553$ (see Fig. 1.6.4). For $k \geq 4$ the doctor never needs to wait.

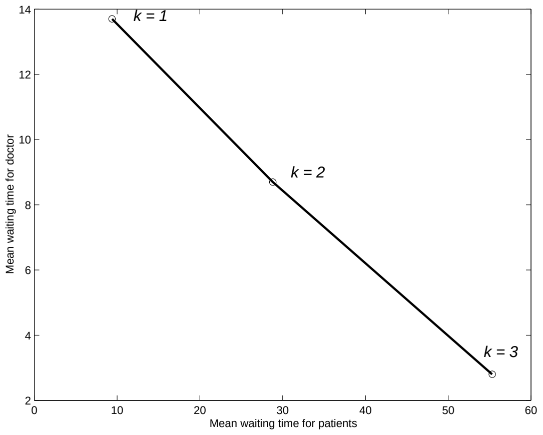


Figure 1.5.4. *Mean waiting times for doctor/patients at polyclinic.*

One cannot, of course, draw any tenable conclusions from one experiment. More experiment should be made in order to put the conclusions ons statistically solid ground. Even isolated experiments, however, can give valuable suggestions for the planning of subsequent experiments, or perhaps suggestions of appropriate approximations to be made in the analytic treatment of the problem. The large-scale use of Monte Carlo methods requires careful planning to avoid drowning in in enormous quantities of unintelligible results.

Two methods for **reduction of variance** have here been introduced: antithetic sequence of random numbers and the technique of using the same random numbers in corresponding situations. The latter technique is used when studying the changes in behavior of a system when a certain parameter is changed (e.g., the parameter k in Exercise 4). Many effective methods have been developed for reducing variance, e.g., **importance sampling** and **splitting techniques** (see

Hammersley and Handscomb [6]). Another important rule is that, *if a part of a problem can be treated with analytical or traditional numerical methods, then one should use such methods.* There are many ways to combine analytical methods and Monte Carlo methods.

Review Questions

1. What is a uniformly distributed random number?

2. Describe general methods for obtaining random numbers with given discrete or continuous distribution. Give examples of their use.

3. What are the most important properties of a Poisson process? How can one generate a Poisson process with the help of random numbers?

4. What is the mixed congruential method for generating pseudo-random numbers? What important difference is there between the numbers generated by this method and “genuine” random numbers?

Problems

1. (a) Let X, Y be independent uniform deviates (on the interval $[0, 1]$). Show that $P(X^2 + Y^2 \leq 1) = \pi/4$, and estimate this probability by a Monte Carlo experiment with (say) 1000 pairs of random numbers. For example, make graphical output like in the Buffon needle problem.

(b) Make an antithetic experiment, and take the average of the two results. Is the average better than one can expect if the second experiment had been independent of the first one.

(c) Estimate similarly the volume of the four-dimensional unit ball. If you have enough time, use more random numbers. (The exact volume of the whole unit ball is $\pi^2/2$.

2. Simulate (say) 360 throws with two usual dices. Denote the sum of the number of dots on the two dice in the n'th throw by Y_n, $2 \leq Y_n \leq 12$. Tabulate or draw a histogram, i.e., the (absolute) frequency of the occurrence of j dots versus j, $j = 2 : 12$. Make a conjecture about the true value of $P(Y_n = j)$. Try to confirm it by repeating the experiment with fresh uniform deviates. When you have found the right conjecture, it is not hard to prove it.

3. Write a program that uses uniform deviates (and perhaps uses the the formula $X = \text{ceil}(kU)$ for several values of k) to simulate a random “shuffle” of a deck of 52 cards. This is for a *numerical* game; do not spend time on drawing beautiful hearts, clubs etc.

4. Suppose we have a random number generator which generates random numbers uniformly distributed in $[-1, 1]$. To generate random points in the unit circle generate a pair of random numbers x_n, y_n. If $x_n^2 + y_n^2 < 1$, then accept (x_n, y_n); otherwise discard (x_n, y_n) and generate new random numbers.

(a) Use, e.g., the function rand in Matlab to generate points in the unit circle. Plot the accepted points.

(b) What is the probability that a pair of points is discarded? Use the observed number of discarded points in a run of 1000 points to approximate the value of π.

5. Repeat the simulation in Example 1.5.6 for $k = 1$ and $k = 2$ using the sequence of exponentially distributed numbers R

$$13 \quad 365 \quad 88 \quad 23 \quad 154 \quad 122 \quad 87 \quad 112 \quad 104 \quad 213\,,$$

antithetic to that used in Example 1.5.6. Compute the mean of the waiting times for the doctor and for all patients for this and the previous experiment.

1.6 Linear Algebra and Matrix Computations

In this section the basic elements of vector spaces and matrix algebra are recalled notations to be used in the rest of the text are introduced. For a more detailed expositions and proofs the reader is referred to Leon [9] or Strang [12].

1.6.1 Linear Vector Spaces

Let $\mathbf{K}$ be the field of real numbers $\mathbf{R}$ or complex numbers $\mathbf{C}$. A **vector space** over $\mathbf{K}$ is a set $\mathbf{V}$ of elements called **vectors** for which the operation addition and scalar multiplications are defined with the following properties:

1. addition is commutative and associative;
2. scalar multiplication is associative;
3. the following distributive properties hold:

$$\alpha(v+w) = \alpha v + \alpha w, \qquad (\alpha+\beta)v = \alpha v + \beta v,$$

for all $\alpha, \beta \in \mathbf{K}$ and $v, w \in \mathbf{V}$;

4. there is an element $0 \in \mathbf{V}$ called the **null vector** such that $v + 0 = v$ for all $v \in \mathbf{V}$;
5. for each vector v there exists a vector $-v$ such that $v + (-v) = 0$;
6. $0 \cdot v = 0$ and $1 \cdot v = v$ where 0 and 1 are the zero and unity in $\mathbf{K}$.

Familiar examples of a vector space are $\mathbf{V} = \mathbf{R}^n$ ($\mathbf{V} = \mathbf{C}^n$), i.e. the set of n-tuples, $1 \le n < \infty$, of real (complex) numbers. In approximation theory the vector space $\mathcal{P}_n$ of polynomials $p_n(x) = \sum_{k=0}^{n-1} a_k x^k$ of degree less than n plays an important role. Another example is $\mathbf{V} = C^p([a,b])$, the set of complex-valued functions which are continuous up to their pth derivatives $(0 \le p < \infty)$ on $[a,b]$.

If $W \subset V$ is a vector space then W is called a **vector subspace** of V. The set of all linear combinations of $v_1, \ldots, v_k \in V$ form a vector subspace denoted by

$$\operatorname{span}\{v_1,\ldots,v_k\} = \sum_{i=1}^{k} \alpha_i v_i, \quad \alpha_i \in \mathbf{K}, \quad i = 1:k.$$

If $S_1, \dots, S_k$ are vector subspaces of $\mathbf{V}$ then their sum $S = S_1 \cup S_2 \cdots \cup S_k$, defined by

$$S = \{v_1 + \cdots + v_k \mid v_i \in S_i,\ i = 1 : k\}$$

is also a vector subspace. The intersection T of a set of vector subspaces is also a subspace,

$$T = S_1 \cap S_2 \cdots \cap S_k.$$

If the intersection of the subspaces are empty, $S_i \cap S_j = 0$, $i \neq j$, then the sum of the subspaces is called their **direct sum** and denoted by

$$S = S_1 \oplus S_2 \cdots \oplus S_k.$$

A set of vectors $\{v_1, v_2, \dots, v_k\}$ in $\mathbf{V}$ is said to be **linearly independent** if

$$\sum_{i=1}^{k} c_i v_i = 0, \quad \Rightarrow \quad c_1 = c_2 = \cdots = c_k = 0.$$

Otherwise, if a nontrivial linear combination of $v_1, \dots, v_k$ is zero, the vectors are said to be linearly dependent. Then at least one of vector v_i will be a linear combination of the rest.

A **basis** in $\mathbf{V}$ is any set of linearly independent vectors $v_1, v_2, \dots, v_n \in \mathbf{V}$ such that all vectors $v \in \mathbf{V}$ can be uniquely decomposed as

$$v = \sum_{i=1}^{n} \xi_i v_i.$$

The scalars ξ_i are called the components or coordinates of v with respect to the basis $\{v_i\}$.

If the vector space $\mathbf{V}$ has a basis of k vectors, then every system of linearly independent vectors of $\mathbf{V}$ has at most k elements and any other basis of $\mathbf{V}$ has the same number k of elements. The number k is called the **dimension** of $\mathbf{V}$ and denoted by $\dim(\mathbf{V})$.

The **standard basis** for $\mathbf{C}^n$ is the set of unit vectors $e_1, e_2, ..., e_n$, where the jth component of e_i equals 1 if $j = i$, and 0 otherwise. We shall use the same name for a vector as for its coordinate representation by a column vector, with respect to the standard basis. For the the vector space $\mathcal{P}_n$ of polynomials of degree less the n monomials $1, x, \dots, x^{n-1}$ form a basis.

1.6.2 Matrix and Vector Algebra

A **matrix** A is a collection of $m \times n$ numbers ordered in m rows and n columns

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

We write $A \in \mathbf{R}^{m\times n}$, where $\mathbf{R}^{m\times n}$ denotes the set of all real $m \times n$ matrices. If $m = n$, then the matrix A is said to be square and of order n. If $m \neq n$, then A is said to be rectangular. A **column vector** is a matrix consisting of just one column and we write $x \in \mathbf{R}^m$ instead of $x \in \mathbf{R}^{m\times 1}$. Similarly a **row vector** is a matrix consisting of just one row.

A **linear map** from the vector space $\mathbf{C}^n$ to $\mathbf{C}^m$ is a function f such that

$$f(\alpha v + \beta w) = \alpha f(u) + \beta f(v)$$

for all $\alpha, \beta \in \mathbf{K}$ and $u, v \in \mathbf{C}^n$. Let x and y be the column vectors representing the vectors v and $f(v)$, respectively , using the standard basis of the two spaces. Then there is a unique matrix $A \in \mathbf{C}^{m\times n}$ representing this map such that

$$y = Ax.$$

This gives a link between linear maps and matrices.

We will follow a convention introduced by Householder[10] and use capital letters (e.g. A, B) to denote matrices. The corresponding lower case letters with subscripts ij then refer to the (i, j) component of the matrix (e.g. a_{ij}, b_{ij}). Greek letters $\alpha, \beta, \ldots$ are usually used to denote scalars. Column vectors are usually denoted by lower case letters (e.g. x, y).

Two matrices in $\mathbf{R}^{m\times n}$ are said to be **equal**, $A = B$, if

$$a_{ij} = b_{ij}, \quad i = 1 : m, \quad j = 1 : n.$$

The basic operations with matrices are defined as follows. The product of a matrix A with a scalar α is

$$B = \alpha A, \qquad b_{ij} = \alpha a_{ij}.$$

The **sum** of two matrices A and B in $\mathbf{R}^{m\times n}$ is

$$C = A + B, \qquad c_{ij} = a_{ij} + b_{ij}. \tag{1.6.1}$$

The **product** of two matrices A and B is defined if and only if the number of columns in A equals the number of rows in B. If $A \in \mathbf{R}^{m\times n}$ and $B \in \mathbf{R}^{n\times p}$ then

$$C = AB \in \mathbf{R}^{m\times p}, \qquad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \tag{1.6.2}$$

and can be computed with mnp multiplications. As a special case of the multiplication rule if $A \in \mathbf{R}^{m\times n}$, $x \in \mathbf{R}^n$ then

$$y = Ax \in \mathbf{R}^m, \qquad y_i = \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1 : m.$$

Matrix multiplication is *not commutative.* The product BA is defined only if $p = m$. Then the matrices $AB \in \mathbf{R}^{m\times m}$ and $BA \in \mathbf{R}^{n\times n}$ are both square, but if

[10] A. S. Householder 1904–1993, mathematician who pioneered the use of matrix factorization and orthogonal transformations in numerical linear algebra.

$m \neq n$ of different orders. In general, $AB \neq BA$ even when $m = n$. If $AB = BA$ the matrices are said to **commute**.

Matrix multiplication satisfies the rules

$$A(BC) = (AB)C, \qquad A(B + C) = AB + AC.$$

Note, however, that the number of arithmetic operations required to compute, respectively, the left- and right-hand sides of these equations can be very different! If $C \in \mathbf{R}^{p\times q}$ then computing the product ABC as $(AB)C$ requires $mp(n + q)$ operations whereas $A(BC)$ requires $nq(m + p)$ operations. For example, if A and B are square $n \times n$ matrices and x a column vector of length n then computing the product ABx as $(AB)x$ requires $n^3 + n^2$ operations whereas $A(Bx)$ only requires $2n^2$ operations. When $n \gg 1$ this makes a great difference!

The **transpose** A^T of a matrix $A = (a_{ij})$ is the matrix whose rows are the columns of A, i.e., if $C = A^T$ then $c_{ij} = a_{ji}$. For a complex matrix we denote by A^H the complex conjugate transpose of A

$$A = (a_{ij}), \qquad A^H = (\bar{a}_{ji}),$$

and it holds that $(AB)^H = B^H A^H$.

Row vectors are obtained by transposing column vectors (e.g. x^T, y^T). For the transpose of a product we have

$$(AB)^T = B^T A^T,$$

i.e., the product of the transposed matrices in *reverse order.*

The Euclidian **inner product** of two vectors x and y in $\mathbf{R}^n$ is given by

$$x^T y = \sum_{i=1}^{n} x_i y_i = y^T x.$$

In particular

$$x^T x = \sum_{i=1}^{n} |x_i|^2$$

is the Euclidian length of the vector x.

The **outer product** of $x \in \mathbf{R}^m$ and $y \in \mathbf{R}^n$ is the matrix

$$xy^T = \begin{pmatrix} x_1y_1 & \dots & x_1y_n \\ \vdots & & \vdots \\ x_my_1 & \dots & x_my_n \end{pmatrix} \in \mathbf{R}^{m\times n}.$$

For many problems it often is more relevant and convenient to work with complex vectors and matrices, i.e., the vector space $\mathbf{C}^{n\times m}$ of all complex $n \times m$ matrices whose components are complex numbers.[11]

[11]In MATLAB the only data type used is a matrix with either real or complex elements.

Most concepts introduced here carry over to complex matrices. Addition and multiplication of vectors and matrices follow the same rules as before. However, the complex, or Hermitian, inner product of two vectors x and y in $\mathbf{C}^n$ is defined by

$$x^H y = \sum_{k=1}^{n} \bar{x}_k y_k, \tag{1.6.3}$$

where $x^H = (\bar{x}_1, \dots, \bar{x}_n)$ and $\bar{x}_k$ denotes the complex conjugate of x_k. Hence $x^H y = \overline{y^H x}$.

It is useful to define **array operations**, which are carried out element-by-element on vectors and matrices. Following the convention in Matlab we denote array multiplication and division by $.*$ and $./$, respectively. If A and B have the same dimensions $A\,.*\,B$ is the matrix with elements equal to $a_{ij} \cdot b_{ij}$ and $A\,./B$ has elements a_{ij}/b_{ij}. (Note that for $+, -$ array operations coincides with matrix operations so no distinction is necessary.)

Any matrix D for which $d_{ij} = 0$ if $i \neq j$ is called a **diagonal matrix**. If $x \in \mathbf{R}^n$ is a vector then $D = \operatorname{diag}(x) \in \mathbf{R}^{n\times n}$ is the diagonal matrix formed by the elements of x. For a matrix $A \in \mathbf{R}^{n\times n}$ the elements a_{ii}, $i = 1 : n$ form the **main diagonal** of A, and we write

$$\operatorname{diag}(A) = \operatorname{diag}(a_{11}, a_{22}, \dots, a_{nn}).$$

For $k = 1 : n-1$ the elements $a_{i,i+k}$ ($a_{i+k,i}$), $i = 1 : n-k$ form the kth **super-diagonal** (**subdiagonal**) of A. The elements $a_{i,n-i+1}$, $i = 1 : n$ form the (main) **antidiagonal** of A.

The **unit matrix** $I_n \in \mathbf{R}^{n\times n}$ is defined by

$$I_n = \operatorname{diag}(1, 1, \dots, 1) = (e_1, e_2, \dots, e_n),$$

and the k-th column of I_n is denoted by e_k. We have that $I_n = (\delta_{ij})$, where δ_{ij} is the **Kronecker symbol** $\delta_{ij} = 0, i \neq j$, and $\delta_{ij} = 1, i = j$. For all square matrices of order n it holds $AI = IA = A$. If desirable, we set the size of the unit matrix as a subscript of I, e.g., I_n.

A matrix A for which all nonzero elements are located in consecutive diagonals is called a **band matrix**. A is said to have **upper bandwidth** r if r is the smallest integer such that

$$a_{ij} = 0, \quad j > i + r,$$

and similarly **lower bandwidth** s if r is the smallest integer such that

$$a_{ij} = 0, \quad i > j + s.$$

The number of nonzero elements in each row of A is then at most equal to $w = r + s + 1$, which is the **bandwidth** of A. For a matrix $A \in \mathbf{R}^{m\times n}$ which is not square we define the bandwidth as

$$w = \max_{1\leq i\leq m} \{j - k + 1 \mid a_{ij}a_{ik} \neq 0\}.$$

Several classes of band matrices that occur frequently have special names. Thus, a matrix for which $r = s = 1$ is called **tridiagonal**, if $r = 0$, $s = 1$ ($r = 1$, $s = 0$) it is called lower (upper) **bidiagonal** etc. A matrix with $s = 1$ ($r = 1$) is called an upper (lower) **Hessenberg** matrix.

An **upper triangular** matrix is a matrix R for which $r_{ij} = 0$ whenever $i > j$. A square upper triangular matrix has form

$$R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{nn} \end{pmatrix}.$$

If also $r_{ij} = 0$ when $i = j$ then R is **strictly** upper triangular. Similarly a matrix L is **lower triangular** if $l_{ij} = 0, i < j$, and strictly lower triangular if $l_{ij} = 0, i \le j$. Sums, products and inverses of square upper (lower) triangular matrices are again triangular matrices of the same type.

A square matrix A is called **symmetric** if its elements are symmetric about its main diagonal, i.e. $a_{ij} = a_{ji}$, or equivalently $A^T = A$. The product of two symmetric matrices is symmetric if and only if A and B commute, that is, $AB = BA$. If $A^T = -A$, then A is called **skew-symmetric**. A square matrix A is called **persymmetric** if it is symmetric about its antidiagonal, i.e., $a_{ij} = a_{n-j+1,n-i+1}$.

The **determinant** of a square matrix A is denoted by $\det(A)$. It can be defined by the expression

$$\det A = \sum_{\sigma \in S_n} \operatorname{sgn} \sigma \cdot a_{1,\sigma_1} \cdot a_{n,\sigma_n} \tag{1.6.4}$$

where the sum is over all permutations $\sigma \in S_n$ of the set $\{1, \dots, n\}$ and $\operatorname{sgn} \sigma$ is ± 1 according to whether σ is an even or odd permutation. If $\det A \neq 0$ the solution of the linear system $Ax = b$ can be expressed as

$$x_i = \det A_j / \det A, \quad i = 1 : n, \tag{1.6.5}$$

where A_j is the matrix A where the jth column has been replaced by the right hand side b. This expression is known as **Cramer's rule**.[12] Cramer's rule is useful for numerical computation only in very special cases, e.g., if $n = 2$.

Using the definition (1.6.4) to evaluate $\det A$ requires $n \cdot n!$ arithmetic operations. By the following three rules $\det(A)$ can be computed much more efficiently:

(i) The value of the determinant is unchanged if a row (column) multiplied by a scalar is added to another row (column).

(ii) The determinant of a triangular matrix equals the product of the elements in the main diagonal, i.e., if R is upper triangular

$$\det(R) = r_{11} r_{22} \cdots r_{nn}.$$

[12] Named after Gabriel Cramer 1704–1752.

(iii) If two rows (columns) are interchanged the value of the determinant is multiplied by (-1).

Obviously $\det(\alpha A) = \alpha^n \det(A)$. The following rules are also valid:

$$\det(A^T) = \det(A), \qquad \det(AB) = \det(A)\det(B).$$

A matrix is **nonsingular** if and only if $\det(A) \neq 0$. Otherwise the matrix is **singular**. Hence a triangular matrix is nonsingular if and only if all its diagonal elements are nonzero. If A is nonsingular then there exists an **inverse matrix** denoted by A^{-1} with the property that

$$A^{-1}A = AA^{-1} = I.$$

By A^{-T} we will denote the matrix $(A^{-1})^T = (A^T)^{-1}$. For the inverse of a product of two matrices we have

$$(AB)^{-1} = B^{-1}A^{-1},$$

where the product of the inverse matrices are taken in reverse order.

1.6.3 Partitioning and Block Matrices

It is often suitable to think of a matrix (vector) as being built up of matrices (vectors) of lower dimensions. This can be achieved by **partitioning** the matrix or vector into blocks. We write, e.g.,

$$A = \begin{array}{c} \\ p_1\{ \\ p_2\{ \\ \vdots \\ p_M\{ \end{array} \begin{array}{c} \begin{array}{cccc} q_1 & q_2 & \cdots & q_N \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1N} \\ A_{21} & A_{22} & \ldots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \ldots & A_{MN} \end{pmatrix} \end{array}, \qquad x = \begin{array}{c} p_1\{ \\ p_2\{ \\ \vdots \\ p_M\{ \end{array} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix} \tag{1.6.6}$$

where A_{IJ} is a matrix of dimension $p_I \times q_J$. We call such a matrix a **block matrix**. The partitioning can be carried out in many ways, and is often suggested by the structure of the underlying problem. For square matrices the most important partitionings are those for which $M = N$, and $p_I = q_I$. In this case the diagonal blocks $A_{II}, I = 1 : N$ are square matrices.

The great convenience of block matrices lies in the fact that the operations of addition and multiplication can be performed by treating the blocks A_{IJ} as *non-commuting scalars* and applying the definitions (1.6.1) and (1.6.2). Therefore many algorithms defined for matrices with scalar elements have another simple generalization to partitioned matrices. Of course the dimensions of the blocks must correspond in such a way that the operations can be performed. When this is the case, the matrices are said to be partitioned **conformally**.

Let $A = (A_{IK})$ and $B = (B_{KJ})$ be two block matrices of block dimensions $M \times N$ and $N \times P$ respectively, where the partitioning corresponding to the index K is the same for each matrix. Then we have $C = AB = (C_{IJ})$, where

$$C_{IJ} = \sum_{K=1}^{N} A_{IK}B_{KJ}, \quad 1 \leq I \leq M, \quad 1 \leq J \leq P.$$

For example, if $N = M = P = 2$ we have

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}.$$

Be careful to note the *order* of the factors in the products! In the special case of block upper triangular matrices this reduces to

$$\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} = \begin{pmatrix} R_{11}S_{11} & R_{11}S_{12} + R_{12}S_{22} \\ 0 & R_{22}S_{22} \end{pmatrix}.$$

Note that the product is again block upper triangular and its block diagonal equals the products of the diagonal blocks of the factors.

As a simple example consider the case when A and x are as in (1.6.6). Then the product $z = Ax$ is a block vector with blocks

$$z_I = \sum_{K=1}^{N} A_{IK} x_K, \quad I = 1 : M.$$

Often it is convenient to partition a matrix into rows or columns. In the special case when $M = 1$ and $A \in \mathbf{R}^{m\times n}$ we write

$$A = (a_1, a_2, \dots, a_n),$$

where $a_j \in \mathbf{R}^m$, $j = 1 : n$, is the j-th column of A. Similarly, when $N = 1$, we write

$$A = \begin{pmatrix} a_1^T \\ \vdots \\ a_m^T \end{pmatrix},$$

$a_i \in \mathbf{R}^n$, $i = 1 : m$, means that a_i^T is the i-th row of A. Let $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{n\times p}$. Then the matrix product $C = AB \in \mathbf{R}^{m\times p}$ can be written

$$C = AB = (a_1 \ a_2 \ \cdots \ a_n) \begin{pmatrix} b_1^T \\ b_2^T \\ \vdots \\ b_n^T \end{pmatrix} = \sum_{k=1}^{n} a_k b_k^T, \tag{1.6.7}$$

where $a_k \in \mathbf{R}^m$, $b_k \in \mathbf{R}^p$. Note that each term in the sum of (1.6.7) is an *outer product*. When the matrices A and B only have relatively few nonzero elements it can be shown that this is a more efficient way to compute their products than using the more common *inner product* formula (1.6.2) obtained from the partitioning

$$C = AB = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{pmatrix} (b_1 \ b_2 \ \cdots \ b_p) = (c_{ij}), \qquad c_{ij} = a_i^T b_j.$$

with $a_i, b_j \in \mathbf{R}^n$.

1.6.4 Inner Products, Orthogonality and Projections

An inner product on a vector space $\mathbf{V}$ defined over $\mathbf{K}$ is a continuous mapping $(\cdot,\cdot)$ from $\mathbf{V} \times \mathbf{V}$ onto $\mathbf{K}$ with the properties

1. $(v,v) > 0 \iff v \neq 0$;
2. $(v,w) = \overline{(w,v)}$;
3. $(\alpha u + \beta v, w) = \alpha(u,w) + \beta(v,w)$.

A vector space for which an inner product is defined is called an **inner product space**. We have already seen examples of an inner product space, namely $\mathbf{R}^n$ ($\mathbf{C}^n$) with the Euclidian inner product $(x,y) = x^T y$ ($(x,y) = x^H y$).

For a nonsingular linear transformation A which maps a vector space $\mathbf{V}$ onto $\mathbf{V}$ there is a unique **adjoint** transformation A^*, such that

$$(x, A^* y) = (Ax, y).$$

A matrix $A \in \mathbf{C}^{n\times n}$ is called **self-adjoint** if $A^* = A$.

For $A \in \mathbf{R}^{n\times n}$ with the Euclidian inner product we have

$$(Ax,y) = (Ax)^T y = x^T A^T y,$$

that is $A^* = A^T$, the transpose of A. Hence A is self-adjoint if A is symmetric. A symmetric matrix A is called **positive definite** if

$$x^T A x > 0, \quad \forall x \in \mathbf{R}^n, \quad x \neq 0. \tag{1.6.8}$$

and **non-negative definite** if $x^T A x \geq 0$, for all $x \in \mathbf{R}^n$. Otherwise it is called **indefinite**.

Similarly, $A \in \mathbf{C}^{n\times n}$ is self-adjoint or **Hermitian** if $A^H = A$, conjugate transpose of A. A Hermitian matrix has analogous properties to a real symmetric matrix. If A is Hermitian, then $(x^H A x)^H = x^H A x$ is real, and A is **positive definite** if

$$x^H A x > 0, \quad \forall x \in \mathbf{C}^n, \quad x \neq 0,$$

For the vector space $\mathbf{R}^n$ ($\mathbf{C}^n$) any inner product can be written as $(x,y) = y^T G x$ ($(x,y) = y^H G x$), where the matrix G is positive definite.

Any matrix $A \in \mathbf{C}^{n\times n}$ can be written as the sum of its Hermitian and a skew-Hermitian part, $A = H(A) + S(A)$, where

$$H(A) = \frac{1}{2}(A + A^H), \qquad S(A) = \frac{1}{2}(A - A^H).$$

A is Hermitian if and only if $S(A) = 0$. It is easily seen that A is positive definite if and only if its symmetric part $H(A)$ is positive definite.

Two vectors v and w in $\mathbf{R}^n$ are said to be **orthogonal** if $(v,w) = 0$. A set of vectors $v_1, \ldots, v_k$ in $\mathbf{R}^n$ is called orthogonal with respect to the Euclidian inner product if

$$v_i^T v_j = 0, \quad i \neq j,$$

and **orthonormal** if also $v_i^T v_i = 1, i = 1 : k$. An orthogonal set of vectors is linearly independent. More generally, a collection of subspaces $S_1, \dots, S_k$ of $\mathbf{R}^n$ are mutually orthogonal if

$$x^T y = 0, \quad x \in S_i, \quad y \in S_j, \quad i \neq j.$$

The **orthogonal complement** $S^\perp$ of a subspace $S \in \mathbf{R}^n$ is defined by

$$S^\perp = \{y \in \mathbf{R}^n |\; y^T x = 0,\; x \in S\}.$$

The vectors $q_1, \dots, q_k$ form an orthonormal basis for a subspace $S \subset \mathbf{R}^n$ if they are orthonormal and $\operatorname{span}\{q_1, \dots, q_k\} = S$. Such a basis can always be extended to a full orthonormal basis $q_1, \dots, q_n$ for $\mathbf{R}^n$, and then $S^\perp = \operatorname{span}\{q_{k+1}, \dots, q_n\}$.

Let $q_1, \dots, q_n \in \mathbf{R}^m$ be orthonormal and form the matrix $Q = (q_1, \dots, q_n) \in \mathbf{R}^{m\times n}$, $m \geq n$. Then the matrix Q is called orthogonal and $Q^T Q = I_n$. If $m = n$ then it follows that $Q^{-1} = Q^T$, and hence also $QQ^T = I_n$. Further we have $\det(Q)^2 = \det(Q^T Q) = 1$ and hence $|\det(Q)| = 1$.

Let the intersection of the two subspaces S_1 and $S_2 = 0$ be empty. Then any vector $x \in S$ can be decomposed in a unique way as

$$x = v_1 + v_2, \quad v_1 \in S_1, \quad v_2 \in S_2.$$

The transformation P_1 that maps x into v_1 is a linear transformation that is **idempotent**, i.e., $P_1^2 = P_1$. It is called a **projector** onto S_1 along S_2. Hence P_1 is a **projector** onto the subspace S_1 if and only if it holds:

$$(i)\;\; P_1 b = b \;\; \forall \;\; b \in S_1, \qquad (ii)\;\; P_1^2 = P_1. \tag{1.6.9}$$

The decomposition of an arbitrary vector $b \in \mathbf{R}^n$ can be written

$$b = P_1 b + (I - P_1)b = b_1 + b_2. \tag{1.6.10}$$

If it also holds that $(iii)\;\; P_1^T = P_1$, then for all $b \in \mathbf{R}^n$

$$P_1^T b_2 = P_1^T (I - P_1)b = (P_1 - P_1^2)b = 0,$$

and it follows that $b_2^T b = b_2^T P_1 b = 0$ for all $b \in S_1$. Hence $b_2 \perp S_1$, i.e., b_2 lies in the orthogonal complement $S^\perp$ of S_1; in particular $b_2 \perp b_1$. In this case P is the **orthogonal projector** onto S_1 and $P_2 = I - P_1$ the orthogonal projector onto $S_1^\perp$. It can be shown that the orthogonal projector P_1 is unique. Orthogonal projections play a central role in the study of least squares problems (see Chapter 8).

In the complex case, $A = (a_{ij}) \in \mathbf{C}^{m\times n}$ the Hermitian inner product leads to modifications in the definition of symmetric and orthogonal matrices Two vectors x and y in $\mathbf{C}^n$ are called orthogonal if $x^H y = 0$. A square matrix U for which $U^H U = I$ is called **unitary**. From (1.6.3) we find that

$$(Ux)^H Uy = x^H U^H Uy = x^H y.$$

Hence unitary matrices are characterized by the property that they preserve the Hermitian inner product. In particular the Euclidian length of a vector is invariant under unitary transformations, i.e., $\|Ux\|_2^2 = \|x\|_2^2$. Note that in every case, the new definition coincides with the old when the vectors and matrices are real.

1.6.5 Linear Least Squares Problems

Four **fundamental subspaces** are associated with a matrix $A \in \mathbf{R}^{m\times n}$. Two of them are the **range** of A

$$\mathcal{R}(A) = \{y \in \mathbf{R}^m |\ y = Ax,\ x \in \mathbf{R}^n\}, \tag{1.6.11}$$

and the **null space** of A^T

$$\mathcal{N}(A^T) = \{z \in \mathbf{R}^m |\ A^T z = 0\}. \tag{1.6.12}$$

The other two, the range $\mathcal{R}(A^T)$ of A^T and the null space $\mathcal{N}(A)$ of A are subspaces of $\mathbf{R}^n$ and defined analogously.

If $y \in \mathcal{R}(A)$ and $z \in \mathcal{N}(A^T)$ then $y^T z = x^T A^T z = 0$, i.e., y is orthogonal to z. It follows that $\mathcal{N}(A^T)$ is the orthogonal complement to $\mathcal{R}(A)$ in $\mathbf{R}^m$. Likewise $\mathcal{N}(A)$ is the orthogonal complement to $\mathcal{R}(A^T)$ in $\mathbf{R}^n$.

The linear system $Ax = b$, where $A \in \mathbf{R}^{m\times n}$ is said to be **consistent** if $b \in \mathcal{R}(A)$, or equivalently $\operatorname{rank}(A,\ b) = \operatorname{rank}(A)$. A consistent linear system always has at least one solution x; see Sec. 1.3.?. If $b \notin \mathcal{R}(A)$, or equivalently $\operatorname{rank}(A,\ b) > \operatorname{rank}(A)$ the system is inconsistent and has no solution. If $m > n$ the system is there always are right hand sides b such that $Ax = b$ is inconsistent.

For an inconsistent linear system $Ax = b$ there are many possible ways of defining a vector x, which in some sense "best" satisfies the system. A choice which can often be motivated for statistical reasons and also leads to a simple computational problem is to take x to be a vector which minimizes the Euclidian length of the **residual vector** $r = b - Ax$

$$\min_x \|b - Ax\|_2, \tag{1.6.13}$$

where we have used the notation

$$\|x\|_2 = (|x_1|^2 + \cdots + |x_n|^2)^{1/2} = (x^T x)^{1/2}.$$

for the Euclidian length of a vector x. We call (1.6.13) a **linear least squares problem** and any minimizer x a **least squares solution** of the system $Ax = b$.

The set of all solutions to problem (1.6.13) can the be characterized as follows:

Theorem 1.6.1.

The vector x minimizes $\|b - Ax\|_2$ if and only if the residual vector $r = b - Ax$ is orthogonal to $\mathcal{R}(A)$, or equivalently

$$A^T(b - Ax) = 0. \tag{1.6.14}$$

Proof. Let x be a vector for which $A^T(b - Ax) = 0$. Then for any $y \in \mathbf{R}^n$ $b - Ay = (b - Ax) + A(x - y)$. Squaring this and using (1.6.14) we obtain

$$\|b - Ay\|_2^2 = \|b - Ax\|_2^2 + \|A(x - y)\|_2^2 \geq \|b - Ax\|_2^2.$$

On the other hand assume that $A^T(b - Ax) = z \neq 0$. Then if $x - y = -\epsilon z$ we have for sufficiently small $\epsilon \neq 0$,

$$\|b - Ay\|_2^2 = \|b - Ax\|_2^2 - 2\epsilon\|z\|_2^2 + \epsilon^2\|Az\|_2^2 < \|b - Ax\|_2^2$$

so x does not minimize $\|b - Ax\|_2$. $\square$

Theorem 1.6.1 shows that any least squares solution x decomposes the right hand side b into two orthogonal components

$$b = Ax + r, \qquad r \perp Ax. \tag{1.6.15}$$

Here Ax is the orthogonal projection onto $\mathcal{R}(A)$ and $r \in \mathcal{N}(A^T)$; see Fig. 1.7.1. Note that although the least squares solution x may not be unique the decomposition (1.6.15) always is unique,

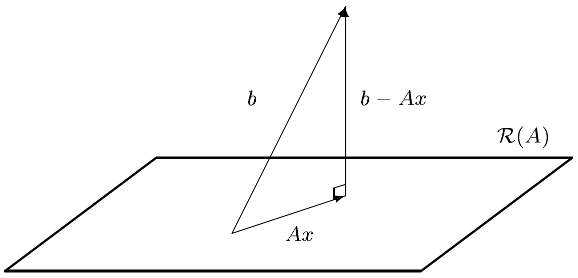


Figure 1.6.1. *Geometric characterization of the least squares solution.*

The above characterization of a least squares solution immediately leads to a classical method for solving the least squares problem (1.6.13). Multiplying in the factor A^T in (1.6.14) it follows that a least squares solution always satisfies the **normal equations**

$$A^TAx = A^Tb. \tag{1.6.16}$$

Here $A^TA \in \mathbf{R}^{n\times n}$ is a symmetric, positive semidefinite matrix. The normal equations are always *consistent* since

$$A^Tb \in \mathcal{R}(A^T) = \mathcal{R}(A^TA),$$

and therefore a least squares solution always exists.

We now give a condition for the least squares solution to be unique.

Theorem 1.6.2.

The matrix A^TA is positive definite if and only if the columns of A are linearly independent, i.e., when rank$(A) = n$. *In this case the least squares solution x is unique and given by*

$$x = (A^TA)^{-1}A^Tb. \tag{1.6.17}$$

Proof. If the columns of A are linearly independent, then $x \neq 0 \Rightarrow Ax \neq 0$. Therefore $x \neq 0 \Rightarrow x^T A^T A x = \|Ax\|_2^2 > 0$, and hence $A^T A$ is positive definite. On the other hand, if the columns are linearly dependent, then for some $x_0 \neq 0$ we have $Ax_0 = 0$. Then $x_0^T A^T A x_0 = 0$, and therefore $A^T A$ is not positive definite. When $A^T A$ is positive definite it is also nonsingular and (1.6.17) follows. □

In the full column rank case, rank $(A) = n$, the residual $r = b - Ax$ can be written

$$r = b - P_{\mathcal{R}(A)} b, \qquad P_{\mathcal{R}(A)} = A(A^T A)^{-1} A^T, \tag{1.6.18}$$

which gives an expression for $P_{\mathcal{R}(A)}$, the orthogonal projector onto $\mathcal{R}(A)$, the range space of A. It follows that any solution to the consistent linear system $Ax = P_{\mathcal{R}(A)} b$ is a least squares solution.

In more general least squares problems $Ax = b$ we can have rank $(A) < n$, and then A has a nontrivial nullspace. In this case if $\hat{x}$ is any vector that minimizes $\|Ax - b\|_2$, then the set of all least squares solutions is

$$\mathcal{S} = \{x = \hat{x} + y \mid y \in \mathcal{N}(A)\}. \tag{1.6.19}$$

In this set there is a unique solution of minimum norm characterized by $x \perp \mathcal{N}(A)$.

1.6.6 Similarity Transformations and Eigenvalues

Consider the linear transformation $y = Ax$, where $A \in \mathbf{R}^{n \times n}$. Let V be nonsingular and suppose we change basis by setting $x = V\xi$, $y = V\eta$, Then the column vectors ξ and η represents the vectors x and y with respect to the basis $V = (v_1, \ldots, v_n)$. Now $V\eta = AV\xi$, and hence

$$\eta = V^{-1} A V \xi,$$

which shows that the matrix $V^{-1}AV$ represents the operator A with respect to the basis V. The mapping $A \to V^{-1}AV$ is called a **similarity transformation** of the matrix A.

An **eigenvector** of A is a non-zero vector w that satisfies the equation,

$$Aw = \lambda w,$$

for some real or complex value λ that is called an **eigenvalue** of A. Equivalently we can write $(A - \lambda I)w = 0$, and the eigenvalues are therefore determined by the **characteristic equation**,

$$p_n(\lambda) = \det(A - \lambda I) = 0.$$

One can show that this is an algebraic equation of degree n, so counting multiplicities the matrix A has precisely n (possibly complex) eigenvalues. To each *distinct* eigenvalue λ_i there is at least one eigenvector w_i. The set of eigenvalues of a matrix is denoted by $\lambda(A)$ and called its **spectrum**. The largest modulus of an eigenvalue is called the **spectral radius** and denoted by

$$\rho(A) = \max_i |\lambda_i(A)|.$$

Note that if $Aw = \lambda w$, and we change basis, $w = V\hat{w}$, then $V^{-1}AV\hat{w} = \lambda\hat{w}$. This shows that $V^{-1}AV$ has the same eigenvalues as A, and the eigenvectors are $\hat{w}_i = V^{-1}w_i$. In other words: eigenvalues and eigenvectors are properties of the operator itself, independent of the basis used for its representation by a matrix.

The **trace** of a square matrix of order n is the sum of its diagonal elements

$$\operatorname{trace}(A) = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i.$$

The last equality follows using the relation between the coefficients and roots of the characteristic equation. Hence the trace of the matrix is invariant under similarity transformations.

Given $A \in \mathbf{C}^{n\times n}$ there exists a unitary matrix $U \in \mathbf{C}^{n\times n}$ such that

$$U^H AU = T = \begin{pmatrix} \lambda_1 & t_{12} & \dots & t_{1n} \\ & \lambda_2 & \dots & t_{2n} \\ & & \ddots & \vdots \\ & & & \lambda_n \end{pmatrix},$$

where T is upper triangular. This is the **Schur normal form** of A. Since $\det(T - \lambda I) = (\lambda_1 - \lambda)(\lambda_2 - \lambda)\cdots(\lambda_n - \lambda)$ the diagonal elements $\lambda_1, \cdots, \lambda_n$ of T are the eigenvalues of A.

A matrix $A \in \mathbf{C}^{\mathbf{n}\times\mathbf{n}}$ is said to be **normal** if $A^H A = AA^H$. It follows that for a normal matrix the upper triangular matrix T in the Schur normal form is normal, i.e.

$$T^H T = TT^H.$$

It can be shown that from this implies that all nondiagonal elements in T vanishes. Hence for a normal matrices the matrix T in the Schur normal form is diagonal. Then we have $AU = UT = U\Lambda$, where $\Lambda = \operatorname{diag}(\lambda_i)$, or with $U = (u_1, \dots, u_n)$,

$$Au_i = \lambda_i u_i, \quad i = i : n.$$

This shows the important result that a normal matrix always has a set of mutually unitary (orthogonal) eigenvectors.

Important classes of normal matrices are Hermitian ($A = A^H$), skew-Hermitian ($A^H = -A$), unitary ($A^{-1} = A^H$). Hermitian matrices have real eigenvalues, skew-Hermitian matrices have imaginary eigenvalues, and unitary matrices have eigenvalues on the unit circle.

Let $V = (v_1, \dots, v_n)$ be the eigenvectors and $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ the eigenvalues of a matrix A. Then, $Av_i = \lambda_i v_i$, $i = 1 : n$, or

$$AV = V\Lambda.$$

If the eigenvalues are linearly independent then V is non-singular, $\Lambda = V^{-1}AV$, and A is said to be **diagonalizable**. An example of a non-diagonalizable matrix are the matrices of the form

$$J_m(\lambda) = \begin{pmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{pmatrix} \in \mathbf{C}^{m\times m}.$$

The matrix $J_m(\lambda)$ is called a **Jordan block**. It has one eigenvalue λ of multiplicity m to which corresponds only one eigenvector $v_1 = e_1$.

1.6.7 The Singular Value Decomposition

Let $A \in \mathbf{R}^{m\times n}$ be a matrix of rank r. Then there is a decomposition of A into a product of three matrices

$$A = U\Sigma V^T, \qquad \Sigma = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \in \mathbf{R}^{m\times n}, \tag{1.6.20}$$

where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$ are orthogonal, $\Sigma_1 = \operatorname{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$, and

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0.$$

(Note that if $r = n$ and/or $r = m$, some of the zero submatrices in Σ disappear.) The σ_i are called the **singular values** of A and if we write

$$U = (u_1, \dots, u_m), \qquad V = (v_1, \dots, v_n),$$

the u_i, $i = 1 : m$, and v_i, $i = 1 : n$, are left and right **singular vectors**, respectively. The rank of A equals the number of nonzero singular values.

Similarly, for any complex matrix $A \in \mathbf{C}^{m\times n}$ we have the decomposition $A = U\Sigma V^H$, where U and V are unitary matrices and Σ a *real* diagonal matrix. (A proof of the singular value decomposition (SVD) will be given in Sec. 8.3.)

The SVD is of great theoretical and practical importance.[13] The geometrical significance of the SVD can be described as follows. The rectangular matrix A represents a mapping from $\mathbf{R}^n$ to $\mathbf{R}^m$. From the SVD it follows that there is an orthogonal basis in each of these two spaces, with respect to which this mapping is represented by the generalized diagonal matrix Σ. Note that transposing (1.6.20) we obtain the SVD of A^T,

$$A^T = V\Sigma^T U^T. \tag{1.6.21}$$

The singular values of A are uniquely determined. For any distinct singular value $\sigma_j \neq \sigma_i$, $i \neq j$, the corresponding singular vector v_j is unique (up to a factor ± 1). For multiple singular values, the corresponding singular vectors can be chosen as any orthonormal basis for the unique subspace that they span. Once the singular

[13]The SVD was independently published more than a century ago by Eugenio Beltrami 1873 and Camille Jordan 1874. Its use in numerical computations is much more recent.

vectors v_j, $1 \le j \le r$ have been chosen, the vectors u_j, $1 \le j \le r$ are uniquely determined, and vice versa, by

$$u_j = \frac{1}{\sigma_j} A v_j, \qquad v_j = \frac{1}{\sigma_j} A^T u_j \quad j = 1 : r. \tag{1.6.22}$$

If U and V are partitioned according to

$$U = (U_1,\ U_2), \quad U_1 \in \mathbf{R}^{m\times r}, \quad V = (V_1,\ V_2), \quad V_1 \in \mathbf{R}^{n\times r}. \tag{1.6.23}$$

then the SVD can be written in the more compact form

$$A = U_1 \Sigma_1 V_1^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T. \tag{1.6.24}$$

The last expression expresses A as a sum of r matrices of rank one.

The **pseudoinverse** of A is defined as

$$A^\dagger = V \Sigma^\dagger U^T, \qquad \Sigma^\dagger = \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbf{R}^{n\times m}, \tag{1.6.25}$$

The pseudoinverse solution of the linear system $Ax = b$ is

$$x = A^\dagger b = V \Sigma^\dagger U^T b$$

and equals the least squares solution of minimum Euclidian length.

The SVD gives complete information about the four fundamental subspaces associated with A. Using (1.6.20)–(1.6.21) it is easy to verify that the range and nullspace of A and A^T are given by

$$\mathcal{R}(A) = \mathcal{R}(U_1) \qquad \mathcal{N}(A^T) = \mathcal{R}(U_2) \tag{1.6.26}$$

$$\mathcal{R}(A^T) = \mathcal{R}(V_1) \qquad \mathcal{N}(A) = \mathcal{R}(V_2). \tag{1.6.27}$$

Hence we immediately find the well-known relations

$$\mathcal{R}(A)^\perp = \mathcal{N}(A^T), \qquad \mathcal{N}(A)^\perp = \mathcal{R}(A^T),$$

In general we have

$$\dim \mathcal{R}(A) = \dim \mathcal{R}(A^T), \quad \dim \mathcal{N}(A) = n - r, \qquad \dim \mathcal{N}(A^T) = m - r,$$

where $r = \operatorname{rank}(A)$.

The rank r equals the maximum number of independent row or column vectors of A, and thus $r \le \min(m, n)$. If $\operatorname{rank}(A) = n$ we say that A has full **column rank**. If $\operatorname{rank}(A) = m$, then A is said to have full **row rank**. A square matrix $A \in \mathbf{R}^{n\times n}$ is nonsingular if and only if $\mathcal{N}(A) = \{0\}$.

If $S = \operatorname{span}(U)$ and $U = (u_1, \dots, u_k)$ is orthogonal, $U^T U = I$, then it is easily seen that the orthogonal projector onto S can be written $P = UU^T$. Similarly the orthogonal projectors onto the four fundamental subspaces of A can be expressed in terms of the singular vectors of A as

$$P_{\mathcal{R}(A)} = AA^\dagger = U_1 U_1^T, \qquad P_{\mathcal{N}(A^T)} = U_2 U_2^T, \tag{1.6.28}$$

$$P_{\mathcal{R}(A^T)} = A^T (A^T)^\dagger = V_1 V_1^T, \qquad P_{\mathcal{N}(A)} = V_2 V_2^T.$$

1.6.8 Norms of Vectors and Matrices

In many applications it is useful to have a measure of the size of a vector or a matrix. An example is the quantitative discussion of errors in matrix computation. Such measures are provided by vector and matrix norms, which can be regarded as generalizations of the absolute value function on $\mathbf{R}$.

A **norm** on a vector space $\mathbf{V}$ over $\mathbf{C}$ is a function $\mathbf{V} \to \mathbf{R}$ denoted by $\|\cdot\|$ that satisfies the following three conditions:

1. $\|x\| > 0, \quad \forall x \in \mathbf{V}, \quad x \neq 0$ (definiteness)
2. $\|\alpha x\| = |\alpha|\,\|x\|, \quad \forall \alpha \in \mathbf{C}, \quad x \in \mathbf{C}^n$ (homogeneity)
3. $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in \mathbf{V}$ (triangle inequality)

The triangle inequality is often used in the form (see Problem 11) $\|x \pm y\| \geq \big|\, \|x\| - \|y\| \,\big|$.

The most common vector norms are special cases of the family of **Hölder** norms or l_p-norms (see Sec. 4.1.3)

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}, \qquad 1 \leq p < \infty. \tag{1.6.29}$$

The l_p-norms have the property that $\|x\|_p = \|\,|x|\,\|_p$. Vector norms with this property are said to be **absolute**. The three most important particular cases are $p = 1, 2$ and the limit when $p \to \infty$:

$$\begin{aligned} \|x\|_1 &= |x_1| + \cdots + |x_n|, \\ \|x\|_2 &= (|x_1|^2 + \cdots + |x_n|^2)^{1/2} = (x^H x)^{1/2}, \\ \|x\|_\infty &= \max_{1 \leq i \leq n} |x_i|. \end{aligned} \tag{1.6.30}$$

The vector 2-norm is also called the Euclidean norm. It is invariant under unitary (orthogonal) transformations since

$$\|Qx\|_2^2 = x^H Q^H Q x = x^H x = \|x\|_2^2$$

if Q is orthogonal.

Another important property of the l_p-norms is the **Hölder inequality**

$$|x^H y| \leq \|x\|_p \|y\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1, \quad p \geq 1. \tag{1.6.31}$$

For $p = q = 2$ this becomes the **Cauchy–Schwarz inequality**

$$|x^H y| \leq \|x\|_2 \|y\|_2.$$

Norms can be obtained from inner products by tasking

$$\|x\|^2 = (x, x) = x^H G x,$$

where G is Hermitian and positive definite. It can be shown that the unit ball $\{x : \|x\| \le 1\}$ corresponding to this norm is an ellipsoid, and hence they are also called elliptic norms. A special case that frequently is useful is the **scaled** l_p-norms defined by

$$\|x\|_{p,D} = \|Dx\|_p, \quad D = \operatorname{diag}(d_1, \dots, d_n), \quad d_i \ne 0, \quad i = 1 : n. \tag{1.6.32}$$

All norms on $\mathbf{C}^n$ are equivalent in the following sense: For each pair of norms $\|\cdot\|$ and $\|\cdot\|'$ there are positive constants c and c' such that

$$\frac{1}{c}\|x\|' \le \|x\| \le c'\|x\|', \quad \forall x \in \mathbf{C}^n. \tag{1.6.33}$$

In particular it can be shown that for the ℓ_p-norms we have

$$\|x\|_q \le \|x\|_p \le n^{\left(\frac{1}{p}-\frac{1}{q}\right)}\|x\|_q, \quad 1 \le p \le q \le \infty. \tag{1.6.34}$$

We now consider **matrix norms**. We can construct a matrix norm from a vector norm by defining

$$\|A\| = \sup_{x \ne 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|. \tag{1.6.35}$$

This norm is called the **operator norm**, or the matrix norm **subordinate** to the vector norm. From the definition it follows directly that

$$\|Ax\| \le \|A\| \, \|x\|, \qquad x \in \mathbf{C}^n.$$

Whenever this inequality holds, we say that the matrix norm is **consistent** with the vector norm.

It is an easy exercise to show that operator norms are **submultiplicative**, i.e., whenever the product AB is defined it satisfies the condition

4. $N(AB) \le N(A)N(B)$

The matrix norms

$$\|A\|_p = \sup_{\|x\|=1} \|Ax\|_p, \quad p = 1, 2, \infty,$$

subordinate to the vector p-norms are especially important. For these it holds that $\|I_n\|_p = 1$. The 1-norm and ∞-norm are easily computable from

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|, \qquad \|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|, \tag{1.6.36}$$

respectively. Note that $\|A\|_1 = \|A^H\|_\infty$.

The 2-norm is also called the **spectral norm**. Its major drawback is that it is expensive to compute. We have

$$\|A\|_2 = \sup_{\|x\|=1} (x^H A^H A x)^{1/2} = \sigma_1(A), \tag{1.6.37}$$

where $\sigma_1(A)$ is the largest singular value of A. Since the nonzero eigenvalues of A^HA and AA^H are the same it follows that $\|A\|_2 = \|A^H\|_2$. A useful upper bound for the matrix 2-norm is

$$\|A\|_2 \le (\|A\|_1\|A\|_\infty)^{1/2}. \tag{1.6.38}$$

The proof of this bound is given as an exercise in Problem 16.

Another way to proceed in defining norms for matrices is to regard $\mathbf{C}^{m\times n}$ as an mn-dimensional vector space and apply a vector norm over that space. With the exception of the **Frobenius norm** derived from the vector 2-norm

$$\|A\|_F = \Big(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\Big)^{1/2} \tag{1.6.39}$$

such norms are not much used. Note that $\|A^H\|_F = \|A\|_F$. Useful alternative characterizations of the Frobenius norm are

$$\|A\|_F^2 = \mathrm{trace}\,(A^HA) = \sum_{i=1}^{k}\sigma_i^2(A), \quad k = \min(m,n), \tag{1.6.40}$$

where $\sigma_i(A)$ are the nonzero singular values of A. The Frobenius norm is submultiplicative. However, it is often larger than necessary; e.g., $\|I_n\|_F = n^{1/2}$. This tends to make bounds derived in terms of the Frobenius norm not as sharp as they might be. From (1.6.40) we also get lower and upper bounds for the matrix 2-norm

$$\frac{1}{\sqrt{n}}\|A\|_F \le \|A\|_2 \le \|A\|_F.$$

An important property of the Frobenius norm and the 2-norm is that both are invariant with respect to orthogonal transformations, i.e. for all orthogonal matrices Q and P ($Q^HQ = I$, and $P^HP = I$) of appropriate dimensions

$$\|QAP^H\| = \|A\|.$$

We finally remark that the 1-,∞- and the Frobenius norm satisfy

$$\|\,|A|\,\| = \|A\|, \qquad |A| = (|a_{ij}|),$$

but for the 2-norm the best result is that $\|\,|A|\,\|_2 \le n^{1/2}\|A\|_2$. The vector and matrix norms defined in this section can immediately be extended to complex vectors and matrices.

One use of norms is the study of *limits of sequences of vectors and matrices* (see Sec. 9.2.4). Consider an infinite sequence $x_1, x_2, \dots$ of elements of a vector space $\mathbf{V}$ and let $\|\cdot\|$ be a norm on $\mathbf{V}$. The sequence is said to converge (strongly if V is infinite dimensional) to a limit $x \in \mathbf{V}$, and we write $\lim_{k\to\infty} x_k = x$ if

$$\lim_{k\to\infty}\|x_k - x\| = 0,$$

For a finite dimensional vector space the equivalence of norms (1.6.33) shows that convergence is independent of the choice of norm. The particular choice $\|\cdot\|_\infty$ shows that convergence of vectors in $\mathbf{C}^n$ is equivalent to convergence of the n sequences of scalars formed by the components of the vectors. By considering matrices in $\mathbf{C}^{m\times n}$ as vectors in $\mathbf{C}^{mn}$ the same conclusion holds for matrices. For an excellent survey of vector and matrix norms we refer to Stewart and Sun [13, Ch. II].

Review Questions

1. Define the concepts:

(i) Real symmetric matrix. (ii) Real orthogonal matrix.

(iii) Real skew-symmetric matrix. (iv) Triangular matrix.

(v) Hessenberg matrix.

2. (a) Give conditions for a matrix P to be the orthogonal projector onto a subspace $S \in \mathbf{R}^n$.

(b) Define the orthogonal complement of S in $\mathbf{R}^n$.

3. What is the Schur normal form of a matrix $A \in \mathbf{C}^{n\times n}$?

(b)What is meant by a normal matrix? How does the Schur form simplify for a normal matrix?

4. (a) Show that $A^\dagger = A^{-1}$ when A is a nonsingular matrix.

(b) Construct an example where $G \neq A^\dagger$ despite the fact that $GA = I$.

5. (a) Construct an example where $(AB)^\dagger \neq B^\dagger A^\dagger$.

(b) Show that if A is an $m \times r$ matrix, B is an $r \times n$ matrix, and $\operatorname{rank}(A) = \operatorname{rank}(B) = r$, then $(AB)^\dagger = B^\dagger A^\dagger$.

6. Show, using the SVD, that $P_{\mathcal{R}(A)} = AA^\dagger$ and $P_{\mathcal{R}(A^T)} = A^\dagger A$.

7. Define the matrix subordinate norm to a given vector norm.

8. Define the l_p norm of a vector x. Give explicit expressions for the matrix l_p norms for $p = 1, 2, \infty$. Show that

$$\frac{1}{n}\|x\|_1 \leq \frac{1}{\sqrt{n}}\|x\|_2 \leq \|x\|_\infty.$$

which are special cases of (1.6.34).

9. Show that for any consistent matrix norm $\|\cdot\|$ it holds that $\rho(A) \leq \|A\|$, where $\rho(A)$ is the spectral radius of A.

Problems

1. Show that if $A, B \in \mathbf{R}^{n\times n}$ are both symmetric and persymmetric, then $AB + BA$ also has this property.

2. Let $A \in \mathbf{R}^{m\times n}$ have rows a_i^T, i.e., $A^T = (a_1, \ldots, a_m)$. Show that

$$A^T A = \sum_{i=1}^{m} a_i a_i^T.$$

What is the corresponding expression for $A^T A$ if A is instead partitioned into columns?

3. (a) Let $A, B \in \mathbf{R}^{n\times n}$ have lower bandwidth r and s respectively. Show that the product AB has lower bandwidth $r+s$.

(b) An upper Hessenberg matrix H is a matrix with lower bandwidth $r=1$. Using the result in (a) deduce that the product of H and an upper triangular matrix is again an upper Hessenberg matrix.

(c) Show that if $R \in \mathbf{R}^{n\times n}$ is strictly upper triangular, then $R^n = 0$.

4. To solve a linear system $Ax = b$, where $A \in \mathbf{R}^n$, by Cramer's rule (see Equation (1.6.5)) requires the evaluation of $n+1$ determinants of order n. Estimate the number of multiplications needed for $n = 50$ if the determinants are evaluated in the naive way. Estimate the time it will take on a computer performing 10^9 floating point operations per second!

5 (a) Show that if $w \in \mathbf{R}^n$ and $w^T w = 1$, then the matrix $P(w) = I - 2ww^T$ is both symmetric and orthogonal.

(b) Given two vectors $x, y \in \mathbf{R}^n$, $x \neq y$, $\|x\|_2 = \|y\|_2$, then

$$P(w)x = y, \qquad w = (y-x)/\|y-x\|_2.$$

6. Show that if the complex matrix $U = Q_1 + iQ_2$ is unitary, then the real matrix

$$\tilde{U} = \begin{pmatrix} Q_1 & -Q_2 \\ Q_2 & Q_1 \end{pmatrix}$$

is orthogonal.

7. Let $A \in \mathbf{R}^{n\times n}$ be a given matrix. Show that if $Ax = y$ has *at least one* solution for any $y \in \mathbf{R}^n$, then it has *exactly one* solution for any $y \in \mathbf{R}^n$. (This is a useful formulation for showing uniqueness of approximation formulas.)

8. Show that for $x \in \mathbf{R}^n$,

$$\lim_{p\to\infty} \|x\|_p = \max_{1\le i\le n} |x_i|.$$

9. Prove the following inequalities are valid and best possible:

$$\|x\|_2 \le \|x\|_1 \le n^{1/2}\|x\|_2, \qquad \|x\|_\infty \le \|x\|_1 \le n\|x\|_\infty.$$

Derive similar inequalities for the comparison of the operator norms $\|A\|_1$,$\|A\|_2$,and $\|A\|_\infty$.

10. Show that any vector norm is uniformly continuous by proving the inequality

$$|\,\|x\| - \|y\|\,| \le \|x-y\|, \qquad x, y \in \mathbf{R}^n.$$

11. Show that for any matrix norm there exists a consistent vector norm.

Hint: Take $\|x\| = \|xy^T\|$ for any vector $y \in \mathbf{R}^n$, $y \neq 0$.

12. Derive the formula for $\|A\|_\infty$ given in (1.6.36).

13. Show that for any subordinate matrix norm

$$\|A+B\| \le \|A\| + \|B\|, \qquad \|AB\| \le \|A\|\|B\|.$$

14. Show that $\|A\|_2 = \|PAQ\|_2$ if P and Q are orthogonal matrices.

15. Use the result $\|A\|_2^2 = \rho(A^T A) \le \|A^T A\|$, valid for any matrix operator norm $\|\cdot\|$, where $\rho(A^T A)$ denotes the spectral radius of $A^T A$, to deduce the upper bound in (1.6.38).

16. (a) Let T be a nonsingular matrix, and let $\|\cdot\|$ be a given vector norm. Show that the function $N(x) = \|Tx\|$ is a vector norm.

(b) What is the matrix norm subordinate to $N(x)$?

(c) If $N(x) = \max_i |k_i x_i|$, what is the subordinate matrix norm?

17. Let $B \in \mathbf{R}^{n\times n}$ be a matrix for which $\|B\| < 1$. Show that the infinite sequence and product

$$(I-B)^{-1} = \begin{cases} I + B + B^2 + B^3 + B^4 \cdots, \\ (I+B)(I+B^2)(I+B^4)(I+B^8)\cdots \end{cases}$$

both converge to the indicated limit.

Hint: Use the identity $(I + B + \cdots + B^k)(I - B) = I - B^{k+1}$.

(b) Show that the matrix $(I-B)$ is nonsingular and that $\|(I-B)^{-1}\| \le 1/(1-\|B\|)$.

Notes and References

A good paper explaining to a mathematical audience problems inherent in numerical computations is Forsythe [3, 1970], The paper by Fox [5, 1971] gives numerous examples in which incorrect answers are obtained from plausible numerical methods.

For a summary of the theory and current practice of random number generators, see Knuth [8, 1981] and [10, 1988]. Press et al.[11, 1997, *Ch.*7] gives an up to date survey reflecting recent progress in random number generators.

[1] N. T. J. Bailey. A study of queues and appointment systems in hospital outpatient departments, with special reference to waiting times. *J. Roy. Stat. Soc.*, 3:14:185ff, 1951.

[2] RAND Corporation. *A Million Random Digits and 100,000 Normal Deviates.* Free Press, Glencoe, IL, 1955.

[3] G. E. Forsythe. Pitfalls in computation, or why a math book isn't enough. Technical Report CS 147, Computer Science Department, Stanford University, Stanford, CA, 1970.

[4] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Method for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[5] L. Fox. How to get meaningless answers in scientific computation (and what to do about it),. *IMA Bulletin*, 7:10:296–302, 1971.

[6] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* Methuen, London, UK, 1964.

[7] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, 2002.

[8] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* Addison-Wesley, Reading, MA, second edition, 1981.

[9] S. J. Leon. *Linear Algebra with Applications.* Macmillan, New York, fourth edition, 1994.

[10] S. K. Park and K. W. Miller. Random number generators: good ones are hard to find. *Comm. ACM*, 22:1192–1201, 1988.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran 77; The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, second edition, 1993.

[12] G. Strang. *Linear Algebra and Its Applications.* Academic Press, New York, third edition, 1988.

[13] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.

Chapter 2

How to Obtain and Estimate Accuracy

2.1 Basic Concepts in Error Estimation

[1]

2.1.1 Sources of Error

The main purpose of numerical analysis and scientific computing is to develop efficient and accurate methods to compute approximations to quantities that are difficult or impossible to obtain by analytic means. However, numerical analysts must also be experts at controlling different sources of errors so that these will not interfere with the computed results.

Numerical results are affected by many types of errors. Some sources of error are difficult to influence; others can be reduced or even eliminated by, for example, rewriting formulas or making other changes in the computational sequence.

Errors are propagated from their sources to quantities computed later, sometimes with a considerable amplification or damping. It is important to distinguish between the new error produced at the computation of a quantity (a source error), and the error inherited (propagated) from the data that the quantity depends on.

A. *Errors in Given Input Data.* Input data can be the result of measurements which have been influenced by systematic errors or by temporary disturbances. A **rounding error** occurs, for example, whenever an irrational number is shortened ("rounded off") to a fixed number of decimals. It can also occur when a decimal fraction is converted to the form used in the computer.

B. *Rounding Errors During the Computations.* The limitation of floating point numbers in a computer leads at times to a loss of information that, depending on the context, may or may not be important. Two typical cases are:

(i) If the computer cannot handle numbers which have more than, say, s digits, then the exact product of two s-digit numbers (which contains $2s$ or $2s-1$ digits) cannot be used in subsequent calculations; the product must be rounded off.

[1]This section last revised by Åke Björck 2003 02 24.

(ii) If, in a floating point computation, a relatively small term b is added to a, then some digits of b are "shifted out" (see Example 2.2.4), and they will not have any effect on future quantities that depend on the value of $a + b$.

The effect of such roundings can be quite noticeable in an extensive calculation, or in an algorithm which is numerically unstable (see Example 1.3.4).

C. *Truncation Errors.* These are errors committed when a limiting process is truncated (broken off) before one has come to the limiting value. A **truncation error** occurs, for example, when an infinite series is broken off after a finite number of terms, or when a derivative is approximated with a difference quotient (although in this case the term **discretization error** is better). Another example is when a nonlinear function is approximated with a linear function as in Newton's method. Observe the distinction between truncation error and rounding error.

D. *Simplifications in the Mathematical Model.* In most of the applications of mathematics, one makes idealizations. In a mechanical problem, for example, one might assume that a string in a pendulum has zero mass. In many other types of problems it is advantageous to consider a given body to be homogeneously filled with matter, instead of being built up of atoms. For a calculation in economics, one might assume that the rate of interest is constant over a given period of time. The effects of such sources of error are usually more difficult to estimate than the types named in A, B, and C.

E. *"Human" Errors and Machine Errors.* In all numerical work, one must expect that clerical errors, errors in hand calculation, and misunderstandings will occur. One should even be aware that textbooks (!), tables and formulas may contain errors. When one uses computers, one can expect errors in the program itself, typing errors in entering the data, operator errors, and (more seldom).

Errors which are purely machine errors are responsible for only a very small part of the strange results which (occasionally with great publicity) are produced by computers. Most of the errors depend on the so-called human factor. As a rule, the effect of this type of error source cannot be analyzed with the help of the theoretical considerations of this chapter! We take up these sources of error in order to emphasize that both the person who carries out a calculation and the person who guides the work of others can plan so that such sources of error are not damaging. One can reduce the risk for such errors by suitable adjustments in working conditions and routines. Stress and tiredness are common causes of such errors.

Intermediate results that may reveal errors in a computation are not visible when using a computer. Hence the user must be able to verify the correctness of his results or be able to prove that his process cannot fail! Therefore one should carefully consider what kind of checks can be made, either in the final result or in certain stages of the work, to prevent the necessity of redoing a whole project for the sake of a small error in an early stage. One can often discover whether calculated values are of the wrong order of magnitude or are not sufficiently regular, for example using difference checks (see Section 4.5).

Occasionally one can check the credibility of several results at the same time by checking that certain relations are true. In linear problems, one often has the possibility of sum checks. In physical problems, one can check, for example, to see whether energy is conserved, although because of the error sources A–D one cannot expect that it will be exactly conserved. In some situations, it can be best to treat a problem in two independent ways, although one can usually (as intimated above) check a result with less work than this.

Errors of type E do occur, sometimes with serious consequences. For example, the first American Venus probe was lost due to a program fault caused by the inadvertent substitution of a statement in a Fortran program of the form `DO 3 I = 1.3` for one of the form `DO 3 I = 1,3`. A hardware error that got much publicity surfaced in 1994, when it was found that the INTEL Pentium processor gave wrong results for division with floating point numbers of certain patterns. This was discovered by a mathematician doing research on prime numbers.

From a different point of view, one may distinguish between controllable and uncontrollable (or unavoidable) error sources. Errors of type A and D are usually considered to be uncontrollable in the numerical treatment (although a feedback to the constructor of the mathematical model may sometimes be useful). Errors of type C are usually controllable. For example, the number of iterations in the solution of an algebraic equation, or the step size in a simulation can be chosen, either directly or by setting a tolerance, see Sec. 1.4.1.

The rounding error in the individual arithmetic operation (type B) is, in a computer, controllable only to a limited extent, mainly through the choice between single and double precision. A very important fact is, however, that it can often be controlled by appropriate rewriting of formulas or by other changes of the algorithm, see, e.g., Example 2.3.4.

If it doesn't cost too much, a controllable error source should be controlled so that its effects are evidently negligible, for example compared to the effects of the uncontrollable sources. A reasonable interpretation of "full accuracy" is that the controllable error sources should not increase the error of a result more than about 20%. Sometimes, "full accuracy" may be expensive, for example in terms of computing time, memory space or programming efforts. Then it becomes important to estimate the relation between accuracy and these cost factors. One goal of the rest of this chapter is to introduce concepts and techniques useful to this purpose.

We strongly encourage the reader to use quality library programs when possible, since a lot of experience and profound theoretical analysis has often been built into these (sometimes far beyond the scope of this text). It is not practical to "reinvent the wheel"! Nevertheless, many real-word problems contain some non-standard features, where understanding the general principles of numerical methods can save much time in the preparation of a program as well as in in the computer runs.

2.1.2 Absolute and Relative Errors

Approximation is a central concept in almost all the uses of mathematics. One must often be satisfied with approximate values of the quantities with which one works.

Another type of approximation occurs when one ignores some quantities which are small compared to others. Such approximations are often necessary to insure that the mathematical and numerical treatment of a problem does not become hopelessly complicated.

We make the following definition.

Definition 2.1.1.

Let $\tilde{x}$ be an approximate value whose exact value is x. Then the **absolute error** *in $\tilde{x}$ is:*

$$\Delta x = \tilde{x} - x,$$

and if $x \neq 0$ the **relative error** *is:*

$$\Delta x / x = (\tilde{x} - x)/x.$$

In some books the error is defined with the opposite sign to that we use here. It makes almost no difference which convention one uses, as long as one is consistent. Using our definition $x - \tilde{x}$ is the correction which should be *added* to $\tilde{x}$ to get rid of the error. The correction and the error have then the same magnitude but different sign.

It is important to distinguish between the error $\tilde{x} - x$, which can be positive or negative, and a *bound* for the magnitude of the error. In many situations one wants to compute strict or approximate **error bounds** for the absolute or relative error. Since it is sometimes rather hard to obtain an error bound that is both strict and sharp, one sometimes prefers to use less strict but often realistic **error estimates**. These can be based on the first neglected term in some expansion or some other asymptotic considerations.

The notation $x = \tilde{x} \pm \epsilon$ means, in this book, $|\tilde{x} - x| \leq \epsilon$. For example, if $x = 0.5876 \pm 0.0014$ then $0.5862 \leq x \leq 0.5890$, and $|\tilde{x} - x| \leq 0.0014$. In other texts, the same plus-minus notation is sometimes used for the "standard error" (see Sec. 2.3.4) or some other measure of deviation of a statistical nature. If x is a vector $\|\cdot\|$ then the error bound and the relative error bound may be defined as bounds for

$$\|\tilde{x} - x\| \quad \text{and} \quad \|\tilde{x} - x\|/\|x\|,$$

respectively, where $\|\cdot\|$ denotes some vector norm (see Sec. 1.6.8). A bound $\|\tilde{x} - x\|/\|x\| \leq 1/2 \cdot 10^{-p}$ then implies that components $\tilde{x}_i$ with $|\tilde{x}_i| \approx \|x\|$ have about p significant digits but this is not true for components of smaller absolute value. An alternative is to use componentwise relative errors, e.g.,

$$\max_i |\tilde{x}_i - x_i|/|x_i|, \tag{2.1.1}$$

but this assumes that $x_i \neq 0$, $\forall i$.

We will distinguish between the terms accuracy and precision. By **accuracy** we mean the absolute or relative error of an approximate quantity. The term **precision** will be reserved for the accuracy with which the basic arithmetic operations $+, -, *, /$ are performed. For floating point operations this is given by the unit roundoff; see (2.2.8).

Numerical results which are not followed by any error estimations should often, though not always, be considered as having an uncertainty of $\frac{1}{2}$ a unit in the last decimal place. In presenting numerical results, it is a good habit, if one does not want to go to the difficulty of presenting an error estimate with each result, to give explanatory remarks such as:

- "All the digits given are thought to be significant."
- "The data has an uncertainty of at most 3 units in the last digit."
- "For an ideal two-atomed gas, $c_P/c_V = 1.4$ (exactly)."

We shall also introduce some notations, useful in practice, though their definitions are not exact in a mathematical sense:

> $a \ll b$ ($a \gg b$) is read: "a is much smaller (much greater) than b". What is meant by "much smaller"(or "much greater") depends on the context—among other things, on the desired precision.
>
> $a \approx b$ is read: "a is approximately equal to b" and means the same as $|a - b| \ll c$, where c is chosen appropriate to the context. We *cannot generally* say, for example, that $10^{-6} \approx 0$.
>
> $a \lessapprox b$ (or $b \gtrapprox a$) is read: "a is less than or approximately equal to b" and means the same as "$a \leq b$ or $a \approx b$."

Occasionally we shall have use for the following more precisely defined mathematical concepts:

> $f(x) = O(g(x))$, $x \to a$, means that $|f(x)/g(x)|$ is bounded as $x \to a$ (a can be finite, $+\infty$, or $-\infty$).
>
> $f(x) = o(g(x))$, $x \to a$, means that $\lim_{x\to a} f(x)/g(x) = 0$.
>
> $f(x) \sim g(x)$, $x \to a$, means that $\lim_{x\to a} f(x)/g(x) = 1$.

2.1.3 Rounding and Chopping

When one counts the *number of digits* in a numerical value one should not include zeros in the beginning of the number, as these zeros only help to denote where the decimal point should be. If one is counting the *number of decimals*, one should of course include leading zeros to the right of the decimal point. For example, the number 0.00147 is given with three digits but has five decimals. The number 12.34 is given with four digits but has two decimals.

If the magnitude of the error in $\tilde{a}$ does not exceed $\frac{1}{2} \cdot 10^{-t}$, then $\tilde{a}$ is said to have t **correct decimals**. The *digits* in $\tilde{a}$ which occupy positions where the unit is greater than or equal to 10^{-t} are called, then, **significant digits** (any initial zeros are not counted). Thus, the number 0.001234 ± 0.000004 has five correct decimals and three significant digits, while 0.001234 ± 0.000006 has four correct decimals and two significant digits. The number of correct decimals gives one an idea of the magnitude of the *absolute error*, while the number of significant digits gives a rough idea of the magnitude of the *relative error*.

We distinguish here between two ways of rounding off a number x to a given number t of decimals. In **chopping** (or round toward zero) one simply leaves off all the decimals to the right of the tth. That way is generally *not recommended* since the rounding error has, systematically, the opposite sign of the number itself. Also, the magnitude of the error can be as large as 10^{-t}.

In **rounding to nearest** (sometimes called "correct" or "optimal" rounding"), one chooses, a number with s decimals which is *nearest* to x. Hence if p is the part of the number which stands to the right of the sth decimal one leaves the tth decimal unchanged if and only if $|p| < 0.5 \cdot 10^{-s}$. Otherwise one raises the sth decimal by 1. In case of a tie, when x is equidistant to two s digit numbers then one raises the sth decimal if it is odd or leaves it unchanged if it is even (round to even). In this way, the error is positive or negative about equally often. The error in rounding a decimal number to s decimals will always lie in the interval $\big[-\frac{1}{2}10^{-s}, \frac{1}{2}10^{-s}\big]$.

Suppose that you are tabulating a transcendental function and a particular entry has been evaluated as 5.0835 correct to the digits given. You want to round the value to three decimals. Should the final digit be 4 or 5? The answer depends on whether there is a nonzero trailing digit. You compute the entry more accurately and find the 5.08350, then 5.083500, then 5.0835000, etc. Since the function is transcendental clearly there is no bound on how many digits you have to be compute before distinguishing if to round to 5.084 or 5.085. This is called the **tablemaker's dilemma**. This can be used to advantage in order to protect mathematical tables from illegal copying by rounding a few entries incorrectly where the error in doing so is insignificant due to several trailing zeros. An illegal copy could then be exposed simply by looking up these entries!

Example 2.1.1.

Shortening to three decimals:

0.2397	rounds to 0.240	(is chopped to 0.239)
−0.2397	rounds to −0.240	(is chopped to −0.239)
0.23750	rounds to 0.238	(is chopped to 0.237)
0.23650	rounds to 0.236	(is chopped to 0.236)
0.23652	rounds to 0.237	(is chopped to 0.236)

Observe that when one rounds off a numerical value one produces an error; thus it is occasionally wise to give more decimals than those which are correct. Take, for example, $a = 0.1237 \pm 0.0004$, which has three correct decimals according to the definition given previously. If one rounds to three decimals, one gets 0.124; here the third decimal is not correct, since the least possible value for a is 0.1233.

Example 2.1.2.

The difference between chopping and rounding can be important as is born out by the following story. The index of the Vancouver Stock Exchange, founded at the initial value 1000.000 in 1982, was hitting lows in the 500s at the end of 1983 even though the exchange apparently performed well. It was discovered (The Wall Street Journal, Nov. 8, 1983, p. 37) that the discrepancy was caused by a computer program which updated the index thousands of times a day and used chopping instead of rounding to nearest! The rounded calculation gave a value of

1098.892.

Review Questions

1. Clarify with examples the various types of error sources which occur in numerical work.
2. (a) Define "absolute error" and "relative error" for an approximation $\bar{x}$ to a scalar quantity x. What is meant by an error bound?

   (b) Generalize the definitions in (a) to a vector x.
3. (a) How is "rounding to nearest" performed.
4. Give π to four decimals using: (a) chopping; (b) rounding.
5. What is meant by the "tablemaker's dilemma"?

2.2 Computer Number Systems

2.2.1 The Position System

In order to represent numbers, we use in daily life a **position system** with base 10 (the decimal system). Thus to represent the numbers we use ten different characters, and the magnitude with which the digit a contributes to a number's value depends on the digit's position in the number. If the digit stands n steps to the right of the decimal point, the value contributed is $a \cdot 10^{-n}$. For example, the sequence of digits 4711.303 means

$$4 \cdot 10^3 + 7 \cdot 10^2 + 1 \cdot 10^1 + 1 \cdot 10^0 + 3 \cdot 10^{-1} 0 \cdot 10^{-2} + 3 \cdot 10^{-3}.$$

Every real number has a unique representation in the above way, except for the possibility of infinite sequences of nines—for example, the infinite decimal fraction 0.3199999... represents the same number as 0.32.

One can very well consider other position systems with base different from 10. Any natural number $\beta \geq 2$ can be used as base. One can show that every positive real number a has, with exceptions analogous to the nines-sequences mentioned above, a unique representation of the form

$$a = d_n\beta^n + d_{n-1}\beta^{n-1} + \ldots + d_1\beta^1 + d_0\beta^0 + d_{-1}\beta^{-1} + d_{-2}\beta^{-2} + \ldots,$$

or more compactly $a = (d_n d_{n-1} \ldots d_0.d_{-1}d_{-2}\ldots)_\beta$, where the coefficients d_i, the "digits" in the system with base β, are positive integers d_i such that $0 \leq d_i \leq \beta - 1$.

One of the greatest advantages of the position system is that one can give simple, general rules for the arithmetic operations. The smaller the base is, the simpler these rules become. This is just one reason why most computers operate in base 2, the **binary number system**. The addition and multiplication tables then take the following simple form:

$$0 + 0 = 0; \qquad 0 + 1 = 1 + 0 = 1; \qquad 1 + 1 = 10;$$
$$0 \cdot 0 = 0; \qquad 0 \cdot 1 = 1 \cdot 0 = 0; \qquad 1 \cdot 1 = 1;$$

In the binary system, the number seventeen, for example, becomes 10001, since $1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \text{sixteen} + \text{one} = \text{seventeen}$. Put another way $(10001)_2 = (17)_{10}$, where the index (in decimal representation) denotes the base of the number system. The numbers become longer written in the binary system; large integers become about 3.3 times as long, since N binary digits suffice to represent integers less than $2^N = 10^{N \log_{10} 2} \approx 10^{N/3.3}$.

Occasionally one groups together the binary digits in subsequences of three or four, which is equivalent to using 2^3 and 2^4, respectively, as base. These systems are called the **octal** and **hexadecimal** number systems, respectively. The octal system uses the digits from 0 to 7; in the hexadecimal system the digits 0 through 9 and the letters A, B, C, D, E, F ("ten" through "fifteen") are used.

Example 2.2.1.

$$\begin{aligned}
(17)_{10} &= (10001)_2 = (21)_8 = (11)_{16}, \\
(13.25)_{10} &= (1101.01)_2 = (15.2)_8 = (D.4)_{16}, \\
(0.1)_{10} &= (0.000110011001\ldots)_2 = (0.199999\ldots)_{16}.
\end{aligned}$$

Note that *the finite decimal fraction* 0.1 *cannot be represented exactly by a finite fraction in the binary number system*! (For this reason some pocket calculators use the base 10.)

Example 2.2.2. In 1991 a Patriot missile launched in Saudi Arabia failed to track and interrupt an incoming Scud due to a precision problem. The Scud then hit an Army barrack and killed 28 Americans. The computer used to control the Patriot missile was based on a design dating from the 1970's using 24-bit arithmetic. For the tracking computations time was recorded by the system clock in tenth of a second but converted to a 24-bit floating point number. Rounding errors in the time conversions caused an error in the tracking. After 100 hours of consecutive operations the calculated time in seconds was 359999.6567 instead of the correct value 360000, an error of 0.3433 seconds leading to an error in the calculated range of 687 meters. Modified software was later installed.

In the binary system the "point" used to separate the integer and fractional part of a number (corresponding to the decimal point) is called the binary point. The digits in the binary system are called **bits**(=**b**inary dig**its**).

We are so accustomed to the position system that we forget that it is built upon an ingenious idea. The reader can puzzle over how the rules for arithmetic operations would look if one used Roman numerals, a number system without the position principle described above.

Recall that rational numbers are precisely those real numbers which can be expressed as a quotient between two integers. Equivalently rational numbers are those whose representation in a position system have a finite number of digits or whose digits are repeating.

We now consider the problem of conversion between two number systems with different base. Since almost all computers use a binary system this problem arises as soon as one want to input data in decimal form or print results in decimal form.

Algorithm 2.2.1 Conversion between number systems

Let a be an integer given in number systems with base α. We want to determine its representation in a number systems with base β:

$$a = b_n\beta^m + b_{m-1}\beta^{n-1} + \cdots + b_0, \quad 0 \le b_i < \beta. \tag{2.2.1}$$

The computations are to be done in the system with base α and thus also β is expressed in this representation. The conversion is done by successive divisions of a with β: Set $q_0 = a$, and

$$q_k/\beta = q_{k+1}\beta + b_k, \quad k = 0, 1, 2, \dots. \tag{2.2.2}$$

(q_{k+1} is the quotient and r_k the remainder in the division.)

If a is not an integer, we write $a = b + c$, where b is the integer part and

$$c = b_{-1}\beta^{-1} + b_{-2}\beta^{-2} + b_{-3}\beta^{-3} + \cdots \tag{2.2.3}$$

is the fractional part, where $b_{-1}, b_{-2}, \dots$ are to be determined. These digits are obtained as the integer parts when successively multiplying c with β: Set $p_{-1} = c$, and

$$p_k \cdot \beta = b_k\beta + p_{k-1}, \quad k = -1, -2, -3 \dots. \tag{2.2.4}$$

Since a finite fraction in a number system with base α usually does not correspond to a finite fraction in the number system with base β rounding of the result is in general needed.

When converting by hand between decimal system and, for example, the binary system all computations are made in the decimal system ($\alpha = 10$ and $\beta = 2$). (It is then more convenient to convert the decimal number first to octal or hexadecimal, from which the binary representation easily follows.) If, on the other hand, the conversion is carried out on a binary computer, the computations are made in the binary system ($\alpha = 2$ and $\beta = 10$).

Example 2.2.3. Convert the decimal number 176.524 to ternary form (base $\beta = 3$). For the integer part we get $176/3 = 58$ with remainder 2; $58/3 = 19$ with remainder 1; $19/3 = 6$ with remainder 1; $6/3 = 2$ with remainder 0; $2/3 = 0$ with remainder 2. It follows that $(176)_{10} = (20112)_3$.

For the fractional part we compute $.524 \cdot 3 = 1.572$, $.572 \cdot 3 = 1.716$, $.716 \cdot 3 = 2.148, \dots$. Continuing in this way we obtain $(.524)_{10} = (.112010222\dots)_3$. The finite decimal fraction does not correspond to a finite fraction in the ternary number system!

2.2.2 Fixed and Floating Point Representation

A computer is in general built to handle pieces of information of a fixed size called a **word**. The number of digits in a word (usually binary) is called the **word-length** of the computer. Typical word-lengths are 32, 48, or 64 bits. A real or integer number is usually stored in a word. Integers can be exactly represented, provided that the word-length suffices to store all the digits in its representation.

In the first generation of computers calculations were made in a **fixed-point** number system, that is, real numbers were represented with a fixed number of t binary digits. If the word-length of the computer is $s+1$ bits (including the sign bit), then only numbers in the interval $I=[-2^{s-t},2^{s-t}]$ are permitted. Some common conventions in fixed point are $t=s$ (fraction convention) or $t=0$ (integer convention). This limitation causes difficulties, since even when $x\in I$, $y\in I$, we can have, e.g., $x-y\notin I$ or $x/y\notin I$. In a fixed-point number system one must see to it that all numbers, even intermediate results, remain within I. This can be attained by multiplying the variables by appropriate **scale factors**, and then transforming the equations accordingly. This is a tedious process. Moreover it is complicated by the risk that if the scale factors are chosen carelessly, certain intermediate results can have many leading zeros which can lead to poor accuracy in the final results. As a consequence, fixed point is very seldom used for computations with real numbers. An exception is in some on-line real-time computations, e.g., in digital filtering, where fixed point systems still are used. Otherwise it is limited to computations with integers as in subscript expressions for vectors and matrices.

By a **normalized floating point representation** of a real number a, we mean a representation in the form

$$a=\pm m\cdot\beta^q,\qquad \beta^{-1}\le m<1,\qquad q \text{ an integer.}\tag{2.2.5}$$

(Alternatively the representation can be normalized by the condition $1\le m<\beta$.) Such a representation is possible for all real numbers a, and unique if $a\neq 0$. Here the fraction part m is called the **mantissa** (also called **significand**), q is the **exponent** and β the **base** (also called the **radix**).)

To measure the difference between a floating point number and the real number it approximates we shall occasionally use "**unit in last place**" or **ulp**. For example, if in a decimal floating point system the number 3.14159 is represented as $0.3142\cdot 10^1$ this has an error of 0.41 ulps. We shall say that "the quantity is perturbed by a few ulps".

In a computer, the number of digits for q and m is limited by the word-length. Suppose that t digits is used to represent m. Then we can only represent floating point numbers of the form

$$\bar a=\pm\overline{m}\cdot\beta^e,\quad \overline{m}=(.d_1d_2\cdots d_p)_\beta,\quad 0\le d_i<\beta,\tag{2.2.6}$$

where $\overline{m}$ is the mantissa m rounded to p digits, and the exponent is limited to a finite range

$$e_{\min}\le e\le e_{\max}.\tag{2.2.7}$$

A floating point number system F is characterized by the base β, the precision t, and the numbers $e_{\min}$, $e_{\max}$. Only a finite set F of rational numbers can be represented in the form (2.2.7). The numbers in this set are called **floating point numbers**. Since $d_1\neq 0$ this set contains precisely $2(\beta-1)\beta^{p-1}(e_{\max}-e_{\min}+1)+1$

numbers. (Show this!) The limited number of digits in the exponent implies that a is limited in magnitude to an interval which is called the **range** of floating point system. If a is larger in magnitude than the largest number in the set F, then a cannot be represented at all (**exponent spill**). The same is true, in a sense, of numbers smaller than the smallest nonzero number in F.

Example 2.2.4. Consider the floating point number system for $\beta = 2$, $p = 3$, $e_{\min} = -1$, and $e_{\max} = 2$. The normalized numbers in the corresponding set F are shown in Fig. 2.2.1. The set F contains exactly $2 \cdot 16 + 1 = 33$ numbers. In this

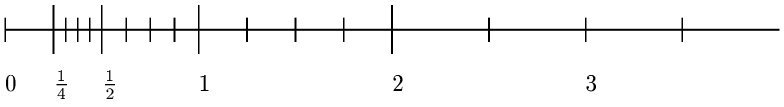


Figure 2.2.1. *Normalized numbers when $\beta = 2$, $p = 3$, $e_{\min} = -1$, and $e_{\max} = 2$.*

example the nonzero numbers of smallest magnitude that can be represented are $(0.100)_2 \cdot 2^{-1} = \frac{1}{4}$ and the largest is $(0.111)_2 \cdot 2^2 = \frac{7}{2}$.

Notice that floating point numbers are not equally spaced; the spacing jumps by a factor β at each power of β. (This wobbling is smallest for $\beta = 2$.) The spacing of floating point numbers is characterized by the **machine epsilon**, which is the distance ϵ_M from 1.0 to the next larger floating point numbers.

Even if the operands in an arithmetic operation are floating point numbers in F, the *exact* result of the operation may not be in F. For example, the exact product of two floating point p-digit numbers has $2p$ or $2p-1$ digits.

If a real number a is in the range of the floating point system the obvious way is to represent a by $\bar{a} = fl\,(a)$, where $fl\,(a)$ denotes a number in F which is nearest to a. This corresponds to rounding of the mantissa m, and according to Sec. **??**, we have

$$|\overline{m} - m| \le \frac{1}{2}\beta^{-p}.$$

(There is one exception. If $|m|$ after rounding should be raised to 1, then $|\overline{m}|$ is set equal to 0.1 and e raised by 1.) Since $m \ge 0.1$ this means that the magnitude of the relative error in $\bar{a}$ is at most equal to

$$\frac{\frac{1}{2}\beta^{-p} \cdot \beta^{e}}{m \cdot \beta^{e}} \le \frac{1}{2}\beta^{-p+1}.$$

Even with the exception mentioned above this relative bound still holds. (If chopping is used, this doubles the error bound above.) This proves the following theorem:

Theorem 2.2.1.

In a floating point number system $F = F(\beta, p, e_{\min}, e_{\max})$ every real number in the floating point range of F can be represented with a relative error, which does *not exceed the* **unit roundoff** u*, which is defined by*

$$u = \begin{cases} \frac{1}{2}\beta^{-p+1}, & \textit{if rounding is used,} \\ \beta^{-p+1}, & \textit{if chopping is used.} \end{cases} \tag{2.2.8}$$

The quantity u is, in many contexts, a natural unit for relative changes and relative errors. For example, termination criteria in iterative methods usually depend on the unit roundoff.

Example 2.2.5.

Sometimes it is useful to be able to approximately determine the unit roundoff in a program at run time. This may be done using the observation that $u \approx \mu$, where μ is *the smallest floating point number* x *such that* $fl\,(1+x) > 1$. The following program computes a number μ which differs from the unit roundoff u at most by a factor of 2:

```
$x := 1;$
**while** $1 + x > 1$   $x := x/2;$ **end**;
$\mu := x;$
```

One reason why u does not exactly equal μ is that so called double rounding occurs. This is when a result is first rounded to extended format and then to the target precision.

A floating point number system can be extended by including **denormalized numbers** (also called subnormal numbers). These are numbers with the minimum exponent and with the most significant digit equal to zero. The three numbers

$$(.001)_2 2^{-1} = 1/16, \quad (.010)_2 2^{-1} = 2/16, \quad (.011)_2 2^{-1} = 3/16,$$

can then also be represented. Denormalized numbers have fewer digits of precision than normalized numbers.

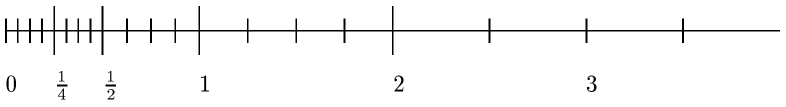


Figure 2.2.2. *Normalized and denormalized numbers when* $\beta = 2$, $p = 3$, $e_{\min} = -1$, *and* $e_{\max} = 2$.

2.2.3 IEEE Floating Point Standard

Actual computer implementations of floating point representations may differ in detail from the one given above. Although some pocket calculators use a floating point number systems with base $\beta = 10$, almost all modern computers use base $\beta = 2$. Most current computer now conform to the IEEE 754 standard for binary floating point arithmetic[2] This standard from 1985 (see [4]), which is the result of several years work by a subcommittee of the IEEE, is now implemented on almost all chips used for personal computers and workstations. (There is also a standard IEEE 854 for floating point arithmetic for base 2 and 10, which is used by several hand calculators.)

The IEEE 754 standard specifies basic and extended formats for floating point numbers, elementary operations and rounding rules available, conversion between different number formats, and binary-decimal conversion. Also the handling of exceptional cases like exponent overflow or underflow, division by zero are specified.

Two main basic formats, single and double precision are defined, using 32 and 64 bits respectively. In **single precision** a floating point number a is stored as a sign s (one bit), the exponent e (8 bits), and the mantissa m (23 bits). In **double precision** of the 64 bits 11 are used for the exponent, and 52 bits for the mantissa; see Fig. 2.2.2. The value v of a is in the normal case

$$v = (-1)^s (1.m)_2 2^e, \quad -e_{\min} \le e \le e_{\max}.$$

Note that the digit before the binary point is always 1 for a normalized number. This bit is not stored (the hidden bit). In that way one bit is gained for the mantissa. A biased exponent is stored and no sign bit used. For example, in single precision $e_{\min} = -126$ and $e_{\max} = 127$ and $e + 127$ is stored.

There are distinct representations for $+0$ and -0. ± 0 is represented by a sign bit, the exponent $e_{\min} - 1$ and a zero mantissa. Comparisons are defined so that $+0 = -0$. One use of a signed zero is to distinguish an positive and negative underflowed numbers. Another use occurs in complex arithmetic.

Example 2.2.6. The function $\sqrt{x}$ is multivalued and there is no way to select the values so the function is continuous over the whole complex plane. If a branch cut is made by excluding all real negative numbers from consideration the square root becomes continuous. Signed zero provides a way to distinguish numbers of the form $x + i(+0)$ and $x + i(-0)$ and to select one or the other side of the cut.

Infinity is also signed and $\pm\infty$ is represented by the exponent $e_{\max} + 1$ and a zero mantissa. When an overflows occurs the result is set to $\pm\infty$. This is safer than simply returning the largest representable number, that may be nowhere near the correct answer. The result $\pm\infty$ is also obtained from the illegal operations $a/0$, where $a \neq 0$. The infinity symbol obeys the usual mathematical conventions, such as $\infty + \infty = \infty$, $(-1) \times \infty = -\infty$, $a/\infty = 0$ if $a \neq 0$.

The IEEE standard also includes two extended precision formats that offer extra precision and exponent range. The standard only specifies a lower bound on how many extra bits the provides. Hardware implementation of extended precision normally does not use a hidden bit, so the double extended format uses 80 bits rather than 79. Extended formats simplify tasks such as computing elementary functions accurately in single or double precision. Extended precision formats are

[2]W. Kahan, University of California, Berkeley, was given the Turing Award by the Association of Computing Machinery for his contribution to this standard.

used also by hand calculators. These will often display 10 decimal digits but use 13 digits internally—"the calculator knows more than it shows"!

The characteristics of the IEEE formats are summarized in Table 2.2.1.

Table 2.2.1. *IEEE floating point formats.*

	Format	t	e	$e_{\min}$	$e_{\max}$
Single	32 bits	$23+1$	8 bits	-126	127
Single extended	≥ 43 bits	≥ 32	≥ 11 bits	≤ -1022	≥ 1023
Double	64 bits	$52+1$	11 bits	-1022	1023
Double extended	≥ 79 bits	≥ 64	≥ 15 bits	≤ -16382	≥ 16383

(The hidden bit in the mantissa accounts for the +1 in the table.)

The unit roundoff equals $2^{-24} \approx 5.96 \cdot 10^{-8}$ in single and $2^{-53} \approx 1.11 \cdot 10^{-16}$ in double precision. (The machine epsilon is twice as large.) The largest number that can be represented is $2.0 \cdot 2^{127} \approx 3.4028 \times 10^{38}$ in single precision and $2.0 \cdot 2^{1023} \approx 1.7977 \times 10^{308}$ in double precision. The smallest number is $1.0 \cdot 2^{-126} \approx 1.1755 \times 10^{-38}$ in single precision and $1.0 \cdot 2^{-1022} \approx 2.2251 \times 10^{-308}$ in double precision.

Example 2.2.7. Although the exponent range of the floating point formats seems reassuringly large, even simple programs can quickly give exponent spill. If $x_0 = 2$, $x_{n+1} = x_n^2$, then already $x_{10} = 2^{1024}$ is larger than what IEEE double precision permits. One should also be careful in computations with factorials, e.g., $35! > 10^{40}$ and $459! > 10^{1026}$.

Four rounding modes are supported by the standard. The default rounding mode is round to nearest representable number, with round to even in case of a tie. (Some computers in case of a tie round away from zero, i.e., raise the absolute value of the number, because this is easier to realize technically.) Chopping is also supported as well as directed rounding to ∞ and to $-\infty$. The latter mode simplifies the implementation of interval arithmetic, see Section 2.4.5.

The standard specifies that all arithmetic operations should be performed as if they were first calculated to infinite precision and then rounded to a floating point number according to one of the four modes mentioned above. This also includes the square root and conversion between integer and floating point. The standard also requires the conversion between internal formats and decimal to be correctly rounded.

This can be implemented using extra **guard digits** in the intermediate result of the operation before normalization and rounding. Using a single guard digit, however, will not always ensure the desired result. However by introducing a second guard digit and a third sticky bit (the logical OR of all succeeding bits) the rounded exact result can be computed at only a little more cost (Goldberg [16]). One reason for specifying precisely the results of arithmetic operations is to improve the portability of software. If a program is moved between two computers both supporting the IEEE standard intermediate results should be the same.

IEEE arithmetic is a closed system, i.e. every operation, even mathematical invalid operations, even $0/0$ or $\sqrt{-1}$ produces a result. To handle exceptional situations without aborting the computations some bit patterns (see Table 2.2.2) are reserved for special quantities like NaN ("Not a Number") and ∞. NaNs (there are more than one NaN) are represented by $e = e_{\max} + 1$ and $m \neq 0$.

Table 2.2.2. *IEEE 754 representation.*

Exponent	Mantissa	Represents
$e = e_{\min} - 1$	$m = 0$	± 0
$e = e_{\min} - 1$	$m \neq 0$	$\pm 0.m \cdot 2^{e_{\min}}$
$e_{\min} < e < e_{\max}$		$\pm 1.m \cdot 2^{e}$
$e = e_{\max} + 1$	$m = 0$	$\pm\infty$
$e = e_{\max} + 1$	$m \neq 0$	NaN

An exponent $e = e_{\min} - 1$ and $m \neq 0$, signifies the denormalized number

$$v = (-1)^s (0.m)_2 2^{e_{\min}};$$

see Table 2.2.2. The smallest denormalized number that can be represented is $2^{-126-23} \approx 7.14 \times 10^{-44}$ in single precision and $2^1022 - 52 \approx 4.94 \times 10^{-324}$ in double precision.

Note that the gap between 0 and the smallest normalized number is $1.0 \times 2^{e_{\min}}$. This is much larger than for the spacing $2^{-p+1} \times 2^{e_{\min}}$ for the normalized numbers for numbers just larger than the underflow threshold; compare Example 2.2.4. With denormalized numbers the spacing becomes more regular and permits what is called **gradual underflow**.. This makes many algorithms well behaved also close to the underflow threshold. Another advantage of having gradual underflow is that it makes it possible to preserve the property

$$x = y \quad \Leftrightarrow \quad x - y = 0$$

as well as other useful relations. Several examples of how denormalized numbers makes writing reliable floating point code easier are analyzed by Demmel [12].

One illustration of the use of extended precision is in converting between IEEE 754 single precision and decimal. The converted single precision number should ideally be converted with enough digits so that when it is converted back the binary single precision number is recovered. It might be expected that since $2^{24} < 10^8$ eight decimal digits in the converted number would suffice. However, it can be shown that nine decimal digits are needed to recover the binary number uniquely (see Goldberg [16, Theorem. 15] and Problem 3). When converting back to binary form a rounding error as small as one ulp will give the wrong answer. To do this conversion efficiently extended single precision is needed!

A NaN is generated by operations such as $0/0$, $+\infty + (-\infty)$, $0 \times \infty$ and $\sqrt{-1}$. A NaN compares unequal with everything including itself. (Note that $x \neq x$ is a simple way to test if x equals a NaN.) When a NaN and an ordinary floating-point number is combined the result is the same as the NaN operand. A NaN is often used also for uninitialized or missing data.

Exceptional operations also raise a flag. The default is to set a flag and continue, but it is also possible to pass control to a trap handler. The flags are “sticky” in that they remain set until explicitly cleared. There is one flag for each of the following five exceptions: underflow, overflow, division by zero, invalid operation and inexact. By testing the flags it is, for example, possible to test if an overflow is genuine or the result of division by zero.

Because of cheaper hardware and increasing problem sizes double precision is more and more used in scientific computing. With increasing speed and memory becoming available, bigger and bigger problems are being solved and actual problems may soon require more than IEEE double precision! When the IEEE 754 standard was defined no one expected computers able to execute more than 10^{12} floating point operations per second!

2.2.4 Elementary Functions

Although the square root is included, the IEEE 754 standard does not deal with the implementation of elementary functions, i.e., the exponential function exp, the logarithm log, and the trigonometric and hyperbolic functions sin, cos, tan, sinh, cosh, tanh, and their inverse functions. With the IEEE 754 standard more accurate implementations are possible which in many cases give almost correctly rounded exact results. To always guarantee correctly rounded exact results sometimes require computing many more digits than the target accuracy (cf. the tablemaker’s dilemma) and therefore is in general too costly. It is also important to preserve monotonicity, e.g, $0 \leq x \leq y \leq \pi/2 \Rightarrow \sin x \leq \sin y$, and range restrictions, e.g., $\sin x \leq 1$, but these demands may conflict with rounded exact results!

The first step in computing an elementary function is to perform a **range reduction**. To compute $\sin x$ an additive range reduction is first performed, in which a reduced argument x^*, $-\pi/4 \leq x^* \leq \pi/4$ is computed by finding an integer k such that $x^* = x - k\pi/2$ ($\pi/2 = 1.57079\,63267\,9489661923$). (Quantities that are often used in standard subroutines are listed in decimal form to 30 digits and octal form to 40 digits in Hart et al. [Appendix C][21] and to 40 and 44 digits in Knuth [27, Appendix A].) Then $\sin x = \pm \sin x^*$ or $\sin x = \pm \cos x^*$, depending on if k mod 4 equals $0, 1, 2$ or 3. Hence approximation for $\sin x$ and $\cos x$ need only be provided for $0 \leq x \leq \pi/4$. If the argument x is very large then cancellation in the range reduction can lead to poor accuracy; see Example **??**.

For the exponential an integer k is determined such that $x^* = x - k \ln 2$, $x^* \in [0, \ln 2]$ ($\ln 2 = 0.69314\,71805\,59945\,30942\ldots$) then it holds that $\exp(x) = \exp(x^*) \cdot 2^k$. To compute $\ln x$, $x > 0$, a multiplicative range reduction is used. An integer k is determined such that $x^* = x/2^k$, $x^* \in [1/2, 1]$. Then $\ln x = \ln x^* + k \cdot \ln 2$.

Details about implementations can be found in older references like Hart et al. [21] and Cody and Waite [9]. These contain many tables of coefficients of polynomial and rational approximations, suitable for software implementations. A trend now is that elementary functions are more and more implemented in hardware. Hardware implementations are discussed by Muller [32].

To test the implementation of elementary function a Fortran package ELEFUNT has been developed by Cody [10]. This checks the quality using indentities like $\cos x = \cos(x/3)(4\cos^2(x/3) - 1)$. For complex elementary functions a package CELEFUNT serves the same purpose; see Cody [11].

2.2.5 Multiple Precision Arithmetic

Occasionally one may want to perform some calculations, e.g., the evaluation of some elementary functions, to very high precision. Some important algorithms, which may be used include power series, continued fractions, solution of equations (with Newton's method or other superlinearly convergent methods etc.). For performing such tasks it is convenient to have routines for performing floating point operations on numbers represented as arrays in floating point with a large base and a long mantissa. In this way arithmetic of any given precision can be used to *simulate arithmetic of arbitrarily high precision.*

Brent [7, 6] developed one of the first such multiple-precision Fortran software package. This package represent multiple precision numbers as arrays of integers and operates on them with integer arithmetic. A more recent package is that of Bailey [5], which is written in Fortran 77 code. It represents multiple precision numbers as a vector of single precision floating point numbers and uses a base of 2^{24}. Complex multiprecision numbers are also represented.

In an Appendix A we describe the basics of **Mulprec**, a collection of MATLAB m-files for, in principle, unlimited multiple precision floating point computation. and give examples of its use.

Review Questions

1. What base β is used in the binary, octal and hexadecimal number systems?

2. Show that any *finite* decimal fraction corresponds to a binary fraction that eventually is periodic.

3. What is meant by a normalized floating point representation of a real number?

3. How large can the maximum relative error be in representation of a real number a in the floating point system $F = F(\beta, p, e_{\min}, e_{\max})$? It is assumed that a is in the range of F.

4. How are the quantities "machine epsilon" and "unit round off" defined?

5. What are the characteristics of the IEEE single and double precision formats?

6. What are the advantages of including denormalized numbers in the IEEE standard?

7. Give examples of operations that give NaN as result.

Problems

1. Which rational numbers can be expressed with a finite number of binary digits to the right of the binary point?

2. (a) Prove the conversion algorithms described in Sec. 2.2.1.

(b) Show that the octal form of 0.1 is $(0.1)_{10} = 0.063146\,3146\,3146\ldots$. What error is incurred in rounding this number to IEEE 754 single precision and double precision, respectively?

3. (W. Kahan) An (over-)estimate of u can be obtained for almost any computer by evaluating $|3 \times (4/3 - 1) - 1|$ Using rounded floating point for every operation. Test this on a calculator or computer available to you.
4. (D. Goldberg) The binary single precision numbers in the half-open interval $[10^3, 1024)$ have 10 bits to the left and 14 bits to the right of the binary point. Show that there are $(2^{10} - 10^3) \cdot 2^{14} = 393,216$ such numbers, but only $(2^{10} - 10^3) \cdot 10^4 = 240,000$ decimal numbers with 8 decimal digits in the same interval. Conclude that 8 decimal digits are not enough to uniquely represent single precision binary numbers in the IEEE 754 standard.
5. Suppose one wants to compute the power A^n of a square matrix A, where n is a positive integer. To compute $A^{k+1} = A \cdot A^k$, for $k = 1 : n - 1$ requires $n - 1$ matrix multiplications. Show that the number of multiplications can be reduced to less than $2\lfloor \log_2 n \rfloor$ by converting n into binary form and successive squaring $A^{2k} = (A^k)^2$, $k = 1 : \lfloor \log_2 n \rfloor$.
6. Give in decimal representation: (a) $(10000)_2$; (b) $(100)_8$; (c) $(64)_16$; (d) $(FF)_16$; (e) $(0.11)_8$; (g) the largest positive integer which can be written with thirty–one binary digits (answer with one significant digit).
7. (a) Show how the following numbers are stored in the basic single precision format of the IEEE 754 standard: 1.0; -0.0625; 250.25; 0.1.

   (b) Give in decimal notation the largest and smallest positive numbers which can be stored in this format.
8. (N. J. Higham.) Let a and b be floating point numbers with $a \le b$. Show that the inequalities $a \le fl((a + b)/2) \le b$ can be violated in base 10 arithmetic. Show that $a \le fl(a + (b - a)/2) \le b$ in base β arithmetic, assuming the use of a guard digit.

2.3 Accuracy and Rounding Errors

2.3.1 Floating Point Arithmetic

It is useful to have a model of how the basic floating point operations are carried out. If x and y are two floating point numbers we denote by

$$fl\,(x + y), \quad fl\,(x - y), \quad fl\,(x \cdot y), \quad fl\,(x/y)$$

the results of floating addition, subtraction, multiplication, and division, which the machine stores in memory (after rounding or chopping). We will in the following assume that underflow or overflow does not occur. and that the following **standard model** for the arithmetic holds:

Definition 2.3.1. *Assume that $x, y \in F$. Then in the* **standard model** *it holds*

$$fl\,(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \le u, \tag{2.3.1}$$

where u is the unit roundoff and "op" stands for one of the four elementary operations $+$, $-$, $\cdot$, and $/$.

The standard model holds, also for with the default rounding mode for computers implementing the IEEE 754 standard. In this case we also have

$$fl\,(\sqrt{x}) = \sqrt{x}(1+\delta), \quad |\delta| \le u, \tag{2.3.2}$$

If a guard digit is lacking then instead of (2.3.1) only the weaker model

$$fl\,(x \text{ op } y) = x(1+\delta_1) \text{ op } y(1+\delta_2), \quad |\delta_i| \le u, \tag{2.3.3}$$

holds for addition/subtraction. The lack of a guard digit is a serious drawback and can lead to damaging inaccuracy caused by cancellation. Many algorithms can be proved to work satisfactorily only if the standard model (2.3.1) holds. We remark that on current computers multiplication is as fast as addition/subtraction. Division usually is 5–10 times slower than a multiply and a square root about twice slower than division.

Some earlier computers lack a guard digit in addition/subtraction. Notable examples are several models of Cray computers (Cray 1,2, X-MP,Y-MP, and C90) before 1995, which were designed to have the highest possible floating-point performance. The IBM 360, which used a hexadecimal system, lacked a (hexadecimal) guard digit between 1964–1967. The consequences turned out to be so intolerable that a guard digit had to be retrofitted.

Sometimes the floating point computation is more precise than what the standard model assumes. An obvious example is that when the exact value x op y can be represented as a floating point number there is no rounding error at all.

Some computers can perform a *fused multiply-add* operation, i.e. an expression of the form $a \times x + y$ can be evaluated with just one instruction and there is only *one rounding error* at the end

$$fl\,(a \times x + y) = (a \times x + y)(1+\delta), \quad |\delta| \le u.$$

Fused multiply add can be used to advantage in many algorithms. For example, Horner's rule to evaluate the polynomial $p(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$, which uses the recurrence relation $b_0 = a_0$, $b_i = b_{i-1} \cdot x + a_i$, $i = 1 : n$, needs only n fused multiply-add operations.

It is important to realize that these floating point operations have, to some degree, other properties than the exact arithmetic operations. For example, floating point addition and multiplication are commutative, but not associative and the distributive law also fails for them. This makes the analysis of floating point computations more difficult.

Example 2.3.1.

To show that associativity does not, in general, hold for floating addition, consider adding the three numbers

$$a = 0.1234567 \cdot 10^0, \qquad b = 0.4711325 \cdot 10^4, \qquad c = -b.$$

in a decimal floating point system with $t = 7$ digits in the mantissa. The following scheme indicates how floating point addition is performed:

$$fl\,(b + c) = 0, \qquad fl\,(a + fl\,(b + c)) = a = 0.1234567 \cdot 10^0$$

$$\begin{array}{rcr|r} a & = & 0.0000123 & 4567\cdot 10^4 \\ +b & = & 0.4711325 & \cdot 10^4 \\ \hline fl\,(a+b) & = & 0.4711448 & \cdot 10^4 \\ c & = & -0.4711325 & \cdot 10^4 \end{array}$$

The last four digits to the right of the vertical line are lost by **outshifting**, and

$$fl\,(fl\,(a+b)+c) = 0.0000123\cdot 10^4 = 0.1230000\cdot 10^0 \neq fl\,(a+fl\,(b+c)).$$

An interesting fact is that assuming a guard digit is used *floating point subtraction of two sufficiently close numbers is always exact.*

Lemma 2.3.2 (Sterbenz).
Let the floating point numbers x and y satisfy

$$y/2 \le x \le 2y.$$

If subtraction is performed with a guard digit then $fl(x-y) = x-y$, unless $x-y$ underflows.

Proof. By the assumption the exponent of x and y in the floating point representations of x and y can differ at most by one unit. If the exponent is the same then the exact result will be computed. Therefore assume the exponents differ by one. After scaling and, if necessary, interchanging x and y it holds that Then $x/2 \le y \le x < 2$ and the exact difference $z = x - y$ is of the form

$$\begin{array}{rl} x = & x_1 . x_2 \ldots x_t \\ y = & 0\,. y_1 \ldots y_{t-1} y_t \\ \hline z = & z_1 . z_2 \ldots z_t z_{t+1} \end{array}$$

But from the assumption $x/2 - y \le 0$ or $x - y \le y$. Hence we must have $z_1 = 0$, so after shifting the exact result is obtained also in this case. $\square$

With gradual underflow, as in the IEEE 754 standard, the condition that $x-y$ does not underflow can be dropped.

Example 2.3.2. A corresponding result holds for any base β. For example, using four digit floating decimal arithmetic we get with guard digit

$$fl\,(0.1000\cdot 10^1 - 0.9999) = 0.0001 = 1.000\cdot 10^{-4},$$

(exact) but without guard digit

$$fl\,(0.1000\cdot 10^1 - 0.9999) = (0.1000 - 0.0999)10^1 = 0.0001\cdot 10^1 = 1.000\cdot 10^{-3}.$$

The last result satisfies equation (2.3.3) with $|\delta_i| \le 0.5\cdot 10^{-3}$ since $0.10005\cdot 10^1 - 0.9995 = 10^{-3}$.

Outshiftings are common causes of loss of information that may lead to **catastrophic cancellation** later, in the computations of a quantity that one would have liked to obtain with good relative accuracy.

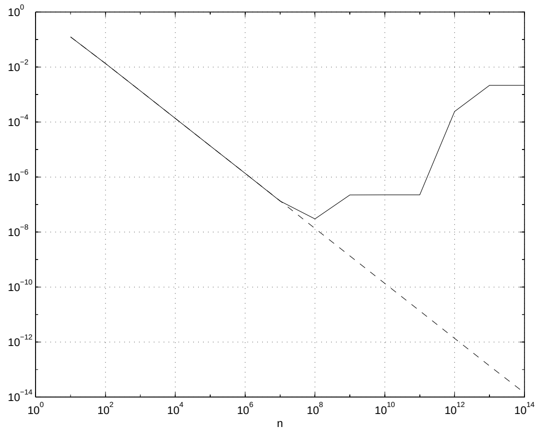


Figure 2.3.1. *Computed values for* $n = 10^p$*,* $p = 1 : 14$*, of the sequences: dotted line* $|(1 + 1/n)^n - e|$*; solid line* $|\exp(n \log(1 + 1/n)) - e|$ *using* (2.3.4).

Example 2.3.3.

An example where the result of Lemma 2.3.2 can be used to advantage is in computing compounded interest. Consider depositing the amount c every day on an account with an interest rate i compounded daily. Then with the accumulated capital at the end of the year equals

$$c[(1 + x)^n - 1]/x, \quad x = i/n \ll 1,$$

and $n = 365$. Using this formula does not give a accurate results. The reason is that a rounding error occurs in computing $fl(1+x) = 1+\bar{x}$ and low order bits of x is lost. For example, if $i = 0.06$ then $i/n = 0.0001643836$ and in decimal arithmetic using six digits when this is added to one we get $fl(1+i/n) = 1.000164$ so four low order digits are lost.

The problem then is to accurately compute $(1 + x)^n = \exp(n \ln(1 + x))$. The formula

$$\ln(1 + x) = \begin{cases} x, & \text{if } fl\,(1 + x) = 1; \\ x\dfrac{\ln(1 + x)}{(1 + x) - 1}, & \text{otherwise.} \end{cases} \tag{2.3.4}$$

can be shown to yield accurate results when $x \in [0, 3/4]$ provided subtraction is performed with a guard digit and the computed value of $\ln(1 + x)$ equals the exact result rounded; see Goldberg [16, p. 12].

To check this formula we recall that the base e of the natural logarithm can be defined by the limit

$$e = \lim_{n\to\infty} (1 + 1/n)^n$$

In Figure **??** we show computed values, using double precision floating point arithmetic, of the sequence $|(1 + 1/n)^n - e|$ for $n = 10^p$, $p = 1 : 14$. More precisely, the expression was computed as

$$|\exp(n\log(1+1/n))-\exp(1)|$$

The smallest difference $3\cdot 10^{-8}$ occurs for $n=10^8$, for which about half the number of bits in $x=1/n$ are lost. For larger values of n rounding errors destroy the convergence. However, using (2.3.4) we obtain correct results for all values of n! (The Maclaurin series $\ln(1+x)=x-x^2/2+x^3/3-x^4/4+\cdots$ will also give good results; see Computer Exercise 1.

A fundamental insight from the above examples can be expressed in the following way:

> *"mathematically equivalent" formulas or algorithms are not in general "numerically equivalent".*

This adds a new dimension to calculations in finite precision arithmetic and it will be a recurrent theme in the analysis of algorithms in this book!

By **mathematical equivalence** of two algorithms we mean here that the algorithms give exactly the same results from the same input data, if the computations are made without rounding error ("with infinitely many digits"). One algorithm can then, as a rule, formally be derived from the other using the rules of algebra for real numbers, and with the help of mathematical identities. Two algorithms are **numerically equivalent** if their respective floating point results, using the same input data are the same.

In error analysis for compound arithmetic expressions based on the standard model (2.3.1) one often needs an upper bound for quantities of this form

$$\epsilon \equiv |(1+\delta_1)(1+\delta_2)\cdots(1+\delta_n)-1|,\quad |\delta_i|\le u,\quad i=1:n.$$

Since $\epsilon\le(1+u)^n-1$ and

$$\ln(1+u)^n=n\ln(1+u)<nu,\quad (u<1),$$

it holds that $\epsilon<e^{nu}-1$. An elementary calculation now gives

$$\begin{aligned}\epsilon &< e^{nu}-1\le nu+\frac{nu^2}{2!}+\frac{nu^3}{3!}+\cdots\\ &< nu\Big(1+\frac{nu}{2}+\Big(\frac{nu}{2}\Big)^2+\cdots\Big)<\frac{nu}{1-nu/2}=:\gamma_n,\end{aligned}\qquad (2.3.5)$$

provided that $nu<2$. (The notation γ_n was introduced by N. J. Higham [24].) If we make the realistic assumption that $nu<0.1$, then $\epsilon<1.06nu=n\tilde{u}$.

Complex arithmetic can be reduced to real arithmetic. Let $x=a+ib$ and $y=c+id$ be two complex numbers. Then we have:
Complex division

$$\begin{aligned}x\pm y&=a\pm c+i(b\pm d),\\ xy&=(ac-bd)+i(ad+bc),\\ x/y&=\frac{ac+bd}{c^2+d^2}+i\frac{bc-ad}{c^2+d^2},\end{aligned}\qquad (2.3.6)$$

Hence, complex addition (subtraction) needs two real additions and multiplying two complex numbers requires four real multiplications using the formula

For computing the square root of a complex number $u+iv = \sqrt{x+iy}$ we have

$$u = \left(\frac{r+x}{2}\right)^{1/2}, \qquad v = \left(\frac{r-x}{2}\right)^{1/2}, \quad r = \sqrt{x^2+y^2}. \tag{2.3.7}$$

When $x > 0$ there will be cancellation when computing v, which can be severe if also $|x| \gg |y|$ (cf. Sec. 2.3.6). To avoid this we note that $uv = \sqrt{r^2-x^2}/2 = y/2$, so v can be computed from $v = y/(2u)$. When $x < 0$ we instead compute v from (2.3.7) and set $u = y/(2v)$.

Lemma 2.3.3. *Assuming the standard model* (2.3.1) *the complex operations computed according to* (2.3.6) *satisfy*

$$\begin{aligned} fl\,(x \pm y) &= (x \pm y)(1+\delta), \quad |\delta| \le u, \\ fl\,(xy) &= xy(1+\delta), \quad |\delta| \le \sqrt{2}\gamma_2, \\ fl\,(x/y) &= x/y(1+\delta), \quad |\delta| \le \sqrt{2}\gamma_4, \end{aligned} \tag{2.3.8}$$

where δ is a complex number and γ_n is defined in (2.3.5).

Proof. See Higham [24, Sec. 3.6]. □

Most rounding error analysis given in this book are formulated for real arithmetic. Since the bounds in Lemma 2.3.3 are of the same form as the standard model for real arithmetic, these can simply be extended to complex arithmetic.

2.3.2 Basic Results

We now use the notation of Sec. 2.3.1 and the standard model of floating point arithmetic (Definition 2.3.1) to carry out rounding error analysis of some basic computations. Most but not all results are still true if only the weaker bound (2.3.3) hold for addition and subtraction. Note that $fl\,(x \text{ op } y) = (x \text{ op } y)(1+\delta)$, $|\delta| \le u$, can be interpreted for multiplication to mean that $fl\,(x \cdot y)$ is the *exact result* of $x \cdot y(1+\delta)$ for some δ, $|\delta| \le u$. In the same way, the results using the three other operations can be interpreted as *the result of exact operations where the operands have been perturbed by a relative amount which does not exceed* u. In **backward error analysis** (see Sec. 2.4.6) one applies the above interpretation step by step backwards in an algorithm.

By repeatedly using formula (2.3.1) in case of multiplication, one can show that

$$x_1x_2(1+\delta_2)x_3(1+\delta_3)\cdots x_n(1+\delta_n), \quad |\delta_i| \le u, \quad i = 2:n.$$

That is, the computed **product** $fl\,(x_1x_2\cdots x_n)$ is *exactly* equal to a product of the factors

$$\tilde{x}_1 = x_1, \quad \tilde{x}_i = x_i(1+\delta_i), \quad i = 2:n.$$

Using the estimate and notation of (2.3.5) it follows from this analysis that

$$|fl\,(x_1x_2\cdots x_n) - x_1x_2\cdots x_n| < \gamma_{n-1}|x_1x_2\cdots x_n|, \tag{2.3.9}$$

which bounds the **forward error** of the computed result.

For a **sum** of n floating point numbers similar results can be derived. If the sum is computed in the natural order we have

$$\begin{aligned} &fl\,(\cdots(((x_1+x_2)+x_3)+\cdots+x_n)) \\ &\qquad = x_1(1+\delta_1)+x_2(1+\delta_2)+\cdots+x_n(1+\delta_n), \end{aligned}$$

where

$$|\delta_1| < \gamma_{n-1}, \qquad |\delta_i| < \gamma_{n+1-i}. \quad i = 2:n,$$

and thus the computed sum is *the exact sum* of the numbers $x_i(1+\delta_i)$. This also gives an estimate the forward error

$$\begin{aligned} &|fl\,(\cdots(((x_1+x_2)+x_3)+\cdots+x_n)) - (x_1+x_2+x_3+\cdots+x_n)| \\ &\qquad < \sum_{i=1}^{n} \gamma_{n+1-i}|x_i| \le \gamma_{n-1}\sum_{i=1}^{n}|x_i|, \end{aligned} \tag{2.3.10}$$

where the last upper bound holds independent of the summation order.

Notice that to minimize the first upper bound in equation (2.3.10), the terms *should be added in increasing order of magnitude*! For large n an even better bound than can be shown if the summation is done using the divide-and-conquer technique described in Sec. 1.5; see Problem 2.

Example 2.3.4.

Using a hexadecimal machine ($\beta = 16$), with $t = 6$ and chopping ($u = 16^{-5} \approx 10^{-6}$) one computed

$$\sum_{n=1}^{10{,}000} n^{-2} \approx 1.644834$$

in two different orders. Using the natural summation order $n = 1, 2, 3, \ldots$ the error was $1.317\cdot 10^{-3}$. Summing in the opposite order $n = 10,000, 9,999, 9,998\ldots$ the error was reduced to $2\cdot 10^{-6}$. This was not unexpected. Each operation is an addition, where the partial sum s is increased by n^{-2}. Thus, in each operation, one commits an error of about $s\cdot u$, and all these errors are added. Using the first summation order, we have $1 \le s \le 2$ in every step, but using the other order of summation we have $s < 10^{-2}$ in $9,900$ of the $10,000$ additions.

Similar bounds for roundoff errors can easily be derived for basic vector and matrix operations; see Wilkinson [1965, pp.114–118]. For an **inner product** x^Ty computed in the natural order we have

$$fl\,(x^Ty) = x_1y_1(1+\delta_1)+x_2y_2(1+\delta_2)+\cdots+x_ny_n(1+\delta_n)$$

where

$$|\delta_1| < \gamma_n, \qquad |\delta_r| < \gamma_{n+2-i}, \quad i = 2 : n.$$

The corresponding forward error bound becomes

$$|fl\,(x^T y) - x^T y| < \sum_{i=1}^{n} \gamma_{n+2-i} |x_i||y_i| < \gamma_n \sum_{i=1}^{n} |x_i||y_i|,$$

If we let $|x|$, $|y|$ denote vectors with elements $|x_i|$, $|y_i|$ the last estimate can be written in the simple form

$$|fl\,(x^T y) - x^T y| < \gamma_n |x^T||y|. \tag{2.3.11}$$

This bound is independent of the summation order and holds also for the weaker model (2.3.3) valid with no guard digit rounding.

The bound (2.3.11) can easily be extended to matrix multiplication. Let $A \in \mathbf{R}^{m\times n}$, $B \in \mathbf{R}^{n\times p}$, and denote by $|A|$ and $|B|$ matrices with elements $|a_{ij}|$ and $|b_{ij}|$. Then it holds that

$$|fl\,(AB) - AB| < \gamma_n |A||B|. \tag{2.3.12}$$

The product of two t digit floating point numbers can be exactly represented with at most $2t$ digits. This allows inner products may often be computed in extended precision without much extra cost. If fl_e denotes computation with extended precision and u_e the corresponding unit roundoff then the forward error bound for an inner product becomes

$$|fl\,(fl_e((x^T y)) - x^T y| < u|x^T y| + \frac{nu_e}{1 - nu_e/2}(1 + u)|x^T||y|, \tag{2.3.13}$$

where the first term comes form the final rounding. If $|x^T||y| \leq u|x^T y|$ then the computed inner product is almost as accurate as the correctly rounded exact result. These accurate inner products can be used to improve accuracy by iterative refinement (see Chapter 7–9) in many linear algebra problems. However, since computations in extended precision are machine dependent it has been difficult to make such programs portable.[3] The recent development of Extended and Mixed Precision BLAS (Basic Linear Algebra Subroutines) (see [28] may now make this more feasible!

2.3.3 Compensated Summation

To reduce the effects of rounding errors in computing a sum $\sum_{i=1}^{n} x_i$ one can use **compensated summation**. In this algorithm the rounding error in each addition is estimated and then compensated for with a correction term. Compensated summation can be useful when a large number of small terms are to be added as in

[3]It was suggested that the IEEE 754 standard should require inner products to be precisely specified, but that did not happen.

numerical quadrature. Another example is the case in the numerical solution of initial value problems for ordinary differential equations. Note that in this application the terms have to be added in the order in which they are generated.

Compensated is based on the possibility to *simulate double precision floating point addition in single precision arithmetic.* To illustrate the basic idea we take as in Example 2.3.1

$$a = 0.1234567 \cdot 10^0, \qquad b = 0.4711325 \cdot 10^4,$$

so that $s = fl\,(a+b) = 0.4711448 \cdot 10^4$, Suppose we form

$$c = fl\,(fl\,(b-s)+a) = -0.1230000 \cdot 10^0 + 0.1234567 \cdot 10^0 = 4567000 \cdot 10^{-3}.$$

Note that the variable c is computed without error and picks up the information that was lost in the operation $fl\,(a+b)$.

The following algorithm uses this idea to accurately computing $\sum_{i=1}^n x_i$:

Algorithm 2.3.1 Compensated Summation.

```
$s := x_1$;  $c := 0$;
for $i = 2 : n$
   $y := c + x_i$;
   $t := s + y$;
   $c := (s - t) + y$;
   $s := t$;
end
```

It can be proved (see Goldberg [16, 1991]) that on binary machines with a guard digit the computed sum satisfies

$$s = \sum_{i=1}^{n} (1+\xi_i)x_i, \quad |\xi_i| < 2u + O(nu^2). \tag{2.3.14}$$

This formulation is a typical example of a backward error analysis; see Sec. 2.4.6. The single precision term in the error bound is independent of n.

2.3.4 Standard Error

The bounds for the accumulated rounding error we have derived so far are estimates of the **maximal error**. These bounds are often much too pessimistic when the number of variables is large. As a complement one can use the **standard error**.

The theory of standard error is based on probability theory and will not be treated in detail here. The standard error of an estimate of a given quantity is the same as the *standard deviation of its sampling distribution.*

If in a sum $y = \sum_{i=1}^n x_i$ each x_i has error $|\Delta_i| \le \delta$, then the maximum error bound for y is $n\delta$. Thus, *the maximal error grows proportionally to n.* If

n is large—for example, $n = 1000$—then it is in fact highly improbable that the real error will be anywhere near $n\delta$, since that bound is attained only when every Δx_i has the same sign and the same maximal magnitude. Observe, though, that if positive numbers are added, each of which has been abridged to t decimals by chopping, then each Δx_i has the same sign and a magnitude which is on the average $\frac{1}{2}\delta$, where $\delta = 10^{-t}$. Thus, the real error is often about 500δ.

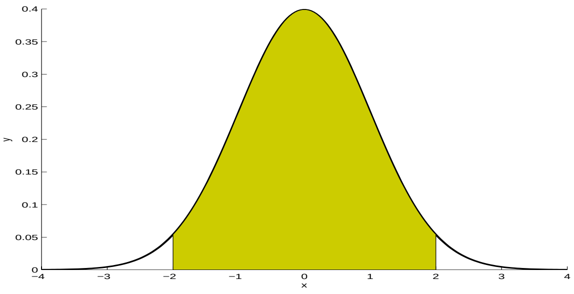


Figure 2.3.2. *The normal distribution for $\sigma = 1$.*

If the numbers are rounded instead of chopped, and if one can assume that the errors in the various terms are stochastically independent with standard deviation ϵ, then the standard error in y becomes (see Theorem 2.4.5)

$$(\epsilon^2 + \epsilon^2 + \ldots + \epsilon^2)^{1/2} = \epsilon\sqrt{n}.$$

Thus the standard error of the sum grows only proportionally to $\sqrt{n}$. This supports the following rule of thumb: *if a rounding error analysis gives a bound $Cn^p u$ for the maximum error, then one can expect the real error to be of size $Cn^{p/2}u$.*

If $n \gg 1$, then the error in y is, under the assumptions made above, approximately **normally distributed** with standard deviation $\sigma = \epsilon\sqrt{n}$. The corresponding frequency function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(1/2)x^2/\sigma^2},$$

is illustrated in Fig. 2.2.2; the curve shown there is also called the Gauss curve. The assumption that the error is normally distributed with standard deviation σ means, e.g., that the statement "the magnitude of the error is greater than 2σ" (see the shaded area of Fig. 2.2.2) is false in about only 5 % of all cases (the clear area under the curve). More generally, the assertion that the magnitude of the error is less than σ, 2σ, and 3σ respectively, is about 32%, 5%, and 0.27%.

One can show that if the individual terms in a sum $y = \sum_{i=1}^{n} x_i$ have a **uniform** probability distribution in the interval $[-\frac{1}{2}\delta, \frac{1}{2}\delta]$, then the standard deviation of an individual term is $\delta/12$. Therefore, in only about 5% of the cases is the error in the sum of $1,000$ terms greater than $2\delta\sqrt{1000/12} \approx 18\delta$, which can be compared to the maximum error 500δ. This shows that rounding can be far superior to chopping when a statistical interpretation (especially, the assumption of independence)

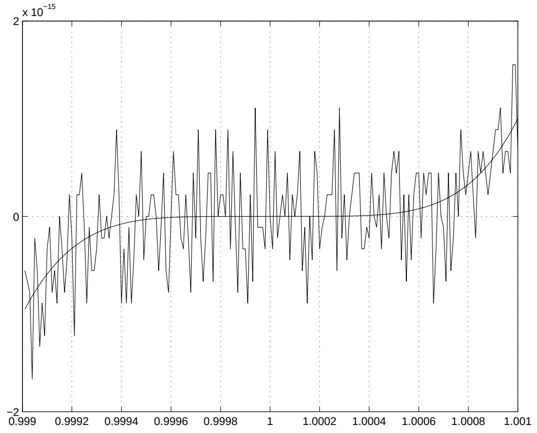


Figure 2.3.3. *Calculated values of a polynomial: dashed line* $p(x) = (x - 1)^5$*; solid line* $p(x) = x^5 - 5x^4 + 10x^3 - 10x^2 + 5x - 1 = 0$.

can be given to the principal sources of errors. Observe that, in the above, we have only considered the propagation of errors which were present in the original data, and have ignored the effect of possible round-off errors in the additions themselves.

Rounding errors are not independent random variables, but behave in a more complicated way. However, the assumption that the errors are normally distributed is justified in many computational situations and scientific experiments where the error can be considered to have arisen from the addition of a large number of *independent* error sources of about the same magnitude.

Statistical analysis of rounding errors goes back to an early paper of Goldstine and von Neumann [18, 1951]. Probabilistic models are often useful and give an adequate description of the observed behavior.

Example 2.3.5.

Figure 2.4.3 illustrates the effect of rounding errors on the evaluation of two different expressions for the polynomial $p(x) = (x - 1)^5$ for $x \in [0.999, 1.001]$, using a machine precision of about $2.2 \cdot 10^{-16}$. Among other things it shows that the monotonicity of a function can be lost due to rounding errors. The model of rounding errors as independent random variables works well in this example. It is obvious that it would be impossible to locate the zero of $p(x)$ to a precision better than about $(0.5 \cdot 10^{-14})^{1/6} \approx 0.0007$ using the expanded form of $p(x)$. However, using the expression $p(x) = (1 - x)^5$ function values can be evaluated with constant *relative* precision even close to $x = 1$, and the problem disappears!

In science and technology, one generally should be careful to discriminate between **systematic errors** and **random errors**. A systematic error can, e.g., be produced by insufficiencies in the construction of an instrument; such an error is the same in each trial. Random errors depend on the variation in the experimental environment which cannot be controlled; then the formula for standard errors is used. For systematic errors, however, the formula for maximal error (2.4.6) should be used.

2.3.5 Avoiding Overflow

In the rare cases when input and output data are so large or small in magnitude that the range of the machine is not sufficient, one can, for example, use higher precision or else work with logarithms or some other transformation of the data. One should, however, keep in mind the risk that *intermediate results* in a calculation can produce an exponent which is too large exponent overflow or too small underflow for the floating point system of the machine. Different machines take different actions in such situations, as well for division by zero. Too small an exponent is usually, but not always, un-provoking. If the machine does not signal underflow, but simply sets the result equal to zero, there is a risk of harmful consequences. Occasionally, "unexplainable errors" in output data are caused by underflow somewhere in the computations.

The **Pythagorean sum**The $c = \sqrt{a^2 + b^2}$ occurs frequently, e.g., in conversion to polar coordinates and in computing the complex modulus and complex multiplication. If the obvious algorithm is used, then damaging underflows and overflows may occur in the squaring of a and b even if a and b and the result c are well within the range of the floating point system used. This can be avoided by using instead the algorithm: If $a = b = 0$ then $c = 0$; otherwise set $p = \max(|a|, |b|)$, $q = \min(|a|, |b|)$, and compute

$$\rho = q/p; \qquad c = p\sqrt{1 + \rho^2}. \tag{2.3.15}$$

The formula (2.3.6) for complex division suffers from the problem that intermediate results can overflow even if the final result is well within the range of the floating point system. This problem can be avoided by rewriting the formula as for the Pythagorean sum:

If $|c| > |d|$ then compute

$$\frac{a + ib}{c + id} = \frac{a + be}{r} + i\frac{b - ae}{r}, \quad e = d/c, \quad r = c + de.$$

If $|d| > |c|$ then $e = c/d$ is computed and a corresponding formula used.

Similar precautions are also needed for computing the Euclidian length (norm) of a vector $\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$, $x \neq 0$. We could avoid overflows by first finding $x_{max} = \max_{1 \le i \le n} |x_i|$ and then forming

$$s = \sum_{i=1}^{n} (x_i/x_{max})^2, \qquad \|x\|_2 = x_{max}\sqrt{s}. \tag{2.3.16}$$

This has the drawback of needing *two passes* through the data. The following algorithm requiring only one pass is due to S. J. Hammarling:

$$t = 0; \quad s = 1;$$

```
for $i = 1 : n$
    if $|x_i| > t$
        $s = 1 + s(t/x_i)^2;\quad t = |x_i|;$
    else
        $s = s + s(x_i/t)^2;$
    end
end
$\|x\|_2 = t\sqrt{s};$
```

2.3.6 Cancellation of Terms

One very common reason for poor accuracy in the result of a calculation is that somewhere a subtraction has been carried out in which the difference between the operands is considerably less than either of the operands.

Consider the computation of $y = x_1 - x_2$ where $\tilde{x}_1 = x_1 + \Delta x_1$, $\tilde{x}_2 = x_2 + \Delta x_2$ are approximations to the exact values. If the operation is carried out exactly the result is $\tilde{y} = y + \Delta y$, where $\Delta y = \Delta x_1 - \Delta x_2$. But, since the errors Δx_1 and Δx_2 can have opposite sign, the best error bound for $\tilde{y}$ is

$$|\Delta y| \le |\Delta x_1| + |\Delta x_2|. \tag{2.3.17}$$

Notice the plus sign! Hence for the relative error we have

$$\left|\frac{\Delta y}{y}\right| \le \frac{|\Delta x_1| + |\Delta x_2|}{|x_1 - x_2|}. \tag{2.3.18}$$

This shows that *there can be very poor relative accuracy in the difference between two nearly equal numbers.* This phenomenon is called **cancellation of terms**.

Example 2.3.6.

For computing the roots of the quadratic equation $ax^2 + bx + c = 0$ $(a \ne 0)$ we have the well-known formula

$$r_{1,2} = \big(-b \pm \sqrt{b^2 - 4ac}\big)/(2a).$$

Hence the quadratic equation $x^2 - 56x + 1 = 0$ has the two roots

$$r_1 = 28 + \sqrt{783} \approx 28 + 27.982 = 55.982 \pm \tfrac{1}{2}10^{-3}.$$
$$r_2 = 28 - \sqrt{783} \approx 28 - 27.982 = 0.018 \pm \tfrac{1}{2}10^{-3}.$$

In spite of the fact that the square root is given to five digits, we get only two significant digits in r_2, while the relative error in r_1 is less than 10^{-5}. Notice that the subtraction in the calculation of r_2 has been carried out exactly.
The cancellation in the subtraction only gives an indication of the unhappy consequence of a loss of information in previous steps, due to the rounding of one of the operands, and is not the cause of the inaccuracy.

In general one should if possible try to avoid cancellation, as in the example above, by an appropriate **rewriting of formulas**, or by other changes in the algorithm. For the quadratic equation above, by comparing coefficients in
$x^2 + (b/a)x + c/a = (x - r_1)(x - r_2) = x^2 - (r_1 + r_2) + r_1 r_2$,
we get the dependence between coefficients and roots

$$r_1 + r_2 = -b/a, \qquad r_1 r_2 = c/a. \tag{2.3.19}$$

Computing the the root of *smaller magnitude* from the latter of these relations, we get $x_2 = 1/55.982 = 0.0178629 \pm 0.0000002$, i.e., five significant digits instead of two. In general we can avoid cancellation by using the algorithm:

Algorithm 2.3.2 Solving a quadratic equation.

```
d := b^2 - 4ac;
if $d \geq 0$ % real roots
    $r_1 := -\mathrm{sign}(b)\big(|b| + \sqrt{d}\big)/(2a)$;
    $r_2 := c/(a \cdot r_1)$;
else % complex roots $x + iy$
    $x := -b/(2a)$;
    $y := \sqrt{-d}/(2a)$;
end
```

Note that we define $\mathrm{sign}(b) = 1$, if $b \geq 0$, else $\mathrm{sign}(b) = -1$.[4] It can be shown that this algorithm computes *a slightly wrong solution to a slightly wrong problem.*

Lemma 2.3.4.

Assume that the Algorithm (2.3.6) *is used to compute the roots of the quadratic equation* $ax^2 + bx + c = 0$. *Denote the computed root* $\bar{r}_{1,2}$ *and denote by* $\tilde{r}_{1,2}$ *the exact roots of the nearby equation* $ax^2 + bx + \tilde{c} = 0$, *where* $|\tilde{c} - c| \leq \gamma_2 |\tilde{c}|$. *Then it holds that* $|\tilde{r}_i - \bar{r}_i| \leq \gamma_5 |\tilde{r}_i|$, $i = 1, 2$.

Proof. See Kahan [25]. ◻

More generally, if $|\delta| \ll x$, then one should rewrite

$$\sqrt{x+\delta} - \sqrt{x} = \frac{x + \delta - x}{\sqrt{x+\delta} + \sqrt{x}} = \frac{\delta}{\sqrt{x+\delta} + \sqrt{x}}.$$

There are other exact ways of rewriting formulas which are as useful as the above; for example,

$$\cos(x + \delta) - \cos x = -2\sin(\delta/2)\sin(x + \delta/2).$$

[4]In MATLAB $sign(0) = 0$, which can lead to failure of this algorithm!

If one cannot find an exact way of rewriting a given expression of the form $f(x+\delta)-f(x)$, it is often advantageous to use one or more terms in the Taylor series

$$f(x+\delta)-f(x)=f'(x)\delta+\frac{1}{2}f''(x)\delta^2+\cdots$$

Example 2.3.7. (Cody [10])

To compute $\sin 22$ we first find $\lfloor 22/(\pi/2)\rfloor=14$. It follows that $\sin 22=-\sin x^*$, where $x^*=22-14/(\pi/2)$. Using the correctly rounded 10 digit approximation $\pi/2=1.57079\,6327$ we obtain

$$x^*=22-1.57079\,6327=8.85142\cdot 10^{-3}.$$

Here cancellation has taken place and the reduced argument has a maximal error of $7\cdot 10^{-9}$, The actual error is slightly smaller since the correctly rounded value is $x^*=8.85144\,8711\cdot 10^{-3}$ which corresponds to a relative error in the computed $\sin 22$ of about $2.4\cdot 10^{-6}$ in spite of using a ten digit approximation to $\pi/2$.

For very large arguments the relative error can be much larger. Techniques For carrying out accurate range reductions without actually needing multiple precision calculations are discussed by Muller [32].

In previous examples we got a warning that cancellation would occur, since x_2 was found as the difference between two nearly equal numbers each of which was, relatively, much larger than the difference itself. In practice, one does not always get such a warning, for two reasons: first, in using a computer one has no direct contact with the individual steps of calculation; secondly, cancellation can be spread over a great number of operations. This may occur in computing a partial sum of an infinite series. For example, in a series where the size of some terms are many order of magnitude larger than the sum of the series small relative errors in the computation of the large terms can then produce large errors in the result.

Example 2.3.8.

Set $y_0=28$ and define y_n, for $n=1:100$, by the recursion formula:

$$y_n=y_{n-1}-\sqrt{783}/100.$$

As previously, we use the approximate value 27.982 for the square root. We then compute with five decimals in each operation in order to make the effect of further rounding errors negligible. We get the same bad value for y_{100} that we got for x_1 in the previous example. Of all the subtractions, only the last would lead one to suspect cancellation, $y_{100}=0.29782-0.27982=0.01800$, but this result in itself gives one no reason to suspect that only two digits are significant. (With four significant digits, the result is 0.01786.)

2.3.7 Automatic Control of Accuracy

A different approach to rounding error analysis is to perform the analysis automatically, for *each particular computation*. This gives an *a posteriori* error analysis as compared to the *a priori* error analysis discussed above.

A simple form of a posteriori analysis, called running error analysis, was used in the early days of computing, see Wilkinson [38]. To illustrate his idea we rewrite the basic model for floating point arithmetic as

$$x \operatorname{op} y = fl\,(x \operatorname{op} y)(1+\epsilon).$$

These are also satisfied for most implementations of floating point arithmetic. Then, the actual error can be estimated $|fl\,(x \operatorname{op} y) - x \operatorname{op} y| \le u|fl\,(x \operatorname{op} y)|$. Note that the error is now given in terms of the *computed* result and is available in the computer at the time the operation is performed. This running error analysis can often be easily implemented. We just take an existing program and modify it, so that as each arithmetic operation is performed, the absolute value of the computed results is added into the accumulating error bound.

Example 2.3.9.

The inner product $fl\,(x^Ty)$ is computed by the program

```
$s = 0$;    $\eta = 0$;
**for** $i = 1, 2, \dots, n$
    $t = fl\,(x_iy_i)$;    $\eta = \eta + |t|$;
    $s = fl\,(s+t)$;    $\eta = \eta + |s|$;
**end**
```

For the final error we have the estimate $|fl\,(x^Ty) - x^Ty| \le \eta u$. Note that a running error analysis takes advantage of cancellations in the sum. This is in contrast to the previous estimates, which we call a priori error analysis, where the error estimate is the same for all distribution of signs of the elements x_i and y_i.

Efforts have been made to design the computational unit of a computer so that it gives, in every arithmetic operation, only those digits of the result which are judged to be significant (possibly with a fixed number of extra digits), so-called *unnormalized floating arithmetic*. This method reveals poor construction in algorithms, but in many other cases it gives a significant and unnecessary loss of accuracy. The mechanization of the rules, which a knowledgeable and experienced person would use for control of accuracy in hand calculation, is not as free from problems as one might expect. As complement to arithmetical operations of conventional type, the above type of arithmetic is of some interest, but it is doubtful that it will ever be widely used.

A fundamental difficulty in automatic control of accuracy is that to decide how many digits is needed in a quantity to be used in later computation, one needs to consider the *entire context of the computations*. It can in fact occur that the errors in many operands depend on each other in such a way that they cancel each other. Such **cancellation of error**, is a completely different phenomenon from the previously discussed cancellation of terms, is most common in larger problems, but will be illustrated here with a simple example.

Example 2.3.10.

Suppose we want to compute $y = z_1 + z_2$, where $z_1 = \sqrt{x^2+1}$, $z_2 = 200 - x$, $x = 100 \pm 1$, with a rounding error which is negligible compared to that resulting from the errors in z_1 and z_2. The best possible error bounds in the intermediate results are $z_1 = 100 \pm 1$, $z_2 = 100 \pm 1$. It is then tempting to be satisfied with the result $y = 200 \pm 2$.

However, the errors in z_1 and z_2 due to the uncertainty in x will, to a large extent, cancel each other! This becomes clear if we rewrite the expression as

$$y = 200 + (\sqrt{x^2+1} - x) = 200 + \frac{1}{\sqrt{x^2+1}+x}.$$

It follows that $y = 200 + u$, where $1/202 \lessapprox u \le 1/198$. Thus y can be computed with an absolute error less than about $2/(200)^2 = 0.5 \cdot 10^{-4}$. Therefore using the expression $y = z_1 + z_2$ the intermediate results z_1 and z_2 should be computed to four decimals even though the last integer in these is uncertain! The result is $y = 200.0050 \pm \frac{1}{2}10^{-4}$.

In larger problems, such a cancellation of errors can occur even though one cannot easily give a way to rewrite the expressions involved. The authors have seen examples where the final result, a sum of seven terms, was obtained correctly to eight decimals even though the terms, which were complicated functions of the solution to a system of nonlinear equations with fourteen unknowns, were correct only to three decimals.!

2.3.8 Interval Arithmetic

In **interval arithmetic** one assumes that all input values are given as intervals and systematically calculates an inclusion interval for each intermediate result. It is partly an automatization of calculation with maximal error bounds.

The most frequently used representations for the intervals are the **infimum-supremum** representation

$$I = [a, b] := \{x \mid a \le x \le b\}, \quad (a \le b). \tag{2.3.20}$$

The **magnitude** of an interval is defined as

$$|[a, b]| \max\{|x| \mid x \in [a, b]\} = \max\{|a|, |b|\}. \tag{2.3.21}$$

The **midpoint** and the **radius** of the interval $[a, b]$ is defined as

$$\mathrm{mid}\,([a, b]) = \frac{1}{2}(a + b), \qquad \mathrm{rad}\,([a, b]) = \tfrac{1}{2}(b - a). \tag{2.3.22}$$

An alternative representation for an interval is the **midpoint-radius** representation, for which we use brackets

$$\langle a, \rho \rangle := \{x \mid |x - a| \le r\} \quad (0 \le r). \tag{2.3.23}$$

The result of an interval operation equals the the range of the corresponding real operation. For example, the difference between the intervals $[a_1, a_2]$ and $[b_1, b_2]$, is defined as the shortest interval which contains all the numbers $x_1 - x_2$, where $x_1 \in [a_1, a_2]$, $x_2 \in [b_1, b_2]$, i.e. $[a_1, a_2] - [b_1, b_2] := [a_1 - b_2, a_2 - b_1]$. Other elementary interval arithmetic operations are similarly defined:

$$[a_1, a_2] \operatorname{op} [b_1, b_2] := \{x_1 \operatorname{op} x_2 \mid x_1 \in [a_1, a_2],\ x_2 \in [b_1, b_2]\}, \tag{2.3.24}$$

where op $\in \{+, -, \cdot, \operatorname{div}\}$. The interval value of a standard function ϕ (e.g., $\sin, \cos, \exp, \log$) evaluated on an interval is defined as

$$\phi([a, b])[\min_{x \in [a,b]} \phi(x), \max_{x \in [a,b]} \phi(x)].$$

Although (2.3.24) characterizes interval arithmetic operations we also need **operational definitions**. We take

$$\begin{aligned}
&[a_1, a_2] + [b_1, b_2] = [a_1 + b_1, a_2 + b_2],\\
&[a_1, a_2] - [b_1, b_2] = [a_1 - b_2, a_2 - b_1],\\
&[a_1, a_2] \cdot [b_1, b_2] = \big[\min\{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\}, \max\{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\}\big],\\
&1/[a_1, a_2] = [1/a_2, 1/a_1], \quad \text{for} \quad a_1 a_2 > 0,\\
&[a_1, a_2]/[b_1, b_2] = [a_1, a_2] \cdot (1/[b_1, b_2]).
\end{aligned} \tag{2.3.25}$$

It is easy to prove that in exact interval arithmetic the operational definitions above give the exact range (2.3.24) of the interval operations. Division by an interval containing zero is not defined.

For intervals in the midpoint-radius representation $\langle \cdot \rangle$ we take

$$\begin{aligned}
&\langle a_1, r_1 \rangle + \langle a_2, r_2 \rangle = \langle a_1 + a_2, r_1 + r_2 \rangle,\\
&\langle a_1, r_1 \rangle - \langle a_2, r_2 \rangle = \langle a_1 + a_2, r_1 + r_2 \rangle,\\
&\langle a_1, r_1 \rangle \times \langle a_2, r_2 \rangle = \langle a_1 a_2, |a_1| r_2 + r_1 |a_2| + r_1 r_2 \rangle,\\
&1/\langle a, r_1 \rangle = \langle 1/r_1, 1/a_1 \rangle, \quad \text{for} \quad a_1 r_1 > 0,\\
&\langle a_1, r_1 \rangle / \langle a_2, r_2 \rangle = \langle a_1, r_1 \rangle \cdot (1/\langle a_2, r_2 \rangle).
\end{aligned} \tag{2.3.26}$$

For addition, subtraction and inversion, these operational definitions give the exact range. For multiplication and division they overestimate the range. For multiplication we have for any $x_1 \in \langle a_1, r_1 \rangle$ and $x_2 \in \langle a_2, r_2 \rangle$

$$\begin{aligned}
|x_1 x_2 - a_1 a_2| &= |a_1(x_2 - a_2) + a_2(x_1 - a_1) + (x_1 - a_1)(x_2 - a_2)|\\
&\le |a_1| r_2 + |a_2| r_1 + r_1 r_2.
\end{aligned}$$

A degenerate interval with radius zero is called a point interval and can be identified with a single number $a \equiv [a, a]$. In this way the usual arithmetic is recovered as a special case. The intervals $0 = [0,0]$ and $1 = [1,1]$ are the neutral elements with respect to interval addition and interval multiplication, respectively. A nondegenerate interval has no inverse with respect to addition or multiplication For example, we have

$$[1,2] - [1,2] = [-1,1], \qquad [1,2]/[1,2] = [0.5,2].$$

For interval operations the commutative law $[a_1,a_2] \operatorname{op} [b_1,b_2] = [b_1,b_2] \operatorname{op} [c_1,c_2]$ holds. However, the distributive law has to be replaced by so called **subdistributivity**

$$[a_1,a_2]([b_1,b_2]+[c_1,c_2]) \subseteq [a_1,a_2]([b_1,b_2]+[c_1,c_2]). \tag{2.3.27}$$

This unfortunately means that expressions, which are equivalent in real arithmetic, differ in exact interval arithmetic. The simple example

$$[-1,1]([1,1]+[-1,-1]) = 0 \subset [-1,1][1,1]+[-1,1][-1,-1] = [-2,2]$$

shows that $-[-1,1]$ is not the additive inverse to $[-1,1]$ and also illustrates (2.3.27).

The operations introduced are **inclusion monotonic**, i.e,

$$[a_1,a_2] \subseteq [c_1,c_2], \quad [b_1,b_2] \subseteq I_d \quad \Rightarrow \quad [a_1,a_2] \operatorname{op} [b_1,b_2] \subseteq [c_1,c_2] \operatorname{op} I_d. \tag{2.3.28}$$

In implementing interval arithmetic using floating point arithmetic the operational interval results may not be exactly representable as floating point numbers. Then if the lower bound is rounded down to the nearest smaller machine number and the upper bound rounded up, the exact result must be contained in the resulting interval. Recall that these rounding modes (rounding to $-\infty$ and $+\infty$) are supported by the IEEE 754 standard. For example, using 5 significant decimal arithmetic, we would like to get

$$[1,1] + [-10^{-10}, 10^{-10}] = [0.99999, 1.0001] = < 1, 10^{-10} > .$$

Note that in the midpoint-radius representation there is no roundoff. The conversion between the infimum-supremum representation is straight forward but the midpoint may not be exactly representable.

One use of interval arithmetic is to enclose the range of a real valued function. This can be used, e.g., for localizing and enclosing global minimizers and global minima of a real function of one or several variables in a region. It can also be used for verifying the nonexistence of a zero of $f(x)$.

Let $f(x)$ be a real function composed of a finite number of elementary operations and standard functions. If one replaces the variable x by an interval $[\underline{x_i}, \overline{x_i}]$ and evaluates the resulting interval expression one gets as result an interval $f([\underline{x_i}, \overline{x_i}])$. (We assume that all operations can be carried out.) A fundamental result is that this evaluation is inclusion monotonic, i.e.,

$$[\underline{x}, \overline{x}] \subseteq [\underline{y}, \overline{y}], \quad \Rightarrow \quad f([\underline{x}, \overline{x}]) \subseteq f([\underline{y}, \overline{y}]).$$

In particular this means that $x \subseteq [x] \Rightarrow f(x) \subseteq f([x])$, i.e., $f([x])$ contains the range of $f(x)$ over the interval $[x]$. A similar result holds also for functions of several variables $f(x_1, \dots, x_n)$.

There is an important case when interval evaluation gives the exact range of a function. This is when $f(x_1,\dots,x_n)$ is a rational expression, where each variable x_i occurs only once in the function.

Example 2.3.11. In general narrow bounds cannot be guaranteed. For example, if $f(x) = x/(1-x)$ then

$$f([2,3]) = [2,3]/(1-[2,3]) = [2,3]/[-2,-1] = [-3,-1].$$

The result contains but does not coincide with the exact range $[-2,-3/2]$. However, if we rewrite the expression as $f(x) = 1/(1/x-1)$, where x only occurs once, then we get

$$f([2,3]) = 1/(1/[2,3]-1) = 1/[-2/3,-1/2] = [-2,-3/2],$$

which is the exact range.

A complex interval In the infimum-supremum representation

$$[z_1,z_2] = \{x+iy \mid x \in [x_1,x_2],\ y \in [y_1,y_2]\}$$

is a *rectangle in the complex plane* defined by the two real intervals,

$$[z_1,z_2] = [x_1,x_2] + i[y_1,y_2], \quad x_1 \le x_2, \quad y_1 \le y_2.$$

This can be written more compactly as $[z_1,z_2] := \{z \mid z_1 \le z \le z_2\}$, where we use the partial ordering

$$z \le w \quad \Leftrightarrow \quad \Re z \le \Re w \ \ \& \ \ \Im z \le \Im w.$$

Complex interval operations are defined in terms of the real intervals $[x] = [x_1,x_2]$ and $[y] = [y_1,y_2]$ in the same way as the complex operations are defined for complex numbers $z = x+iy$. For addition and subtraction the result coincides with the exact range. This is not the case for complex interval multiplication, where the result

$$([x]+i[y])\cdot([u]+i[v]) = [x]\cdot[u] + [y]\cdot[v] + i\big([x]\cdot[v] - [y]\cdot[u]\big).$$

is a rectangle in the complex plane, whereas the actual range is not of this shape. Therefore multiplication will for complex intervals result in an overestimation.

In the complex case the midpoint-radius representation is

$$\langle w,r\rangle := \{z \in \mathbf{C} \mid |z-w| \le r\}, \quad 0 \le r,$$

where the midpoint w is a complex number and $|z-w|$ the magnitude of the complex number $z-w$. This is a *disk in the complex plane*. The operational definitions (2.3.26) now generalize directly. Interval arithmetic using the midpoint-radius representation is therefore also called **circular arithmetic**. For complex multiplications it generates less overestimation than the infimum-supremum notation. In the following, for simplicity, we only consider real interval arithmetic.

When interval analysis is used in a naive manner as a simple technique for simulating forward error analysis it does not in general give sharp bounds on the total computational error. To get useful results the computations in general need to be arranged so that overestimation as far as possible is minimized. Often a refined design of the algorithm is required in order to prevent the bounds for the intervals from becoming unacceptably coarse. The answer $[-\infty, \infty]$ is of course always correct but not at all useful!

Example 2.3.12. Evaluate for $[x] = [2, 3]$ the cubic polynomial

$$p(x) = 1 - x + x^2 - x^3 + x^4 - x^5$$

Using exact interval arithmetic we find $p([2, 3]) = [-252, 49]$. (verify this!) This is an overestimate of the exact range, which is $[-182, -21]$. Rewriting $p(x)$ in the form $p(x) = (1 - x)(1 + x^2 + x^4)$ we obtain the correct range. In the first example there is a **cancellation of errors** in the intermediate results, which is not revealed by the interval calculations.

The remainder term in Taylor expansions includes a variable ξ, which is known to lie in an interval $\xi \in [a, b]$. This makes it suitable to treat the remainder term with interval arithmetic.

Let $f(x)$ be a twice differentiable function in an interval $[x_0]$. Enclosures of a real simple zero $x^* \in [x_0]$ of $f(x)$ may be computed by a variant of Newton's method. By the mean value theorem

$$0 = f(\tilde{x}) + f'(\xi)(x^* - \tilde{x}),$$

for some ξ between x^* and $\tilde{x}$. If $\tilde{x}$ and the root x^* both lie in the interval $[x_0]$ then so does ξ. If $0 \notin f'([x_k])$ it follows that

$$x^* \in N([x_0]) = \tilde{x} - \frac{f(\tilde{x})}{f'([x_0])},$$

and hence the intersection $N([x_0]) \cap [x_0]$, must contain a root. By iterating this procedure we can to compute a sequence of intervals $[x_k]$ containing a simple root x^*.

Interval Newton method

Given a starting interval $[x_0]$ enclosing a simple root, compute for $k = 0, 1, 2, \ldots$ the sequence of intervals $[x_{k+1}]$ given by

$$N([x_k]) = \operatorname{mid}([x_k]) - \frac{f(m[x_k])}{f'([x_k])}, \quad [x_{k+1}] = N([x_k]) \cap [x_k],$$

It is assumed that $0 \notin f'([x_k])$ since otherwise the computation breaks down. Moore [29] shows that if $N([x_k])$ is defined and $N([x_k]) \cap [x_k]$ is empty the $[x_k]$ does not contain a root. Further, if $[x_0]$ does not contain a root then after a finite number of steps the iteration will stop with an empty interval.

Example 2.3.13. Take $f(x) = x^2 - 2$ and $[x_0] = [1, 2]$. Using interval Newton method

$$N([x_k]) = m([x_k]) - \frac{(m[x_k])^2 - 2}{2\,[x_k]}, \quad [x_{k+1}] = N([x_k]) \cap [x_k].$$

we obtain the sequence of intervals

$$[x_1] = N([x_0]) = 3/2 - 1/4/[2, 4] = [22/16, 23/16]\,,$$
$$[x_2] = N([x_1]) = \frac{45}{32} - \frac{(45/32)^2 - 2}{2[22/16, 23/16]} = \frac{45}{32} - \frac{(45)^2 - 2(32)^2}{128\,[22, 23]} \subset [1.41406, 1.41442].$$

The radius of the intervals can be shown to converge quadratically to zero.

In order to treat multidimensional problems interval vectors $[x] = ([x_i])$ with interval components $[x_i]$, $i = 1 : n$ and interval matrices $[A] = ([a_{ij}])$ with interval elements $[a_{ij}]$, $i = 1 : m$, $j = 1 : n$, are introduced.

Operations between interval matrices and interval vectors are defined in an obvious manner. The interval matrix-vector product $[A][x]$ is the smallest interval vector, which contains the set $\{Ax \mid A \in [A],\ x \in [x]\}$, but normally does not coincide with it. We have the inclusion property

$$\{Ax \mid A \in [A],\ x \in [x]\} \subseteq [A][x] = \left(\sum_{j=1}^{n} [a_{ij}][x_j]\right).$$

In general there will be an overestimation in enclosing the image with an interval vector caused by the fact that the image of an interval vector under a transformation in general is not an interval vector, This phenomenon, intrinsic to interval computations, is called the **wrapping effect**.

Example 2.3.14. We have

$$[A] = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad [x] = \begin{pmatrix} [0, 1] \\ [0, 1] \end{pmatrix}, \quad \Rightarrow \quad [A][x] = \begin{pmatrix} [0, 2] \\ [-1, 1] \end{pmatrix}.$$

Hence $b = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \in [A][x]$, but there is no $x \in [x]$ such that $Ax = b$.

The magnitude of an interval vector or matrix is interpreted componentwise and defined by

$$|\,[x]\,| = (|\,[x_1]\,|, |\,[x_2]\,|, \ldots, |\,[x_n]\,|)^T,$$

where the magnitude of the components are defined by (2.3.21). The ∞-norm of a vector or matrix is defined as the ∞-norm norm of ∞-norm of their magnitude,

$$\|\,[x]\,\|_\infty = \|\,|\,[x]\,|\,\|_\infty, \qquad \|\,[A]\,\|_\infty = \|\,|\,[A]\,|\,\|_\infty. \tag{2.3.29}$$

We now consider the implementation of matrix multiplication. It is important to avoid case distinctions in the inner loop, because that would make it impossible to use fast vector an matrix operation. Interval arithmetic it is possible to compute strict enclosures of the product of two matrices. Note that this is needed also in the case of the product of two point matrices since rounding errors will occur. Then we want to compute an interval matrix $[C]$ such that $fl(A\cdot B)\subset[C]=[C_{\text{inf}},C_{\text{sup}}]$. The following simple code does that using two matrix multiplications:

$$\begin{aligned}&\text{setround}(-1);\quad C_{\text{inf}}=A\cdot B;\\&\text{setround}(1);\qquad C_{\text{sup}}=A\cdot B;\end{aligned}$$

Here and in the following we assume that switching rounding mode is performed by the command setround(i), $i=-1,0,1$ corresponding to rounding to $-\infty$, to nearest, and to ∞, respectively.

We next consider the product of a point matrix A and an interval matrix $[B]=[B_{\text{inf}},B_{\text{sup}}]$. The following code, suggested by A. Neumeier, performs this using four matrix multiplications:

$$\begin{aligned}&A_-=\min(A,\ 0);\quad A_+=\max(A,\ 0);\\&\text{setround}(-1);\\&C_{\text{inf}}=A_+\cdot B_{\text{inf}}+A_-\cdot B_{\text{sup}};\\&\text{setround}(1);\\&C_{\text{sup}}=A_-\cdot B_{\text{inf}}+A_+\cdot B_{\text{sup}};\end{aligned}$$

For an algorithm using eight matrix multiplications for computing the product of two interval matrices, see Rump [35]. There also several faster implementation are given, provided a certain overestimation can be allowed.

A square interval matrix $[A]$ is called nonsingular if it does not contain a singular matrix. An interval linear system is a system of the form $[A]\,x=[b]$, where A is a nonsingular interval matrix and b an interval vector. The solution set of such an interval linear system is the set

$$\mathcal{X}=\{x\mid Ax=b,\ A\in[A],\ b\in[b]\}. \tag{2.3.30}$$

Computing this solution set can be shown to be an intractable problem (NP-complete). Even for a 2×2 linear system this set may not be easy to represent.

Example 2.3.15. (Hansen [19, Chapter 4])

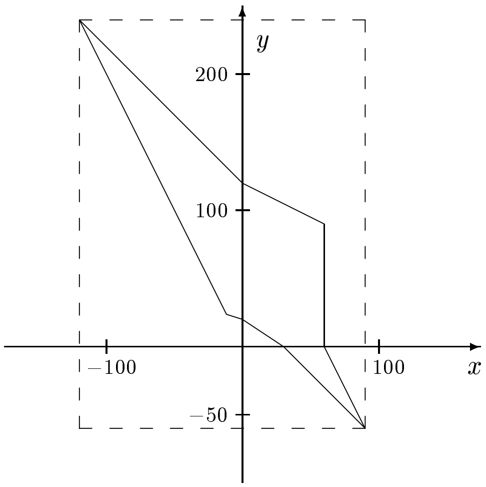
Consider a linear system with

$$[A]=\begin{pmatrix}[2,3]&[0,1]\\ [1,2]&[2,3]\end{pmatrix},\quad [b]=\begin{pmatrix}[0,120]\\ [60,240]\end{pmatrix}. \tag{2.3.31}$$

The solution set $\mathcal{X}$ in (2.3.30) is the star shaped region in Figure 2.3.4.

An enclosure of the solution set of an interval linear system can be computed by a generalization of Gaussian elimination adopted to interval coefficients.. A triangular system is computed in interval arithmetic. The solution of this will give an inclusion of the solution set. Realistic bounds can obtained in this way for

Figure 2.3.4. *The solution set (solid line) and its enclosing hull (dashed line) for the linear system in Example* 2.3.16.



special classes of matrices, e.g., for diagonally dominant matrices and tridiagonal matrices; see Hargreaves [20]. For general systems this approach unfortunately tends to give interval sizes which grow exponentially during the elimination. Even for well-conditioned linear systems the elimination can break down prematurely, because all remaining possible pivot elements contain zero.

For example, if $[x]$ and $[y]$ are intervals then in the 2×2 reduction

$$\begin{pmatrix} 1 & [x] \\ 1 & [y] \end{pmatrix} \sim \begin{pmatrix} 1 & [x] \\ 0 & [y]-[x] \end{pmatrix}.$$

If $[x] \approx [y]$ the size of the interval $[y]-[x]$ will be twice the size of $[x]$. This growth is very likely to happen

Verified bounds on a point or interval linear system can be computed using an idea that goes back to E. Hansen [1965] of using an approximate inverse to precondition the system with an approximate inverse C.

Assume that an initial interval $[x^{(0)}] \subseteq \mathcal{X}$ is known, where $\mathcal{X}$ is the solution set (2.3.30). An improved enclosure can then be obtained by **Krawczyck's method**. For all $\tilde{A} \in [A]$ and $\tilde{b} \in [b]$ it holds that

$$\tilde{A}^{-1}\tilde{b} = C\tilde{b} + (I - C\tilde{A})\tilde{A}^{-1}\tilde{b} \in C\,[b] + (I - C\,[A])[x^{(0)}] =: [x^{(1)}].$$

This suggests the iteration

$$[x^{(i+1)}] = \Big(C\,[b] + (I - C[A])[x^{(i)}]\Big) \cap [x^{(i)}], \quad i = 0,1,2,\dots, \tag{2.3.32}$$

for computing a sequence of interval enclosures $[x^{(i)}]$ of the solution. The interval vector $[c] = C\,[b]$ and interval matrix $[E] = I - C\,[A]$ need to be computed only once. The cost per iteration therefore is about one interval matrix times interval vector multiplication.

As approximate inverse we can take the inverse of the midpoint matrix $C = (\text{mid}\,[A])^{-1}$. An initial interval can be chosen of the form $[x^{(0)}] = C\text{mid}\,[b] + \beta[-1,1]$, with β sufficiently large. The iterations are terminated when the bounds are no longer improving. A measure of convergence can be computed as $\rho = \|[E]\|_\infty$.

Rump [35, 34] has developed a MATLAB toolbox INTLAB (INTerval LABoratory). This is very efficient and easy to use and includes many useful subroutines. INTLAB uses a variant of Krawczyck's method applied to a residual system to compute an enclosure of the difference between the solution and an approximate solution $x_m = C\text{mid}\,[b]$; see Rump [35].

Example 2.3.16. A method for computing an enclosure for the inverse of an interval matrix can be obtained by taking $[b]$ equal to the identity matrix and solving the system $[A][X] = I$. For the symmetric interval matrix

$$[A] = \begin{pmatrix} [0.999, 1.01] & [-0.001, 0.001] \\ [-0.001, 0.001] & [0.999, 1.01] \end{pmatrix}$$

the identity $C = \text{mid}\,[A] = I$ is an approximate point inverse, and we find

$$[E] = I - C[A] = \begin{pmatrix} [-0.01, 0.001] & [-0.001, 0.001] \\ [-0.001, 0.001] & [-0.01, 1.001] \end{pmatrix}.$$

An enclosure for the inverse matrix is given by $[X^{(0)}] = [0.9, 1.1][A]$,

$$[X^{(i+1)}] = \left(I + E[X^{(i)}]\right) \cap [X^{(i)}], \quad i = 0, 1, 2, \ldots.$$

The iteration converges rapidly in this case.

Another application of interval arithmetic is to initial value problems for ordinary differential equations

$$y' = f(x, y), \quad y(x_0) = y_0, \quad y \in \mathbf{R}^n.$$

Interval techniques can be used to provide for errors in the initial values, as well as truncation and rounding errors, so that at each step intervals are computed that contain the actual solution. However, it is a most demanding task to construct an interval algorithm for the initial value problem, and they tend to be significantly slower than corresponding point algorithms. One problem is that a wrapping effect occurs at each step and causes the computed interval widths to grow exponentially. This is illustrated in the following example.

Example 2.3.17.

The recursion formulas $x_{n+1} = (x_n - y_n)/\sqrt{2}$, $y_{n+1} = (x_n + y_n)/\sqrt{2}$, mean a series of 45-degree rotations in the xy-plane (see Fig. 2.3.4). By a two-dimensional interval one means a rectangle whose sides are parallel to the coordinate axes. If

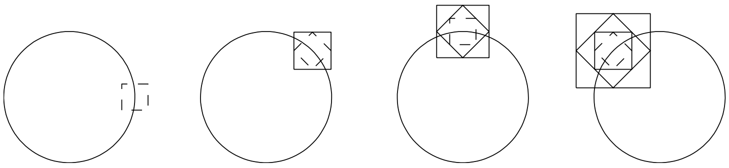


Figure 2.3.5. *Wrapping effect in interval analysis.*

(x_0, y_0) is given as some interval $|x_0 - 1| \leq \epsilon$, $|y_0| \leq \epsilon$ (see the dashed square, in the leftmost portion of Fig. 2.4.1), then (x_n, y_n) will, with *exact* performance of the transformations, also be a square with side 2ϵ, for all n (see the other squares in Fig. 2.4.1). If the computations are made using interval arithmetic, rectangles with sides parallel to the coordinate axis will, in each step, be circumscribed about the exact image of the interval one had in the previous step. Thus the interval is is multiplied by $\sqrt{2}$ in each step. After forty steps, for example, the interval has been multiplied by $2^{20} > 10^6$.

Review Questions

1. What is the standard model for floating point arithmetic? What weaker model holds if a guard digit is lacking?

2. Give examples to show that some of the axioms for arithmetic with real numbers do not always hold for floating point arithmetic.

3. (a) Give the results of a backward and forward error analysis for computing $fl\,(x_1 + x_2 + \cdots + x_n)$. It is assumed that the standard model holds.

(b) Describe the idea in compensated summation.

4. Explain the terms "maximum error" and "standard error". What statistical assumption about rounding errors is often made, for example, when calculating the standard error in a sum due to rounding?

5. Explain, what is meant by "cancellation of terms". Give an example how this can be avoided by rewriting a formula.

6. Describe two possibilities of representing intervals in interval arithmetic.

Problems

1. (a) The expression $x^2 - y^2$ exhibits catastrophic cancellation if $|x| \approx |y|$. Show that it is more accurate to evaluate it as $(x + y)(x - y)$.

(b) Consider using the trigonometric identity $\sin^2 x + \cos^2 x = 1$ to compute $\cos x = (1 - \sin^2 x)^{1/2}$. For which arguments in the range $0 \leq x \leq \pi/4$ will this formula fail to give good accuracy?

2. The polar representation of a complex number is

$$z = x + iy = r(\sin\phi + \cos\phi) \equiv r \cdot e^{i\phi}.$$

Develop accurate formulas for computing this polar representation from x and y using real operations.

3. Suppose that the sum $s = \sum_{i=1}^{n} x_i$, $n = 2^k$, is computed using the the divide and conquer technique described in Sec. 1.5. Show that this summation algorithm computes an exact sum

$$\bar{s} = \sum_{i=1}^{n} x_i(1+\delta_i), \quad |\delta_i| \le \tilde{u}\log_2 n.$$

Hence for large values of n this summation order can be much more accurate than the conventional order.

4. Show that for the evaluation of a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$ by Horner's rule the following roundoff error estimate holds:

$$|fl\,(p(x)) - p(x)| < \gamma_1 \sum_{i=0}^{n} (2i+1)|a_i|x|^i, \quad (2nu \le 0.1).$$

5. In solving linear equations by Gaussian elimination there often occurs expressions of the form $s = (c - \sum_{i=1}^{n-1} a_i b_i)/d$. Show that by a slight extension of the result above shows that the computed $\bar{s}$ satisfies

$$\left|\bar{s}d - c + \sum_{i=1}^{n-1} a_i b_i\right| \le \gamma_n\left(|\bar{s}d| + \sum_{i=1}^{n-1} |a_i||b_i|\right),$$

where the inequality holds independent of the summation order.

6. One has measured two sides and the included angle of a triangle to be $a = 100.0 \pm 0.1$, $b = 101.0 \pm 0.1$, and the angle $C = 1.00^o \pm 0.01^o$. Then the third side is given by the cosine theorem

$$c = (a^2 + b^2 - 2ab\cos C)^{1/2}.$$

(a) How accurately is it possible to determine c from the given data?

(b) How accurately does one get c if one uses the value $\cos 1^o = 0.9998$, which is correct to four decimal places.

(c) Rewrite the cosine theorem so that it is possible to compute c to full accuracy using only a four-decimal table for the trigonometric functions.

7. (W. Kahan [1983]) The area A of a triangle with sides equal to a, b, c is given by Heron's formula

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = (a+b+c)/2.$$

Show that this formula fails for needle-shaped triangles, using five digit decimal floating arithmetic and $a = 100.01$, $b = 99.995$, $c = 0.025$.
The following formula can be proved to work if addition/subtraction satisfies (2.3.15): Order the sides so that $a \ge b \ge c$, and use

$$A = \frac{1}{4}\sqrt{(a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c))}.$$

Compute a correct result for the data above using this modified formula. If a person tells you that this gives an imaginary result if $a - b > c$, what do you answer him?

8. Carry out the following calculations in exact interval arithmetic:

(a) $[0, 1] + [1, 2]$; (b) $[3, 3.1] - [0, 0, 2]$; (c) $[-4. - 1] \cdot [-6, 5]$;

(d) $[2, 2] \cdot [-1, 2]$; (e) $[-1, 1]/[-2, -0.5]$; (f) $[-3, 2] \cdot [-3.1, 2.1]$;

9. Treat the Example 1.3.2 using interval analysis and four decimal digits. Starting with $I_{10} < [0, 1/60] == [0, 0.01667]$ generate successively I_k $k = 9 : -1 : 5$ using interval arithmetic and the recursion $I_{n-1} = 1/(5n) - I_n/5$.

Computer Exercises

1. (a) To compute $\ln(1 + x)$ for $0 < x \ll 1$, the Mclaurin series $\ln(1 + x) = x - x^2/2 + x^3/3 - x^4/4 + x^5/5 + \cdots$ is useful. How many terms in the series are needed to get IEEE 754 double precision accuracy for all $x < 10^{-3}$?

(b) Show that $\ln(1 + x) = \ln(1 + y) - \ln(1 - y)$, where $y = (x/2)/(1 + x/2)$, and deduce that $\ln(1 + x) = 2(y + y^3/3 + y^5/5 + \cdots$. How many terms in this series are needed for the same computation as in (a)?

Hint: Assume that the error from truncating the series can be estimated by the first neglected term.

2. In the statistical treatment of data, one often needs to compute the quantities

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

If the numbers x_i are the results of statistically independent measurements of a quantity with expected value m, then $\bar{x}$ is an estimate of m, whose standard deviation is estimated by $s/\sqrt{n-1}$.

(a) The computation of $\bar{x}$ and m using the formulas above have the drawback that they require two passes through the data x_i. Let α be *a provisional mean*, chosen as an approximation to $\bar{x}$, and set $x_i' = x_i - \alpha$. Show that the formulas

$$\bar{x} = \alpha + \frac{1}{n}\sum_{i=1}^{n} x_i', \qquad s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i')^2 - (\bar{x} - \alpha)^2.$$

hold for an arbitrary α.

(b) In sixteen measurements of a quantity x one got the following results:

i	x_i	i	x_i	i	x_i	i	x_i
1	546.85	5	546.81	9	546.96	13	546.84
2	546.79	6	546.82	10	546.94	14	546.86
3	546.82	7	546.88	11	546.84	15	546.84
4	546.78	8	546.89	12	546.82	16	546.84

Compute $\bar{x}$ and s^2 to two significant digits using $\alpha = 546.85$.

(c) In the computations in (b), one never needed more than three digits. If one uses the value $\alpha = 0$, how many digits is needed in $(x_i')^2$ in order to get two significant digits in s^2? If one uses five digits throughout the computations, why is the cancellation in the s^2 more fatal than the cancellation in the subtraction $x_i' - \alpha$? (one can even get negative values for s^2!)

(d) If we define

$$m_k = \frac{1}{k}\sum_{i=1}^{k} x_i, \qquad q_k = \sum_{i=1}^{k}(x_i - m_k)^2 = \sum_{i=1}^{k} x_i^2 - \frac{1}{k}\Big(\sum_{i=1}^{k} x_i\Big)^2,$$

then it holds that $\bar{x} = m_n$, and $s^2 = q_n/n$. Show the recursion formulas

$$\begin{aligned} m_1 &= x_1, \qquad & m_k &= m_{k-1} + (x_k - m_{k-1})/k \\ q_1 &= 0, \qquad & q_k &= q_{k-1} + (x_k - m_{k-1})^2(k-1)/k \end{aligned}$$

3. Compute the sum in Example 2.3.4 using the natural summation ordering in IEEE 754 double precision. Repeat the computations using compensated summation Algorithm 2.2.1.

2.4 Error Propagation and Condition Numbers

2.4.1 Numerical Problems, Methods and Algorithms

By a **numerical problem** we mean here a clear and unambiguous description of the *functional connection* between **input data** —that is, the "independent variables" in the problem—and **output data**—that is, the desired results. Input data and output data consist of a finite number of real (or complex) quantities and are thus representable by finite dimensional vectors. The functional connection can be expressed in either explicit or implicit from. We require for the following discussion also that *the output data should be uniquely determined and depend continuously on the input data.*

By an **algorithm**[5] for a given numerical problem we mean a *complete description of well-defined operations* through which each permissible input data vector is transformed into an output data vector. By "operations" we mean here arithmetic and logical operations, which a computer can perform, together with references to previously defined algorithms. (The concept "algorithm" can be analogously defined for problems completely different from numerical problems, with other types of input data and fundamental operations—for example, inflection, merging of words, and other transformations of words in a given language.)

By the term **numerical method** we mean in this book a procedure either to approximate a mathematical problem with a numerical problem or to solve a numerical problem (or at least it to a simpler problem). A numerical method should be more generally applicable than an algorithm, and set lesser emphasize on the completeness of the computational details. The transformation of a differential equation problem to a system of nonlinear equations, as in Example 1.4.1 can be called a numerical method—even without instructions as to how to solve the system of nonlinear equations. Newton's method is a numerical method for determining

[5]The term "algorithm" is a latinization of the name of the Arabic 9th century mathematician Al-Khowârizmî. He also introduced the word algebra (Al-jabr). Western Europe became acquainted with the Hindu positional number system from a latin translation of his book entitled "Algorithmi de numero Indorum".

a root of a large class of nonlinear equations. In order to call it an algorithm conditions for starting and stopping the iteration process should be added.

For a given numerical problem one can consider many differing algorithms. In floating point computations these can give approximations of widely varying accuracy to the exact solution.

Example 2.4.1.

To determine the largest real root of the cubic equation

$$p(z) = a_0 z^3 + a_1 z^2 + a_2 z + a_3 = 0,$$

with real coefficients a_0, a_1, a_2, a_3, is a numerical problem. The input data vector is (a_0, a_1, a_2, a_3). The output data is the desired root x; it is an implicitly defined function of the input data. An algorithm for this problem can be based on Newton's method, supplemented with rules for how the initial approximation should be chosen and how the iteration process is to be terminated. One could also use other iterative methods, or even algorithms based upon Cardano's exact solution of the cubic equation. Cardano's solution uses square roots and cube roots, so one needs to assume that algorithms for the computation of these functions have been specified previously.

One often begins the construction of an algorithm for a given problem by breaking down the problem into subproblems in such a way that the output data from one subproblem is the input data to the next subproblem. Thus the distinction between problem and algorithm is not always so clearcut. The essential point is that, in the formulation of the problem, one is only concerned with the initial state and the final state. In an algorithm, however, one should clearly define each step along the way, from start to finish. Before an algorithm can be used it has to be implemented in an algorithmic program language in a reliable and efficient manner. *This is far from a trivial task—it has been said that when the novice thinks the job is done then the expert knows that most of the hard work lies ahead!*

Example 2.4.2.

The problem of solving the differential equation

$$\frac{d^2y}{dx^2} = x^2 + y^2$$

with boundary conditions $y(0) = 0$, $y(5) = 1$, is not a numerical problem according to the definition stated above. This is because the output data is the *function* y, which cannot in any conspicuous way be specified by a finite number of parameters. The above mathematical problem can be *approximated with a numerical problem* if one specifies the output data to be the values of y for $x = h, 2h, 3h, \dots, 5 - h$. Also the domain of variation of the unknowns must be restricted in order to show that the problem has a unique solution. This can be done, however, and there are a number of different algorithms for solving the problem approximately, which have different properties with respect to number of arithmetic operations needed and the accuracy obtained.

2.4.2 Propagation of Errors

In scientific computing the given input data is usually imprecise. The errors in the input will propagate and give rise to errors in the output. In this section we develop some general tools for studying the propagation of errors. These error-propagation formulas are also of great interest in the **planning and analysis of scientific experiments.**

Note that rounding errors from each step in a calculation are also propagated to give errors in the final result. For many algorithms a rounding error analysis can be given, which shows that the computed result always equals the exact (or slightly perturbed) result of a nearby problem, where the input data has been slightly perturbed, (See, e.g, Lemma 2.3.4.) Then the effect of rounding errors on the final result can be estimated using the tools of this section.

We first consider two simple special cases of error propagation. For a sum of an arbitrary number of terms we get from (2.3.17) by induction:

Lemma 2.4.1.

In addition and subtraction, a bound for the absolute errors in the result is given by the sum of the bounds for the absolute errors of the operands

$$y = \sum_{i=1}^{n} \pm x_i, \qquad |\Delta y| \le \sum_{i=1}^{n} |\Delta x_i|. \tag{2.4.1}$$

To obtain a corresponding result for the error propagation in multiplication and division, we start with the observations that for $y = \ln(x)$ we have $\Delta(\ln(x)) \approx \Delta(x)/x$. In words: *the relative error in a quantity is approximately equal to the absolute error in its natural logarithm.* This is related to the fact that displacements of the same length at different places on a logarithmic scale, mean the same relative change of the value. From this we obtain the following result:

Lemma 2.4.2.

In multiplication and division, an approximate bound for the relative error is obtained by adding the relative errors of the operands. More generally, for $y = x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}$,

$$\left|\frac{\Delta y}{y}\right| \lesssim \sum_{i=1}^{n} |m_i| \left|\frac{\Delta x_i}{x_i}\right|. \tag{2.4.2}$$

Proof. The proof follows by differentiating $\ln y = m_1 \ln x_1 + m_2 \ln x_2 + \cdots + m_n \ln x_n$ and estimating the perturbation in each term. □

Example 2.4.3.

In Newton's method for solving a nonlinear equation a correction is to be calculated as a quotient $y = f(x_k)/f'(x_k)$, Assume that $f(x_k)$ is known only to a certain relative accuracy. How accurately should one compute $f'(x_k)$, assuming that the work grows as one demands higher accuracy? Since the limit for the relative error in y is equal to the sum of the bounds for the relative errors in $f(x_k)$ and $f'(x_k)$, there is no gain in making the relative error in $f'(x_k)$ very much less than the relative error in $f(x_k)$. This observation is of great importance in particular in the generalization of Newton's method to *systems* of nonlinear equations.

We now study the propagation of errors in more general non-linear expressions. Consider first the simple case when we want to compute a function $y = f(x)$ of a single real variable x. How is the error in x propagated to y? Let $\tilde{x} - x = \Delta x$. Then, a natural way is to approximate $\Delta y = \tilde{y} - y$ with the differential of y (see Fig. 2.4.1). By the mean value theorem,

$$\Delta y = f(x + \Delta x) - f(x) = f'(\xi)\Delta x,$$

where ξ is a number between x and $x + \Delta x$. Suppose that $|\Delta x| \le \epsilon$. Then it follows that

$$|\Delta y| \le \max_{\xi} |f'(\xi)|\epsilon, \quad \xi \in [x - \epsilon, x + \epsilon]. \tag{2.4.3}$$

In practice, it is usually sufficient to replace ξ by the available estimate of x. *Even if high precision is needed in the value of $f(x)$, one rarely needs a high relative precision in an error bound or an error estimate.* It is only in the neighborhood of zeros of the first derivative $f'(x)$ that one has to be more careful.

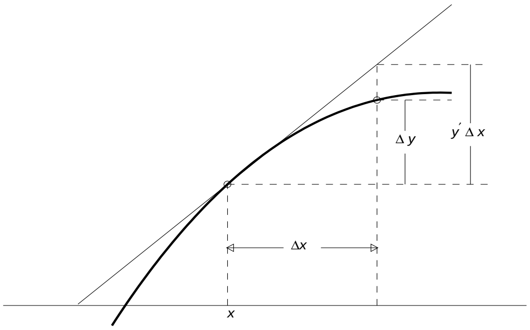


Figure 2.4.1. *Propagated error in function $y = f(x)$.*

By the implicit function theorem a similar result holds if y is an implicit function of x defined by $g(x, y) = 0$. If $\frac{\partial g}{\partial y} \neq 0$, then

$$|\Delta y| \le \max_{\xi} \left| \frac{\partial g}{\partial y}(\xi) \Big/ \frac{\partial g}{\partial y}(\xi) \right| \epsilon, \quad \xi \in [x - \epsilon, x + \epsilon]. \tag{2.4.4}$$

Example 2.4.4. The result in Lemma 2.3.4 does not say that the computed roots of the quadratic equation are close to the exact roots r_1, r_2. To answer that question we must determine how sensitive the roots are to a relative perturbations in the coefficient c. Differentiating $ax^2 + bx + c = 0$, where $x = x(c)$ with respect to c we obtain $(2ax + b)dx/dc + 1 = 0$, $dx/dc = -1/((2ax + b)$. With $x = r_1$ and using $r_1 + r_2 = -b/a$, $r_1r_2 = c/a$ this can be written

$$\frac{dr_1}{r_1} = -\frac{dc}{c}\frac{r_2}{r_1 - r_2}.$$

This shows that when $|r_1 - r_2| \ll |r_2|$ the roots can be very sensitive to small relative perturbations in c.

When $r_1 = r_2$, i.e. when there is a double root, this linear analysis breaks down. Indeed it is easy to see that the equation $(x - r)^2 + \Delta c = 0$ has roots $x = r \pm \sqrt{\Delta c}$.

To analyze error propagation in a function of several variables $f = f(x_1, x_2, \ldots, x_n)$ we need the following generalization of the mean value theorem:

Theorem 2.4.3.

Assume that the real valued function f is differentiable in a neighborhood of the point $x = (x_1, x_2, \ldots, x_n)$, and let $x = x + \Delta x$ be a point in this neighborhood. Then there exists a number θ, such that

$$\Delta f = f(x + \Delta x) - f(x) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x + \theta \Delta x)\Delta x_i, \quad 0 \le \theta \le 1.$$

Proof. The proof follows by considering the function $F(t) = f(x + t\Delta x)$ and using the mean value theorem for functions of one variable and the chain rule. □

From Theorem 2.4.3 it follows that the perturbation Δf is approximately equal to the total differential. The use of this approximation means that the function $f(x)$ is, in a neighborhood of x that contains the point $x + \Delta x$, approximated by a linear function. All the techniques of differential calculus, such as logarithmic differentiation, implicit differentiation etc. may be useful for the calculation of the total differential; see the examples and the problems at the end of this section.

Theorem 2.4.4. General Formula for Error Propagation*:*

Let the real valued function $f = f(x_1, x_2, \ldots, x_n)$ be differentiable in a neighborhood of the point $x = (x_1, x_2, \ldots, x_n)$ with error $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$. Then it holds

$$\Delta f \approx \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i. \tag{2.4.5}$$

where the partial derivatives are evaluated at x.

For the maximal error in $f(x_1, x_2, \ldots, x_n)$ we have the approximate bound

$$|\Delta f| \lessapprox \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}\right| |\Delta x_i|. \tag{2.4.6}$$

In order to get a *strict* bound for $|\Delta f|$, one should use in (2.4.6) the maximum absolute values of the partial derivatives in a neighborhood of the known point x. In most practical situations it suffices to calculate $|\partial f/\partial x_i|$ at x and then add a certain marginal amount (5 to 10 percent, say) for safety. Only if the Δx_i are large or if the derivatives have a large relative variation in the neighborhood of x, need the maximal values be used. (The latter situation occurs, for example, in a neighborhood of an extremal point of $f(x)$.)

The bound in Theorem 2.4.4 is the best possible, unless one knows some dependence between the errors of the terms. Sometimes it can, for various reasons, be a coarse overestimate of the real error, as we have seen in Example 2.3.8.

Example 2.4.5.

Compute error bounds for $f = x_1^2 - x_2$, where $x_1 = 1.03 \pm 0.01, x_2 = 0.45 \pm 0.01$. We obtain

$$\left|\frac{\partial f}{\partial x_1}\right| = |2x_1| \leq 2.1, \qquad \left|\frac{\partial f}{\partial x_2}\right| = |-1| = 1,$$

and find $|\Delta f| \leq 2.1 \cdot 0.01 + 1 \cdot 0.01 = 0.031$, or $f = 1.061 - 0.450 \pm 0.032 = 0.611 \pm 0.032$; the error bound has been raised 0.001 because of the rounding in the calculation of x_1^2.

One is seldom asked to give mathematically guaranteed error bounds. More often it is satisfactory to give an estimate of the *order of magnitude* of the anticipated error. The bound for $|\Delta f|$ obtained with Theorem 2.4.3 estimates the maximal error, i.e, covers the worst possible cases, where the sources of error Δx_i contribute with the same sign and magnitudes equal to the error bounds for the individual variables.

In practice, the trouble with formula (2.4.6) is that it often gives bounds which are too coarse. More realistic estimates are often obtained using the standard error. In Sec. 2.3.4 we introduced the standard error for a sum. Here we give without proof the result for the general case, which can be derived using probability theory and the formula (2.4.5).

Theorem 2.4.5.

Assume that the errors $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ are independent random variables with mean zero and standard deviations $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$. Then the standard error ϵ for $f(x_1, x_2, \ldots, x_n)$ is given by the formula:

$$\epsilon \approx \left(\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 \epsilon_i^2 \right)^{1/2} \tag{2.4.7}$$

Analysis of error propagation is more than just a means for judging the reliability of calculated results. As remarked above, it has an equally important function as a means for the *planning of a calculation or scientific experiment.* For example, it can help in the choice of algorithm, and in making certain decisions during a calculation. Examples of such decisions are the choice of step length during a numerical integration. Increased accuracy often has to be bought at the price of more costly or complicated calculations. One can also shed some light, to what degree it is advisable to obtain a new apparatus to improve the measurements of a given variable, when the measurements of other variables are subject to error as well.

2.4.3 Condition Numbers of Problems

It is useful to have a measure of how sensitive the output data is for variations in the input data. In general, if small changes in the input data can result in "large" changes in the output data, we call the problem **ill-conditioned**; otherwise it is called **well-conditioned**. (The definition of large may differ from problem to problem depending on the accuracy of the data and the accuracy needed in the solution.)

We have seen in Sec. 2.4.1 that $|f'(x)|$ can be interpreted *as a measure of the sensitivity of $f(x)$ to a perturbation Δx of x.* In many contexts, the ratio of the *relative* perturbations in $f(x)$ and x is of more interest.

Definition 2.4.6. *Assume that $x \neq 0$ and $f(x) \neq 0$, Then the* **condition number** *κ for the numerical problem of computing $y = f(x)$ is*

$$\kappa = \lim_{|\Delta x|\to 0} \frac{|f(x+\Delta x) - f(x)|}{|f(x)|} \Big/ \frac{|\Delta x|}{|x|} = |x|\frac{|f'(x)|}{|f(x)|}. \tag{2.4.8}$$

We say that the problem of computing $f(x)$ given x is ill-conditioned if κ is "large" and well-conditioned otherwise.

Note that the condition number is a property of the numerical problem and does not depend on the algorithm used! An ill-conditioned problem is *intrinsically* difficult to solve accurately using *any* numerical algorithm. Even if the input data is exact rounding errors made during the calculations in floating point arithmetic may cause large perturbations in the final result.

Example 2.4.6.

Consider the linear system

$$\begin{pmatrix} 1 & \alpha \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

where $\alpha \neq 1$ is the input data. The exact solution is

$$x = 1/(1-\alpha^2), \qquad y = -\alpha/(1-\alpha^2).$$

The matrix is singular for $\alpha = 1$, and the problem of computing x and y is ill-conditioned when $\alpha \approx 1$. Using equation (2.4.8) we find that the condition number for computing x is

$$\kappa = \alpha x'(\alpha)/x(\alpha) = 2\alpha^2/|1-\alpha^2|.$$

For $\alpha = 0.9950$ we get using Gaussian elimination and four decimal digits the computed values

$$\bar{y} = -0.995/(1-0.9902) = -99,80, \qquad \bar{x} = 1 + 0.9950 \cdot 99.80 = 100.30,$$

instead of the correct values $y = -99.7494$, $x = 100.2506$. Note that two digits were lost through cancellation in the divisor when computing y. The condition number $\kappa = 198$ indicates correctly that we can expect to loose two significant decimal digits.

Consider now a multivariate numerical problem P where the output data $y_j = f_j(x)$, $j = 1 : m$ depends non-linearly on the input data $x = (x_1, \ldots, x_n)$. Then by the general error propagation formula (see Theorem 2.4.4) we have the maximal error estimate

$$|\Delta y_j| \lessapprox \sum_{i=1}^{n} \left| \frac{\partial f_j}{\partial x_i} \right| |\Delta x_i|, \quad j = 1 : m. \tag{2.4.9}$$

This gives us a matrix of (relative) condition numbers

$$\kappa_{ij} = \left| \frac{\partial f_j}{\partial x_i} \right| \frac{|x_i|}{|y_j|}, \quad i = 1 : n, \quad j = 1 : m.$$

It is often more convenient to have a single number to measure the conditioning of the problem. This can be achieved, as in the linear case, by using norms.

Definition 2.4.7. *The condition number κ of a problem P with input data $(x_1, \ldots, x_n)$ and output data $(y_1, \ldots, y_m)$ is*

$$\kappa(P) = \lim_{\epsilon \to 0} \sup \frac{1}{\epsilon} \frac{\|\tilde{y} - y\|}{\|y\|}, \quad \|\tilde{x} - x\| \le \epsilon \|x\|. \tag{2.4.10}$$

Note that $\kappa(P)$ is a function of the input data x and depends on the choice of norms in the data space and solution space. If the condition number of a problem is κ, then for sufficiently small ϵ we have the estimate

$$\|\tilde{y} - y\| \le \kappa\epsilon\|y\| + O(\epsilon^2).$$

It follows that the solution will have roughly $s = \log_{10} \kappa$ less significant decimal digits than the input data.

If we get an inaccurate solution to a ill-conditioned problem P, then often nothing can be done about the situation, since the difficulty is intrinsic to the problem P. But *sometimes the difficulty can depend on the form one has chosen to represent the input and output data of the problem.*

The conditioning of a problem can to some degree be illustrated geometrically. A numerical problem P means a mapping of the space X of possible input data onto the space Y of the output data. The dimensions of these spaces are usually quite large. In Fig 2.3.2 we picture a mapping in two dimensions. Since we are considering relative changes, we take the coordinate axis to be logarithmically scaled. A small circle of radius r is mapped onto an ellipse whose major axis is κr, where κ is the condition number of the problem P.

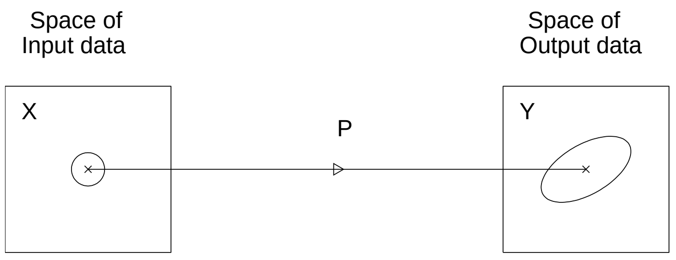


Figure 2.4.2. *Geometrical illustration of the condition number.*

Example 2.4.7.

The polynomial

$$P(x) = (x-10)^4 + 0.200(x-10)^3 + 0.0500(x-10)^2 - 0.00500(x-10) + 0.00100,$$

is identical with a polynomial Q which if the coefficients are rounded to six digits, becomes

$$\tilde{Q}(x) = x^4 - 39.8000x^3 + 594.050x^2 - 3941.00x + 9805.05.$$

One finds that $P(10.11) = 0.0015 \pm 10^{-4}$, where only three digits are needed in the computation, while $\tilde{Q}(10.11) = -0.0481 \pm \frac{1}{2}\cdot 10^{-4}$, in spite of the fact that eight digits were used in the computation. The rounding to six digits of the coefficients of Q has thus caused an error in the polynomial's value at $x = 10.11$; the erroneous value is more than 30 times larger than the correct value. When the coefficients of Q are input data, the problem of computing the value of the polynomial for $x \approx 10$ is far more ill-conditioned than when the coefficients of P are input data.

2.4.4 Perturbation Analysis for Linear Systems

Let x be the solution x to a system of linear equations $Ax = b$, where A is nonsingular and $b \neq 0$. We shall investigate the sensitivity of x to perturbations δA and δb in A and b. The perturbation δx satisfies

$$(A + \delta A)(x + \delta x) = b + \delta b.$$

Subtracting out $Ax = b$ we get $(A + \delta A)\delta x = (-\delta A x + \delta b)$. Assuming that also the matrix $A + \delta A$ is nonsingular, we can multiply by A^{-1} and solve for δx which yields the basic identity

$$\delta x = (I + A^{-1}\delta A)^{-1} A^{-1}(-\delta A x + \delta b). \tag{2.4.11}$$

Taking norms gives

$$\|\delta x\| \le \|(I + A^{-1}\delta A)^{-1}\| \, \|A^{-1}\| \left(\|\delta A\| \, \|x\| + \|\delta b\|\right). \tag{2.4.12}$$

In the simple case that $\delta A = 0$ we have $\delta x = A^{-1}\delta b$ and

$$\|\delta x\| \le \|A^{-1}\|\,\|\delta b\|.$$

Usually it is more appropriate to consider normwise *relative* perturbations,

$$\frac{\|\delta x\|}{\|x\|} \le \kappa(A,x)\frac{\|\delta b\|}{\|b\|}, \quad \kappa(A,x) := \frac{\|Ax\|}{\|x\|}\|A^{-1}\|$$

where $\kappa(A,x)$ is the condition number for computing x. This inequality is sharp in the sense that for any matrix norm and for any A and b there exists a perturbation δb such that equality holds.

When $\delta A \ne 0$ we obtain from (2.4.12), neglecting second order terms and using the inequality $\|b\| = \|Ax\| \le \|A\|\,\|x\|$,

$$\frac{\|\delta x\|}{\|x\|} \lessapprox \kappa(A)\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right), \quad \kappa(A) = \|A\|\,\|A^{-1}\|. \tag{2.4.13}$$

This shows that a normwise relative perturbation in A or b can at most be amplified by the factor $\kappa = \|A\|\,\|A^{-1}\|$. Note that in (2.4.13) equality will hold only for rather *special right hand sides b*. For given x (or b) the bound (2.4.13) may be unachievable for any perturbation δb!

Clearly the condition number $\kappa(A)$ depends on the choice of norms in the data space and solution space. The most common norms are special cases, $p = 1, 2$ and ∞, of the family of vector p-norms, (see Sec. 1.6.8)

$$\|x\|_p = (x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}, \qquad 1 \le p < \infty$$

and the corresponding subordinate matrix norm. In particular, using the Euclidian vector and matrix norm ($p = 2$) we define:

Definition 2.4.8.

The **condition number** *for a square nonsingular matrix A is*

$$\kappa_2 = \kappa_2(A) = \|A\|_2\,\|A^{-1}\|_2 = \sigma_1/\sigma_n, \tag{2.4.14}$$

where σ_1 and σ_n are the largest and smallest singular value of A.

The condition number κ_2 measures the sensitivity of the solution x to perturbations in A and b. Note that $\kappa(\alpha A) = \kappa(A)$, i.e., the condition number is invariant under multiplication of A by a scalar. From the definition and the identity $AA^{-1} = I$ it also follows that $\kappa(AB) \le \kappa(A)\kappa(B)$ and

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 \ge \|I\| = 1,$$

that is, the condition number κ_2 is always greater or equal to one.

Matrices with small condition numbers are said to be **well-conditioned**. For any real, orthogonal matrix Q we have

$$\kappa_2(Q) = \|Q\|_2\|Q^{-1}\|_2 = 1,$$

so Q is perfectly conditioned. Furthermore, $\kappa_2(A)$ is invariant under orthogonal transformations, i.e. for any orthogonal P and Q we have $\kappa_2(PAQ) = \kappa_2(A)$.

How large may κ be before we consider the problem to be ill-conditioned? That depends on the accuracy of the data and the accuracy desired in the solution. If the data have a relative error of 10^{-7} then we can guarantee a relative error in the solution to be $\leq 10^{-3}$ if $\kappa \leq 0.5 \cdot 10^4$. However, to guarantee a relative error in the solution $\leq 10^{-6}$ we need to have $\kappa \leq 0.5 \cdot 10^4$.

Table 2.4.1. *Condition numbers of Hilbert matrices of order* ≤ 12.

n	$\kappa_2(H_n)$	n	$\kappa_2(H_n)$
1	1	7	4.753+08
2	19.281	8	1.526+10
3	5.241+02	9	4.932+11
4	1.551+04	10	1.602+13
5	4.766+05	11	5.220+14
6	1.495+07	12	1.678+16

Example 2.4.8.

The Hilbert matrix H_n of order n is an $n \times n$ matrix with elements

$$H_n(i,j) = h_{ij} = 1/(i+j-1).$$

It is a notable example of an ill-conditioned matrix. In Table 2.4.4 approximate condition numbers of Hilbert matrices of order ≤ 12, computed in IEEE double precision, are given. The condition numbers $\kappa_2(H_n)$ are seen grow exponentially with n. For $n > 12$ the Hilbert matrices are too ill-conditioned even for IEEE double precision! Gautschi [11, p. 34] remarks that from a result by G. Szegö it follows that

$$\kappa_2(H_n) \approx \frac{(\sqrt{2}+1)^{4(n+1)}}{2^{15/4}\sqrt{\pi n}} \sim e^{3.5n},$$

which shows that the condition number grows exponentially.

Although the severe ill-conditioning exhibited by the Hilbert matrices is rare, moderately and severely ill-conditioned linear systems do occur regularly in many practical applications!

The normwise analysis in the previous section may not work well unless the linear system is "well scaled", i.e., the elements in the A, b, and x have roughly similar sizes. The a **component-wise perturbation** analysis may give sharper bounds. We first need to introduce some notations. The absolute values $|A|$ and $|b|$ of a matrix A and vector b is interpreted componentwise,

$$|A|_{ij} = (|a_{ij}|), \qquad |b|_i = (|b_i|).$$

The partial ordering "$\le$" for matrices A, B and vectors x, y, is also to be interpreted component-wise[6]

$$A \le B \iff a_{ij} \le b_{ij}, \qquad x \le y \iff x_i \le y_i.$$

It follows easily that if $C = AB$, then $|C| \le |A|\,|B|$, or

$$|c_{ij}| \le \sum_{k=1}^{n} |a_{ik}|\,|b_{kj}|,$$

A similar rule holds for matrix-vector multiplication.

Assume now that we have component-wise relative bounds for the perturbations in A and b,

$$|\delta A| \le \omega |A|, \quad |\delta b| \le \omega |b|. \tag{2.4.15}$$

By taking absolute values in (2.4.11) we obtain component-wise error bounds for the corresponding perturbations in x,

$$|\delta x| \le |(I + A^{-1}\delta A)^{-1}|\,|A^{-1}|(|\delta A||x| + |\delta b|)$$

Here the matrix $(I - |A^{-1}||\delta A|)$ is guaranteed to be nonsingular if $\|\,|A^{-1}|\,|\delta A|\,\| < 1$. Neglecting second order terms and using (2.4.15) we get the **component-wise relative error bounds**

$$|\delta x| \lessapprox |A^{-1}|(|\delta A||x| + |\delta b|) \le \omega |A^{-1}|(|A||x| + |b|), \tag{2.4.16}$$

Taking norms in (2.4.16) we obtain

$$\|\delta x\| \le \omega \|\,|A^{-1}|(|A|\,|x| + |b|)\,\| \tag{2.4.17}$$

and

$$\kappa_{|A|}(A) = \|\,|A^{-1}|\,|A|\,\| \tag{2.4.18}$$

is the **Bauer–Skeel condition number** of the matrix A.

2.4.5 Experimental Perturbations

In larger problems, the relations between input data and output data are so complicated that it is difficult to directly apply the general formulas for error propagation. One can then investigate the sensitivity of the output data for perturbations in the input data by means of an **experimental perturbational calculation**: one performs the calculations many times with perturbed input data and studies the perturbations in the output data.

Important data, such as the step length in a numerical integration or the parameter which determines when an iterative process is going to be broken off, should be varied with all the other data left unchanged. If one can easily *vary the precision of the machine* in the arithmetic operations one can get an idea of the

[6]Note that $A \le B$ in other contexts means that $B - A$ is positive semidefinite.

influence of rounding errors. It is generally not necessary to make a perturbational calculation for each and every data component; one can instead *perturb many input data simultaneously*–for example, by using random numbers.

A perturbational calculation often gives not only an error estimate, but also greater insight into the problem. Occasionally, it can be difficult it can be difficult to interpret the perturbational data correctly, since the disturbances in the output data depend not only on the mathematical problem but also on the choice of numerical method and the details in the design of the algorithm.The rounding errors during the computation are not the same for the perturbed and unperturbed problem. Thus if the output data reacts more sensitively than one had anticipated, it can be difficult to immediately point out the source of the error. It can then be profitable to plan a series of perturbation experiments with the help of which one can separate the effects of the various sources of error. If the dominant source of error is the method or the algorithm, then one should try another method or another algorithm.

It is beyond the scope of this book to give further comments on the planning of such experiments; imagination and the general insights regarding error analysis which this chapter is meant to give play a large role. Even in the special literature, the discussion of planning of such experiments is surprisingly meager.

2.4.6 Forward and Backward Error Analysis

An **algorithm** for a given numerical problem is a complete description of well-defined operations through which each permissible input data vector is transformed into an output data vector. By "operations" we mean here arithmetic and logical operations which a computer can perform. In this section we

Consider a finite algebraic algorithm which from the data $(a_1, \dots, a_r)$, through intermediate values computes by arithmetic operations a solution $(w_1, \dots, w_t)$. There are two basic forms of roundoff error analysis for such an algorithm, which both are useful:

(i) In **forward error analysis** one attempts to find bounds for the errors in the solution $|\bar{w}_i - w_i|$, $i = 1 : t$, where $\bar{w}_i$ denotes the computed value of w_i.

(ii) In **backward error analysis**, pioneered by J. H. Wilkinson in the late fifties, one attempts to determine a modified set of data $\tilde{a}_i$ such that the computed solution $\bar{w}_i$ is the *exact solution*, and give bounds for $|\tilde{a}_i - a_i|$. There may be an infinite number of such sets; sometimes there is just one and it can happen, even for very simple algorithms, that no such set exists.

Sometimes, when a pure backward error analysis is difficult to achieve, one can show that the computed solution is a slightly perturbed solution to a problem with slightly modified input data. This is called a **mixed error analysis**.

Notice that in backward error analysis no reference is made to the exact solution for the original data. In practice, when the data is known only to a certain accuracy, the "exact" solution may not be well-defined. Then any solution, whose backward error is smaller than the domain of uncertainty of the data, may be considered to be satisfactory.

To yield error bounds for $\bar{w}_i$, a backward error analysis needs to be complemented with a perturbation analysis. For this the general error propagation formula in Section 2.4.2 can be used. A great advantage of backward error analysis is that when it applies, it tends to give much sharper results than a forward error analysis. Perhaps more important, it usually also gives a better insight into the stability (or lack of it) of the algorithm. It should be stressed that *the primary purpose of a rounding error analysis is to give insight in the properties of the algorithm.*

By means of backward error analysis it has been shown, even for many quite complicated algorithms, that the output data which the algorithm produce under the influence of roundoff error is the *exact* output data of a problem of the same type in which the data has been changed relatively by a few u.

2.4.7 Stability of Algorithms

One reason for poor accuracy in output data is that the problem is ill-conditioned. But poor accuracy can also be caused by a poorly constructed algorithm. We say in general that an algorithm is **unstable** if it introduces large errors in computed solutions to problems which are well-conditioned.

Example 2.4.9. .

Compute $z = x + y$ where x and y satisfies the linear system in Example 2.4.6. This problem is well-conditioned when $\alpha \approx 1$, since

$$z = x + y = 1/(1-\alpha^2) - \alpha/(1-\alpha^2) = 1/(1+\alpha).$$

The condition number is about 0.5. On the other hand, the problem of computing $x = 1/(1-\alpha^2)$ (and $y = -\alpha/(1-\alpha^2)$) is ill-conditioned when $\alpha \approx 1$ (see Example 2.4.6).

An algorithm which first solves the linear system for x and y and then adds to get z gives bad accuracy when $\alpha \approx 1$. For example, with $\alpha = 0.9900$, using four digit decimal arithmetic one gets $z = x + y = 50.25 - 49.75 = 0.5000$, while the correct answer is 0.5025, a loss of two digits. The lack of stability is revealed here by the fatal cancellation that occurs.

There are several different definitions of stability of an algorithm.

Definition 2.4.9.

An algorithm is **backward stable** if the computed solution $\bar{w}$ for the data a is the exact solution of a problem with slightly perturbed data $\bar{a}$ such that for some norm $\|\cdot\|$ it holds

$$\|\bar{a} - a\|/\|a\| < c_1 u, \tag{2.4.19}$$

where c_1 a not too large constant and u is the unit roundoff.

We are usually satisfied if we can prove forward or backward stability for $\|\cdot\|_2$ or $\|\cdot\|_\infty$, although we may like the estimates to hold element-wise, e.g.

$$|\bar{a}_i - a_i|/|a_i| < c_2 u, \qquad i = 1:r. \tag{2.4.20}$$

For example, by equation (2.3.12) the usual algorithm for computing an inner product x^Ty is backward stable, for element-wise relative perturbations.

We would like stability to hold for some *class of input data.* For example, a numerical algorithm for solving systems of linear equations $Ax = b$ is backward stable for a class of matrices $\mathcal{A}$ if for each $A \in \mathcal{A}$ and for each b the computed solution $\bar{x}$ satisfies $\bar{A}\bar{x} = \bar{b}$ where $\bar{A}$ and $\bar{b}$ are close to A and b.

A backward stable algorithm will not necessarily compute an accurate solution. However, if the condition number of the problem is κ, then it follows that

$$||\bar{w} - w|| \le c_1 u\kappa ||w|| + O(u^2). \tag{2.4.21}$$

Hence the error in the solution may be large if the problem is ill-conditioned. However, we have obtained an answer which is the exact mathematical solution to a problem with data close to the one we wanted to solve. If the perturbations $\bar{a} - a$ are within the uncertainties of the given data, *the computed solution is as good as our data warrants*!

An important property of backward stable algorithms for the solution of linear systems is given in the following theorem.

Theorem 2.4.10.

An algorithm for solving $Ax = b$ is backward stable according to Definition 2.4.9 if and only if the computed solution $\bar{x}$ has a small residual, that is,

$$||b - A\bar{x}|| \le c_3 u ||A|| ||\bar{x}||. \tag{2.4.22}$$

Proof. Suppose that (2.4.22) holds. If we define for the 2-norm

$$\delta A = r\bar{x}^T / ||\bar{x}||_2^2, \qquad r = b - A\bar{x},$$

then it holds exactly that $(A + \delta A)\bar{x} = A\bar{x} + r = b$, where

$$||\delta A||_2 \le ||r||_2 / ||\bar{x}||_2 \le c_3 u ||A||_2.$$

We can take $\delta b = 0$ and hence the algorithm is backward stable by Definition 2.5.3.

Conversely, if the algorithm is backward stable then, $\bar{A}\bar{x} = \bar{b}$, where

$$||\bar{A} - A|| \le c_2 u ||A||, \qquad ||\bar{b} - b|| \le c_2 u ||b||.$$

Since $b - A\bar{x} = (\bar{A} - A)\bar{x} + b - \bar{b}$ it follows that an estimate of the form (2.4.22) holds for the norm of the residual. $\square$

Many important algorithms for solving linear systems, for example, most iterative methods, are not backward stable. The following weaker definition of stability is also useful.

Definition 2.4.11. *An algorithm is* **stable** *if the computed solution $\bar{w}$ satisfies* (2.4.21)*, where c_1 a not too large constant, u is the unit roundoff, and κ is the condition number of the problem.*

By the definition of the condition number κ it follows that backward stability implies forward stability, but *the converse is not true.*

Sometimes it is necessary to weaken the definition of stability. Often an algorithm can be considered stable if *it produces accurate solutions for well-conditioned problems.* Such an algorithm can be called **weakly stable**. Weak stability may be sufficient for giving confidence in an algorithm.

Example 2.4.10. (Higham [24, Chapter 3.1])

The outer product of two vectors $x, y \in \mathbf{R}^n$ is $A = xy^T = (a_{ij})$, where $a_{ij} = x_i y_j$. In floating point arithmetic we compute $\bar{A} = fl(xy^T) = (\bar{a}_{ij})$, where $\bar{a}_{ij} = x_i y_j(1+\delta_{ij})$, $\delta_{ij} \le u$, and so

$$\bar{A} = xy^T + \Delta, \quad |\Delta| \le u|xy^T|.$$

This is a satisfactory result for many purposes, but the computation is not backward stable. The computed $\bar{A}$ is not in general a rank one matrix and thus it is not possible to find perturbations Δx and Δy so that $\bar{A} = (x+\Delta x)(x+\Delta y)^T$.

In the method of normal equations for computing the solution of a linear least squares problem one first forms the matrix A^TA. This product matrix can be expressed in outer form as

$$A^TA = \sum_{i=1}^{m} a_i a_i^T,$$

where a_i^T is the ith row of A, i.e. $A^T = \begin{pmatrix} a_1 & a_2 & \dots & a_m \end{pmatrix}$. By the result above it follows that this computation is not backward stable, i.e. it is not true that $\mathrm{fl}\,(A^TA) = (A+E)^T(A+E)$ for some small error matrix E. In order to avoid loss of significant information double precision need to be used.

Backward stability is easier to prove when there is a sufficiently large set of input data compared to the number of output data. This makes it harder to show backward stability when the input data is structured rather than general.

goodbreak

Example 2.4.11. A **Toeplitz matrix** T is a matrix whose entries are constant along every diagonal $T = (t_{i-j})_{1\le i,j\le n}$,

$$T = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \dots & t_0 \end{pmatrix} \in \mathbf{R}^{n\times n},$$

and is defined by the $2n-1$ values of $t_{-n+1}, \dots, t_0, \dots, t_{n-1}$. Toeplitz matrices arising in applications are often large, and dimensions of $10,000$ not uncommon.

Consequently there is a need for special fast methods for solving Toeplitz systems. In large problems also storage requirements are important. The original matrix T only requires $2n - 1$ storage. However, if standard factorization methods are used, at least $n(n + 1)/2$ storage is needed.

In the construction of an algorithm for a given problem, one often breaks down the problem into a chain of subproblems, $P_1, P_2, \dots, P_k$ for which algorithms $A_1, A_2, \dots, A_k$ are known, in such a way that the output data from P_{i-1} is the input data to P_i. Different ways of decomposing the problem give numerically different algorithms. It is dangerous if the last subproblem in such a chain is ill-conditioned. On the other hand, it need not be dangerous if the first subproblem of such a decomposition is ill-conditioned, even if the problem itself is well-conditioned.

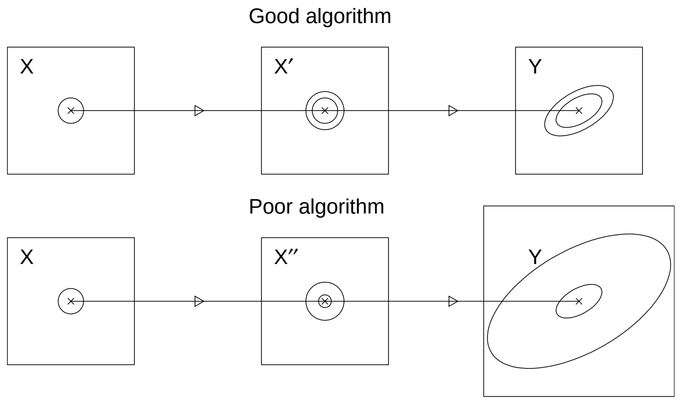


Figure 2.4.3. *Two examples of a decomposition of a problem P into two subproblems.*

In Fig. 2.5.2 we see two examples of a decomposition of the problem P into two subproblems. From X to X'' there is a strong contraction which is followed by an expansion about equally strong in the mapping from X'' to Y. The roundoff errors which are made in X'' when the intermediate results are stored have as a consequence that one arrives somewhere in the surrounding circle, which is then transformed into a very large region in Y. *The important conclusion is that even if the algorithms for the subproblems are stable we cannot conclude that the composed algorithm is stable!*

Example 2.4.12.

The problem of computing the eigenvalues λ_i of a symmetric matrix A from its elements (a_{ij}) is always well-conditioned functions, cond $= 1$. Consider an algorithm which breaks down this problem into two subproblems:

P_1: to compute the coefficients of the characteristic polynomial $P(\lambda)$ of A.

P_2: to compute the roots of the characteristic equation $P(\lambda) = 0$.

It is well known that the second subproblem P_2 can be very ill-conditioned. For example, for a symmetric matrix A with eigenvalues $\pm 1, \pm 2, \ldots, \pm 20$ the condition number for P_2 is 10^{14} in spite of the fact that the origin lies exactly between the largest and smallest eigenvalues, so that one cannot blame the high condition number on a difficulty of the same type as that encountered in Example 2.4.7.

Review Questions

1. The maximal error bounds for addition and subtraction can for various reasons be a course overestimate of the real error. Give, preferably with examples, two such reasons.

2. How is the condition number $\kappa(A)$ of a matrix A defined? How does $\kappa(A)$ relate to perturbations in the solution x to a linear system $Ax = b$, when A and b are perturbed?

3. Define the condition number of a numerical problem P of computing output data $y_1, \ldots, y_m$ given input data $x_1, \ldots, x_n$.

4. Give examples of well-conditioned and ill-conditioned problems.

5. What is meant by (a) a forward error analysis; (b) a backward error analysis; (c) a mixed error analysis?

6. What is meant by (a) a backward stable algorithm; (b) a forward stable algorithm; (c) a mixed stable algorithm; (d) a weakly stable algorithm?

Problems and Computer Exercises

1. (a) Determine the maximum error for $y = x_1 x_2^2/\sqrt{x_3}$, where $x_1 = 2.0 \pm 0.1$, $x_2 = 3.0 \pm 0.2$, and $x_3 = 1.0 \pm 0.1$. Which variable contributes most to the error?

(b) Compute the standard error using the same data as in (a), assuming that the error estimates for the x_i indicate standard deviations.

2. One wishes to compute $f = (\sqrt{2} - 1)^6$, using the approximate value 1.4 for $\sqrt{2}$. Which of the following mathematically equivalent expressions gives the best result

$$\frac{1}{(\sqrt{2}+1)^6}; \quad (3 - 2\sqrt{2})^3; \quad \frac{1}{(3+2\sqrt{2})^3}; \quad 99 - 70\sqrt{2}; \quad \frac{1}{99+70\sqrt{2}}?$$

3. Analyze the error propagation in x^α:

(a) If x is exact and α in error. (b) If α is exact and x in error.

4. One is observing a satellite in order to determine its speed. At the first observation, $R = 30,000 \pm 10$ miles. Five seconds later, the distance has increased by $r = 125.0 \pm 0.5$ miles and the change in the angle was $\phi = 0.00750 \pm 0.00002$ radians. What is the speed of the satellite, assuming that it moves in a straight line and with constant speed in the interval?

5. One has an algorithm for computing the integral

$$I(a,b) = \int_0^1 \frac{e^{-bx}}{a+x^2} dx.$$

The physical quantities a and b have been measured to be $a = 0.4000 \pm 0.003$, $b = 0.340 \pm 0.005$. Using the algorithms for various values of a and b one performs experimental perturbations and obtains:

a	b	I
0.39	0.34	1.425032
0.40	0.32	1.408845
0.40	0.34	1.398464
0.40	0.36	1.388198
0.41	0.34	1.372950

How large is the uncertainty in $I(a,b)$?

6. (a) Use the results in Table 2.4.4 to Determine constants c and q such that $\kappa(H_n) \approx c \cdot 10^q$.

(b) Compute the Bauer–Skeel condition number cond $(H_n) = \| \, |H_n^{-1}||H_n| \, \|_2$, of the Hilbert matrices for $n = 1 : 12$. Compare the result with the values of $\kappa(H_n)$ given in Example 2.4..

6. (a) Let two vectors u and v be given with components (u_1, u_2) and (v_1, v_2). Then the angle ϕ between u and v is given by the formula

$$\cos\phi = \frac{u_1v_1 + u_2v_2}{(u_1^2+u_2^2)^{1/2}(v_1^2+v_2^2)^{1/2}}.$$

Show that computing ϕ from the components of u and v is always a well-conditioned problem.

Hint: Take the partial derivative of $\cos\phi$with respect to u_1, and from this compute $\partial\phi/\partial u_1$. The other partial derivatives are obtained by symmetry.

(b) Show that the formula in (a) is *not* stable for small angles ϕ.

(c) Show that the following algorithm is stable. First normalize the vectors $\tilde{u} = u/\|u\|_2$, $\tilde{v} = v/\|v\|_2$, and then compute $\alpha = \|\tilde{u} - \tilde{v}\|_2$, $\beta = \|\tilde{u} + \tilde{v}\|_2$. Now take

$$\phi = \begin{cases} 2\arctan(\alpha/\beta), & \text{if } \alpha \le \beta; \\ \pi - 2\arctan(\beta/\alpha), & \text{if } \alpha > \beta. \end{cases}$$

7. Derive a forward and a backward recursion formula for calculating the integrals

$$I_n = \int_0^1 \frac{x^n}{4x+1}\, dx.$$

Why is one algorithm stable and the other unstable?

Notes and References

In the early days of computing floating point computations were not built into the hardware but implemented in software. The earliest subroutines for floating point arithmetic were probably those developed by J. H. Wilkinson at the National Physical Laboratory, England, in 1947. An excellent source of information on floating point computation, rounding error analysis, and related topics is Higham [24, Chapter 2]

A treatment of many different aspects of number systems and floating point computations is given in Knuth [27, Chapter 4]. He gives an interesting overview of the historical development of number representation Leibniz 1703 seems to have been the first to discuss binary arithmetic. He did not advocate it for practical calculations, but stressed its importance for number-theoretic investigations. King Charles XII of Sweden came upon the idea of radix 8 arithmetic in 1717. He felt this to be more convenient than the decimal notation and considered introducing octal arithmetic into Sweden. He died in battle before decreeing such a change!

Another general source on floating point computation is Sterbenz [36]. The IEEE standard for binary floating point arithmetic is defined in [3, 1985] An excellent tutorial on IEEE floating-point standard is Goldberg [16, 1991]; see also Overton [33, 2001].

The leading significant digit of numbers represented in a number system with base β has been observed to closely fit a logarithmic distribution, i.e., the proportion of numbers with leading digit equal to n is $\log_\beta(1 + 1/n)$ $(n = 0, 1, \dots, \beta - 1)$. A discussion of this intriguing fact with historic references is found in Higham [24, Section 2.5].

The modern development of interval arithmetic was initiated by the work of R. E. Moore [29, 1966]. It has since been developed into a useful tool for many problems in scientific computing and engineering. Only recently has it been possible to exploit high-performance computers. By making use of the Basic Linear Algebra Subroutines (BLAS) and IEEE 754 standard fast portable codes can now be written. The MATLAB toolbox INTLAB developed by Rump, which is very efficient and easy to use, is described in [35, 34]. An excellent introduction to interval arithmetic, which includes a short tutorial on INTLAB is Hargreaves [20]

Backward error analysis was developed and popularized by J. H. Wilkinson in the 1950s and 1960s and the classic treatise on rounding error analysis is [37]. The more recent survey [38] gives a good summary and a historical background.

A collection of software tools called PRECISE has been developed by Chaitin-Chatelin et al., see [8]. These are designed to help the user set up computer experiments to explore the impact of the quality of convergence of numerical methods. It involves a statistical analysis of the effect on a computed solution of random perturbations in data

Starting in the 1960s much general purpose software, often collected in large libraries or packages have been developed. Two large suppliers of commercial scientific subroutine libraries are NAG and IMSL. MATLAB is a much used interactive system for matrix computations, with “toolboxes” available for many application areas, e.g., control problems. It has been used for this book in testing algorithms and we also have borrowed some of its notation. Many programs and packages are available in the public domain and can be downloaded free. A prime example is LAPACK, which superseded LINPACK and EISPACK in the mid 1990s, and contains programs for solving linear systems and eigenvalue problems. Other packages like DASSL are available for solving ordinary systems of differential equations. For a survey we refer to Vol. III, Chapter 15.

For software the National Institute of Standards and Technology (NIST) Guide to Available Mathematical Software (GAMS) is available at the Internet URL "gams.nist.gov". GAMS is an on-line cross-index of mathematical and statistical software providing abstracts, documentation, and source code of software modules and provides access to multiple repositories operated by others. Currently four repositories are indexed, three within NIST, and netlib. Both public-domain and proprietary software are indexed although source code of proprietary software is not redistributed by GAMS. Netlib is a repository of public domain mathematical software, data, address lists, and other useful items for the scientific computing community. Access to netlib is via the Internet URL "www.netlib.bell-labs.com"

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computation.* Academic Press, New York, NY, 1983.

[2] G. Alefeld and G. Mayer. *Interval analysis: theory and applications.* J. Comput. Appl. Math., 121 (2000), 421–464.

[3] Anon. IEEE Standard 754-1985 for Binary Floating-Point Arithmetic. *SIGPLAN*, 22:2:9–25, 1985.

[4] Anon. *IEEE Standard for Binary Floating Point Arithmetic, ANSI/IEEE Standard 854-1985.* IEEE, New York, 1987.

[5] D. H. Bailey. Algorithm 719: Multiprecision translation and execution of FORTRAN programs. *ACM Trans. Math. Software*, 19:3:288–319, 1993.

[6] R. P. Brent. Algorithm 524: A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Software*, 4:1:71–81, 1978.

[7] R. P. Brent. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Software*, 4:1:57–70, 1978.

[8] F. Chaitin-Chatelin and V. Frayssé. *Lectures on Finite Precision Computations.* SIAM, Philadelphia, PA, 1996.

[9] W. J. Cody and W. Waite. *Software Manual for the Elementary Functions.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[10] W. J. Cody Implementation and testing of function software. In *Problems and Methodologies in Mathematical Software Production*, P. C. Messina and A. Murli, eds., Springer-Verlag, Berlin, 1982, pp. 24–47.

[11] W. J. Cody Algorithm 714: CELEFUNT: A portable test package for complex elementary functions. *ACM Trans. Math. Software*, 14:4 p. 121, 1993.

[12] J. Demmel. Underflow and the reliability of numerical software. *SIAM J. Sci. Stat. Comput.*, 5:4:887–919, 1984.

[13] C. T. Fike. *Computer Evaluation of Mathematical Functions.* Prentice-Hall, Englewood Cliffs, NJ, 1968.

[14] G. E. Forsythe. Pitfalls in computation, or why a math book isn't enough. Amer. Math. Monthly, 77 (1970), pp. 931–956.

[15] W. Gautschi. *Numerical Analysis.* Birkhäuser, Boston, MA, 1997.

[16] D. Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, 23:5–48, 1991.

[17] H. H. Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century.* Springer-Verlag, New York, 1977.

[18] H. H. Goldstine and J. von Neumann. Numerical inverting of matrices of high order ii. *Proc. Amer. Math. Soc.*, 2:188–202, 1951.

[19] E. Hansen. *Topics in Interval Analysis.* Oxford University Press, Oxford, 1969.

[20] G. I. Hargreaves. Interval analysis in MATLAB. Numer. Anal. Report 418, Department of Mathematics, University of Manchester, 2002.

[21] J. F. Hart, E. W. Cheney, C. L. Lawson, H. J. Maehly, C. K. Mesztenyi, J. F. Rice, Jr. H. G. Thacher, and C. Witzgall. *Computer Approximations.* Wiley, New York, New York, 1968.

[22] C. Hastings. *Approximations for Digital Computers.* Princeton University Press, Princeton, NJ, 1955.

[23] I. Gargantini and P. Henrici. Circular arithmetic and the determination of polynomial zeros. Numer. Math., 18:4 (1972), pp. 305–320.

[24] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, second edition, 2002.

[25] W. Kahan. A survey of error analysis. In *Proc. IFIP Congress Ljubljana, Information Processing 1971,* North-Holland, Amsterdam, 1972, pp. 1214–1239.

[26] R. Kearfoot. *Interval computations: Introduction, uses, and resources*, Euromath. Bulletin, 2:1 (1996), pp. 95–112.

[27] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* Addison-Wesley, Reading, MA, second edition, 1981.

[28] X. S. Li, J. W Demmel et al. Design, implementation and testing of Extended and Mixed Precision BLAS. Tech. Report CS-00-451, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, October 2000. LAPACK working note 149.

[29] R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[30] R. E. Moore. *Methods and Applications in Interval Analysis.* SIAM, Philadelphia, PA, 1979.

[31] R. E. Moore. *Reliability in Computing. The Role of Interval Methods in Scientific Computing.* John Wiley, New York, 1988.

[32] J.-M. Muller, *Elementary Functions: Algorithm and Implementation.* Birkhäuser, Boston, MA, 1997.

[33] M. Overton. *Numerical Computing with IEEE Floating Point Arithmetic: Including One Theorem, One Rule of Thumb, and One Hundred and One Exercises.* SIAM, Philadelphia, PA, 2001.

[34] S. M. Rump. *Fast and parallel interval arithmetic.* BIT, 39:3 (1999), pp. 534–554.

[35] S. M. Rump. *INTLAB—INTerval LABoratory.* In Developments in Reliable Computing, T. Csendes, ed. Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104.

[36] P. H. Sterbenz *Floating Point Computation.* Prentice-Hall, Englewood Cliffs, N.J., 1974.

[37] J. H. Wilkinson. *Rounding Error in Algebraic Processes.* Notes on Applied Science No. 32, Her Majesty's Stationery Office, London, UK, 1963. Reprinted by Dover, New York, 1994.

[38] J. H. Wilkinson. Error analysis revisited. *IMA Bull.*, 22:192–200, 1986.

$$\begin{aligned} &= 2c_2 + 6c_3x + 12c_4x^2 + \cdots + (m+2)(m+1)c_{m+2}x^m + \cdots \\ -xy(x) &= -c_0x - c_1x^2 - c_2x^3 - \cdots - c_{m-1}x^m - \cdots . \end{aligned}$$

Equating coefficients of x^m in these series gives

$$c_2 = 0, \qquad (m+2)(m+1)c_{m+2} = -c_{m-1}, \quad m \geq 1.$$

It follows from the initial conditions that $c_0 = 1$, $c_1 = 0$. Thus $c_n = 0$, if n is not a multiple of 3, and using the recursion we obtain

$$y(x) = 1 - \frac{x^3}{6} + \frac{x^6}{180} - \frac{x^9}{12,960} + \cdots .$$

This gives $y(0.5) = 0.97925$. The x^9 term is ignored, since it is less than $2 \cdot 10^{-7}$. In this example also the first neglected term gives a strict bound for the error (i.e. for the remaining terms), since the absolute value of the term decreases, and the terms alternate in sign.

Since the calculation was based on a trial substitution, one should , strictly speaking, prove that the series obtained defines a function which satisfies the given problem. Clearly, the series converges at least for $|x| < 1$, since the coefficients are bounded. (In fact the series converges for all x.) Since a power series can be differentiated term by term in the interior of its interval of convergence, the proof presents no difficulty. Notice, in addition, that the finite series obtained for $y(x)$ by breaking off after the x^9-term is the exact solution to the following differential equation:

$$y'' = -xy - \frac{x^{10}}{12,960}, \quad y(0) = 1, \quad y'(0) = 0,$$

where the "perturbation term" $-x^{10}/12,960$ has magnitude less than 10^{-7} for $|x| \leq 0.5$. A similar backward analysis can, even in analogous more complicated cases, be extended to give a strict error estimate by the logarithmic norm technique, see Sec. 13.1.5.

3.1.2 Estimating the Remainder

In practice, one is seldom seriously concerned about a strict error bound when the computed terms decrease rapidly and it is "obvious" that the terms will continue to decrease equally quickly. One can then break off the series and use either the last included term or a coarse estimate of the **first neglected term** as an estimate of the remainder.

This rule is not very precise. *How rapidly is "rapidly"?* Questions like this occur everywhere in scientific computing. If mathematical rigor costs little effort or little extra computing time, then it should, of course, be used. Sometimes, however, obtaining an error bound that is both rigorous and realistic may cost more than what is felt reasonable for a one-shot problem in the laboratory (say).

In problems, where guaranteed error bounds are not asked for, i.e., when it is enough to obtain a feeling for the reliability of the results, one can handle these matters in the same spirit as one handles risks in every day life. It is then a matter of experience to formulate a simple and *sufficiently* reliable *termination criterion* based on the automatic inspection of the successive terms. [1]

The unexperienced scientific programmer may, however, find such questions hard, also in simple cases. And in the production of general purpose mathematical software, or in a context where an inaccurate numerical result can cause a disaster, such questions are serious and sometimes hard also for the experienced scientific programmer.

For this reason, we shall formulate a few theorems, with which one can often transform the feeling that "the remainder is negligible" to a mathematical proof. There are, in addition, actually numerically useful *divergent* series; see Sec. 3.1.8. When one uses such series, estimates of the remainder are clearly essential.

Assume that we want to compute a quantity S, which can be expressed in a series expansion, $S = \sum_{j=0}^{\infty} a_j$, and set

$$S_n = \sum_{j=0}^{n} a_j, \quad R_n = S - S_n.$$

We call $\sum_{j=n+1}^{\infty} a_j$ the **tail** of the series; a_n is the "last included term" and a_{n+1} is the "first neglected term". The *remainder* R_n with reversed sign is called the *truncation error.* [2]

The tail of a convergent series can often be compared to a series with a known sum, for example, a geometric series, or with an integral which can be computed directly.

Theorem 3.1.1. *Comparison with a Geometric Series.*

If $|a_{j+1}| \le k|a_j|$, $\forall j \ge n$, *where* $k < 1$, *then*

$$|R_n| \le \frac{|a_{n+1}|}{1-k} \le \frac{k|a_n|}{1-k}.$$

In particular if $k < 1/2$, then it is strictly valid that the absolute value of the remainder is less than the last included term.

Proof. By induction, one finds that $|a_j| \le k^{j-1-n}|a_{n+1}|$, $j \ge n+1$, since $|a_j| \le k^{j-1-n}|a_{n+1}| \Rightarrow |a_{j+1}| \le k|a_j| \le k^{j-n}|a_{n+1}|$. Thus

$$|R_n| \le \sum_{j=n+1}^{\infty} |a_j| \le \sum_{j=n+1}^{\infty} k^{j-1-n}|a_{n+1}| = \frac{|a_{n+1}|}{1-k} \le \frac{k|a_n|}{1-k},$$

according to the formula for the sum of an infinite geometric series. The last statement follows from the inequality $k/(1-k) < 1$, when $k < 1/2$. $\square$

[1] We postpone the general discussion of termination criteria until Sec. 5.5, where it will be discussed for iterative methods.

[2] In this terminology the remainder is the *correction* one has to make in order to eliminate the error.

Example 3.1.3. *Power series with slowly varying coefficients.*

Let $a_j = j^{1/2}\pi^{-2j}$. Then $a_6 = 2.4{\cdot}0.0000011 < 3{\cdot}10^{-6}$. Further,

$$\frac{|a_{j+1}|}{|a_j|} \le \frac{(j+1)^{1/2}}{j^{1/2}}\frac{\pi^{2j-2}}{\pi^{-2j}} \le (1+1/6)^{1/2}\pi^{-2} < 0.11,$$

for $j \ge 6$. Thus by Theorem 3.1.1 $|R_6| < 3{\cdot}10^{-6}\frac{0.11}{1-0.11} < 4{\cdot}10^{-7}$.

Figure 3.1.1. *Comparison with an integral.*

Theorem 3.1.2. *Comparison with an Integral.*

If $|a_j| \le f(j)$ for all $j \ge n$, where $f(x)$ is a nonincreasing function for $x \ge n$, then

$$|R_n| \le \int_n^\infty f(x)dx,$$

which yields an upper bound for R_n, if the integral is finite.

If $a_j = f_j > 0$ for all $j \ge n+1$, we also obtain a lower bound for the error, namely $\int_{n+1}^\infty f(x)dx$.

Proof. See Fig. 3.1.1. □

Example 3.1.4. When a_j is slowly decreasing, the two error bounds are typically rather close to each other, and are hence rather realistic bounds, much larger than the first neglected term a_{n+1}. Let $a_j = 1/(j^3+1)$, $f(x) = x^{-3}$. It follows that

$$R_n \le \int_n^\infty x^{-3}dx = n^{-2}/2.$$

In addition this bound gives an asymptotically correct estimate of the remainder, as $n \to \infty$. which shows that R_n is here significantly larger than the first neglected term.

For alternating series, however, the situation is typically quite different.

Definition 3.1.3. *A series is* **alternating** *for $j \ge n$ if, for all $j \ge n$, a_j and a_{j+1} have opposite signs, or equivalently* sign $a_j = -$sign a_{j+1}, *where* sign x *(read* *signum of x), is defined by*

$$\text{sign } x = \begin{cases} +1, & \text{if } x > 0; \\ 0, & \text{if } x = 0; \\ -1, & \text{if } x < 0. \end{cases}$$

Theorem 3.1.4. *If for a certain n it holds that R_n and R_{n+1} have opposite signs, then S lies between S_n and S_{n+1}. Furthermore*

$$S = \frac{1}{2}(S_n + S_{n+1}) \pm \frac{1}{2}|a_{n+1}|.$$

We also have the weaker results:

$$|R_n| \le |a_{n+1}|, \qquad |R_{n+1}| \le |a_{n+1}|, \qquad \text{sign } R_n = \text{sign } a_{n+1}.$$

Proof. The fact that R_{n+1} and R_n have opposite signs means, quite simply, that one of S_{n+1} and S_n is too large and the other is too small, i.e., that S lies between S_{n+1} and S_n. Since $a_{n+1} = S_{n+1} - S_n$, one has for positive values of a_{n+1}, the situation shown in Fig. 3.1.2, etc. ... From this figure, and an analogous one for the case of $a_{n+1} < 0$, the remaining assertions of the theorem clearly follow. $\square$

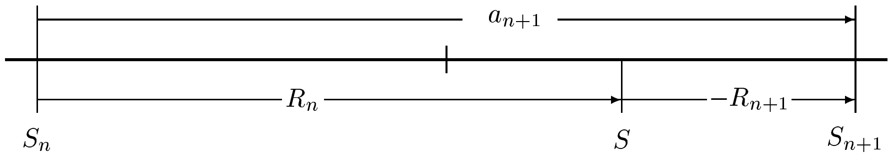


Figure 3.1.2. *Illustration to Theorem 3.1.4*

The actual error of the average $\frac{1}{2}(S_n + S_{n+1})$ is, for slowly convergent alternating series, usually much smaller than the error bound $\frac{1}{2}|a_{n+1}|$. For example, if $S_n = 1 - \frac{1}{2} + \frac{1}{3} - \ldots \pm \frac{1}{n}$, $\lim S_n = \ln 2 \approx 0.6931$, the error bound for $n = 4$ is 0.1, while the actual error is less than 0.01. See Sec. 3.3.3 for a systematic exploration of this observation, by means of repeated averaging.

We shall see that, in many power series, the remainder has the same sign as the first neglected term. If the series is alternating, then the above theorem can be used. An important consequence is:

Theorem 3.1.5.

For an alternating series, the absolute values of whose terms approach zero monotonically, the remainder has the same sign as the first neglected term a_{n+1}, and the absolute value of the remainder does not exceed $|a_{n+1}|$.

Figure 3.1.3. *The sum of an alternating series.*

Proof. (Sketch) That the theorem is true is almost clear from Fig. 3.1.2, and an analogous figure for the case $a_{n+1} < 0$. The figure shows how S_j depends on j when the premises of the theorem are fulfilled. A formal proof is left to the reader. □

For the use of the Theorem 3.1.5 see Examples 3.1.1 and 3.1.2. An important generalization is given as Problem 3.2.1(e).

In the preceding theorems the ideas of well known convergence criteria are extended to bound or estimates of the error of a truncated expansion. In Sec. 3.3, we shall see a further extension of these ideas, namely for improving the accuracy of a truncated expansion. This is called em convergence acceleration.

3.1.3 Power Series.

Consider an expansion into powers of a complex variable z, and suppose that it is convergent for some $z \neq 0$, and denote its sum by $f(z)$,

$$f(z) = \sum_{j=0}^{\infty} a_j z^j, \qquad z \in \mathbf{C}. \tag{3.1.1}$$

It is then known from complex analysis that the series (3.1.1) either converges for all z, or it has a **circle of convergence** with radius ρ, such that it either converges for all $|z| < \rho$, and diverges for $|z| > \rho$. (For $|z| = \rho$ either convergence or divergence is possible). The radius of convergence is determined by the relation $\rho = \limsup |a_n|^{-1/n}$. Another formula is $\rho = \lim |a_n|/|a_{n+1}|$, *if this limit exists.*

The function $f(z)$ can be expanded into powers of $z - a$ around any point of analyticity,

$$f(z) = \sum_{j=0}^{\infty} a_j (z-a)^j, \qquad z \in \mathbf{C}. \tag{3.1.2}$$

By **Taylor's formula** the coefficients are given by [3]

$$a_0 = f(a), \quad a_j = f^{(j)}(a)/j!, \quad j \geq 1. \tag{3.1.3}$$

The function $f(z)$ is analytic inside its circle of convergence, and has at least one singular point on its boundary. The singularity of f, which is closest to the origin, can often be found easily from the expression that defines $f(z)$; so the radius of convergence of a Maclaurin series can often be easily found.

Note that these Taylor coefficients are *uniquely determined* for the function f. This is true also for a non-analytic function, for example if $f \in C^p[a,b]$, although in this case the coefficient a_j exists only for $j \leq p$. [4]

There are several expressions for the remainder $R_n(z)$, when the expansion for $f(z)$ is truncated after the term that contains z^{n-1}. In order to simplify the notation, we put $a = 0$, i.e., we consider the Maclaurin series. The following *integral form* can be obtained by the application of repeated integration by parts to the integral $z\int_0^1 f'(zt)dt$; the details are left for Problem 14 b.

$$R_n(z) = z^n \int_0^1 \frac{(1-t)^{n-1}}{(n-1)!} f^{(n)}(zt)dt; \quad |R_n(z)| \leq \frac{|z|^n \max_{0\leq t\leq 1} |f^{(n)}(zt)|}{n!}. \tag{3.1.4}$$

This holds also in the complex case; if f is analytic on the segment from 0 to z one integrates along this segment, i.e., for $0 \leq t \leq 1$, otherwise another path is to be chosen. [5]

For a real-valued function, **Lagrange's formula** [6] for the remainder

$$R_n(x) = \frac{f^{(n)}(\xi)x^n}{n!}, \qquad \xi \in [0,x], \tag{3.1.5}$$

is obtained by the mean value theorem of integral calculus.

For complex-valued functions and, more generally, for vector-valued functions the mean value theorem and Lagrange's remainder term are not valid with a single ξ. (Sometimes componentwise application with different ξ is possible.) A different form for the remainder, valid in the complex plane is given in Sec. 3.1.4, in terms of the *maximum modulus* $M(r) = \max_{|z|=r} |f(z)|$, which may sometimes be easier to estimate than the n'th derivative.

A power series is uniformly convergent in any closed bounded region strictly inside its circle of convergence. Roughly speaking, the series can be manipulated like a polynomial, as long as z belongs to such a region;

- it can be integrated or differentiated term by term,

[3]This infinite series is in the general case called a Taylor series, while the special case, $a = 0$, is by tradition called a Maclaurin series. Brook Taylor(1685–1731), who announced his theorem in 1712, and Colin Maclaurin (1698–1746) were British mathematicians.

[4]Also the remainder formulas mentioned here require only that $f \in C^n$. It is thus not necessary that the infinite expansion converges or even exists.

[5]A generalization of this to vector-valued functions of vector-valued variables will be given in the appendix to Ch. 11.

[6]Joseph Louis Lagrange was born at Turin 1736 and died at Paris 1813. He made fundamental contributions to most branches of Mathematics and Mechanics.

- substitutions can be performed, and terms can be rearranged,
- it can be multiplied by another power series, etc.

If $f(z) = \sum a_j z^j$, $g(z) = \sum b_k z^k$, then $f(z)g(z) = \sum c_n z^n$, where

$$c_n = a_0 b_n + a_1 b_{n-1} + \ldots + a_n b_0 = \sum_{j=0}^{n} a_j b_{n-j}. \tag{3.1.6}$$

The expression on the right side of (3.1.6) is called the **convolution** of the coefficient sequences of f and g.

The use of the Taylor coefficient formula and Lagrange's form of the remainder may be inconvenient, and it is often easier to obtain an expansion by manipulating some known expansions. The geometric series,

$$\frac{1}{1-z} = 1 + z + z^2 + z^3 + \cdots + z^{n-1} + \frac{z^n}{1-z}, \quad z \neq 1, \tag{3.1.7}$$

is of particular importance; note that the remainder $z^n/(1-z)$ is valid even when the expansion is divergent. (In Sec. 3.1.8 we shall see that this can be a very useful fact.)

Example 3.1.5. Set $x = -t^2$ in the geometric series, and integrate:

$$\int_0^x (1+t^2)^{-1} dt = \sum_{j=0}^{n-1} \int_0^x (-t^2)^j dt + \int_0^x (-t^2)^n (1+t^2)^{-1} dt.$$

Using the mean-value theorem of integral calculus on the last term we get

$$\arctan x = \sum_{j=0}^{n-1} \frac{(-1)^j x^{2j+1}}{2j+1} + \frac{(1+\xi^2)^{-1}(-1)^j x^{2n+1}}{2n+1}, \tag{3.1.8}$$

for some $\xi \in \operatorname{int}[0, x]$. Both the remainder term and the actual derivation are much simpler than what one would get by using Taylor's formula with Lagrange's remainder term. Notice also that Theorem 3.1.4 is applicable to the series obtained above for all x and n, even for $|x| > 1$, when the infinite power series is divergent.

Some useful expansions are collected in Table 3.1.1.These formulas are used quite often; the reader is recommended to memorize the expansions. "Remainder ratio" means the *ratio* of the remainder to the first neglected term. In the table, ξ means a number between 0 and x.

The exponent k in $(1+x)^k$ is not necessarily an integer; it can even be an irrational or a complex number. This function may be defined as $(1+x)^k = e^{k \ln(1+x)}$. Since $\ln(1+x)$ is **many-valued**, $(1+x)^k$ is many-valued too, unless k is an integer. We can, however, make them single-valued by forbidding the complex variable x to take real values less than -1. In other words, we make a **cut** along the real axis from -1 to ∞ that the complex variable must not cross. [7] We obtain the **principal**

[7]The cut is outside the circle of convergence.

Table 3.1.1. *Maclaurin expansions for elementary functions.*

Function	Expansion ($x \in \mathbf{C}$)	Remainder ratio ($x \in \mathbf{R}$)
$(1-x)^{-1}$	$1 + x + x^2 + x^3 + \cdots$ if $\lvert x\rvert < 1$	$(1-x)^{-1}$ if $x \neq 1$
$(1+x)^k$	$1 + kx + \binom{k}{2}x^2 + \binom{k}{3}x^3 + \cdots$ if $\lvert x\rvert < 1$	$(1+\xi)^{k-n}$ if $x > -1$
$\ln(1+x)$	$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$ if $\lvert x\rvert < 1$	$(1+\xi)^{-1}$ if $x > -1$
e^x	$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$ all x	e^ξ, all x
$\sin x$	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$ all x	$\cos\xi$, all x, n odd
$\cos x$	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$ all x	$\cos\xi$, all x, n even
$\frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$	$x + \frac{x^3}{3} + \frac{x^5}{5} + \cdots$ if $\lvert x\rvert < 1$	$\frac{1}{1-\xi^2}$, $\lvert x\rvert < 1$, n even
$\arctan x$	$x - \frac{x^3}{3} + \frac{x^5}{5} + \cdots$ if $\lvert x\rvert < 1$	$\frac{1}{1+\xi^2}$, all x

branch by requiring that $\ln(1+x) > 0$ if $x > 0$. Let $1 + x = re^{i\phi}$, $r > 0$, $\phi \to \pm\pi$. Note that

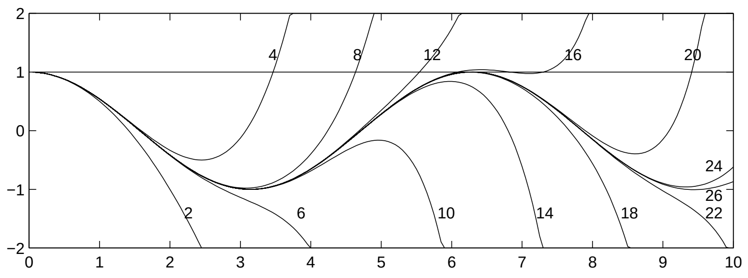
$$1 + x \to -r, \qquad \ln(1+x) \to \ln r + \begin{cases} +i\pi, & \text{if } \phi \to \pi; \\ -i\pi, & \text{if } \phi \to -\pi; \end{cases}. \tag{3.1.9}$$

Series expansions for many other functions can be found in the classical handbook of Abramowitz and Stegun [1]. Lebedev's treatise on Special Functions [24] provides, in particular in the chapter about the gamma function, numerous examples of the use of series expansions and analytic continuation, which are efficient as well as important and beautiful.

Example 3.1.6. The following procedure can generally be used in order to find the *expansion of the quotient of two expansions.* We illustrate it on a case, where the result is of interest to us later.

The **Bernoulli numbers** B_n are defined by the following Maclaurin series, [8]

[8]Jacob (or James) Bernoulli (1654-1705) Swiss mathematician, one of the earliest to realize how powerful was the infinitesimal calculus. The Bernoulli numbers were published posthumously in 1713, in his fundamental work Ars Conjectandi (on Probability). The notation for Bernoulli numbers varies in the literature. Our notation seems to be the most common in modern texts.



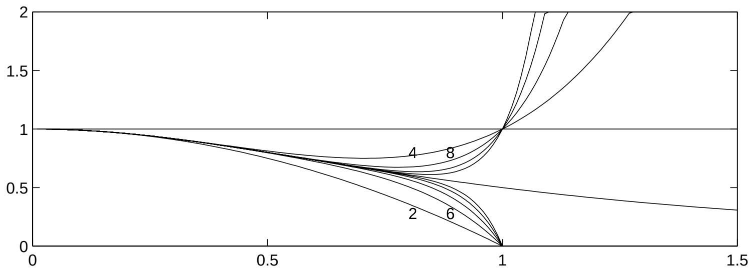


Figure 3.1.4. *The partial sums of the Maclaurin expansions for two functions. The upper curves are for $f(x) = \cos x$, $n = 0 : 2 : 26$, $0 \le x \le 10$. This series converges for all x, but the rounding errors cause trouble for large values of x, see Sec. 3.1.7, Ill-conditioned series. The lower curves are for $f(x) = 1/(1+x^2)$, $n = 0 : 2 : 18$, $0 \le x \le 1.5$. The convergence radius is 1 in this case.*

$$\frac{x}{e^x - 1} \equiv \sum_{j=0}^{\infty} \frac{B_j x^j}{j!} \tag{3.1.10}$$

If we multiply this equation by the denominator, we obtain

$$x \equiv \left(\sum_{i=1}^{\infty} \frac{x^i}{i!} \right) \left(\sum_{j=0}^{\infty} \frac{B_j x^j}{j!} \right).$$

Matching the coefficients of x^n, $n \ge 1$, on both sides, we obtain a recurrence relation for the Bernoulli numbers, which can be written in the form

$$B_0 = 1, \quad \sum_{j=0}^{n-1} \frac{1}{(n-j)!} \frac{B_j}{j!} = 0, \quad n \ge 2, \quad \text{i.e.,} \sum_{j=0}^{n-1} \binom{n}{j} B_j = 0, \tag{3.1.11}$$

hence, $B_0 = 1$, $B_1 = -\frac{1}{2}$, $B_2 = \frac{1}{6}$, $B_3 = 0$, $B_4 = -\frac{1}{30}$, $B_5 = 0$, $B_6 = \frac{1}{42}, \dots$.

Several members of the same family enriched mathematics by their teaching and writings. Their role in the history of mathematics reminds of the role of the Bach family in the history of music.

We see that the Bernoulli numbers are rational. We shall now demonstrate that $B_n = 0$, *when* n *is odd, except for* $n = 1$.

$$\frac{x}{e^x-1}+\frac{x}{2}=\frac{x}{2}\frac{e^x+1}{e^x-1}=\frac{x}{2}\frac{e^{x/2}+e^{-x/2}}{e^{x/2}-e^{-x/2}}. \tag{3.1.12}$$

Since the last term is an even function, i.e., its value is unchanged when x is replaced by $-x$, its Maclaurin expansion contains only even powers of x.

The singularities of this function are poles at $x = 2n\pi i$, $n = \pm1, \pm2, \pm3, \ldots$, hence the radius of convergence is 2π. It can be shown by means of (3.1.20) or Lemma 3.3.1 that [9]

$$B_j/j! = (-1)^{j/2}(2\pi)^{-j} + O\Big((2\pi)^{-2j}\Big), \quad (j \text{ even}, \ j \to \infty). \tag{3.1.13}$$

Further properties of Bernoulli numbers and the related Bernoulli polynomials are presented in Sec. 3.3.4, where they occur as coefficients in the important Euler–Maclaurin formula.

The **Euler numbers** E_n, which will be used later, are similarly defined by the generating function

$$\frac{1}{\cosh z} \equiv \sum_{n=0}^{\infty} \frac{E_n z^n}{n!}, \quad |z| < \frac{\pi}{2}. \tag{3.1.14}$$

Obviously $E_n = 0$ for all odd n. It can be shown that the Euler numbers are integers, $E_0 = 1$, $E_2 = -1$, $E_4 = 5$, $E_6 = -61$; see Problem 7e.

Example 3.1.7. Let $f(x) = (x^3+1)^{-\frac{1}{2}}$. Compute $\int_{10}^{\infty} f(x)dx$ to 9 decimal places, and $f'''(10)$, with at most 1% error. Since x^{-1} is fairly small, we expand in powers of x^{-1}:

$$\begin{aligned} f(x) &= x^{-3/2}(1+x^{-3})^{-1/2} = x^{-3/2}\Big(1-\frac{1}{2}x^{-3}+\frac{1\cdot3}{8}x^{-6}-\ldots\Big) \\ &= x^{-1.5}-\frac{1}{2}x^{-4.5}+\frac{3}{8}x^{-7.5}-\ldots. \end{aligned}$$

By integration,

$$\int_{10}^{\infty} f(x)dx = 2\cdot10^{-0.5}-\frac{1}{7}10^{-3.5}+\frac{3}{52}10^{-6.5}+\ldots = 0.632410375.$$

Each term is less than 0.001 of the previous term.

By differentiating the series three times, we similarly obtain

$$f'''(x) = -\frac{105}{8}x^{-4.5}-\frac{1,287}{16}x^{-7.5}+\ldots.$$

For $x = 10$ the second term is less than 1% of the first; the terms after the second decrease quickly and are negligible. One can show that the magnitude of each term

[9] This is useful, even when j is rather small, e.g., we obtain $B_6 \approx 1/42.7$

is less than $8\,x^{-3}$ of the previous term. We get $f'''(10) = -4.12\,10^{-4}$ to the desired accuracy. The reader is advised to carry through the calculation in more detail.

Example 3.1.8. *How to compute* $\sinh x$*.* On a binary computer with machine unit $u = 2^{-36} \approx 1.46\cdot 10^{-11}$, one wishes to compute $\sinh x$ with good *relative* accuracy, both for small and large $|x|$, at least moderately large. Assume that e^x is computed with a relative error less than $5u$ in the given interval. The formula $(e^x - e^{-x})/2$ for $\sinh x$ is sufficiently accurate except when $|x|$ is small and cancellation occurs. Hence for $|x| \ll 1$, e^x and e^{-x} and hence $(e^x - e^{-x})/2$ can have *absolute* errors of order of magnitude (say) $5u$. Then the *relative* error in $(e^x - e^{-x})/2$ can have magnitude $\approx 5u/|x|$; for example, this is more than 500% for $x \approx 10^{-11}$.

When $|x|$ is small one can instead use two terms in the series expansion for $\sinh x$,

$$\sinh x = x + x^3/3! + x^5/5! + \dots,$$

one gets an absolute truncation error which is about $x^5/120$, and a round-off error of the order of $2u|x|$. Thus the formula $x + x^3/6$ is better than $(e^x - e^{-x})/2$ if

$$|x|^5/120 + 2u|x| < 5u.$$

If $2u|x| \ll 5u$, we have $|x|^5 < 600u \approx 300\cdot 2^{-35}$, or $|x| < 300^{1/5}\cdot 2^{-7} \approx 0.0243$, (which shows that $2u|x|$ really could be ignored in this rough calculation). Thus, if one switches from $(e^x - e^{-x})/2$ to $x + x^3/6$ for $|x| < 0.0243$, the relative error will nowhere exceed $5u/0.0243 \approx 4\cdot 10^{-9}$. If one needs higher accuracy, one can take more terms in the series, so that the switch can occur at a larger value of $|x|$.

For very large values of $|x|$ one must expect a relative error of order of magnitude $|xu|$ because of round-off error in the argument x. See Problem 5 for a modernization of this example.

The analytic functions have many important properties that you may find in any text on complex analysis. A good summary for the purpose of numerical mathematics is found in the first chapter of Stenger [32]. Two important properties are contained in the following lemma.

Lemma 3.1.6. *An analytic function can only have a finite number of zeros in a compact subset of the region of analyticity, unless the function is identically zero.* [10]

Suppose that two functions f_1 and f_2 are analytic in regions D_1 and D_2, respectively. Suppose that $D_1 \cap D_2$ contains an interval throughout which $f_1(z) = f_2(z)$.

Then $f_1(z) = f_2(z)$ in the intersection $D_1 \cap D_2$.

Proof. We refer, for the first part, to any text on Complex Analysis. We here follow Titchmarsh [34] closely. The second part follows by the application of the

[10]The region of analyticity of a function $f(z)$ is an *open* set. If, e.g., we say that $f(z)$ is analytic on a closed real interval, it means that there exists an open set, where $f(z)$ is analytic, which contains this interval.

first part to the function $f_1 - f_2$. □

A consequence of this is known as *the permanence of functional equations*, i.e., in order to prove the validity of a functional equation (or "a formula for a function") in a region of the complex plane, it may be sufficient to prove its validity in (say) an interval of the real axis, under the conditions specified in the lemma.

Example 3.1.9. *The permanence of functional equations.* We know from elementary real analysis that the functional equation

$$e^{(p+q)z} = e^{pz}e^{qz}$$

holds for all $z \in \mathbf{R}$. We also know that all the three functions involved are analytic for all $z \in \mathbf{C}$. Set in the lemma $D_1 = D_2 = \mathbf{C}$, and let "the interval" be any compact interval of $\mathbf{R}$. The lemma then tells that that the displayed equation holds for all complex z. The right and the left hand side then have identical power series. Applying the convolution formula and matching the coefficients of z^n, we obtain

$$\frac{(p+q)^n}{n!} = \sum_{j=0}^{n} \frac{p^j}{j!}\frac{q^{n-j}}{(n-j)!}, \quad \text{i.e.,} \quad (p+q)^n = \sum_{j=0}^{n} \binom{n}{j} p^j q^{n-j}.$$

This is not a very sensational result. It is more interesting to start from the following functional equation

$$(1+z)^{p+q} = (1+z)^p(1+z)^q.$$

The same argumentation holds, except that D_1, D_2 are equal to the complex plane with a cut, and that the Maclaurin series is convergent in the unit disk only. We obtain the equations

$$\binom{p+q}{n} = \sum_{j=0}^{n} \binom{p}{j}\binom{q}{n-j}, \qquad n = 0, 1, 2, \ldots. \tag{3.1.15}$$

(They can also be proved by induction, but it is not needed.) This sequence of algebraic identities, where *each identity contains a finite number of terms*, is equivalent to the above functional equation.

We shall see that this observation is useful for motivating certain "*symbolic computations*" with power series, that can provide elegant derivations of useful formulas in numerical mathematics.

Now we may consider the aggregate of values of $f_1(z)$ and $f_2(z)$ at points interior to D_1 or D_2 as a single analytic function f. Thus f is analytic in the union $D_1 \cup D_2$, and $f(z) = f_1(z)$ in D_1, $f(z) = f_2(z)$ in D_2.

The function f_2 may be considered as extending the domain in which f_1 is defined, and it is called a (single-valued) **analytic continuation** of f_1. In the same way f_1 is an analytic continuation of f_2. Analytic continuation denotes both this process of extending the definition of a given function, and the result if the process.

We shall see examples of this, e.g. in Sec. 3.3. Under certain conditions the analytic continuation is unique.

Theorem 3.1.7. *Suppose that a region D is overlapped by regions D_1, D_2, and that $(D_1 \cap D_2) \cap D$ contains an interval. Let f be analytic in D, and let f_1 be an analytic continuation of f to D_1, and let f_2 an analytic continuation of f to D_2, so that $f(z) = f_1(z) = f_2(z)$ in $(D_1 \cap D_2) \cap D$.*

Then either of these functions provides a single-valued analytic continuation of f to $D_1 \cap D_2$. The results of the two processes are the same.

Proof. Since $f_1 - f_2$ is analytic in $D_1 \cap D_2$, and $f_1 - f_2 = 0$ in the set $(D_1 \cap D_2) \cap D$, which contains an interval, it follows from the lemma that $f_1(z) = f_2(z)$ in $D_1 \cap D_2$, which proves the theorem. $\square$

If the set $(D_1 \cap D_2) \cap D$ is *void*, the conclusion in the theorem *may not be valid.* We may still consider the aggregate of values as a single analytic function, but *this function can be multi-valued in $D_1 \cap D_2$.*

In some contexts, algebraic recurrence relations are convenient to use for computing the coefficients in Maclaurin expansions, in particular if only a moderate number of coefficients are wanted. We shall study a few examples.

Example 3.1.10. *Expansion of a composite function.*

Let $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots$, $\Phi(z) = c_0 + c_1 z + c_2 z^2 + \ldots$, be given, analytic at the origin. Find the power series for $g(x) = \Phi(f(x)) = b_0 + b_1 x + b_2 x^2 + \ldots$. In particular, we shall study the case $\Phi(z) = e^z$.

The first idea we may think of is to substitute the expansion $a_0 + a_1 x + a_2 x^2 + \ldots$ for z into the power series for $\Phi(z)$. This is, however, *no good unless* $a_0 = 0$, because $(f(x))^k = a_0^k + k a_0^{k-1} a_1 x + \ldots$ gives a contribution to, e.g., b_0, b_1 for every k, so we cannot successively compute the b_j by *finite* computation.

Now suppose that $a_0 = 0$, $a_1 = 1$, [11] i.e., $f(x) = x + a_2 x^2 + a_3 x^3 + \ldots$. Then $z^k = x^k + k a_2 x^{k+1} + \ldots$. We obtain

$$g(x) = c_0 + c_1 x + (c_1 a_2 + c_2) x^2 + (c_1 a_3 + 2 c_2 a_2 + c_3) x^3 + \ldots,$$

and the coefficients of $g(x)$ come out recursively,

$$b_0 = c_0;\ b_1 = c_1;\ b_2 = c_1 a_2 + c_2;\ b_3 = c_1 a_3 + 2 c_2 a_2 + c_3; \ldots$$

b_j depends only on a_k, c_k, $k \le j$. Now consider the case $\Phi(z) = e^z$, i.e., $c_n = 1/n!$. We first see that it is then easy to handle the case that $a_0 \neq 0$, since $e^{f(x)} = e^{a_0} e^{a_1 x + a_2 x^2 + a_3 x^3 + \cdots}$.

But there exists a more important simplification if $\Phi(z) = e^z$. Note that g satisfies the differential equation $g'(x) = f'(x) g(x)$, $g(0) = e^{a_0}$. Hence

$$\sum_{n=0}^{\infty} (n+1) b_{n+1} x^n \equiv \sum_{j=0}^{\infty} (j+1) a_{j+1} x^j \sum_{k=0}^{\infty} b_k x^k.$$

[11] The assumption $a_1 = 1$ is not important, but it simplifies the writing.

Apply the convolution formula (3.1.6), and match the coefficients of x^n on the two sides.

$$b_0 = e^{a_0}, \quad (n+1)b_{n+1} = a_1 b_n + 2a_2 b_{n-1} + \ldots + (n+1)a_{n+1}b_0, \quad (n = 0, 1, 2, \ldots).$$

This recurrence relation is more easily programmed than the general procedure indicated above. Other functions that satisfy appropriate differential equations can be treated similarly; see Problem 8. See also Knuth [23], Sec. 4.7.

Example 3.1.11. *A matrix representation of a truncated power series.*

In a programming language, where matrices are easy to handle, it is often practical to represent a truncated power series $f_N(z) = \sum_{j=0}^{N-1} a_j z^j$ by a triangular $N \times N$ matrix, $f_N(S) = \sum_{j=0}^{N-1} a_j S^j$, where S is a **shift matrix**. For $N = 4$,

$$S = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}; \qquad S \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ 0 \end{pmatrix}; \qquad f_N(S) = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ 0 & a_0 & a_1 & a_2 \\ 0 & 0 & a_0 & a_1 \\ 0 & 0 & 0 & a_0 \end{pmatrix}.$$

Sums and products of matrices of the same structure as $f_N(S)$ do still have the same structure [12] , and similarly for the inverse of $f_N(S)$ if $a_0 \neq 0$. Note that $S^N = 0$, a so-called **nilpotent** matrix. This is why this matrix representation is suited for truncated power series. If the Maclaurin series of $f(z)$, $g(z)$ are truncated to $f_N(z)$, $g_N(z)$ then you can read the first N coefficients of the Maclaurin series for $f(z)g(z)$ in the first row of the matrix $f_N(S) \cdot g_N(S)$.

Similarly, you find the Bernoulli numbers (divided by factorials), see Example 3.1.6, by inverting the matrix $f_N(S)$ associated with a truncated expansion of $(e^z - 1)/z$.

If you know $g(S)$, and if your language contains functions for matrix exponential, matrix logarithm, matrix square root etc., these can be used for obtaining the Maclaurin expansions of $\exp(g(z)), \ln(g(z)), \sqrt{g(z)}, \ldots$.

More generally, if $f_N(z) = \sum_{j=0}^{N-1} a_j z^j$, the first N coefficients of the expansion of $f(g(z))$ are found in the first row of $\sum_{j=0}^{N-1} a_j g_N(S)^j$. You can use Horner's rule for computing the matrix polynomial.

These algorithms may not seem efficient from the point of view of the number of arithmetic operations or the storage requirements. Since the derivation of expansion coefficients is usually a small task for a modern personal computer, you do not need to worry about smart matrix handling. The programming becomes very easy. If, in some sense, you measure the total efforts of yourself and your computer, these algorithms may become competitive. See, e.g., Problem 8(d,e).

Henrici [20, § 1.3] represents formal power series by *infinite* upper triangular Toeplitz matrices.

[12] Matrices (not necessarily triangular), whose entries are constant along each diagonal, are called **Toeplitz matrices**. The product of two non-triangular Toeplitz matrices, however, is in general not a Toeplitz matrix. Similarly for the inverse.

When a large number of coefficients is needed, the *Cauchy+FFT method*, see Sec. 3.1.4, can be an efficient alternative to the procedures exemplified in the present subsection and in Problem 8.

Knuth, loc. cit., treats other general devices for the manipulation of power series. He presents, for example, several algorithms for **power series reversion**, i.e., for finding the power series for the inverse function to the function defined by a given power series with $a_0 = 0$. He gives both a classical algorithm due to Lagrange 1768, and a relatively recent algorithm due to Brent and Kung 1978. The latter is based on an adaptation, to formal power series, of Newton's idea for solving a numerical algebraic equation. It doubles the number of terms in each iteration; compare the quadratic convergence in the numerical case, e.g., in the square root algorithm, Sec. 1.2.

The following simple cases of power series reversion are often sufficient and useful in low order computations.

$$y = x + ax^k + \ldots, (k > 1), \Rightarrow \; x = y - ax^k - \ldots = y - ay^k - \ldots; \quad (3.1.16)$$

$$y = x + a_2x^2 + a_3x^3 + \ldots \;\Rightarrow\; x = y - a_2y^2 + (2a_2^2 - a_3)y^3 + \ldots; \quad (3.1.17)$$

Formulas like those mentioned in this subsection, and in Problem 8, are often used in packages for **symbolic differentiation** and **automatic differentiation**. Expanding a function into a Taylor series is equivalent to finding the sequence of derivatives of the function at a given point.

COMMENT: The goal of *symbolic* differentiation is to obtain analytic *expressions* for derivatives of functions given in analytic form. This is handled by computer algebra systems, e.g., Maple or Mathematica.

In contrast, the goal of *automatic* differentiation is to create an algorithm (a program) for the computation of the *numerical values* of the derivatives of a function that is given in the form of an algorithm (a program). Typical applications are in the solution of ordinary differential equations by Taylor expansion, see Example 3.1.2 and Sec. 13.3. Such techniques are also used in optimization. See examples in Ch. 11. See, e.g., Griewank and Corliss [15]; a software package is presented in [16].

In numerical computation a series should be regarded as a finite expansion together with a remainder. Taylor's formula with the remainder (3.1.7) is valid for any function $f \in C^n[a, a + x]$, but *the infinite series is valid only if the function is analytic in a complex neighborhood of* a.

If a function is not analytic at 0, *it can happen that the Maclaurin expansion converges to a wrong result.* A classical example, see Appendix to Ch. 6 in Courant [8], reads: $f(x) = e^{-1/x^2}$ for $x \neq 0$, $f(0) = 0$. It can be shown that all its Maclaurin coefficients are zero. This trivial Maclaurin expansion converges for all x, *but the sum is wrong for* $x \neq 0$. There is nothing wrong with the use for Taylor's formula as a finite expansion with a remainder. Although the remainder, which in this case equals $f(x)$ itself, does not tend to 0 as $n \to \infty$ (for a fixed $x \neq 0$), it tends to 0 faster than any power of x, as $x \to 0$, for any fixed n. The "expansion" gives excellent absolute accuracy when x is small, e.g., 43 decimal digits for $x = 0.1$, but the relative error is 100%. Also note that this function (and there are lots of other examples) can be added to any function without changing its Maclaurin expansion.

From the point of view of complex analysis, however, the origin is a singular point for this function, note, e.g., that $|f(z)| \to \infty$ as $z \to 0$ along the imaginary axis, and this prevents the application of any theorem that would guarantee that the infinite Maclaurin series represents the function. This trouble does not occur for a truncated Maclaurin expansion around a point, where the function under consideration is analytic. The size of the first non-vanishing neglected term then gives a good hint about the truncation error, when $|z|$ is a small fraction of the radius of convergence.

The above example may sound like a purely theoretical matter of curiosity. We emphasize this *distinction between the convergence and the validity of an infinite expansion* in this text, as a background to other expansions of importance in numerical computation, e.g., the Euler–Maclaurin expansion in Sec. 3.3.4, which may converge to the wrong result, also in the application to a well-behaved analytic function. On the other hand, we shall see, e.g., in Sec. 3.1.8, that divergent expansions can sometimes be very useful. The universal recipe is to consider an infinite series as a finite expansion plus a remainder term. Convergence of an expansion is neither necessary nor sufficient for its success in practical computation.

A power series is, however, not only a means for numerical computation; it is also an aid for deriving formulas in numerical mathematics and in other branches of applied mathematics. Then one has another, a more algebraic, aspect of power series that we shall briefly introduce. A more strict and detailed treatment (65 pages) is found in Henrici [20, Ch. 1] and the literature quoted there (including Bourbaki!). This aspect will be applied extensively to operator series in Sec. 3.2.

In a **formal power series**, $\mathbf{P} = a_0 + a_1\mathbf{x} + a_2\mathbf{x}^2 + \cdots$, the coefficients a_j may be real or complex numbers (or elements in some other field), while $\mathbf{x}$ is an algebraic "indeterminate"; $\mathbf{x}$ and its powers can be viewed as placekeepers. No real or complex values are assigned to $\mathbf{x}$ and $\mathbf{P}$. Convergence, divergence and remainder term have no relevance for formal power series. In the language of algebra, the set of formal power series is an integral domain.

We do *not* consider formal power series with several indeterminates. There may occur expressions with several boldtype symbols. Only one of them is the indeterminate, and the other must be shorthand notations for formal power series with respect to this indeterminate.

The sum of $\mathbf{P}$ and another formal power series, $\mathbf{Q} = b_0 + b_1\mathbf{x} + b_2\mathbf{x}^2 + \cdots$, is *defined* as $\mathbf{P} + \mathbf{Q} = (a_0 + b_0) + (a_1 + b_1)\mathbf{x} + (a_2 + b_2)\mathbf{x}^2 + \cdots$. Similarly, the *Cauchy product* is *defined* as $\mathbf{PQ} = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots$, where the coefficients are given by the convolution formula (3.1.6),

$$c_n = a_0 b_n + a_1 b_{n-1} + \ldots + a_n b_0 = \sum_{j=0}^{n} a_j b_{n-j}.$$

Other operations are defined without surprises, e.g., the derivative of $\mathbf{P}$ is *defined* as $\mathbf{P}' = 1a_1 + 2a_2\mathbf{x} + 3a_3\mathbf{x}^2 + \ldots$. The usual rules for differentiation are still valid. The identity element is the series $\mathbf{I} := 1 + 0\mathbf{x} + 0\mathbf{x}^2 + \ldots$. If $a_0 \neq 0$, $\mathbf{P}^{-1}$ exists, and the division of two formal power series is performed as indicated in Example 3.1.6.

If a function f of a complex variable z is analytic at the origin, then *we define*

[13] $f(\mathbf{x})$ as the formal power series with the same coefficients as the Maclaurin series for $f(z)$. In the case of a multivalued function we take the principal branch.

The other operations on power series studied in this subsection, are valid also for formal power series, e.g., substitution and change of variable (indeterminate)—in Example 3.1.10 and in Problem 8—and the reversal of power series.

There is a kind of "permanence of functional equations" also for the generalization from a function $g(z)$ of a complex variable that is analytic at the origin, to the formal power series $g(\mathbf{x})$. We illustrate the general principle on an important special example that we formulate as a lemma, since we shall need it in the next section.

Lemma 3.1.8.

$$(e^{\mathbf{x}})^{\theta} = e^{\theta\mathbf{x}}, \quad (\theta \in \mathbf{R}). \tag{3.1.18}$$

Proof. Let the coefficient of $\mathbf{x}^j$ in the expansion of the left hand side be $\phi_j(\theta)$. The corresponding coefficient for the right hand side is $\theta^j/j!$. If we replace $\mathbf{x}$ by a complex variable z, the power series coefficients are the same, and we know that $(e^z)^{\theta} = e^{\theta z}$, hence $\phi_j(\theta) = \theta^j/j!$, $j = 1, 2, 3\ldots$, hence $\sum_0^{\infty} \phi_j(\theta)\mathbf{x}^j = \sum_0^{\infty}(\theta^j/j!)\mathbf{x}^j$, and the lemma follows. $\square$

The theory of formal power series has been developed with other applications than ours in mind, and works under more general conditions than we need. When *we* replace the indeterminate by a complex variable z, our formal series becomes the Maclaurin series of a function that is assumed to be analytic for $z = 0$. We have not seen any discussion of this particular case in the literature about formal power series. The possibilities offered by the one-to-one correspondence between such power series and functions analytic at the origin seem to be ignored in that literature. The applications to operator series in Sec. 3.2, show that, in spite of this analyticity assumption, the operator series, after multiplication by an operand, may lead to a much more general class of series, not necessarily convergent in the usual sense but still numerically useful.

Example 3.1.12.

Find (if possible) a formal power series $\mathbf{Q} = 0 + b_1\mathbf{x} + b_2\mathbf{x}^2 + b_3\mathbf{x}^3 + \ldots$, that satisfies the equation

$$e^{-\mathbf{Q}} = 1 - \mathbf{x}, \tag{3.1.19}$$

where $e^{-\mathbf{Q}} = 1 - \mathbf{Q} + \mathbf{Q}^2/2! - \ldots$. We can, in principle, determine an arbitrarily long sequence $b_1, b_2, b_3, \ldots b_k$ by matching the coefficients of $\mathbf{x}, \mathbf{x}^2, \mathbf{x}^3, \ldots \mathbf{x}^k$, in the two sides of the equation. We display the first three equations.

$$1 - (b_1\mathbf{x} + b_2\mathbf{x}^2 + b_3\mathbf{x}^3 + \ldots) + (b_1\mathbf{x} + b_2\mathbf{x}^2 + \ldots)^2/2 - (b_1\mathbf{x} + \ldots)^3/6 + \ldots$$

$$= 1 - 1\mathbf{x} + 0\mathbf{x}^2 + 0\mathbf{x}^3 + \ldots.$$

[13]Henrici, loc. cit., does not use this concept—it may not be established.

We see that the matching can be done in a unique [14] way:

$$\begin{aligned} -b_1 = -1 &\Rightarrow\ b_1 = 1; \\ -b_2 + b_1^2/2 = 0 &\Rightarrow\ b_2 = 1/2; \\ -b_3 + b_1 b_2 - b_1/6 = 0 &\Rightarrow\ b_3 = 1/3; \end{aligned}$$

There exists, however, *a much easier way.* For the analogous problem with a complex variable z, i.e., to solve the equation $e^{-q(z)} = 1 - z$, $q(0) = 0$, we can apply the same procedure, and we obtain the same recursive formulas, and hence *the same coefficients* $b_j, j = 1, 2, 3, \ldots$. In this case, however, we know the explicit solution: $q(z) = -\ln(1 - z) = \sum_1^\infty z^j/j$ (the principal branch). It follows that *the solution for the formal series reads* [15]

$$\mathbf{Q} = -\ln(1 - \mathbf{x}) = \sum_{j=1}^{\infty} \mathbf{x}^j/j.$$

This example will be applied in Example 3.2.18 to the derivation of formulas for numerical differentiation.

The theory of formal power series can in a similar way justify many elegant "symbolic" applications of power series for deriving mathematical formulas. Note that it is not necessary to set up the recurrence relations explicitly. We did it in the previous examples just for explaining the principle. It is enough to make sure that such relations exist, and that they determine the coefficients uniquely. The applications to operator series in Sec. 3.2, hopefully will give you a feeling for the circumstances under which the calculation of the coefficients in a formal series certainly lead to the same coefficients as in an analogous computation for an analytic function of a complex variable, where a rich collection of methods is available, including also the matrix representation described in Example 3.1.11 and the Cauchy+FFT method, which is the topic of the next subsection.

3.1.4 The Cauchy+FFT Method.

Suppose that the value $f(z)$ of an analytic function can be computed at any point inside and on the circle $C_r = \{z : |z - a| = r\}$, and set

$$M(r) = \max |f(z)|, \quad z \in C_r, \quad z = a + re^{i\theta}, \quad z' = a + r'e^{i\theta}, \quad (r' < r).$$

Then the following formulas, due to Cauchy, for the coefficients of the Taylor expansion around a can be very useful,

$$\begin{aligned} a_n &= \frac{1}{2\pi i} \int_{C_r} (z - a)^{-(n+1)} f(z)\, dz \\ &= \frac{r^{-n}}{2\pi} \int_0^{2\pi} e^{-ni\theta} f(a + re^{i\theta})\, d\theta. \end{aligned} \tag{3.1.20}$$

[14] Note the role that the assumption that $b_0 = 0$ (the principal branch) plays here.

[15] The first three coefficients are, of course, the same as the coefficients computed above.

For a derivation, multiply the Taylor expansion (3.1.2) by $(z-a)^{-n-1}$, integrate term by term over C_r, and note that

$$\frac{1}{2\pi i}\int_{C_r}(z-a)^{j-n-1}\,dz=\frac{1}{2\pi}\int_0^{2\pi}r^{j-n}e^{(j-n)i\theta}\,d\theta=\begin{cases}1, & \text{if } j=n;\\ 0, & \text{if } j\neq n.\end{cases}\tag{3.1.21}$$

The following inequalities are useful consequences of (3.1.20).

$$\begin{aligned}|a_n|&\le r^{-n}M(r), &&(3.1.22)\\ |R_n(z')|&\le\sum_n^{\infty}|a_j(z'-a)^j|\le\frac{M(r)(r'/r)^n}{1-r'/r}.\end{aligned}$$

This form of the remainder term of a Taylor series is useful in theoretical studies, and also for practical purpose, if the **maximum modulus** $M(r)$ is easier to estimate than the n'th derivative.

Set $z=a+re^{i\theta}$, $\Delta\theta=2\pi/N$, and apply the trapezoidal rule to the second integral in (3.1.20). Then [16]

$$a_n\approx\tilde a_n=\frac{1}{Nr^n}\sum_{k=0}^{N-1}e^{-ink\Delta\theta}f(a+re^{ik\Delta\theta}),\quad n=0:N-1.\tag{3.1.23}$$

The approximate Taylor coefficients $\tilde a_n$ are here expressed by means of the so-called **Discrete Fourier Transform** of the function $f(a+re^{i\theta})$, also called discrete Fourier *analysis*. This transform will be studied more systematically in Ch. 4 and Ch. 9.

If N is a power of 2, it is shown in Sec. 4.5 that, given the N values $f(a+re^{i\Delta\theta})$, $0\le k\le N-1$, *no more than $N\log_2 N$ complex multiplications and additions are needed for the computation of all the N coefficients $\tilde a_n$*, if an implementation of the discrete Fourier transform known as the **Fast Fourier Transform (FFT)** is used; see Ch. 4. (Packages for interactive mathematical computation usually contain commands related to FFT.)

The *inverse* transformation, sometimes called discrete Fourier *synthesis*, reads (if $a=0$, $r=1$): Compute $\sum_{j=0}^{N-1}a_jz^j$, for $z=e^{ik\Delta\phi}$, $k=0:N-1$. *It is of the same type as the sum in (3.1.23), apart from the sign of i.* It can therefore also be performed by means of $O(N\log N)$ elementary operations, instead of the $O(N^2)$ operations that the most direct approach to this task requires.

Since the Taylor coefficients are equal to $f^{(n)}(a)/n!$, this is de facto a method for the accurate *numerical differentiation of an analytic function.* If r and N are chosen appropriately, it is more well-conditioned than most alternative methods, such as the difference approximations mentioned in Ch. 1; see also Sec. 3.2 and Ch. 4. It requires, however, complex arithmetic for convenient implementation. We call this the **Cauchy+FFT method** for Taylor coefficients and differentiation.

The question arises, how to choose N and r. Theoretically, any r less than the radius of convergence would do, but there may be trouble with cancellation, if

[16] See (1.2.6). Note that the integrand has the same value for $\theta=2\pi$ as for $\theta=0$.

r is small. On the other hand, the truncation error of the numerical integration usually increases with r. "Scylla and Charybdis situations" [17] like this are very common with numerical methods. As a rule such a situation is handled by a combination of numerical experimentation and theoretical analysis of a more or less simplified model, including a few elementary optimization calculations. We take the opportunity to exemplify below this type of "hard analysis" on this question.

We first derive two lemmas, which are important also in many other contexts. First we have a discrete analogue of equation (3.1.21).

Lemma 3.1.9. *Let p, N be integers. Then $\sum_{k=0}^{N-1} e^{2\pi ipk/N} = 0$, unless $p = 0$ or a multiple of N, in which case every term equals 1, and the sum equals N.*

Proof. If p is not a multiple of N, the sum is a geometric series, the sum of which is equal to $(e^{2\pi ip} - 1)/(e^{2\pi ip/N} - 1) = 0$. The rest of the statement is obvious. □

Lemma 3.1.10. *Suppose that $f(z) = \sum_0^\infty a_n(z-a)^n$ is analytic in the disc $|z-a| < \rho$. Let $\tilde{a}_n$ be defined by* (3.1.23)*, where $r < \rho$. Then*

$$\tilde{a}_n - a_n = a_{n+N}\, r^N + a_{n+2N}\, r^{2N} + a_{n+3N}\, r^{3N} + \dots, \quad 0 \le n < N. \qquad (3.1.24)$$

Proof. Since $\Delta\theta = 2\pi/N$,

$$\tilde{a}_n = \frac{1}{Nr^n} \sum_{k=0}^{N-1} e^{-2\pi ink/N} \sum_{m=0}^{\infty} a_m \left(re^{2\pi ik/N}\right)^m = \frac{1}{Nr^n} \sum_{m=0}^{\infty} a_m r^m \sum_{k=0}^{N-1} e^{2\pi i(-n+m)k/N}.$$

By the previous lemma, the inner sum of the last expression is zero, unless $m - n$ is a multiple of N. Hence (recall that $0 \le n < N$),

$$\tilde{a}_n = \frac{1}{Nr^n} \left(a_n r^n N + a_{n+N}\, r^{n+N} N + a_{n+2N}\, r^{n+2N} N + \dots\right),$$

from which equation (3.1.24) follows. □

Remark 3.1.1. If the expansion is two-sided, e.g., a complex Fourier series or a Laurent series, see Sec. 3.1.5, there are *more terms in* (3.1.24).

The details of the following example may be omitted without consequences for the understanding of most of the rest of the text, but the ideas may be important for some problems.

Example 3.1.13. "Scylla and Charybdis" in the Cauchy+FFT.

[17] According to American Heritage Dictionary Scylla is a rock on the Italian side of the Strait of Messina, opposite to the whirlpool Charybdis, personified by Homer (Ulysses) as a female sea monster who devoured sailors. The problem is to navigate safely between them.

Let $M(r)$ be the maximum modulus for the function $f(z)$ on the circle C_r, and denote by $M(r)U$ an upper bound for the error of a computed function value $f(z)$, $|z| = r$, where $U \ll 1$. Assume that rounding errors during the computation of $\tilde{a}_n$ are of minor importance.

Then, by (3.1.23), $M(r)U/r^n$ is a bound for the *rounding error* of $\tilde{a}_n$. (The rounding errors during the computation can be included by a redefinition of U.)

Next we shall consider the *truncation error* of (3.1.23). First we *estimate* the coefficients that occur in (3.1.24) by means of $\max|f(z)|$ on a circle with radius r'; $r' > r$, where r is the radius of the circle used in the *computation* of the first N coefficients. So, in (3.1.20) we substitute r', j for r, n, respectively, and obtain the inequality

$$|a_j| \le M(r')(r')^{-j}, \qquad 0 < r < r' < \rho.$$

The actual choice of r', strongly depends on the function f. [18] Put this inequality into (3.1.24), where we shall choose $r < r' < \rho$. Then

$$\begin{aligned}|\tilde{a}_n - a_n| &\le M(r')\left((r')^{-n-N}r^N + (r')^{-n-2N}r^{2N} + (r')^{-n-3N}r^{3N} + \dots\right)\\ &= M(r')(r')^{-n}\left((r/r')^N + (r/r')^{2N} + (r/r')^{3N} + \dots\right)\\ &= \frac{M(r')(r')^{-n}}{(r'/r)^N - 1}.\end{aligned}$$

We make a digression here, because this is an amazingly good result. The trapezoidal rule that was used in the calculation of the Taylor coefficients is typically expected to have an error that is $O\big((\Delta\theta)^2\big) = O\big(N^{-2}\big)$. This application is, however, a very special situation: a periodic analytic function is integrated over a full period. We shall return to this several times, next time in Sec. 3.3.4. In this case, for fixed values of r, r', the truncation error is $O\big((r/r')^N\big) = O\big(e^{-\eta/\Delta\theta}\big)$, where $\eta > 0$, $\Delta\theta \to 0$.

It follows that a bound for the total error of $\tilde{a}_n$, i.e., the sum of the bounds for the rounding and the truncation errors, is given by

$$UM(r)r^{-n} + \frac{M(r')(r')^{-n}}{(r'/r)^N - 1}. \tag{3.1.25}$$

Assume that a_n is requested to have a uniformly small absolute error for (say) $n = 0 : \hat{n}$, $\hat{n} \gg 1$.

First consider the *rounding error*. By the maximum modulus theorem, $M(r)$ is an increasing function, hence, for $r > 1$, $\max_n M(r)r^{-n} = M(r) > M(1)$. On the other hand, for $r \le 1$, $\max_n M(r)r^{-n} = M(r)r^{-\hat{n}}$. Let r^* be the value of r, for which this is minimal. Note that $r^* = 1$ unless $rM'(r)/M(r) = \hat{n}$ for some $r \le 1$.

Then try to determine N and $r' \in [r^*, \rho)$ so that, for $r = r^*$, the second term of (3.1.25) becomes much smaller than the first term, i.e., the truncation error is made negligible compared to the rounding error. *This works well if $\rho \gg r^*$. In such cases, we may therefore choose* $r = r^*$, and the total error is then just a little larger than $UM(r^*)(r^*)^{-\hat{n}}$.

[18] In rare cases we may choose $r' = \rho$.

For example, if $f(z) = e^z$ then $M(r) = e^r$, $\rho = \infty$. In this case $r^* = 1$ (since $\hat{n} \gg 1$). Then we shall choose N and $r' = N$, so that $e^{r'}/((r')^N - 1) \ll eU$. One can show that it is sufficient to choose $N \gg |\ln U/\ln|\ln U||$. For instance, if $U = 10^{-15}$, this is satisfied with a wide margin by $N = 32$. On a computer, the choice $r = 1$, $N = 32$, gave (with 53 bits floating point arithmetic) an error less than $2\cdot 10^{-16}$. The results were much worse for $r = 10$, and for $r = 0.1$; the maximum error of the first 32 coefficients became $4\cdot 10^{-4}$ and $9\cdot 10^{13}$(!), respectively. In the latter case the errors of the first 8 coefficients did not exceed 10^{-10}, but the rounding error of a_n, due to cancellations, increase rapidly with n.

If ρ is not much larger than r^*, the procedure described above may lead to a value of N that is much larger than $\hat{n}$. In order to avoid this, we now set $\hat{n} = \alpha N$. We assume that $r < r' < \rho \le 1$, $n = 0 : \hat{n}$. Then, with all other parameters fixed, the bound in (3.1.25) is maximal for $n = \hat{n}$. We simplify this bound; $M(r)$ is replaced by the larger quantity $M(r')$, and the denominator is replaced by $(r'/r)^N$.

Then, for given r', α, N, we set $x = (r/r')^N$ and determine x so that

$$M(r')(r')^{-\alpha N}(Ux^{-\alpha} + x)$$

is minimized. The minimum is obtained for $x = (\alpha U)^{1/(1+\alpha)}$, i.e., for $r = r' x^{1/N}$, and the minimum is equal to

$$M(r')(r')^{-n} U^{1/(1+\alpha)} c(\alpha), \quad \text{where} \quad c(\alpha) = (1+\alpha)\alpha^{-\alpha/(1+\alpha)}.$$

19

We see that the error bound contains the factor $U^{1/(1+\alpha)}$. This is, e.g., proportional to $2U^{1/2}$ for $\alpha = 1$, and to $1.65U^{4/5}$ for $\alpha = \frac{1}{4}$. The latter case is thus much more accurate, but, for the same $\hat{n}$, one has to choose N four times as large, which gives more than four times as many arithmetic operations. In practice, $\hat{n}$ is usually given, and the order of magnitude of U can be estimated. Then α is to be chosen to make a compromise between the requirements for good accuracy and for small volume of computation. if ρ is not much larger than r^*, we may choose

$$N = \hat{n}/\alpha, \quad x = (\alpha U)^{1/(1+\alpha)}, \quad r = r' x^{1/N}.$$

Experiments were made with $f(z) = \ln(1-z)$. Then $\rho = 1$, $M(1) = \infty$. Take $\hat{n} = 64$, $U = 10^{-15}$, $r' = 0.999$. Then $M(r') = 6.9$. For $\alpha = 1, 1/2, 1/4$, we have $N = 64$, 128, 256, respectively. The above theory suggests $r = 0.764$, 0.832, 0.894, respectively. The theoretical estimates of the absolute errors become, 10^{-9}, $2.4\,10^{-12}$, $2.7\,10^{-14}$, respectively. The smallest errors obtained in experiments with these three values of α are, $6\,10^{-10}$, $1.8\,10^{-12}$, $1.8\,10^{-14}$, which were obtained for $r = 0.766$, 0.838, 0.898, respectively. So, the theoretical predictions of these experimental results are very satisfactory.

[19]This is a strict upper bound of the error for this value of r, in spite of the simplifications in the formulation of the minimization.

3.1.5 A brief introduction to Laurent, Fourier and Chebyshev Series.

A **Laurent series** is a series of the form

$$\sum_{n=-\infty}^{\infty} c_n z^n. \tag{3.1.26}$$

Its convergence region is the intersection of the convergence regions of the expansions $\sum_{n=0}^{\infty} c_n z^n$ and $\sum_{m=1}^{\infty} c_{-m} z^{-m}$, the interior of which are determined by conditions of the form $|z| < r_2$ and $|z| > r_1$. The convergence region can be void, e.g., if $r_2 < r_1$.

If $0 < r_1 < r_2 < \infty$ the convergence region is an *annulus*, $r_1 < |z| < r_2$. The series defines an analytic function in the annulus. Conversely, if $f(z)$ is a **single-valued analytic function** in this annulus, it is there represented by a Laurent series, that converges uniformly in every closed subdomain. The coefficients are determined by Cauchy's formula,

$$c_n = \frac{1}{2\pi i}\int_{|z|=r} z^{-n-1} f(z)dz, \quad r_1 < r < r_2, \quad -\infty < n < \infty. \tag{3.1.27}$$

The extension to the case when $r_2 = \infty$ is obvious; the extension to $r_1 = 0$ depends on whether there are any terms with negative exponents or not. [20]

Example 3.1.14.

A function may have several Laurent expansions (with different regions of convergence), e.g.,

$$(z-a)^{-1} = \begin{cases} -\sum_{n=0}^{\infty} a^{-n-1} z^n & \text{if } |z| < |a| \\ \sum_{m=1}^{\infty} a^{m-1} z^{-m} & \text{if } |z| > |a|. \end{cases}$$

The function $(z-1)^{-1} + (z-2)^{-1}$ has three Laurent expansions, with validity conditions $|z| < 1$, $1 < |z| < 2$, $2 < |z|$, respectively. The series contains both positive and negative powers of z in the middle case only. The details are left for Problem 9c.

Remark. The restriction to *single-valued* analytic functions is important in this subsection. In this book we cannot entirely avoid to work with **multi-valued** functions such as $\sqrt{z}$, $\ln z, z^\alpha$, (α non-integer). We always work with such a function, however, in some region where one branch of it, determined by some convention, is single-valued. In the examples mentioned, the natural conventions are to require the function to be positive when $z > 1$, and to forbid z to cross the negative real axis. In other words, the complex plane has a **cut** along the negative

[20] In the extension of *formal* power series to *formal Laurent series*, however, only a finite number of terms with negative indices are allowed to be different from zero, see Henrici loc.cit. Sec. 1.8. If you substitute z for z^{-1} an infinite number of negative indices is OK, if the number of positive indices is finite.

real axis. The annulus mentioned above is in these cases incomplete; its intersection with the negative real axis is missing, and we cannot use a Laurent expansion.

For a function like $\ln \frac{z+1}{z-1}$, we can, depending on the context, cut out either the interval $[-1, 1]$ or the complement of this interval with respect to the real axis. We then use an expansion into negative or into positive powers of z, respectively.

See Problem 17 for further comments on multi-valued functions in theory and computation.

If $r_1 < 1 < r_2$, we set $F(t) = f(e^{it})$. Note that $F(t)$ is a periodic function; $F(t + 2\pi) = F(t)$. By (3.1.26) and (3.1.27), the Laurent series then becomes for $z = e^{it}$ a **Fourier series**:

$$F(t) = \sum_{n=-\infty}^{\infty} c_n e^{int}, \quad c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-int} F(t) dt. \tag{3.1.28}$$

Note that $c_{-m} = O(r_1^m)$ for $m \to +\infty$, and $c_n = O(r_2^{-n})$ for $n \to +\infty$. *The formulas in* (3.1.28), *however, are valid in much more general situations*, where $c_n \to 0$ much more slowly, and where $F(t)$ cannot be continued to an analytic function $f(z)$, $z = re^{it}$, in an annulus. (In such a case $r_1 = 1 = r_2$, typically.)

A Fourier series is often written in the following form,

$$F(t) = \tfrac{1}{2} a_0 + \sum_{k=1}^{\infty} a_k \cos kt + b_k \sin kt. \tag{3.1.29}$$

Consider $c_k e^{ikt} + c_{-k} e^{-ikt} \equiv a_k \cos kt + b_k \sin kt$. Since $e^{\pm ikt} = \cos kt \pm i \sin kt$, we obtain for $k \geq 0$:

$$a_k = c_k + c_{-k} = \frac{1}{\pi} \int_{-\pi}^{\pi} F(t) \cos kt dt; \quad b_k = i(c_k - c_{-k}) = \frac{1}{\pi} \int_{-\pi}^{\pi} F(t) \sin kt dt. \tag{3.1.30}$$

Also note that $a_k - ib_k = 2c_k$. If $F(t)$ is real for $t \in \mathbf{R}$ then $c_{-k} = \bar{c}_k$. We mention without proof the important **Riemann–Lebesgue theorem**, by which the Fourier coefficients c_n *tend to zero as* $n \to \infty$ *for any function that is integrable* (in the sense of Lebesgue), a fortiori for any periodic function that is continuous everywhere. A finite number of finite jumps in each period are also allowed. [21] Another classical result in the theory of Fourier series reads: *If* $F(t)$ *is of bounded variation in the closed interval* $[-\pi, \pi]$ *then* $c_n = O(n^{-1})$; see Titchmarsh [34, §13.21,§13.73].

This can be generalized. *Suppose that* $F^{(p)}$ *is of bounded variation on* $[-\pi, \pi]$, *and that* $F^{(j)}$ *is continuous everywhere for* $j < p$. *Denote the Fourier coefficients of* $F^{(p)}(t)$ *by* $c_n^{(p)}$. *Then*

$$c_n = (in)^{-p} c_n^{(p)} = O(n^{-p-1}). \tag{3.1.31}$$

[21] $F(t)$ is of bounded variation in an interval if, in this interval, it can expressed in the form $F(t) = F_1(t) - F_2(t)$ where F_1 and F_2 are non-decreasing bounded functions. A finite number of jump discontinuities are allowed. For a differentiable function on $[a, b]$ the variation of F is defined as $\int_a^b |F'(t)| dt$.

This follows from the above classical result, after the integration of the formula for c_n in (3.1.27) by parts p times. Bounds for the truncation error of a Fourier series can also be obtained from this. The details are left for Problem 9e, together with a further generalization. A similar result is that $c_n = o(n^{-p})$ if $F^{(p)}$ is integrable, hence a fortiori if $F \in C^p$.

In particular, we find for $p = 1$ (since $\sum n^{-2}$ is convergent) that the Fourier series (3.1.27) *converges absolutely and uniformly* in $\mathbf{R}$. It can also be shown that *the Fourier series is valid*, i.e., the sum is equal to $F(t)$.

The **Chebyshev polynomials of the first and the second kind** are

$$T_n(z) = \cos n\phi, \quad U_{n-1}(z) = \frac{\sin n\phi}{\sin \phi}, \quad \text{where} \quad z = \cos\phi, \tag{3.1.32}$$

respectively. When we write just Chebyshev polynomial we refer to the first kind. Note that $T_0(z) = 1$, $T_1(z) = z$. That $T_n(z)$ is a n'th degree polynomial follows, by induction, from the important *recurrence relation*,

$$T_{n+1}(z) = 2zT_n(z) - T_{n-1}(z), \quad (n \ge 1), \tag{3.1.33}$$

which is derived from the well known trigonometric formula $\cos(n+1)\phi + \cos(n-1)\phi = 2\cos\phi\cos n\phi$. We obtain,

$$T_2(z) = -1 + 2z^2; \quad T_3(z) = -3z + 4z^3; \quad T_4(z) = 1 - 8z^2 + 8z^4.$$

Note that $|T_n(x)| \le 1$ for $x \in [-1, 1]$, in spite that its leading coefficient is as large as 2^{n-1}. This is a unique property of the Chebyshev polynomials and is one of the reasons why they are very important for Numerical Mathematics; see, e.g., Problem 19. $U_{n-1}(z)$ satisfies the same recurrence relation, with the initial conditions

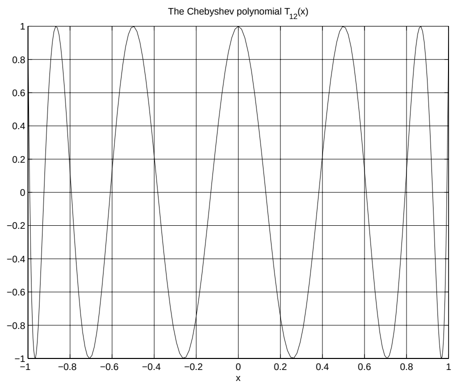


Figure 3.1.5. *The Chebyshev polynomial* $T_{12}(x), x \in [-1, 1]$.

$U_{-1}(z) = 0$, $U_0(z) = 1$; its degree is $n - 1$.

Set $e^{i\phi} = w$; ϕ and z may be complex. Then

$$z = \tfrac{1}{2}(w + w^{-1}), \quad T_n(z) = \tfrac{1}{2}(w^n + w^{-n}), \tag{3.1.34}$$

$$w = z \pm \sqrt{z^2-1}, \quad (z+\sqrt{z^2-1})^n = T_n(z) + U_{n-1}(z)\sqrt{z^2-1}.$$

It follows that a **Chebyshev expansion**,

$$f(z) = \sum_{j=0}^{\infty} c_j T_j(z), \tag{3.1.35}$$

formally corresponds to a Laurent expansion,

$$g(w) = f\left(\frac{w+w^{-1}}{2}\right) = \sum_{-\infty}^{\infty} a_j w^j; \quad a_{-j} = a_j = \begin{cases} \frac{1}{2}c_{|j|}, & \text{if } j \neq 0; \\ c_0, & \text{if } j = 0. \end{cases}$$

It can be shown, e.g., by the parallelogram law, that $|z+1| + |z-1| = |w| + |w|^{-1}$, (Problem 9f). Hence, if $R > 1$, $z = \frac{1}{2}(w + w^{-1})$ maps the annulus $\{w : R^{-1} < |w| < R\}$, twice onto an ellipse $\mathcal{E}_R$, determined by the relation,

$$\mathcal{E}_R = \{z : |z-1| + |z+1| \leq R + R^{-1}\}, \tag{3.1.36}$$

with foci at 1 and -1. The axes are, respectively, $R + R^{-1}$ and $R - R^{-1}$, and hence R means the sum of the semi-axes.

Note that, as $R \to 1$, the ellipse degenerates into the interval $[-1, 1]$. As $R \to \infty$, it becomes close to the circle $|z| < \frac{1}{2}R$. It follows from (3.1.34) etc. that this family of confocal ellipses are level curves of $|w| = |z \pm \sqrt{z^2-1}|$. In fact, we can also write,

$$\mathcal{E}_R = \{z : 1 \leq |z + \sqrt{z^2-1}| \leq R\}. \tag{3.1.37}$$

Theorem 3.1.11.

Let $f(z)$ be real-valued for $z \in [-1, 1]$, analytic and single-valued for $z \in \mathcal{E}_R$, $R > 1$. Assume that $|f(z)| \leq M$ for $z \in \mathcal{E}_R$. Then

$$|f(x) - \sum_{j=0}^{n-1} c_j T_j(x)| \leq \frac{2MR^{-n}}{(1 - R^{-1})} \quad \textit{for } x \in [-1, 1].$$

Proof. Set as before, $z = \frac{1}{2}(w + w^{-1})$, $g(w) = f(\frac{1}{2}(w + w^{-1}))$. Then $g(w)$ is analytic in the annulus $R^{-1} + \epsilon \leq |w| \leq R - \epsilon$, and hence the Laurent expansion (1.2) converges there. In particular it converges for $|w| = 1$, hence the Chebyshev expansion for $f(x)$ converges when $x \in [-1, 1]$.

Set $r = R - \epsilon$. By Cauchy's formula, we obtain, for $j > 0$,

$$|c_j| = 2|a_j| = \left|\frac{2}{2\pi i}\int_{|w|=r} g(w)w^{-(j+1)}dw\right| \leq \frac{2}{2\pi}\int_0^{2\pi} Mr^{-j-1}r\,d\phi = 2Mr^{-j}.$$

We then obtain, for $x \in [-1, 1]$,

$$\left|f(x) - \sum_{j=0}^{n-1} c_j T_j(x)\right| = \left|\sum_{n}^{\infty} c_j T_j(x)\right| \leq \sum_{n}^{\infty} |c_j| \leq 2M \sum_{n}^{\infty} r^{-j} \leq 2Mr^{-n}/(1 - 1/r).$$

This holds for any $\epsilon > 0$. We can here let $\epsilon \to 0$ and thus replace r by R. □

If a Chebyshev expansion converges rapidly, the truncation error is, by and large, determined by the first few neglected terms. As indicated by Fig. 3.1.5 and Fig. 3.1.8 the error curve is oscillating with slowly varying amplitude in [-1,1]. In contrast, the truncation error of a power series is proportional to a power of x.

Note that $f(z)$ is allowed to have a singularity arbitrarily close to the interval $[-1, 1]$, and the convergence of the Chebyshev expansion will still be exponential, although the exponential rate deteriorates, as $R \downarrow 1$.

Important properties of trigonometric functions and Fourier series can be reformulated in the terminology of Chebyshev polynomials. For example, they satisfy certain **orthogonality relations**, see Problem 20. Also results like (3.1.31) concerning how the rate of decrease of the coefficients or the truncation error of a Fourier series, is related to the smoothness properties of its sum, can be translated to Chebyshev expansions. So, even *if F is not analytic, a Chebyshev expansion converges under amazingly general conditions* (unlike a power series), but the convergence is much slower than exponential. A typical result reads: *if $f \in C^k[-1, 1]$, $k > 0$, there exists a bound for the truncation error that decreases uniformly like $O(n^{-k} \log n)$.* Sometimes convergence acceleration can be successfully applied to such series, see Sec. 3.3. More about Chebyshev polynomials and related questions in Sections 4.4 and 4.5, and in Ch. 12.

The numerical value of a truncated Chebyshev expansion can be computed by means of **Clenshaw's algorithm** given in Problem 21, which holds for any sum of the form $S = \sum_{k=1}^{n-1} c_k\phi_k$, where $\{\phi_k\}$ satisfies a **three term recurrence relation** [22]

$$\phi_{k+1} = \alpha_k\phi_k - \gamma_k\phi_{k-1}, \quad k = 1 : n - 2.$$

Clenshaw's algorithm can therefore also be applied to series of Legendre functions, Bessel functions, Coulomb wave functions etc., because they satisfy recurrence relations of this type, where the α_k, γ_k depend on x; see Abramowitz and Stegun or any text on *special functions*. Other applications are the case when the ϕ_k are the denominators or numerators of the convergents of a *continued fraction*, see Sec. 3.4, and, above all, the case when $\phi_k(x)$ is a family of *orthogonal polynomials*, see Ch. 12. In that case $\alpha_k = \alpha'_k \cdot (x - \beta_k)$, where α'_k, β_k, γ_k are independent of x.

3.1.6 Perturbation Expansions

In the equations of applied mathematics it is often possible to identify a small dimensionless parameter (say) ϵ. The case when $\epsilon = 0$ is called the *reduced problem* or the unperturbed case, and one asks for a **perturbation expansion**, i.e. an expansion of the solution of the perturbed problem into powers of the perturbation parameter ϵ. In many cases it can be proved that the expansion has the form $c_0+c_1\epsilon+c_2\epsilon^2+\ldots$, but there are also important cases, where the expansion contains fractional or some negative powers.

[22] This is sometimes given for $k = 0 : n-2$, with the initial conditions $\phi_0 = 1$, $\phi_{-1} = 0$. Warning: Then, for the Chebyshev polynomials, $\alpha_k = 2x$ for $k > 0$, but $\alpha_0 = x$.

We first consider an analytic equation $\phi(z,\epsilon) = 0$ and seek the roots $z_i(\epsilon)$. It is practical to move the origin to $z_i(0)$. (See Problem 1.4.1 or 1.4.2.) If $z_i(0)$ is a simple root, i.e., if $\partial\phi/\partial z \neq 0$, $(z,\epsilon) = (z_i(0), 0)$ then a theorem of complex analysis tells us that $z_i(\epsilon)$ is an analytic function in a neighborhood of the origin, hence the expansion

$$z_i(\epsilon) = c_0 + c_1\epsilon + c_2\epsilon^2 + \dots$$

exists. We call this a **regular perturbation problem**.

If $z_i(0)$ is a k-fold root then (under certain conditions) there exists an expansion of the form $z_i(\epsilon) = c_0 + c_1\epsilon^{1/k} + c_2\epsilon^{2/k} + \dots$ for each of the k k'th roots of ϵ. *If one tries to determine the coefficients in an expansion of the wrong form, one usually runs into contradictions.*

A simple perturbation example for a *differential equation* is given in Problem 12. More interesting examples are presented in Sec. 13.8.

Example 3.1.15.

We shall expand the roots of $\phi(z,\epsilon)) \equiv \epsilon z^2 - z + 1 = 0$ into powers of ϵ. The reduced problem $-z + 1 = 0$ has only one finite root $z_1(0) = 1$. Set $z = 1 + x\epsilon$, $x = c_1 + c_2\epsilon + c_3\epsilon^2 + \dots$. Then $\phi(1 + x\epsilon,\, \epsilon)/\epsilon = (1 + x\epsilon)^2 - x = 0$, i.e.,

$$(1 + c_1\epsilon + c_2\epsilon^2 + \dots)^2 - (c_1 + c_2\epsilon + c_3\epsilon^2 + \dots) = 0.$$

Matching the coefficients of ϵ^0, ϵ^1, ϵ^2, we obtain the system

$$\begin{aligned} 1 - c_1 = 0 &\;\Rightarrow\; c_1 = 1; \\ 2c_1 - c_2 = 0 &\;\Rightarrow\; c_2 = 2; \\ 2c_2 + c_1^2 - c_3 = 0 &\;\Rightarrow\; c_3 = 5; \end{aligned}$$

hence $z_1(\epsilon) = 1 + \epsilon + 2\epsilon^2 + 5\epsilon^3 + \dots$.

Now, the easiest way to obtain the expansion for the second root $z_2(\epsilon)$, is to use the fact that the sum of the roots of the quadratic equation equals ϵ^{-1}, hence $z_2(\epsilon) = \epsilon^{-1} - 1 - \epsilon - 2\epsilon^2 + \dots$.

Note the appearance of the term ϵ^{-1}. This is due to a characteristic feature of this example. The degree of the polynomial is lower for the reduced problem than it is for $\epsilon \neq 0$; one of the roots escapes to ∞ as $\epsilon \to 0$. This is an example of a **singular perturbation** problem; an important type of problem for differential equations, see Sec. 13.8.

Although we have already determined $z_2(\epsilon)$, we shall use this problem to illustrate a general **balancing procedure**, recommended in Lin–Segel [25, Sec. 9.1], where it is applied to singular perturbation problems for differential equations too. The basic idea is very simple: each term in an equation behaves like some power of ϵ. *The equation cannot hold, unless there is a β, such that a pair of terms of the equation behave like ϵ^β, and the ϵ-exponents of the other terms are larger than or equal to β.* (Recall that larger exponents make smaller terms.)

In the example above, suppose that $z_2(\epsilon) \sim A\epsilon^\alpha$, $(\alpha < 0)$. The three terms have the exponents

$$1 + 2\alpha, \quad \alpha, \; 0.$$

Try the first two terms as the candidates for being the dominant pair. Then $1+2\alpha = \alpha$, hence $\alpha = -1$. The three exponents become -1, -1, 0. Since the third exponent is larger than the exponent of the candidates, this choice of pair seems possible, but we have not shown that it is the only possible choice.

Now try the first and the third terms as candidates. Then $1 + 2\alpha = 0$, hence $\alpha = -\frac{1}{2}$. The exponent of the non-candidate is $-\frac{1}{2} \leq 0$; this candidate pair is thus impossible.

Finally try the second and the third terms. Then $\alpha = 0$, but we are only interested in negative values of α. (This pair gives the balance for the finite root z_1.)

The conclusion is that we can try coefficient matching in the expansion $z_2(\epsilon) = c_{-1}\epsilon^{-1} + c_0 + c_1\epsilon + \dots$. We don't need to do it, since we know the answer already, but it indicates how to proceed in more complicated cases.

Example 3.1.16.

First consider the equation $z^3 - z^2 + \epsilon = 0$. The reduced problem $z^3 - z^2 = 0$ has a single root, $z_1 = 1$, and a double root, $z_{2,3} = 0$. By a similar coefficient matching as in the previous example we find that $z_1(\epsilon) = 1 - \epsilon - 2\epsilon^2 + \dots$. For the double root, try $z(\epsilon) = c_0\epsilon^{1/2} + c_1\epsilon + c_2\epsilon^{3/2} + \dots$. By matching the coefficients of ϵ, $\epsilon^{3/2}$, ϵ^2, we obtain the system

$$\begin{aligned} -c_0^2 + 1 = 0 \ &\Rightarrow c_0 = \pm 1 \\ -2c_0c_1 + c_0^3 = 0 \ &\Rightarrow c_1 = \frac{1}{2} \\ -2c_0c_2 - c_1^2 + 2c_0^2c_1 + c_1c_0^2 = 0 \ &\Rightarrow c_2 = \pm\frac{5}{8}, \end{aligned}$$

hence $z_{2,3}(\epsilon) = \pm\epsilon^{1/2} + \frac{1}{2}\epsilon \pm \frac{5}{8}\epsilon^{3/2} + \dots$. Since $z_1 + z_2 + z_3 = 1$, we conclude from the expansion of $z_1(\epsilon)$ that the next term reads ϵ^2.

There are, however, exceptions, where we do not have a pair of roots which behave like $\pm c_0\epsilon^{1/2}$ as $\epsilon \to 0$. In such a case the balancing procedure described above may help; this time with a positive α.

Take the equation $\phi(z,\epsilon) = (1+\epsilon)z^2 + 4\epsilon z + \epsilon^2 = 0$. The reduced problem reads $z^2 = 0$, with a double root. Try $z \sim A\epsilon^\alpha$, $\alpha > 0$. The exponents of the three terms become 2α, $\alpha + 1$, 2. We see that $\alpha = 1$ makes the three exponents all equal to 2. (This is OK.) So, set $z = \epsilon y$. The equation reads, after division by ϵ^2, $(1+\epsilon)y^2 + 4y + 1 = 0$, hence $y(0) = a \equiv -2 \pm \sqrt{3}$. Coefficient matching yields the result

$$z = \epsilon y = a\epsilon + (-a + a^2/2)\epsilon^2 + \dots, \quad a = -2 \pm \sqrt{3}.$$

The intention of the two previous examples is to give hints, how to proceed in the case of an equation or a system of a more complicated type, where a small parameter ϵ can be identified. When more terms are needed in such expansions, automatic formula manipulation may be useful.

If ϵ is small enough, the last term included can serve as an error estimate. A more reliable error estimate (or even an error bound) can be obtained by inserting the truncated expansion into the equation. It shows that the truncated expansion satisfies exactly a perturbed equation. A general error estimate for such a situation is given in (6.5.3). The same idea was indicated for a differential equation in Example 3.1.2, and can be applied to equations of many other types.

One may, however, ask, whether one really needs the power series or just any polynomial approximation that represents $z(\epsilon)$ with a certain accuracy for (say) $0 < \epsilon \le \epsilon'$. It depends on the purpose. In the latter case, there are purely numerical alternatives to a series; one may solve the equation (or system) $\phi(z,\epsilon) = 0$ numerically for (say) k different values of ϵ and then *interpolate* an (approximating) *algebraic expression of the right form* to the points $\bigl(\epsilon_i, z(\epsilon_i)\bigr)$, $i = 1 : k$, see e.g., the introduction of Ch.4.

Such a calculation has to be tested carefully, because of the risk for catastrophic cancellations, for example, if the points ϵ_i have been chosen too densely, or if the values of $z(\epsilon_i)$ have too large irregular errors. *Least squares polynomial fitting*, see Sec. 4.1 and Sec. 9.3, is an alternative to interpolation.

3.1.7 Ill-Conditioned Series.

Slow convergence is not the only numerical difficulty which occurs in connection with infinite series. There are also series where the size of certain terms are many orders of magnitude larger than the sum of the series. Small relative errors in the computation of the large terms can then produce large errors in the result. We call such a series ill-conditioned. The following example—in itself of no practical interest—provides some insight.

Example 3.1.17.

The Maclaurin series

$$e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!},$$

converges for all x. The ratio of consecutive terms is

$$\frac{a_{n+1}}{a_n} = -\frac{x^{n+1}}{x^n}\frac{n!}{(n+1)!} = -\frac{x}{n+1}.$$

Thus, the magnitude of the terms grows, as long as $n+1 < x$. For $x \approx 20$, the largest term is about $20^{20}/(20)! \approx 4{\cdot}10^7$, but the sum is only $e^{-20} \approx 2{\cdot}10^{-9}$. Hence a relative error of 10^{-16} in the largest term can lead to an error of more than 100% in the result. The convergence of the series is rather slow to begin with, but no worse than what one is forced to accept in other situations. The remainder term after 100 terms is $20^{100}/(100!) \approx 7{\cdot}10^{-20}$. So one would have about ten significant digits in the result, if there were no rounding errors, but an actual floating point computation with $u \approx 10^{-16}$ gave almost 100% error. Hence this series can be considered as quite difficult from a numerical point of view, in spite of the fact that it is convergent.

It goes without saying that there are many other simple procedures by which e^{-20} can be computed to full accuracy.

A less trivial case is the Bessel function, $J_0(x) = \sum_{n=0}^{\infty}(-1)^n \frac{(x/2)^{2n}}{(n!)^2}$. It converges for all x. The terms for $x = 50$ are displayed in Fig. 3.1.7 that shows that this case is very ill-conditioned.

We shall in Ch. 12 and in Sec. 3.3 see methods that transform a rather big class of ill-conditioned series either into an integral (methods based on formulas of Plana and Lindelöf) or into a "bell sum", i.e., a special kind of series of positive terms, where shortcuts can be made in the summation. These methods have been successful on some examples, where it was hard to find an alternative to the power series quickly.

For the computation of $J_0(50)$, however, a method based on Plana's formula needed an amount of work equivalent to the evaluation of about 2500 terms to obtain an error less than $5\,10^{-6}$. The Bessel functions are of great importance in Mathematical Physics and belong to the most intensively studied mathematical functions. One has developed several well-conditioned procedures for computing $J_0(x)$ for large values of x, where the power series is avoided. Some of these procedures would handle this problem by only a few per cent of the effort mentioned.

At the present state of the art of handling ill-conditioned series, it may often be worthwhile to try to avoid the series, by the use of special information related to the given problem. (See Problems 10 and 15.)

Figure 3.1.6. *The terms of the ill-conditioned power series for the Bessel function $J_0(z)$, $z = 50$. The largest term is $\approx 10^{19}$, while the sum is ≈ 0.056. Direct summation with $u = 10^-16$ gave the useless result 655.29.*

3.1.8 Numerical Use of Divergent or Semiconvergent Series

In the previous sections, we saw that the fact that a series is convergent is no guarantee that it is numerically useful. In this section, we shall see examples of the reverse situation: a divergent series can be of use in numerical computations. This sounds strange, but it refers to series where the size of the terms decreases rapidly at first and increases later (see Fig. 3.1.8), where one can compare the magnitude of the remainder with the absolute and value of the first neglected term. Such series are sometimes called **semiconvergent**.

Figure 3.1.7. *The terms of a semiconvergent series. (The ratio of the consecutive terms is usually much bigger.)*

Example 3.1.18. Give a semiconvergent series for the computation of

$$f(x) = e^x \int_x^\infty e^{-t} t^{-1} dt = \int_0^\infty e^{-u}(u+x)^{-1} du$$

for large values of x. (The second integral was obtained from the first by the substitution $t = u + x$.) The expression $(u+x)^{-1}$ should first be expanded in a geometric series with remainder term, valid even for $u > x$,

$$(u+x)^{-1} = x^{-1}(1+x^{-1}u)^{-1} = x^{-1}\sum_{j=0}^{n-1}(-1)^j x^{-j} u^j + (-1)^n (u+x)^{-1}(x^{-1}u)^n$$

We can write $f(x) = S_n(x) + R_n(x)$, where

$$\begin{aligned} S_n(x) &= x^{-1}\sum_{j=0}^{n-1}(-1)^j x^{-j}\int_0^\infty u^j e^{-u} du \\ &= \frac{1}{x} - \frac{1!}{x^2} + \frac{2!}{x^3} - \ldots + (-1)^{n-1}\frac{(n-1)!}{x^n}, \\ R_n(x) &= (-1)^n \int_0^\infty (u+x)^{-1}\Big(\frac{u}{x}\Big)^n e^{-u} du. \end{aligned}$$

The terms in $S_n(x)$ qualitatively behave as in Fig. 3.1.7. The ratio between the last term in S_{n+1} and the last term in S_n is

$$-\frac{n!}{x^{n+1}}\frac{x^n}{(n-1)!} = -\frac{n}{x}, \tag{3.1.38}$$

and since the absolute value of that ratio for fixed x is unbounded as $n \to \infty$, the sequence $\{S_n(x)\}_{n=1}^\infty$ *diverges for every positive* x. But since sign $R_n(x) = (-1)^n$ for $x > 0$, it follows from Theorem 3.1.4 that

$$f(x) = \frac{1}{2}\Big(S_n(x) + S_{n+1}(x)\Big) \pm \frac{1}{2}\frac{n!}{x^{n+1}}.$$

The idea is now to choose n so that the estimate of the remainder is as small as possible. According to (3.1.38), this happens when n is equal to the integer part of x. For $x = 5$ we choose $n = 5$,

$$S_5(5) = 0.2 - 0.04 + 0.016 - 0.0096 + 0.00768 = 0.17408,$$
$$S_6(5) = S_5(5) - 0.00768 = 0.16640,$$

which gives $f(5) = 0.17024 \pm 0.00384$. The correct value is 0.17042, so the actual error is only 5% of the error bound.

For larger values of x the accuracy attainable increases. One can show that the bound for the relative error using the above computational scheme decreases approximately as $(\pi \cdot x/2)^{1/2} e^{-x}$. Thus the above divergent series gives extremely good accuracy for large values of x, if one stops at the smallest term. It can even be improved further, by the use of *repeated averages*, see Sec. 3.3.2. The algorithms for the transformation of a power series into a rational function described in Sec. 3.4 can also sometimes be applied to divergent expansions (continued fractions, Padé approximants).

One can derive the same series expansion as above by repeated integration by parts. This is often a good way to derive numerically useful expansions, convergent or semi-convergent, with a remainder in the form of an integral. For convenient reference, we formulate this as a lemma that is easily proved by induction and the mean value theorem of integral calculus. See Problem 14 for applications.

Lemma 3.1.12. *(Repeated Integration by Parts)*
Let $F \in C^p(a,b)$, let G_0 be a piecewise continuous function, and let $G_0, G_1, \ldots$ be a sequence of functions such that $G'_{j+1}(x) = G_j(x)$ with suitably chosen constants of integration. Then

$$\int_a^b F(t)G_0(t)dt = \sum_{j=0}^{p-1} (-1)^j F^{(j)}(t) G_{j+1}(t)\Big|_{t=a}^{b} + (-1)^p \int_a^b F^{(p)}(t)G_p(t)dt.$$

The sum is the "expansion", and the last integral is the "remainder". If $G_p(t)$ has a constant sign in (a,b), the remainder term can also be written in the form

$$(-1)^p F^{(p)}(\xi)(G_{p+1}(b) - G_{p+1}(a), \quad \xi \in (a,b).$$

This expansion in Lemma 3.1.12 is valid as an *infinite* series, if and only if the remainder tends to 0 as $p \to \infty$. Even if the sum converges as $p \to \infty$, it may converge to the wrong result. (See, e.g., Sec. 3.1.3 and 3.3.3.)

The series in Example 3.1.18 is an expansion in negative powers of x, with the property that for all n, the remainder, when $x \to \infty$, approaches zero faster than the last included term. Such an expansion is said to **represent** $f(x)$ **asymptotically** as $x \to \infty$. Such an **asymptotic series** can be either convergent or divergent (semi-convergent).

It is important to appreciate that *an asymptotic series does not define a sum uniquely.* For example $f(x) = e^{-x}$ is asymptotically represented by the series $\sum 0x^{-j}$, as $x \to \infty$. So e^{-x}, (and many other functions), can therefore be added to the function, for which the expansion was originally obtained.

Asymptotic expansions are not necessarily expansions into negative powers of x. An expansion into *positive* powers of $x - a$, $f(x) \sim \sum_{\nu=0}^{n-1} c_\nu (x-a)^\nu + R_n(x)$, *represents* $f(x)$ *asymptotically when* $x \to a$ *if* $\lim_{x\to a}(x-a)^{-(n-1)}R_n(x) = 0$. Asymptotic expansions of the error of a numerical method into positive powers of a step length h are of great importance in the more advanced study of numerical methods. Such expansions form the basis of simple and effective acceleration methods for improving numerical results, see Sec. 3.3.5.

In many branches of applied mathematics, divergent asymptotic series are an important aid, though they are often needlessly surrounded by an air of mysticism.

Review Questions

1. Formulate three general theorems that can be used for estimating the remainder term in numerical series.

What can you say about the remainder term, if the n'th term is $O(n^{-k}$, $k > 1$? Suppose in addition that the series is alternating. What further condition should you add, in order to guarantee that the remainder term will be $O(n^{-k})$?

2. Give, with convergence conditions, the Maclaurin series for $\ln(1+x)$, e^x, $\sin x$, $\cos x$, $(1+x)^k$, $(1-x)^{-1}$, $\ln \frac{1+x}{1-x}$. For which of these functions does also another Laurent expansion exist?

3. Give the Cauchy formula for the coefficients of Taylor and Laurent series, and describe the Cauchy+FFT method. Give the formula for the coefficients of a Fourier series.

4. Give the generating functions of the Bernoulli and the Euler numbers. Describe generally how to derive the coefficients in a quotient of two Maclaurin series.

5. If a functional equation, e.g. $4(\cos x)^3 = \cos 3x + 3\cos x$, is known to be valid for real x, how do you know that it holds also for all complex x? Explain what is meant by the statement that it holds also for formal power series, and why is this true?

6. If the Maclaurin series is given for the function $f(z)$, that is analytic at the origin, describe an easy way to compute the Maclaurin series for $e^{f(z)}$? Is this procedure valid also for formal power series?

7. Describe by an example the balancing procedure that was mentioned in the subsection about perturbation expansions.

8. Define the Chebyshev polynomials, and tell some interesting properties of these and of Chebyshev expansions. For example, what do you know about the speed of convergence of a Chebyshev expansion for various classes of functions? (The detailed expressions are not needed.)

9. Describe and exemplify, what is meant by an ill-conditioned power series.

10. Define what is meant, when one says that the series $\sum_0^\infty a_n x^{-n}$
(a) converges to a function $f(x)$ for $x \geq R$;
(b) represents a function $f(x)$ asymptotically as $x \to \infty$.
Give an example of a series that represents a function asymptotically as $x \to \infty$, although it diverges for every finite positive x.
What is meant by semi-convergence? Say a few words about termination criteria and error estimation.

Problems and Computer Exercises

1. In how large a neighborhood of $x = 0$ does one get, respectively, four and six correct decimals using the following approximations?
(a) $\sin x \approx x$; (b) $(1+x^2)^{-1/2} \approx 1 - x^2/2$; (c) $(1+x^2)^{-1/2} e^{\sqrt{\cos x}} \approx e(1 - \frac{3}{4}x^2)$.
Comment: The truncation error is asymptotically qx^p where you know(?) p.
An alternative to an exact algebraic calculation of q, is a numerical estimation of q, by means of the actual error for a suitable value of x—neither too big nor too small(!). (Check the estimate of q for another value of x.)

2. (a) Let a, b, be the lengths of the two smaller sides of a right angles triangle, $b \ll a$. Show that the hypotenuse is approximately $a + b^2/(2a)$ and estimate the error of this approximation. If $a = 100$, how large is b allowed to be, in order that the absolute error should be less than 0.01?
(b) How large relative error do you commit, when you approximate the length of a small circular arc by the length of the chord? How big is the error i the arc is 100 km on the earth? (Approximate the earth by a ball of radius $40000/(2\pi)$ km.)
(c) $P(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4$ is a polynomial approximation to $\cos x$ for small values of $|x|$. Estimate the errors of $P(x)$, $P'(x)$, $\frac{1}{x}\int_0^x P(t)dt$, and compare them, e.g., for $x = 0.1$.
(d) How accurate is the formula $\arctan x \approx \frac{1}{2}\pi - 1/x$ for $x \gg 1$?

3. (a) Compute $10 - (999.999)^{1/3}$ to 9 significant digits, by the use of the binomial expansion. Compare with the result obtained by a computer, directly from the first expression.
(b) How many terms of the Maclaurin series for $\ln(1+x)$ would you need in order to compute $\ln 2$ with an error less than 10^{-6} ? How many terms do you need, if you use instead the the series for $\ln{(1+x)/(1-x)}$, with an appropriate choice of x?

4. Give an approximate expression of the form $ah^b f^{(c)}(0)$ for the error of the estimate of the integral $\int_{-h}^{h} f(x)dx$ obtained by Richardson extrapolation (according to Sec. 1.2.2) from the trapezoidal values $T(h), T(2h)$.

5. Adapt Example 3.1.8 in Sec. 3.1.3 (concerning $\sinh x$) to the precision of your computer. Replace the assumption that "e^u is computed with a relative error less than (say) $5u$by a more useful assumption.

6. Compute, by means of appropriate expansions, the following integrals to (say) five correct decimals.

$$\text{(a)} \int_0^{0.1} (1 - 0.1\sin t)^{1/2} dt; \qquad \text{(b)} \int_{10}^{\infty} (t^3 + t)^{-1/2} dt.$$

7. (a) Expand $\arcsin x$ into powers of x by the integration of the expansion of $(1 - x^2)^{-1/2}$.

(b) Prove the expansion

$$x = \sinh x - \frac{1}{2}\frac{\sinh^3 x}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{\sinh^5 x}{5} - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{\sinh^7 x}{7} + \dots$$

(See an application in Example 3.2.9.)

(c) Express the coefficients of the power series expansions of $y \cot y$ and $\ln \frac{\sin y}{y}$ in terms of the Bernoulli numbers. *Hint:* Set $x = 2iy$ into (3.1.12). Differentiate the second function.

(d) Show that

$$\ln\frac{z+1}{z-1} = 2\Big(\frac{1}{z} + \frac{1}{3z^3} + \frac{1}{5z^5} + \dots\Big), \quad |z| > 1.$$

Find a recurrence relation for the coefficients of the expansion

$$\Big(\ln\frac{z+1}{z-1}\Big)^{-1} = \frac{1}{2}z - \mu_1 z^{-1} - \mu_3 z^{-3} - \mu_5 z^{-5} - \dots, \quad |z| > 1.$$

Compute μ_1, μ_3, μ_5 and determine $\sum_0^\infty \mu_{2j+1}$ by letting $z \downarrow 1$. (Full rigor is not required.) *Hint:* Look at Example 3.1.6.

(e) Find a recurrence relation for the Euler numbers E_n, see Example 3.1.6, and use it for showing that these numbers are integers.

8. Let the power series expansion $f(x) = a_0 + a_1 x + a_2 x^2 + \dots$ be given. Find recurrence relations for the coefficients of the expansion for $g(x) \equiv \phi(f(x)) = b_0 + b_1 x + b_2 x^2 + \dots$ in the following cases:

(a) $\phi(y) = \ln y$, if $a_0 \neq 0$.
Hint: Use $f(x) = e^{g(x)}$, and rearrange the formulas given in Example 3.1.10.
Answer: $b_0 = \ln a_0, \quad b_n = \frac{1}{a_0}\big(a_n - \frac{1}{n}\sum_{j=1}^{n-1}(n-j)a_j b_{n-j}\big)$.

(b) $\phi(y) = y^\alpha$, if $a_0 \neq 0$, $\alpha \neq 1$.
Hint: Show that $fg' = \alpha g f'$, where $'$ means differentiation with respect to x. Then proceed analogously to Example 3.1.10. *Answer:* $b_0 = a_0^c, \quad b_n = \frac{1}{na_0}\sum_{j=0}^{n-1}\big(\alpha n - (\alpha+1)j\big)b_j a_{n-j}$.

(c) $\phi_1(y) = \cos y$, $\phi_2(y) = \sin y$, simultaneously.

Hint: Consider instead $\phi(y) = e^{iy}$, and separate real and imaginary parts afterwards.

(d)If you have the matrix representation, described in Example 3.1.11, for a function such that $f_N(0) = 0$, how do you obtain a square matrix representation for $f_N(z)/z$?

(e) Let $f(z) = -z^{-1}\ln(1-z)$. Find the first five coefficients of the Maclaurin series for the functions $f(z)^k$, $k = 2 : 5$, by means of the matrix representation. (The answer and an application to numerical differentiation are given in Example 3.2.6.)

(f) If your computer language has a convenient matrix handling , compute the first twelve Maclaurin coefficients of arcsinh$x = \ln(x + \sqrt{1+x^2})$, by means of the matrix representation. Compare with Problem 7(b).

(g) If you know the Maclaurin expansion $f(z) = \sum a_k z^k$ and want the Maclaurin expansions for $f(z)^k$, $k = 1 : K$, you can also ask for a **generating function** $F(t,z) = \sum_{j=0}^{\infty} f(z)^k t^k/k! = e^{f(z)t} = e^{a_0}e^{a_1 zt}e^{a_2 z^2 t}\cdots$. Write a program that transforms the product (with suitable truncations) into an expansion into powers of t, the coefficients of which are the requested expansions into powers of z. Test this on Problem (d), i.e., for $f(z) = -\ln(1-z)$.

In the case of Problem (d), you also have a chance to find an analytic expression for $F(t,z)$, which easily leads to an approximating polynomial in two variables t, z, where the terms have to be rearranged in order to yield a truncated power series in t, where the K coefficients are the requested truncated power series in z. For a change, you may like to use a computer algebraic system like Maple or Mathematica this time. For the different approaches mentioned above, and also for the Cauchy+FFT method, compare both the results and the efforts made, your own and your computer's.

9. (a) Apply (on a computer) the Cauchy+FFT method to find the Maclaurin coefficients a_n of (say) e^z and $(1+z)^{1/2}$. Make experiments with different values of r and N, and compare with the exact coefficients. This presupposes that you have access to good programs for FFT.

Try to summarize your experiences of how the error of a_n depends on r, N. You may find some guidance in 3.1.5.

(b) Reconstruct, by means of the Cauchy+FFT method, the table for the Maclaurin coefficients of $1/\Gamma(z)$ given in Abramowitz and Stegun, Sec. 6.1.34, and improve the relative accuracy of the small coefficients. Check experimentally the accuracy you obtain by means of this series for different values of $|z|$. *Hint:* Some ideas of Example 3.1.5 may be useful.

This presupposes that you have access to good programs for FFT and for the Gamma function with a complex argument. Alternatively, you may write a program yourself for $\ln\Gamma(z+1)$ according to the indications given in Example 3.3.16.

(c) Find the Laurent expansions for $f(z) = \frac{1}{z-1} + \frac{1}{z-2}$.

(d) How do you use the Cauchy+FFT method for finding Laurent expansions? Test your ideas on the function in the previous subproblem (and on a few other functions). There may be some pitfalls with the interpretation of the output from the FFT program, related to so-called **aliasing**, see Ch. 4 and Strang [33].

(e) As in Sec. 3.1.5, suppose that $F^{(p)}$ is of bounded variation in $[-\pi, \pi]$ and denote the Fourier coefficients of $F^{(p)}$ by $c_n^{(p)}$. Derive the following generalization of (3.1.31):

$$c_n = \frac{(-1)^{n-1}}{\pi} \sum_{j=0}^{p-1} \frac{F^{(j)}(\pi) - F^{(j)}(-\pi)}{(in)^{j+1}} + \frac{c_n^{(p)}}{(in)^p},$$

and show that if we add the condition that $F \in C^j[-\infty, \infty]$, $j < p$, then the asymptotic results given in (and after) (3.1.31) hold.

(f) Let $z = \frac{1}{2}(w + w^{-1})$. Show that $|z-1| + |z+1| = |w| + |w|^{-1}$.
Hint: Use the parallelogram law, $|p-q|^2 + |p+q|^2 = 2(|p|^2 + |q|^2)$.

(g) Theorem 3.1.11 is concerned with the convergence of a Chebyshev expansion when $x \in [-1, 1]$. Make a small change in the proof, in order to find a statement about the speed of convergence of a Chebyshev expansion for complex values of x.

10. (a) The error function is an important function in many branches of applied mathematics. It is defined by an integral

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

The integral cannot be expressed in terms of more elementary functions. Given below are two methods of varying accuracy and efficiency. Write a program which uses each of these to compute the error function. You should evaluate $\operatorname{erf}(x)$ in the range $0 \le x \le 5$ in steps of 0.5. Use the built in function, if it exists, or values from a table as a standard of comparison for your computed approximate results. Print out the number of terms used.

• Substituting the Maclaurin expansion of the exponential function into the integral definition of $\operatorname{erf}(x)$ and integrating term by term we get

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \left(1 - t^2 + \frac{t^4}{2!} - \cdots\right) dt = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}.$$

This is an alternating series. For which values of x and n can Theorem 3.1.5 be used for estimating the truncation error ?

• The above Maclaurin series is an ill-conditioned series when x is large. The following formula is an alternative that can be derived by integration by parts, or by the hints given in Problem 23c of Sec. 3.2.

$$\operatorname{erf}(x) = \frac{2x}{\sqrt{\pi}} e^{-x^2} \left(1 + \frac{2x^2}{1 \cdot 3} + \frac{(2x^2)^2}{1 \cdot 3 \cdot 5} + \frac{(2x^2)^3}{1 \cdot 3 \cdot 5 \cdot 7} + \ldots\right)$$

Hint: To evaluate the series $S = \sum_{n=0}^{\infty} t_n$, first derive a recurrence relation $t_n = q_n t_{n-1}$ for computing successive terms.

See Problem 15 for another approach to the computation of $\operatorname{erf}(x)$ for large x.

(b) For a given x, which is the largest term of the second series? Plot for (say) $x = 10$, both the natural values of the terms (including the factor in front of the parenthesis) and their logarithms.

COMMENT: We call such a series a *a bell sum*; we shall return to bell sums in Problem 23 of Sec. 3.2 and in Sec. 3.3.4.

11. Compute a few terms of the expansions into powers of ϵ or k of each of the roots of the following equations, so that the error is $O(\epsilon^2)$ or $O(k^{-2})$ (ϵ is small and positive; k is large and positive). Note that some terms may have fractional or negative exponents. Also try to fit an expansion of the wrong form in some of these examples, and see what happens.

(a) $(1+\epsilon)z^2 - \epsilon = 0$; (b) $\epsilon z^3 - z^2 + 1 = 0$; (c) $\epsilon z^3 - z + 1 = 0$;

(d) $z^4 - (k^2+1)z^2 - k^2 = 0$, $(k^2 \gg 1)$.

12. The solution of the boundary value problem

$$(1+\epsilon)y'' - \epsilon y = 0, \qquad y(0) = 0, \quad y(1) = 1,$$

has an expansion of the form $y(t;\epsilon) = y_0(t) + y_1(t)\epsilon + y_2(t)\epsilon^2 + \ldots$. (Make sure you use the right boundary conditions for y_1, y_2.) By coefficient matching, set up differential equations and boundary conditions for y_0, y_1, y_2, and solve them.

COMMENT: It may seem strange that the expansion of the solution does not contain fractional powers, when the expansion of its characteristic roots does so (see Problem 11a). "Explain" this paradox.

13. In order to compute the Bessel function $J_0(x)$ for moderately large values of x, typically $x = 20$, to six correct decimals one considers the use of the power expansion

$$J_0(x) = \sum_{n=0}^{\infty} (-1)^n \frac{(x/2)^{2n}}{(n!)^2},$$

which converges for all x. Answer the following questions for $x = 20$:

(a) How many terms must one compute?

(b) How large is the largest term?

(c) How many digits must one use in the calculations?

(d) For how large values of x can the series provide almost full accuracy on your computer ?

COMMENT: This series converges for every x. It becomes more and more ill-conditioned as x increases. See examples and problems to Sec. 3.2.3 conerning other methods for computing Bessel functions.

(e) Construct by computer a picture for this expansion, similar to the upper part of Fig. 3.1.4. Extend to larger values of x and n, so that the illconditioned nature becomes visible.

14. (a) Derive the expansion of Example 3.1.18 by repeated integration by parts.

(b) Derive the Maclaurin expansion with the remainder according to (3.1.4) by the application of repeated integration by parts to the equation $f(z) - f(0) = z\int_0^1 f'(zt)dt$.

15. Derive three terms of a semiconvergent series for

$$2e^{x^2}\int_x^\infty e^{-t^2}dt, \quad (x \to \infty).$$

16. Give a strict proof of Theorem 3.1.5.

17. Some of the functions appearing in Table 3.1.1, in Problem 6, and in other examples and problems are *not single-valued* in the complex plane. Brush up your Complex Analysis, and find out how to define the branches, where these expansions are valid, and (if necessary) define cuts in the complex plane that must not be crossed. It turns out not to be necessary for these expansions. Why?

If you have access to programs for functions of complex variables (or to commands in some package for interactive computation), find out the conventions used for functions like square root, logarithm, powers, arctan etc. If the manual does not give enough detail, invent numerical tests, both with strategically chosen values of z and with random complex numbers in some appropriate domain around the origin.

For example, do you obtain $\ln\frac{z+1}{z-1} - \ln(z+1) + \ln(z-1) = 0$ for all z ?
Or, what values of $\sqrt{z^2-1}$ do you obtain for $z = \pm i$? What values should you obtain, if you want the branch which is positive for $z > 1$?
What do you obtain, if you apply Cauchy's coefficient formula or the Cauchy +FFT method to find a Laurent expansion for $\sqrt{z}$? Note that $\sqrt{z}$ is analytic everywhere in an annulus, but that does not help. The expansion is likely to become weird. Why?

18. (a) Suppose that r is located inside the unit circle; t is real. Show that

$$\frac{1-r^2}{1-2r\cos t + r^2} = 1 + 2\sum_{n=1}^\infty r^n \cos nt \qquad \frac{2r\sin t}{1-2r\cos t + r^2} = 2\sum_{n=1}^\infty r^n \sin nt.$$

Hint: First suppose that r is real. Then make analytic continuation.
(b) Let a be positive, $x \in [-a, a]$, while w is complex, $w \notin [-a, a]$. Let $r = r(w)$, $|r| < 1$ be a root of the quadratic $r^2 - \frac{2w}{a}r + 1 = 0$. Show that (with an appropriate definition of the square root)

$$\frac{1}{w-x} = \frac{1}{\sqrt{w^2-a^2}} \cdot \Big(1 + 2\sum_{n=1}^\infty r^n T_n(\frac{x}{a})\Big), \quad (w \notin [-a,\ a],\ x \in [-a, a]).$$

Find the corresponding expansion for $\frac{1}{1+x^2}$.
Hint: Set $w = i$, and take the imaginary part. Explain the order of magnitude of the error and the main features of the error curve in fig. 3.1.8.

19. Denote the n'th degree Chebyshev polynomial by $T_n(x)$, see Sec. 3.1.5. Note that $\deg\big(x^n - 2^{1-n}T_n(x)\big) = n - 2$.

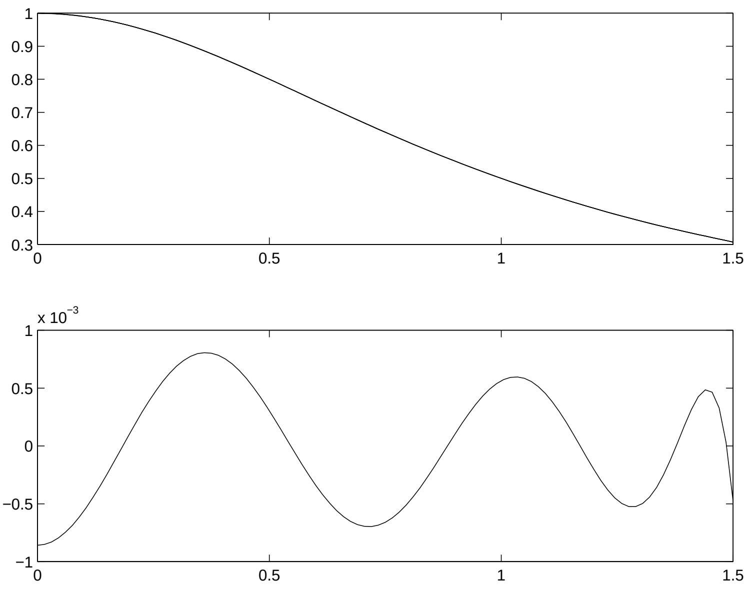


Figure 3.1.8. *Illustrations to Problem 18b.* Upper part: *The function* $f(x) = \frac{1}{1+x^2}$, $x \in [0, 1.5]$. Lower part: *The error of the expansion of* $f(x)$ *as a sum of Chebyshev polynomials* $T_n(x/1.5)$, $n \leq 10$. *(The scale is* 10^{-3}.*)*

(a) Find a polynomial $p(x)$ of degree $n-2$, such that $|x^n - p(x)| \leq 2(r/2)^n$ for $-r \leq x \leq r$. (p depends on r). If this is applied to the term of highest degree of a truncated power series for a function f, the series might be replaced by a polynomial of lower degree with a moderately increased bound for the truncation error. This process, which is called **economization of a power series**, can be repeated, so that one may end up with a useful polynomial approximation to $f(x)$ with a considerably smaller number of terms than the original truncated power series.

(b) The expansion of arsinh t into powers of t, truncated after t^7, is obtained from Problem 7b. Economize this to a polynomial approximation of the form $c_1 t + c_3 t^3$ in the interval $-\frac{1}{2} \leq t \leq \frac{1}{2}$. Give bounds for the truncation error for the original truncated expansion and for the economized expansion.

Hint: $T_5(x) = 5x - 20x^3 + 16x^5$; $\quad T_7(x) = -7x + 56x^3 - 112x^5 + 64x^7$.

(c) You see in Fig.3.1.5 that $|T_{12}(x)| \leq 1$ for $x \in [-1, 1]$, but is it really true that $T_{12}(3) > 10^8$?

20. (a) Show that

$$\int_{-1}^{1} T_m(x) T_n(x) \frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0, & \text{if } m \neq n; \\ \frac{1}{2}\pi & \text{if } m = n \neq 0; \\ \pi, & \text{if } m = n = 0 \end{cases}.$$

Comment: Because of the similarity with orthogonality conditions in $\mathbf{R}^n$, the Chebyshev polynomials of the first kind is called a system of **orthogonal polynomials**, with respect to the density function $(1-x^2)^{-1/2}$ on the interval $[-1,1]$.

Hint: Derive first the analogous relations for the functions $\{\cos nt\}$, $t \in [0,\pi]$. Show that the coefficients of a uniformly convergent Chebyshev series $f(x) = \sum_{m=0}^{\infty} c_m T_m(x)$, are given by

$$c_n = \frac{1}{K_n}\int_{-1}^{1} f(x)T_n(x)\frac{dx}{\sqrt{1-x^2}},$$

where $K_0 = \pi$, $K_n = \frac{1}{2}\pi$ if $n \neq 0$.

(b) Find a similar orthogonality condition for the Chebyshev polynomials of the second kind (with respect to a different density function).

21. **Clenshaw's algorithm**; ordinary (backward) version. Suppose that a sequence $\{\phi_k\}$ satisfies a three term recurrence relation

$$\phi_{k+1} = \alpha_k\phi_k - \gamma_k\phi_{k-1}, \quad k = 1 : n-2, \tag{3.1.39}$$

(a) Show that $S = \sum_{k=0}^{n-1} c_k\phi_k$ can be computed by the following algorithm due to Clenshaw, where we set $\alpha_0 = 0$.

```
$y_{n+1} = 0; \quad y_n = 0;$
for $k = n-1 : -1 : 0$, $y_k = \alpha_k y_{k+1} - \gamma_{k+1} y_{k+2} + c_k$; end
$S = y_0\phi_0 + y_1(\phi_1 - \alpha_0\phi_0).$
```

(b) Let α_j, $j = 1 : n$ be the zeros of the Chebyshev polynomial $T_n(x)$, $n \geq 1$. (There are, of course, simple trigonometric expressions for them.) Apply Clenshaw's algorithm to compute $\sum_{m=0}^{n-1} T_m(\alpha_1)T_m(x)$, for $x = \alpha_j$, $j = 1 : n$. It turns out that the results are remarkably simple. (An explanation to this will be found in Sec. 4.5.

(c) A forward version of Clenshaw's algorithm reads

```
$y_{-2} = 0; \quad y_{-1} = 0;$
for $k = 0 : n-1$, $y_k = \dfrac{-y_{k-2} + \alpha_k y_{k-1} + c_k}{\gamma_{k+1}}$; end
$S = c_n\phi_n + \gamma_n y_{n-1}\phi_{n-1} - y_{n-2}\phi_n.$
```

Add this version as an option to your program, and study Numerical Recipes [28, Sec. 5.4] from which this formula is quoted (with adaptation to our notation etc.). Make some test example of your own choice.

3.2 Difference Operators and Operator Expansions

Difference operators are handy tools for the deduction, analysis, and practical use of formulas for interpolation, differentiation, and quadrature of a function in terms of its values at equidistant arguments.

The simplest notations for difference operators and applications to derivatives, e.g., $d^2y/dx^2 \approx \Delta^2 y/(\Delta x)^2$, were mentioned in § 1.2.3. Difference schemes and the algebra of operators are treated in § 3.2.1. Operator expansions with applications are developed in § 3.2.2 and § 3.2.4. Peano's remainder theorem is presented in § 3.2.3. Finally, the basic facts about single **linear difference equations** are presented in § 3.2.5.

Other applications of difference operators include the convergence acceleration of infinite sequences, see Sec. 3.3, and multistep methods for differential equations, see Sec. 13.4.

3.2.1 Difference Operators and Their Simplest Properties

Let y denote a sequence $\{y_n\}$. Then we define the **shift operator** E (or translation operator) and the **forward difference operator** Δ by the relations

$$Ey = \{y_{n+1}\}, \qquad \Delta y = \{y_{n+1} - y_n\},$$

(see Sec. 1.2). E and Δ are thus operators which map one sequence to another sequence. Note, however, that if y_n is defined for $a \le n \le b$ only, then Ey_b is not defined, and the sequence Ey has fewer elements than the sequence y. (It is therefore sometimes easier to extend the sequences to infinite sequences, e.g., by adding zeros in both directions outside the original range of definition.)

These operators are **linear**, i.e., if α, β are real or complex constants and if y, z are two sequences, then $E(\alpha y + \beta z) = \alpha Ey + \beta Ez$, and similarly for Δ.

Powers of E and Δ are defined recursively, i.e.,

$$E^k y = E(E^{k-1}y), \qquad \Delta^k y = \Delta(\Delta^{k-1}y).$$

By induction, the first relation yields $E^k y = \{y_{n+k}\}$. We extend the validity of this relation to $k = 0$ by setting $E^0 y = y$ and to negative values of k. $\Delta^k y$ is called the kth difference of the sequence y. We make the convention that $\Delta^0 = 1$. There will be little use of Δ^k for negative values of k in this book, although Δ^{-1} can be interpreted as a summation operator.

Note that $\Delta y = Ey - y$, and $Ey = y + \Delta y$ for any sequence y. It is therefore convenient to express these as equations between operators: [23]

$$\Delta = E - 1, \qquad E = 1 + \Delta.$$

It can be shown that all formulas derived from the axioms of commutative algebra can be used for these operators, for example, the binomial theorem for positive integral k.

$$\Delta^k = (E-1)^k = \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} E^j, \tag{3.2.1}$$

[23]The identity operator is in this context traditionally denoted by 1.

$$E^k = (1+\Delta)^k = \sum_{j=0}^{k} \binom{k}{j} \Delta^j,$$

$$(\Delta^k y)_n = \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} y_{n+j},$$

$$y_{n+k} = (E^k y)_n = \sum_{j=0}^{k} \binom{k}{j} (\Delta^j y)_n.$$

We abbreviate the notation further and write, for example, $Ey_n = y_{n+1}$ instead of $(Ey)_n = y_{n+1}$, and $\Delta^k y_n$ instead of $(\Delta^k y)_n$. However, it is important to remember that Δ *operates on sequences* and not on elements of sequences. Thus, strictly speaking, this abbreviation is incorrect, though convenient. The formula for E^k will, in next subsection, be extended to an infinite series for non-integral values of k, but that is beyond the scope of algebra. A **difference scheme** consists of a sequence and its difference sequences, arranged in the following way:

$$\begin{array}{lllll}
y_0 & & & & \\
 & \Delta y_0 & & & \\
y_1 & & \Delta^2 y_0 & & \\
 & \Delta y_1 & & \Delta^3 y_0 & \\
y_2 & & \Delta^2 y_1 & & \Delta^4 y_0 \\
 & \Delta y_2 & & \Delta^3 y_1 & \\
y_3 & & \Delta^2 y_2 & & \\
 & \Delta y_3 & & & \\
y_4 & & & &
\end{array}$$

A difference scheme is best computed by successive subtractions; the formulas in (3.2.1) are used mostly in theoretical contexts.

In many applications the quantities y_n are computed in increasing order $n = 0, 1, 2, \ldots$, and it is natural that a difference scheme is constructed by means of the quantities previously computed. One therefore introduces the **backward difference operator** $\nabla y_n = y_n - y_{n-1} = (1 - E^{-1})y_n$. For this operator we have

$$\nabla^k = (1 - E^{-1})^k, \qquad E^{-k} = (1 - \nabla)^k. \tag{3.2.2}$$

Note the **reciprocity** in the relations between ∇ and E^{-1}.

Any linear combination of the elements y_n, $y_{n-1}, \ldots$ y_{n-k} can also be expressed as a linear combination of y_n, $\nabla y_n, \ldots, \nabla^k y_n$, and vice versa [24] . For example, $y_n + y_{n-1} + y_{n-2} = 3y_n - 3\nabla y_n + \nabla^2 y_n$, because
$1 + E^{-1} + E^{-2} = 1 + (1-\nabla) + (1-\nabla)^2 = 3 - 3\nabla + \nabla^2$. By the reciprocity, we also obtain $y_n + \nabla y_n + \nabla^2 y_n = 3y_n - 3y_{n-1} + y_{n-2}$.

[24] An analogous statement holds for the elements y_n, $y_{n+1}, \ldots, y_{n+k}$ and forward differences.

In this notation the difference scheme reads

$$\begin{array}{ccccc}
y_0 & & & & \\
 & \nabla y_1 & & & \\
y_1 & & \nabla^2 y_2 & & \\
 & \nabla y_2 & & \nabla^3 y_3 & \\
y_2 & & \nabla^2 y_3 & & \nabla^4 y_4 \\
 & \nabla y_3 & & \nabla^3 y_4 & \\
y_3 & & \nabla^2 y_4 & & \\
 & \nabla y_4 & & & \\
y_4 & & & &
\end{array}$$

In the backward difference scheme the subscripts are constant along diagonals directed upwards (backwards) to the right, while, in the forward difference scheme, subscripts are constant along diagonals directed downwards (forwards). Note, e.g., that $\nabla^k y_n = \Delta^k y_{n-k}$. In a computer, a backward difference scheme is preferably stored as a lower triangular matrix.

Example 3.2.1.

Part of the difference scheme for the sequence $y = \{\dots, 0, 0, 0, 1, 0, 0, 0, \dots\}$ is given below.

$$\begin{array}{ccccccccc}
 & 0 & & 0 & & 1 & & -7 & \\
0 & & 0 & & 1 & & -6 & & 28 \\
 & 0 & & 1 & & -5 & & 21 & \\
0 & & 1 & & -4 & & 15 & & -56 \\
 & 1 & & -3 & & 10 & & -35 & \\
1 & & -2 & & 6 & & -20 & & 70 \\
 & -1 & & 3 & & -10 & & 35 & \\
0 & & 1 & & -4 & & 15 & & -56 \\
 & 0 & & -1 & & 5 & & -21 & \\
0 & & 0 & & 1 & & -6 & & 28 \\
 & 0 & & 0 & & -1 & & 7 &
\end{array}$$

This example shows the *effect of a disturbance in one element* on the sequence of the higher differences. Because the effect broadens out and grows quickly, difference schemes are useful in the investigation and correction of computational and other errors, so-called **difference checks**. Notice that, since the differences are *linear* functions of the sequence, a **superposition principle** holds. The effect of errors can thus be estimated by studying simple sequences such as the one above.

Example 3.2.2.

For the sequence $y_n = (-1)^n$ one finds easily that $\nabla y_n = 2y_n$, $\nabla^2 y_n = 2\nabla y_n = 4y_n, \dots, \nabla^k y_n = 2^k y_n$. If the error in the elements of the sequence are bounded by ϵ, it follows that the errors of the kth differences are bounded by $2^k\epsilon$. A rather small reduction of this bound is obtained if the errors are assumed to be independent random variables (Problem 3.3.26).

It is natural also to consider *difference operations on functions* not just on sequences. E and Δ map the function f onto functions whose values at the point x are

$$Ef(x) = f(x+h), \qquad \Delta f(x) = f(x+h) - f(x), \tag{3.2.3}$$

where h is the *step size*. Of course, Δf depends on h; in some cases this should be indicated in the notation. One can, for example, write $\Delta_h f(x)$, or $\Delta f(x;h)$. If we set $y_n = f(x_0 + nh)$, the difference scheme of the function with step size h is the same as for the sequence $\{y_n\}$. Again it is important to realize that, in this case, the operators act on *functions*, not on the values of functions. It would be more correct to write $f(x_0 + h) = (Ef)(x_0)$. Actually, the notation $(x_0)Ef$ would be even more logical, since the insertion of the value of the argument x_0 is the last operation to be done, and the convention for the order of execution of operators proceeds from right to left, but this notation would be too revolutionary. [25]

Example 3.2.3. The following is a difference scheme for a 4 decimal table of the function $f(x) = \tan x$, $x \in [1.30, 1.35]$, with step $h = 0.01$. The differences are given with 10^{-4} as unit.

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$	$\nabla^5 y$
1.30	3.6021					
		1450				
1.31	3.7471		113			
		1563		13		
1.32	3.9034		126		5	
		1689		18		−4
1.33	4.0723		144		1	
		1833		19		
1.34	4.2556		163			
		1996				
1.35	4.4552					

We see that the differences decrease roughly by a factor of 0.1—that indicates that the step size has been chosen suitably for the purpose of interpolation, numerical quadrature etc.—until the last two columns, where the rounding errors of the function values have a visible effect. In fact, even the sign of $\nabla^5 y$ is wrong, which is compatible with the result of Example 3.2.2.

Note, however, that *no new errors are introduced during the computation of the differences, but the effects of the original irregular errors of y grow exponentially.* We emphasize the word *irregular errors*, e.g., rounding errors in y, since systematic errors, e.g., the truncation errors in the numerical solution of a differential equation, often have a smooth difference scheme. For example, if the values of y have been produced by the iterative solution of an equation, where x is a parameter, with the same number of iterations for every x and y and the same algorithm for the first approximation, then the truncation error of y is likely to be a smooth function of x.

[25] The notation $[x_0]f$ occurs, however, naturally in connection with divided differences, Sec. 4.2.

Difference operators are in many respects similar to differentiation operators. Let f be a polynomial. By Taylor's formula, $\Delta f(x) = f(x+h) - f(x) = hf'(x) + \frac{1}{2}h^2 f''(x) + \ldots$. We see from this that $\deg \Delta f = \deg f - 1$. Similarly for differences of higher order; *if f is a polynomial of degree less than k, then $\Delta^{k-1} f(x) =$const., and $\Delta^p f(x) = 0$, $\forall p \geq k$.* The same holds for backward differences.

The following important result can be derived directly from Taylor's theorem with the integral form of the remainder. Assume that all derivatives of f up to kth order are continuous. If $f \in C^k$,

$$\Delta^k f(x) = h^k f^{(k)}(\zeta), \qquad \zeta \in [x, x+kh]. \tag{3.2.4}$$

Hence $h^{-k}\Delta^k f(x)$ is an approximation to $f^{(k)}(x)$; the error of this approximation approaches zero as $h \to 0$ (i.e., as $\zeta \to x$). As a rule, the error is approximately proportional to h. We postpone the proof to Ch. 4, where it appears as a particular case of a theorem concerning divided differences.

Even though difference schemes do not have the same importance today that they had in the days of hand calculations or calculation with desk calculators, they are still important conceptually, and we shall also see how they are still useful also in practical computing. In a computer it is more natural to store a difference scheme as an array, e.g. with y_n, ∇y_n, $\nabla^2 y_n, \ldots$, $\nabla^k y_n$ in a row (instead of along a diagonal).

Many formulas for differences are analogous to formulas for derivatives, though usually more complicated. The following results are among the most important.

Lemma 3.2.1.
It holds that $\Delta^k(a^x) = (a^h - 1)^k a^x$, $\nabla^k(a^x) = (1 - a^{-h})^k a^x$.
For sequences, i.e., if h=1, $\Delta^k\{a^n\} = (a-1)^k\{a^n\}$, $\Delta^k\{2^n\} = \{2^n\}$.

Proof. Let c be a given constant. For $k = 1$ we have

$$\Delta(ca^x) = ca^{x+h} - ca^x = ca^x a^h - ca^x = c(a^h - 1)a^x$$

The general result follows easily by induction. The backward difference formula is derived in the same way. ☐

Lemma 3.2.2. *The Difference of a Product*

$$\Delta(u_n v_n) = u_n \Delta v_n + \Delta u_n \, v_{n+1}.$$

Proof. We have $\Delta(u_n v_n) = u_{n+1}v_{n+1} - u_n v_n = u_n(v_{n+1} - v_n) + (u_{n+1} - u_n)v_{n+1}$.

Compare the above result with the formula for differentials, $d(uv) = udv + vdu$. Note that we have v_{n+1} (not v_n) on the right-hand side. ☐

Lemma 3.2.3. *Summation by Parts*

$$\sum_{n=0}^{N-1} u_n \Delta v_n = u_N v_N - u_0 v_0 - \sum_{n=0}^{N-1} \Delta u_n \, v_{n+1}.$$

Proof. (Compare the rule for integration by parts and its proof!) Notice that

$$\sum_{n=0}^{N-1} \Delta w_n = (w_1 - w_0) + (w_2 - w_1) + \ldots + (w_N - w_{N-1}) = w_N - w_0.$$

Use this on $w_n = u_n v_n$. From the result in Lemma 4.5.2 one gets after summation,

$$u_N v_N - u_0 v_0 = \sum_{n=0}^{N-1} u_n \Delta v_n + \sum_{n=0}^{N-1} \Delta u_n v_{n+1},$$

and the result follows. (For an extension, see Problem 1d.) □

3.2.2 The Calculus of Operators

Formal calculations with operators, using the rules of algebra and analysis, are often an elegant means of assistance in *finding approximation formulas that are exact for all polynomials of degree less than (say) k*, and they should therefore be useful for functions that can be accurately approximated by such a polynomial. Our calculations often lead to divergent (or semi-convergent) series, but the way we handle them can usually be justified by means of the theory of formal power series, of which a brief introduction was given at the end of § 3.1.3. The operator calculations also provide error estimates, asymptotically valid as the step size $h \to 0$. Strict error bounds can be derived by means of Peano's remainder theorem, § 3.2.3.

Operator techniques are sometimes, see, e.g., Sec. 3.3, successfully used in a way that it is hard, or even impossible, to justify by means of formal power series. It is then not trivial to formulate appropriate conditions for the success and to derive satisfactory error bounds and error estimates, but it can sometimes be done.

We make a digression about terminology. More generally, *the word* **operator** *is in this book used for a function that maps a linear space $\mathcal{S}$ into another linear space $\mathcal{S}'$.* $\mathcal{S}$ can, for example, be a space of functions, a coordinate space, or a space of sequences. The dimension of these spaces can be finite or infinite. For example, the differential operator D maps the infinite-dimensional space $C^1[a, b]$ of functions with a continuous derivative, defined on the interval $[a, b]$, into the space $C[a, b]$ of continuous functions on the same interval.

In the following we denote by $\mathcal{P}_k$ the set of polynomials of degree *less than k*. [26] Note that $\mathcal{P}_k$ is a k-dimensional linear space, for which $\{1, x, x^2, \ldots, x^{k-1}\}$ is a

[26] Some authors use similar notations to denote the set of polynomials of degree less than or equal to k. We regret that.

basis called the *power basis*; the coefficients $(c_1, c_2, \ldots, c_k)$ are then the *coordinates* of the polynomial p defined by $p(x) = \sum_{i=1}^{k} c_i x^{i-1}$.

For simplicity, we shall assume that the space of functions on which the operators are defined is $C^\infty(-\infty, \infty)$, i.e., the functions are infinitely differentiable on $(-\infty, \infty)$. This sometimes requires (theoretically) a modification of a function outside the bounded interval where it is interesting. There are techniques for achieving this, but they are beyond the scope of this book. Just imagine that they have been applied.

We define the following operators:

$Ef(x) = f(x+h)$	Shift (or translation) operator
$\Delta f(x) = f(x+h) - f(x)$	Forward difference operator
$\nabla f(x) = f(x) - f(x-h)$	Backward difference operator
$Df(x) = f'(x)$	Differentiation operator
$\delta f(x) = f(x+\frac{1}{2}h) - f(x-\frac{1}{2}h)$	Central difference operator
$\mu f(x) = \frac{1}{2}\big(f(x+\frac{1}{2}h) + f(x-\frac{1}{2}h)\big)$	Averaging operator

Suppose that the values of f are given on an equidistant grid only, e.g., $x_j = x_0 + jh$, $j = -M : N$, (j is integer). Set $f_j = f(x_j)$. Note that δf_j, $\delta^3 f_j \ldots$, (odd powers) and μf_j *cannot* be exactly computed; they are available halfway between the grid points. [27] The even powers $\delta^2 f_j$, $\delta^4 f_j \ldots$, and $\mu\delta f_j$, $\mu\delta^3 f_j \ldots$, *can* be exactly computed. This follows from the formulas

$$\mu\delta f(x) = \frac{1}{2}\big(f(x+h) - f(x-h)\big), \quad \mu\delta = \tfrac{1}{2}(\Delta + \nabla), \quad \delta^2 = \Delta - \nabla. \qquad (3.2.5)$$

Several other notations are in use, e.g., at the study of difference methods for partial differential equations D_{+h}, D_{0h}, D_{-h} are used instead of $\Delta, \mu\delta, \nabla$, respectively.

An operator P is said to be a **linear operator** if

$$P(\alpha f + \beta g) = \alpha P f + \beta P g$$

holds for arbitrary complex constants α, β and arbitrary functions f, g. The above six operators are all linear. The operation of multiplying by a constant α, is also a linear operator.

If P and Q are two operators, then their sum, product, etc., can be defined in the following way:

$$\begin{aligned}
(P+Q)f &= Pf + Qf,\\
(P-Q)f &= Pf - Qf,\\
(PQ)f &= P(Qf),\\
(\alpha P)f &= \alpha(Pf),\\
P^n f &= P \cdot P \cdots Pf, \quad n \text{ factors.}
\end{aligned}$$

[27] We shall see a way to get around this in Example 3.2.11.

Two operators are equal, $P = Q$ if $Pf = Qf$, for all f in the space of functions considered. Notice that $\Delta = E - 1$. One can show that the following rules hold for all linear operators:

$$P + Q = Q + P, \qquad P + (Q + R) = (P + Q) + R,$$
$$P(Q + R) = PQ + PR, \qquad P(QR) = (PQ)R.$$

The above six operators, E, Δ, ∇, hD, δ, and μ, and the combinations of them by these algebraic operations make a *commutative ring*, so $PQ = QP$ holds for these operators, and any algebraic identity that is generally valid in such rings can be used.

If $\mathcal{S} = \mathbf{R^n}$, $\mathcal{S}' = \mathbf{R^m}$, and the elements are *column* vectors, then the linear operators are matrices of size $[m, n]$. They do generally not commute.

If $\mathcal{S}' = \mathbf{R}$ or $\mathbf{C}$, the operator is called a **functional**. Examples of functionals are, if x_0 denotes a fixed (though arbitrary) point,

$$Lf = f(x_0),\ Lf = f'(x_0),\ Lf = \int_0^1 e^{-x} f(x)dx,\ \int_0^1 |f(x)|^2 dx;$$

all except the last one are **linear** functionals.

There is a subtle distinction here. For example, E is a linear operator that maps a function to a function. Ef is the function whose value at the point x is $f(x + h)$. If we consider a fixed point, e.g. x_0, then $(Ef)(x_0)$ is a scalar. This is therefore a *linear functional.* We shall allow ourselves to simplify the notation and to write $Ef(x_0)$, but it must be understood that E operates on the function f, not on the function value $f(x_0)$. This was just one example; simplifications like this will be made with other operators than E, and similar simplifications in notation were suggested earlier in this chapter. There are, however, situations, where it is, for the sake of clarity, advisable to return to the more specific notation with a larger number of parentheses.

If we represent the vectors in $\mathbf{R}^n$ by *columns* y, the linear functionals in $\mathbf{R}^n$ are the scalar products $a^T x = \sum i = 1^n a_i y_i$; every *row* a^T thus defines a linear functional.

Examples of linear functionals in $\mathcal{P}_k$ are linear combinations of a finite number of function values, $Lf = \sum a_j f(x_j)$. If $x_j = x_0 + jh$ the same functional can be expressed in terms of differences, e.g., $\sum a'_j \Delta^j f(x_0)$, see Problem 3. The main topic of this subsection is to show how operator methods can be used for finding approximations of this form to linear functionals in more general function spaces. First, we need a general theorem.

Theorem 3.2.4.

Let $x_1, x_2, \ldots, x_k$ be k distinct real (or complex) numbers. Then no non-trivial relation of the form

$$\sum_{j=1}^{k} a_j f(x_j) = 0 \tag{3.2.6}$$

can hold for all $f \in \mathcal{P}_k$. If we add one more point (x_0), there exists only one non-trivial relation of the form $\sum_{j=0}^{k} a'_j f(x_j) = 0$, (except that it can be multiplied by an arbitrary constant). In the equidistant case, i.e., if $x_j = x_0 + jh$, then $\sum_{j=0}^{k} a'_j f(x_j) \equiv c\Delta^k f(x_0)$, $c \neq 0$.

Proof. If (3.2.6) were valid for all $f \in \mathcal{P}_k$, then the linear system

$$\sum_{j=1}^{k} x_j^{i-1} a_j = 0, \quad i = 1, \ldots, k,$$

would have a non-trivial solution $(a_1, a_2, \ldots, a_k)$. The matrix of the system, however, is a Vandermonde matrix; its determinant is thus equal to the product of all differences $(x_i - x_j)$, $i > j$, $1 < i \leq k$, which is nonzero.

Now we add the point x_0. Suppose that there exist two relations,

$$\sum_{j=0}^{k} b_j f(x_j) = 0, \qquad \sum_{j=0}^{k} c_j f(x_j) = 0.$$

with linearly independent coefficient vectors. Then we can find a (non-trivial) linear combination, where x_0 has been eliminated, but this contradicts the result that we have just proved. Hence the hypothesis is wrong; the two coefficient vectors must be proportional. We have seen above that, in the equidistant case, $\Delta^k f(x_0) = 0$ is such a relation. More generally, we shall see in Chapter 4 that, for $k+1$ arbitrary distinct points, the kth order *divided difference* is zero for all $f \in \mathcal{P}_k$. ☐

COROLLARY: *Suppose that a formula for interpolation, numerical differentiation or integration etc. has been derived, for example by an operator technique, (although there are many other ways to derive such formulas). If it is a linear combination of the values of $f(x)$ at k* given *distinct points x_j, $j = 1 : k$, and is exact for all $f \in \mathcal{P}_k$, this formula is unique. (If it is exact for all $f \in \mathcal{P}_m$, $m < k$, only, it is not unique.)*

In particular, for any $\{c_j\}_{j=1}^{k}$, a unique polynomial $P \in \mathcal{P}_k$ is determined by the interpolation conditions $P(x_j) = c_j$, $j = 1 : k$.

Proof. The difference between two formulas that use the same function values would lead to a relation that is impossible, by the theorem. ☐

Now we shall go outside of polynomial algebra and consider also *infinite series of operators.* The Taylor series

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(x) + \ldots$$

can be written symbolically as

$$Ef = \Big(1 + hD + \frac{(hD)^2}{2!} + \frac{(hD)^3}{3!} + \ldots\Big) f.$$

We can here treat hD like an algebraic indeterminate, and consider the series inside the parenthesis (without the operand) as a *formal power series* [28] for the function e^{hD}, which is defined by this series. The reader is advised to take a look again at the last part of § 3.1.3.

For a formal power series the concepts of convergence and divergence do not exist. When the operator series acts on a function f, and is evaluated at a point c, we obtain an ordinary numerical series, related to the linear functional $Ef(c) = f(c+h)$. We know that this Taylor series may converge or diverge, depending on f, c, and h.

Roughly speaking, the last part of §3.1.3 tells that, with some care, "analytic functions" of one indeterminate can be handled with the same rules as analytic functions of one complex variable.

Theorem 3.2.5.

$$e^{hD} = E = 1+\Delta, \qquad e^{-hD} = E^{-1} = 1-\nabla, \qquad 2\sinh \tfrac{1}{2}hD = e^{hD/2} - e^{-hD/2} = \delta.$$

$$(1+\Delta)^{\theta} = (e^{hD})^{\theta} = e^{\theta hD}, \quad (\theta \in \mathbf{R}).$$

Proof. The first formula follows from the previous discussion. The second and the third formulas are obtained in a similar way. (Recall the definition of δ.) The last formula follows from the first formula together with Lemma 3.1.8 (in Sec. 3.1.3). □

It follows from the power series expansion that $(e^{hD})^{\theta} f(x) = e^{\theta hD} f(x) = f(x+\theta h)$, when it converges. Since $E = e^{hD}$ it is natural to *define* $E^{\theta} f(x) = f(x+\theta h)$, and we extend this definition also to such values of θ that the power series for $e^{\theta hD} f(x)$ is divergent. Note that,e.g., the formula $E^{\theta_2} E^{\theta_1} f(x) = E^{\theta_2+\theta_1} f(x)$, follows from this definition.

When one works with operators or functionals it is advisable to avoid notations like Δx^n, $De^{\alpha x}$, where the variables appear in the operands. For two important functions we therefore set

$$F_\alpha : F_\alpha(x) = e^{\alpha x}; \qquad f_n : f_n(x) = x^n. \tag{3.2.7}$$

Let P be any of the operators mentioned above. When applied to F_α it acts like a scalar that we shall call **the scalar of the operator** [29] and denote it by $\mathrm{sc}(P)$,

$$PF_\alpha = \mathrm{sc}(P) F_\alpha.$$

We may also write $(P; h\alpha)$ if it is desirable to emphasize its dependence on $h\alpha$. (We normalize the operators so that this is true, e.g., we work with hD instead of D.) Note that

$$\mathrm{sc}(\beta P + \gamma Q) = \beta \mathrm{sc}(P) + \gamma \mathrm{sc}(Q), \ (\beta, \gamma \in \mathbf{C}), \quad \mathrm{sc}(PQ) = \mathrm{sc}(P)\mathrm{sc}(Q),$$

[28] We now abandon the bold-type notation for indeterminates and formal power series used in § 3.1.3.

[29] In applied Fourier analysis this scalar is, for $\alpha = i\omega$, often called the *symbol of the operator*.

For our most common operators we obtain

$$(E^\theta) = e^{\theta h\alpha}; \qquad \mathrm{sc}(\nabla) = \mathrm{sc}(1 - E^{-1}) = 1 - e^{-h\alpha}; \tag{3.2.8}$$
$$\mathrm{sc}(\Delta) = \mathrm{sc}(E - 1) = e^{h\alpha} - 1; \tag{3.2.9}$$
$$\mathrm{sc}(\delta) = \mathrm{sc}(E^{1/2} - E^{-1/2}) = e^{h\alpha/2} - e^{-h\alpha/2}.$$

Let Q_h be one of the operators hD, Δ, δ, ∇. It follows from the last formulas that

$$\mathrm{sc}(Q_h) \sim h\alpha, \ (h \to 0); \quad |\mathrm{sc}(Q_h)| \le |h\alpha| e^{|h\alpha|}$$

The main reason for grouping these operators together is that each of them has the important property (3.2.4), i.e., $Q_h^k f(c) = h^k f^{(k)}(\zeta)$, where ζ lies in the smallest interval that contains all the arguments used in the computation of $Q_h^k f(c)$. Hence,

$$f \in \mathcal{P}_k \quad \Rightarrow \quad Q_h^n f = 0, \quad \forall n \ge k. \tag{3.2.10}$$

This property [30] makes each of these four operators well suited to be the indeterminate in a formal power series that, hopefully, will be able to generate a sequence of approximations, L_1, L_2, $L_3 \ldots$, to a given linear operator L. L_n is the n'th partial sum of a formal power series for L. Then

$$f \in \mathcal{P}_k \quad \Rightarrow \quad L_n f = L_k f, \quad \forall n \ge k. \tag{3.2.11}$$

We shall see in the next theorem that, for expansion into powers of Q_h, $\lim_{n\to\infty} L_n f(x) = Lf(x)$ if f is a polynomial. This is not quite self-evident, because it is not true for all functions f, and we have seen in § 3.1.3 that it can happen that an expansion converges to a "wrong result". We shall see more examples of that later. Convergence does not necessarily imply validity.

Suppose that z is a complex variable, and that $\phi(z)$ is analytic at the origin, i.e., $\phi(z)$ is equal to its Maclaurin series, (say) $\phi(z) = a_0 + a_1 z + a_2 z^2 + \ldots$, if $|z| < \rho$ for some $\rho > 0$. For multivalued functions we always refer to the principal branch. The operator function $\phi(Q_h)$ is usually defined by the *formal* power series, $\phi(Q_h) = a_0 + a_1 Q_h + a_2 Q_h^2 + \ldots$, where Q_h is treated like an algebraic indeterminate.

The operators E, hD, Δ, δ, ∇ and μ are related to each others. See Table 3.2.2 that is adapted from an article by the eminent blind British mathematician W. G. Bickley (1948). Some of these formulas follow almost directly from the definitions, others are derived in this subsection, and the rest are left for Problem 5e. We find the value sc(·) for each of these operators by *substituting α for D in the last column of the table.* (Why?)

Example 3.2.4. *Express E in terms of ∇.*

The definition of ∇ reads in operator form $E^{-1} = 1 - \nabla$. This can be looked upon as a formal power series (with only two non-vanishing terms) for the reciprocal of E with ∇ as the indeterminate. By the rules for formal power series mentioned in § 3.1.3 , we obtain *uniquely* $E = (E^{-1})^{-1} = (1 - \nabla)^{-1} = 1 + \nabla + \nabla^2 + \ldots$. We find in the table an equivalent expression containing a fraction line.

[30] The operators E and μ do *not* possess this property.

Table 3.2.1. *Bickley's table of relations between difference operators*

	E	Δ	δ	∇	hD
E	E	$1+\Delta$	$1+\frac{1}{2}\delta^2+\delta\sqrt{1+\frac{1}{4}\delta^2}$	$\dfrac{1}{1-\nabla}$	e^{hD}
Δ	$E-1$	Δ	$\delta\sqrt{1+\frac{1}{4}\delta^2}+\frac{1}{2}\delta^2$	$\dfrac{\nabla}{1-\nabla}$	$e^{hD}-1$
δ	$E^{1/2}-E^{-1/2}$	$\Delta(1+\Delta)^{-1/2}$	δ	$\nabla(1-\nabla)^{-1/2}$	$2\sinh\frac{1}{2}hD$
∇	$1-E^{-1}$	$\dfrac{\Delta}{1+\Delta}$	$\delta\sqrt{1+\frac{1}{4}\delta^2}-\frac{1}{2}\delta^2$	∇	$1-e^{-hD}$
hD	$\ln E$	$\ln(1+\Delta)$	$2\sinh^{-1}\frac{1}{2}\delta$	$-\ln(1-\nabla)$	hD
μ	$\frac{1}{2}(E^{1/2}+E^{-1/2})$	$\dfrac{1+\frac{1}{2}\Delta}{(1+\Delta)^{1/2}}$	$\sqrt{1+\frac{1}{4}\delta^2}$	$\dfrac{1-\frac{1}{2}\nabla}{(1-\nabla)^{1/2}}$	$\cosh\frac{1}{2}hD$

Suppose that we have proved the last column of the table. So, $\mathrm{sc}(\nabla) = 1-e^{-h\alpha}$, hence $\mathrm{sc}((1-\nabla)^{-1}) = (e^{-h\alpha})^{-1} = e^{h\alpha} = \mathrm{sc}(E)$.

Example 3.2.5.

Suppose that we have proved the first and the last columns of Bickley's table (except for the equation $hD = \ln E$). We shall prove one of the formulas in the second column, namely the equation $\delta = \Delta(1+\Delta)^{-1/2}$.

By the first column, the right hand side is equal to $(E-1)E^{-1/2} = E^{1/2} - E^{-1/2} = \delta$, Q.E.D.

We shall also compute $\mathrm{sc}(\Delta(1+\Delta)^{-1/2})$. Since $\mathrm{sc}(\Delta) = e^{h\alpha}-1$ we obtain $\mathrm{sc}(\Delta(1+\Delta)^{-1/2}) = (e^{h\alpha}-1)(e^{h\alpha})^{-1/2} = e^{\frac{1}{2}h\alpha} - e^{-\frac{1}{2}h\alpha} = 2\sinh\frac{1}{2}h\alpha = \mathrm{sc}(\delta)$.

By the aid of Bickley's table, we are in a position to transform L into the form $\phi(Q_h)R_h$. [31]

- Q_h is the one of the four operators, hD, Δ, δ, ∇, which we have chosen to be the "indeterminate".

$$Lf \simeq \phi(Q_h)f = (a_0 + a_1Q_h + a_2Q_h^2 + \ldots)f. \tag{3.2.12}$$

The coefficients a_j are the same as the Maclaurin coefficients of $\phi(z)$, $z \in \mathbf{C}$ if $\phi(z)$ is analytic at the origin. They can be determined by the techniques described in §3.1.3 and §3.1.4. The meaning of the relation $\simeq$ will hopefully be clear from the following theorem.

- R_h is, e.g., $\mu\delta$ or E^k, k integer, or more generally any linear operator with the properties that $R_hF_\alpha = \mathrm{sc}(R_h)F_\alpha$, and that the values of $R_hf(x_n)$ on the grid $x_n = x_0 + nh$, n integer, are determined by the values of f on the same grid.

Theorem 3.2.6. *Recall the notation Q_h for either of the operators $\Delta, \delta, \nabla, hD$,*

[31] A sum of several such expressions with different indeterminates is also OK.

and the notations $F_\alpha(x) = e^{\alpha x}$, $f_n(x) = x^n$. *Note that*

$$F_\alpha(x) = \sum_{n=0}^{\infty} \frac{\alpha^n}{n!} f_n(x), \tag{3.2.13}$$

Also recall the scalar of an operator and its properties, e.g., $LF_\alpha = \mathrm{sc}(L)F_\alpha$, $Q_h^j F_\alpha = (\mathrm{sc}(Q_h))^j F_\alpha$; *for the operators under consideration the scalar depends on* $h\alpha$.

Assumptions:

1. *A formal power series equation* $L = \sum_{j=0}^{\infty} a_j Q_h^j$ *has been derived.* [32] *Furthermore,* $|\mathrm{sc}(Q_h)| < \rho$, *where* ρ *is the convergence radius of the series* $\sum a_j z^j$, $z \in \mathbf{C}$, *and*

$$\mathrm{sc}(L) = \sum_{j=0}^{\infty} a_j (\mathrm{sc}(Q_h))^j. \tag{3.2.14}$$

2.
$$L \frac{\partial^n}{\partial \alpha^n} F_\alpha(x) = \frac{\partial^n}{\partial \alpha^n} (LF_\alpha)(x)$$
at $\alpha = 0$, *or equivalently,*
$$L \int_C \frac{F_\alpha(x)\, d\alpha}{\alpha^{n+1}} = \int_C \frac{(LF_\alpha)(x)\, d\alpha}{\alpha^{n+1}}. \tag{3.2.15}$$
where C *is any circle with the origin as center.*

3. *The domain of* x *is a bounded interval* I_1 *in* $\mathbf{R}$.

Then

$$LF_\alpha = \Big(\sum_{j=0}^{\infty} a_j Q_h^j\Big) F_\alpha, \quad \text{if } |\mathrm{sc}(Q_h)| < \rho, \tag{3.2.16}$$

$$Lf(x) = \sum_{j=0}^{k-1} a_j Q_h^j f(x), \quad \text{if } f \in \mathcal{P}_k, \tag{3.2.17}$$

for any positive integer k.

A **strict error bound** *for (3.2.17), if* $f \notin \mathcal{P}_k$, *is obtained in Theorem 3.2.7, due to Peano.*

An **asymptotic error estimate** *(as* $h \to 0$ *for fixed* k*) is given by the first neglected non-vanishing term* $a_r Q_h^r f(x) \sim a_r (hD)^r f(x)$, $r \ge k$, *if* $f \in C^r[I]$, *where the interval* I *must contain all the points used in the evaluation of* $Q_h^r f(x)$.

Proof. By Assumption 1,

$$LF_\alpha = \mathrm{sc}(L)F_\alpha = \lim_{J\to\infty} \sum_{0}^{J-1} a_j \mathrm{sc}(Q_h^j) F_\alpha = \lim_{J\to\infty} \sum_{0}^{J-1} a_j Q_h^j F_\alpha = \lim_{J\to\infty} (\sum_{0}^{J-1} a_j Q_h^j) F_\alpha,$$

[32]To simplify the writing, the operator R_h is temporarily neglected. See one of the comments below.

hence $LF_\alpha = (\sum_0^\infty Q_h^j)F_\alpha$. This proves the first part of the theorem.

By (3.2.13), Cauchy's formula (3.1.20) and Assumption 2,

$$\frac{2\pi i}{n!} Lf_n(x) = L\int_C \frac{F_\alpha(x)\,d\alpha}{\alpha^{n+1}} = \int_C \frac{(LF_\alpha)(x)\,d\alpha}{\alpha^{n+1}}$$
$$= \int_C \sum_{j=0}^{J-1} \frac{a_j Q_h^j F_\alpha(x)\,d\alpha}{\alpha^{n+1}} + \int_C \sum_{j=J}^{\infty} \frac{a_j \mathrm{sc}(Q_h)^j F_\alpha(x)\,d\alpha}{\alpha^{n+1}}.$$

Let ϵ be any positive number. Choose J so that the modulus of the last term becomes $\epsilon\theta_n 2\pi/n!$, where $|\theta_n| < 1$. This is possible, since $|\mathrm{sc}(Q_h)| < \rho$, see Assumption 1. Hence, for every $x \in I_1$,

$$Lf_n(x) - \epsilon\theta_n = \frac{n!}{2\pi i} \sum_{j=0}^{J-1} a_j Q_h^j \int_C \frac{F_\alpha(x)\,d\alpha}{\alpha^{n+1}} = \sum_{j=0}^{J-1} a_j Q_h^j f_n(x) = \sum_{j=0}^{k-1} a_j Q_h^j f_n(x).$$

The last step holds if $J \geq k > n$, because, by (3.2.10), $Q_h^j f_n = 0$ for $j > n$. It follows that $|Lf_n(x) - \sum_{j=0}^{k-1} a_j Q_h^j f_n(x)| < \epsilon$ for every $\epsilon > 0$, hence $Lf_n = \sum_{j=0}^{k-1} a_j Q_h^j f_n$.

If $f \in \mathcal{P}_k$, f is a linear combination of f_n, $n = 0 : k-1$. Hence $Lf = \sum_{j=0}^{k-1} a_j Q_h^j f$ if $f \in \mathcal{P}_k$. This proves the second part of the theorem.

The error bound is derived in § 3.2.3. Recall the important formula (3.2.4) that expresses the k'th difference as the value of the k'th derivative in a point located in an interval that contains all the points used in the in the computation of the k'th difference., i.e., the ratio of the error estimate $a_r(hD)^r f(x)$ to the true truncation error tends to 1, as $h \to 0$. □

Comments to Theorem 3.2.6:

- This theorem is concerned with series of powers of the four operators collectively denoted Q_h. One may try to use operator techniques also to *find* a formula involving, e.g., an infinite expansion into powers of the operator E. Then one should try afterwards to find sufficient conditions for the validity of the result. This procedure will be illustrated in connection with Euler-Maclaurin's formula in Sec. 3.3.
  Sometimes, operator techniques which are not covered by this theorem can, after appropriate restrictions, be justified (or even replaced) by *transform methods*, e.g., z-transforms, Laplace or Fourier transforms.
- The operator R_h that was introduced just before the theorem, was neglected in the proof, in order to simplify the writing. We now have to multiply the operands by R_h in the proof and in the results. This changes practically nothing for F_α, since $R_h F_\alpha = \mathrm{sc}(R_h)F_\alpha$. In (3.2.17) there is only a trivial change, because the polynomials f and $R_h f$ may not have the same degree. For example, if $R_h = \mu\delta$ and $f \in P_k$—an interesting case in Example 3.2.11—then $R_h f \in P_{k-1}$.
- The verification of the assumptions typically offers no difficulties.
- It follows from the linearity of Eq. (3.2.16) that *it is satisfied also if F_α is replaced by a linear combination of exponential functions F_α with different α*,

provided that $|\mathrm{sc}(Q_h)| < \rho$ for all the occurring α. With some care, one can let the linear combination be an infinite series or an integral.

- There are two things to note in connection with the asymptotic error estimates. First, the step size should be small enough; this means in practice that, in the beginning, the magnitude of the differences should decrease rapidly, as their order increases. When the order of the differences becomes large, it often happens that the moduli of the differences also become increasing. This can be due to two causes: semi-convergence (see the next comment) and/or rounding errors.
  The *rounding errors* of the data may have so large effects on the high order differences [33] that the error estimation does not make sense. One should then use a smaller value of the order k, where the rounding errors have a smaller influence. An advantage with the use of a difference scheme is that it is relatively easy to choose the order k adaptively, and sometimes also the step size h.
  This comment is of particular importance for numerical differentiation. Numerical illustrations and further comments are given below in Example 3.2.6 and Problem 6b, and in several other places.
- The sequence of approximations to Lf may converge or diverge, depending on f and h. It is also often *semiconvergent*, recall § 3.1.8, but in practice the rounding errors mentioned in the previous comment, have often, though not always, taken over already, when the truncation error passes its minimum. See Problem 6b.

Example 3.2.6. *The Backwards Differentiation Formula.*

By Theorem 3.2.5, $e^{-hD} = 1 - \nabla$. We look upon this as a formal power series; the indeterminate is $Q_h = \nabla$. By Example 3.1.12,

$$L = hD = -\ln(1-\nabla) = \nabla + \frac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3 + \dots \tag{3.2.18}$$

Verification of the assumptions of Theorem 3.2.6: [34]

1. $\mathrm{sc}(\nabla) = 1 - e^{-h\alpha}$; the convergence radius is $\rho = 1$.

$$\mathrm{sc}(L) = \mathrm{sc}(hD) = h\alpha; \quad \sum_{j=1}^{\infty} \mathrm{sc}(\nabla)^j/j = -\ln(1-(1-e^{-h\alpha})) = h\alpha.$$

The convergence condition $|\mathrm{sc}(\nabla)| < 1$ reads $h\alpha > -\ln 2 = -0.69$ if α is real, $|h\omega| < \pi/3$ if $\alpha = i\omega$.

2. For $\alpha = 0$, $D\frac{\partial^n}{\partial\alpha^n}(e^{\alpha x}) = Dx^n = nx^{n-1}$.
   By Leibniz' rule: $\frac{\partial^n}{\partial\alpha^n}(\alpha e^{\alpha x}) = 0x^n + nx^{n-1}$.

By the theorem, we now obtain *a formula for numerical differentiation that is exact for all $f \in \mathcal{P}_k$.*

$$hf'(x) = \Big(\nabla + \frac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3 + \dots + \frac{1}{k-1}\nabla^{k-1}\Big) f(x) \tag{3.2.19}$$

[33] Recall Example 3.2.2

[34] Recall the definition of the scalar sc(·), after (3.2.7).

By Theorem 3.2.4, this is the *unique* formula of this type that uses the values of $f(x)$ at the k points $x_n : -h : x_{n-k+1}$. The same approximation can be derived in many other ways, perhaps with a different appearance, see Ch. 4. This derivation has several advantages; the same expansion yields approximation formulas for every k, and *if $f \in C^k$, $f \notin \mathcal{P}_k$, the first neglected term, i.e., $\frac{1}{k}\nabla_h^k f(x_n)$, provides an* **asymptotic error estimate**, if $f^{(k)}(x_n) \neq 0$.

We now apply this formula to the table in Example 3.2.3, where $f(x) = \tan x$, $h = 0.01$, $k = 6$,

$$0.01f'(1.35) \approx 0.1996 + \frac{0.0163}{2} + \frac{0.0019}{3} + \frac{0.0001}{4} - \frac{0.0004}{5},$$

i.e., we obtain a sequence of approximate results,

$$f'(1.35) \approx 19.96, 20.78, 20.84, 20.84, 20.83.$$

The correct value to 3D is $(\cos 1.35)^{-2} = 20.849$. Note that the last result is worse than the next to last. Recall the last comments to the theorem. In this case this is due to the rounding errors of the data. Upper bounds for their effect of the sequence of approximate values of $f'(1.35)$ is, by Example 3.2.2, shown in the series

$$10^{-2}(1 + \frac{2}{2} + \frac{4}{3} + \frac{8}{4} + \frac{16}{5} + \dots)$$

A larger version of this problem was run on a computer with the machine unit $2^{-53} \approx 10^{-16}$; $f(x) = \tan x$, $x = 1.35 : -0.01 : 1.06$. In the beginning the error decreases rapidly, but after 18 terms the rounding errors take over, and the error then grows almost exponentially (with constant sign). The eighteenth term and its rounding error have almost the same modulus (but opposite sign). The smallest error equals $5\,10^{-10}$, and is obtained after 18 terms; after 29 terms the actual error has grown to $2\,10^{-6}$. Such a large number of terms is seldom used in practice, unless a very high accuracy is demanded. See also Problem 6b, a computer exercise that offers both similar and different experiences.

Eq. (3.2.18)—or its variable step size variant in Ch. 4—is called the *Backwards Differentiation Formula.* It is the basis of the important *BDF method* for the numerical integration of ordinary differential equations, see Ch.13.

Coefficients for *Backwards differentiation formulas for higher derivatives*, are obtained from the equations

$$(hD/\nabla)^k = (-\ln(1-\nabla)/\nabla)^k.$$

The following formulas were computed by means of the matrix representation of a truncated power series [35] , see Example 3.1.11 and Problem 3.1.8(d).

$$\begin{pmatrix} hD/\nabla \\ (hD/\nabla)^2 \\ (hD/\nabla)^3 \\ (hD/\nabla)^4 \\ (hD/\nabla)^5 \end{pmatrix} = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1 & 1 & 11/12 & 5/6 & 137/180 \\ 1 & 3/2 & 7/4 & 15/8 & 29/15 \\ 1 & 2 & 17/6 & 7/2 & 967/240 \\ 1 & 5/2 & 25/6 & 35/6 & 1069/144 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \nabla \\ \nabla^2 \\ \nabla^3 \\ \nabla^4 \end{pmatrix}. \qquad (3.2.20)$$

[35]The rows of the matrix in (3.2.20) are the first rows taken from the matrix representation of each of the expansions $(hD/\nabla)^k$, $k = 1 : 5$.

When the effect of the *irregular errors* of the data on a term becomes larger in magnitude than the term itself, the term should, of course, be neglected; it does more harm than good. This happens relatively early for the derivatives of high order; see Problem 6. When these formulas are to be used inside a program (rather than during an interactive post-processing of results of an automatic computation), some rules for *automatic truncation* have to be designed; an interesting kind of detail in scientific computing.

The *forwards differentiation formula*, which is analogously based on the operator series,

$$hD = \ln(1+\Delta) = \Delta - \frac{1}{2}\Delta^2 + \frac{1}{3}\Delta^3 \pm \dots \tag{3.2.21}$$

is sometimes useful too. We obtain the coefficients for derivatives of higher order by inserting minus signs in the second and fourth columns of the matrix in (3.2.20).

A grid (or a table) may be too sparse to be useful for numerical differentiation and for the computation of other linear functionals. For example, we saw above that the successive backward differences of $e^{i\omega x}$ increase exponentially if $|\omega h| > \pi/3$. In such a case the grid, where the values are given, gives insufficient information about the function. One also says that "the grid does not *resolve* the function". This is often indicated by a strong variation in the higher differences. However, even this indication can sometimes be absent. An extreme example is, $f(x) = \sin(\pi x/h)$, on the grid $x_j = jh$, $j = 0, \pm 1, \pm 2, \dots$. All the values, all the higher differences, and thus the estimates of $f'(x)$ at all grid points are zero, but the correct values of $f'(x_j)$ are certainly not zero. So, this is an example where the expansion (trivially) converges, but it is not valid! (Recall the discussion of a Maclaurin expansion for a non-analytic function at the end of Sec. 3.1.3. Now a similar trouble can occur also for an analytic function.)

A less trivial example is given by the functions

$$f(x) = \sum_{n=1}^{20} a_n \sin(2\pi n x), \quad g(x) = \sum_{n=1}^{10} (a_n + a_{10+n}) \sin(2\pi n x).$$

$f(x) = g(x)$ on the grid, hence they have the same difference scheme, but $f'(x) \neq g'(x)$ on the grid, and typically $f(x) \neq g(x)$ between the grid points.

*** This is perhaps the right place to introduce the "local time scale" and relate it to a difference scheme and to the radius of convergence of the local Taylor series. This is not yet written, except for its application to ODE's in Ch. 13. ***

3.2.3 The Peano Theorem

One can often, by a combination of theoretical and numerical evidence, rely on asymptotic error estimates. Since there are exceptions, it is interesting that there are two general methods for deriving strict error bounds. We call one of them the **norms and distance formula**. It is not restricted to polynomial approximation, and it is typically easy to use, but it requires some advanced concepts and it often overestimates the error. We therefore postpone the presentation of that method to a later chapter.

We shall now present another method, due to Peano. [36] Consider a linear functional $\tilde{L}$, e.g., $\tilde{L}f = \sum_{j=1}^{p} b_j f(x_j)$, suggested for the approximate computation of another linear functional L, e.g., $Lf = \int_0^1 \sqrt{x} f(x)dx$. Suppose that it is exact, when it is applied to any polynomial of degree less than k: In other words, $\tilde{L}f = Lf$, for all $f \in \mathcal{P}_k$. The remainder is then itself a linear functional, $R = L - \tilde{L}$, with the special property that

$$Rf = 0 \quad \text{if} \quad f \in \mathcal{P}_k.$$

Next theorem gives a representation for such functionals, which provides a universal device for deriving error bounds for approximations of the type that we are concerned with. Let $f \in C^n[a,b]$. In order to make the discussion less abstract we confine it to functionals of the following form, $0 \le m < n$,

$$Rf = \int_a^b \phi(x)f(x)dx + \sum_{j=1}^{p} \big(b_{j,0}f(x_j) + b_{j,1}f'(x_j) + \ldots + b_{j,m}f^{(m)}(x_j)\big), \quad (3.2.22)$$

where the function ϕ is integrable, and the points x_j lie in the bounded real interval $[a,b]$, and $b_{j,m} \neq 0$ for at least one value of j. Moreover, *we assume that*

$$Rp = 0 \text{ for all } p \in \mathcal{P}_k. \quad (3.2.23)$$

We need some terminology. The function [37]

$$t_+ = \max(t,0); \quad t_+^j = \left(t_+\right)^j; \quad t_+^0 = \frac{1+\text{sign}t}{2}; \quad (3.2.24)$$

Note that $t_+^j \in C^{j-1}$, $(j \ge 1)$.

The **Peano kernel** $K(u)$ of the functional R is defined by the equation,

$$K(u) = \frac{1}{(k-1)!}R_x(x-u)_+^{k-1}, \quad x \in [a,b], u \in (-\infty,\infty). \quad (3.2.25)$$

The subscript in R_x indicates that R acts on the variable x (not u).

The function $K(u)$ *vanishes outside* $[a,b]$, because:

• if $u > b$ then $u > x$, hence $(x-u)_+^{k-1} = 0$ and $K(u) = 0$,

• if $u < a$ then $x > u$. It follows that $(x-u)_+^{k-1} = (x-u)^{k-1} \in \mathcal{P}_k$, hence $K(u) = 0$, by (3.2.25) and (3.2.23).

If $\phi(x)$ is a polynomial then $K(u)$ becomes a piecewise polynomial; the points x_j are the joints of the pieces. In this case $K \in C^{k-m-2}$; the order of differentiability may be lower, if ϕ has singularities.

We are now in a position to prove an important theorem.

Theorem 3.2.7. *Peano's Remainder Theorem.*

[36] Giuseppe Peano (1858-1932), Italian mathematician and logician.

[37] We use the neutral notation t here for the variable, to avoid to tie up the function too closely with the variables x and u which play a special role in the following. The function t_+^0 is often denoted $H(t)$ an is known as the *Heaviside unit step function.* The function sign is defined in §3.1.2.

Suppose that $Rp = 0$ for all $p \in \mathcal{P}_k$. Then[38] *, for all $f \in C^k[a,b]$,*

$$Rf = \int_{-\infty}^{\infty} f^{(k)}(u)K(u)du. \tag{3.2.26}$$

The definition and some basic properties of the Peano kernel $K(u)$ were given above.

Proof. By Taylor's formula,

$$f(x) = \sum_{j=1}^{k-1} \frac{f^{(j)}(a)}{j!}(x-a)^j + \int_a^x \frac{f^{(k)}(u)}{(k-1)!}(x-u)^{k-1}du.$$

This follows from putting $n = k$, $z = x - a$, $t = (u-a)/(x-u)$ into (3.1.4). We rewrite the last term as $\int_a^\infty f^{(k)}(u)(x-u)_+^{k-1}du$. Then apply the functional $R = R_x$ to both sides. This yields

$$Rf = 0 + R\int_a^\infty \frac{f^{(k)}(u)(x-u)_+^{k-1}}{(k-1)!}du = \int_a^\infty \frac{f^{(k)}(u)R_x(x-u)_+^{k-1}}{(k-1)!}du,$$

since we can allow the interchange of the functional R with the integral, for the class of functionals that we are working with. The theorem then follows from (3.2.25). ☐

Corollary. *Suppose that $Rp = 0$ for all $p \in \mathcal{P}_k$. Then*

$$R_x(x-a)^k = k!\int_{-\infty}^{\infty} K(u)du. \tag{3.2.27}$$

For any $f \in C^k[a,b]$, $Rf = \frac{f^{(k)}(\xi)}{k!}R_x((x-a)^k)$, holds for some $\xi \in (a,b)$, if and only if $K(u)$ does not change its sign.

If $K(u)$ changes its sign, the best possible error bound *reads*

$$|Rf| \le \sup_{u\in[a,b]} |f^{(k)}(u)| \int_{-\infty}^{\infty} |K(u)|du;$$

a formula with $f^{(k)}(\xi)$ is not generally true in this case.

Proof. First suppose that $K(u)$ does not change sign. Then, by (3.2.26) and the mean value theorem of Integral Calculus, $Rf = f^{(k)}(\xi)\int_{-\infty}^{\infty} K(u)du$, $\xi \in [a,b]$. For $f(x) = (x-a)^k$ this yields (3.2.27). The "if" part of the corollary follows from the combination of these formulas for Rf and $R(x-a)^k$.

If $K(u)$ changes its sign, the "best possible bound" is approached by a sequence of functions f chosen so that (the continuous functions) $f^{(k)}(u)$ approach (the discontinuous function) sign $K(u)$. The "only if" part follows. ☐

[38]The definition of $f^{(k)}(u)$ for $u \notin [a,b]$ is arbitrary.

Example 3.2.7. The remainder of the *trapezoidal rule* (one step of length h) reads $Rf = \int_0^h f(x)dx - \frac{h}{2}(f(h) + f(0))$. We know that $Rp = 0$ for all $p \in \mathcal{P}_2$. The Peano kernel is zero for $u \notin [0, h]$, while for $u \in [0, h]$,

$$K(u) = \int_0^h (x-u)_+ dx - \frac{h}{2}((h-u)_+ + 0)) = \frac{(h-u)^2}{2} - \frac{h(h-u)}{2} = \frac{-u(h-u)}{2} < 0.$$

We also compute

$$\frac{Rx^2}{2!} = \int_0^h \frac{x^2}{2} dx - \frac{h \cdot h^2}{2 \cdot 2} = \frac{h^3}{6} - \frac{h^3}{4} = -\frac{h^3}{12}.$$

Since the Peano kernel does not change sign, we conclude that

$$Rf = -\frac{h^3}{12} f''(\xi), \quad \xi \in (0, h).$$

Example 3.2.8. *Peano kernels for difference operators.* Let $Rf = \Delta^3 f(a)$, and set $x_i = a + ih$, $i = 0 : 3$. Note that $Rp = 0$ for $p \in \mathcal{P}_3$. Then

$$Rf = f(x_3) - 3f(x_2) + 3f(x_1) - f(x_0),$$
$$2K(u) = (x_3 - u)_+^2 - 3(x_2 - u)_+^2 + 3(x_1 - u)_+^2 - (x_0 - u)_+^2,$$

i.e.,

$$2K(u) = \begin{cases} 0, & \text{if } u > x_3; \\ (x_3 - u)^2, & \text{if } x_2 \le u \le x_3; \\ (x_3 - u)^2 - 3(x_2 - u)^2, & \text{if } x_1 \le u \le x_2; \\ (x_3 - u)^2 - 3(x_2 - u)^2 + 3(x_1 - u)^2 \equiv (u - x_0)^2, & \text{if } x_0 \le u \le x_1; \\ (x_3 - u)^2 - 3(x_2 - u)^2 + 3(x_1 - u)^2 - (x_0 - u)^2 \equiv 0, & \text{if } u < x_0. \end{cases} \tag{3.2.28}$$

For the simplification of the last two lines we used that $\Delta_u^3 (x_0 - u)^2 \equiv 0$.

Note that $K(u)$ is a piecewise polynomial in $\mathcal{P}_3$ and that $K''(u)$ is discontinuous at $u = x_i$, $i = 0 : 3$.

It can be shown (numerically or analytically) that $K(u) > 0$ in the interval (u_0, u_3). This is no surprise, for, by (3.2.4), $\Delta^n f(x) = h^n f^{(n)}(\xi)$ for any integer n, and, by the above corollary, this could not be generally true if $K(u)$ changes its sign. These calculations can be generalized to $\Delta^k f(a)$ for an arbitrary integer k. This example will be generalized in Sec. 4.2.5 to divided differences of non-equidistant data.

In general it is rather laborious to determine a Peano kernel. Sometimes one can show that the kernel is piecewise a polynomial, that it has a symmetry, and that has a simple form in the intervals near the boundaries. All this can simplify the computation, and might have been used in these examples.

It is usually much easier to compute $R((x - a)^k)$, and an *approximate error estimate* is often given by

$$Rf \sim \frac{f^{(k)}(a)}{k!} R\big((x - a)^k\big), \quad f^{(k)}(a) \neq 0. \tag{3.2.29}$$

For example, suppose that $x \in [a, b]$, where $b - a$ is of the order of magnitude of a step size parameter h, and that f is analytic in $[a, b]$. By Taylor's formula,

$$f(x) = p(x) + \frac{f^{(k)}(a)}{k!}(x-a)^k + \frac{f^{(k+1)}(a)}{(k+1)!}(x-a)^{k+1} + \dots, \qquad f^{(k)}(a) \neq 0,$$

where $p \in \mathcal{P}_k$, hence $Rp = 0$. Most of the common functionals can be applied term by term. Then

$$Rf = 0 + \frac{f^{(k)}(a)}{n!}R_x(x-a)^k + \frac{f^{(k+1)}(a)}{(k+1)!}R_x(x-a)^{k+1} + \dots .$$

Assume that, for some c, $R_x(x-a)^k = O(h^{k+c})$, for $k = 1, 2, 3, \dots$. (This is often the case.) Then (3.2.29) becomes an **asymptotic error estimate** as $h \to 0$. It was mentioned above that for formulas derived by operator methods, an asymptotic error estimate is directly available anyway, but if a formula is derived by other means (see Chapter 4) this error estimate is important.

If $Rp = 0$ for $p \in \mathcal{P}_k$, then a fortiori $Rp = 0$ for $p \in \mathcal{P}_{k-i}$, $i = 0 : k$. We may thus obtain a Peano kernel for each i, which is temporarily denoted by $K_{k-i}(u)$. They are obtained by integration by parts,

$$R_k f = \int_{-\infty}^{\infty} K_k(u) f^{(k)}(u)du = \int_{-\infty}^{\infty} K_{k-1}(u) f^{(k-1)}(u)du = \int K_{k-2}(u) f^{(k-2)}(u)du \dots, \tag{3.2.30}$$

where $K_{k-i} = (-D)^i K_k$, $i = 1, 2, \dots$, *as long as* K_{k-i} *is integrable. The lower order kernels are useful, e.g., if the actual function* f *is not as smooth as the usual remainder formula requires.*

For the trapezoidal rule we obtain $K_1(u) = \frac{h}{2}u_+^0 + \frac{h}{2} - u + \frac{h}{2}(u-h)_+^0$.

A second integration by parts can only be performed within the framework of Dirac's delta functions (distributions); K_0 is not integrable. A reader, who is familiar with these generalized functions, may enjoy the following formula:

$$Rf = \int_{-\infty}^{\infty} K_0(u) f(u)du \equiv \int_{-\infty}^{\infty} \Big(-\frac{h}{2}\delta(u) + 1 - \frac{h}{2}\delta(u-h)\Big) f(u)du.$$

This is for one step of the trapezoidal rule, but many functionals can be expressed analogously.

3.2.4 Applications of Operator Techniques for Finding Approximation Formulas

Example 3.2.9. *Finding interpolation formulas by operator methods.*

Consider the operator expansion

$$f(b - \gamma h) = E^{-\gamma} f(b) = (1 - \nabla)^{\gamma} f(b) = \sum_{j=0}^{\infty} \binom{\gamma}{j} (-\nabla)^j f(b).$$

The verification of the assumptions of Theorem 3.2.6 offers no difficulties, and we omit the details. Truncate the expansion before $(-\nabla)^k$. By the theorem we obtain, for every γ:

• an approximation formula for $f(b-\gamma h)$ that uses the function values $f(b-jh)$ for $j = 0 : k-1$; it is exact if $f \in \mathcal{P}_k$, and is unique in the sense of Theorem 3.2.4;

• an asymptotic error estimate if $f \notin \mathcal{P}_k$, namely the first neglected term of the expansion, i.e., $\binom{\gamma}{k}(-\nabla)^k f(b) \sim \binom{\gamma}{k}(-h)^k f^{(k)}(b)$

Note that the binomial coefficients are polynomials in the variable γ, and hence also in the variable $x = b - \gamma h$. It follows that the approximation formula yields **a unique polynomial** $P_B \in \mathcal{P}_k$, that solves the **interpolation problem**: $P_B(b-hj) = f(b-hj)$, $j = 0 : k-1$; (B stands for Backward). If we set $x = b-\gamma h$, we obtain

$$\begin{aligned} P_B(x) &= E^{-\gamma} f(b) = (1-\nabla)^{\gamma} f(a) \\ &= \sum_{j=0}^{k-1} \binom{\gamma}{j} (-\nabla)^j f(b) = f(b-\gamma h) + O(h^k f^{(k)}). \end{aligned} \tag{3.2.31}$$

Due to the uniqueness, see the corollary of Theorem 3.2.4, the approximation to $f'(b)$ obtained by the first $k-1$ terms in Example 3.2.4 for $x_n = b$ is exactly the derivative $P_B'(b)$.

Similarly, the interpolation polynomial $P_F \in \mathcal{P}_k$ that uses *forward* differences based on the values of f at $a, a+h, \dots, a+(k-1)h$, reads, if we set $x = a + \theta h$,

$$P_F(x) = E^{\theta} f(a) = (1+\Delta)^{\theta} f(a) = \sum_{j=0}^{k-1} \binom{\theta}{j} \Delta^j f(a) = f(a+\theta h) + O(h^k f^{(k)}). \tag{3.2.32}$$

These formulas are known as **Newton's interpolation formulas for constant step size**, backwards and forwards. The generalization to variable step size will be found in Sec. 4.2.

There exists a similar expansion for *central differences*. Set

$$\phi_0(\theta) = 1, \quad \phi_1(\theta) = \theta, \quad \phi_j(\theta) = \frac{\theta}{j} \binom{\theta + \frac{1}{2}j - 1}{j-1}, \quad (j > 1). \tag{3.2.33}$$

ϕ_j is an even function if j is even, and an odd function if j is odd. It can be shown that $\delta^j \phi_k(\theta) = \phi_{k-j}(\theta)$, and $\delta^j \phi_k(0) = \delta_{j,k}$, (Kronecker's delta). The functions ϕ_k have thus an analogous relation to the operator δ as, e.g., the functions $\theta^j/j!$ and $\binom{\theta}{j}$ have to the operators D and Δ, respectively. We obtain the following expansion, analogous to Taylor's formula and Newton's forward interpolation formula. The proof is left for Problem 4(b). Then

$$E^{\theta} f(a) = \sum_{j=0}^{k-1} \phi_j(\theta) \delta^j f(a) = f(a+\theta h) + O(h^k f^{(k)}). \tag{3.2.34}$$

The direct practical importance of this formula is small, since $\delta^j f(a)$ cannot be expressed as a linear combination of the given data when j is odd. There are several formulas, where this drawback has been eliminated by various transformations. They were much in use before the computer age; each formula had its own group of fans. We shall derive only one of them, by a short break-neck application of the formal power series techniques. [39] Note that

$$\begin{aligned}
E^\theta &= e^{\theta hD} = \cosh\theta hD + \sinh\theta hD, \\
\delta^2 &= e^{hD} - 2 + e^{-hD}, \qquad e^{hD} - e^{-hD} = 2\mu\delta, \\
\cosh\theta hD &= \tfrac{1}{2}(E^\theta + E^{-\theta}) = \sum_{j=0}^{\infty} \phi_{2j}(\theta)\delta^{2j}, \\
\sinh\theta hD &= \frac{1}{\theta}\frac{d(\cosh\theta hD)}{d(hD)} = \sum_{j=0}^{\infty} \phi_{2j}(\theta)\frac{1}{\theta}\frac{d\delta^{2j}}{d\delta^2}\frac{d\delta^2}{d(hD)} \\
&= \sum_{j=0}^{\infty} \phi_{2j}(\theta)\frac{j\delta^{2(j-1)}}{\theta}(e^{hD} - e^{-hD}) = \sum_{j=0}^{\infty} \phi_{2j}(\theta)\frac{2j}{\theta}\mu\delta^{2j-1}.
\end{aligned}$$

Hence,[40]

$$f(x_0 + \theta h) = E^\theta f_0 = f_0 + \theta\mu\delta f_0 + \frac{\theta^2}{2!}\delta^2 f_0 + \sum_{j=2}^{\infty} \phi_{2j}(\theta)\Big(\frac{2j}{\theta}\mu\delta^{2j-1} f_0 + \delta^{2j} f_0\Big). \tag{3.2.35}$$

This is known as **Stirling's interpolation formula** [41] . Note that

$$\phi_{2j}(\theta) = \theta^2(\theta^2 - 1)(\theta^2 - 4)\cdots(\theta^2 - (j-1)^2)/(2j)!.$$

The expansion yields a true interpolation formula [42] iff it is truncated after an *even* power of δ.

Strict error bounds can be found by means of Peano's theorem, but the remainder terms given in Sec. 4.2 for Newton's general interpolation formula (that does not require equidistant data) typically give the answer easier. Both are typically of the form $c_{k+1}f^{(k+1)}(\xi)$ and require a bound for a derivative of high order. The assessment of such a bound typically costs much more work than performing interpolation in one point.

A more practical approach is to estimate a bound for this derivative by means of a bound for the differences [43] of the same order. This is not a rigorous *bound*, but it typically yields a quite reliable error *estimate*, in particular if you put a moderate

[39]Differentiation of a formal power series with respect to an indeterminate has a purely algebraic definition. See the last part of § 3.1.3.

[40]The first three terms have been taken out from the sum, in order to show their simplicity and their resemblance to Taylor's formula. They yield the most practical formula for quadratic interpolation; it is easily remembered and worth to be remembered. An approximate error bound for this quadratic interpolation reads $|0.016\delta^3 f|$ if $|\theta| < 1$.

[41]James Stirling (1692-1770), British mathematician, perhaps most famous for his amazing approximation to $n!$.

[42]For $k = 1$ you see that $f_0 + \theta\mu\delta f_0$ is not a formula for linear interpolation; it uses three data points instead of two. It is similar for all odd values of k.

[43]Recall the important formula in (3.2.4).

safety factor on the top of it. There is much more to be said about the choice of step size and order; we shall return to this kind of questions in later chapters.

You can make error estimates during the run; it can happen sooner ot later that it does not decrease, when you increase the order. You may just as well stop there, and accept the most recent value as the result. This event is most likely due to the influence of irregular errors, e.g. rounding errors, but it can also indicate that the interpolation process is semi-convergent only.

The attainable accuracy of polynomial interpolation applied to a table with n equidistant values of an analytic function, depends strongly on θ; the results are much poorer near the boundaries of the data set than near the center. This question will be illuminated in Sec. 4.4 by means of complex analysis.

More information about the classical methods for polynomial interpolation of equidistant data is found in, e.g., Fröberg [12] and Steffensen [31], in particular § 18 about "the calculus of symbols". For the history of these matters see, e.g., Goldstine [14].

Example 3.2.10. *Continuation of the difference scheme of a polynomial.* The following is a classical application of a difference scheme for obtaining a smooth extrapolation of a function outside its original domain. Given the values $y_{n-i} = f(x_n - ih)$ for $i = 1 : k$ and the backward differences, $\nabla^j y_{n-1}$, $j = 1 : k-1$. Recall that $\nabla^{k-1} y$ is a constant for $y \in \mathcal{P}_k$. Consider the algorithm

$$\begin{aligned} &\nabla^{k-1} y_n = \nabla^{k-1} y_{n-1}; \\ &\quad \textbf{for } j = k-1 : -1 : 1,\ \nabla^{j-1} y_n = \nabla^{j-1} y_{n-1} + \nabla^j y_n;\ \textbf{end} \\ &\quad y_n = \nabla^0 y_n; \end{aligned} \tag{3.2.36}$$

It is left for Problem 7a to show that the result y_n is the value at $x = x_n$ of the interpolation polynomial which is determined by y_{n-i}, $i = 1 : k$. This is a kind of inverse use of a difference scheme; there are additions from right to left along a diagonal, instead of subtractions from left to right.

This algorithm, which needs additions only, was, in principle, used long ago for the production of mathematical tables, e.g., for logarithms. Suppose that one knows, e.g., by means of a series expansion, a relatively complicated polynomial approximation to (say) $f(x) = \ln x$, that is accurate enough in (say) the interval $[a, b]$, and that this has been used for the computation of k very accurate values $y_0 = f(a)$, $y_1 = f(a+h), \ldots y_{k-1}$, needed for starting the difference scheme. The algorithm is then used for $n = k,\ k+1,\ k+2, \ldots,\ (b-a)/h$. $k-1$ additions only are needed for each value y_n. Some analysis must have been needed for the choice of the step h to make the tables useful with (say) linear interpolation, and for the choice of k to make the basic polynomial approximation accurate enough over a substantial number of steps. The precision used was higher, when the table was produced than when it was used. When $x = b$ was reached, a new approximating polynomial was needed for continuing the computation over an other interval; at least a new value of $\nabla^{k-1} y_n$.

This procedure was the basis of the unfinished Difference Engine project of the great 19th century British computer pioneer Charles Babbage. He abandoned it after a while in order to spend more time on his huge “Analytic Engine” project, which was also unfinished, but he documented a lot of ideas, where he was (say) 100 years ahead of his time.“Difference engines” based on Babbage’s ideas were, however, constructed in Babbage’s own time, by the Swedish inventors Scheutz (father and son) 1834 and by Wiberg 1876, and they were applied, among other things, to the automatic calculation and printing of tables of logarithms. See, e.g., Goldstine [14].

A generalization of the algorithm in (3.2.36) to the non-equidistant case will be presented in Ch. 4, with the use of the notion of *scaled divided differences*. In Sec. 13.4, both the equidistant and the non-equidistant case will be applied to the implementation of multistep methods for the numerical solution of differential equations.

Example 3.2.11. *Central difference formulas for numerical differentiation.*

From the definition and from Bickley’s table, i.e., Table 3.2.1,

$$\delta \equiv E^{1/2} - E^{-1/2} = 2\sinh\Big(\frac{1}{2}hD\Big). \tag{3.2.37}$$

We may therefore put $x = \frac{1}{2}hD$, $\sinh x = \frac{1}{2}\delta$ into the following expansion (see Problem 3.1.7),

$$x = \sinh x - \frac{1}{2}\frac{\sinh^3 x}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{\sinh^5 x}{5} - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{\sinh^7 x}{7} \pm \dots,$$

with the result

$$hD = 2\text{arcsinh}\frac{\delta}{2} = \delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} - \frac{5\delta^7}{7,168} + \frac{35\delta^9}{294,912} - \frac{63\delta^{11}}{2,883,584} \pm \dots. \tag{3.2.38}$$

The verification of the assumptions of Theorem 3.2.6 follows the pattern of Example 3.2.6, and we omit the details. Since $\text{arcsinh}z$, $z \in \mathbf{C}$ has the same singularities as its derivative $(1+z^2)^{-1/2}$, namely $z = \pm i$, it follows that the expansion in (3.2.38), if $\text{sc}(\delta/2)$ is substituted for $\delta/2$, converges if $\text{sc}(\delta/2) < 1$, hence $\rho = 2$.

By squaring the above relation, we obtain

$$(hD)^2 = \delta^2 - \frac{\delta^4}{12} + \frac{\delta^6}{90} - \frac{\delta^8}{560} + \frac{\delta^{10}}{3,150} - \frac{\delta^{12}}{16,632} \pm \dots,$$

$$f''(x_0) \approx \left(1 - \frac{\delta^2}{12} + \frac{\delta^4}{90} - \frac{\delta^6}{560} + \frac{\delta^8}{3,150} - \frac{\delta^{10}}{16,632} \pm \dots\right)\frac{\delta^2 f_0}{h^2}. \tag{3.2.39}$$

By Theorem 3.2.6 Eq. (3.2.39) holds for all polynomials. Since the first neglected non-vanishing term of (3.2.39) (when it is applied to f) is (asymptotically) $c\delta^{12}f''(x_0)$, the formula for $f''(x)$ is exact if $f'' \in \mathcal{P}_{12}$, i.e. if $f \in \mathcal{P}_{14}$, although only 13 values of $f(x)$ are used. [44] We thus gain one degree and, in the application to other functions than polynomials, one order of accuracy, compared to what we

[44]Recall that $\mathcal{P}_{14}$ is the space of polynomials of degree *less than* 10,

may have expected by counting unknowns and equations only, see Theorem 3.2.4. *This is typical for a problem that has a symmetry with respect to the hull of the data points.*

Suppose that the values $f(x)$ are given on the grid $x = x_0 + nh$, n integer. Since Eq. (3.2.38) contains odd powers of δ, it cannot be used to compute f'_n on the same grid. [45] This difficulty can, however, be overcome by means of a formula given in Bickley's table, namely

$$\mu = \sqrt{1 + \frac{1}{4}\delta^2}. \tag{3.2.40}$$

This is derived as follows. The formulas $\mu = \cosh \frac{hD}{2}$, $\frac{\delta}{2} = \sinh \frac{hD}{2}$ follow rather directly from the definitions; the details are left for Problem 5a. The formula $(\cosh hD)^2 - (\sinh hD)^2 = 1$ holds also for formal power series. Hence

$$\mu^2 - \frac{1}{4}\delta^2 = 1, \quad \text{or} \quad \mu^2 = 1 + \frac{1}{4}\delta^2,$$

from which the relation (3.2.40) follows.

Now multiply the right hand side of equation (3.2.38) by the expansion

$$1 = \mu\Big(1 + \frac{1}{4}\delta^2\Big)^{-1/2} = \mu\Big(1 - \frac{\delta^2}{8} + \frac{3\delta^4}{128} - \frac{5\delta^6}{1,024} + \frac{35\delta^8}{32,768} \pm \ldots\Big). \tag{3.2.41}$$

By (3.2.41) and (3.2.38),

$$\begin{aligned} hD &= \mu\Big(1 - \frac{\delta^2}{8} + \frac{3\delta^4}{128} + \ldots\Big)\Big(\delta - \frac{\delta^3}{24} + \frac{3\delta^5}{640} + \ldots\Big) \\ &= \Big(1 - \frac{\delta^2}{6} + \frac{\delta^4}{30} - \frac{\delta^6}{140} \pm \ldots\Big)\mu\delta. \end{aligned} \tag{3.2.42}$$

This leads to a useful central difference formula for the first derivative (where we have used more terms than we displayed in the above derivation).

$$f'(x_0) = \Big(1 - \frac{\delta^2}{6} + \frac{\delta^4}{30} - \frac{\delta^6}{140} + \frac{\delta^8}{630} - \frac{\delta^{10}}{2772} + \frac{\delta^{12}}{12012} \pm \ldots\Big)\frac{f_1 - f_{-1}}{2h}. \tag{3.2.43}$$

If you truncate the operator expansion in (3.2.43) after the δ^{2k} term, you obtain exactly the derivative of the interpolation polynomial of degree $2k+1$ for $f(x)$ that is determined by the $2k+2$ values f_i, $i = \pm 1, \pm 2, \ldots, \pm(k+1)$. Note that all the neglected terms in the expansion vanish when $f(x)$ is any polynomial of degree $2k+2$, independent of the value of f_0. [46] So, although we search for a formula that is exact in $\mathcal{P}_{2k+2}$, we actually find a formula that is exact in $\mathcal{P}_{2k+3}$.

By the multiplication of the expansions in (3.2.39) and (3.2.42), we obtain the following formulas, which have applications in other sections

$$(hD)^3 = (1 - \frac{1}{4}\delta^2 + \frac{7}{120}\delta^4 + \ldots)\mu\delta^3 \tag{3.2.44}$$

[45]This was pointed out in the beginning of § 3.2.2.

[46]Check the statements first for $k = 0$; you will recognize a familiar property of the parabola.

$$(hD)^5 = (1 - \frac{1}{3}\delta^2 + \ldots)\mu\delta^5$$
$$(hD)^7 = \mu\delta^7 + \ldots$$

Another valuable feature typical for δ^2*-expansions*, i.e., for expansions in powers of δ^2, is the rapid convergence. It was mentioned earlier that $\rho = 2$, hence $\rho^2 = 4$, (while $\rho = 1$ for the backwards differentiation formula). The error constants of the differentiation formulas obtained by (3.2.39) and (3.2.43) are thus relatively small.

All this is typical for the symmetric approximation formulas which are based on central differences, see, e.g., the above formula for $f''(x_0)$, or the next example. In view of this, can we forget the forward and backward difference formulas altogether? Well, this is not quite the case, since one must often deal with data that are unsymmetric with respect to the point where the result is needed. For example, given f_{-1}, f_0, f_1, how would you compute $f'(x_1)$? Asymmetry is also typical for the application to *initial value problems* for differential equations, see Sec. 13.4 and Ch. 14; in such applications methods based on symmetric rules for differentiation or integration have sometimes inferior properties of numerical stability.

When a problem has a symmetry around some point x_0, you are advised to try to derive a δ^2-expansion. The first step is to express the relevant operator in the form $\Phi(\delta^2)$, where the function Φ is analytic at the origin.

To find a δ^2-expansion for $\Phi(\delta^2)$ is algebraically the same thing as expanding $\Phi(z)$ into powers of a complex variable z. So, the methods for the manipulation of power series mentioned in Sec. 3.1.3 and Prob 3.1.8 are available, and so is the Cauchy+FFT method (Sec. 3.1.4). *For suitably chosen r, N you evaluate $\Phi(re^{2\pi ik/N})$, $k = 0 : N-1$, and obtain the coefficients of the δ^2-expansion by the FFT!* You can therefore derive a long expansion, and later truncate it as needed. [47]

Suppose that you have found a truncated δ^2-expansion, (say) $A(\delta^2) \equiv a_1 + a_2\delta^2 + a_3\delta^4 + \ldots + a_{k+1}\delta^{2k}$, but you want instead an equivalent symmetric expression of the form $B(E) \equiv b_1 + b_2(E + E^{-1}) + b_3(E^2 + E^{-2}) + \ldots + b_{k+1}(E^k + E^{-k})$. Note that $\delta^2 = E - 2 + E^{-1}$. The transformation $A(\delta^2) \mapsto B(E)$ can be performed in several ways. Since it is linear it can be expressed by a matrix multiplication of the form $b = M_{k+1}a$, where a, b are column vectors for the coefficients, and M_{k+1} is the $k+1 \times k+1$ submatrix in the northwest corner of a matrix M that turns out to be

$$M = \begin{pmatrix} 1 & -2 & 6 & -20 & 70 & -252 & 924 & -3432 \\ 0 & 1 & -4 & 15 & -56 & 210 & -792 & 3003 \\ 0 & 0 & 1 & -6 & 28 & -120 & 495 & -2002 \\ 0 & 0 & 0 & 1 & -8 & 45 & -220 & 1001 \\ 0 & 0 & 0 & 0 & 1 & -10 & 66 & -364 \\ 0 & 0 & 0 & 0 & 0 & 1 & -12 & 91 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (3.2.45)$$

Note that the matrix elements are binomial coefficients that can be generated

[47]You also obtain error estimates for all these truncated expansions for free.

recursively (Problem 13). It is therefore easy to extend the matrix; this 8×8 matrix is sufficient for a δ^2-expansion up to the term $a_8\delta^{14}$.

The operator D^{-1} is defined by the relation $(D^{-1}f)(x) = \int^x f(t)\,dt$. The lower limit is not fixed, so $D^{-1}f$ contains an arbitrary integration constant. Note that $DD^{-1}f = f$, while $D^{-1}Df = f + C$, where C is the integration constant. A difference expression like $D^{-1}f(b) - D^{-1}f(a) = \int_a^b f(t)\,dt$ is uniquely defined. So is also $\delta D^{-1}f$, but $D^{-1}\delta f$ has an integration constant.

A right-hand inverse can be defined also for the operators Δ, ∇, δ. For example, $(\nabla^{-1}u)_n = \sum^{j=n} u_j$ has an arbitrary summation constant but, e.g., $\nabla\nabla^{-1} = 1$, and $\Delta\nabla^{-1} = E\nabla\nabla^{-1} = E$ are uniquely defined.

One can make the inverses unique by restricting the class of sequences (or functions). For example, if we require that $\sum_{j=0}^{\infty} u_j$ is convergent, and make the convention that $(\Delta^{-1}u)_n \to 0$ as $n \to \infty$, then $\Delta^{-1}u_n = -\sum_{j=n}^{\infty} u_j$; notice the minus sign. Also notice that this is consistent with the following formal computation: $(1 + E + E^2 + \ldots)u_n = (1 - E)^{-1}u_n = -\Delta^{-1}u_n$. We recommend, however, some extra care with infinite expansions into powers of operators like E that is not covered by Theorem 3.2.6, but the finite expansion

$$1 + E + E^2 + \ldots + E^{n-1} = (E^n - 1)(E - 1)^{-1} \tag{3.2.46}$$

is valid.

Example 3.2.12. *Finding the symmetric integration formulas of Newton–Cotes by operator methods.* Let m, n be given integers. let h be a positive step size, set $H = mh$. H is called the "bigstep". **The Newton–Cotes formulas** [48] for numerical integration are usually written as follows.

$$\int_0^{mh} f(x)\,dx = h\sum_{i=0}^{n} w_i f(ih) + R_{m,n}(h), \qquad (w_{n-i} = w_i).$$

In order to utilize the symmetry of the problem easier, we move the origin to the midpoint of the interval of integration. Set $j = -n/2 : 1 : n/2$, $x_j = jh$, $f_j = f(jh)$. The same formula now reads

$$\int_{-mh/2}^{mh/2} f(x)\,dx = h \sum_{j=-n/2}^{n/2} w_j f_j + R_{m,n}(h), \qquad w_{-j} = w_j. \tag{3.2.47}$$

j, $n/2$ and $m/2$ are not necessarily integers. For a Newton–Cotes formula $n/2 - j$ and $m/2 - j$ are evidently integers, hence $(m - n)/2$ is an integer too, but there may be other formulas, perhaps almost as good, where this is not the case. The coefficients $w_j = w_{j;m,n}$ are to be determined so that the remainder $R_{m,n}$ vanishes if $f \in \mathcal{P}_q$, with q as large as possible for given m, n.

The left hand side, divided by h reads, in operator form,

$$(e^{hDm/2} - e^{-hDm/2})(hD)^{-1}f(x_0),$$

[48] Roger Cotes (1682-1716) was an English mathematician. He was a highly appreciated young colleague of Isaac Newton (1643-1727).

which is an even function of hD. By (3.2.38), hD is an odd function of δ. It follows that the left hand side is an even function of δ, hence we can, for every m, write

$$(e^{hDm/2} - e^{-hDm/2})(hD)^{-1} \quad \mapsto A_m(\delta^2) = a_{1m} + a_{2m}\delta^2 + a_{3m}\delta^4 + \ldots + a_{i+1,m}\delta^{2i} \ldots \tag{3.2.48}$$

The coefficients are computed by means of the Cauchy+FFT method. Figure 3.2.12 was obtained for $N = 32$, $r = 2$. the absolute errors of the coefficients [49] are then less than 10^{-13}.

We truncate after (say) δ^{2k}; the first neglected term is then $a_{k+2}\delta^{2k+2}$. We saw above how to bring a truncated δ^2-expansion to $B(E)$-form, $b_1 + b_2(E + E^{-1}) + b_3(E^2 + E^{-2}) + \ldots + b_k(E^k + E^{-k})$. By comparison with (3.2.47), we conclude that $n/2 = k$, that the indices j are integers, and that $w_j = b_{j+1}$ (if $j \geq 0$). *If m is even, this becomes a Newton–Cotes formula.* If m is odd, it may still be a useful formula, but it does not belong to the Newton–Cotes family, because $(m - n)/2 = m/2 - k$ is no integer.

If $n = m$ a formula is said to be of **the closed type**; the function values on the boundary of the interval of integration are used. Its remainder term is the first neglected term of the operator series, truncated after δ^{2k}, $2k = n = m$ (and multiplied by h), hence the remainder of (3.2.47) is estimated by $a_{2+m/2}\delta^{m+2}f_0$, or (better) $R_{m,m} \sim (a_{2+m/2}/m)H(hD)^{m+2}f_0$, where $H = mh$. [50]

If $n < m$ a formula is said to be of **the open type**. Among the open formulas we shall only consider the case that $n = m - 2$; this is called "the" open Newton–Cotes formula. The operator expansion is truncated after δ^{m-2} and we obtain $R_{m-2,m} \sim (a_{1+m/2}/m)H(hD)^m f_0$. [51]

We shall need the case $m = 8$ in § 3.3.5. One result of our computations reads,

$$A_8(\delta^2) = 8 + \frac{64}{3}\delta^2 + \frac{688}{45}\delta^4 + \frac{736}{189}\delta^6 + \frac{3956}{14175}\delta^8 - \frac{2368}{467775}\delta^{10} + \ldots \tag{3.2.49}$$

The closed integration formula becomes (Problem 13)

$$\int_{-x_4}^{x_4} f(x)dx = \frac{4h}{14175}\Big(-4540f_0 + 10496(f_1 + f_{-1}) - 928(f_2 + f_{-2}) + 5888(f_3 + f_{-3)}) + 989(f_4 + f_{-4})\Big) + R, \tag{3.2.50}$$

$$R \sim \frac{296}{467775}Hh^{10}f^{(10)}(x_0). \tag{3.2.51}$$

The coefficients are first obtained in floating point representation. The transformation to rational form is obtained by a continued fraction algorithm, described

[49] See Lemma 3.1.2 about the error estimation.

[50] If the integral is computed over $[a, b]$ by means of a sequence of "bigsteps", each of length H, an estimate of the *global* error has the same form, except that H is replaced by $b - a$, and f_0 is replaced by $\max_{x\in[a,b]} |f(x)|$. The exponent of hD in an error estimate that contains H or $b - a$, is known as the *global order of accuracy* of the method, see Ch.5.

[51] Formulas with $n > m$ are rarely mentioned in the literature (except for $m = 1$). We do not understand why; it is rather common that an integrand has a smooth continuation outside the interval of integration. We shall take up this question in Ch 5.

Figure 3.2.1. *The coefficients $|a_j|$ of the δ^2-expansions for $m = 2:14$.*

in Example 3.4.1. [52]

It was mentioned that, *if m is odd*, Eq. (3.2.48) does not provide formulas of the Newton–Cotes family, since $(m-n)/2$ is no integer, nor are the indices j in (3.2.47) integers. So, the operator associated with the right hand side of (3.2.47) is of the form $c_1(E^{1/2}+E^{-1/2})+c_2(E^{3/2}+E^{-3/2})+c_3(E^{5/2}+E^{-5/2})+\ldots$. If it is divided algebraically by $\mu = \frac{1}{2}(E^{1/2}+E^{-1/2})$, however, it becomes of the $B(E)$-form (say)

$$b_1' + b_2'(E+E^{-1}) + b_3'(E^2+E^{-2}) + \ldots + b_k(E^k+E^{-k}).$$

If m is odd we therefore expand

$$(e^{hDm/2} - e^{-hDm/2})(hD)^{-1}/\mu, \qquad \mu = \sqrt{1+\delta^2/4},$$

into a δ^2-series, with coefficients a_j', again numerically by the Cauchy+FFT method.

For each m two truncated δ^2-series, one for the closed an one for the open case, are then transformed into $B(E)$-expressions numerically by means of the matrix M, as described above. The expressions are then multiplied *algebraically* by $\mu = \frac{1}{2}(E^{1/2}+E^{-1/2})$. We then have the coefficients of a Newton–Cotes formula with m odd. This simple algebraic calculation is easily programmed in a language for floating-point computation. Otherwise we have succeeded in avoiding extensive algebraic calculations by using the numerical Cauchy+FFT method.

The asymptotic error is

$$a'_{m/2+1}H(hD)^{m+1}) \quad \text{and} \quad a'_{m/2-1}H(hD)^{m-1})$$

for the closed type, and open type, respectively $(2k = m-1)$. The global orders of accuracy for Newton–Cotes methods with odd m are thus the same as for the methods, where m is one less.

[52]It can be shown that the exact coefficients are rational numbers, though it is sometimes hard to estimate in advance the order of magnitude of the denominators. The algorithm must be used with judgment. We checked our results with formulas in the Handbook [1], Sec. 25.4.

It goes without saying that this is not how Newton and Cotes found their methods. Our method may seem complicated, but the Matlab programs for this are rather short, and to a large extent useful for other purpose. The whole computation of about 150 Cotes-coefficients and 25 remainders ($m = 2 : 14$), including graphical output to the screen took less than two seconds on a PC. [53] In Chapters 4 and 5 we shall look at the Newton–Cotes formulas from other points of view. There are some tables of coefficients and remainder terms in Ch. 5.

Operator techniques can also be extended to *functions of several variables.* The basic relation is again the operator form of Taylor's formula, which in the case of two variables reads,

$$\begin{aligned} u(x_0 + h, y_0 + k) &= \exp\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) u(x_0, y_0) \\ &= \exp\left(h\frac{\partial}{\partial x}\right) \exp\left(k\frac{\partial}{\partial y}\right) u(x_0, y_0). \end{aligned} \tag{3.2.52}$$

There will be applications in some problems of Ch. 4.

3.2.5 Single Linear Difference Equations

Historically, the term difference equation was probably first used in connection with an equation of the form

$$b_0\Delta^k y_n + b_1\Delta^{k-1} y_n + \dots b_{k-1}\Delta y_n + b_k y_n = 0, \quad n = 0, 1, 2, \dots$$

which reminds of a linear homogeneous differential equation. It follows, however, from the discussion after (3.2.1) and (3.2.2) that this equation can also be written in the form

$$y_{n+k} + a_1 y_{n+k-1} + \dots + a_k y_n = 0, \tag{3.2.53}$$

and nowadays this is what one usually means by a **single homogeneous linear difference equation** of kth order with *constant coefficients*; a difference equation without differences. More generally, if we let the coefficients a_i depend on n; we have a linear difference equation with *variable coefficients.* If we replace the zero on the right hand side with some known quantity r_n, we have a *nonhomogeneous* linear difference equation.

These types of equations are the main topic of this subsection. The coefficients and the unknown are real or complex numbers. We shall occasionally see examples of more general types of difference equations, e.g., a nonlinear difference equation $F(y_{n+k}, y_{n+k-1}, \dots, y_n) = 0$), and we shall, in Sec 13.9, deal with *first order systems* of difference equations, i.e. $y_{n+1} = A_n y_n + r_n$, where r_n, y_n, etc. are vectors while A_n is a square matrix. Finally, *partial difference equations* where you have two

[53]This includes the calculation of several alternatives for rational approximations to the floating-point results. For a small number of the 150 coefficients the judicious choice among the alternatives took, however, much more than 2 (human) seconds; this detail is both science and art.

(or more) subscripts in the unknown, occur often as numerical methods for partial differential equations, but they have many other important applications too.

A difference equation can be viewed as a *recurrence relation.* With given values of $y_0,\ y_1, \dots,\ y_{k-1}$, called the **initial values** or the **seed** of the recurrence, we can successively compute $y_k,\ y_{k+1},\ y_{k+2}, \dots$; we see that *the general solution of a k'th order difference equation contains k arbitrary constants*, just like the general solution of the k'th order differential equation.

There are other important similarities between difference and differential equations, for example the following superposition result.

Lemma 3.2.8. *The general solution of a nonhomogeneous linear difference equation (also with variable coefficients) is the sum of one particular solution of it, and the general solution of the corresponding homogeneous difference equation.*

In practical computing, the recursive computation of the solution of a difference equations is most common. It was mentioned at the end of Sec. 3.1 that many important functions, e.g., Bessel functions and orthogonal polynomials, satisfy second order linear difference equations with variable coefficients, (although this terminology was not used there). Other important applications are the multistep methods for ordinary differential equations.

In such an application you are usually interested in the solution for one particular initial condition, but due to rounding errors in the initial values you obtain another solution. It is therefore of interest to know the behaviour of the solutions of the corresponding homogeneous difference equation. The questions are: • *Can* we use a recurrence to find the wanted solution accurately?
• *How* shall we use a recurrence, **forward** or **backward**?

Forward recurrence is the type we described above. In backward recurrence we choose some large integer N, and give (almost) arbitrary values of $y_{N+i},\ i = 0 : k-1$ as seed, and compute y_n for $n = N-1 : -1 : 0$.

We have seen this already in Example 1.3.3 (and in Problem 10a of Sec. 1.3) for an inhomogeneous first order recurrence relation. It was there found that the forward recurrence was useless, while backward recurrence, with a rather naturally chosen seed, gave satisfactory results; (see Example 1.3.4 and Problem 10b).

It is often like this, though not always. In Problem 9 of Sec. 1.3 it is the other way around: the forward recurrence is useful, and the backward recurrence is useless.

Sometimes **boundary values** are prescribed for a difference equation instead of initial values, (say) p values at the beginning and $q = k - p$ values at the end, e.g., the values of $y_0,\ y_1, \dots, y_{p-1}, \dots,\ y_{N-q}, \dots, y_{N-1},\ y_N$ are given. Then the difference equation can be treated as a *linear system* with $N-k$ unknown. This also holds for a difference equation with variable coefficients and for an inhomogeneous difference equation. *From the point of view of numerical stability, such a treatment can be better than either recurrence.* The amount of work is somewhat larger, not very much though, for the matrix is of a special type. It is, for obvious reasons, called a **band matrix**. We shall see in Ch.6 that *the amount of work is then only proportional to the number of unknown.* An important particular case has

$k = 2$, $p = q = 1$; the matrix and the linear system are then called **tridiagonal**, again for obvious reasons. An algorithm is described in § 4.6.3.

Another similarity for differential and difference equations, is that the general solution of a linear equation with constant coefficients has a simple closed form. Although, in most cases, the real world problems have variable coefficients (or are nonlinear), one can often formulate a class of model problems with constant coefficients, with similar features. The analysis of such model problems can give hints, e.g., whether forward or backward recurrence should be used, or other questions related to the design and the analysis of the numerical stability of a numerical method for a more complicated problem.

We shall therefore now study how to solve a *single homogeneous linear difference equation with constant coefficients* (3.2.53), i.e.,

$$y_{n+k} + a_1 y_{n+k-1} + \ldots + a_k y_n = 0.$$

It is satisfied by the sequence $\{y_j\}$, where $y_j = cu^j$, $(u \neq 0,\ c \neq 0)$, if and only if $u^{n+k} + a_1 u^{n+k-1} + \ldots + a_k u^n = 0$, that is when

$$\phi(u) \equiv u^k + a_1 u^{k-1} + \ldots + a_k = 0. \tag{3.2.54}$$

Equation (3.2.54) is called the **characteristic equation** of (3.2.53); $\phi(u)$ is called the **characteristic polynomial**.

Theorem 3.2.9.

If the characteristic equation has k different *roots, $u_1, \ldots, u_k$, then the general solution of equation* (3.2.53) *is given by the sequences $\{y_n\}$, where*

$$y_n = c_1 u_1^n + c_2 u_2^n + \cdots + c_k u_k^n, \tag{3.2.55}$$

where $c_1, c_2, \ldots, c_k$ are arbitrary constants.

Proof. That $\{y_n\}$ satisfies equation (3.2.53) follows from the previous comments and from the fact that the equation is linear. The parameters $c_1, c_2, \ldots, c_k$ can be adjusted to arbitrary initial conditions $y_0, y_1, \ldots, y_{k-1}$ by solving the system of equations

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ u_1 & u_2 & \cdots & u_k \\ \vdots & \vdots & & \vdots \\ u_1^{k-1} & u_2^{k-1} & \cdots & u_k^{k-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{k-1} \end{pmatrix}.$$

The matrix is a Vandermonde matrix and its determinant is thus equal to the product of all differences $(u_i - u_j)$, $i \geq j$, $1 < i \leq k$, which is nonzero. ☐

Example 3.2.13. Consider the difference equation $y_{n+2} - 5y_{n+1} + 6y_n = 0$ with initial conditions $y_0 = 0$, $y_1 = 1$. Forward recurrence yields $y_2 = 5$, $y_3 = 19$, $y_4 = 65, \ldots$.

The characteristic equation $u^2 - 5u + 6 = 0$ has roots $u_1 = 3$, $u_2 = 2$. Hence, the general solution is $y_n = c_1 3^n + c_2 2^n$. The initial conditions give the system of equations

$$\begin{aligned} c_1 + c_2 &= 0, \\ 3c_1 + 2c_2 &= 1, \end{aligned}$$

with solution $c_1 = 1$, $c_2 = -1$, hence $y_n = 3^n - 2^n$.

As a check we find $y_2 = 5$, $y_3 = 19$ in agreement with the results found by using forward recurrence.

Example 3.2.14.

Consider the difference equation

$$T_{n+1}(x) - 2xT_n(x) + T_{n-1}(x) = 0, \qquad n \ge 1, \qquad -1 < x < 1,$$

with initial conditions $T_0(x) = 1$, $T_1(x) = x$. We obtain $T_2(x) = 2x^2 - 1$, $T_3(x) = 4x^3 - 3x$, $T_4(x) = 8x^4 - 8x^2 + 1, \dots$. By induction, $T_n(x)$ is an nth degree polynomial in x.

We can obtain a simple formula for $T_n(x)$ by solving the difference equation. The characteristic equation is $u^2 - 2xu + 1 = 0$, with roots $u = x \pm i\sqrt{1-x^2}$. Set $x = \cos\phi$, $0 < x < \pi$. Then $u = \cos\phi \pm i \sin\phi$, and thus

$$u_1 = e^{i\phi}, \qquad u_2 = e^{-i\phi}, \qquad u_1 \ne u_2.$$

The general solution is $T_n(x) = c_1 e^{in\phi} + c_2 e^{-in\phi}$, and the initial conditions give

$$\begin{aligned} c_1 + c_2 &= 1, \\ c_1 e^{i\phi} + c_2 e^{-i\phi} &= \cos\phi, \end{aligned}$$

with solution $c_1 = c_2 = 1/2$. Hence, $T_n(x) = \cos(n\phi)$, $x = \cos\phi$.

These polynomials are thus identical to the important Chebyshev polynomials (of the first kind) that were introduced in (3.1.32), and were there in fact denoted by $T_n(x)$.

We excluded the cases $x = 1$ and $x = -1$, i.e., $\phi = 0$ and $\phi = \pi$, respectively. For the particular initial values of this example, there are no difficulties; the solution $T_n(x) = \cos n\phi$ depends continuously on ϕ, and as $\phi \to 0$ or $phi \to \pi$, $T_n(x) = \cos n\phi$ converges to 1 $\forall n$ or $(-1)^n$ $\forall n$, respectively.

When we ask for the general solution of the difference equation, the matters are a little more complicated, because the characteristic equation has in these cases a double root; $u = 1$ for $x = 1$, $u = -1$ for $x = -1$. Although they are thus covered by the next theorem, we shall look at them directly, because they are easy to solve, and they give a good preparation for the general case.

If $x = 1$, the difference equation reads $T_{n+1} - 2T_n + T_{n-1} = 0$, i.e., $\Delta^2 T_n = 0$. We know from before [54] that this is satisfied iff $T_n = an + b$. The solution is no longer built up by exponentials; a linear term is there too.

[54] See, e.g., Theorem 3.2.4.

If $x = -1$, the difference equation reads $T_{n+1} + 2T_n + T_{n-1} = 0$. Set $T_n = (-1)^n V_n$. The difference equation becomes, after division by $(-1)^{n+1}$, $V_{n+1} - 2V_n + V_{n-1} = 0$, with the general solution, $V_n = an + b$, hence $T_n = (-1)^n(an + b)$.

Theorem 3.2.10.

When u_i is an m_i-fold root of the characteristic equation, then the difference equation (12.3.3) *is satisfied by the sequence $\{y_n\}$, where*

$$y_n = P_i(n)u_i^n,$$

and P_i is an arbitrary polynomial in $\mathcal{P}_{m_i}$. The general solution of the difference equation is a linear combination of solutions of this form using all the distinct roots of the characteristic equation.

Proof. We can write the polynomial $P \in \mathcal{P}_{m_i}$ in the form

$$P_i(n) = b_1 + b_2 n + b_3 n(n-1) + \cdots + b_{m_i} n(n-1)\cdots(n - m_i + 2).$$

Thus it is sufficient to show that equation (3.2.53) is satisfied when

$$y_n = n(n-1)\cdots(n-p+1)u_i^n = (u^p \partial^p (u^n)/\partial u^p)_{u=u_i}, \qquad p = 1, 2, \ldots, m_i - 1. \tag{3.2.56}$$

Substitute this in the left-hand side of equation (3.2.53):

$$\begin{aligned} u^p \frac{\partial^p}{\partial u^p}\Big(u^{n+k} + a_1 u^{n+k-1} + \cdots + a_k u^n\Big) &= u^p \frac{\partial^p}{\partial u^p}(\phi(u)u^n) \\ = u^p \Big(\phi^{(p)}(u)u^n + \binom{p}{1}\phi^{(p-1)}(u) n u^{n-1} + \cdots &+ \binom{p}{p}\phi(u)\frac{\partial^p}{\partial u^p}(u^n)\Big). \end{aligned}$$

The last manipulation was made using Leibniz's rule.

Now ϕ and all the derivatives of ϕ which occur in the above expression are 0 for $u = u_i$, since u_i is an m_i-fold root. Thus the sequences $\{y_n\}$ in equation (3.2.56) satisfy the difference equation. We obtain a solution with $\sum m_i = k$ parameters by the linear combination of such solutions derived from the different roots of the characteristic equation.

It can be shown (see, e.g., Henrici [18, p. 214]) that these solutions are linearly independent. [55] $\square$

Note that the double root cases discussed in the previous example are completely in accordance with this theorem. We take one more example.

Example 3.2.15.

Consider the difference equation $y_{n+3} - 3y_{n+2} + 4y_n = 0$. The characteristic equation is $u^3 - 3u^2 + 4 = 0$ with roots $u_1 = -1$, $u_2 = u_3 = 2$. Hence, the general

[55] This also follows from a different proof given in Sec. 13.8, where a difference equation of higher order is transformed to a system of first order difference equations. This transformation also leads to other ways of handling inhomogeneous difference equations than those which are presented in this subsection.

solution reads

$$y_n = c_1(-1)^n + (c_2 + c_3 n)2^n.$$

For a **nonhomogeneous** linear difference equation of order k, one can often find a *particular solution* by the use of an *"Ansatz"* with undetermined coefficients; thereafter, by Lemma 3.2.8 one can get the general solution by adding the general solution of the homogeneous difference equation.

Example 3.2.16.
Consider the difference equation $y_{n+1} - 2y_n = a^n$, with initial condition $y_0 = 1$. Try the "Ansatz" $y_n = ca^n$. One gets $ca^{n+1} - 2ca^n = a^n$, $c = 1/(a-2)$, $a \neq 2$. Thus the general solution is $y_n = a^n/(a-2) + c_1 2^n$. By the initial condition, $c_1 = 1 - 1/(a-2)$, hence

$$y_n = \frac{a^n - 2^n}{a-2} + 2^n. \tag{3.2.57}$$

When $a \to 2$, l'Hospital's rule gives $y_n = 2^n + n2^{n-1}$. Notice how the "Ansatz" must be modified when a is a root of the characteristic equation.

The general rule when the right hand side is of the form $P(n)a^n$ (or a sum of such terms), where P is a polynomial, is that the contribution of this term to y_n is $Q(n)a^n$, where Q is a polynomial. If a does not satisfy the characteristic equation then $\deg Q = \deg P$; if a is a single or a double root of the characteristic equation, then $\deg Q = \deg P + 1$ or $\deg Q = \deg P + 2$, respectively, etc. The coefficients of Q are determined by the insertion of $y_n = Q(n)a^n$ on the left hand side of the equation and matching the coefficients with the right hand side.

Another way to find a particular solution is based on the calculus of operators. Suppose that an inhomogeneous difference equation is given in the form $\psi(Q)y_n = b_n$, where Q is one of the operators Δ, δ and ∇, or an operator easily derived from these, e.g., $\frac{1}{6}\delta^2$, see Problem 27.

In § 3.2.2 $\psi(Q)^{-1}$ was defined by the formal power series with the same coefficients as the Maclaurin series for the function $1/\psi(z)$, $z \in \mathbf{C}$, $\psi(0) \neq 0$. In simple cases, e.g., if $\psi(Q) = a_0 + a_1 Q$, these coefficients are easily found. Example 3.1.6 indicates how to find the coefficients in more general cases. Then $\psi(Q)^{-1}b_n$ *is a particular solution* of the difference equation $\psi(Q)y_n = b_n$; the truncated expansions approximate this. Note that if $Q = \delta$ or ∇, the infinite expansion demands that b_n is defined also if $n < 0$.

Note that a similar technique, with the operator D, can also be applied to linear differential equations. Today this technique has to a large extent been replaced by the Laplace transform, that yields essentially the same algebraic calculations as operator calculus.

In some branches of applied mathematics it is popular to treat nonhomogeneous difference equations by means of a **generating function**, also called the **z-transform**, since both the definition and the practical computations are analogous to the Laplace transform. The z-transform of the sequence $y = \{y_n\}_0^\infty$ is

$$Y(z) = \sum_{n=0}^{\infty} y_n z^{-n}. \tag{3.2.58}$$

Note that the sequence $\{Ey\} = \{y_{n+1}\}$ has the z-transform $zY(z) - y_0$, $\{E^2 y\} = \{y_{n+2}\}$ has the z-transform $z^2 Y(z) - y_0 z - y_1$, etc.
If $Y(z)$ is available in *analytic* form, it can often be brought to a sum of functions, whose inverse z-transforms are known, by means of various analytic techniques, notably expansion into partial fractions, e.g., if $Y(z)$ is a rational function.
On the other hand, if *numerical values* of $Y(z)$ have been computed for complex values of z on some circle in $\mathbf{C}$ by means of an algorithm, then y_n can be determined by an obvious modification of the Cauchy+FFT method described in Sec. 3.1.3 (for expansions into negative powers of z). More information about the z-transform can be found in Strang [33, Sec. 6.3].

We are now in a position to exemplify in more detail the use of linear difference equations to studies of numerical stability, of the type mentioned above.

Theorem 3.2.11.

Necessary and sufficient for boundedness (stability) of all solutions of the difference equation (3.2.53) for all positive n is the following **root condition***:* [56]

i. *All roots of characteristic equation (3.2.54) should be located inside or on the unit circle $|z| \le 1$;*
ii. *The roots on the unit circle should be simple.*

Proof. Follows directly from Theorem 3.2.10.

This root condition corresponds to cases, where it is the absolute error that matters. It is basic in the theory of linear multistep methods for ordinary differential equations; see Sec. 13.4. Computer Graphics and an algebraic criterion due to Schur are useful for investigations of the root condition, see Sections 13.2 and 13.8.

The following example is based on a study by J.Todd [57] [35](1950) that was one of the first studies of the numerical stability of an algorithm for the approximate solution of ordinary differential equations.

Example 3.2.17.

Consider the initial-value problem

$$y''(x) = -y, \quad y(0) = 0, \quad y'(0) = 1, \tag{3.2.59}$$

with the exact solution $y(x) = \sin x$. To compute an approximate solution $y_k = y(x_k)$ at equidistant points $x_k = kh$, where h is a step length, we approximate the

[56]We shall say either that a difference equation or that a characteristic polynomial satisfies the root condition; the meaning is the same.
[57]John Todd, Irish-American numerical analyst.

second derivative according to (3.2.39),

$$y_k'' = h^{-2}(\delta^2 y_k + \frac{\delta^4 y_k}{12} + \frac{\delta^6 y_k}{90} + \ldots). \tag{3.2.60}$$

We first use the first term only; the second term shows that the truncation error of this approximation of y_k'' is asymptotically $h^2 y^{(4)}/12$. We then obtain the difference equation $h^{-2}\delta^2 y_k = -y_k$ or, in other words,

$$y_{k+2} = (2 - h^2)y_{k+1} - y_k, \quad y_0 = 0, \tag{3.2.61}$$

where a suitable value of y_1 is to be assigned. In the third column of Table 3.2.2 we show the results obtained using this recursion formula with $h = 0.1$ and $y_1 = \sin 0.1$. We obtain about 3 digits accuracy at the end, $x = 1.5$.

Since the algorithm was based on a second order accurate approximation of y'' one may expect that the solution of the differential equation is also second order accurate. This turns out to be correct in this case, e.g., if we divide the step size by 2, the errors will be divided by 4, approximately. We shall, however, see that we cannot always draw conclusions of this kind; we also have to take the numerical stability into account.

In the hope to obtain a more accurate solution, we shall now use one more term in the expansion (3.2.60); the third term then shows that the truncation error of this approximation is asymptotically $h^4 y^{(6)}/90$. The difference equation now reads

$$\delta^2 y_k - \frac{1}{12}\delta^4 y_k = -h^2 y_k \tag{3.2.62}$$

or, in other words,

$$y_{k+2} = 16y_{k+1} - (30 - 12h^2)y_k + 16y_{k-1} - y_{k-2}, \quad k \geq 2, \quad y_0 = 0, \tag{3.2.63}$$

where starting values for y_1, y_2, and y_3 need to be assigned. You see them in Table 3.2.2, where the results from this algorithm are shown in the fourth column. [58] We see that disaster has struck—the recursion is severely unstable! Already for $x = 0.6$ the results are less accurate than the second order scheme. For $x \geq 0.9$ the errors dominate completely.

We shall now look at these difference equations from the point of view of the root condition. The characteristic equation for Eq. (3.2.61) reads $u^2 - (2-h^2)u + 1 = 0$, and since $|2 - h^2| < 2$, direct computation shows that it has simple roots of unit modulus. The root condition is satisfied. [59]

For Eq.(3.2.63) the characteristic equation reads $u^4 - 16u^3 + (30 - 12h^2)u^2 - 16u + 1 = 0$. We see immediately that *the root condition cannot be satisfied.* Since the sum of the roots equals 16, it is impossible that all roots are inside or on the unit circle. In fact, the largest root equals 13.94. So, a tiny error at $x = 0.1$ has been multiplied by $13.94^{14} \approx 10^{16}$ at the end.

[58] The letters U and S in the headlines of the last two columns refer to "Unstable" and "Stable".
[59] By Example 3.2.14, the solution of (3.2.61) is $y_n = T_n(1 - h^2/2)$.

Table 3.2.2. *Integrating* $y'' = -y$, $y(0) = 0$, $y'(0) = 1$.

x_k	$\sin x_k$	2nd order	4th orderU	4th orderS
0.1	0.0998334166	0.0998334	0.0998334166	0.0998334166
0.2	0.1986693308	0.1986685	0.1986693307	0.1986693303
0.3	0.2955202067	0.2955169	0.2955202067	0.2955202050
0.4	0.3894183423	0.3894101	0.3894183688	0.3894183382
0.5	0.4794255386	0.4794093	0.4794126947	0.4794255305
0.6	0.5646424734	0.5646143	0.5643841035	0.5646424593
0.7	0.6442176872	0.6441732	0.6403394433	0.6442176650
0.8	0.7173560909	0.7172903	0.6627719932	0.7173560580
0.9	0.7833269096	0.7832346	0.0254286676	0.7833268635
1.0	0.8414709848	0.8413465	−9.654611899	0.8414709226
1.1	0.8912073601	0.8910450	−144.4011267	0.8912072789
1.2	0.9320390860	0.9318329	−2010.123761	0.9320389830
1.3	0.9635581854	0.9633026	−27834.59620	0.9635580577
1.4	0.9854497300	0.9851393	−385277.6258	0.9854495749
1.5	0.9974949866	0.9971245	−5332730.260	0.9974948015

It is easy to construct a stable fourth order accurate method. Just replace the term $\delta^4 y_k$ in (3.2.62) by $h^2\delta^2 y_k'' = -h^2\delta^2 y_k$. This leads to the recursion formula

$$y_{k+1} = (2 - \frac{h^2}{1 + h^2/12})y_k - y_{k-1}, \quad y_0 = 0. \tag{3.2.64}$$

This difference equation satisfies the root condition if $h^2 < 6$ (Problem 20(b)). This algorithm can be generalized to differential equations of the form $y'' = f(x, y)$. It is known under several names, e.g., Numerov's method. See Sec. 3.3 (Problem 31) and Ch. 13.

In the fifth column of Table 3.2.2 we show the results obtained using this recursion formula with $h = 0.1$ and $y_1 = \sin 0.1$. The error at the end is about $2\,10^{-7}$.

If the interesting solution of the original problem is itself strongly decreasing or strongly increasing, one should consider the location of the characteristic roots with respect to a circle in the complex plane that corresponds to the interesting solution. For example, if the interesting root is 0.8 then a root equal to -0.9 causes oscillations that may eventually become disturbing, if one is interested in *relative* accuracy also in a long run, even if the oscillating solution is small in the beginning.

We now confine the discussion to the cases where the original problems are to compute a particular solution of a *second order difference equation with variable coefficients*; several interesting problems of this type were mentioned above, and we formulated the questions: *can* we use a recurrence to find the wanted solution accurately, and *how* shall we use a recurrence, forwards or backwards. Typically the original problem contains some parameter, and one usually wants to make a study for an interval of parameter values.

Such questions are sometimes studied with *frozen coefficients*, i.e., the model problems are in the class of difference equations with constant coefficients in the range of the actual coefficients of the original problem; if one of the types of recurrence is satisfactory (i.e., numerically stable in some sense) for all model problems, one would like to conclude that they are satisfactory also for the original problem, but *the conclusion is not always valid* without further restrictions on the coefficients—see a counterexample in Problem 26c.

The technique with *frozen coefficients provides just a hint* that should always be *checked by numerical experiments* on the original problem. It is beyond the scope of this text to discuss what restrictions are needed. (Some ideas may be found in Sec. 13.8.) *If the coefficients of the original problem are slowly varying, however, there is a good chance that the numerical tests will confirm the hint*—but again: how slowly is "slowly"?

A warning against the use of one of the types of recurrence may also be a valuable result of a study, although it is negative.

The following lemma exemplifies a type of tool that may be useful in such cases. The proof is left for Problem 25a. Another useful tool is presented in Problem 26a and applied in Problem 26b.

Lemma 3.2.12. *Suppose that the wanted sequence y_n^* satisfies a difference equation (with constant coefficients),*

$$\alpha y_{n+1} + \beta y_n - \gamma y_{n-1} = 0, \quad (\alpha > \gamma > 0,\ \beta > 0),$$

and that y_n^* is known to be positive for all sufficiently large n. Then the characteristic roots can be written $0 < u_1 < 1,\ u_2 < 0$ and $|u_2| > u_1$. Then y_n^* is unique apart from a positive factor c; $y_n^* = cu_1^n,\ c > 0$.

A solution $\bar{y}_n$, called the trial solution *that is approximately of this form can be computed for $n = N : -1 : 0$ by* backward *recurrence starting with the "seed" $y_{N+1} = 0,\ y_N = 1$. If an accurate value of y_0^* is given, the wanted solution is*

$$y_n^* = \bar{y}_n y_0^* / \bar{y}_0,$$

with a relative error approximately proportional to $(u_2/u_1)^{n-N}$. (neglecting a possible error in y_0^*). [60]

The forward *recurrence is not recommended for finding y_n^* in this case, since the positive term $c_1 u_1^n$ will eventually be drowned by the oscillating term $c_2 u_2^n$ that will be introduced by the rounding errors.*

Comment: The proof is left for Problem 26c. Even if y_0 (in the use of the forward recurrence) has no rounding errors, such errors committed at later stages will yield similar contributions to the numerical results. See Problem 16d.

Example 3.2.18.

The "original problem" is to compute the parabolic cylinder function $U(a, x)$ which satisfies the difference equation

$$(a + \tfrac{1}{2})U(a+1, x) + xU(a, x) - U(a-1, x) = 0,$$

[60] If y_n^* is defined by some other condition, one can proceed analogously.

see Handbook of mathematical functions [1, Ch. 19] (written by J.C.P. Miller); see in particular Example 19.28.1.
To be more precise, we consider the case $x = 5$. Given $U(3,5) = 5.2847\,10^{-6}$ (obtained from a table in Handbook, p. 710), [61] we want to determine $U(a,5)$ for integer values of a, $a > 3$, as long as $|U(a,5)| > 10^{-15}$. We guess (a priori) that the discussion can be restricted to the interval (say) $a = [3,15]$. The above lemma then gives the hint of a backward recurrence, for $a = a' - 1 : -1 : 3$ for some appropriate a' (see below), in order to obtain a trial solution $\bar{U}_a$ with the seed $\bar{U}_{a'} = 1$, $\bar{U}_{a'+1} = 0$. Then the wanted solution becomes, by the Lemma, (with changed notation),

$$U(a,5) = \bar{U}_a U(3,5)/\bar{U}_3.$$

The positive characteristic root of the frozen difference equation varies from 0.174 to 0.14 for $a = 5 : 15$; while the modulus of the negative root is between 6.4 and 3.3 times as large. This motivates a choice of $a' \approx 4 + (-9 - log 5.3)/\ln 0.174 \approx 17$ for the backward recursion; it seems advisable to choose a' (say) 4 units larger than the value where U becomes negligible. (J.C.P. Miller starts at $a' = 19$ in Example 19.28.1.)
Forward recurrence with correctly rounded starting values $U(3,5) = 5.2847\,10^{-6}$, $U(4,5) = 9.172\,10^{-7}$, gives oscillating (absolute) errors of relatively slowly decreasing amplitude (approximately 10^{-11} that gradually drowns the exponentially decreasing true solution; the estimate of $U(a,5)$ itself became negative for $a = 10$, and then the results oscillated with approximate amplitude 10^{-11}, while the correct results decrease from the order of 10^{-11} to 10^{-15} as $a = 10 : 15$. The details are left for Problem 25b.
It is conceivable that this procedure can be used for all x in some interval around 5, but we refrain from presenting the properties of the parabolic cylinder function needed for determining the interval.

If the problem is nonlinear, one can instead solve the original problem with two seeds, (say) y'_N, y''_N, and study how the results deviate. The seeds should be so close that a linearization like $f(y'_n) - f(y''_n) \approx r_n(y'_n - y''_n)$ is acceptable, but $y'_n - y''_n$ should be well above the rounding error level. A more recent and general treatment of these matters is found in [10, Ch. 6].

Review Questions

1. Give expressions for the shift operator E^k in terms of Δ, ∇, and hD, and expressions for the central difference operator δ^2 in terms of E and hD.
2. Derive the best upper bound for the error of $\Delta^n y_0$, if we only know that the absolute value of the error of y_i, $i = 0, \dots, n$ does not exceed ϵ.
3. There is a theorem (and a corollary) about existence and uniqueness of approximation formulas of a certain type that are exact for polynomials of certain

[61] $U(3,5)$ corresponds to y_0^* in the Lemma.

class. Formulate these results, and sketch the proofs.

4. What bound can be given for the k'th difference of a function in terms of a bound for the k'th derivative of the same function?

5. Formulate the basic theorem concerning the use of operator expansions for deriving approximation formulas for linear operators.

6. Formulate Peano's Remainder Theorem, and compute the Peano kernel for a given symmetric functional (with at most four subintervals).

7. Express polynomial interpolation formulas in terms of forward and backward difference operators.

8. Give Stirling's interpolation formula for quadratic interpolation with approximate bounds for truncation error and irregular error.

9. Derive central difference formulas for $f'(x_0)$ and $f''(x_0)$ that are exact for $f \in \mathcal{P}_4$. They should only use function values at x_j, $j = 0, \pm 1, \pm 2, \ldots$, as many as needed. Give asymptotic error estimates.

10. Derive the formula for the general solution of the difference equation $y_{n+k} + a_1 y_{n+k-1} + \ldots + a_k y_n = 0$, when the characteristic equation has simple roots only. What is the general solution, when the characteristic equation has multiple roots?

11. What is the general solution of the difference equation $\Delta^k y_n = an + b$?

12. Prove Lemma 3.2.12, and present the main features of its application to the parabolic cylinder function.

Problems and Computer Exercises

1. (a) Show that $(1+\Delta)(1-\nabla) = 1$, $\Delta - \nabla = \Delta\nabla = \delta^2 = E - 2 + E^{-1}$, and that $\delta^2 y_n = y_{n+1} - 2y_n + y_{n-1}$.

(b) Let $\Delta^p y_n, \nabla^p y_m, \delta^p y_k$ all denote the same quantity. How are n, m, k connected? Along which lines in the difference scheme are the subscripts constant?

(c) Given the values of y_n, $\nabla y_n, \ldots,$ $\nabla^k y_n$, for a particular value of n. Find a recurrence relation for computing y_n, $y_{n-1}, \ldots,$ y_{n-k}, by simple additions only. On the way you obtain the full difference scheme of this sequence.

(d) *Repeated summation by parts.* Show that if $u_1 = u_N = v_1 = v_N = 0$, then

$$\sum_{n=1}^{N-1} u_n \Delta^2 v_{n-1} = -\sum_{n=1}^{N-1} \Delta u_n \Delta v_n = \sum_{n=1}^{N-1} v_n \Delta^2 u_{n-1}.$$

(e)Show that if $\Delta^k v_n \to 0$, as $n \to \infty$, then $\sum_{n=m}^{\infty} \Delta^k v_n = -\Delta^{k-1} v_m$.

(f)Show that $(\mu\delta^3 + 2\mu\delta) f_0 = f_2 - f_{-2}$

(g) Prove, e.g., by means of summation by parts, that $\sum_{n=0}^{\infty} u_n z^n$, $|z| = 1$, $z \neq 1$, is convergent if $u_n \to 0$ monotonically. Formulate similar results for real cosine and sine series.

2. (a) Prove, e.g., by induction, the following two formulas:

$$\Delta_x^j \binom{x}{k} = \binom{x}{k-j}, \quad j \le k,$$

where Δ_x means differencing with respect to x, with $h = 1$.

$$\Delta^j x^{-1} = \frac{(-h)^j j!}{x(x+h)\cdots(x+jh)}.$$

Find the analogous expression for $\nabla^j x^{-1}$.

(b) What formulas with derivatives instead of differences are these formulas analogous to?

(c) Show the following formulas, if $x,\ a$ are integers:

$$\sum_{n=a}^{x-1} \binom{n}{k-1} = \binom{x}{k} - \binom{a}{k},$$

$$\sum_{n=x}^{\infty} \frac{1}{n(n+1)\cdots(n+j)} = \frac{1}{j} \cdot \frac{1}{x(x+1)\cdots(x+j-1)}.$$

Modify these results for non-integer x; $x - a$ is still an integer.

(d) Suppose that $b \neq 0,\ -1,\ -2, \ldots$, and set

$$c_0(a,b) = 1, \quad c_n(a,b) = \frac{a(a+1)\ldots(a+n-1)}{b(b+1)\ldots(b+n-1)}, \quad n = 1,2,3,\ldots$$

Show, e.g., by induction that $(-\Delta)^k c_n(a,b) = c_k(b-a,b)c_n(a,b+k)$, hence $(-\Delta)^n c_0(a,b) = c_n(b-a,b)$.

(e) Compute for $a = e,\ b = \pi$ (say), $c_n(a,b),\ n = 1:100$. How do you avoid overflow? Compute $\Delta^n c_0(a,b)$, both numerically by the difference scheme, and according to the formula in (d). Compare the results and formulate your experiences. Do the same with $a = e,\ b = \pi^2$.

Do the same with $\Delta^j x^{-1}$ for various values of $x,\ j$ and h.

3. Set

$$\begin{aligned} \text{YORD} &= (y_{n-k}, y_{n-k+1}, \ldots, y_{n-1}, y_n), \\ \text{YDIF} &= (\nabla^k y_n,\ \nabla^{k-1} y_n, \ldots, \nabla y_n,\ y_n). \end{aligned}$$

Note that the results of this problem also hold if the y_j are column vectors.

(a) Find a matrix PASC, such that YDIF = YORD · PASC. Show that

$$\text{YORD} = \text{YDIF} \cdot \text{PASC}, \quad \text{hence} \quad \text{PASC}^{-1} = \text{PASC}.$$

How do you generate this matrix by means of a simple recurrence relation? *Hint:* PASC stands for Pascal, but do not forget the minus signs in this triangular matrix. Compare Problem 3 of Sec. 1.3.

(b) Suppose that $\sum_{j=0}^{k} \alpha_j E^{-j}$ and $\sum_{j=0}^{k} a_j \nabla^j$ represent the same operator. Set $\alpha = (\alpha_k, \alpha_{k-1}, \ldots, \alpha_0)^T$, and $a = (a_k, a_{k-1}, \ldots, a_0)^T$, i.e., $\textsc{Yord} \cdot \alpha \equiv \textsc{Ydif} \cdot a$. Show that $\textsc{Pasc} \cdot a = \alpha, \quad \textsc{Pasc} \cdot \alpha = a$.

(c) The matrix PASC depends on the integer k. Is it true that the matrix which is obtained for a certain k is a submatrix of the matrix you obtain for a larger value of k?

(d) Compare this method of performing the mapping $\textsc{Yord} \mapsto \textsc{Ydif}$ with the ordinary construction of a difference scheme. Consider the number of arithmetic operations, the kind of arithmetic operations, rounding errors, convenience of programming in a language with matrix operations as primary operations etc.
Compare in the same way this method of performing the inverse mapping with the algorithm in Problem 1c.

4. (a) Set $f(x) = \tan x$. Compute by the use of the table of $\tan x$ in Example 3.2.3, and the interpolation and differentiation formulas given in the above examples (almost) as accurately as possible $f'(1.35)$, $f(1.322)$, $f'(1.325)$, $f''(1.32)$. Estimate the influence of rounding errors of the function values and estimate the truncation errors.

(b) Write a program for computing a difference scheme. Use it for computing the difference scheme for more accurate values of $\tan x$, $x = 1.30 : 0.01 : 1.35$, and calculate improved values of the functionals in (a). Compare the error estimates with the true errors.

(c) Verify the assumptions of Theorem 3.2.6 for one of the three interpolation formulas in Example 3.2.9 .

(d) It is rather easy to find the values at $\theta = 0$ of the first two derivatives of Stirling's interpolation formula. You find thus explicit expressions for the coefficients in the formulas for $f'(x_0)$ and $f''(x_0)$ in (3.2.43) and (3.2.39), respectively. Check numerically a few coefficients in these equations, and explain why they are reciprocals of integers. Also note that each coefficient in (3.2.43) has a simple relation to the corresponding coefficient in (3.2.39).

5. (a) Study Bickley's table (Table 3.2.1), and derive some of the formulas, in particular the expressions for δ and μ in terms of hD, and vice versa.

(b) Show that $h^{-k}\delta^k - D^k$ has an expansion into *even* powers of h, when k is even. Find an analogous result for $h^{-k}\mu\delta^k - D^k$ when k is odd.

6. (a) Compute $f'(10)/12$, $f^{(3)}(10)/720$, $f^5(10)/30240$, by means of (3.2.20), given values of $f(x)$ for integer values of x. [62] Do this for $f(x) = x^{-3/2}$. Compare with the correct derivatives. Then do the same also for $f(x) = (x^3+1)^{-1/2}$.

(b) Study the backwards differentiation formula, see Example 3.2.6, on a computer. Compute $f'(1)$ for $f(x) = 1/x$, for $h = 0.02$ and $h = 0.03$, and compare with the exact result. Make a semi-logarithmic plot of the total error after n

[62]This is asked for, e.g., in applications of Euler–Maclaurin's formula, § 3.3.4.

terms, $n = 1 : 29$. Study also the sign of the error. For each case, try to find out whether the achievable accuracy is set by the rounding errors or by the semiconvergence of the series.
Hint: A formula mentioned in Problem 2(a) can be helpful. Also note that this problem is both similar and very different from the function $tan(x)$ that was studied in Example 3.2.6.
(c) Set $x_i = x_0 + ih$, $t = \frac{x-x_2}{h}$. Show that

$$y(x) = y_2 + t\Delta y_2 + \frac{t(t-1)}{2}\Delta^2 y_2 + \frac{t(t-1)(t-2)}{6}\Delta^3 y_1$$

equals the interpolation polynomial in $\mathcal{P}_4$ determined by the values (x_i, y_i), $i = 1 : 4$. (Note that $\Delta^3 y_1$ is used instead of $\Delta^3 y_2$ which is located outside the scheme. Is this OK?)

7. (a) Show the validity of the algorithm in (3.2.36).
(b) A well known formula reads $P(D)(e^{\alpha t}u(t)) = e^{\alpha t}P(D+\alpha)u(t)$, where P is an arbitrary polynomial. Prove this, as well as the following analogous formulas:

$$P(E)(a^n u_n) = a^n P(aE)u_n,$$
$$P(\Delta/h)\big((1+\alpha h)^n u_n\big) = (1+\alpha h)^n P((1+\alpha h)\Delta/h + \alpha)u_n.$$

Can you find a more beautiful or more practical variant?

8. Find the Peano kernel $K(u)$ for the functional $\Delta^2 f(x_0)$. Compute $\int_{\mathbf{R}} K(u)\,du$ both by direct integration of $K(u)$, and by computing $\Delta^2 f(x_0)$ for a suitably chosen function f.

9. Let $y_j = y(t_j)$, $y'_j = y'(t_j)$. The following relations are of great interest in the numerical integration of ordinary differential equations, $y' = f(y)$, see Sec. 13.4. [63]
(a) *The implicit Adams formula*:

$$y_{n+1} - y_n = h(a_0 y'_{n+1} + a_1 \nabla y'_{n+1} + a_2 \nabla^2 y'_{n+1} + \cdots).$$

Show that $\nabla = -\ln(1-\nabla)\sum a_i \nabla^i$, and find a recurrence relation for the coefficients. The coefficients a_i, $i = 0 : 6$, read as follows. Check a few of them.

$$a_i = \; 1, \frac{-1}{2}, \frac{-1}{12}, \frac{-1}{24}, \frac{-19}{720}, \frac{-3}{160}, \frac{-863}{60480}.$$

Alternatively, derive the coefficients by means of the matrix representation, of a truncated power series, see 3.1.11
(b) *The explicit Adams formula*:

$$y_{n+1} - y_n = h(b_0 y'_n + b_1 \nabla y'_n + b_2 \nabla^2 y'_n + \cdots).$$

Show that $\sum b_i \nabla^i E^{-1} = \sum a_i \nabla^i$, and show that

$$b_n - b_{n-1} = a_n, \quad (n \geq 1).$$

[63]The formulas can also be used for systems; just interpret y_n, y'_n as vectors.

The coefficients b_i, $i = 0:6$, read as follows. Check a few of them.

$$b_i = \ 1, \frac{1}{2}, \frac{5}{12}, \frac{3}{8}, \frac{251}{720}, \frac{95}{288}, \frac{19087}{60480}.$$

(c) Apply the the second order explicit Adams formula, i.e., $y_{n+1} - y_n = h(y'_n + \frac{1}{2}\nabla y'_n)$, to the differential equation $y' = -y^2$ with the initial condition $y(0) = 1$ and the step size $h = 0.1$. Two initial values are needed for the recurrence; $y_0 = y(0) = 1$, of course, and we choose [64] $y_1 = 0.9090$. Then compute $y'_0 = -y_0^2$, $y'_1 = -y_1^2$. Then the explicit Adams formula yields y_2, and so on. Compute a few steps, and compare with the exact solution. [65]

10. Let $y_j = y_0 + jh$. Find the asymptotic behavior as $h \to 0$ of

$$(5(y_1 - y_0) + (y_2 - y_1))/(2h) - y'_0 - 2y'_1.$$

Comment: This is of interest in the analysis of cubic spline interpolation in § 4.6.4.

11. *Subtabulation or dense output.* It sometimes happens that the values of some function $f(x)$ can be computed by some very time-consuming algorithm only, and that one therefore computes it much sparser than is needed for the application of the results. It was common in the pre-computer age to compute sparse tables that needed interpolation by polynomials of a high degree; then one needed a simple procedure to obtain a denser table for some section of the table, so that linear interpolation can be used there, possibly with lower accuracy. Today a similar situation may occur in connection with the graphical output of the results of (say) a numerical solution of a differential equation.

Define the operators ∇ and ∇_k by the equations

$$\nabla f(x) = f(x) - f(x-h), \quad \nabla_k f(x) = f(x) - f(x-kh), \ (k < 1),$$

and set $\nabla_k^r = \sum_{s=r}^{\infty} c_{rs}(k)\nabla^s$.

(a) In order to compute the coefficients c_{rs}, $r \le s \le m$, you are advised to use a subroutine for finding the coefficients in the product of two polynomials, truncate the result, and apply the subroutine $m-1$ times.

(b) Given

f_n	∇f_n	$\nabla^2 f_n$	$\nabla^3 f_n$	$\nabla^4 f_n$
1	.181269	.032858	.005956	.001080

Compute for $k = \frac{1}{2}$, $f_n = f(x_n)$, $\nabla_k^j f_n$ for $j = 1:4$.

Compute $f(x_n - h)$ and $f(x_n - 2h)$, by means of both $\{\nabla^j f_n\}$ and $\{\nabla_k^j f_n\}$ and compare the results. How big difference of the results did you expect, and how big difference do you obtain?

[64] There are several ways of obtaining $y_1 \approx y(h)$, e.g., by one step of Runge's 2nd order method, see § 1.4.3, or by a series expansion, like in Example 3.1.2.

[65] For an *implicit* Adams formula it is necessary, in this example, to solve a quadratic equation in each step. See in Sec. 13.2.4 how to proceed for general differential systems.

12. (a) Find **Simpson's formula**, i.e., the unique quadrature formula of the form $\int_{-h}^{h} f(x)dx \approx c_{-1}f(-h) + c_0 f(0) + c_1 f(h)$ that is exact whenever $f \in \mathcal{P}_3$. By symmetry it is exact even for $f \in \mathcal{P}_4$; recall the discussion in Example 3.2.11. Find an error estimate. Try to find several derivations of this classical formula, with or without the use of difference operators. Is Simpson's formula a particular case of Newton–Cotes's formulas?

(b) Show that a remainder functional for Simpson's formula is of the form $Rf = \int_{\mathbf{R}} f^{(4)}(u)K(u)\,du$, where the Peano kernel equals $K(u) = -\frac{1}{72}(h-u)^3(3u+h)^2$ for $0 \le u \le h$; $K(u) = K(|u|)$ for $u < 0$, $K(u) = 0$ for $|u| > h$. Also show that the remainder equals $-\frac{1}{180}f^{(4)}(\xi)Hh^4$, $|\xi| < h$. ($H = 2h$ is the length of the interval of integration.) If Simpson's formula is used over the interval $[a, b]$ with an even number of steps of length h, s show that the error is $\frac{1}{180}f^{(4)}(\xi)(b-a)h^4$, $\xi \in (a, b)$.

(c) Find the kernel $K_2(u)$, such that $Rf = \int_{\mathbf{R}} f''(u)K_2(u)\,du$, and find the best constants c, p, such that

$$|Rf| \le ch^p \max|f''(u)|, \quad \forall f \in C^2[-h, h].$$

If you are going to deal with functions that are not in C^3, would you still prefer Simpson's formula to the trapezoidal rule?

13. (a) We shall use the notation that was introduced in connection with Eq. (3.2.45). Show hat $b = M_{k+1}a$, and show how the matrix M can be generated by a recurrence. (Think of Pascal's triangle.)

(b) Use the matrix M for deriving (3.2.50) from (3.2.49). How do you obtain the remainder term? If you obtain the coefficients as decimal fractions, multiply them by 14175/4 in order to check that they agree with (3.2.50).

(c) Use Cauchy+FFT for deriving (3.2.49), and the open formula and the remainder for the same interval.

(d) Set $z_n = \nabla^{-1}y_n - \Delta^{-1}y_0$. We have, in the literature, seen the interpretation that $z_n = \sum_{j=0}^{n} y_j$ if $n \ge 0$. It seems to require some extra conditions to be true. Investigate if the conditions $z_{-1} = y_{-1} = 0$ are necessary and sufficient. Can you suggest better conditions? (The equations $\Delta\Delta^{-1} = \nabla\nabla^{-1} = 1$ mentioned earlier are assumed to be true.)

COMMENT: This formula will be used in a problem in Sec. 3.3 concerning Gregory's Quadrature Formula.

14. Solve the following difference equations. A solution in complex form should be transformed to real form. As a check, compute (say) y_2 both by recurrence and by your closed form expression.

(a) $y_{n+2} - 2y_{n+1} - 3y_n = 0$, $y_0 = 0$, $y_1 = 1$;

(b) $y_{n+2} - 4y_{n+1} + 5y_n = 0$, $y_0 = 0$, $y_1 = 2$;

(c) There exist problems with two-point boundary conditions for difference equations, as for differential equations. $y_{n+2} - 2y_{n+1} - 3y_n = 0$, $y_0 = 0$, $y_{10} = 1$;

(d) $y_{n+2} + 2y_{n+1} + y_n = 0$, $y_0 = 1$, $y_1 = 0$;

(e) $y_{n+1} - y_n = 2^n$, $y_0 = 0$;

(f) $y_{n+2} - 2y_{n+1} - 3y_n = 1 + \cos\frac{\pi n}{3}$, $y_0 = y_1 = 0$;

Hint: The right hand side is $\Re(1 + a^n)$, where $a = e^{\pi i/3}$.

(g) $y_{n+1} - y_n = n$, $y_0 = 0$;

(h) $y_{n+1} - 2y_n = n2^n$, $y_0 = 0$;

15. (a) Prove Lemma 3.2.8 (in the beginning of Sec. 3.2.3).

(b) Consider the difference equation $y_{n+2} - 5y_{n+1} + 6y_n = 2n + 3(-1)^n$. Determine a particular solution of the form $y_n = an + b + c(-1)^n$.

(c) Solve also the difference equation $y_{n+2} - 6y_{n+1} + 5y_n = 2n + 3(-1)^n$. Why and how must you change the form of the particular solution?

16. (a) Show that the difference equation $\sum_{i=0}^{k} b_i\Delta^i y_n = 0$ has the characteristic equation: $\sum_{i=0}^{k} b_i(u-1)^i = 0$.

(b) Solve the difference equation $\Delta^2 y_n - 3\Delta y_n + 2y_n = 0$, with initial condition $\Delta y_0 = 1$.

(c) Find the characteristic equation for the equation $\sum_{i=0}^{k} b_i\nabla^i y_n = 0$?

17. The influence of wrong boundary slopes for cubic spline interpolation (with equidistant data)—see Sec. 4.6—is governed by the difference equation

$$e_{n+1} + 4e_n + e_{n-1} = 0, \quad 0 < n < m,$$

e_0, e_m given. Show that $e_n \approx u^n e_0 + u^{m-n} e_m$, $u = \sqrt{3} - 2 \approx -0.27$. More precisely

$$|e_n - (u^n e_0 + u^{m-n} e_m)| \le \frac{2|u^{3m/2}| \max(|e_0|, |e_m|)}{1 - |u|^m}.$$

Generalize the simpler of these results to other difference and differential equations.

18. The Fibonacci sequence is defined by the recurrence relation

$$y_n = y_{n-1} + y_{n-2}, \quad y_0 = 0, \quad y_1 = 1.$$

(a) Calculate $\lim_{n\to\infty} y_{n+1}/y_n$.

(b) The error of the secant method (see Sec. 6.4) satisfies approximately the difference equation $\epsilon_n = C\epsilon_{n-1}\epsilon_{n-2}$. Solve this difference equation. Determine p, such that $\epsilon_{n+1}/\epsilon_n^p$ tends to a finite nonzero limit as $n \to \infty$. Calculate this limit.

19. For several algorithms, such as the Fast Fourier Transform and some sorting methods, one can find that the work $W(n)$ for the application of them to data of size n satisfies a recurrence relation of the form:

$$W(n) = 2W(n/2) + kn,$$

where k is a constant. Find $W(n)$.

20. When the recursion $x_{n+2} = (32x_{n+1} - 20x_n)/3$, $x_0 = 3$, $x_1 = 2$, was solved numerically in low precision (23 bits mantissa), one obtained the (rounded) values

i	2	3	4	5	6	7	8	9	10	11	12
$\bar{x}_i$	1.33	0.89	0.59	0.40	0.26	0.18	0.11	0.03	−0.46	−5.05	−50.80

Explain the difference from the exact values $x_n = 3(2/3)^n$.

21. (a) k, N are given integers $0 \le k << \le N$. A "discrete Green's function" $G_{n,k}$, $0 \le n \le N$ for the central difference operator [66] $-\Delta\nabla$, is defined as the solution $u_n = G_{n,k}$ of the difference equation with boundary conditions,

$$-\Delta\nabla u_n = \delta_{n,k}, \quad u_0 = u_N = 0, \quad \text{(Kronecker delta)}.$$

Derive a fairly simple expression for $G_{n,k}$.

(b) Find (by computer) the inverse of the tri-diagonal matrix

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}.$$

What is the relation between Probs (a) and (b)? Find a formula for the elements of A^{-1}. Express the solution of the inhomogeneous difference equation $-\Delta\nabla u_n = b_n$, $u_0 = u_N = 0$, both in terms of the Green function $G_{n,k}$ and in terms of A^{-1} (for general N).

(c) Try to find an analogous formula [67] for the solution of an inhomogeneous boundary value problem for the *differential* equation ,$-u'' = f(x)$, $u(0) = u(1) = 0$. COMMENT: Since A^{-1} is a filled matrix, it should be avoided at the solution of the linear system $Ay = b$ with a tridiagonal matrix A. There exist much better procedures, see Sec. 6.4.6 for numerical work. Conceptually Green's function is very important, above all in connection with boundary and eigenvalue problems for ordinary and partial differential equations, and their discretizations.

22. Vacant. (The problem which was here will be moved to Ch.13.

23. (a) Demonstrate the formula

$$\sum_0^\infty \frac{(-x)^n c_n}{n!} = e^{-x} \sum_0^\infty \frac{x^n(-\Delta)^n c_0}{n!}. \tag{3.2.65}$$

Hint: Use the relation $e^{-xE} = e^{-x(1+\Delta)} = e^{-x}e^{-x\Delta}$.

(b) For an important class of sequences called *completely monotonic*, see

[66]together with the boundary conditions given below,

[67]In a *differential* equation, analogous to Problem 21(a), the Kronecker delta is to be replaced by the Dirac delta function. Also note that the inverse of the differential operator here can be described as an integral operator with the Green's function as the "kernel".

Sec. 3.3.6, $\{c_n\}$ and $\{(-\Delta)^n c_0\}$ are typically positive and decreasing sequences. For such sequences, the left hand side becomes extremely ill-conditioned for large x, (say) $x = 100$, while the graph of the terms on the right hand side (if exactly computed) are bell-shaped, almost like the normal probability density with mean x and standard deviation $\sqrt{x}$. We tentatively call such a sum a **bell sum**. We shall see in Sec. 3.3 that such positive sums can be computed with little effort and no trouble with rounding errors, *if their coefficients are accurate.*

Compute the left hand side of (3.2.65), for $c_n = 1/(n+1)$, $x = 10 : 10 : 100$, and compute the right hand side, both with numerically computed differences and with exact differences; the latter are found in Problem(2a). (In this particular case you can also find the exact sum.)

Suppose that the higher differences $\{(-\Delta)^n c_0\}$ have been computed recursively from rounded values of c_n. Explain why one may fear that the right hand side of (3.2.65) does not provide much better results than the left hand side.

(c) Use (3.2.65) to derive the second expansion for $\mathrm{erf}(x)$ in Problem 10 of Sec. 3.1 from the first expansion. *Hint:* Use one of the results of Problem 2a.

(d) If $c_n = c_n(a, b)$ is defined as in Problem 2d, then the left hand side becomes the Maclaurin expansion of the Kummer function $M(a, b, -x)$, see Abramowitz-Stegun, Ch. 13. Show that $M(a, b, -x) = e^{-x}M(b - a, b, x)$ by means of the results of Problem 23a and 2d. [68]

24. (a) The difference equation $y_n + 5y_{n-1} = n^{-1}$ was discussed in Sec. 1.3.3. It can also be written thus: $(6 + \Delta)y_{n-1} = n^{-1}$. The expansion of $(6+\Delta)^{-1}n^{-1}$ into powers of $\Delta/6$ provides a particular solution of the difference equation. Compute this numerically for a few values of n. Try to prove the convergence, with or without the expression in Problem 2b. Is this the same as the particular solution $I_n = \int_0^1 x^n(x + 5)^{-1}dx$ that was studied in Ch.1?
Hint: What happens as $n \to \infty$? Can more than one solution of this difference equation be bounded as $n \to \infty$?

(b) Make a similar study to the difference equation related to the integral in Problem 9 of Sec.1.3. Why does the argument suggested by the hint of (a) not work in this case? Try another proof.

25. (a) Prove Lemma 3.2.12. How is the conclusion to be changed, if we do not suppose that $\gamma < \alpha$, though the coefficients are still positive? Show that a backward recurrence is still to be recommended.

(b) Work out on a computer the numerical details of Example 3.2.18, and compare with Abramowitz-Stegun, Example 19.28.1. (Some deviations are to be expected, since Miller used other rounding rules.) Try to detect the oscillating component by computing the difference scheme of the the computed $U(a, 5)$, and estimate roughly the error of the computed values.

26. (a) For which constant real a does the difference equation $y_{n+1} - 2ay_n + y_{n-1} =$

[68]This formula is due to Kummer, German mathematician (1810-1893). It is well known in the theory of the confluent hypergeometric functions, where it is usually proved in other ways.

0 satisfy the root condition?
For which values of the real constant a does there exist a solution, such that $\lim_{n\to\infty} y_n = 0$? For these values of a, how do you construct a solution $y_n = y_n^*$ by a recurrence and normalization, so that this condition as well as the condition $y_0^* + 2\sum_{m=1}^{\infty} y_{2m}^* = 1$ are satisfied. Is y_n^* unique? Give also an explicit expression for y_n^*.
For the other real values of a, show that y_n^* does not exist, but that for any given y_0, y_1 a solution can be accurately constructed by forward recurrence. Give an explicit expression for this solution in terms of Chebyshev polynomials (of the first and the second kind). Is it true that backward recurrence is also stable, though more complicated than forward recurrence?
(b) The Bessel function $J_k(z)$ satisfies the difference equation,

$$J_{k+1}(z) - (2k/z)J_k(z) + J_{k-1}(z) = 0, \quad k = 1,\ 2,\ 3, \ldots,$$

and the identities,

$$J_0(z) + 2J_2(z) + 2J_4(z) + 2J_6(z) + \ldots = 1;$$

$$J_0(z) - 2J_2(z) + 2J_4(z) - 2J_6(z) + \ldots = \cos z;$$

see Abramowitz and Stegun [1], 9.1.27, 9.1.46 and 9.1.47.
Show how one of the identities can be used for normalizing the trial sequence obtained by a backwards recurrence. Under what condition does Problem 26(a) give the hint to use the backwards recurrence for this difference equation? Study the section on Bessel functions of integer order in Numerical Recipes. Apply this technique for $z = 10,\ 1,\ 0.1$ (say). The asymptotic formula [1], 9.3.1,

$$J_k(z) \sim \frac{1}{\sqrt{2\pi k}}\left(\frac{ez}{2k}\right)^k, \quad k \gg 1,\ z \text{ fixed.}$$

may be useful for your decision where to start the backward recurrence. Use at least two starting points, and subtract the results (after normalization).
COMMENT: The above difference equation for $J_k(z)$ is also satisfied by a function denoted $Y_k(z)$, $Y_k(z) \sim \frac{-2}{\sqrt{2\pi k}}\left(\frac{ez}{2k}\right)^{-k}$, $(k \gg 1)$. How do these two solutions disturb each other, when forward or backward recurrence is used?
(c) *A counterexample* to the technique with frozen coefficients. Consider the difference equation $y_{n+1} - (-1)^n y_n + y_{n-1} = 0$. The technique with frozen coefficients leads to the consideration of the difference equations $z_{n+1} - 2az_n + z_{n-1} = 0$, $a \in [-0.5, 0.5]$; all of them have only bounded solutions.
Find by numerical experiment that, nevertheless, there seems to exist unbounded solutions y_n of the first difference equation.
COMMENT: A theoretical proof of this is found by noting that the mapping $(y_{2n},\ y_{2n+1}) \mapsto (y_{2n+2},\ y_{2n+3})$ is represented by a matrix that is independent of n and has an eigenvalue that is less than -1. (This type of ideas will be elaborated more in Sec. 13.9.)

27. Let $\{b_n\}_{-\infty}^{\infty}$ be a given sequence, and consider the difference equation,

$$y_{n-1} + 4y_n + y_{n+1} = b_n,$$

which can also be written in the form $(6 + \delta^2)y_n = b_n$. (a) Show that the difference equation has at most one solution that is bounded for $-\infty < n < +\infty$. Find a particular solution in the form of an expansion into powers of the operator $\delta^2/6$. (This is hopefully bounded.)

(b) Apply it numerically to the sequence $b_n = (1+n^2h^2)^{-1}$, for a few values of the step size h, e.g., $h = 0.1, 0.2, 0.5, 1$. Study for $n = 0$ the rate of decrease(?) of the terms in the expansion. Terminate when you estimate that the error is (say) 10^{-6}. Check how well the difference equation is satisfied by the result.

(c) Study theoretically bounds for the terms when $b_n = \exp(i\omega hn)$, $\omega \in \mathbf{R}$. Does the expansion converge? Compare your conclusions with numerical experiments. Extend to the case when $b_n = B(nh)$, where $B(t)$ can be represented by an absolutely convergent Fourier integral, $B(t) = \int_{-\infty}^{\infty} e^{i\omega t}\beta(\omega)d\omega$. Note that $B(t) = (1+t^2)^{-1}$ if $\beta(\omega) = \frac{1}{2}e^{-|\omega|}$. Compare the theoretical results with the experimental results in (b).

(d) Put $Q = \delta^2/6$. Show that $\tilde{y}_n \equiv (1 - Q + Q^2 + \ldots \pm Q^{k-1})b_n/6$ satisfies the difference equation $(1 + Q)(\tilde{y}_n - y_n) = Q^k b_n/6$.

Comment: This procedure is worthwhile if the sequence b_n is so smooth that (say) 2 or 3 terms give satisfactory accuracy.

3.3 Acceleration of Convergence

3.3.1 Introduction

If a sequence $\{s_n\}_0^{\infty}$ converges slowly towards a limit s but has a sort of regular behavior when n is large, it can under certain conditions be transformed into another infinite sequence $\{s'_n\}$, that converges much faster to the same limit. Here s'_n usually depends on the first n elements of the original sequence only.

This is called convergence acceleration. Such a *sequence* transformation may be iterated, to yield a sequence of infinite sequences, $\{s''_n\}$, $\{s'''_n\}$ etc., hopefully with improved convergence towards the same limit s. For an *infinite series* convergence acceleration means the convergence acceleration of its sequence of partial sums. Recall a comment in the beginning of Sec. 3.1.1. Some algorithms are most easily discussed in terms of sequences, others in terms of series.

Several transformations, linear as well as nonlinear, have been suggested and are successful, under various conditions. Some of them are most successful on *oscillating sequences* (alternating series or series in a complex variable), see Sec. 3.3.2 (Aitken etc.), Sec. 3.3.3 (repeated averages, Euler's transformation). and Sec. 3.3.7 (Gustafson-Chebyshev). Others work primarily on *monotonic sequences* (series with positive terms), see Sec. 3.3.2 (variants of Aitken acceleration), Sec. 3.3.4 (Euler–Maclaurin) and Sec. 3.3.5 (Richardson etc.).

Convergence acceleration cannot be applied to "arbitrary sequences"; some sort of conditions are necessary that restrict the variation of the future elements of the sequence, i.e., the elements which are not computed numerically. In this section, these conditions are of a rather general type, in terms of *higher monotonicity, analyticity or asymptotic behavior of simple and usual types.* In Sec. 3.3.6 there is an introduction to the completely monotonic functions and some related classes of analytic functions, for which the techniques of convergence acceleration can be put on a relatively solid theoretical basis.

Nevertheless some of these techniques may even sometimes be successfully applied to *semiconvergent sequences.* Several of them can also use a limited number of coefficients of a power series for the computation of values of an *analytic continuation* of a function, outside the circle of convergence of the series that defined it.

Convergence acceleration is related to the summation of divergent series employed in Pure Mathematics, but the purpose and the aspects are different. For example, Cesaro summation, which is important in the theory of Fourier series—it can make a divergent Fourier expansion for a non-smooth periodic function convergent—is of little interest for our typical applications of convergence acceleration. In Example 3.3.2, Cesaro summation would converge only like $O(1/n)$, while the procedures recommended in this section converge like $O(a^n)$ where $|a| \leq \frac{1}{3}$. (This comparison holds for most slowly convergent alternating series with completely monotonic terms.)

Some techniques for convergence acceleration transform a power series into a *sequence of rational functions.* We postpone the treatment of some of these to Sec. 3.4, e.g., continued fractions and Padé approximation, including the ϵ-algorithm of P. Wynn.

There are methods (due to Lindelöf, Plana and others) that transform an infinite series to an integral in the complex plane. They can, with appropriate numerical procedures for computing the integral, compete with the methods mentioned for the purposes mentioned, but they have the additional property to be applicable to some *ill-conditioned series.* They will be discussed in Ch. 12, because certain tools are needed for the explanation.

Ch. 12 also includes the Poisson summation formula, which is not "a general purpose formula", though it can be amazingly successful to a certain class of series $\sum a(n)$, namely if $a(x)$ has a rapidly decreasing Fourier Transform. The Poisson formula is also an invaluable tool for the design and analysis of numerical methods for several problems, see Theorem 3.3.3.

In addition to the "general purpose" techniques to be discussed in this chapter, there are other techniques of convergence acceleration based on the use of more specific knowledge about a problem.

Irregular errors are very disturbing, when these techniques are used, and they sometimes set the limit for the reachable accuracy. For the sake of simplicity we therefore use high precision, `macheps` $= 2^{-53} \approx 10^{-16}$. in most examples, but lower precision can often be used, if you need (say) four decimal places only in the results.

3.3.2 Comparison Series and Aitken Acceleration

Suppose that the terms in the series $\sum_{j=1}^{\infty} a_j$ behave, for large j, like the terms of a series $\sum_{j=1}^{\infty} b_j$, i.e., $\lim_{j\to\infty} a_j/b_j = 1$. Then if the sum $s = \sum_{j=1}^{\infty} b_j$ is known one can write

$$\sum_{j=1}^{\infty} a_j = s + \sum_{j=1}^{\infty}(a_j - b_j),$$

where the series on the right hand side converges more quickly than the given series. We call this making use of a simple **comparison problem**. The same idea is used in many other contexts—for example, in the computation of integrals where the integrand has a singularity. Usual comparison series are $\sum n^{-2} = \pi^2/6$, $\sum n^{-4} = \pi^4/90$, etc. A general expression is found in Lemma 3.3.1. No simple closed form is known for $\sum n^{-3}$.

Example 3.3.1.

The term $a_j = (j^4+1)^{-1/2}$ behaves, for large j, like $b_j = j^{-2}$, whose sum is $\pi^2/6$. Thus

$$\sum_{j=1}^{\infty} a_j = \pi^2/6 + \sum_{j=1}^{\infty}\big((j^4+1)^{-1/2} - j^{-2})\big) = 1.64493 - 0.30119 = 1.3437.$$

The terms on the right hand side are easily computed; the use of five of these is sufficient for four-place accuracy in the final result. Using the series on the left hand side, one would not get four-place accuracy until after 20,000 terms.

This technique is unusually successful in this example. The reader is advised to find out that and why it is less successful for $a_j = (j^4 + j^3 + 1)^{-1/2}$.

A usual kind of comparison sequence is a geometric sequence $y_n = a + bk^n$, fitted to the three most recently computed terms of a given sequence. If we put $y_n = s_n$ for (say) $n = j, j-1, j-2$, then $\nabla y_j = \nabla s_j$, $\nabla y_{j-1} = \nabla s_{j-1}$, and

$$k = \nabla s_j/\nabla s_{j-1}, \quad bk^j - bk^{j-1} = y_j - y_{j-1} = \nabla s_j.$$

Hence

$$bk^j = \frac{\nabla s_j}{1 - 1/k} = \frac{\nabla s_j}{1 - \nabla s_{j-1}/\nabla s_j} = \frac{(\nabla s_j)^2}{\nabla^2 s_j}.$$

This yields a comparison sequence for each j. Suppose that $|k| < 1$. Then the comparison sequence has a limit s'_j (say), and we have $s'_j = \lim_{n\to\infty} y_n = a = y_j - bk^j$, i.e.,

$$s \approx s'_j = s_j - \frac{(\nabla s_j)^2}{\nabla^2 s_j}. \tag{3.3.1}$$

This is called **Aitken acceleration**. A variant that works with the terms of a series instead of its partial sums is given in Problem 12.

If $\{s_n\}$ is exactly a geometric sequence, i.e., if $s_n = y_n \ \forall n$, then $s'_j = s \ \forall j$. Otherwise it can be shown (Henrici [19, 1964]) that under the assumptions

$$\lim_{j\to\infty} s_j = s, \quad \text{and} \quad \lim \frac{s_{j+1} - s_j}{s_j - s_{j-1}} = k^*, \quad |k^*| < 1,$$

the sequence $\{s'_j\}$ converges faster than does the sequence $\{s_j\}$. The above assumptions can often be verified for sequences arising from iterative processes and for many other applications.

The condition $|k^*| < 1$ is a *sufficient* condition only. In practice, Aitken acceleration seems *most efficient if* $k^* = -1$. Indeed, it often converges even if $k^* < -1$, see Problem 6. It is *much less successful if* $k^* \approx 1$, e.g., for slowly convergent series with positive terms.

The Aitken acceleration process can often be *iterated*, to yield sequences, $\{s''_n\}_0^\infty$, $\{s'''_n\}_0^\infty$, etc., defined by the formulas

$$s''_j = s'_j - \frac{(\nabla s'_j)^2}{\nabla^2 s'_j}. \ s'''_j = s''_j - \frac{(\nabla s''_j)^2}{\nabla^2 s''_j} \ldots$$

Example 3.3.2.

By (3.1.8), it follows for $x = 1$ that

$$1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots = \arctan 1 = \pi/4 \approx 0.7853981634.$$

This series converges very slowly. Even after 500 terms there still occur changes in the third decimal. Consider the partial sums $s_j = \sum_{n_0}^{j} (-1)^j (2n+1)^{-1}$, with $n_0 = 5$, and compute the **iterated Aitken** sequences as indicated above.

The (sufficient) theoretical condition mentioned above is not satisfied, since $\nabla s_n / \nabla s_{n-1} \to -1$ as $n \to \infty$. Nevertheless, we shall see that the Aitken acceleration works well, and that the iterated accelerations converge rapidly. One gains two digits for every pair of terms in spite of the slow convergence of the original series. The results below were obtained using a computer with `macheps` $\approx 10^{-16}$. The errors of s'_j, s''_j, ... are denoted e'_j, e''_j, ...

j	s_j	e_j	e'_j	e''_j	e'''_j
5	0.744012 ...	−4.1387e−2			
6	0.820935 ...	3.5536e−2			
7	0.754268 ...	−3.1130e−2	−1.7783e−4		
8	0.813092 ...	2.7693e−2	1.1979e−4		
9	0.760460 ...	−2.4938e−2	−8.4457e−5	−1.3332e−6	
10	0.808079 ...	2.2681e−2	6.1741e−5	7.5041e−7	
11	0.764601 ...	−2.0797e−2	−4.6484e−5	−4.4772e−7	−1.0289e−8

Example 3.3.3.

Consider the iteration formula $s_{n+1} = \phi(s_n)$, with $s_0 = 0.8$, $\phi(s_n) = 1 - 0.5s_n^2$, $n = 0, 1, 2, \ldots$. We obtain $s_1 = 0.68$, $s_2 = 0.7688$, and $s'_2 = 0.7688 - (0.0888)^2/0.2088 = 0.73103$. The error in s'_2 is only about 3% of the error in s_2.

In this example the theoretical criterion mentioned above is satisfied, since one can show that $s_n \to a = \sqrt{3} - 1 \approx 0.73205$, $\nabla s_n / \nabla s_{n-1} \to \phi'(a) = -a$, as $n \to \infty$. (See the discussion of iteration in Sec. 1.2.1.)

Iterated Aitken yields, with $u = 2^{-53} = 1.1{\cdot}10^{-16}$ the results $s_2' = 0.73103\ldots$, $s_4'' = 0.73211$, etc., and the errors

$$e_2' = -10\,10^{-3},\ e_4'' = 5.6\,10^{-5},\ e_6''' = 7.8\,10^{-7},\ e_8^{(4)} = -2.8\,10^{-9},$$

after $2, 4, 6, 8$ evaluations of the function ϕ, respectively.

We shall see (Problem 6) that the Aitken acceleration may converge, even if the basic iteration diverges.

When the basic sequence $\{s_n\}$ is, as in this example, produced by a convergent iterative process, one can apply Aitken acceleration in a different way that is usually even much more efficient. We call this **active Aitken acceleration**, since the result of an acceleration is actively used in the basic iterative process, i.e., one computes $s_3 = \phi(s_2')$, $s_4 = \phi(s_3)$, and makes an Aitken acceleration by means of s_2', s_3, s_4 that yields s_4'. This can evidently be repeated, until some termination criterion is satisfied. See Problem 13.

Example 3.3.4.

Set $a_n = e^{-\sqrt{n+1}}$, $n \ge 0$. As before, the s_n are the partial sums of $\sum a_n$, $s = \lim s_n = 1.67040681796634$, and use the same notations as above. Note that $\nabla s_n / \nabla s_{n-1} = a_n / a_{n-1} \approx 1 - \frac{1}{2} n^{-1/2}$, $(n \gg 1]$, so this series is slowly convergent. Computations with plain and iterated Aitken, `macheps` $\approx 10^{-16}$, gave the results below:

j	e_{2j}	$e_{2j}^{(j)}$
0	−1.304	−1.304
1	−8.82e−1	−4.10e−1
2	−6.40e−1	−1.08e−1
3	−4.83e−1	−3.32e−2
2	−3.74e−1	−4.41e−3
5	−2.95e−1	−7.97e−4
6	−2.37e−1	−1.29e−4
7	−1.92e−1	−1.06e−5
8	−1.58e−1	−1.13e−5

The sequence $\{e_{2j}^{(j)}\}$ is monotonic until $j = 8$. After this $|e_{2j}^{(j)}|$ is mildly fluctuating around 10^{-5} (at least until $j = 24$), and the differences $\nabla s_{2j}^{(j)} = \nabla e_{2j}^{(j)}$ are sometimes several powers of 10 smaller than the actual errors and are misleading as error estimates. The rounding errors have taken over, and it is almost no use to compute more terms.

It is possible to use more terms for obtaining higher accuracy by applying iterated Aitken acceleration to a **thinned sequence** e.g., $s_4, s_8, s_{12}, \ldots$, Problem 3. [69] The convergence ratios of the thinned sequence are much smaller; for the series of

[69]Note the thinning is performed on a *sequence* that converges to the limit to be computed, e.g.,

the previous example they become approximately $(1-\frac{1}{2}n^{-1/2})^4 \approx 1-2n^{-1/2}, n \gg 1$. The most important point is, though, that the rounding errors become more slowly amplified, so that terms far beyond the eighth number of the unthinned sequence can be used in the acceleration, resulting in a much improved final accuracy.

How to realize the thinning depends on the sequence; a different thinning will be used in the next example.

Example 3.3.5.

We shall compute

$$s = \sum n^{-3/2} = 2.612375348685488, \quad n_0 = 1, \quad \texttt{macheps} = 10^{-16}.$$

If all partial sums are used in Aitken acceleration, it turns out that the error $|e_{2j}^{(j)}|$ is decreasing until $j = 5$, when it is 0.07, and it remains on approximately this level for a long time.

A much better result is obtained by means of thinning, but since the convergence is much slower here than in the previous case, we shall try "geometric" thinning rather than the "arithmetic" thinning used above, i.e. we now set $S_m = s_{2^m}$. Then

$$\nabla S_m = \sum_{1+2^{m-1}}^{2^m} a_n, \quad S_j = S_0 + \sum_{m=1}^{j} \nabla S_m, \quad E_j = S_j - s.$$

(If maximal accuracy is wanted, it may be advisable to use the "divide and conquer technique" for computing these sums, see Problem 2.4.1, but it has not been used here.) By the approximation of the sums by integrals one can show that $\nabla S_m / \nabla S_{m-1} \approx 2^{-1/2}$, $m \gg 1$. The table below shows the errors of the first thinned sequence and the results after iterated Aitken acceleration. The last result has used 1024 terms of the original series, but since

$$s_n - s = -\sum_{j=n}^{\infty} j^{-3/2} \approx -\int_n^{\infty} t^{-3/2}\, dt = -\frac{2}{3} n^{-1/2}, \tag{3.3.2}$$

10^{20} terms would have been needed for obtaining this accuracy without convergence acceleration.

j	E_{2j+1}	$E_{2j+1}^{(j)}$
0	−1.61	−1.61
1	−0.94	−1.85
2	−4.92e−1	−5.06e−2
3	−2.49e−1	−2.37e−4
4	−1.25e−1	−2.25e−7
5	−6.25e−2	2.25e−10

the partial sums of a series. Only in so-called *bell sums*, see Problem 36, we shall do *a completely different kind of thinning*, namely a thinning of the *terms* of a series.

For sequences such that $s_n - s = c_0 n^{-p} + c_1 n^{-p-1} + O(n^{-p-2})$, $p > 0$, where s, c_0, c_1 are unknown, the following variant of Aitken acceleration, is more successful, [3]:

$$s'_n = s_n - \frac{p+1}{p} \frac{\Delta s_n \nabla s_n}{\Delta s_n - \nabla s_n}. \tag{3.3.3}$$

It turns out that s'_n is two powers of n more accurate than s_n,

$$s'_n - s = O(n^{-p-2}),$$

see Problem 14. More generally, suppose that there exists a longer (unknown) asymptotic expansion of the form

$$s_n = s + n^{-p}(c_0 + c_1 n^{-1} + c_2 n^{-2} + \ldots), \quad n \to \infty. \tag{3.3.4}$$

This is a rather common case (see Problem 15). Then we can extend this to an to an *iterative variant*, where p is to be increased by 2 in each iteration; $i = 0, 1, 2, \ldots$ is a superscript, i.e.,

$$s_n^{i+1} = s_n^i - \frac{p+2i+1}{p+2i} \frac{\Delta s_n^i \nabla s_n^i}{\Delta s_n^i - \nabla s_n^i}. \tag{3.3.5}$$

If p is also unknown, it can be estimated by means of the equation,

$$\frac{1}{p+1} = -\Delta \frac{\Delta s_n}{\Delta s_n - \nabla s_n} + O(n^{-2}). \tag{3.3.6}$$

Example 3.3.6.

We consider the same series as in the previous example, i.e. $s = \sum n^{-3/2}$. We use (3.3.5) without thinning. Here $p = -1/2$, see Problem 15. As usual, the errors are denoted $e_j = s_j - s$, $e_{2j}^i = s_{2j}^i - s$. In the right column of the table, we show the errors from a computation with 12 terms of the original series, `macheps` $\approx 10^{-16}$.

j	e_{2j}	e_{2j}^j
0	−1.612	−1.612
1	−1.066	−8.217e−3
2	−8.52e−1	−4.617e−5
3	−7.30e−1	+2.528e−7
4	−6.49e−1	−1.122e−9
5	−5.90e−1	−6.34e−12
6	−5.44e−1	−1.322e−9

From this point the errors were around 10^{-10} or a little below. The rounding errors have taken over, and the differences are, as in Example 3.3.4, misleading for error estimation. If needed, higher accuracy can be obtained by "arithmetic thinning" with more terms.

In this computation only 12 terms were used. In the previous example a less accurate result was obtained by means of 1024 terms of the same series, but we must appreciate that the technique of Example 3.3.5 did not require the existence of an asymptotic expansion for s_n and may therefore have a wider range of application.

The best known extensions of Aitken acceleration are based on comparison sequences of the form

$$y_n = s + \sum_{m=1}^{p} \alpha_m k_m^n; \tag{3.3.7}$$

there are thus $2p+1$ parameters to be determined by $2p+1$ elements of the sequence, (say) $n = j : -1 : j - 2p$. The parameters may be complex. The first algorithms of this type were developed by Shanks around 1950, and a few years later a very elegant solution was found in the **ϵ-algorithm** of Wynn [39]. Since

$$\nabla y_n = \sum_{m=1}^{p} \alpha'_m k_m^n, \quad \alpha'_m = \alpha_m(1 - k_m^{-1}), \tag{3.3.8}$$

can be interpreted as the n'th coefficient of the power series for a rational function, with poles at (the unknown points) k_m, $m = 1, 2, \dots p$, we postpone the discussion till Sec. 3.4, where the related subject of *Padé approximation* is treated.

We have seen that iterated Aitken accelerations work excellently for oscillating sequences but, without modification, they are not so efficient for monotonic sequences.

There are not yet so many theoretical results that give justice to the practically observed efficiency of iterated Aitken accelerations for oscillating sequences. One reason for this can be that the transformation (3.3.1), which the algorithms are based on, is *nonlinear*). We shall in the following subsections study methods of convergence acceleration that are based on *linear* transformations, where theoretical estimates of convergence rates and errors are closer to to the practical performance of the methods.

3.3.3 Alternating Series and Complex Power Series

Example 3.3.7.

Consider again the same series as in Example 3.3.2, i.e..

$$\sum_{j=0}^{\infty}(-1)^j(2j+1)^{-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \frac{\pi}{4} = 0.7853981634.$$

We shall now apply another method of acceleration that is based on **repeated averaging** of the partial sums. Let S_N be the sum of the first N terms. The columns to the right of the S_N-column in the scheme given in Table 3.2.1 are formed by building averages.

Each number in a column is the mean of the two numbers which stand to the left and upper left of the number itself. In other words, each number is the mean of its "west" and "northwest" neighbor. The row index of M tells how many terms are used from the original series, while the column index -1 equals the number of averagings. Only the digits which are different from those in the previous column are written out.

Table 3.3.1. *Summation by repeated averaging.*

N	S_N	M_2	M_3	M_4	M_5	M_6	M_7
6	0.744 012						
7	0.820 935	782 474					
8	0.754 268	787 602	5038				
9	0.813 092	783 680	5641	340			
10	0.760 460	786 776	5228	434	387		
11	0.808 079	784 270	5523	376	405	396	
12	0.764 601	786 340	5305	414	395	400	398

Notice that the values in each column oscillate. In general, for an alternating series, it follows from the next theorem together with (3.2.4) that *if the absolute value of the* jth *term, considered as a function of j, has a* kth *derivative which approaches zero monotonically for $j > N_0$, then every other value in column M_{k+1} is larger than the sum, and every other is smaller.* The above premise is satisfied here, since if $f(j) = (2j+1)^{-1}$ then $f^{(k)}(j) = c_k(2j+1)^{-1-k}$, which approaches zero monotonically.

If round-off is ignored, it follows from column M_6 that $0.785396 \le \pi/4 \le 0.785400$.

To take account of round-off error, we set $\pi/4 = 0.785398 \pm 3 \cdot 10^{-6}$. The actual error is only $1.6\,10^{-7}$. In Example 3.3.2 iterated Aitken accelerations gave about one decimal digit more with the same data.

It is evident how the above method can be applied to any *alternating series*. It is often, though not always, equally successful. It is equivalent to a particular case of the **Euler transformation**, which is defined for the convergence acceleration of the following general complex power series.

$$S(z) = \sum_{j=1}^{\infty} u_j z^{j-1}, \tag{3.3.9}$$

The alternating series obtained for $z = -1$. Other applications include *Fourier series*. They can be brought to this form, with $z = e^{i\phi}$, $-\pi \le \phi \le \pi$, see Problem 16 and Example 3.3.9. The irregular errors of the coefficients play a big role if $|\phi| \ll \pi$, and it is important to reduce their effects by means of a variant of the *thinning* technique, described (for Aitken acceleration) in the previous subsection. Another interesting application is the *analytic continuation* of the power series outside its circle of convergence, see Example 3.3.10.

Theorem 3.3.1.

The tail of the power series in Eq.(3.3.9) *can formally be transformed into the expansion,* $(z \neq 1)$.

$$S(z) - \sum_{j=1}^{n} u_j z^{j-1} = \sum_{j=n+1}^{\infty} u_j z^{j-1} = \frac{z^n}{1-z} \sum_{s=0}^{\infty} P^s u_{n+1}, \tag{3.3.10}$$

where $P^0 = 1$, $P = \frac{z\Delta}{1-z}$. *For* $n = 0$, *this is the* **classical Euler transformation**. *Set* $N = n + k - 1$, *and set*

$$M_{n,1} = \sum_{j=1}^{n} u_j z^{j-1}; \quad M_{N,k} = M_{n,1} + \frac{z^n}{1-z} \sum_{s=0}^{k-2} P^s u_{n+1}; \quad n = N - k + 1. \tag{3.3.11}$$

These quantities can be computed by the following recurrence formula that yields several estimates based on N terms from the original series. [70] *This is called the* **generalized Euler Transformation**.

$$M_{N,k} = \frac{M_{N,k-1} - zM_{N-1,k-1}}{1-z}, \quad k = 2, 3, 4 \ldots, N. \tag{3.3.12}$$

For $z = -1$, *this is the repeated average algorithm described above, and* $P = -\frac{1}{2}\Delta$.

Assume that $|z| \leq 1$, *that* $\sum u_j z^{j-1}$ *converges, and that* $\Delta^s u_N \to 0$, $s = 0 : k$ *as* $N \to \infty$. *Then* $M_{N,k} \to S(z)$, *as* $N \to \infty$.

If, moreover, $\Delta^{k-1} u_j$ *has a constant sign for* $j \geq N - k + 2$, *then strict error bounds are obtained:*

$$|M_{N,k} - S(z)| \leq |z(M_{N,k} - M_{N-1,k-1})| = |M_{N,k} - M_{N,k-1}|, \ (k \geq 2). \tag{3.3.13}$$

Proof. We first note that, as $N \to \infty$, $P^s u_N \to 0$, $s = 0 : k$, and hence, by (3.3.11), $\lim M_{N,k} = \lim M_{N,0} = S(z)$.

Euler's transformation can be formally derived by operators as follows:

$$\begin{aligned} S(z) - M_{n,1} &= z^n \sum_{i=0}^{\infty} (zE)^i u_{n+1} = \frac{z^n}{1 - zE} u_{n+1} \\ &= \frac{z^n}{1 - z - z\Delta} u_{n+1} = \frac{z^n}{1-z} \sum_{s=0}^{\infty} P^s u_{n+1}. \end{aligned}$$

In order to derive (3.3.12), note that this relation can equivalently be written thus,

$$M_{N,k} - M_{N,k-1} = z(M_{N,k} - M_{N-1,k-1}), \tag{3.3.14}$$

$$M_{N,k-1} - M_{N-1,k-1} = (1-z)(M_{N,k} - M_{N-1,k-1}). \tag{3.3.15}$$

Remembering that $n = N - k + 1$, we obtain, by (3.3.11),

$$M_{N,k} - M_{N-1,k-1} = \frac{z^{N-k+1}}{1-z} P^{k-2} u_{N-k+2}, \tag{3.3.16}$$

[70] See Algorithm 3.3.1 for an adaptive choice of a kind of optimal output.

and it can be shown (Problem 19) that

$$M_{N,k-1} - M_{N-1,k-1} = z^n P^{k-2} u_{n+1} = z^{N-k+1} P^{k-2} u_{N-k+2}. \tag{3.3.17}$$

By (3.3.16) and (3.3.17), we now obtain (3.3.15) and hence also the equivalent equations (3.3.14) and (3.3.12).

Now substitute j for N into (3.3.17), and add the p equations obtained for $j = N+1, \ldots, N+p$. We obtain: $M_{N+p,k-1} - M_{N,k-1} = \sum_{j=N+1}^{N+p} z^{j-k+1} P^{k-2} u_{j-k+2}$. Then substitute $k+1$ for k, and $N+1+i$ for j. Let $p \to \infty$, while k is fixed. It follows that

$$S(z) - M_{N,k} = \sum_{j=N+1}^{\infty} z^{j-k} P^{k-1} u_{j-k+1} = \frac{z^{N-k+1} \cdot z^{k-1}}{(1-z)^{k-1}} \sum_{i=0}^{\infty} z^i \Delta^{k-1} u_{N-k+2+i}, \tag{3.3.18}$$

hence $|S(z) - M_{N,k}| \le |(z/(1-z))^{k-1} z^{N-k+1}| \sum_{i=0}^{\infty} |\Delta^{k-1} u_{N-k+2+i}|$. We now use the assumption that $\Delta^{k-1} u_j$ has constant sign for $j \ge N-k+2$.
Since $\sum_{i=0}^{\infty} \Delta^{k-1} u_{N-k+2+i} = -\Delta^{k-2} u_{N-k+2}$, it follows that

$$|S(z) - M_{N,k}| \le \left| z^{N-k+1} \frac{z^{k-1} \Delta^{k-2} u_{N-k+2}}{(1-z)^{k-1}} \right| = \left| \frac{z \cdot z^{N-k+1}}{1-z} P^{k-2} u_{N-k+2} \right|.$$

Now, by (3.3.16), $|S(z) - M_{N,k}| \le |z| \cdot |M_{N,k} - M_{N-1,k-1}|$. This is the first part of Eq.(3.3.13). The second part then follows from (3.3.14). $\square$

COMMENTS:

- The elements $M_{N,k}$ become rational functions of z for fixed N, k.
- If the term u_n, as a function of n, belongs to $\mathcal{P}_k$, then the classical Euler transformation (for $n = 0$) yields the exact value of $S(z)$ after k terms, if $|z| < 1$. This follows from (3.3.10), because $\sum u_j z^j$ is convergent, and $P^s u_{n+1} = 0$ for $s \ge k$. In this particular case, $S(z) = Q(z)(1-z)^{-k}$, where Q is a polynomial; in fact the Euler transformation gives $S(z)$ correctly for all $z \ne 1$.

The advantage of the recurrence formula (3.3.12), instead of a more direct use of (3.3.10), is that it provides a whole lower triangular matrix of estimates, so that the following algorithm can, by means of a simple test, decide when to stop. That is what we call the **generalized Euler Transformation**.

This yields a result with strict error bound, if $\Delta^{k-1} u_j$ has a constant sign (for all j with a given k), and if the effect of rounding errors is evidently smaller than TOL. If these conditions are not satisfied, there is a small risk that the algorithm may terminate if the error estimate is incidentally small, e.g., near a sign change of $\Delta^{k-1} u_j$.

The irregular errors of the initial data are propagated to the results, in a way that is discussed in Ch.11, see in particular Example 11.1.2. In the long run, they are multiplied by approximately $|z/(1-z)|$ from a column to the next—this is less than one if $\Re z < 1/2$—but in the beginning this growth factor can be as large as $(1+|z|)/|1-z|$. It plays no role for alternating series; its importance when $|1-z|$ is smaller will be commented in Example 3.3.9.

The following algorithm is mainly based on the above theorem, but the possibility for the irregular errors to become dominant has been taken into account (somewhat) in the third alternative of the termination criterion.

Algorithm 3.3.1 [
The Generalized Euler Transformation]
This algorithm is based on Theorem 3.3.1, with a tolerance named TOL, and a termination criterion based on (3.3.13), by the computation and inspection of the elements of M in a certain order, until it finds a pair of neighboring elements that satisfies the criterion.
The classical Euler Transformation would only consider the diagonal elements M_{NN}, $N = 1, 2, ...$ and the termination would have been based on $|M_{NN} - M_{N-1,N-1}|$. The theory in Sec. 3.3.6 will explain why the strategy used in this algorithm is superior for an important class of series.
Give: $maxN$ and the coefficients u_j, $j = 1 : maxN$.

```
$N := 0$;   $errest := 1000$;   allocate storage for $M$;
while( $errest > \text{TOL}$)&($N \le Nmax$)&($errest < olderrest$)
  $N := N + 1; olderrest := errest$;
  if $N = 1$,   $M_{1,1} := u_1$;
  else   $M_{N,1} := M_{N-1,1} + u_N * z^{N-1}$;
  end
  for $k = 2 : N$,
    $M_{N,k} := (M_{N,k-1} - z * M_{N-1,k-1})/(1 - z)$;
    $temp := |M_{N,k} - M_{N,k-1}|/2$;
    if $temp < errest$,   $kk := k$;   $errest := temp$;   end
  end
end
$sum := (M_{N,kk} + M_{N,kk-1})/2$;   $NN := N$;
```

Output: sum, $errest$, N, kk, and perhaps a few rows with $M_{N,k}$.

An oscillatory behavior of the values $|M_{N,k} - M_{N,k-1}$ in the same row, indicates that the irregular errors have become dominant. The smallest error estimates may then become unreliable.

The above algorithm gives a strict error bound if, in the notation used in the theorem, $\Delta^{k-1}u_i$ has a constant sign for $i \ge N - k + 2$ (in addition to the other conditions of the theorem). It may seem difficult to check if this condition is satisfied. A sequence, for which this condition is satisfied *for every* k, is called **completely monotonic**.

Properties of such sequences, and simple criteria related to them will be given in § 3.3.6. It turns out that many sequences that can be formed from sequences like $\{n^{-\alpha}\}$, $\{e^{-\alpha n}\}$ by simple operations and combinations, belong to this class. The generalized Euler transformation yields a sequence that converges at least as

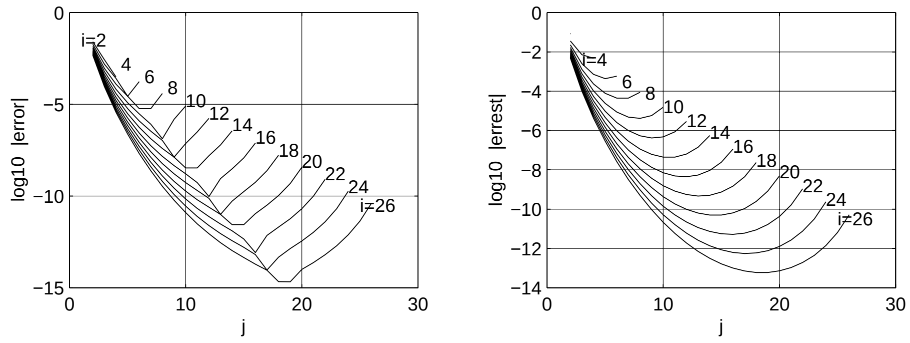


Figure 3.3.1. *Logarithms of the actual errors and the error estimates for $M_{N,k}$ in a more extensive computation for the alternating series in Example 3.3.7. These graphs are typical for alternating series with completely monotonic terms. The tolerance is here set above the level, where the irregular errors become important; for a smaller tolerance parts of the lowest curves may become less smooth in some parts.*

fast as a geometric series. The convergence ratio depends on z; it is less than one in absolute value for any complex z, except for $z > 1$ on the real axis. So, the generalized Euler transformation often provides an analytic continuation of a power series outside its circle of convergence.

For *alternating series*, with completely monotonic terms i.e., for $z = -1$, the convergence ratio typically becomes $\frac{1}{3}$. This is in good agreement with Fig. 3.3.1. See Theorem 3.3.9 in § 3.3.6 that also explains why the minimum points for the errors lie almost on a straight line in Fig. 3.3.1 , and why the optimal value of k/N is approximately $\frac{2}{3}$, if $N \gg 1$, and if there are no irregular errors. Hopefully this optimum is close to the ratio is close to the ratio kk/N that is found by the termination criterion. In practice it is usually successful, unless the irregular errors have become dominant.

Example 3.3.8. A program, essentially the same as Algorithm 3.3.1, is applied to the series

$$\sum_{j=1}^{\infty}(-1)^j j^{-1} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \ldots = \ln 2 = 0.6931471805599453.$$

with $\text{Tol} = 10^{-6}$, `macheps` $\approx 10^{-16}$. It stops when $N = 12$, $kk = 9$. The errors $e_k = M_{N,k} - \ln 2$ and the differences $\frac{1}{2}\nabla_k M_{N,k}$ along the last row of M read:

k	1	2	3	...	8	9	10	11	12
e_k	-3.99e-2	1.73e-3	-1.64e-4	...	5.35e-7	-4.51e-7	5.35e-7	-9.44e-7	2.75e-6
$\nabla/2$		2.03e-2	-9.47e-4	...	7.05e-7	-4.93e-7	4.93e-7	-7.40e-7	1.85e-6

Note that $|errest| = 4.93\,10^{-7}$ and $sum - \ln 2 = \frac{1}{2}(e_9 + e_8) = 4.2\,10^{-8}$.

Almost full accuracy is obtained for $\text{TOL} = 10^{-16}$, $maxN = 40$. The results are $N = 32$, $kk = 22$, $errest = 10^{-16}$, $|error| = 2\,10^{-16}$. Note that $errest < |error|$; this can happen when we ask for such a high accuracy that the rounding errors are not negligible.

Example 3.3.9. *Application to Fourier series.* Consider a complex power series $S(z) = \sum_{n=1}^{\infty} u_n z^{n-1}$, $z = e^{i\phi}$. A Fourier series that is originally of the form $\sum_{-\infty}^{\infty}$ or in trigonometric form, can easily be brought to this form, see Problem 16. As we shall see, the results can often be improved considerably by the application of *thinning.* Let THIN be a positive integer. The thinned form of $S(z)$ reads

$$S(z) = \sum_{p=1}^{\infty} u_p^* z^{\text{THIN}\cdot(p-1)}, \qquad u_p^* = \sum_{j=1}^{thin} u_{j+thin\cdot(p-1)}\, z^{j-1}.$$

For example, if $z = e^{i\pi/3}$ and THIN $=3$, the series becomes an alternating series, perhaps with complex coefficients. It does not matter in the numerical work that u_p^* depends on z.

We consider the case $S(z) = -\ln(1-z)/z = \sum z^{n-1}/n$, which is typical for a power series with completely monotonic terms. Numerical computation, essentially by the above algorithm, gave the following results. The coefficients u_j are computed with macheps$=10^{-16}$. We make the rounding errors during the computations less important by subtracting the first row of partial sums by its last element; it is, of course, added again to the final result. [71] The first table shows, for various ϕ, the most accurate result that can be obtained without thinning. These limits are due to the rounding errors; we can make the pure truncation error arbitrarily small by choosing N large enough.

ϕ	π	$2\pi/3$	$\pi/2$	$\pi/3$	$\pi/4$	$\pi/6$	$\pi/8$	$\pi/12$	$\pi/180$
|error|	2e-16	8e-16	1e-14	6e-12	1e-9	7e-8	5e-7	3e-5	2e-1
N	30	33	36	36	36	36	40	40	100
kk	21	22	20	21	20	14	13	10	(3)

Note that a rather good accuracy is obtained also for $\phi = \pi/8$ and $\phi = \pi/12$, where the algorithm is "unstable", since $|\frac{z}{1-z}| > 1$. In this kind of computations "instability" does not mean that the algorithm is hopeless, but it shows the importance of a good termination criterion. The question is to navigate safely between Scylla and Charybdis. For a small value like $\phi = \pi/180$, the sum is approximately $4.1 + 1.5i$. The smallest error with 100 terms (or less) is 0.02; it is obtained for $k = 3$. Also note that kk/N increases with ϕ.

By *thinning*, much better results are obtained for $\phi \ll \pi$, in particular for $\phi = \pi/180$. This series that has "essentially positive" terms originally can become "essentially alternating" by thinning. We present the errors obtained for four values of the parameter THIN, with different amount of work. Compare $|error|$, kk, etc. with appropriate values in the table above. We see that, by thinning, it is possible to calculate the Fourier series very accurately also for small values of ϕ.

[71]Tricks like this can often be applied in linear computations with a slowly varying sequence of numbers. See e.g., the discussion of rounding errors in Richardson extrapolation in Sec. 3.3.5.

THIN	180	120	90	15
$thin \cdot \phi$	π	$2\pi/3$	$\pi/2$	$\pi/12$
|error|	2e-14	1e-14	3e-13	3e-5
N	28	31	33	41
kk	20	22	18	10
total no. terms	5040	3720	2970	615

Roughly speaking, the optimal convergence rate of the Euler Transformation depends on z in the same way for all power series with completely monotonic coefficients; independently of the rate of convergence of the original series. The above tables from a particular example can therefore—with some safety margin—be used as a guide for the application of the Euler transformation with thinning to any series of this class.

Say that you want the sum of a series $\sum u_n z^n$ for $z = e^{i\phi}$, $\phi = \pi/12$, with relative $|error| < 10^{-10}$. You see in the first table that $|error| = 6\,10^{-12}$ for $\phi = \pi/3 = 4\pi/12$ without thinning. The safety margin is hopefully large enough. Therefore, try $Thin = 4$. We make two tests with completely monotonic terms: $u_n = n^{-1}$ and $u_n = \exp(-\sqrt{n})$. $Tol = 10^{-10}$ is hopefully large enough to make the irregular errors relatively negligible. In both tests the actual $|error|$ turns out to be $4\,10^{-11}$, and the total number of terms is $4 \cdot 32 = 128$. The values of *errest* are $6\,10^{-11}$ and $7\,10^{-11}$; both slightly overestimate the actual errors and are still smaller than Tol.

Example 3.3.10. *Application to a divergent power series, (analytic continuation).* Consider a complex power series $S(z) = \sum_{n=1}^{\infty} u_n z^{n-1}$, $|z| > 1$. As in the previous example we study in detail the case of $u_n = 1/n$. It was mentioned above that the generalized Euler transformation theoretically converges in the z-plane, cut along the interval $[1, \infty]$. The limit is $-z^{-1}\ln(1-z)$, a single-valued function in this region. For various z outside the unit circle, we shall see that rounding causes bigger problems here than for Fourier series. The error estimate of Algorithm 3.3.1, usually underestimated the error, sometimes by a factor of ten. The table reports some results from experiments without thinning.

z	-2	-4	-10	-100	-1000	$2i$	$8i$	$1+i$	$2+i$
|error|	2e-12	2e-8	4e-5	3e-3	5e-2	8e-11	1e-3	1e-7	2e-2
N	38	41	43	50	51	40	39	38	39
kk	32	34	39	50	51	28	34	22	24

Thinning can be applied also in this application, but here not only the argument ϕ is increased (this is good), but also $|z|$ (this is bad). Nevertheless, for $z = 1 + i$, the error becomes 10^{-7}, $3\,10^{-9}$, 10^{-9}, $4\,10^{-8}$, for $thin = 1, 2, 3, 4$, respectively. For $z = 2 + i$, however, thinning improved the error only from 0.02 to 0.01. All this is for macheps$= 10^{-16}$.

Finally, we mention an application of the generalized Euler transformation to a *semiconvergent* expansion.

Example 3.3.11. Recall Example 3.1.15 with the semiconvergent expansion $\sum_{j=1}^{\infty}(j-1)!(-x)^{-j}$. For $x=5$ the first terms decrease, but after the fifth term they increase; the error of the average of the fifth and the sixth term is $1.8\,10^{-4}$.

We now compute ten terms and apply repeated averaging. $M_{9,7}$ and $M_{7.5}$ are the best values, their errors are, respectively, $-2.4\,10^{-6}$ and $2.6\,10^{-6}$; we thus gain two decimal digits. The error estimation and the termination criterion must be modified, but that is outside the scope of this text.

We shall encounter several other methods for alternating series and complex power series, which are even more efficient than the generalized Euler transformation, see the GCA method in § 3.3.7, the epsilon algorithm in § 3.4.4, and the methods of Plana and Lindelöf in Ch.12.

3.3.4 Euler–Maclaurin's Formula with Applications

In the summation of series with essentially positive terms the tail of the sum can be approximated by an integral by means of the trapezoidal rule.

Example 3.3.12.

Consider the sum $S=\sum_{j=1}^{\infty}j^{-2}$. The sum of the first nine terms is, to four decimal places, 1.5398. It immediately occurs to one to compare the tail of the series with the integral of x^{-2} from 10 to ∞. We approximate the integral according to the trapezoidal rule, see Sec. 1.2 and Fig. 3.3.2:

$$\int_{10}^{\infty}x^{-2}\,dx \approx T_1+T_2+T_3+\ldots=\frac{1}{2}(10^{-2}+11^{-2})$$
$$+\frac{1}{2}(11^{-2}+12^{-2})+\frac{1}{2}(12^{-2}+13^{-2})+\ldots=\sum_{j=10}^{\infty}j^{-2}-\frac{1}{2}10^{-2}.$$

Hence it follows that

$$\sum_{j=1}^{\infty}j^{-2}=\sum_{j=1}^{9}j^{-2}+\sum_{j=10}^{\infty}j^{-2}$$
$$\approx 1.5398+[-x^{-1}]_{10}^{\infty}+0.0050=1.5398+0.1050=1.6448.$$

The correct answer is $\pi^2/6=1.64493406684823$. We would have needed about 10,000 terms to get the same accuracy by direct addition of the terms!

The above procedure is not a coincidental trick, but a very useful method. A further systematic development of the idea leads to the important Euler–Maclaurin summation formula. We first derive this heuristically by operator techniques and exemplify its use, including a somewhat paradoxical example that shows that a strict treatment with the consideration of the remainder term is necessary for very practical reasons. Since this formula has several other applications, e.g., in numerical integration, we formulate it more generally than needed for the summation of infinite series.

Figure 3.3.2. *Comparison with an integral*

Consider to begin with a rectangle sum on the finite interval $[a,b]$, with n steps of equal length h, $a+nh=b$; with the operator notation introduced in Sec. 3.2.2.

$$h\sum_{i=0}^{n-1} f(a+ih) = h\sum_{i=0}^{n-1} E^i f(a) = h\frac{E^n-1}{E-1}f(a) = \frac{(E^n-1)}{D}\frac{hD}{e^{hD}-1}f(a).$$

We apply, to the second factor, the expansion derived in Example 3.1.6, with the Bernoulli numbers B_ν. (Recall that $a+nh=b$, $E^n f(a)=f(b)$, etc.)

$$\begin{aligned} h\sum_{i=0}^{n-1} f(a+ih) &= \frac{(E^n-1)}{D}\left(1+\sum_{\nu=1}^{\infty}\frac{B_\nu(hD)^\nu}{\nu!}\right)f(a) \qquad (3.3.19)\\ &= \int_a^b f(x)\,dx + \sum_{\nu=1}^{k}\frac{h^\nu B_\nu}{\nu!}\left(f^{(\nu-1)}(b)-f^{(\nu-1)}(a)\right) + R_{k+1}. \end{aligned}$$

Here R_{k+1} is a remainder term that will be discussed thoroughly in Theorem 3.3.3. Set $h=1$, and assume that $f(b)$, $f'(b)$, ... tend to zero as $b\to\infty$. Recall that $B_1=-\frac{1}{2}$, $B_{2j+1}=0$ for $j>0$, and set $k=2r+1$. This yields **Euler–Maclaurin's summation formula** footnoteL.Euler (1707-1783), incredibly prolific Swiss mathematician. He gave fundamental contributions to many branches of mathematics and to the mechanics of rigid and deformable bodies as well as to fluid mechanics. C.Maclaurin (1698-1764), British mathematician. They apparently discovered the summation formula independently, see Goldstine [, p. 84]. Euler's publication came 1738.

$$\begin{aligned} \sum_{i=0}^{\infty} f(a+i) &= \int_a^\infty f(x)\,dx + \frac{f(a)}{2} - \sum_{j=1}^{r}\frac{B_{2j}f^{(2j-1)}(a)}{(2j)!} + R_{2r+2} \qquad (3.3.20)\\ &= \int_a^\infty f(x)\,dx + \frac{f(a)}{2} - \frac{f'(a)}{12} + \frac{f^{(3)}(a)}{720} - \dots \end{aligned}$$

in a form suitable for the convergence acceleration of series of essentially positive terms. We tabulate a few coefficients related to the Bernoulli and the Euler numbers.

There are some obscure points in this operator derivation, but we shall consider it as a heuristic calculation only and shall not try to legitimate the various steps of

Table 3.3.2. *Bernoulli and Euler numbers;* $B_1 = -1/2, E_1 = 1.$

$2j$	0	2	4	6	8	10	12
B_{2j}	1	$\frac{1}{6}$	$-\frac{1}{30}$	$\frac{1}{42}$	$-\frac{1}{30}$	$\frac{5}{66}$	$-\frac{691}{2730}$
$\frac{B_{2j}}{(2j)!}$	1	$\frac{1}{12}$	$-\frac{1}{720}$	$\frac{1}{30240}$	$-\frac{1}{1209600}$	$\frac{1}{47900160}$	
$\frac{B_{2j}}{2j(2j-1)}$	1	$\frac{1}{12}$	$-\frac{1}{360}$	$\frac{1}{1260}$	$-\frac{1}{1680}$	$\frac{1}{1188}$	$-\frac{691}{360360}$
E_{2j}	1	-1	5	-61	1385	-50521	2702765

it. With an appropriate interpretation, a more general version of this formula will be proved by other means in Theorem 3.3.3. A general remainder term is obtained there, if you let $b \to \infty$ in (3.3.26). You do not need it often, because the following much simpler error bound is usually applicable—but there are exceptions.

The Euler–Maclaurin expansion (on the right hand side) is typically semiconvergent only. Nevertheless a few terms of the expansion often gives startlingly high accuracy with simple calculations. *If, e.g., $f(x)$ is completely monotonic* [72] *, i.e., if $(-1)^j f^{(j)}(x) \geq 0$ for $x \geq a$, $j \geq 0$, the partial sums oscillate strictly around the true result; the first neglected term is then a strict error bound.* (This statement also follows from the Theorem below.)

Before we prove the theorem we shall exemplify how the summation formula is used in practice.

Example 3.3.13.

We return to the case of Example 3.3.12 with $f(x) = x^{-2}$, $a = 10$, and treat it with more precision and accuracy. We find $\int_a^\infty f(x)dx = a^{-1}$, $f'(a) = -2a^{-3}$, $f'''(a) = -24a^{-5}, \ldots$. By (3.3.20), $(r = 2)$,

$$\begin{aligned}\sum_{x=1}^{\infty} x^{-2} &= \sum_{x=1}^{9} x^{-2} + \sum_{i=0}^{\infty}(10+i)^{-2}\\ &= 1.53976\,7731 + 0.1 + 0.005 + 0.00016\,6667 - 0.00000\,0333 + R_6\\ &= 1.64493\,4065 + R_6.\end{aligned}$$

Since $f(x) = x^{-2}$ is completely monotonic, the first neglected term is a strict error bound; it is less than $720\,10^{-7}/30240 < 3 \cdot 10^{-9}$. (The actual error is approximately $2 \cdot 10^{-9}$.)

Although the Euler–Maclaurin expansion, in this example, seems to converge rapidly, it is in fact, only semi-convergent for any $a > 0$, and this is rather typical. We have namely $f^{(2r-1)}(a) = -(2r)!a^{-2r-1}$, and, by Example 3.1.6, $B_{2r}/(2r)! \approx (-1)^{r+1}2(2\pi)^{-2r}$. The ratio of two successive terms is thus $-(2r+2)(2r+1)/(2\pi a)^2$, hence the modulus of terms increase when $2r + 1 > 2\pi a$.

[72] See Sec. 3.3.5 for properties, criteria etc. for complete monotonicity, and for the extension to the case when f can be expressed as the difference of two completely monotonic functions.

The "rule" that one should terminate a semiconvergent expansion at the term of smallest magnitude, is in general no good for Euler–Maclaurin applications, since the high order derivatives (on the right hand side) are typically much more difficult to obtain than a few more terms in the expansion on the left hand side. Typically, you first choose r, $r \le 3$, depending on how tedious the differentiations are, and then you choose a in order to meet the accuracy requirements.

In this example we were lucky to have access to simple closed expressions for the derivatives and the integral of f. In other cases, one may use the possibilities for the numerical integration on an infinite interval mentioned in Chapter 5. See also Problem 28. In Problem 22(a) you find two formulas that result from the substitution of the formulas (3.2.44) that express higher derivatives in terms of central differences into the Euler–Maclaurin expansion.

An expansion of $f(x)$ into negative powers of x is often useful both for the integral and for the derivatives.

Example 3.3.14.
We consider $f(x) = (x^3 + 1)^{-1/2}$, for which the expansion

$$f(x) = x^{-3/2}(1 + x^{-3})^{-1/2} = x^{-1.5} - \frac{1}{2}x^{-4.5} + \frac{3}{8}x^{-7.5} - \dots$$

was derived and applied in Example 3.1.7. It was found that $\int_{10}^{\infty} f(x)dx = 0.632410375$, correctly rounded, and that $f'''(10) = -4.13\,10^{-4}$ with less than 1% error. The $f'''(10)$ term in the Euler–Maclaurin expansion is thus $-5.73\,10^{-7}$, with absolute error less than $6\,10^{-9}$. Inserting this into Euler–Maclaurin's summation formula, together with the numerical values of $\sum_{n=0}^{9} f(n)$ and $\frac{1}{2}f(10) - \frac{1}{12}f'(10)$, we obtain $\sum_{n=0}^{\infty} f(n) = 3.7941\,1570 \pm 10^{-8}$. The reader is advised to work out the details as an exercise.

Example 3.3.15. Let $f(x) = e^{-x^2}$, $a = 0$. Since all derivatives of odd order vanish at $a = 0$, then the expansion (3.3.20) may give the impression that $\sum_{j=0}^{\infty} e^{-j^2} = \int_0^{\infty} e^{-x^2}\,dx + 0.5 = 1.386\,2269$, but the sum (that is easily computed without any convergence acceleration) is actually 1.386 3186, hence the remainder R_{2r+2} cannot tend to zero as $r \to \infty$. The infinite Euler–Maclaurin expansion, where all terms but two are zero, *is convergent but is not valid.* Recall the distinction between the convergence and the validity of an infinite expansion, made in Sec. 3.1.2.

In this case $f(x)$ is not completely monotonic; for example $f''(x)$ changes sign at $x = 1$. With appropriate choice of r, the general error bound (3.3.26) will tell that the error is very small, but it cannot be used for proving that it is zero—because this is not true.

The mysteries of these examples have hopefully raised the appetite for a more substantial theory, including an error bound for the Euler–Maclaurin formula. We first need some tools that are interesting in their own right.

The **Bernoulli polynomial** $B_n(t)$ is an $n'th$ degree polynomial defined by the **symbolic** relation $B_n(t) = (B + t)^n$, where the exponents of B become subscripts after the expansion according to the binomial theorem. The Bernoulli numbers B_j were defined in Example 3.1.6. Their recurrence relation (3.1.11) can be written in the form

$$\sum_{j=0}^{n-1} \binom{n}{j} B_j = 0, \quad n \geq 2,$$

or "symbolically" $(B+1)^n = B^n = B_n$, (for the computation of B_{n-1}), $n \neq 1$, hence

$$B_n(1) = B_n(0) = B_n, \quad \text{for} \quad n \geq 2,$$
$$B_0(t) = 1, \qquad B_1(t) = t + B_1 = t - 1/2.$$

The **Bernoulli function** $\hat{B}_n(t)$ is a *piecewise polynomial* defined for $t \in \mathbf{R}$ by the equation $\hat{B}_n(t) = B_n(t - [t])$. (Note that $\hat{B}_n(t) = B_n(t)$ if $0 \leq t < 1$.)

Lemma 3.3.2.

(a) $\hat{B}'_{n+1}(t)/(n+1)! = \hat{B}_n(t)/n!$, $(n > 0)$,
$\hat{B}_n(0) = B_n$. *(For $n = 1$ this is the limit from the right.)*

$$\int_0^1 \frac{B_n(t)}{n!}\,dt = \begin{cases} 1, & \textit{if } n = 0; \\ 0, & \textit{otherwise.} \end{cases}$$

(b) *The piecewise polynomials $\hat{B}_p(t)$ are periodic; $\hat{B}_p(t+1) = \hat{B}_p(t)$. $\hat{B}_1(t)$ is continuous, except when t is an integer. For $n \geq 2$, $\hat{B}_n \in C^{n-2}(-\infty, \infty)$.*

(c) *The Bernoulli functions have the following (modified) Fourier expansions, $(r \geq 1)$,*

$$\frac{\hat{B}_{2r-1}(t)}{(2r-1)!} = (-1)^r 2 \sum_{n=1}^{\infty} \frac{\sin 2n\pi t}{(2n\pi)^{2r-1}}, \qquad \frac{\hat{B}_{2r}(t)}{(2r)!} = (-1)^{r-1} 2 \sum_{n=1}^{\infty} \frac{\cos 2n\pi t}{(2n\pi)^{2r}}.$$

Note that $\hat{B}_n(t)$ is an even (odd) function, when n is (even odd).

(d) $|\hat{B}_{2r}(t)| \leq |B_{2r}|$.

Proof. Statement (a) follows directly from the symbolic binomial expansion of the Bernoulli polynomials.

The demonstration of statement (b) is left for a problem. The reader is advised to draw the graphs of a few low order Bernoulli functions.

The Fourier expansion for $\hat{B}_1(t)$ follows from the Fourier coefficient formulas (3.1.30), (modified for the period 1 instead of 2π). The expansions for $\hat{B}_p(t)$, are then obtained by repeated integrations, term by term, with the use of (a).

Statement (d) then follows from the Fourier expansion, because $\hat{B}_{2r}(0) = B_{2r}$.
☐ COMMENTS: For $t = 0$ we obtain an interesting classical formula, together with

a useful asymptotic approximation that was obtained in a different way in Sec. 3.1.

$$\sum_{n=1}^{\infty} n^{-2r} = \frac{|B_{2r}|(2\pi)^{2r}}{2(2r)!}; \qquad \frac{|B_{2r}|}{(2r)!} \sim \frac{2}{(2\pi)^{2r}}. \tag{3.3.21}$$

Also note, how the rate of decrease of the Fourier coefficients is related to the type of singularity of the Bernoulli function at the integer points. (It does not help that the functions are smooth in the interval $[0,1]$.)

The Bernoulli polynomials have a generating function that is elegantly obtained by means of the following "symbolic" calculation.

$$\sum_0^\infty \frac{B_n(y)x^n}{n!} = \sum_0^\infty \frac{(B+y)^n x^n}{n!} = e^{(B+y)x} = e^{Bx}e^{yx} = \frac{xe^{yx}}{e^x-1}. \tag{3.3.22}$$

If the series is interpreted as a power series in the complex variable x, the convergence radius is 2π.

Theorem 3.3.3. The Euler–Maclaurin Formula.

Set $x_i = a + ih$, $x_n = b$, suppose that $f \in C^{2r+2}(a,b)$, and let $\hat{T}(a:h:b)f$ be the trapezoidal sum

$$\hat{T}(a:h:b)f = \sum_{i=1}^{n} \frac{h}{2}\big(f(x_{i-1}) + f(x_i)\big) = h\left(\sum_{i=0}^{n-1} f(x_i) + \tfrac{1}{2}(f(b) - f(a))\right). \tag{3.3.23}$$

Then

$$\begin{aligned}\hat{T}(a:h:b)f - \int_a^b f(x)\,dx &= \frac{h^2}{12}\big(f'(b) - f'(a)\big) - \frac{h^4}{720}\big(f'''(b) - f'''(a)\big) \\ &\quad + \ldots + \frac{B_{2r}h^{2r}}{(2r)!}\big(f^{(2r-1)}(b) - f^{(2r-1)}(a)\big) + R_{2r+2}(a,h,b)f.\end{aligned} \tag{3.3.24}$$

The remainder $R_{2r+2}(a,h,b)f$ is $O(h^{2r+2})$. It is represented by an integral with a kernel of constant sign in (3.3.25)*. An upper bound for the remainder is given in* (3.3.26)*.*

The estimation of the remainder is very simple in certain important particular cases: If $f^{(2r+2)}(x)$ does not change sign in the interval $[a,b]$ then $R_{2r+2}(a,h,b)f$ has the same sign as the first neglected[73] *term.*

If $f^{(2r+2)}(x)$ and $f^{(2r)}(x)$ have the same constant sign in $[a,b]$, then the value of the left hand side of (3.3.24) *lies between the values of the partial sum of the expansion displayed in* (3.3.24) *and the partial sum with one term less.* [74]*. In the limit, as $b \to \infty$, these statements still hold—also for the summation formula* (3.3.20)*—provided that the left hand side of* (3.3.24) *and the derivatives $f^{(\nu)}(b)$ $(\nu = 1:2r+1)$ tend to zero, if it is also assumed that*

$$\int_a^\infty |f^{(2r+2)}(x)|\,dx < \infty.$$

[73]If $r = 0$ all terms of the expansion are "neglected".

[74]Formally this makes sense for $r \geq 2$ only, but if we interpret $f^{(-1)}$ as " the empty symbol", it makes sense also for $r = 1$. If f is completely monotonic *the statement holds for every* $r \geq 1$. This is easy to apply, because simple criteria for complete monotonicity etc. are given in § 3.3.6.

Proof. To begin with we consider a single term of the trapezoidal sum, and set $x = x_{i-1} + ht$, $t \in [0,1]$, $f(x) = F(t)$. Suppose that $F \in C^p[0,1]$, where p is an *even* number.

We shall apply *repeated integration by parts*, Lemma 3.1.12, to the integral $\int_0^1 F(t)\,dt = \int_0^1 F(t)B_0(t)\,dt$. Use statement (a) of Lemma 3.3.1 in the equivalent form, $\int B_j(t)/j!\,dt = (B_{j+1}(t)/(j+1)!$

Consider the first line of the expansion in the next equation. Recall that $B_\nu = 0$ if ν is odd and $\nu > 1$. Since $B_{j+1}(1) = B_{j+1}(0) = B_{j+1}$, j will thus be odd in all non-zero terms, except for $j = 0$. Then, with no loss of generality, we assume that p is even.

$$\int_0^1 F(t)\,dt = \sum_{j=0}^{p-1}(-1)^j F^{(j)}(t)\frac{B_{j+1}(t)}{(j+1)!}\bigg|_{t=0}^{1} + (-1)^p\int_0^1 F^{(p)}(t)\frac{B_p(t)}{p!}\,dt$$

$$= \frac{F(1)+F(0)}{2} + \sum_{j=1}^{p-1}\frac{-B_{j+1}}{(j+1)!}(F^{(j)}(1) - F^{(j)}(0)) + \int_0^1 F^{(p)}(t)\frac{B_p(t)}{p!}\,dt$$

$$= \frac{F(1)+F(0)}{2} - \sum_{j=1}^{p-3}\frac{B_{j+1}}{(j+1)!}(F^{(j)}(1) - F^{(j)}(0)) - \int_0^1 F^{(p)}(t)\frac{B_p - B_p(t)}{p!}\,dt.$$

The upper limit of the sum is reduced to $p-3$, since the last term (with $j = p-1$) has been moved under the integral sign, and all values of j are odd. Set $j+1 = 2k$ and $p = 2r+2$. Then k is an integer that runs from 1 to r. Hence

$$\sum_{j=1}^{p-3}\frac{B_{j+1}}{(j+1)!}(F^{(j)}(1) - F^{(j)}(0)) = \sum_{k=1}^{r}\frac{B_{2k}}{(2k)!}(F^{(2k-1)}(1) - F^{(2k-1)}(0)).$$

Now set $F(t) = f(x_{i-1}+ht)$, $t \in [0,1]$. Then $F^{(2k-1)}(t) = h^{2k-1}f^{(2k-1)}(x_{i-1}+ht)$, and make abbreviations like $f_i = f(x_i)$, $f_i^{(j)} = f^{(j)}(x_i)$ etc..

$$\int_{x_{i-1}}^{x_i} f(x)\,dx = h\int_0^1 F(t)\,dt = \frac{h(f_{i-1}+f_i)}{2} - \sum_{k=1}^{r}\frac{B_{2k}h^{2k}}{(2k)!}(f_i^{(2k-1)} - f_{i-1}^{(2k-1)}) - R,$$

where R is the local remainder that is now an integral over $[x_{i-1}, x_i]$. Adding these equations, for $i = 1:n$, yields a result equivalent to (3.3.24), namely

$$\int_a^b f(x)\,dx = \hat{T}(a:h:b)f - \sum_{k=1}^{r}\frac{B_{2k}h^{2k}}{(2k)!}f^{(2k-1)}(x)\bigg|_{x=a}^{b} - R_{2r+2}(a,h,b)f,$$

$$R_{2r+2}(a,h,b)f = h^{2r+2}\int_a^b\Big(B_{2r+2} - \hat{B}_{2r+2}((x-a)/h)\Big)\frac{f^{(2r+2)}(x)}{(2r+2)!}\,dx. \quad (3.3.25)$$

By Lemma 3.3.1, $|\hat{B}_{2r+2}(t)| \le |B_{2r+2}|$, hence the kernel $B_{2r+2} - \hat{B}_{2r+2}((x-a)/h)$ has the same sign as B_{2r+2}. Suppose that $f^{(2r+2)}(x)$ does not change sign on (a,b). Then

$$\operatorname{sign} f^{(2r+2)}(x) = \operatorname{sign}\big(f^{(2r+1)}(b) - f^{(2r+1)}(a)\big),$$

hence $R_{2r+2}(a,h,b)f$ has the same sign as the first neglected term.
The second statement about "simple estimation of the remainder" then follows from Theorem 3.1.3, since the Bernoulli numbers (with even subscripts) have alternating signs.

If sign $f^{(2r+2)}(x)$ is not constant, then we note instead that

$$|B_{2r+2}-\hat{B}_{2r+2}((x-a)/h)| \le |2B_{2r+2}|,$$

and hence

$$\begin{aligned}|R_{2r+2}(a,h,b)f| &\le h^{2r+2}\frac{|2B_{2r+2}|}{(2r+2)!}\int_a^b |f^{(2r+2)}(x)|dx\\ &\approx 2\Big(\frac{h}{2\pi}\Big)^{2r+2}\int_a^b |f^{(2r+2)}(x)|dx. \qquad (3.3.26)\end{aligned}$$

If $\int_a^\infty |f^{(2r+2)}(x)|dx < \infty$ this holds also in the limit as $b \to \infty$.
□

Note that there are (at least) three parameters here that can be involved in *different* natural limit processes: For example, one of the parameters can tend to its limit, while the two others are kept fixed. The remainder formula (3.3.26) contains all you need for settling various questions about convergence.

- $b \to \infty$; natural when Euler–Maclaurin's formula is used as a summation formula, or for deriving an approximation formula valid when b is large.
- $h \to 0$; natural when Euler–Maclaurin's formula is used in connection with numerical integration. You see how the values of derivatives of f at the endpoints a, b can highly improve the estimate of the integral of f, obtained by the trapezoidal rule with constant step size. Euler–Maclaurin's formula is also useful for the design and analysis of other methods for numerical integration, see Romberg's method in the next subsection.
- $r \to \infty$; $\lim_{r\to\infty} R_{2r+2}(a,h,b)f = 0$ can be satisfied only if $f(z)$ is an entire function, such that $|f^{n)}(a)| = o((2\pi/h)^n)$ as $n \to \infty$. Fortunately, this type of convergence is rarely needed in practice. With appropriate choice of b and h, the expansion is typically rapidly semiconvergent. Since the derivatives of are typically more expensive to compute than the values of f, one frequently reduces h (in integration) or increases b (in summation or integration over an infinite interval), and truncates the expansion several terms before one has reached the smallest term that is otherwise the standard procedure with alternating semiconvergent expansion, cf Sec. 3.1.8.

Variations of the Euler–Maclaurin summation formula, with *finite differences instead of derivatives* in the expansion, are given in Problem 22, where you also find *a more general form of the formula*, and two more variations of it.

Euler–Maclaurin's formula can also be used for finding an algebraic expression for a finite sum, see Problem 35 or, as in the following example, for finding an expansion that determines the asymptotic behavior of a sequence or a function.

Example 3.3.16. *An expansion that generalizes Stirling's formula.*

We shall use Euler–Maclaurin formula for $f(x) = \ln x$, $a = m > 0$, $h = 1$, $b = n \geq m$. We obtain

$$\hat{T}(m:1:n)f = \sum_{i=m+1}^{n} \ln i - \tfrac{1}{2}\ln n + \tfrac{1}{2}\ln m = \ln(n!) - \tfrac{1}{2}\ln n - \ln(m!) + \tfrac{1}{2}\ln m.$$

$$f^{(2k-1)}(x) = (2k-2)!x^{1-2k}, \qquad \int_m^n f(x)\,dx = n\ln n - n - m\ln m + m.$$

Note that $\hat{T}(m:1:n)f$ and $\int_m^n f(x)\,dx$ are unbounded as $n \to \infty$, but their difference is bounded. Putting these expressions into (3.3.24), and separating the terms containing n from the terms containing m gives

$$\ln(n!) - (n + \tfrac{1}{2})\ln n + n - \sum_{k=1}^{r} \frac{B_{2k}}{2k(2k-1)n^{2k-1}} \tag{3.3.27}$$

$$= \ln(m!) - (m + \tfrac{1}{2})\ln m + m - \sum_{k=1}^{r} \frac{B_{2k}}{2k(2k-1)m^{2k-1}} - R_{2r+2}(m:1:n).$$

By (3.3.26), after a translation of the variable of integration,

$$|R_{2r+2}(m:1:n)| \leq \int_m^n \frac{|2B_{2r+2}|\,dx}{(2r+2)x^{2r+2}} \leq \frac{|2B_{2r+2}|}{(2r+2)(2r+1)|m^{2r+1}|} \approx \frac{(2r)!}{\pi|2\pi m|^{2r+1}} \tag{3.3.28}$$

Now let $n \to \infty$ with fixed r, m. First, note that the integral in the error bound converges. Next, in most texts of calculus Stirling's formula is derived in the following form:

$$n! \sim \sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n}, (n \to \infty) \tag{3.3.29}$$

If you take the natural logarithm of this, it follows that the left hand side of (3.3.27) tends to $\frac{1}{2}\ln(2\pi)$ [75] , and hence

$$\ln(m!) = (m + \tfrac{1}{2})\ln m - m + \tfrac{1}{2}\ln(2\pi) + \sum_{k=1}^{r} \frac{B_{2k}}{2k(2k-1)m^{2k-1}} + R, \tag{3.3.30}$$

where a bound for R is given by (3.3.28). The numerical values of the coefficients are found in Table 3.3.4.

Almost the same derivation works also for $f(x) = \ln(x+z)$, $m = 0$, where z is a complex number, not on the negative real axis. A few basic facts about the Gamma function are needed, see details in Henrici [21, Vol. 2, Sec. 11.11, Example 3].

[75] You may ask why we refer to (3.3.29). Why not? Well, it is not necessary, because it is easy to prove that the left hand side of (3.3.27) increases with n and is bounded; it thus tends to some limit C (say). The proof that $C = \ln\sqrt{2\pi}$ *exactly* is harder, without the Wallis product idea (from 1655) that is probably used in your calculus text, or something equally ingenious or exotic. However, if you compute the right hand side of (3.3.27) for $m = 17$, $r = 5$ (say), and estimate the remainder, you will obtain C to a fabulous guaranteed accuracy, in invisible computer time after a rather short programming time. And you may then replace $\frac{1}{2}\ln 2\pi$ by your own C in (3.3.30), if you like.

The result is that *you just replace the integer m by the complex number z in the expansion (3.3.30).* According to [1, Sec. 6.1.42] R is to be multiplied by $K(z) = \text{upper bound}_{u \geq 0}|z^2/(u^2 + z^2)|$. For z real and positive, $K(z) = 1$, and since $f'(x) = (z + x)^{-1}$ is completely monotonic, it follows from Theorem 3.3.3 that, *in this case, R is less in absolute value than the first term neglected and has the same sign.*

It is customary to *write* $\ln \Gamma(z + 1)$ *instead of* $\ln(z!)$. The Gamma function is one of the most important transcendental functions. See, e.g., Handbook [1, Chapter 6] and Lebedev[24].

This formula (with $m = z$) is useful for the practical computation of $\ln \Gamma(z+1)$. Its semiconvergence is best if $\Re z$ is large and positive. If this condition is not satisfied, the situation can easily be improved by means of logarithmic forms of the

- *reflection formula*: $\Gamma(z)\Gamma(1 - z) = \pi/\sin \pi z$,
- *recurrence formula*: $\Gamma(z + 1) = z\Gamma(z)$.

By simple applications of these formulas the computation of $\ln \Gamma(z + 1)$ for an arbitrary $z \in \mathbf{C}$ is reduced to the computation of the function for a number z', such that $|z'| \geq 17$, $\Re z' > \frac{1}{2}$, for which the total error, if $r = 5$, becomes typically less than 10^{-14}. See Problem 25.

Although, in this section, the main emphasis is on the application of the Euler–Maclaurin formula to the computation of sums and limits, we shall comment a little on its possibilities for other applications [76] .

- It shows that the *global truncation error of the trapezoidal rule* for $\int_a^b f(x)\,dx$ with step size h, *has an expansion into powers of h^2*. Note that although the expansion contains derivatives at the boundary points only, the remainder requires the integrability of $|f^{(2r+2)}|$ in the interval $[a, b]$. The Euler–Maclaurin formula is thus the theoretical basis for the application of *repeated Richardson extrapolation* to the results of the trapezoidal rule, known as *Romberg's method*, see Sec. 3.3.5. Note that *the validity depends on the differentiability properties of f.*
- The Euler–Maclaurin formula can be used for highly accurate numerical integration when the values of some derivatives of f are known at $x = a$ and $x = b$. More about this in Chapter 5.
- Theorem 3.3.3 shows that the trapezoidal rule is second order accurate, unless $f'(a) = f'(b)$, but there exist *interesting exceptions.* Suppose that the function f is infinitely differentiable for $x \in \mathbf{R}$, and that f *has* $[a, b]$ *as an interval of periodicity*, i.e., $f(x + b - a) = f(x), \forall x \in \mathbf{R}$. Then $f^{(k)}(b) = f^{(k)}(a)$, for $k = 0, 1, 2, \ldots$, hence *every term in the Euler–Maclaurin expansion is zero* for

[76] As you may have noted, we write "the Euler–Maclaurin formula" mainly for Eq. (3.3.24) that is used in general theoretical discussions, or if other applications than the summation of an infinite series are the primary issue. The term "the Euler–Maclaurin summation formula" is mainly used in connection with Eq. (3.3.20), i.e. when the summation of an infinite series is the issue. "The Euler–Maclaurin expansion" denotes both the right hand side of (3.3.24), except for the remainder, and for the corresponding terms of (3.3.20). These distinctions are convenient for us, but they are neither important nor in general use.

the integral over *the whole period* $[a, b]$. One could be led to believe that the trapezoidal rule gives the exact value of the integral, but this is usually not the case; for most periodic functions f, $\lim_{r\to\infty} R_{2r+2}f \neq 0$; *the expansion converges, of course, though not necessarily to the correct result.*

- On the other hand, the convergence as $h \to 0$ for a fixed (though arbitrary) r is a different story; the error bound (3.3.26) shows that $|R_{2r+2}(a, h, b)f| = O(h^{2r+2})$. Since r is arbitrary, this means that *for this class of functions, the trapezoidal error tends to 0 faster than any power of h, as $h \to 0$* . We may call this **superconvergence**. In such cases Romberg's method yields no improvement; actually the excellent results become worse.
  Suppose that this periodic function $f(z)$, $z = x + iy$, is analytic in a strip, $|y| < c$, around the real axis. It can then be shown that the error of the trapezoidal rule is $O(e^{-\eta/h})$ as $h \downarrow 0$; η is related to the width of the strip. A similar result was obtained in Example 3.1.6, for an annulus instead of a strip.
  As a rule, this discussion does *not* apply to periodic functions which are defined by periodic continuation of a function originally defined on $[a, b]$ (like the Bernoulli functions). They usually become non-analytic at a and b, and at all points $a + (b - a)n$, $n = 0, \pm 1, \pm 2, \ldots$.
- The application of the trapezoidal rule to an *integral over* $[0, \infty)$ of a function $f \in C^\infty(0, \infty)$ often yields similar features, sometimes even more striking. Suppose that, for $k = 1, 2, 3, \ldots$, $f^{(2k-1)}(0) = 0$ and $f^{(2k-1)}(x) \to 0$, as $x \to \infty$, and $\int_0^\infty |f^{(2k)}(x)|\,dx < \infty$. [77]
  Then all terms of the Euler–Maclaurin expansion are zero, and one can be misled to believe that the trapezoidal sum gives $\int_0^\infty f(x)\,dx$ exactly for any step size h! We have already seen an example of this in Example 3.3.15. See also Theorem 3.3.4 and Problem 34.
  The explanation is that the remainder $R_{2r+2}(a, h, \infty)$ will typically not tend to zero, as $r \to \infty$ for fixed h. On the other hand: if we consider the behavior of the truncation error as $h \to 0$ for given r, we find that it is $o(h^{2r})$ for any r, just like the case of a periodic function.
  For a finite subinterval of $[0, \infty)$, however, the remainder is still typically $O(h^2)$, and for each step the remainder is typically $O(h^3)$. So, there is an *enormous cancellation of the local truncation errors*, when a C^∞-function, with vanishing odd-order derivatives at the origin, is integrated by the trapezoidal rule over $[0, \infty)$.

The **Poisson summation formula** is, however, even better than the Euler–Maclaurin formula for the quantitative study of the trapezoidal truncation error on an infinite interval. For convenient reference we now formulate the following surprising result, although the proof, by means of the Poisson formula, comes in Ch.12.

Theorem 3.3.4. *Suppose that the trapezoidal rule (or, equivalently, the rectangle rule) is applied with* constant *step size h to $\int_{-\infty}^{\infty} f(t)\,dt$. The Fourier transform of*

[77]Note that for any function $g \in C^\infty(-\infty, \infty)$ the function $f(x) = g(x) + g(-x)$ satisfies such conditions at the origin.

f reads $\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,dt$.

Then the integration error decreases like $2\hat{f}(2\pi/h)$ as $h \downarrow 0$.

Example 3.3.17.

For the normal probability density, we have

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(t/\sigma)^2}, \qquad \hat{f}(\omega) = e^{-\frac{1}{2}(\omega\sigma)^2}.$$

The integration error is thus approximately $2\exp(-2(\pi\sigma/h)^2)$. Roughly speaking, the number of correct digits is doubled if h is divided by $\sqrt{2}$, e.g., the error is approximately $5.4\,10^{-9}$ for $h = \sigma$, and $1.4\,10^{-17}$ for $h = \sigma/\sqrt{2}$.

We shall illuminate these amazing properties of the trapezoidal rule from different points of view in several places in this book. First, See examples in Problem 34, and applications to the so-called bell sums in Problem 36.

3.3.5 Repeated Richardson Extrapolation and Related Algorithms.

Let $F(h)$ denote the value of a certain quantity obtained with step length h. In many calculations one wants to know the limiting value of $F(h)$ as the step length approaches zero. However, the work to compute $F(h)$ often increases sharply as $h \to 0$. In addition, the effects of round-off errors often set a practical bound for how small h can be chosen.

Often, one has some knowledge of how the truncation error $F(h) - F(0)$ behaves when $h \to 0$. If

$$F(h) = a_0 + a_1 h^p + O(h^r), \qquad h \to 0, r > p,$$

where $a_0 = F(0)$ is the quantity we are trying to compute and a_1 is unknown, then a_0 and a_1 can be estimated if we compute F for two step lengths, h and qh, $q > 1$:

$$F(h) = a_0 + a_1 h^p + O(h^r),$$
$$F(qh) = a_0 + a_1 (qh)^p + O(h^r),$$

from which eliminating a_1 we get

$$F(0) = a_0 = F(h) + \frac{F(h) - F(qh)}{q^p - 1} + O(h^r). \qquad (3.3.31)$$

This formula is called **Richardson extrapolation**, or the *deferred approach to the limit*. Examples of this were mentioned in Chap. 1—the application of the above process to the trapezoidal rule for numerical integration (where $p = 2$, $q = 2$), and for differential equations—$p = 1$, $q = 2$ for Euler's method, $p = 2$, $q = 2$ for Runge's 2nd order method.

The term $\frac{F(h)-F(qh)}{q^p-1}$ is called the *Richardson correction.* It is used in in (3.3.31) for improving the result. Sometimes, however, it is used only for estimating the error [78] $F(h) - a_0$. If the irregular errors are negligible, this error estimate is asymptotically correct. More often, the Richardson correction is used as error estimate for the improved (or extrapolated) value $F(h) + \frac{F(h)-F(qh)}{q^p-1}$, but this is typically a strong overestimate; the error estimate is $O(h^p)$, while the error is $O(h^r)$, $(r > p)$.

Suppose that a more complete expansion of $F(h)$ in powers of h, is known to exist,

$$F(h) = a_0 + a_1 h^{p_1} + a_2 h^{p_2} + a_3 h^{p_3} + \dots \qquad 0 < p_1 < p_2 < p_3 < \dots, \tag{3.3.32}$$

where the exponents are typically known, while the coefficients are unknown. Then one can *repeat the use of Richardson extrapolation* in a way described below. This process is, in many numerical problems—especially in the numerical treatment of integral and differential equations—one of the simplest ways to get results which have tolerable truncation errors. The application of this process becomes especially simple when the step length form a geometric series $H, H/q, H/q^2, \dots$, where $q > 1$ and H is the **basic step length**, or shorter the *bigstep.*

Theorem 3.3.5. *Suppose that an expansion of the form of (3.3.32), where* $0 < p_1 < p_2 < p_3 < \dots$, *holds for* $F(h)$, *and set*

$$F_1(h) = F(h), \qquad F_{k+1}(h) = \frac{q^{p_k}F_k(h) - F_k(qh)}{q^{p_k}-1} = F_k(h) + \frac{F_k(h) - F_k(qh)}{q^{p_k}-1}, \tag{3.3.33}$$

for $k = 1 : (n-1)$, *where* $q > 1$. *Then* $F_n(h)$ *has an expansion of the form*

$$F_n(h) = a_0 + a_n^{(n)} h^{p_n} + a_{n+1}^{(n)} h^{p_{n+1}} + \dots; \qquad a_\nu^{(n)} = \prod_{k=1}^{n-1} \frac{q^{p_k} - q^{p_\nu}}{q^{p_k}-1} a_\nu. \tag{3.3.34}$$

Note that $a_\nu^{(n)} = 0$ *for* $\nu < n$.

Proof. Set temporarily $F_k(h) = a_0 + a_1^{(k)} h^{p_1} + a_2^{(k)} h^{p_2} + \dots + a_\nu^{(k)} h^{p_\nu} + \dots$. Put this expansion into the first expression on the right hand side of (3.3.33), and, substituting $k+1$ for k, put it into the left hand side. By matching the coefficients for h^{p_ν} we obtain $a_\nu^{(k+1)} = a_\nu^{(k)}(q^{p_k} - q^{p_\nu})/(q^{(p_k)} - 1)$. By (3.3.32), the expansion holds for $k = 1$, with $a_\nu^{(1)} = a_\nu$. The recursion formula then yields the product formula for $a_\nu^{(n)}$. Note that $a_\nu^{(\nu+1)} = 0$, hence $a_\nu^{(n)} = 0, \forall \nu < n$. ☐

The product formula is for theoretical purpose. The recurrence formula is for practical use. If an expansion of the form of Eq. (3.3.32) is known to exist, the above theorem gives a way to compute increasingly better estimates of a_0. The leading

[78] This can make sense, e.g., if the values of F are afflicted by other errors, usually irregular, suspected to be comparable in size to the correction.

term of $F_n(h) - a_0$ is $a_n^{(n)} h^{p_n}$, the exponent of h increases with n. A moment's reflection on equation (3.3.33) will convince the reader that (using the notation of the theorem) $F_{k+1}(h)$ is determined by the $k+1$ values

$$F_1(H), F_1(H/q), \ldots, F_1(H/q^k).$$

With some changes in notation we obtain the following algorithm.

Algorithm 3.3.2 Repeated Richardson extrapolation

For $m = 1 : N$, set $T_{m,1} = F(H/q^{m-1})$, and compute, for $m = 2 : N$, $k = 1 : m-1$,

$$T_{m,k+1} = T_{m,k} + \frac{T_{m,k} - T_{m-1,k}}{q^{p_k} - 1}. \tag{3.3.35}$$

It is sometimes advantageous to write

$$T_{m,k+1} = \frac{q^{p_k} T_{m,k} - T_{m-1,k}}{q^{p_k} - 1}.$$

The computations can be set up in a scheme, where an extrapolated value in the scheme is obtained by using the quantity to its left and the correction diagonally above.[79]

Table 3.3.3. *Scheme for repeated Richardson extrapolation*

	$\frac{\Delta}{q^{p_1}-1}$	$\frac{\Delta}{q^{p_2}-1}$	$\frac{\Delta}{q^{p_3}-1}$
T_{11}			
T_{21}	T_{22}		
T_{31}	T_{32}	T_{33}	
T_{41}	T_{42}	T_{43}	T_{44}

Suppose that TOL is the permissible error. Then, according to the argument above, one continues the process, until two values *in the same row* agree to the desired accuracy, i.e., $|T_{m,k} - T_{m,k-1}| < Tol - CU$, where CU is an upper bound of the irregular error, (see below). (TOL should, of course, be larger than CU.) If no other error estimate is available, $\min_k |T_{m,k} - T_{m,k-1}| + CU$ is usually chosen as error estimate, even though it is typically a strong overestimate.

Typically $k = m$, and T_{mm} is accepted as the numerical result, but this is not always the case. For instance, if H has been chosen so large that the use of the basic

[79]This is for small hand-computed examples and for the explanation of the method. In a computer the results are simply stored in a lower triangular matrix.

asymptotic expansion is doubtful, then the uppermost diagonal of the extrapolation scheme contains nonsense and should be ignored, except for its element in the first column. Such a case is detected by inspection of the difference quotients in a column. If for some k, where $T_{k+2,k}$ has been computed and the modulus of the relative irregular error of $T_{k+2,k} - T_{k+1,k}$ is less than (say) 20%, *and*, most important, the difference quotient $(T_{k+1,k} - T_{k,k})/(T_{k+2,k} - T_{k+1,k})$ is is very different from its theoretical value q^{p_k}, then the uppermost diagonal is to be ignored (except for its first element). In such a case, one says that H *is outside the asymptotic regime.* [80]

In this discussion a bound for the inherited irregular error is needed. We shall now derive such a bound. Fortunately, it turns out that the numerical stability of the Richardson scheme is typically very satisfactory, (although the total error bound for T_{mk} will never be smaller than the largest irregular error in the first column).

Denote by ϵ_1 the the column vector with the irregular errors of the initial data. We neglect the rounding errors committed during the computations. [81] Then the inherited errors satisfy the same linear recursion formula as the $T_{m,k}$, i.e.

$$\epsilon_{m,k+1} = \frac{q^{p_k}\epsilon_{m,k} - \epsilon_{m-1,k}}{q^{p_k} - 1}.$$

Denote the k'th column of errors by ϵ_k, and set $\|\epsilon_k\| = \max_m |\epsilon_{m,k}|$, $\|\epsilon_1\| = U$. Then $\|\epsilon_{k+1}\| \le \frac{q^{p_k}+1}{q^{p_k}-1}\|\epsilon_k\|$. Hence, for every k, $\|\epsilon_{k+1}\| \le C\|\epsilon_1\| = CU$, where C is the infinite product

$$C = \prod_{k=1}^{\infty} \frac{q^{p_k}+1}{q^{p_k}-1} = \prod_{k=1}^{\infty} \frac{1+q^{-p_k}}{1-q^{-p_k}}$$

that converges as fast as $\sum q^{-p_k}$; the multiplication of ten factors are thus more than enough for obtaining a sufficiently accurate value of C.

Example 3.3.18. *Values of the infinite product C for a few common cases with $q = 2$.*

The most common special case is an expansion where $p_k = 2k$,

$$F(h) = a_0 + a_1h^2 + a_2h^4 + a_3h^6 + \dots \tag{3.3.36}$$

All the three Examples 3.3.19–3.3.21, are of this type. The headings of the columns of Table 3.3.5 then become $\Delta/3, \Delta/15, \Delta/63, \dots$. In this case *we find that* $C = \frac{5}{3}\cdot\frac{7}{15}\cdots < 2$ (after less than 10 factors).

For (systems of) ordinary differential equations there exist some general theorems, according to which the form of the asymptotic expansion (3.3.32) of the global error can be found. References to the literature are given in Example 3.3.24.

[80] Sometimes several of the uppermost diagonals are to be ignored. It was mentioned at the end of § 3.3.4 that the trapezoidal rule is superconvergent in some common and important cases. Then all the difference quotients in the first column are much larger than $q^{p_1} = q^2$. So, according to the rule just formulated, every element of the Romberg scheme, outside the first column should be ignored. It is all right; *in superconvergent cases Romberg's method is of no use*; it deteriorates the excellent results that the trapezoidal rule has produced.

[81] They are usually for various reasons of less importance. One can also *make* them smaller, in floating-point computation, by subtracting a suitable constant from all initial data. This is applicable to all linear methods of convergence acceleration.

For *Numerov's method* for ordinary differential equations, discussed in Example 3.2.17 and Problem 31, one can show that we have the same exponents in the expansion for the global error, but $a_1 = 0$. (and the first heading disappears). We thus have the same product as above, except that the first factor disappears, *hence* $C < 2 \cdot \frac{3}{5} = 1.2$.

For *Euler's method* for ordinary differential equations, presented in Sec. 1.4.2., $p_k = k$; the headings are $\Delta/1, \Delta/3, \Delta/7, \Delta/15, \ldots$. *Hence* $C = 3 \cdot \frac{5}{3} \cdot \frac{9}{7} \cdots = 8.25$.

For *Runge's 2nd order method*, presented in Sec. 1.4.3, the exponents are the same, but $a_1 = 0$ (and the first heading disappears). We thus have the same product as for Euler's method, except that the first factor disappears, *hence* $C = 8.25/3 = 2.75$.

If $p_j = j \cdot p$, $j = 1, 2, 3, \ldots$ in (3.3.32), i.e. for expansions of the form

$$F(h) = a_0 + a_1 h^p + a_2 h^{2p} + a_3 h^{3p} + \ldots,$$

it is not necessary that the step sizes form a geometric progression. Choose any increasing sequence of integers $q_1 = 1, q_2, \ldots, q_k$ and set $h_i = H/q_i$.

We can then use an algorithm that looks very similar to repeated Richardson extrapolation. *In cases where both are applicable, i.e. if* $p_k = p \cdot k$, $q_i = q^i$, *they are identical, otherwise they have different areas of application.* See Example ex33.ode. Note that the expansion is a usual power series in the variable $x = h^p$, which can be approximated by a polynomial in x. Suppose that $k+1$ values $F(H), F(H/q_2), \ldots, F(H/q_k)$ are known. Then by the corollary to Theorem 3.2.1, a polynomial $Q \in \mathcal{P}_k$ is uniquely determined by the interpolation conditions

$$Q(x_i) = F(H/q_i), \qquad x_i = (H/q_i)^p, \quad i = 1 : k.$$

Our problem is to find $Q(0)$. Many interpolation formulas can be used for this extrapolation. *Neville's algorithm*, which is derived in Sec. 4.2, is particularly convenient in this situation. Eq.(4.2.27) yields, after a change of notation, the following recursion.

Algorithm 3.3.3 Neville's algorithm
For $m = 1 : N$, set $T_{m,1} = F(H/q_m)$, where $1 = q_1 < q_2 < q_3 \ldots$, is any increasing sequence of integers, and compute, for $m = 2 : N$, $k = 1 : m-1$,

$$T_{m,k+1} = T_{m,k} + \frac{T_{m,k} - T_{m-1,k}}{(q_m/q_{m-k})^p - 1} = \frac{(q_m/q_{m-k})^p T_{m,k} - T_{m-1,k}}{(q_m/q_{m-k})^p - 1}. \qquad (3.3.37)$$

The computations can be set up in a triangle matrix as for repeated Richardson extrapolations, without headings.

Example 3.3.19. *Computation of* π *by means of regular polygons.*

The ancient Greeks computed approximate values of the circumference of the unit circle, 2π, by inscribing a regular polygon and computing its perimeter.

Archimedes considered the inscribed 96-sided regular polygon, whose perimeter is 6.2821. In general, a regular n-sided polygon inscribed in a circle with radius 1 has circumference $c_n = 2n\sin\frac{\pi}{n}$. If we put $h = 1/n$, then

$$c(h) = c_{1/h} = \frac{2}{h}\sin\pi h = 2\pi - \frac{\pi^3}{3}h^2 + \frac{\pi^5}{60}h^4 - \dots,$$

so $c(h)$ satisfies the assumptions for repeated Richardson extrapolation with $p_k = 2k$. In order to use this with $q = 2$ we first try to find a recursion formula that leads from c_n to c_{2n}.

$$c_{2n} = 4n\sin\frac{\pi}{2n} = 4n\sqrt{\frac{1}{2}\Big(1-\cos\frac{\pi}{n}\Big)} = 2n\sqrt{2-\sqrt{4-(c_n/n)^2}}$$
$$= 2c_n\Big/\sqrt{2+\sqrt{4-(c_n/n))^2}}.$$

(Derive this! The last transformation is made to avoid cancellation and consequential round-off errors.) The following table gives the *Richardson scheme* using values c_n, $n = 6, 12, \dots, 96$ computed to nine decimals using this recursion (see also Problem 29).

m	n	$k=1$	$k=2$	$k=3$	$k=4$
1	6	6.000000000			
2	12	6.211657083	6.282209444		
3	24	6.265257227	6.283123942	6.283184908	
4	48	6.278700406	6.283181466	6.283185301	6.283185307
5	96	6.282063902	6.283185068	6.283185308	6.283185308

The rounding errors set the limit for the accuracy of $T_{m,n}$ when $m + k > 8$. A correctly rounded value of 2π reads 6.2831853071. Note that, although π was mentioned in the analysis above, it was never used in the computation of the initial data for the algorithm. [82]

We shall also make a short calculation with *Neville's algorithm*, based on three simple polygons, with $n = 2, 3$ and 6 sides, not in geometric progression. We can interpret a 2-sided polygon as a diameter described up and down. Its "circumference" is thus equal to 4.

2	4		
3	$3\sqrt{3}$	6.15307	
6	6	6.26795	6.2823

[82]An extension of this example was used as a test problem for a package for (in principle) arbitrarily high precision floating point arithmetic in Matlab for a PC. For instance, π was obtained to 203 decimal places with 22 polygons and 21 Richardson extrapolations in less than half a minute. The extrapolations took a small fraction of this time. Nevertheless they increased the number of correct decimals from approximately 15 to 203.

This is even a little better than the result (6.2821) obtained for the 96-sided polygon without extrapolations.

Example 3.3.20. *Application to numerical differentiation.*

By Bickley's table for difference operators, i.e., Table 3.2.1 in Sec. 3.2.2, we know that

$$\frac{\delta}{h} = \frac{2\sinh(hD/2)}{h} = D + a_2h^2D^3 + a_4h^4D^5 + \dots,$$
$$\mu = \cosh(hD/2) = 1 + b_2h^2D^2 + b_4h^4D^4 + \dots,$$

where the values of the coefficients are now unimportant to us. Hence

$$f'(x) - \frac{f(x+h) - f(x-h)}{2h} = Df(x) - \frac{\mu\delta f(x)}{h} \quad \text{and} \quad f''(x) - \frac{\delta^2 f(x)}{h^2}$$

have expansions into *even* powers of h. Repeated Richardson extrapolation can thus be used with step sizes H, $H/2$, $H/4, \dots$ and headings $\Delta/3$, $\Delta/15$, $\Delta/63, \dots$. For numerical examples, see Problems of this section.

Richardson extrapolation can be applied in the same way to the computation of higher derivatives. Because of the division by h^k in the difference approximation of $f^{(k)}$, *irregular errors in the values of $f(x)$ are of much greater importance in numerical differentiation than in interpolation and integration.* It is therefore important to use high order approximations in numerical differentiation, so that larger values of h can be used.

Suppose that the irregular errors of the values of f are bounded in magnitude by ERB, these errors are propagated to $\mu\delta f(x)$, $\delta^2 f(x), \dots$ with bounds equal to ERB$/h$, 4ERB$/h^2, \dots$. As mentioned earlier, the Richardson scheme (in the version used here) is no rascal; it multiplies the latter bounds by a factor less than 2.

Example 3.3.21. *Application to numerical integration. Romberg's method.* Set $x_i = a + ih$, also for non-integer subscripts, $x_n = b$. Denote by $\hat{T}(a:h:b)f$, $\hat{R}(a,h,b)f$ the trapezoidal and the midpoint sum, respectively.

$$\hat{T}(a:h:b)f = \sum_{i=1}^{n} \frac{h}{2}\big(f(x_{i-1}) + f(x_i)\big); \qquad \hat{R}(a,h,b)f = \sum_{i=1}^{n} hf(x_{i-1/2}).$$

We know by Euler–Maclaurin's formula that

$$\hat{T}(a:h:b) - \int_a^b f(x)\,dx = c_2h^2 + c_4h^4 + c_6h^6 + \dots,$$

where $c_k = 0$ if $f \in \mathcal{P}_k$. Repeated Richardson extrapolation can thus be used with $q = 2$, $p_k = 2k$, i.e., with step sizes H, $H/2$, $H/4, \dots$ and headings $\Delta/3$, $\Delta/15$, $\Delta/63, \dots$. H is called the *bigstep*. In this section (except for Problem 28) we consider just one bigstep, i.e., $b = a + H$. The important question about the control of step size and order over a long interval is postponed to Ch. 5.

This application of repeated Richardson extrapolation is known as **Romberg's method**, after a publication of Romberg in 1955. A thorough analysis of the method was carried out by Bauer, Rutishauser and Stiefel [2, 1963] that we shall refer to for proof details.

For practical numerical calculations the values of the coefficients c_k are not needed, but they are used, e.g., in the derivation of an error bound, see Theorem 3.3.6. It is also important to remember that the coefficients depend on derivatives of increasing order; the success of repeated Richardson extrapolations is thus related to the behavior in $[a, a+H]$ of the higher derivatives of the integrand.

One easily shows the following simple and useful relation of the trapezoidal rule to the midpoint rule (Problem 37),

$$\hat{T}(h/2) = \frac{1}{2}(\hat{T}(h) + \hat{R}(h)) \tag{3.3.38}$$

that makes it possible to reuse the function values that have been computed earlier.

Example 3.3.22. *A numerical illustration to Romberg's method.* The integral

$$\int_0^{0.8} \frac{\sin x}{x}\, dx$$

is to be computed to at least eight correct decimals.

As an exercise the reader is advised to check some of the midpoint and trapezoidal sums given below, which are correct to ten decimals. Use (3.3.38).

h	$\hat{R}(h)f$	$\hat{T}(h)f$
0.8	0.77883 66846	0.75867 80454
0.4	0.77376 69772	0.76875 73650
0.2	0.77251 27162	0.77126 21711
0.1		0.77188 74437

The correct value, to ten decimals, is 0.7720957855. The trapezoidal sums are now copied to the first column of the Richardson (Romberg) scheme, i.e., $k = 0$ for improving the value of the integral.

m	T_{m1}	$\Delta/3$	T_{m2}	$\Delta/15$	T_{m3}	T_{44}
1	0.7586780454					
		33597732				
2	0.7687573650		0.7721171382			
		8349354		13355		
3	0.7712621711		0.7720971065		0.7720957710	
		2084242		826		
4	0.7718874437		0.7720958678		0.7720957853	0.7720957855

Since none of the differences $|T_{3,i} - T_{3,i-1}| < 5\,10^{-9}$, the termination criterion mentioned above requires that the row with $m = 4$ must be computed. Then, the termination criterion is satisfied with a wide margin, since $|T_{44} - T_{43}| = 2\,10^{-10}$, and the irregular errors are less than 10^{-10}. T_{44} is even better than this error bound indicates; the correct result agrees with T_{44} to all the displayed decimal places.

Romberg's method has been one of the most widely used methods, since, among other things, it gives a simple strategy for the automatic determination of a suitable step size and order. One begins with a fairly large step $h_1 = H$, and then repeatedly halves the step size $h_i = h_{i-1}/2$, $i = 2, 3, \ldots$.

Theorem 3.3.6. *Error bound for Romberg's method.*

The items T_{mk} are estimates of the integral $\int_a^{a+h} f(x)\,dx$ that can be expressed as a linear functional,

$$T_{mk} = H \sum_{j=1}^{n} \alpha_{m,j}^{(k)} f(a+jh), \quad \text{where } n = 2^{m-1},\ h = H/n. \tag{3.3.39}$$

$$\sum_{j=1}^{n} \alpha_{m,j}^{(k)} = 1, \qquad \alpha_{m,j}^{(k)} > 0. \tag{3.3.40}$$

If the magnitude of the irregular error in $f(a+jh)$ is at most $\frac{1}{2}U$, then the magnitude of the inherited irregular error in T_{mk} is at most $H\frac{1}{2}U$.
The remainder functional for T_{mk} is zero for $f \in \mathcal{P}_{2k}$, and its Peano kernel is positive in the interval $(a, a+H)$. The truncation error of T_{mk} reads

$$T_{mk} - \int_a^{a+H} f(x)dx = r_k h^{2k} H f^{(2k)}(a + \tfrac{1}{2}H) + O(h^{2k+2} H f^{(2k+2)}) \tag{3.3.41}$$

$$= r_k h^{2k} H f^{(2k)}(\xi), \quad \xi \in (a, a+H), \tag{3.3.42}$$

$$r_k = 2^{k(k-1)} |B_{2k}|/(2k)!, \quad h = 2^{1-m} H. \tag{3.3.43}$$

Sketch of a proof: Eq.(3.3.39) follows directly from the construction of the Romberg scheme. It is for theoretical use only; the recursion formulas are better for practical use.
The first formula in (3.3.40) holds, because T_{mk} is exact if $f = 1$. The second formula is easily proved for low values of k. The general proof is more complicated, see [2, Theorem 4].
The Peano kernel for $m = k = 1$ (trapezoidal rule) was constructed in Sec. 3.2. For $m = k = 2$ (Simpson's rule), see Problem 3.2.12. The general case is more complicated. Recall that, by the corollary of Theorem 3.2.7, a remainder formula with a mean value $\xi \in (a, a+H)$, exists iff the Peano kernel does not change sign. Bauer, Rutishauser and Stiefel [2, pp. 207–210], constructed a recursion formula for the kernels, and succeeded in proving that they are all positive, by an ingenious use of the recursion. The expression for r_k is also derived there, although with a different notation. See also Problem 30. □

COMMENTS: In an error estimate of the form $ch^pHf^{(p)}(\cdot)$, p is called the *order of accuracy* or **order**. c is called the **error constant**. Note also that the integration formula is **exact in** $\mathcal{P}_p$, but not in $\mathcal{P}_q$ if $q > p$. Suppose that an integration formula is used with constant step size and order over the interval $[a, b]$, a so-called "long formula". The error estimate (or bound) becomes $ch^p(b-a)f^{(p)}(\cdot)$, i.e., $b-a$ is substituted for H. [83]

Example 3.3.23. *Comparison and combination of Romberg's and Newton–Cotes's methods.*

By working algebraically in the Romberg scheme, we obtain

$$\begin{aligned} T_{11} &= \tfrac{1}{2}H\left(f(a)+f(a+H)\right), \\ T_{21} &= \frac{1}{4}H\left(f(a)+f(a+\tfrac{1}{2}H)+f(a+\tfrac{1}{2}H)+f(a+H)\right) \\ &= \tfrac{1}{2}H\left(\tfrac{1}{2}f(a)+f(a+\tfrac{1}{2}H)+\tfrac{1}{2}f(a+H)\right), \\ T_{22} &= \frac{1}{3}(4T_{21}-T_{11}) = \frac{1}{6}H\left(f(a)+4f(a+\tfrac{1}{2}H)+f(a+H)\right). \end{aligned} \tag{3.3.44}$$

We see that T_{22} is the same as Simpson's formula, see Problem 3.2.12.

This can also be proved by a more "philosophical" argument. Recall that, by the corollary of Theorem 3.2.4, there is only one linear combination of the values of the function f at $n+1$ given points that can yield $\int_a^{a+H} f(x)\,dx$ exactly for all polynomials $f \in \mathcal{P}_{n+1}$.

Recall also the closed Newton–Cotes formulas (for numerical integration) that we here shall denote by C_n. See Example 3.2.12 and Problem 12.2.13. In the application to $\int_a^{a+H} f(x)\,dx$, C_n uses $n+1$ equidistant points, including the endpoints $x = a$ and $x = a+h$, and its order is $n+2$. C_1 is the trapezoidal rule, hence $C_1 = T_{11}$. For even values of n, the order of accuracy is equal to $n+2$. The remainder of C_n reads $r_nh^{n+2}Hf^{(n+2)}$. C_2, C_4, C_8 uses the same function values as T_{22}, T_{33}, T_{44}, respectively. Some data concerning these formulas are given in Table 3.3.23. Note that, by the corollary of Theorem 3.2.4,

$$C_2 = T_{22} = \text{Simpson}, \quad C_4 = T_{33}, \quad \text{but} \quad C_8 \neq T_{44},$$

because C_8 and T_{44} have different order.

Note that the remainder of T_{44} is $r_4h^8Hf^{(8)}(\xi) \approx r_4H\Delta^8 f(a)$, where $\Delta^8 f(a)$ uses the same function values as T_{44} and C_8. So we can use $r_4H\Delta^8 f(a)$ as an asymptotically correct error estimate for T_{44}.

More interesting: $T_{44} - r_4H\Delta^8 f(a)$ is an estimate of the integral that is exact in $\mathcal{P}_9$ at least, and C_8 is another estimate that has the same property and is based on the same 9 function values. By the same argument as before,

$$C_8 = T_{44} - r_4H\Delta^8 f(a), \qquad r_4 = 16/4725. \tag{3.3.45}$$

The term $r_4H\Delta^8 f(a)$ can still be used as an error estimate for C_8, usually a strong overestimate. Note, however, that $c_4H\Delta^{10} f(x)$ is an asymptotically correct

[83]This is one of the reasons for our definition of "error constant". You may find other definitions and other values in other texts.

Table 3.3.4. *Data concerning some Romberg and Newton–Cotes formulas.*

		order	order	error const.	error const.
$m = k$	n	T_{kk}	C_n	r_k	c_n
1	1	2	2	1/12	1/12
2	2	4	4	1/180	1/180
3	4	6	6	2/945	2/945
4	8	8	10	16/4725	296/467775

error estimates for C_8 if $x \approx a$. It requires extra function evaluations, unless (in a long computation) e.g., the actual bigstep has neighboring bigsteps, where the required function values are available.

The appearance of the eighth difference may seem frightening, when we think about the amplification of the irregular errors of the 1st column. It is, however, harmless. A bound for the inherited irregular errors of C_8 in this representation reads $\frac{1}{2}HU + \frac{16}{4725} \cdot H \cdot 256 \cdot \frac{1}{2}U < 0.93HU$.
A slightly lower bound for the irregular errors is obtained for C_8 in the representation $\sum_j \alpha_j f(a + jh)$, see (3.2.49). Although three of the nine coefficients are negative, the bound is as low as $\sum_j |\alpha_j| H \frac{1}{2} U < 0.73HU$.

It is a little disappointing that something that should have become a super-improvement of Romberg's method turns out to be a special case of Newton–Cotes's method. A tenth order method is most interesting when high accuracy is needed. A feature of the Romberg algorithm is that it also contains exits with lower accuracy at a lower cost, but this is more of a topic for Chapter 5.

Example 3.3.24. *Applications to differential equations.* Under reconstruction.

3.3.6 Complete Monotonicity and Related Concepts.

A *sequence* $\{u_n\}$ is **completely monotonic, (c.m.)** for $n \geq a$ iff

$$u_n \geq 0,\ (-\Delta)^j u_n \geq 0, \quad \forall j \geq 0,\ n \geq a,\ \text{(integers)}.$$

The abbreviation c.m. will be used, both as an adjective and as a noun, and both in singular and in plural. The abbreviation **d.c.m.** will similarly be used for the difference between two completely monotonic sequences. (These abbreviations are not generally established.)

A c.m. sequence $\{u_n\}_0^\infty$ is **minimal** iff it ceases to be a c.m. if u_0 is decreased, while all the other elements are unchanged. This distinction is of little importance to us, since we usually deal with a tail of some given c.m., and it can be shown (Problem 47) that *if* $\{u_n\}_0^\infty$ *is a c.m. then* $\{u_n\}_1^\infty$ *is a minimal c.m.*. Note, e.g., that the sequence $\{1, 0, 0, 0, \ldots\}$ is a non-minimal c.m.,while $\{0, 0, 0, 0, \ldots\}$ is a minimal c.m..

Unless it is stated otherwise *we shall only deal with minimal c.m.* without stating this explicitly all the time.

Similarly a **function** $u(s)$ is *c.m.* for $s \geq a, s \in \mathbf{R}$, iff

$$u(s) \geq 0,\ (-1)^{(j)} f^{(j)}(s) \geq 0, \quad s \geq a, \forall j \geq 0 \text{ (integer)},\ \forall s \geq a,\ \text{(real)}.$$

$u(s)$ is *d.c.m* if it is difference of two c.m. on the same interval.

We also need variants with an open interval. For example, the function $u(s) = 1/s$ is c.m. in the interval $[a, \infty)$ for any positive a, but it is not c.m. in the interval $[0, \infty]$.

The simplest relation of c.m. functions and c.m. sequences reads: if the function $u(s)$ is c.m. for $s \geq s_0$ then the sequence defined by $u_n = u(s_0 + hn)$, $(h > 0)$, $n = 0, 1, 2, \ldots$ is also c.m. since, by (3.2.4), $(-\Delta)^j u_n = (-hD)^j u(\xi) \geq 0$ for some $\xi \geq s_0$.

A function is **absolutely monotonic** in an (open or closed) interval, if the function and all its derivatives are non-negative there.

The main reason why the analysis of a numerical method is convenient for c.m. and d.c.m. is that the they are "linear combinations of exponentials", according to the theorem below. The more precise meaning of this requires the important concept of a **Stieltjes integral**, $\int_a^b f(x) d\alpha(x)$, which is defined as the limit of sums of the form $\sum_i f(\xi_i)\big(\alpha(x_{i+1}) - \alpha(x_i)\big)$ where $a = x_0 < x_1 < x_2 < \ldots < x_N = b$, $\xi_i \in [x_i, x_{i+1}]$. Here $f(x)$ *is bounded and continuous*, and $\alpha(x)$ is of *bounded variation* in $[a, b]$, i.e. the difference between two non-decreasing and non-negative functions. The extension to improper integrals where, e.g., $b = \infty$, $\alpha(b) = \infty$, is made in a similar way as for Riemann or Lebesgue integrals. The Stieltjes integral is much used also in Probability and Mechanics, since it unifies the treatment of continuous and discrete (and mixed) distributions of probability or mass. If $\alpha(x)$ is piecewise differentiable, then $d\alpha(x) = \alpha'(x)dx$, and the Stieltjes integral is simply $\int_a^b f(x)\alpha'(x)dx$. If $\alpha(x)$ is a *step function*, with jumps (also called point masses) m_i at $x = x_i$, then

$$d\alpha(x_i) = \lim_{\epsilon \downarrow 0} \alpha(x_i + \epsilon) - \alpha(x_i - \epsilon) = m_i, \quad \int_a^b f(x) d\alpha(x) = \sum_i m_i f(x_i).$$

Integration by parts is as usual; the following example is of interest to us. Suppose that $\alpha(0) = 0$, $\alpha(x) = o(e^{cx})$ as $x \to \infty$, and that $\Re s \geq c$. Then

$$\int_0^\infty e^{-sx} d\alpha(x) = s \int_0^\infty \alpha(x) e^{-sx} dx, \tag{3.3.46}$$

The integral on the left side is called a **Laplace–Stieltjes transform**, while the integral on the right side is an ordinary Laplace transform. Many properties of power series, though not all, can be generalized to Laplace–Stieltjes integrals—set $z = e^{-s}$. Instead of a disk of convergence, the Laplace–Stieltjes integral has a (right) half-plane of convergence. A difference is that the half-plane of absolute convergence may be different from the half-plane of convergence.

We shall be rather brief and concentrate on the applicability to the study of numerical methods. We refer to Widder [37], [38], for proofs and more precise information concerning both Stieltjes integrals, Laplace transforms and complete monotonicity. Dahlquist [9] gives more details about applications to numerical methods.

The sequence defined by

$$u_n = \int_0^1 t^n d\beta(t), \ n = 0, \ 1, \ 2, \ldots, \tag{3.3.47}$$

is called a **moment sequence** if $\beta(t)$ is non-decreasing. We make the *convention that* $t^0 = 1$ *also for* $t = 0$, since the continuity of f is required in the definition of the Stieltjes integral.

Consider the special example, where $\beta(0) = 0, \beta(t) = 1$ if $t > 0$. This means a unit point mass at $t = 0$, and no more mass for $t > 0$. Then $u_0 = 1$, $u_n = 0$ for $n > 0$. It is then conceivable that making a sequence minimal just means to remove a point mass from the origin; thus *minimality means to require that* $\beta(t)$ *is continuous at* $t = 0$. (For a strict proof, see [37, § 4.14],)

The following theorem combines parts of several theorems in the books of Widder. It is important that the the functions called $\alpha(x), \beta(t)$ in this theorem *need not to be explicitly known for an individual series*, for applications of an error estimate or a convergence rate etc. of a method of convergence acceleration. Some criteria will be given below that can be used for simple proofs that a particular series is (or is not) c.m. or d.c.m..

Theorem 3.3.7.

1. *The sequence* $\{u_n\}_0^\infty$ *is a c.m., iff it is a moment sequence; it is* minimal *if in addition* $\beta(t)$ *is continuous at* $t = 0$*, i.e., if there is no point mass at the origin. It is a d.c.m., iff (3.3.47) holds for some* $\beta(t)$ *of bounded variation.*
2. *The function* $u(s)$ *is a c.m. for* $s \geq 0$*, iff it can be represented as a Laplace–Stieltjes transform,*

$$u(s) = \int_0^\infty e^{-sx} d\alpha(x), \quad s \geq 0, \tag{3.3.48}$$

with a non-decreasing and bounded function $\alpha(x)$*. For the open interval,* $s > 0$ *we have the same, except for the boundedness of* $\alpha(x)$*. For a d.c.m. the same is true with* $\alpha(x)$ *of bounded variation, (not necessarily bounded as* $x \to \infty$*). The integral representation provides an analytic continuation of* $u(s)$ *from a real interval to a half-plane.*

3. *The sequence* $\{u_n\}_0^\infty$ *is a minimal c.m., iff there exists a completely monotonic function* $u(s)$*, such that* $u_n = u(n)$*,* $n = 0, 1, 2, \ldots$*.*
4. *Suppose that* $u(s)$ *is c.m. in the interval* $s > a$*. Then the Laplace–Stieltjes integral converges absolutely and uniformly if* $\Re s \geq a'$*, for any* $a' > a$*, and defines an analytic continuation of* $u(s)$ *that is bounded for* $\Re s \geq a'$ *and analytic for* $\Re s > a$*. This is true also if* $u(s)$ *is a d.c.m..*

COMMENTS: The "only if" parts of these "iff" statements are deep results mainly due to Hausdorff and Bernstein, and we omit the rather technical proofs. The relatively simple proofs of the "if" parts of the first three statements will be sketched, since they provide some useful insight.

1. Assume that u_n is a moment sequence, $\beta(0) = 0$, β is continuous at $t = 0$ and non-decreasing for $t > 0$. *Note that multiplication by E or Δ outside the integral sign in (3.3.47) corresponds to multiplication by t or $t-1$ inside.* Then, for $j, n = 0, 1, 2, \ldots$,

$$(-1)^j \Delta^j u_n = (-1)^j \int_0^1 (t-1)^j t^n d\beta(t) = \int_0^1 (1-t)^j t^n d\beta(t) \geq 0,$$

hence u_n is c.m.

2. Assume that $u(s)$ satisfies (3.3.48). It is rather easy to legitimate the differentiation under the integral sign in this equation. Differentiation j times with respect to s yields, for $j = 1, 2, 3, \ldots$

$$(-1)^j u^{(j)}(s) = (-1)^j \int_0^\infty (-x)^j e^{-sx} d\alpha(x) = \int_0^\infty x^j e^{-sx} d\alpha(x) \geq 0,$$

hence $u(s)$ is c.m.

3. Assume that $u_n = u(n) = \int_0^\infty e^{-nx} d\alpha(x)$. Define $t = e^{-x}$, $\beta(0) = 0$, $\beta(t) \equiv \beta(e^{-x}) = u(0) - \alpha(x)$, and note that

$$t = 1 \Leftrightarrow x = 0, \quad t = 0 \Leftrightarrow x = \infty,$$

and that $u(0) = \lim_{x\to\infty} \alpha(x)$. It follows that $\beta(t)$ is non-negative and non-decreasing, since x decreases as t increases. Note that $\beta(t) \downarrow \beta(0)$, as $t \downarrow 0$. Then

$$u_n = -\int_1^0 t^n d\beta(t) = \int_0^1 t^n d\beta(t),$$

hence $\{u_n\}$ is a minimal c.m.

4. The distinction is illustrated for $\alpha'(x) = e^{ax}$, $u(s) = (s-a)^{-1}$, for a real a. $u(s)$ is analytic for $\Re s > a$ and bounded only for $\Re s \geq a'$ for any $a' > a$.

The basic formula for the application of complete monotonicity to the summation of power series reads

$$S(z) \equiv \sum_{i=0}^\infty u_i z^i = \sum_0^\infty \int_0^1 z^i t^i d\beta(t) = \int_0^1 \sum_0^\infty z^i t^i d\beta(t) = \int_0^1 (1-zt)^{-1} d\beta(t). \tag{3.3.49}$$

The inversion of the summation and integration is legitimate when $|z| < 1$. Note that the last integral exists for more general z; a classical principle of Complex Analysis then yields the following interesting result.

Lemma 3.3.8. *If the sequence $\{u_i\}$ is d.c.m., then the last integral of formula* (3.3.49) *provides the unique single-valued* analytic continuation *of $S(z)$ to the whole complex plane, save for a cut along the real axis from* 1 *to* ∞.

COMMENT: When z is located in the cut, $(1-zt)^{-1}$ has a non-integrable singularity at $t = 1/z \in [0,1]$ unless, e.g., $\beta(t)$ is constant in the neighborhood of this point. If we remove the cut, $S(z)$ will not be single-valued. Check that this makes sense for $\beta(t) = t$.

Next we shall apply the above results to find interesting properties of the (generalized) Euler Transformation. For example, we shall see that, for any z outside the cut, there is an optimal strategy for the generalized Euler Transformation that provides the unique value of the analytic continuation of $S(z)$. We shall see in § 3.3.7 that this also holds for Gustafson's Chebyshev acceleration. The classical Euler Transformation, however, reaches only the half-plane $\Re z < \frac{1}{2}$.

After that we shall see that there are a number of simple criteria for finding out whether a given sequence is a c.m., a d.c.m. or neither. Many interesting sequences are c.m., e.g., $u_n = e^{-kn}$, $u_n = (n+c)^{-k}$, ($k \geq 0$, $c \geq 0$), all products of these and all linear combinations (i.e., sums or integrals) of such sequences with positive coefficients.

The convergence of a c.m. towards zero can be arbitrarily slow, but an alternating series with c.m. terms will after Euler's transformation converge as rapidly as a geometric series. More precisely, the following will be shown.

Theorem 3.3.9. *On the optimal use of a generalized Euler Transformation.*

We use the notation of Theorem 3.3.1 and Eq. (3.3.18). Suppose that the sequence $\{u_j\}$ is either c.m. or d.c.m. Consider $S(z) = \sum_{j=0}^{\infty} u_j z^j$, $z \in \mathbf{C}$, and its analytic continuation (according to the above lemma). Then:

- *If $z = -1$, a sequence along a descending diagonal of the scheme M or (equivalently) the matrix $\bar{M}$, i.e., $\{M_{n_0,k}\}_{k=0}^{\infty}$ for a fixed n_0, converges at least as fast as 2^{-k}.*

- *More generally: the error behaves like $(\frac{z}{1-z})^k$, $(k \gg 1)$. Note that $|\frac{z}{1-z}| < 1$ iff $\Re z < \frac{1}{2}$. The classical Euler transformation diverges outside this half-plane. If $z = e^{\pm it}$, $\frac{\pi}{3} < t \leq \pi$, it converges as fast as $(2\sin\frac{t}{2})^{-k}$.*

The results above are for the classical *Euler transformation. For the* generalized *Euler transformation we have the following:*

- *If $z = -1$ the smallest error in the i'th row of $\bar{M}$, is $O(3^{-i})$, as $i \to \infty$.*

- *More generally: this error is $O\big((\frac{|z|}{1+|1-z|})^i\big)$, hence the smallest error converges exponentially, unless $z-1$ is real and positive, i.e., the optimal application of the generalized Euler's transformation provides the analytic continuation, whenever it exists according to Lemma 3.3.8. If $N \gg 1$, the optimal value of k/N is $\frac{|1-z|}{|1-z|+1}$.[84] If $z = e^{\pm it}, 0 < t \leq \pi$, the error is $O\big((1+2\sin\frac{t}{2})^{-i}\big)$.*

[84]In practice this is approximately found by the termination criterion of the algorithm in Sec 3.2.3.

Proof. Sketch: The results of the generalized Euler transformation is in Sec. 3.3.3 denoted by $M_{n,k}(z)$. The computation uses $N = n + k$ terms (or partial sums) of the power series for $S(z)$; n terms of the original series—the head—are added, and Euler's transformation is applied to the next k terms—the tail. Set $n/N = \mu$, i.e., $n = \mu N$, $k = (1-\mu)N$, and denote the error of $M_{n,k}$ by $R_{N,\mu}(z)$. Euler's transformation is based on the operator $P = P(z) = \frac{z\Delta}{1-z}$. A multiplication by the operator P corresponds to a multiplication by $\frac{z(t-1)}{1-z}$ inside the integral sign $\int_0^1$.

First suppose that $|z| < 1$. By the definitions of $S(z)$ and $M_{n,k}(z)$ in Theorem 3.3.1,

$$R_{N,\mu}(z) \equiv S - M_{n,k} = \frac{z^n}{(1-z)} \sum_{s=k}^{\infty} P^s u_n = \frac{z^n}{(1-z)} \int_0^1 \sum_{s=k}^{\infty} \Big(\frac{z(t-1)}{(1-z)}\Big)^s t^n d\beta(t),$$

$$R_{N,\mu}(z) = \frac{z^n}{(1-z)} \int_0^1 \Big(\frac{z(t-1)}{(1-z)}\Big)^k \frac{t^n d\beta(t)}{1 - z(t-1)/(1-z)} \tag{3.3.50}$$

$$= (-1)^k \frac{z^N}{(1-z)^k} \int_0^1 (1-t)^k t^n \frac{d\beta(t)}{1-zt}. \tag{3.3.51}$$

$$\tag{3.3.52}$$

We see that the error oscillates as stated in Sec. 3.3.2. Again, by analytic continuation, this holds for all z save for the real interval $[1, \infty]$. Then

$$|R_{N,\mu}(z)|^{1/N} \le |z/(1-z)^{1-\mu}| \max_{t\in[0,1]} ((1-t)^{1-\mu} t^{\mu}) c^{1/N}, \quad c = \int_0^1 \frac{|d\beta(t)|}{|1-zt|}.$$

The *first* part of the theorem has $n = 0$, hence $\mu = 0$. We obtain $\lim_{N\to\infty} |R_{N,0}^{1/N}| \le |z/(1-z)|$ as $N \to \infty$, as stated. This is less than unity if $|z| < |1-z|$, i.e., if $\Re(z) < \frac{1}{2}$.

Now we consider the *second* part of the theorem. The maximum occurring in the above expression for $|R_{N,\mu}(z)|^{1/N}$ (with N, μ fixed) takes place at $t = \mu$. Hence

$$|R_{N,\mu}(z)|^{1/N} \le |z/(1-z)^{1-\mu}| c^{1/N} (1-\mu)^{1-\mu} \mu^{\mu}.$$

An elementary optimization shows that *the value of μ that minimizes this bound for $|R_{N,\mu}(z)|^{1/N}$ is $\mu = \frac{1}{|1-z|+1}$*, i.e., $k = (1-\mu)N = \frac{N|1-z|}{|1-z|+1}$, and the minimum equals $\frac{|z|}{|1-z|+1}$. The details of these two optimizations are left for Problem 41. This proves the second part of the theorem. □

This minimum turns out to be a rather realistic estimate of the convergence ratio of the *optimal* generalized Euler-transformation for power series with d.c.m. coefficients, unless $\beta(t)$ is practically constant in some interval around $t = \mu$; the exception happens, e.g., if $u_n = a^n$, $0 < a < 1$, $a \ne \mu$, see Problem 40.

Example 3.3.25. *Criteria for higher monotonicity.*
We shall here list a few criteria, by which one can often answer the question whether

a *function* is a c.m or a d.c.m or neither. When several c.m. or d.c.m. are involved, the intervals should be reduced to the intersection of the intervals involved. By Theorem 3.3.7, the question is then settled also for the corresponding *sequence*. In simple cases the question can be answered directly by means of the definition or the above theorem, e.g., for $u(s) = e^{-ks}$, s^{-k}, $(k \geq 0)$, for $\Re s \geq 0$ in the first case, for $\Re s > 0$ in the second case.

The problems of Sec.3.3. contain many interesting examples that can be treated by means of these criteria, one of the most important is Problem 42: every rational function that is analytic and bounded in a half-plane is a d.c.m. there.

Sometimes a table of Laplace transforms, see e.g., Abramowitz and Stegun, can be useful in combination with the criteria below.

(A) *If $u(s)$ is c.m., and $a, b \geq 0$, then $g(s) = u(as + b)$ and $(-1)^j u^{(j)}(s)$, are c.m., $j = 1, 2, 3, \ldots$.* $\int_s^\infty u(t)\,dt$ is also c.m., if the integral is convergent. (The interval of complete monotonicity may not be the same for g as for f). Analogous statements hold for sequences.

(B) *The product of two c.m. is c.m. Similarly, the product of two d.c.m. is d.c.m. This can evidently be extended to products of any number of factors, and hence to every positive integral power of a c.m. or d.c.m..* The proof is left for Problem 43.

(C) *A uniformly convergent positive linear combination of c.m. is itself c.m.. The same criterion holds for d.c.m., without the requirement of positivity.* The term "positive linear combination" includes sums with positive coefficients and, more generally, Stieltjes integrals $\int u(s; p) d\gamma(p)$, where $\gamma(p)$ is non-decreasing.

(D) *Suppose that $u(s)$ is a d.c.m. for $s \geq a$. $F(u(s))$ is then a d.c.m for $s > a$, if the radius of convergence of the Taylor expansion for $F(z)$ is greater than* $\max |u(s)|$.
Suppose that $u(s)$ is a c.m. for $s \geq a$. We must then add the assumption that the coefficients of the Taylor expansion of $F(z)$ are non-negative, in order to make sure that $F(u(s))$ is c.m for $s \geq a$. These statements are important particular cases of (C). We also used (B), according to which each term $u(s)^k$ is a c.m. (or a d.c.m. in the first statement).
Two illustrations: $g(s) = (1 - e^{-s})^{-1}$ is a c.m. for $s > 0$;
$h(s) = (s^2 + 1)^{-1}$ a d.c.m at least for $s > 1$ (choose $z = s^{-2}$). The expansion into powers of s^{-2} also provides an explicit decomposition $h(s) = (s^{-2} + s^{-6} + \ldots) - (s^{-4} + s^{-8} + \ldots) = \frac{s^2}{s^4 - 1} - \frac{1}{s^4 - 1}$ where the two components are c.m.for $s > 1$. See also Example 3.3.26.

(E) *If $g'(s)$ is c.m. for $s > a$, and if $u(z)$ is c.m. in the range of $g(s)$ for $s > a$, then $F(s) = u(g(s))$ is c.m. for $s > a$.* (Note that $g(s)$ itself is not c.m.)
For example, we shall show that $1/\ln s$ is c.m. for $s > 1$. Set $g(s) = \ln s$, $u(z) = z^{-1}$, $a = 1$. Then $u(z)$ is c.m. for $z > 0$, and $g'(s) = s^{-1}$ is c.m. for $s > 0$, a fortiori for $s > 1$ where $\ln s > 0$. Then the result follows from (E).

Another set of criteria is related to the *analytic properties of the c.m and d.c.m.*. Let $u(s)$ be a d.c.m for $s > a$. According to statement 4 of Theorem 3.3.7, $u(s)$ is analytic and bounded for $s \geq a'$ for any $a' > a$. The converse of this is not unconditionally true. If, however, we add the conditions that

$$\int_{-\infty}^{\infty} |u(\sigma + i\omega)| d\omega < \infty, \quad u(s) \to 0, \quad \text{as } |s| \to \infty, \ \sigma \ge a', \tag{3.3.53}$$

then it can be shown that $u(s)$ is a d.c.m. for $s > a$. This condition is rather restrictive; there are many d.c.m. that do not satisfy it, e.g., functions of the form e^{-ks} or $k + b(s-c)^{-\gamma}$, ($k \ge 0$, $b \ge 0$, $c > a$, $0 < \gamma \le 1$). The following, however, is a reasonably powerful criterion: *$u(s)$ is a d.c.m. for $s > a$,* e.g., *if we can make a decomposition of the form $u(s) = f_1(s) + f_2(s)$ or $u(s) = f_1(s)f_2(s)$, where $f_1(s)$ is known to be d.c.m for $s > a$, and $f_2(s)$ satisfies the conditions in* (3.3.53).

Theorem 3.3.10.

Suppose that $u(s)$ is c.m. for some s though not for all s. Then a singularity on the real axis, at (say) $s = a$, must be among the rightmost singularities; $u(s)$ is c.m. for $s > a$, hence analytic for $\Re s > a$.

This is not generally true if $u(s)$ is only a d.c.m.. Suppose that $u(s)$ is d.c.m for $s > a$, though not for any $s < a$. Then we cannot even be sure that there exists a singularity s^* such that $\Re s^* = a$.

The first statement is analogous to a familiar theorem about power series with positive coefficients, but the statement about the d.c.m. contrasts with the situation for power series; the latter always have a singularity on the boundary of the region of convergence.

Example 3.3.26.

This theorem can be used for finding that a given function is *not* a c.m., e.g. $u(s) = 1/(1+s^2)$ is not a c.m., since the rightmost singularities are $s = \pm i$, while $s = 0$ is no singularity. $u(s)$ is a d.c.m for $s > 0$, however, since it is analytic and bounded, and satisfies (3.3.53), for any positive a'. This result also comes from the general statement about rational functions bounded in a half-plane, see Problem 42.

Another approach: in any text about Laplace transforms you find that, for $s > 0$, $\frac{1}{s^2+1} = \int_0^\infty e^{-sx} \sin x dx = \int_0^\infty e^{-sx}(1+\sin x)dx - \int_0^\infty e^{-sx} dx$. Now $\alpha'(x) \ge 0$ in both terms. Hence the formula $\left(\frac{1}{s} + \frac{1}{s^2+1}\right) - \frac{1}{s}$ expresses $\frac{1}{s^2+1}$ as the difference of two c.m. for $s > 0$.

The easy application of criterion (D) above gave a smaller interval ($s > 1$), but a faster decrease of the c.m. terms as $s \to \infty$.

Another useful criterion for this kind of negative conclusion is that *a c.m. cannot decrease faster than* every *exponential as $s \to +\infty$, for $s \in \mathbf{R}$*, unless it is identically zero. For there exists a number ξ such that $\alpha(\xi) > 0$, hence $u(s) = \int_0^\infty e^{-sx} d\alpha(x) \ge \int_0^\xi e^{-sx} d\alpha(x) \ge e^{-s\xi}\alpha(\xi)$. For example, e^{-s^2} and $1/\Gamma(s)$ are not c.m..

Why does this not contradict the fact that $s^{-1}e^{-s}$ is a c.m.?

These ideas can be generalized. Suppose that $\{c_i\}_{i=0}^\infty$ is a given sequence, such that the sum $C(t) \equiv \sum_{i=0}^\infty c_i t^i$ is known, and that u_i is c.m. or d.c.m.. (c_i and $C(t)$

may depend on a complex parameter z too.) Then

$$S_c = \sum_{i=0}^{\infty} c_i u_i = \sum_{i=0}^{\infty} c_i \int_0^1 t^i d\beta(t) = \int_0^1 C(t) d\beta(t).$$

It is natural to ask how well S_c is determined if u_i has been computed for $i < N$, if $\{u_n\}_0^\infty$ is constrained to be a c.m.. In Ch.9 we shall find a simple characterization of the monotonic functions $\beta(t)$ which give the *best* upper and lower bounds of S_c, in the case that $C^{(N)}(t)$ has constant sign for $t \in [0,1]$—for $C(t) = (1-zt)^{-1}$ this is applicable iff $z < 1$, $(z \in \mathbf{R})$.

Another systematic way to obtain *very good* bounds, with less restrictions, is to *search a polynomial* $Q \in \mathcal{P}_N$, *such that* $|C(t) - Q(t)| \le \epsilon_N, \forall t \in [0,1]$. *Then*

$$|S_c - Q(E)u_0| = |\int_0^1 \big(C(t) - Q(t)\big) d\beta(t)| \le \epsilon_N \int_0^1 |d\beta(t)|.$$

Note that $Q(E)u_0$ is a linear combination of the computed values u_i, $i < N$, with coefficients independent of $\{u_n\}$. For $C(t;z) = (1-tz)^{-1}$ the generalized Euler-transformation (implicitly) works with a particular array of polynomial approximations, based on Taylor expansion, first at $t = 0$, then at $t = 1$.

Can we find better polynomial approximations? In its general form, this question also belongs to Ch.9., but for $C(t;z) = (1-tz)^{-1}$, an algorithm will be described in the next subsection, which is, in most respects, superior to the optimal Euler-transformation.

3.3.7 Gustafson's Chebyshev acceleration (GCA).

Gustafson's Chebyshev acceleration, Gustafson (1979), (short GCA) is like Euler's Transformation based on linear transformations of sequences and it has the same range of application as the optimal Euler Transformation.

We shall study GCA for power series with d.c.m. coefficients.[85] In this rather general case it is superior to the optimal Euler Transformation, because it is related to a better sequence of polynomial approximations to $(1-tz)^{-1}$, $t \in [0,1)$,(cf. the end of the previous subsection). The superiority is greatest for small values of $\arg z$, see Example 3.3.27. ,

Set, as in Sec. 3.3.6,

$$S(z) \equiv \sum_{i=0}^{\infty} u_i z^i = \int_0^1 \frac{d\beta(t)}{1-zt},$$

where $\beta(t)$ is of bounded variation. Recall Lemma 3.3.8 concerning the analytic continuation of $S(z)$. First, compute

$$I_{n,0} = z^{-n}(u_0 + u_1 z + \ldots + u_{n-1} z^{n-1}),\ n = 1:N, \quad \text{i.e.,} \quad I_{n,0} = \frac{z^{-n} - E^n}{1 - zE} u_0. \tag{3.3.54}$$

[85] d.c.m. is defined in the beginning of the previous subsection.

The basic idea of Gustafson's acceleration algorithms is to construct a sequence of polynomials $\psi_k(t)$, $\deg\psi_k = k$ with unit leading coefficients, such that $\max_{t\in[0,1]}|\psi_k(t)|/|\psi_k(\frac{1}{z})|$ decreases rapidly when k increases. These polynomials should satisfy a three term recursion formula. Define

$$I_{n,k} = \frac{z^{-n}\psi_k(\frac{1}{z}) - E^n\psi_k(E)}{1 - zE}u_0. \tag{3.3.55}$$

Since $\frac{z^{-n}\psi_k(\frac{1}{z}) - t^n\psi_k(t)}{1-zt} \in \mathcal{P}_{n+k}[t]$, $I_{n,k}$ is a linear combination of u_j, $j = 0 : n+k-1$. By means of a recursion formula given below, we eventually obtain $I_{0,k}$, $k = 1 : N$. Note that

$$\frac{I_{0,k}}{\psi_k(\frac{1}{z})} = \frac{1 - \psi_k(E)/\psi_k(1/z)}{1 - zE}u_0 = \int_0^1 \frac{d\beta(t)}{1 - zt} - \int_0^1 \frac{\psi_k(t)d\beta(t)}{\psi_k(\frac{1}{z})(1 - zt)} \equiv S(z) - R_k(z), \tag{3.3.56}$$

i.e., $I_{0,k}/\psi_k(\frac{1}{z})$, $k = 1 : N$, *is a sequence of estimates of* $S(z)$, and $R_k(z)$ are the remainders. Note that

$$|R_k(z)| \le \frac{c\cdot\max_{t\in[0,1]}|\psi_k(t)|}{|\psi_k(1/z)|}, \quad c = \int_0^1 \frac{|d\beta(t)|}{|1 - zt|}. \tag{3.3.57}$$

In GCA,[86] one chooses $\psi_k(t) = a_kT_k(2t-1)$, where $T_k(x)$ is the Chebyshev polynomial defined in Sec. 3.1.5. (Note that $[0,1] \mapsto [-1,1]$ by the substitution $x = 2t-1$.) Then it can be shown (Problem 48) that

$$|R_k(z)| \le \frac{c}{|T_k(\frac{2}{z} - 1)|} \sim c\left|\frac{2}{z} - 1 + 2\sqrt{\frac{1}{z^2} - \frac{1}{z}}\right|^k, \quad k \gg 1. \tag{3.3.58}$$

(The branch of the square root is chosen so that it is positive for $z < 1$.) It can be shown, Problem 49, that the process converges exponentially to $S(z)$ for every z in the complex plane with a cut from 1 to $+\infty$ along the real axis.

We have assumed that the leading coefficient of $\psi_k(t)$ is unity. Since the leading coefficient of $T_k(x)$ is 2^{k-1} if $k > 0$, and 1 if $k = 0$, it follows that $a_k = 2^{1-2k}$ if $k \ge 1$, and $a_0 = 1$.

The recursion formula (difference equation) for $T_k(x)$ given in Sec. 3.1.5 becomes, after an elementary transformation, Problem 49,

$$\psi_k(t) = (t - \tfrac{1}{2})\psi_{k-1}(t) - b_k\psi_{k-2}(t),\ (k \ge 1), \quad \psi_0(t) = 1,\ \psi_{-1}(t) = 0, \tag{3.3.59}$$

where $b_k = a_{k-2}/a_k$, i.e. $b_k = \frac{1}{16}$ if $k \ge 3$, $b_2 = \frac{1}{8}$, b_1 is arbitrary. It can then be shown that the quantities $I_{n,k}$ satisfy the recursion formula, (Problem 49),

$$I_{n,k} = I_{n+1,k-1} - \tfrac{1}{2}I_{n,k-1} - b_kI_{n,k-2}, \quad (k \ge 1). \tag{3.3.60}$$

Note that this is a kind of vector version of the difference equation in (3.3.59), if the operator E is substituted for t.

[86] Gustafson also considers other families of polynomials for special purpose.

Initial conditions: $I_{n,0}$ is obtained by (3.3.54), and $I_{n,-1} = 0$.
The results are $I_{0,k}(z)/\psi_k(\frac{1}{z})$, $k = 1 : N$, where we obtain $\psi_k(\frac{1}{z})$ by setting $t = \frac{1}{z}$ in the above recursion formula for $\psi_k(t)$. An experimental study of the numerical stability of this algorithm is left for Problem 50.

Example 3.3.27.

This study is concerned with power series with d.c.m. coefficients. By (3.3.58). By (3.3.58), the convergence ratio for GCA is $\eta_{GCA} \approx |\frac{2}{z} - 1 + 2\sqrt{\frac{1}{z^2} - \frac{1}{z}}|$, if $k \gg 1$. The approximate number of decimal digits gained per term equals $-\log_{10}\eta_{GCA}$.

For an alternating series, i.e., if $z = -1$, we have $\eta_{GCA} = 3 - \sqrt{8} \approx 0.1716$, $-\log_{10}\eta_{GCA} = 0.766$, hence the number of terms needed for "full accuracy" (16 decimal digits) is at most $16/0.766 \approx 21$.

By Theorem 3.3.9, the convergence ratios for the optimal Euler Transformation and the classical Euler Transformation are, for $z = -1$, $\eta_{OE} = \frac{1}{3}$ and $\eta_{CE} = \frac{1}{2}$, respectively. We similarly find that the number of terms needed for "full accuracy" are at most 34 and 53, respectively.

We see that GCA, for $z = -1$, is $\log(3 - \sqrt{8})/\log\frac{1}{3} \approx 1.6$ times as fast as the optimal Euler transformation, which is $\log 3/\log 2 \approx 1.6$ times as fast as the classical Euler transformation. For $z = e^{i\phi}$, $|\phi| \ll 1$, its superiority is much greater, but the rounding errors may cause trouble. This trouble can be reduced by the use of thinning, see Problem 51.

These figures agree rather well with experimental results for alternating series with c.m. and d.c.m. terms, irrespective of the rate of convergence of the original series. If you want (say) 8 decimals accuracy, you need (approximately) only half the number of terms etc. The results are asymptotic (valid for $k \gg 1$). For moderately large k, it may be advisable to count k from the left bound of the half-plane of analyticity for $u(s)$, unless $u(s)$ is an entire function.

Review Questions

1. Describe three procedures for improving the speed of convergence of certain series. Give examples of their use.

2. (a) What pieces of information appear in the Euler–Maclaurin formula? Give the generating function for the coefficients. What do you know about the remainder term?

(b) Give at least three important uses of the Euler–Maclaurin formula.

Problems and Computer Exercises

1. (a) Compute $\sum_{n=1}^{\infty}(n+1)^{-3}$ to 4 decimal places [87] by using $\sum_{n=x}^{\infty}\frac{1}{n(n+1)(n+2)}$, for a suitable x as a comparison series. Estimate roughly how many terms you would have to add without and with the comparison series. *Hint:* You find the exact sum of this comparison series in Problem 3.2.2.

(b) Compute the sum also by Euler–Maclaurin's formula or one of its variants in Problem 22(a).

2. Study, or write yourself, programs for some of the following methods [88]:

- iterated Aitken acceleration
- modified iterated Aitken, according to (3.3.5) or an a-version.
- generalized Euler transformation
- GCA, (see Sec. 3.3.7)
- ϵ algorithm, (see Sec. 3.4.4)
- one of the central difference variants of Euler–Maclaurin's formula, given in Problem 22(a)

The programs are needed in two slightly different versions:

VERSION I: for studies of the convergence rate, for a series (sequence) where one knows *exa* = a sufficiently accurate value of the sum (the limit). The risk of drowning in figures becomes smaller, if you make graphical output, e.g. like Fig. 3.3.1.

VERSION II: for a run controlled by a tolerance, like in Algorithm 3.3.1, appropriately modified for the various algorithms. Print also i and, if appropriate, jj. If *exa* is known, it should be subtracted from the result, because it is of interest to compare *errest* with the actual error.

COMMENT: If you do not know *exa*, find a sufficiently good *exa* by a couple of runs with very small tolerances, before you study the convergence rates (for larger tolerances).

3. (a) Try iterated Aitken with thinning for $\sum_1^{\infty} e^{-\sqrt{n}}$, according to the suggestions after Example 3.3.4.

(b) Study the effect of small random perturbations to the terms.

4. *Oscillatory series.*
Suggested series of the form $\sum_{n=1}^{\infty} c_n z^n$.
$c_n = e^{-\sqrt{n}},\ 1/(1+n^2),\ 1/n,\ 1/(2n-1),\ n/(n^2+n+1),\ 1/\sqrt{n},\ 1/\ln(n+1)$;
$z = -1,\ -0.9,\ e^{i3\pi/4},\ i,\ e^{i\pi/4},\ e^{i\pi/16}$;
for the appropriate algorithms mentioned above. Apply thinning. Try also classical Euler transformation on some of the cases.

Study how the convergence ratio depends on z, and compare with the theoretical results in § 3.3.6. Compare the various methods with each others.

5. *Essentially positive series.* $n^{-3/2}, n^{-3/2}, \log n$;

[87] This is for a handheld calculator; choose a smaller tolerance on a computer.

[88] G.D. has had Matlab in mind, and hopes that, in the future, some Matlab programs can be electronically available, under certain conditions. You are, of course, welcome to use another language that is also convenient for numerical experiments of this sort, with complex arithmetic and graphical output, etc.

Suggested series of the form
$\sum_{n=1}^{\infty} c_n z^n$, $c_n = e^{-\sqrt{n}}$, $1/(1+n^2)$, $1/(5+2n+n^2))$,
$(n \cdot \ln(n+1))^{-2}$, $1/\sqrt{n^3+n}$, $n^{-4/3}$, $1/((n+1)(\ln(n+1))^2)$;
$z = 1, 0.99, 0.90.7, e^{i\pi/16}, e^{i\pi/4}, i$;
for the appropriate algorithms, see Problem 2.
Try also Euler–Maclaurin's summation formula, or one of its variants, if you can handle the integral with good accuracy. Also try to find a good comparison series; it is not always possible.

Study the convergence rate. Try also *thinning* to the first two methods.

6. *Divergent series.* Apply Aitken acceleration, the generalized Euler Transformation and GCA to the following divergent series $\sum_1^{\infty} c_n z^n$. One of them is impossible. Can you tell in advance which? If possible, compare the numerical results with the results obtained by analytic continuation, using the analytic expression for the sum as a function of z. See also Lemma 3.3.8, and other results in § 3.3.6.
(a) $c_n = 1$, $z = -1$; (b) $c_n = n$, $z = -1$; (c) c_n is an arbitrary polynomial in n;

(d) $c_n = 1$, $z = i$; (e) $c_n = 1$, $z = 2$; (f) $c_n = 1$, $z = -2$.

7. Let y_n be the Fibonacci sequence defined, in Problem 3.2.17 by the recurrence relation,

$$y_n = y_{n-1} + y_{n-2}, \quad y_0 = 0, \quad y_1 = 1.$$

Show that the sequence $\{y_{n+1}/y_n\}_0^{\infty}$ satisfies the sufficient condition for Aitken acceleration, given in the text. Compute a few terms, compute the limit by Aitken acceleration(s), and compare with the exact result.

8. When the current through a galvanometer changes suddenly, its indicator begins to oscillate toward a new stationary value s. The relation between the successive turning points v_0, v_1, v_2, …is $v_n - s \approx A \cdot (-k)^n$, $0 < k < 1$. Determine from the following series of measurements, Aitken extrapolated values v_2', v_3', v_4' which are all approximations to s:

$$v_0 = 659, \quad v_1 = 236, \quad v_2 = 463, \quad v_3 = 340, \quad v_4 = 406.$$

9. *The a-version of Aitken acceleration* If you want the sum of slowly convergent *series*, it may seem strange to compute the sequence of partial sums, and the compute the first and second differences of rounded values of this sequence in order to apply Aitken acceleration.
The *a-version* of Aitken acceleration works on the terms a_j, $j = 1,2,3,\dots$ of an infinite series. The previous version may be called the *s-version*, since it works on the sequence of its partial sums s_j.
Given s_j, $j = 1 : N$, $s_1 = a_1$. Set $a_0 = 0$. Then $a_j = \nabla s_j$, $j = 1 : N$. The Aitken acceleration thus reads $s_j' = s_j - a_j^2/\nabla a_j$, $j = 1 : N$.

(a) We want to determine a_j' so that $\sum_{k=1}^{j} a_k' = s_j'$, $j = 1 : N$. Show that

$$a_1' = 0, \quad a_j' = a_j - \nabla(a_j^2/\nabla a_j), \; j = 2 : N, \quad s_N' = s_N - a_N^2/\nabla a_N.$$

Show that this can be *iterated*, for $i = 0 : N-2$,

$$a_{i+1}^{(i+1)} = 0, \quad a_j^{(i+1)} = a_j^{(i)} - \nabla\left((a_j^{(i)})^2/\nabla a_j^{(i)}\right), \; j = i+2 : N,$$
$$s_N^{(i+1)} = s_N^{(i)} - (a_N^{(i)})^2/\nabla a_N^{(i)}.$$

(Note that $a_j^{(0)} = a_j$, $s_j^{(0)} = s_j$.) We thus obtain N estimates of the sum s. We cannot be sure that the last estimate $s_N^{(N-1)}$ is the best, due to irregular errors in the terms and during the computations. Accept instead, e.g., the average of a few estimates that are close to each other, or do you have a better suggestion? This also gives you a (not quite reliable) error estimate.

(b) Although we may expect that the a-version handles rounding errors better than the s-version, the rounding errors may set a limit for the accuracy of the result. It is easy to combine *thinning* with this version. How?

(c) Study or write yourself a program for the a-version, and apply it on one or two problems, where you have used the s-version earlier. Also use thinning on a problem, where it is needed. We have here considered N as given. Can you suggest a better termination criterion, or a process for continuing the computation, if the accuracy obtained is disappointing?

(d) Design an a-version of the modified Aitken acceleration (3.3.5), or look up in [3].

10. A function $g(t)$ has the form

$$g(t) = c - kt + \sum_{n=1}^{\infty} a_n e^{-\lambda_n t},$$

where c, k, a_n and $0 < \lambda_1 < \lambda_2 < \ldots < \lambda_n$ are unknown constants and $g(t)$ is known numerically for $t_\nu = \nu h$, $\nu = 0, 1, 2, 3, 4$.

Find out how to eliminate c, in such a way that a sufficient condition for estimating kh by Aitken acceleration is satisfied. Apply this to the following data, where $h = 0.1$, $g_\nu = g(t_\nu)$.

$$g_0 = 2.14789, \quad g_1 = 1.82207, \quad g_2 = 1.59763, \quad g_3 = 1.40680, \quad g_4 = 1.22784.$$

Then, estimate also c.

11. The formula for Aitken acceleration is sometimes given in the forms

$$s_n - \frac{(\Delta s_n)^2}{\Delta^2 s_n} \quad \text{or} \quad s_n - \frac{\Delta s_n \nabla s_n}{\Delta s_n - \nabla s_n}.$$

Show that these are equivalent to s'_{n+2} or s'_{n+1}, respectively, in the notations of (3.3.1). Also note that the second formula is $\lim_{p\to\infty} s'_n$ (not s'_{n+1}) in the notation of (3.3.3).

13. Apply active Aitken acceleration, as described in Example 3.3.3, to the iteration formula $s_{n+1} = \phi(s_n)$, where $\phi(s) = 1 - 0.5s^2$, until either you have 10 correct decimals, or there are clear indications that the process is divergent.

(a) with $s_0 = 0.8$; (b) with $s_0 = -2.7$.

14. Suppose that the sequence $\{s_n\}$ satisfies the condition $s_n - s = c_0 n^{-p} + c_1 n^{-p-1} + O(n^{-p-2})$, $p > 0$, $n \to \infty$, and set

$$s'_n = s_n - \frac{p+1}{p} \frac{\Delta s_n \nabla s_n}{\Delta s_n - \nabla s_n},$$

It was stated without proof in Sec. 3.3.2 that $s'_n - s = O(n^{-p-2})$.

Since the difference expressions are symmetrical about n one can conjecture that this result would follow from a continuous analogue with derivatives instead of differences. It has been shown [3] that this conjecture is true, but we shall not prove that. Our (easier) problem is just the continuous analogue: suppose that a function $s(t)$ satisfies the condition $s(t) - s = c_0 t^{-p} + c_1 t^{-p-1} + O(t^{-p-2})$, $p > 0$, $t \to \infty$, and set

$$y(t) = s(t) - \frac{p+1}{p} \frac{s'(t)^2}{s''(t)}.$$

Show that $y(t) - s = O(t^{-p-2})$. Formulate and prove the continuous analogue to (3.3.6).

15. (a) Consider as in Example 3.3.6, $\sum n^{-3/2}$. Show that the partial sum s_n has an asymptotic expansion of the form needed in that example, with $p = -1/2$.

Hint: Apply Euler–Maclaurin's formula (theoretically).

(b) Suppose that $\sum a_n$ is convergent, and that $a_n = a(n)$. $a(z)$ is analytic function at $z = \infty$ (for example a rational function), multiplied by some power of $z - c$. Show that such a function has an expansion like (3.3.4), and that the same holds for a product of such functions.

16. *Rewriting a Fourier series for convergence acceleration.*

Consider a *real* function with the Fourier expansion $F(\phi) = \sum_{n=-\infty}^{\infty} c_n e^{in\phi}$.

(a) Set $z = e^{i\phi}$, and show that $F(\phi) = c_0 + 2\Re \sum_{n=1}^{\infty} c_n z^n$.

Hint: Show that $c_{-n} = \bar{c}_n$.

(b) Set $c_n = a_n - ib_n$, where a_n, b_n are real. Show that $\sum_{n=0}^{\infty}(a_n \cos n\phi + b_n \sin n\phi) = \Re \sum_{n=0}^{\infty} c_n z^n$.

(c) How would you rewrite the Chebyshev series $\sum_{n=0}^{\infty} T_n(x)/(1+n^2)$?

(d) Consider also how to handle a complex function $F(\phi)$.

17. Compute and plot $F(x) = \sum_0^{\infty} \frac{T_m(x)}{1+m^2}$, $x \in [-1, 1]$. Find out experimentally or theoretically how $F'(x)$ behaves near $x = 1$ and $x = -1$.

18. Compute to (say) 6 decimal places the double sum

$$S = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{(-1)^{m+n}}{(m^2 + n^2)}.$$

Set $f(m) = \sum_{n=1}^{\infty} (-1)^n (m^2 + n^2)^{-1}$. Then $S = \sum_{n=1}^{\infty} (-1)^m f(m)$. Compute, to begin with, $f(m)$ for $m = 1 : 10$, by the generalized Euler Transformation or CGA or what have you? Do you need more values of $f(m)$?

COMMENT: There exists an explicit formula for $f(m)$ in this case, but you can solve this problem easily without using that.

19. We use the notation of § 3.3.2 (the generalized Euler transformation). Assume that $N \geq k \geq 1$, and set $n = N - k + 1$. A sum is equal to zero, if the upper index is smaller than the lower index.

(a) Prove Eq.(3.3.17) that was given without proof in the text, i.e.,

$$M_{N,k-1} - M_{N-1,k-1} = z^n P^{k-2} u_{n+1}, \quad (k \geq 2).$$

Hint: By subscript transformations in the definition of $M_{N,k}$, prove that

$$M_{N,k-1} - M_{N-1,k-1} = u_{n+1} z^n + \frac{z^n}{1-z} \sum_{s=0}^{k-3} (zE - 1) P^s u_{n+1}.$$

Next, show that $zE - 1 = (1-z)(P-1)$, and use this to simplify the expression.

(b) Derive the formulas

$$M_{k-1,k} = \frac{1}{1-z} \sum_{s=0}^{k-2} P^s u_1; \qquad M_{N,k} = M_{k-1,k} + \sum_{j=0}^{n-1} z^j P^{k-1} u_{j+1}.$$

COMMENT: The first formula gives the partial sums of the classical Euler transformation. The second formula relates the k'th column to the partial sums of the power series with the coefficients $P^{k-1} u_{j+1}$.

20. (a) If $u_j = a^j$, $z = e^{i\phi}$, $\phi \in [0, \pi]$, for which real values of $a \in [0,1]$ does the series on the right of (3.3.10) converge faster than the series on the left?

(b) Find how the classical Euler transformation works if applied to the series $\sum z^n$, $|z| = 1$, $z \neq 1$.

Compare how it works on $\sum u_n z^n$, for $u_n = a^n$, $z = z_1$, and for $u_n = 1, z = az_1$.

Consider similar questions for other convergence acceleration methods, that are primarily invented for oscillating sequences.

21. Compute $\sum_{k=1}^{\infty} k^{1/2}/(k^2+1)$ with an error of less than 10^{-6}.
Hint: Sum the first ten terms directly and then expand the summand in negative powers of k and use Euler–Maclaurin's summation formula. Or try a central difference variant of Euler–Maclaurin's summation formula given in the next problem; then you do not have to compute derivatives.

22. Variations on the Euler–Maclaurin Theme

Set $x_i = a + ih$, also for non-integer subscripts, $x_n = b$.

(a) TWO VARIANTS WITH CENTRAL DIFFERENCES INSTEAD OF DERIVATIVES. These are interesting alternatives, if the derivatives needed in the Euler–Maclaurin Formula are hard to compute. Check a few of the coefficients on the right hand side of the formula

$$\sum_{j=1}^{\infty} \frac{B_{2j}(hD)^{2j-1}}{(2j)!} \sim \frac{\mu\delta}{12} - \frac{11\mu\delta^3}{720} + \frac{191\mu\delta^5}{60480} - \frac{2497\mu\delta^7}{3628800} + \frac{14797\mu\delta^9}{95800320} - \ldots. \tag{3.3.61}$$

Use the expansion for computing the sum given in the previous problem.
COMMENT: This formula is given by Fröberg [11], who attributes it to Gauss. Compare the size of its coefficients with the corresponding coefficients of the Euler–Maclaurin Formula.

Suppose that $h = 1$, and that the terms of the given series can be evaluated also for non-integer arguments. Then *another variant* is to compute the central differences for (say) $h = 1/2$ in order to approximate *each* derivative needed more accurately by means of Eqn. (3.2.44). This leads to the formula [89]

$$\sum_{j=1}^{\infty} \frac{B_{2j}D^{2j-1}}{(2j)!} \sim \frac{\mu\delta}{6} - \frac{7\mu\delta^3}{180} + \frac{71\mu\delta^5}{7560} - \frac{521\mu\delta^7}{226800} + \dots. \tag{3.3.62}$$

($h = 1/2$ for the central differences; $h = 1$ in the series.) Convince yourself of the reliability of the formula, either by deriving it or by testing it for (say) $f(x) = e^{0.1\,x}$.
Apply it also to some of the series suggested in Problem 5.
Show that the rounding errors of the function values cause almost no trouble in the numerical evaluation of these difference corrections.

(b) A VARIANT WITH UNCENTERED DIFFERENCES, KNOWN AS GREGORY'S QUADRATURE FORMULA. [90] Derive the following formula, in which the operator expansion must be truncated at $\nabla^k y_n$ and $\Delta^l y_0$, where $k \le n$, $l \le n$. Apply the formula to some integral of your own choice

$$\int_{x_0}^{x_0+nh} y\,dx = h\frac{E^n - 1}{hD} y_0 = h\left(\frac{y_n}{-\ln(1-\nabla)} - \frac{y_0}{\ln(1+\Delta)}\right)$$
$$= \hat{T}(x_0; h; x_n) + h\sum_{j=1}^{\infty} a_{j+1}(\nabla^j y_n + (-\Delta)^j y_0),$$

where $\hat{T}$ is the trapezoidal sum, as defined in the Euler–Maclaurin Formula. Explain why the coefficients a_{j+1}, $j \ge 1$, occur in the implicit Adams formula too, see Problem 3.2.9(a). Concerning the interpretation of ∇^{-1} and Δ^{-1}, see Problem 3.2.13(d).

Is it true that (the short version of) Simpson's formula is a particular case of Gregory's formula? (Simpson lived 1710-1761.)

(c) A MIDPOINT VARIANT. Let $\hat{R}(a,h,b)f$ be the midpoint sum $\hat{R}(a,h,b)f = \sum_{i=1}^{n} hf(x_{i-\frac{1}{2}})$. Recall that
$\hat{R}(a,h,b)f = 2\hat{T}(a : \frac{1}{2}h : b)f - \hat{T}(a : h : b)f$. Prove the following expansion that has the same relation to the midpoint sum as the Euler–Maclaurin Formula has to the trapezoidal sum.

[89] The formula is probably very old, but we have not found it in the literature.

[90] James Gregory (1638–1675), Scotch mathematician. This formula was discovered long before the Euler–Maclaurin formula, and seems to have been primarily used for numerical quadrature. It can be used also for summation, but the variants with central differences are typically more efficient.

$$\hat{R}(a,h,b)f = \int_a^b f(x)dx - \frac{h^2}{24}\big(f'(b) - f'(a)\big) + \frac{7h^4}{5760}\big(f'''(b) - f'''(a)\big)$$
$$+ \ldots + \Big(\frac{1}{2^{2r-1}} - 1\Big)\frac{B_{2r}h^{2r}}{(2r)!}\big(f^{(2r-1)}(b) - f^{(2r-1)}(a)\big) + \ldots$$

(d) A VARIANT FOR ALTERNATING SERIES. Derive formally in a similar way the following formula for an *alternating series*. Set x_i, $h = 1$, $b = \infty$, assume that $\lim_{x\to\infty} f(x) = 0$.

$$\sum_{i=0}^{\infty}(-1)^i f(a+i) = \tfrac{1}{2}f(a) - \frac{1}{4}f'(a) + \frac{1}{48}f'''(a) - \ldots - \frac{(2^{2r}-1)B_{2r}}{(2j)!}f^{(2r-1)}(a) - \ldots.$$

Of course, the integral of f is not needed in this case [91] . Compare it with some of the other methods for alternating series on an example of your own choice.

(e) A MORE GENERAL FORM OF THE EULER–MACLAURIN FORMULA. Derive, e.g. by operators (without the remainder R), the following formula, [1], Formula 23.1.32.

$$\sum_{k=0}^{m-1} hf(a+kh+\omega h) = \int_a^b f(t)dt + \sum_{j=1}^{p}(h^j/j!)B_j(\omega)(f^{(j-1)}(b) - f^{(j-1)}(a)) + R,$$

$$R = -(h^p/p!)\int_0^1 \hat{B}_p(\omega - t)\sum_{k=0}^{m-1} f^{(p)}(a + kh + th)\, dt.$$

If you use this formula for deriving the midpoint variant in (c), you will find a quite different expression for the coefficients; nevertheless it is the same formula. Tell how this is explained by Formula 23.1.10 in Handbook [1], i.e., by the "Multiplication Theorem" [92]

$$B_n(mx) = m^{n-1}\sum_{k=0}^{m-1} B_n(x + k/m), \qquad n = 0, 1, 2, \ldots, \quad m = 1, 2, 3, \ldots$$

COMMENT: See also Problem 38; a summation formula based on the Euler numbers.

23. Prove statement (b) of the Lemma 3.3.1. (concerning the periodicity and the regularity of the Bernoulli functions).

[91] Note that the right hand side yields a finite value if f is a constant or, more generally, if f is a polynomial, although the series on the left hand side diverges. The same happens to other summation methods; see comments in the last example of § 3.3.2.

[92] That formula and the remainder R are derived in Nörlund, [27], p. 21 and p. 30, respectively.

24. Euler's constant is defined by $\gamma = \lim_{N\to\infty} F(N)$, where

$$F(N) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N-1} + \frac{1}{2N} - \ln N.$$

(a) Use the Euler–Maclaurin formula with $f(x) = x^{-1}$, $h = 1$, to show that, for any integer M

$$\gamma = F(M) + \frac{1}{12}M^{-2} - \frac{6}{720}M^{-4} + \frac{120}{30240}M^{-6} - \ldots,$$

where every other partial sum is larger than γ, and every other is smaller.

(b) Compute γ to seven decimal places, using $M = 10$, $\sum_{n=1}^{10} n^{-1} = 2.92896825$, $\ln 10 = 2.30258509$.

(c) Show how repeated Richardson extrapolation can be used to compute γ from the following values:

M	1	2	4	8
$F(M)$	0.5	0.55685	0.57204	0.57592

(d) Extend (c) to a computation, where a larger number of values of $F(M)$ have been computed as accurately as possible with the normally used precision of your computer, and so that the final accuracy of γ is limited by the effects of rounding errors. Check the result by looking up in an accurate table of mathematical constants, e.g., in Handbook [1].

(e) Set

$$S(r) = \sum_{m=1}^{r}\sum_{n=1}^{r}(m^2+n^2)^{-1}.$$

By a continuous analog, with a double integral instead of a double sum, you may conjecture that $S(R) \sim a \ln R + b$ as $R \to \infty$. You may even suggest a value of the parameter a. Investigate the conjecture, by computing $S(R)$ for a suitable sequence of values of R. If you find support for it, try to estimate a and b.

25. *A digression about the Gamma function.*

(a) Handbook [1] gives an expansion for $\ln\Gamma(z)$ that agrees with our formula (3.3.30) for $\ln z!$ (if we substitute z for m), except that the handbook writes $(z-\frac{1}{2})\ln z$, where we have $(m+\frac{1}{2})\ln m$. Explain concisely and completely that there is no contradiction here.

(b) An asymptotic expansion for computing $\ln\Gamma(z+1)$, $x \in \mathbf{C}$ is derived in Example 3.3.16. If r terms are used in the asymptotic expansion, the remainder reads:

$$\frac{K(z)(2r)!}{\pi|2\pi z|^{2r+1}} \quad \text{where} \quad K(z) = \sup_{u\geq 0}\frac{|z^2|}{|u^2+z^2|}.$$

Set $z = x + iy$. Show the following more useful bound for $K(z)$, valid for $x > 0$,

$$K(z) \le \begin{cases} 1, & \text{if } x \ge |y|; \\ \frac{1}{2}(x/|y| + |y|/x), & \text{otherwise.} \end{cases}$$

Find a uniform upper bound for the remainder if $r = 5$, $x \ge \frac{1}{2}$, $|z| \ge 17$.

(c) Write a program, e.g., in Matlab, for the computation of $\ln \Gamma(z+1)$. Use the reflection and recurrence formulas to transform the input value z, to another $z = x + iy$ that satisfies $x \ge \frac{1}{2}$, $|z| \ge 17$, for which this asymptotic expansion is to be used with $r = 5$.

Test the program, e.g., by computing the following quantities, and compare with their exact values, e.g.,

$$n!, \quad \Gamma(n + 1/2)/\sqrt{\pi}, \quad n = 0, 1, 2, 3, 10, 20.$$

$$||\Gamma(\tfrac{1}{2} + iy)|^2 = \frac{\pi}{\cosh(\pi y)}, \quad y = \pm 10, \pm 20.$$

If the original input value has a small modulus, there is some cancellation, when when the output from the asymptotic expansion is transformed to $\ln(1 + z_{input})$, resulting in a loss of (say) 1 or 2 decimal digits.

(d) It is often much better to work with $\ln \Gamma(z)$ than with $\Gamma(z)$. For example, one can avoid exponent overflow in the calculation of a binomial coefficient or a value of the beta function, $B(z, w) = \Gamma(z)\Gamma(w)/\Gamma(z + w)$, where (say) the denominator can become too big, even if the final result is of a normal order of magnitude.

Another context where the logarithms are much preferable is in connection with interpolation, numerical differentiation etc.; for $|z| \gg 1$ $\ln \Gamma(z)$ is locally approximated by a polynomial much better than $\Gamma(z)$. The following is an example (for a handheld calculator).

Given $10! = 3628800$; compute $\Gamma(x)$ for $x = 11 : 15$. Compute $\Gamma'(13)$ by using either repeated Richardson extrapolation or the central difference expansion, in two ways:

- Use the values of $\ln \Gamma(x)$, (and multiply the logarithmic derivative by $\Gamma(13)$).
- Use directly the values of $\Gamma(x)$.

The first alternative requires a few more operations. Were they worthwhile?

26. (a) Show that

$$\binom{2n}{n} \sim \frac{2^{2n}}{\sqrt{\pi n}}, \quad n \to \infty,$$

and give an asymptotic estimate of the relative error of this approximation. Check the approximation as well as the error estimate for $n = 5$ and $n = 10$.

(b) *Random errors in a difference scheme.* We know from Example 3.2.2 that if the items y_j of a difference scheme are afflicted with errors less than ϵ in absolute value, then the inherited error of $\Delta^n y_j$ is at most $2^n \epsilon$ in absolute value.

If we consider the errors as independent random variables, uniformly distributed in the interval $[-\epsilon, \epsilon]$, show that the error of $\Delta^n y_j$ has the variance [93] $\binom{2n}{n}\frac{1}{3}\epsilon^2$, hence the standard deviation is approximately $2^n\epsilon(9\pi n)^{-1/4}$, if $n \gg 1$.

Check the result on a particular case by a Monte Carlo study.

Hint: It is known from Probability theory that the variance of $\sum_{j=0}^n a_j\epsilon_j$ is equal to $\sigma^2 \sum_{j=0}^n a_j^2$, and that a random variable, uniformly distributed in the interval $[-\epsilon, \epsilon]$, has the variance $\sigma^2 = \epsilon^2/3$. Finally use (3.1.15) with $p = q = n$.

27. (a) The following table of values of a function $f(x)$ is given:

x	0.6	0.8	0.9	1.0	1.1	1.2	1.4
$f(x)$	1.820365	1.501258	1.327313	1.143957	0.951849	0.752084	0.335920

Compute using repeated Richardson extrapolation $f'(1.0)$ and $f''(1.0)$.

(b) Use Romberg's method to compute the integral $\int_0^4 f(x)dx$, using the following (correctly rounded) values of $f(x)$. Need all the values be used?

x	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
$f(x)$	−4271	−2522	−499	1795	4358	7187	10279	13633	17247

28. (a) Compute $\int_1^\infty (1 + x^2)^{-1}\,dx$. In the notation of Example 3.3.21, compute $\int_1^2$, $\int_2^4$, $\int_4^8$, . . .; choose yourself where to stop. Use, e.g., Aitken acceleration to find $\int_1^\infty$. Compare with the exact result; and think of an error estimate that can be used if the exact result is not known.

(b)*Romberg+Aitken* Treat in the same way $\int_1^\infty \frac{1}{\sqrt{x+x^3}}$. Compare the computational effort for the computation of the tail $\int_R^\infty$ by acceleration and by series expansion with the same accuracy.

29. (a) Run Example 3.3.19 on a computer with machteps$\approx 10^{-16}$, until $k = 10$. (Subtract all results by the most accurate value of 2π that you can obtain with your software, so that you do not drown in a see of digits.)

(b) Do the same computation, but use this time $c_{2n} = 2n\sqrt{2 - \sqrt{4 - (c_n/n)^2}}$, and study the effect of the cancellations.

30. (a) Suppose that form of the error of Romberg's method is known, but the error constant r_k is not known. Determine r_k numerically for $K = 3$ and $k = 4$, by computing the Romberg scheme for $f(x) = x^{2k}$.

(b) Prove the formula for the error constant of Romberg's method.

31. *Numerov's method* [94] *with Richardson extrapolations.* Let $y_j = y(t_j)$, $y_j'' = y''(t_j)$. (a) Show that the formula

$$h^{-2}(y_{n+1} - 2y_n + y_{n-1}) = y_n'' + a(y_{n+1}'' - 2y_n'' + y_{n-1}'')$$

[93] Variance is the square of the standard deviation.

[94] See also Example 3.2.17.

is exact for polynomials of as high degree as possible, if $a = 1/12$. Show that the error has an expansion into *even* powers of h, and determine the first (typically non-vanishing) term of this expansion.

(b) This formula can be applied to the differential equation, $y'' = p(x)y$, with given initial values $y(0)$, $y'(0)$. Show that this yields the recurrence relation

$$y_{n+1} = \frac{(2 + \frac{10p_n h^2}{12})y_n - (1 - \frac{p_{n-1}h^2}{12})y_{n-1}}{1 - \frac{p_{n+1}h^2}{12}}.$$

This formula, which can be traced back at least to B. Numerov 1924, requires y_0, $y_1 \approx y(h)$ as seed. $y(h)$ can be obtained, for a small value of h, from a few terms of the Taylor expansion of the solution, the coefficients of which can be computed in the style of Example 3.1.2.

COMMENT: If h is small, information about $p(t)$ is lost by outshifting in the factors $1 - \frac{p_{n-1}h^2}{12}$ etc. We shall see in Sec. 13.6 how to rewrite the formulas in order to reduce the loss of information. In the application below this causes no trouble with the step sizes suggested, if macheps $= 2^{-53}$. If you must use macheps $= 2^{-24}$ (say), however, the outshifting may set a limit to the accuracy in the repeated Richardson extrapolation.

(c) We shall now apply this method, together with two Richardson extrapolations in (d), to the problem of Example 3.1.2, i.e., $y'' = -xy$ with initial values $y(0) = 1$, $y'(0) = 0$, this time over the interval $0 \le x \le 4.8$. Denote the numerical solution by $y(x; h)$, i.e., $y_n = y(x_n; h)$.
Compute the seeds $y_1 = y(h, h)$ by the Taylor expansion in Example 3.1.2. The error of $y(0.2, 0, 2$ should be less than 10^{-10}, since we expect that the (global) errors after two Richardson extrapolations can be of that order of magnitude.
Compute $y(x; h)$, $x = 0 : h : 4.8$, for $h = 0.05$, $h = 0.1$, $h = 0.2$. Store these data in a 100×3 matrix (where you must put zeros into some places). Plot $y(x; 0.05)$ versus x for $x = 0 : 0.05 : 4.8$.

(d) You proved in (a) that the *local* error has an expansion containing *even powers of h* only. It can be shown that *the same is true for the global error* too. Assume (without proof) that

$$y(x, h) = y(x) + c_1(x)h^4 + c_2(x)h^6 + c_3(x)h^8 + O(h^10).$$

Perform the adequate repeated Richardson extrapolations to your stored results.
Make semi-logarithmic plots of (the modulus of) the 4th order Richardson corrections for $x = 0 : 0.1 : 4.8$, obtained by means of $y(x; 0.05)$ and $y(x; 0.1)$. Plot in the same fashion the 6'th order corrections for $x = 0 : 0.2 : 4.8$, obtained in the second Richardson extrapolation. The 6th order corrections are used as error estimates for the results from both these Richardson extrapolations. [95]

[95]Although the 6th order correction yields an 8th order accurate result, it is hard obtain an error

(e) Express, e.g., by the aid of Handbook [1], Sec. 10.4, the solution of this initial value problem in terms of Airy functions, [96]

$$y(x) = \frac{\text{Ai}(-x) + \text{Bi}(-x)/\sqrt{3}}{2 \cdot 0.3550280539}.$$

Check a few of your results of the repeated Richardson extrapolation by means of Table 10.11 in the Handbook that, unfortunately, gives only 8 decimal places.
COMMENT: Your results should be more accurate than that. If they are not, the reason can be that the rounding errors have a large influence, but that is not the most probable reason in this case, if macheps $= 2^{-53}$. Experience shows that it is hard to avoid programming blunders in this problem. So do not consider the theory or the rounding errors as the primary suspects. Programming errors do not always yield results that are obviously crazy; sometimes the results look reasonable, although the accuracy is much lower than it should be.

(f) *Dense output.* How do you obtain results on the fine grid $x = 0 : 05 : 4.8$ with an accuracy that is comparable to the results on the coarse grid $x = 0 : 0.2 : 4.8$.
Hint: This question is discussed for another method in Example ex33.ode.

32. Compute the integral

$$\frac{1}{2\pi}\int_0^{2\pi} e^{\frac{1}{\sqrt{2}}\sin x}dx$$

by the trapezoidal rule, using $h = \pi/2$ and $h = \pi/4$ (for handheld calculator). Continue on a computer with smaller values of h, until the error is on the level of the rounding errors. Observe how the number of correct digits vary with h? Notice that Romberg is of no use in this problem.

33. (a) Show that the trapezoidal rule, with $h = 2\pi/(n+1)$, is exact for all trigonometric polynomials of period 2π and degree $\le n$—i.e., for all functions of the type

$$\sum_{k=-n}^{n} c_k e^{ikt}, \qquad i^2 = -1.$$

—when it is used for integration *over a whole period.*

(b) Show that if $f(t)$ can be approximated by a trigonometric polynomial of degree n so that the magnitude of the error is less than ϵ, in the interval $(0, 2\pi)$, then the error with the use of the trapezoidal rule with $h = 2\pi/(n+1)$ on the integral $(2\pi)^{-1}\int_0^{2\pi} f(t)\,dt$ is less than 2ϵ.

(c) Use the above to explain the sensationally good result in Problem 2 above,

estimate of that order without extra assumptions or extra computation. Recall the discussion at the end of § 3.3.5. G.D. also computed 8th order corrections, based on y(x;0.4). They were about one tenth of the 6th order corrections; G.D had expected smaller values.

[96] Airy functions are special functions (related to Bessel functions) with many applications to Mathematical Physics, e.g., the theory of diffraction of radio waves around the earth's surface.

when $h = \pi/4$. *Hint:* First estimate how well the function $g(x) = e^{x/\sqrt{2}}$ can be approximated by a polynomial in $\mathcal{P}_8$ for $x \in [-1, 1]$. The estimate found by the truncated Maclaurin expansion is not quite good enough. Theorem 3.1.5 provides a sharper estimate with an appropriate choice of R; remember Scylla and Charybdis.

34. Compute by the Euler–Maclaurin formula, or rather the trapezoidal rule,

$$\text{(a)} \int_0^\infty e^{-x^2/2}dx, \qquad \text{(b)} \int_0^\infty \frac{dx}{\cosh(\pi x)},$$

as accurately as you can with the normal precision of your computer (or software). Then find out empirically how the error depends on h. Make semi-logarithmic plots on the same screen. How long range of integration do you need?

35. a) Determine the Bernoulli polynomials $B_2(x)$ and $B_3(x)$, and find the values and the derivatives at 0 and 1. Factorize the polynomial $B_3(x)$. Draw the graphs of a few periods of $\hat{B}_i(x),\ i = 1, 2, 3..$

b) In an old Cours d'Analyse, we found a "symbolic" formula, essentially

$$h\sum_{j=0}^{n-1} g'(a + jh) = g(b + hB) - g(a + hB). \tag{3.3.63}$$

After the expansion of the right hand side into powers of hB, has been followed by the replacement of the powers of B by Bernoulli numbers, the resulting expansion is not necessarily convergent, even if the first power series converges for any complex value of hB.
Show that the second expansion is equivalent to the Euler–Maclaurin formula, and that it is to be interpreted according to Theorem 3.3.3.
If g is a polynomial, the expansion is finite. Show the following important formulas, and check them with known results for $k = 1 : 3$.

$$\sum_{j=0}^{n-1} j^{k-1} = \frac{(B + n)^k - B^k}{k} = \frac{B_k(n) - B_k}{k}. \tag{3.3.64}$$

Also find that (3.3.63) makes sense for $g(x) = e^{\alpha x}$, with the "symbolic" interpretation of the power series for e^{Bx}, if you accept the formula $e^{(B+\alpha)x} = e^{Bx}e^{\alpha x}$.

36. A **bell sum** is a series $\sum a_n$, where a_n as a function of n has a bell-shaped graph, and where you must add many terms to get the desired accuracy. Under certain conditions you can get an accurate result by adding (say) every tenth term, and multiply this sum by 10, because both sums can be interpreted as trapezoidal approximations to the same integral, with different step size. Inspired by Euler–Maclaurin's formula or, even better, by Theorem 3.3.4 we may hope to be able to obtain high accuracy using an integer stepsize h that is (say) one quarter of the half-width of "the bell". In other words, we do not have to compute and add more than every h'th term.

We shall study a class of series

$$S(t) = \sum_{n=0}^{\infty} c_n t^n/n!, \quad t \gg 1, \tag{3.3.65}$$

where $c_n > 0$, $\log c_n$ is rather slowly varying for n large; (say that) $\Delta^p \log c_n = O(n^{-p})$. Let $c(\cdot)$ be a smooth function such that $c(n) = c_n$. We consider $S(t)$ as an approximation to the integral $\int_0^\infty c(n)t^n/\Gamma(n+1)dn$, with a smooth and bellshaped integrand, almost like the normal frequency function, with standard deviation $\sigma \approx k\sqrt{t}$..

(a) For $p = 1:5$, $t = 4^p$, plot $y = \sqrt{2\pi t}e^{-t}t^n/n!$ versus $x = n/t$, $0 \le x \le 3$; all 5 curves on the same picture.

(b) For $p = 1:5$, $t = 4^p$, plot $y = \ln(e^{-t}t^n/n!)$ versus $x = (n-t)/\sqrt{t}$, $\max(0, t - 8\sqrt{t}) \le n \le t + 8\sqrt{t}$; all 5 curves on the same picture. Give bounds for the error committed if you neglect the terms of the series $e^{-t}\sum_0^\infty t^n/n!$, which are cut out in your picture.

(c) With the same notation as in (b), show theoretically that

$$\frac{e^{-t}t^n}{n!} = \frac{e^{-x^2/2}\big(1 + O(1/\sqrt{t})\big)}{\sqrt{2\pi t}}, \tag{3.3.66}$$

for $t \to \infty$, where the $O(1/\sqrt{t})$-term depends on x. Compare this with the plots. *Hint*: Use Stirling's asymptotic expansion.

COMMENT: If you are familiar with Probability, you recognize that this is related to the normal approximation to the Poisson distribution. It is well known that the mean is t, and the standard deviation is $\sqrt{t}$.

If you are familiar with Mathematical Physics, you see the resemblance to the *saddle point method*, if you interpret the sum of terms like the left hand side (from $n = 0$ to ∞) as an approximation to an integral with stepsize $\Delta n = 1$, i.e., $e^{-t}\int_0^\infty t^n/\Gamma(n+1)dn \sim \int_{-\infty}^\infty \exp(-x^2/2)/\sqrt{2\pi}dx = 1$, as $t \to \infty$. (Note that $dx = dn/\sqrt{t}$.)

A crude approximation for (3.3.65) is $S(t) \approx c(t)e^t$.

We aim, however, at higher accuracy than is common when these approximations are used in Probability and Mathematical Physics, but *who can be interested in high accuracy*?

We think for example, of a situation, where the result is to enter a calculation of cancellation type, such that (say) the first ten digits will be lost, and we need a decent relative accuracy in what will be left. It has been emphasized on several pages of this book that such a situation should be avoided. Well, but there must exist alternatives for the cases, where one has not been able to avoid it, and there is no time to wait for a better theory. Then, high accuracy is needed in intermediate results.[97]

[97]G.D. has recently encountered just this situation in a problem of financial mathematics (!).

(d) Test these ideas by making numerical experiments with the series

$$e^{-t}\sum_{n\in\mathcal{N}} t^n/n!,$$

where $\mathcal{N} = \{\text{round}(t - 8\sqrt{t}) : h : \text{round}(t + 8\sqrt{t})\}$, for some integers h in the neighborhood of suitable fractions of $\sqrt{t}$, inspired by the outcome of the experiments. Do this for t =1000, 500, 200, 100, 50, 30. Compare with the exact result, and see how the trapezoidal error depends on h, and try to formulate an error estimate that can be reasonably reliable, in cases where the answer is not known. Does the behavior of the error resemble what you would have, according to Theorem 3.3.4, for the normal frequency function (3.3.66)? How large must t be, in order that it should be permissible to choose $h > 1$ if you want (say) 6 correct decimals?

(e) Compute, with an error estimate, $e^{-t}\sum_{n=1}^{\infty} t^n/(n\cdot n!)$, with 6 correct decimals for the values of t mentioned in (d). You can also check your result with tables and formulas in Handbook, [1, Ch. 5]. See also Handbook, Introduction, Sec.3.

(f)*An alternative approach to the bell sum technique.* Let h be a natural number, and assume that $x \gg 1$, $\text{T}\small{\text{OL}} \ll 1$. Show that the power series for $\sum_{m=0}^{h-1} F(x\exp(2\pi i m/h))$ equals the sum that consists of every h'th term from the power series for $F(x)$, multiplied by h. *Hint*: Recall Lemma 3.1.9.

Consider, e.g., the case $F(z) = e^z$. Find a simple sufficient condition on x, h for the inequality $|\sum_{m=1}^{h-1} F(x\exp(2\pi i m/h))| < \text{TOL}F(x)$ to be approximately true. Compare it with a condition, which can be conjectured by means of (3.3.66) and Theorem 3.3.4. Hint: $|e^{z-x}| = e^{\Re(z-x)}$.

37. If you have a good program for generating primes, denote the n'th prime by p_n, and try convergence acceleration to series like

$$\sum\frac{(-1)^n}{p_n},\quad \sum\frac{1}{p_n^2},$$

or what have you? Due to the irregularity of the sequence of primes, you cannot expect the spectacular accuracy of the previous examples, but it can be fun to see how these methods work, e.g., in combination with some comparison series derived from asymptotic results about primes. The simplest one reads $p_n \sim n\ln n$, $(n \to \infty)$. [98]

38. A SUMMATION FORMULA BASED ON THE EULER NUMBERS

(a) The Euler numbers E_n were introduced by Equation (3.1.14). The first values read $E_0 = 1$, $E_2 = -1$, $E_4 = 5$, $E_6 = -61$. They are all integers (Problem 3.1.7e). $E_n = 0$ for odd n, and the sign is alternating for even n. Their generating function reads

$$\frac{1}{\cosh z} = \sum_{j=0}^{\infty}\frac{E_j z^j}{j!}.$$

[98]This is equivalent to the classical prime number theorem.

(a) Show, e.g., by means of operators the following expansion

$$\sum_{k=m}^{\infty}(-1)^{k-m}f(k) \approx \sum_{p=0}^{q}\frac{E_{2p}f^{(2p)}(m-\frac{1}{2})}{2^{2p+1}(2p)!} \tag{3.3.67}$$

COMMENT: No discussion of convergence etc. is needed; the expansion behaves much like the Euler–Maclaurin expansion, and so does the error estimation, see, e.g., [9].
The coefficient of $f^{(2p)}(m-\frac{1}{2})$ is approximately $2(-1)^p/\pi^{2p+1}$ when $p \gg 1$, e.g., for $p = 3$ the approximation yields $-6.622\,10^{-4}$, while the exact coefficient is $61/92160 \approx 6.619\,10^{-4}$.

(b) Apply (3.3.67) for explaining the following curious observation, reported by Borwein et al. [4].

$$\sum_{k=1}^{50}\frac{4(-1)^k}{2k-1} = 3.12159465259\ldots$$
$$(\pi = 3.14159265359\ldots).$$

Note that only three digits disagree. There are several variations on this theme. Borwein et al. actually displayed the case with 40 decimal places based on 50,000 terms. Make "an educated guess" concerning how few digits disagreed.

39. Find out, by Stirling's formula, the accuracy that can be obtained by the Euler–Maclaurin expansion with a terms of the original series of Example 3.3.3.

40. What is $\beta(t)$ (in the notation of $\big(3.3.47)\big)$, if $u_n = a^n$, $0 < a < 1$?

41. Work out the details of the two optimizations in the proof of Theorem 3.3.9.

42. Show that every rational function $f(s)$ that is analytic and bounded for $\Re s \geq a$ is d.c.m for $s \geq a$.

43. Show criterion (B) for higher monotonicity (concerning products).

44. Which of the coefficient sequences $\{c_n\}$ mentioned in Problems 4 and 5 are c.m.? Which are d.c.m.?

45. Show criterion (E) for higher monotonicity ()

46. Suppose that $u_n = \int_0^1 t^n d\beta(t)$, where $\beta(t)$ is of bounded variation in $[0,1]$. Show that $\lim u_n = 0$ if $\beta(t)$ is continuous at $t = 1$, but it is not true if $\beta(t)$ has a jump at $t = 1$.

47. Show that if $\{u_n\}_0^\infty$ is c.m. then $\{u_n\}_1^\infty$ is a minimal c.m.

48. Prove the remainder formula for GCA, (3.3.58).

49. Prove the recursion formula (3.3.60) for GCA.

50. Make an experimental stability analysis for GCA, in the style of Example 3.2.1 for difference schemes.

51. Make a study of GCA, analogous to Example3.3.9, Application to Fourier series. Compare with the generalized Euler transformation, in particular the effect of thinning.

3.4 Continued Fractions and Padé Approximants

3.4.1 Continued Fractions

Some functions cannot be well approximated by a power series, but can well be approximated by a quotient of power series. In order to study such approximations we first introduce **continued fractions**. Let

$$x = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 +}}} \ldots = b_0 + \frac{a_1}{b_1+}\,\frac{a_2}{b_2+}\,\frac{a_3}{b_3+}\cdots, \tag{3.4.1}$$

where the second expression is a convenient compact notation. If the number of terms is infinite, x is called an *infinite continued fraction*, and the terminating fraction

$$x_n = \frac{p_n}{q_n} = b_0 + \frac{a_1}{b_1+}\,\frac{a_2}{b_2+}\cdots\frac{a_n}{b_n} \tag{3.4.2}$$

is called *the nth convergent* (or approximant) of the continued fraction. A terminating fraction can be evaluated *backwards* by the recurrence relations,

$$y_1 = b_{n-1} + a_n/b_n, \quad y_2 = b_{n-2} + a_{n-1}/y_1, \ldots \quad x_n = y_n = b_0 + a_1/y_{n-1}. \tag{3.4.3}$$

It can happen that $y_i = \infty$ for some i. That does no harm. In next step you divide by y_i; set the result of this division equal to 0.

The following theorem shows how a continued fraction can be evaluated *forwards*, and tells two other important facts about continued fractions. If it happens in the last step, the result is ∞.

Theorem 3.4.1.

Consider the continued fraction (3.4.1). For $n \geq 1$, $x_n = p_n/q_n$, where, p_n, q_n satisfies the **recursion formula**

$$p_n = b_n p_{n-1} + a_n p_{n-2}, \quad p_{-1} = 1, \quad p_0 = b_0,$$
$$q_n = b_n q_{n-1} + a_n q_{n-2}, \quad q_{-1} = 0, \quad q_0 = 1.$$

Two other useful formulas read

$$p_n q_{n-1} - p_{n-1} q_n = (-1)^{n-1} a_1 a_2 \cdots a_n, \tag{3.4.4}$$

$$\frac{a_1}{b_1+}\,\frac{a_2}{b_2+}\,\frac{a_3}{b_3+}\cdots = \frac{k_1 a_1}{k_1 b_1+}\,\frac{k_1 k_2 a_2}{k_2 b_2+}\,\frac{k_2 k_3 a_3}{k_3 b_3+}\cdots \tag{3.4.5}$$

The last of the formulas is known as an **equivalence transformation**. *where the k_i are any non-zero numbers.*

Proof. We prove the first pair of formulas by induction. First, for $n = 1$, we obtain

$$\frac{p_1}{q_1} = \frac{b_1 p_0 + a_1 p_{-1}}{b_1 q_0 + a_1 q_{-1}} = \frac{b_1 b_0 + a_1}{b_1 + 0} = b_0 + \frac{a_1}{b_1} = x_1.$$

Next, assume that the formulas are valid up to p_{n-1}, q_{n-1}, for *every* continued fraction. Note that p_n/q_n can be obtained from p_{n-1}/q_{n-1}, by the substitution of $b_{n-1} + a_n/b_n$ for b_{n-1}. Hence

$$\frac{p_n}{q_n} = \frac{(b_{n-1} + a_n/b_n)p_{n-2} + a_{n-1}p_{n-3}}{(b_{n-1} + a_n/b_n)q_{n-2} + a_{n-1}q_{n-3}} = \frac{b_n(b_{n-1}p_{n-2} + a_{n-1}p_{n-3}) + a_np_{n-2}}{b_n(b_{n-1}q_{n-2} + a_{n-1}q_{n-3}) + a_nq_{n-2}}$$
$$= \frac{b_np_{n-1} + a_np_{n-2}}{b_nq_{n-1} + a_nq_{n-2}}.$$

This shows that the formulas are valid also for p_n, q_n.

The proofs of equations (3.4.4) and (3.4.5) are left for Problems 2 and 5, respectively. □

Comments. It is sometimes convenient to write the recursion formulas in matrix form, see Problem 2.

It is often useful to speed up the convergence of the sequence $\{p_n/q_n\}$, e.g., by means of Aitken acceleration. On the other hand, the transformation of a slowly convergent or a divergent power series to a continued fraction, see § 3.4.4, often yields an efficient convergence acceleration.

There is a risk of *overflow or underflow* with these formulas. We are usually not interested in the p_n, q_n themselves, but in the ratios only. Then we can normalize p_n and q_n by multiplying them by the same factor after they have been computed. If we shall go on and compute p_{n+1}, q_{n+1}, however, *we have to multiply* p_{n-1}, q_{n-1} *by the same factor also*!

One must also be careful about the numerical stability of these recurrence relations, see a discussion in Sec. 3.2.4.

Example 3.4.1. *Best Rational Approximations to a Real Number.*

Every positive number x can be expanded into a continued fraction of the form,

$$x = b_0 + \frac{1}{b_1+}\,\frac{1}{b_2+}\,\frac{1}{b_3+}\cdots. \tag{3.4.6}$$

Set $x_0 = x$, $p_{-1} = 1$, $q_{-1} = 0$. For $n = 0, 1, 2, \ldots$ we construct a sequence of numbers,

$$x_n = b_n + \frac{1}{b_{n+1}+}\,\frac{1}{b_{n+2}+}\,\frac{1}{b_{n+3}+}\cdots.$$

Evidently [99] $b_n = \lfloor x_n \rfloor$, $x_{n+1} = \frac{1}{x_n - b_n}$. Compute p_n, q_n, according to the recursion formulas of Theorem 3.4.1, which can be written in vector form,

$$(p_n, q_n) = (p_{n-2}, q_{n-2}) + b_n(p_{n-1}, q_{n-1}),$$

(since $a_n = 1$). See Fig. 4.3.1. Stop when $|x - p_n/q_n| < Tol$ or $n > nmax$. The details are left for Problem 1.

This algorithm has been used several times in the previous sections, e.g., in Example 3.2.12, where some coefficients, known to be rational, had been computed by the Cauchy+FFT method in floating point.

[99] $\lfloor x_n \rfloor$ denotes the integer part of x_n.

Figure 3.4.1. *Illustration to Example 3.4.1. The dashed line is* $\{(p,q) : xq = p\}$ *for* $x = \frac{1}{2}(\sqrt{5}+1)$.

The German mathematician Felix Klein [22] gave the following illuminating description of the sequence $\{(p_n, q_n)\}$ obtained by this algorithm (adapted to our notation):

> "Imagine pegs or needles affixed at all the integral points (p_n, q_n), and wrap a tightly drawn string about the sets of pegs to the right and to the left of the ray, $p = xq$. Then the vertices of the two convex string-polygons which bound our two point sets will be precisely the points $(p_n, q_n)\ldots$, the left polygon having the even convergents, the right one the odd."

Klein also points out that "such a ray makes a cut in the set of integral points" and thus makes Dedekind's definition of irrational numbers very concrete. This construction, see Fig. 3.4.1, illustrates in a concrete way that the successive convergents are closer to x than any numbers with smaller denominators, and that the errors alternate in sign. We omit the details of the proof that this description is correct.

Note that, since $a_j = 1$, $\forall j$, equation (3.4.4) reads $p_n q_{n-1} - p_{n-1} q_n = (-1)^{n-1}$. This implies that the triangle with vertices at the points $(0,0)$, (q_n, p_n), (q_{n-1}, p_{n-1}) has the smallest possible area, among triangles with integer coordinates, and hence there can be no integer points inside or on the sides of this triangle.

Fig. 3.4.1 corresponds to the example, (see also Problem 3),

$$x = 1 + \frac{1}{1+}\,\frac{1}{1+}\,\frac{1}{1+}\cdots \tag{3.4.7}$$

Assume that the continued fraction is convergent. [100] Then, note that $x = 1 + 1/x$, $x > 0$, hence $x = \frac{\sqrt{5}+1}{2}$.

Note also that, by (3.4.4) with $a_j = 1$,

$$\left|x - \frac{p_n}{q_n}\right| \le \left|\frac{p_{n+1}}{q_{n+1}} - \frac{p_n}{q_n}\right| = \frac{|p_{n+1}q_n - p_n q_{n+1}|}{q_{n+1}q_n} = \frac{1}{q_{n+1}q_n} < \frac{1}{q_n^2}. \tag{3.4.8}$$

[100] The convergence follows from a theorem of Seidel mentioned below.

COMMENT: If we know or guess that a result x of a computation is a rational number with a reasonably sized denominator, although it was practical to compute it in floating point arithmetic (afflicted by errors of various types), we have a good chance to reconstruct the exact result by applying the above algorithm as a post-processing.

If we just know that the exact x is rational, without any bounds for the number of digits in the denominator and numerator, we must be conservative in claiming that the last fraction that came out of the above algorithm is the exact value of x, even if $|x - p_n/q_n|$ is very small.

In fact, the fraction may depend on TOL that is to be chosen with respect to the expected order of magnitude of the error of x. If TOL has been chosen smaller than the error of x, it may, e.g., happen that the last fraction obtained at the termination is wrong, while the correct fraction (with smaller numerator and denominator) may have appeared earlier in the sequence (or it may not be there at all).

So a certain judgment is needed at the application of this algorithm. The smaller the denominator and numerator are, the more likely it is that the fraction is correct. In a serious context, it is advisable to check the result(s) by using exact arithmetic. If x is the root of an equation (or a component of the solution of a system of equations), it is typically much easier to check afterwards that a suggested result is correct than to perform the whole solution process in exact arithmetic.

More information about arithmetic continued fractions, from a computational point of view is found in Riesel [30]. Continued fractions have also important applications in Anaysis; some of the best algorithms for the numerical computation of important analytic functions are based on continued fractions. We shall not give complete proofs but refer to classical books of Perron [29], Wall [36] and Henrici [20, 21]. Codes and further references are given in Numerical Recipes, Press et al., [28, Sec. 6.2].

A **Theorem of Seidel**, see Cheney [7, p. 184], tells that *a positive continued fraction of the form of* (3.4.6) *converges if and only if the series* $\sum b_n$ *diverges.* (This makes sense also for a finite fraction, if we set $b_{n+1} = \infty$.)

It is typically easy to construct *i.e., for a power series, such that* suppose that $c_j \neq 0,\ \forall j \geq 1$.

A continued fraction is said to be equivalent to a given series, iff the sequence of convergents is equal to the sequence of partial sums. *There is typically an infinite number of such equivalent fractions. The construction of the continued fraction is particularly simple if we require that the denominators* $q_n = 1,\ \forall n \geq 1$. *For a power series we shall thus have* $p_n = c_0 + c_1x + c_2x^2 + \dots c_nx^n,\ n \geq 1$*: We must* assume that $c_j \neq 0,\ \forall j \geq 1$.

We shall determine the the elements a_n, b_n by means of the recursion formulas of Theorem 3.4.1 (for $n \geq 2$) with initial conditions. We thus obtain the following equations,

$$\begin{aligned} p_n &= b_np_{n-1} + a_np_{n-2}; \quad p_0 = b_0, \quad p_1 = b_0b_1 + a_1, \\ 1 &= b_n + a_n; \qquad b_1 = 1. \end{aligned}$$

The solution reads $b_0 = p_0 = c_0$, $b_1 = 1$, $a_1 = p_1 - p_0 = c_1 x$, and for $n \geq 2$,

$$\begin{aligned} a_n &= (p_n - p_{n-1})/(p_{n-2} - p_{n-1}) = -xc_n/c_{n-1}; \\ b_n &= 1 - a_n = 1 + xC_n/C_{n-1}; \end{aligned}$$

$$c_0 + c_1 x + \ldots + c_n x^n \ldots = c_0 + \frac{xc_1}{1-}\,\frac{xc_2/c_1}{1 + xc_2/c_1 -} \cdots \frac{xc_n/c_{n-1}}{1 + xc_n/c_{n-1} -} \cdots$$

Of course, an equivalent continued fraction gives by itself *no convergence acceleration, just because it is equivalent.* We shall therefore leave the subject of continued fractions equivalent to a series, after showing two instances of the numerous pretty formulas that can be obtained by this construction.

For

$$f(x) = e^x = 1 + x + x^2/2! + x^3/3! + \ldots$$

and

$$f(x) = \frac{\arctan\sqrt{x}}{\sqrt{x}} = 1 - x/3 + x^2/5 - x^3/7 + \ldots,$$

we obtain for $x = -1$ and $x = 1$, respectively, after simple equivalence transformations,

$$e^{-1} = 1 - \frac{1}{1+}\,\frac{1}{1+y} = \frac{1}{2+y} \Rightarrow e = 2 + y, \quad \text{where} \quad y = \frac{2}{2+}\,\frac{3}{3+}\,\frac{4}{4+}\,\frac{5}{5+}\ldots;$$

$$\frac{\pi}{4} = \frac{1}{1+}\,\frac{1}{2+}\,\frac{9}{2+}\,\frac{25}{2+}\,\frac{49}{2+}\ldots.$$

The latter formula *needs convergence acceleration* as much as the equivalent series $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$ that was used in § 3.3.3 to demonstrate the efficiency of Euler's Transformation (repeated averaging).

There exist, however, other methods to make a correspondence between a power series and a continued fraction. Some of them lead to a considerable convergence acceleration that often makes continued fractions very efficient for the *numerical computation of functions.* We shall return to such methods in § 3.4.4, and shall here only show examples of continued fractions for the efficient computation of some important functions.

$$\begin{aligned} \frac{1}{2}\ln\left(\frac{1+z}{1-z}\right) &= \frac{z}{1-}\,\frac{z^2}{3-}\,\frac{4z^2}{5-}\,\frac{9z^2}{7-}\,\frac{16z^2}{9-}\ldots \\ \arctan z &= \frac{z}{1+}\,\frac{z^2}{3+}\,\frac{4z^2}{5+}\,\frac{9z^2}{7+}\,\frac{16z^2}{9+}\ldots \qquad (3.4.9) \\ \tan z &= \frac{z}{1-}\,\frac{z^2}{3-}\,\frac{z^2}{5-}\,\frac{z^2}{7-}\ldots. \end{aligned}$$

These expansions can be used also for complex values of z. In fact the fraction for the logarithm can be used in the whole complex plane except in the intervals $(-\infty, -1]$ and $[1, \infty)$. For $\arctan z$, there are similar branch cuts on the imaginary axis. The convergence is slow, when z is near a cut. For an elementary function like these, a program can use some properties of the functions for moving z to a domain, where the continued fraction converges rapidly.

The expansion for $\tan z$ is valid everywhere, except in the poles. In all these cases the region of convergence as well as the speed of convergence is considerably larger than for the power series expansions. For example, the 6'th convergent for $\tan \pi/4$ is almost correct to 11 decimal places.

These continued fractions and several others are found in Abramowitz and Stegun [1]. Codes and some theoretical background are given in Numerical Recipes, Press et al., [28], Chapters 5 and 6. The following example contains a different type of continued fraction.

Example 3.4.2. *Continued fractions for the incomplete gamma function.*

A collection of formulas concerning this important function is found in Abramowitz and Stegun [1, Sec. 6.5]. For the sake of simplicity we assume that $x > 0$, although the formulas can be used also in an appropriately cut complex plane. The parameter a may be complex in $\Gamma(a,x)$. [101]

$$\begin{aligned}
\Gamma(a,x) &= \int_x^\infty e^{-t}t^{a-1}\,dt, \qquad \Gamma(a,0) = \Gamma(a),\\
\gamma(a,x) &= \Gamma(a) - \Gamma(a,x) = \int_0^x e^{-t}t^{a-1}dt, \quad \Re a > 0,\\
\Gamma(a,x) &= e^{-x}x^a\left(\frac{1}{x+}\,\frac{1-a}{1+}\,\frac{1}{x+}\,\frac{2-a}{1+}\,\frac{2}{x+}\cdots\right), \qquad (3.4.10)\\
\gamma(a,x) &= e^{-x}x^a\Gamma(a)\sum_{n=0}^\infty \frac{x^n}{\Gamma(a+1+n)}.
\end{aligned}$$

We mention these functions, because they have many applications. Several other important functions can, by simple transformations, be brought to particular cases of this function, e.g., the normal probability function, the chi-square probability function, the exponential integral, the Poisson distribution. Codes and some theoretical background are given in Numerical Recipes, Sec. 6.2. The continued fraction is used for x greater than about $a+1$. For x less than about $a+1$ the power series for $\gamma(a,x)$ is used.

By the following division algorithm, a rational function can be expressed as a continued fraction that can be evaluated by relatively few arithmetic operations, see Cheney [1966], p. 151. Let R_0, R_1 be polynomials, and set $R = R_0/R_1$. The degree of a polynomial R_j is denoted by d_j. By successive divisions (of R_{j-1} by R_j) we obtain quotients Q_j and remainders R_{j+1} as follows. For $j = 1, 2, \ldots$, until $d_{j+1} = 0$,

$$R_{j-1} = R_jQ_j + R_{j+1}, \quad d_{j+1} < d_j, \qquad (3.4.11)$$

hence

$$R = \frac{R_0}{R_1} = Q_1 + \frac{1}{R_1/R_2} = \ldots = Q_1 + \frac{1}{Q_2+}\,\frac{1}{Q_3+}\cdots\frac{1}{Q_k}. \qquad (3.4.12)$$

[101] There are plenty of other notations for this function.

By means of (3.4.5), this fraction can be transformed into a slightly more economic form, where the polynomials in the denominators have leading coefficient unity, while the numerators are in general different from 1.

3.4.2 The Padé Table.

Let $f(z)$ be a function defined by a power series,

$$f(z) = \sum_{l=0}^{\infty} c_l z^l. \tag{3.4.13}$$

The (m, n) Padé approximant associated with $f(z)$ is, *if it exists*, defined to be a rational function

$$f_{m,n}(z) = \frac{P_{m,n}(z)}{Q_{m,n}(z)} \equiv \frac{\sum_{i=0}^{m} p_i z^i}{\sum_{j=0}^{n} q_j z^j}, \qquad q_0 = 1, \tag{3.4.14}$$

that satisfies

$$f_{m,n}(z) = f(z) + Rz^{m+n+1} + O(z^{m+n+2}), \quad z \to 0. \tag{3.4.15}$$

The Padé table [102] is a table of the expressions for the functions $f_{m,n}(z)$, $m, n = 0, 1, 2, \ldots$. For example, the following is a part of the Padé table for $f(z) = e^z$.

$\frac{1}{1}$	$\frac{1+z}{1}$	$\frac{1+z+\frac{1}{2}z^2}{1}$
$\frac{1}{1-z}$	$\frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}$	$\frac{1+\frac{2}{3}z+\frac{1}{6}z^2}{1-\frac{1}{3}z}$
$\frac{1}{1-z+\frac{1}{2}z^2}$	$\frac{1+\frac{1}{3}z}{1-\frac{2}{3}z+\frac{1}{6}z^2}$	$\frac{1+\frac{1}{2}z+\frac{1}{12}z^2}{1-\frac{1}{2}z+\frac{1}{12}z^2}$

Theorem 3.4.2.
Set $c_l = 0$ for $l < 0$, $c_0 \neq 0$. The coefficients of the denominator of the Padé approximant $f_{m,n}(z)$ are determined by the solution of the linear system,

$$\sum_{j=1}^{n} c_{i-j} q_j + c_i = 0, \qquad i = m+1 : m+n, \tag{3.4.16}$$

if *this system has a unique solution.*
The coefficients of the numerator read $p_i = \sum_{j=0}^{k} c_{i-j} q_j$, $i = 0 : m$, $k = \min(i, n)$.
The error constant R in (3.4.15) *reads $R = -\sum_{j=0}^{k} c_{i-j} q_j$, $i = m+n+1$.*

Proof. Insert (3.4.13) and (3.4.15) into (3.4.14) and multiply both sides by the denominator:

$$\Big(\sum_{l=0}^{\infty} c_l z^l + Rz^{m+n+1} + O(z^{m+n+2})\Big) \sum_{j=0}^{n} q_j z^j = \sum_{i=0}^{m} p_i z^i.$$

[102]Padé (1863-1953), French mathematician. Note that the items of the table are not ordered in the same way as in matrices. For $f(z) = e^z$, there exist general expressions for the coefficients, see e.g., Gautschi [13].

Match the coefficients of z^i, $\quad i = 0, 1, 2, \ldots, m+n+1$, and remember that $q_0 = 1$:

$$\sum_{j=0}^{n} c_{i-j} q_j = \begin{cases} p_i, & \text{if } 0 \le i \le m; \\ 0, & \text{if } m+1 \le i \le m+n; \\ -R, & \text{if } i = m+n+1. \end{cases}$$

The statements follow from this.

□

Note that $f_{m,n}$ uses c_l for $l = 0 : m+n$ only; R uses c_{m+n+1} also. So, if c_l is given for $l = 0 : f$ then $f_{m,n}$ is defined for $m+n \le r$, $m \ge 0$, $n \ge 0$.

Example 3.4.3. Padé approximations of the exponential function. The Padé approximations $f_{m,n}(z)$ were computed by a program using the formulas of Theorem 3.4.2 for $f(z) = e^z$, with $0 \le m \le 4$, $\quad n = 4 - m$. In the Padé table these will be on the fourth diagonal, perpendicularly to the main diagonal. The results were first obtained in floating point arithmetic, but they were then converted into rational form by the algorithm described in Example 3.4.1. The results are exact. (It was numerically verified that the floating point results differed from the rational "approximations" by less than 10^{-15}.)

The coefficients p_i of the numerators $P_{m,4-m}$ and q_j of the denominators $Q_{m,4-m}$ are given in the following tables:

$m\backslash i$	0	1	2	3	4
0	1	0	0	0	0
1	1	1/4	0	0	0
2	1	1/2	1/12	0	0
3	1	3/4	1/4	1/24	0
4	1	1	1/2	1/6	1/24

$m\backslash j$	0	1	2	3	4
0	1	−1	1/2	−1/6	1/24
1	1	−3/4	1/4	−1/24	0
2	1	−1/2	1/12	0	0
3	1	−1/4	0	0	0
4	1	0	0	0	0

We find, for example that

$$f_{3,1}(z) = \frac{1 + \frac{3}{4}z + \frac{1}{4}z^2 + \frac{1}{24}z^3}{1 - \frac{1}{4}z}$$

The program also computed the error term and the error at $z = 1$. For $f_{3,1}$, these are, respectively, $.0021z^5$ and $.0039$. For comparison, for the Maclaurin expansion $f_{4,0}$, the corresponding values are $-.0083z^5$ and $-.0099$. Note that the coefficients of $f_{4,0}$ were the input of the computation of the other Padé approximants.

For $m = n = 4$ the program computed the exact rational values for all coefficients of $f_{4,4}$, and found that the error term is $-4\,10^{-8}z^9$, while the error term of the Maclaurin expansion $f_{8,0}$ is $3\,10^{-6}z^9$. The Padé approximant thus yields some convergence acceleration, but this effect is much more pronounced for functions where the Maclaurin coefficients decay slowly.

When $m = n = 10$ the program gave warnings about divisions by zero, and it estimated the condition number of the linear system (3.4.16) to be 10^{22}. The reciprocal of this number is a measure of how close the matrix of the system is to a singular matrix, (see Theorem 6.5.3).The computed coefficients of the Padé approximant had large errors. Nevertheless e was computed with full machine accuracy (for $z = 1$), and the error term was estimated to be less than $10^{-25}z^{21}$. This indicates that it may be possible to obtain full machine accuracy with smaller values of m, n.

There is an "if" in the theorem. There are in fact simple exceptional situations, where the linear system (3.4.16) is singular or almost singular. We shall indicate, how such situations can often be avoided by a more reasonable formulation of the request. These matters are discussed more thoroughly in Cheney [7, Chap. 5].

Example 3.4.4. Try to find

$$f_{1,1}(z) = (p_0 + p_1 z)/(q_0 + q_1 z), \quad q_0 = 1,$$

for $f(z) = \cos z = 1 - \frac{1}{2}z^2$. The coefficient matching according to the theorem, yields the equations, $p_0 = q_0 = 1$, $p_1 = q_1$, $-\frac{1}{2}q_0 = 0$. The last equation contradicts the condition that $q_0 = 1$. This single contradictory equation is in this case the "system" (3.4.16).

If this equation is ignored, we obtain $f_{1,1}(z) = (1 + q_1 z)/(1 + q_1 z) = 1$, with error $\approx \frac{1}{2}z^2$, in spite that we asked for an error that is $O(z^{m+n+1}) = O(z^3)$. If we instead allow that $q_0 = 0$, then $p_0 = 0$, and we obtain the same final result, since $f_{1,1}(z) = q_1 z/(q_1 z) = 1$.

In a sense, this singular case corresponds to a rather stupid request: we ask to approximate the even function $\cos z$ by a rational function where the numerator and the denominator end with odd powers of z. One should, of course, ask for the approximation by a rational function of z^2. What would you do, if $f(z)$ is an *odd* function?

Example 3.4.5.

Imagine a case where $f_{m-1,n-1}(z)$ happens to be a more accurate approximation to $f(z)$ than usual, say that $f_{m-1,n-1}(z) - f(z) = O(z^{m+n+1})$. (For instance, let $f(z)$ be the ratio of two polynomials of degree $m - 1$ and $n - 1$, respectively.) Let b be an arbitrary number, and choose

$$Q_{m,n}(z) = (z + b)Q_{m-1,n-1}(z), \quad P_{m,n}(z) = (z + b)P_{m-1,n-1}(z).$$

Then

$$f_{m,n}(z) = P_{m,n}(z)/Q_{m,n}(z) = P_{m-1,n-1}(z)/Q_{m-1,n-1}(z) = f_{m-1,n-1}(z),$$

which is an $O(z^{m+n+1})$-accurate approximation to $f(z)$. Hence our request for this accuracy is satisfied by more than one pair of polynomials, $P_{m,n}(z)$, $Q_{m,n}(z)$, since b is arbitrary. This is impossible, unless the system (3.4.16) (that determines $Q_{m,n}$) is singular.

This example illustrates another type of situations where the singular case occurs. Numerically, a similar situation occurs in a natural way, when one wants to approximate $f(z)$ by $f_{m,n}(z)$, although already $f_{m-1,n-1}(z)$ would represent $f(z)$ as well as possible with the limited precision of the computer. In this case we must expect the system (3.4.16) to be very close to a singular system. Actually, such a case was encountered in Example 3.4.3 when $m = n = 10$. A reasonable procedure for handling this is to compute the Padé approximants for a sequence of increasing values of m, n, to estimate the condition numbers (see Sec. 6.5.5), and to stop when it approaches the reciprocal of the machine unit. This illustrates a fact of some generality. *Unnecessary numerical trouble can be avoided by means of a well designed termination criterion.*

For $f(z) = -\ln(1-z)$, we have $c_l = 1/l$, $l > 0$. When $m = n$ the matrix of the system (3.4.16) turns out to be the notorious Hilbert matrix (with permuted columns), for which the condition number grows like $0.014\,10^{1.5n}$. (The elements of the usual Hilbert matrix are $a_{ij} = 1/(i+j-1)$.)

3.4.3 The Epsilon Algorithm.

We shall here briefly introduce the important ϵ-algorithm, and indicate the connections between *Padé approximation, Aitken acceleration, linear difference equations* and this algorithm.

If n is large, the heavy part of the computation of a Padé approximant

$$f_{m,n}(z) = P_{m,n}(z)/Q_{m,n}(z)$$

is the solution of the linear system (3.4.16). We see that if m or n is decreased by 1, most of the equations of the system will be the same. There are therefore relations between the polynomials $Q_{m,n}(z)$ for adjacent values of m, n, which have been subject to intensive research that has resulted in several interesting algorithms. See, e.g., the monographs of Brezinski [5, 6] and the literature cited there.

An extension of the Aitken acceleration, due to Shanks, which uses a comparison series with terms of the form

$$c_l = \sum_{\nu=1}^{n} \alpha'_\nu k_\nu^l, \quad l = 0, 1, 2, \ldots, k_\nu \neq 0, \tag{3.4.17}$$

was mentioned at the end of Sec. 3.3.2. α'_ν and k_ν are $2n$ parameters, to be determined, in principle, by means of c_l, $l = 0 : 2n-1$. The parameters may be complex. If The power series becomes

$$S(z) = \sum_{l=0}^{\infty} c_l z^l = \sum_{\nu=1}^{n} \alpha'_\nu \sum_{l=0}^{\infty} k_\nu^l z^l = \sum_{\nu=1}^{p} \frac{\alpha'_\nu}{1 - k_\nu z}.$$

This is a rational function of z, and the "Ansatz" of Shanks is thus related to Padé approximation, but note that the poles at k_ν^{-1} should be simple and that $m < n$ for $S(z)$, because $S(z) \to 0$, as $z \to \infty$. Recall that the calculations for the Padé approximation determines the coefficients of $S(z)$ *without calculating the* $2n$ *parameters* α'_ν *and* k_ν. It can happen that m becomes larger than n, and if α'_ν and k_ν are afterwards determined, by the expansion of $S(z)$ into partial fractions, it can turn out that some of the k_ν are multiple poles.

This suggests a generalization of the Shanks approach but how? If we consider the coefficients q_j, $j = 1 : n$, occurring in (3.4.16) as known quantities then (3.4.16) can be interpreted as a *linear difference equation* [103] . The general solution of this is given by (3.4.17), if the zeros of the polynomial $Q(x) := 1 + \sum_{j=1}^{n} q_j x^j$ are simple, but if multiple roots are allowed, the general solution reads, by Theorem 3.2.10 (after some change of notation),

$$c_l = \sum_{\nu} p_\nu(l) k_\nu^n,$$

where k_ν runs through the different zeros of $Q(x)$, and p_ν is an arbitrary polynomial, the degree of which equals the multiplicity -1 of the zero k_ν.

Essentially the same mathematical rlations occur in several areas of numerical analysis, such as interpolation and approximation by a sum of exponentials (Sec. 4.9), and in the design of quadrature rules with free nodes (Ch. 5).

In this chapter we are primarily interested in the use of Padé approximants as a convergence accelerator in the *numerical* computation of values of $f(z)$ for (say) $z = e^{i\phi}$, in particular for $z = \pm 1$. A natural question is whether it is possible to omit the calculation of the coefficients p_j, q_j and find a recurrence relation that gives the function values directly. A very elegant solution to this problem, called the ϵ-algorithm, was found in 1956 by P. Wynn [39], after complicated calculations. We shall present the algorithm, but we refer to the original paper of Wynn for the proof.

A two-dimensional array of numbers $\epsilon_k^{(p)}$ is computed by the recurrence relation,

$$\epsilon_{k+1}^{(p)} = \epsilon_{k-1}^{(p+1)} + \frac{1}{\epsilon_k^{(p+1)} - \epsilon_k^{(p)}}. \tag{3.4.18}$$

For *even* subscripts this yields

$$\epsilon_{2n}^{(p)} = f_{p+n,n}(z), \tag{3.4.19}$$

if the following boundary conditions are used:

$$\epsilon_0^{(p)} = f_{p,0}(z) = \sum_{l=0}^{p} c_l z^l, \quad \epsilon_{2n}^{(-n)} = f_{0,n}(z) = \frac{1}{\sum_{l=0}^{n} d_l z^l}, \quad \epsilon_{-1}^{(p)} = 0. \tag{3.4.20}$$

The polynomials $f_{0,n}(z)$ are thus obtained from the Taylor expansion of $\frac{1}{f(z)}$. Several procedures for obtaining this were given in Sec. 3.1. The values of $\epsilon_k^{(p)}$ with *odd* subscripts are auxiliary quantities only.

This algorithm is usually described in a rhombus scheme. It seems easier to program it after a slight change of notation. We introduce an $r \times 2r$ matrix $A = [a_{ij}]$, $a_{ij} = \epsilon_k^{(p)}$, where $k = j - 2$, $p = i - j + 1$. Conversely, $i = k + p + 1, j = k + 2$. The ϵ-algorithm, together with the boundary conditions now takes the form:

```
        **for** $i = 1 : r$
```

[103] This can also be expressed in terms of the z-transform, see S 3.2.3.

```
        $a_{i,1} = 0$;   $a_{i,2} = f_{i-1,0}(z)$;   $a_{i,2i} = f_{0,i-1}(z)$;
        for $j = 2 : 2*i-2$
            $a_{i,j+1} = a_{i-1,j-1} + 1/(a_{ij} - a_{i-1,j})$.
        end
    end
```

Results: $f_{m,n}(z) = a_{m+n+1,2n+2}$, $(m \geq 0,\ n \geq 0, m+n+1 \leq r)$. The above program sketch must be improved for practical use, e.g., something should be done about the risk for a division by zero.

Example 3.4.6. Not yet written The typical convergence rates of the ϵ-algorithm observed in numerical experiments are about the same as for GCA, Sec. 3.3.7.

3.4.4 More about Continued Fractions and the Padé Table.

** **A plan:**

* Toeplitz (look at Recipes Sec. 6.2) and Hankel Matrices, related to a power series, $H_k^{(n)}$ Hankel determinant; $\mathbf{H}_k^{(n)}$ Hankel matrix.

$$q_m^{(n)} = \frac{H_m^{(n+1)} H_{m-1}^{(n)}}{H_m^{(n)} H_{m-1}^{(n+1)}} \qquad e_m^{(n)} = \frac{H_{m+1}^{(n)} H_{m-1}^{(n+1)}}{H_m^{(n)} H_m^{(n+1)}}$$

* **quotient-difference algorithm**, *qd* algorithm; bf rhombus rules refer to Henrici, vol I,§ 7.6.

$$e_m^{(n)} = q_m^{(n+1)} - q_m^{(n)} + e_{m-1}^{(n+1)}; \quad m = 1, 2, \ldots,\ n = 0, 1, 2, \ldots,$$

$$q_{m+1}^{(n)} \frac{e_m^{(n+1)}}{e_m^{(n)}} q_m^{(n+1)};$$

with initial conditions $e_0^{(n)} = 0; q_1(n) = c_{n+1}/c_n,\ n = 0, 1, 2, \ldots.$

* Correspondence of formal power series $c_0 + c_1 z + c_2 z^2 + \ldots;\ c_0 \neq 0$ and a continued fraction

$$\sum_{j=0}^{\infty} c_j x^j = c_0 - \frac{q_1^{(0)} x}{1-} \frac{e_1^{(0)} x}{1-} \frac{q_2^{(0)} x}{1-} \frac{e_2^{(0)} x}{1-} \ldots$$

* Quote Henrici vol II, p. 518, Thm 12.4c +

* qd-algorithm (see § 7.6) (and determinant formulas Thm 7.6a).

* Example $\{ex34.dise\}$, Henrici vol I, p. 610, and qdalg1.dia.

* Series in powers of z^{-1}, Henrici vol II, p. 525.....

* incomplete gamma, different type of cf: S fraction, Henr.II, p. 561.

* Ref. to odd functions (3.4.9) in the text, divide f by z, and put $z^2 = x$.

* Example $\{ex34.logcf\}$,Henrici vol II, p. 534, and logcf.dia.

* The convergents of the corresponding continued fractions are equal to the sequence of Padé approximants with

$$[m, n] = [0, 0], [0, 1], [1, 1], [1, 2], [2, 2], [2, 3], [3, 3], [3, 4], \ldots$$

Review Questions

1. Define a continued fraction, and show how the convergents can be evaluated backwards and forwards.
2. Show how any positive number can be expanded into a continued fraction with integer elements. In what sense are the convergents the best approximations? How accurate are they?
3. What is the Padé table? Describe how the Padé approximants can be computed. Tell something about singular and almost singular situations that can be encountered, and how to avoid them.
4. Describe the ϵ-algorithm, and tell something about its background and its efficiency.

Problems and Computer Exercises

1. Write a program for the algorithm of Example3.4.1. Apply it to find a few coefficients of the continued fractions for $\frac{1}{2}(\sqrt{5}+1)$, $\sqrt{2}$, e, π, $\log 2/\log 3$, $2^{j/12}$ for a few integers j, $1 \le j \le 11$.
   Check the accuracy of the convergents. What happens when you apply your program to a rational number, e.g., 729/768 ? [104]
2. *A matrix formalism for continued fractions.*
   (a) We use the same notations as in Sec. 3.4.1, but we set, with no loss of generality, $b_0 = 0$. Set
   $$P(n) = \begin{pmatrix} p_{n-1} & p_n \\ q_{n-1} & q_n \end{pmatrix}, \qquad A(n) = \begin{pmatrix} 0 & a_n \\ 1 & b_n \end{pmatrix}.$$
   Show that $P(0) = I$, $P(n) = P(n-1)A(n)$, $P(n) = A(1)A(2)\cdots A(n-1)A(n)$, $n \ge 1$
   Comment: This does not minimize the number of arithmetic operations but, in a matrix-oriented programming language, it often gives very simple programs.
   (b) Write a program for this with some termination criterion, and test it on a few cases, e.g.,
   $$1 + \frac{1}{1+}\,\frac{1}{1+}\,\frac{1}{1+}\cdots; \qquad 2 + \frac{1}{3+}\,\frac{1}{2+}\,\frac{1}{3+}\,\frac{1}{2+}\,\frac{1}{3+}\cdots; \qquad 2 + \frac{2}{2+}\,\frac{3}{3+}\,\frac{4}{4+}\cdots.$$
   As a post-processing, apply in the first two cases, e.g., Aitken acceleration in order to obtain a very high accuracy. Does the result look familiar in the last case? See Problem 4 concerning the exact results in the two other cases.

[104] One of the convergents for $\log 2/\log 3$ reads 12/19. This is in a way basic for Western Music, where 13 quints make 7 octaves, i.e. $(3/2)^{12} \approx 2^7$.

(c) Write a version of the program with some strategy for scaling $P(n)$ in order to eliminate the risk of overflow and underflow. *Hint*: Note that the convergents $x_n = p_n/q_n$ are unchanged if you multiply the $P(n)$ by arbitrary scalars.

(d) Use this matrix form for working out a short proof of (3.4.4). *Hint*: What is the determinant of a matrix product?

3.

4. (a) Explain that $x = 1 + 1/X$ for the continued fraction in (3.4.7)?

(b) Compute the periodic continued fraction

$$2 + \frac{1}{3+}\,\frac{1}{2+}\,\frac{1}{3+}\,\frac{1}{2+}\,\frac{1}{3+}\cdots$$

exactly (by paper and pencil). The convergence is assured by a theorem of Seidel, mentioned in § 3.4.1.

(c) Suggest a generalization of (a) and (b), where you can always obtain a quadratic equation with a positive root.

(d) Show that

$$\frac{1}{\sqrt{x^2-1}} = \frac{1}{x-}\,\frac{\frac{1}{2}}{x-y} \quad \text{where} \quad y = \frac{\frac{1}{4}}{x-}\,\frac{\frac{1}{4}}{x-}\,\frac{\frac{1}{4}}{x-}\cdots.$$

5. (a) Prove the equivalence transformation (3.4.5). Show that the errors of the convergents have alternating signs, if the elements of the continued fraction are positive.

(b) Show how to bring a general continued fraction to the special form of equation (3.4.6).

6. (a) Write a program for computing a Padé approximant and the error term. Apply it (perhaps after a transformation), for various values of m, n to, e.g.,[105] e^z, $\arctan z$, $\tan z$. Use the algorithm of Example 3.4.1 for expressing the coefficients as rational numbers. For how large m, n can you (in these examples) use your program without severe trouble with rounding errors. *Hint:* Matlab is rather convenient for this.

(b) The part of the Padé table for e^z shown in the text, indicates a kind of symmetry of the Padé table for this particular function. Formulate and prove this, and try to determine for which other functions the Padé table has a similar symmetry.

7. (a) Show that there is at most one rational function $R(z)$, where the degrees of the numerator and denominator do not exceed, respectively, m and n, such that $f(z) - R(z) = O(z^{m+n+1})$, as $z \to 0$, even if the system (3.4.16) is singular. (Note, however, that P_m and Q_n are not uniquely determined, if the system is singular; they have common factors.)

(b) Is it true that if $f(z)$ is a rational function of degrees m', n', then $f_{m,n}(z) = f(z)$, $\forall m \geq m'$, $n \geq n'$?

[105]Note that two of these examples are odd functions.

8. Apply the ϵ-algorithm to some of the examples and problems of Sec. 3.3, where other methods for convergence acceleration have been used. Compare the efficiency of the methods (in particular the CGA method.)

9. Check that the program sketch for the ϵ-algorithm is equivalent with the scheme with the quantities $\epsilon_k^{(p)}$ given earlier in the text. How do you obtain the boundary values?

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* National Bureau of Standards, Dover Publications, New York, NY, 1965.

[2] F. L. Bauer, H. Rutishauser and E. Stiefel, *New aspects in numerical quadrature.* Proc. of Symposia in Appl. Math., 15, Amer. Math. Soc. 1963, pp. 199–218.

[3] P. Bjørstad, G. Dahlquist, and E. Grosse, *Extrapolations of asymptotic expansions by a modified Aitken δ^2-formula.* BIT, 21 (1981), pp. 56–65.

[4] J. M. Borwein, P. B. Borwein and K. Dilcher, *Pi, Euler numbers and asymptotic expansions*, Amer. Math. Monthly, 96 (1989), pp. 681–687.

[5] C. Brezinski, *Padé-Type Approximations and General Orthogonal Polynomials*, Birkhäuser Verlag, Basel, 1980.

[6] C. Brezinski, *History of Continued Fractions and Padé Approximants*, Springer Verlag, Berlin, 1991.

[7] E. W. Cheney. *Introduction to Approximation Theory.* McGraw-Hill, New York, NY, 1966.

[8] R. Courant. *Differential and Integral Calculus. Vol. I.* Blackie & Son, London, 1934. Reprinted 1988 in Classics Library, Wiley.

[9] G. G. Dahlquist. *On Summation Formulas due to Plana, Lindelöf and Abel, and Related Gauss–Christoffel Rules, II.* BIT 37:4, 804–832, 1997.

[10] P. Deuflhard and A. Hohmann, *Numerical Analysis, A First Course in Scientific Computation.* Walter de Gruyter & Co., Berlin, 1995.

[11] C- E. Fröberg *Lärobok i numerisk analys.* (In Swedish.) Svenska Bokförlaget/Bonniers, 1962.

[12] C- E. Fröberg *Numerical Mathematics.* Benjamin/Cummings Publ. Co., Menlo Park, CA, 1985.

[13] W. Gautschi *Numerical Analysis, an Introduction* Birkhäuser, Boston, MA, 1997.

[14] H. H. Goldstine. *The Computer from Pascal to von Neumann.* Princeton University Press, Princeton, NJ, 1972.

[15] A. Griewank and G. F. Corliss, eds. *Automatic Differentiation of Algorithms: Theory, implementation and Application* . SIAM Proc. Ser., SIAM, Philadelphia, PA, 1991.

[16] A. Griewank, D. Juedes and J. Utke, eds. *ADOL-C: A package for the automatic differentiation of algorithms written in C/C++* . ACM Trans. Math. Software, 1997.

[17] S. Å. Gustafson, *Convergence acceleration on a general class of power series.* Computing, 21, 53–69, 1978.

[18] P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley, New York, 1962.

[19] P. Henrici. *Elements of Numerical Analysis.* John Wiley, New York, NY, 1964.

[20] P. Henrici. *Applied and computational complex analysis.* John Wiley, New York, NY, vol. 1, 1974.

[21] P. Henrici. *Applied and computational complex analysis.* John Wiley, New York, NY, vol. 2, 1977.

[22] F. Klein. *Elementary Mathematics from an Advanced Standpoint,* Dover Publications, New York, 1945. (Translation from German original, 1924.)

[23] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* 2nd ed. Addison-Wesley, Reading, MA, 1981.

[24] N. N. Lebedev. *Special Functions and Their Applications.* Dover Publications, New York, 1972. (Translation from Russian original.).

[25] C. C. Lin and L. A. Segel, *Mathematics Applied to Deterministic Problems in the Natural Sciences.* Macmillan Publ. Co, Inc, New York, 1974.

[26] B. Lindberg, *A simple interpolation algorithm for improvement of the numerical solution of a differential equation.* SIAM J. Numer. Anal., 9, pp. 662–668, 1972.

[27] N. E. Nörlund, *Vorlesungen über Differenzenrechnung.* Springer-Verlag, Berlin, 1924, or Chelsea Publ.Co, New York, 1954.

[28] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes; The Art of Scientific Computing,* 2nd ed. Cambridge University Press, Cambridge, GB, 1992.

[29] O. Perron. *Die Lehre von den Kettenbrüchen,.* 3rd ed., Teubner, Stuttgart, 1957.

[30] H. Riesel. *Prime Numbers and Computer Methods for Factorization.* 2nd ed., Progr. Math.,v. 126, Birkhäuser, Boston, MA, 1994.

[31] J. F. Steffensen. *Interpolation.* , 2nd ed., Chelsea, New York, 1950.

[32] F. Stenger. *Numerical Methods Based on Sinc and Analytic Functions.* Springer-Verlag, Berlin, 1993.

[33] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA, USA, 1986.

[34] E.C. Titchmarsh. *The Theory of Functions.* 2nd. edition, Oxford University Press, London, 1939.

[35] J. Todd. *Notes on numerical analysis I, Solution of differential equations by recurrence relations.* Math. Tables Aids Comput 4, 39-44, 1950.

[36] H. S. Wall. *Analytic Theory of Continued Fractions.* Van Nostrand, Princeton, NJ, 1948.

[37] D. V. Widder. *The Laplace Transform.* Princeton University Press, Princeton, NJ, 1941.

[38] D. V. Widder. *An Introduction to Transform Theory.* Academic Press, New York, 1971.

[39] P. Wynn. *On a device for computing the $e_m(s_n)$ transformation.* M.T.A.C., 10, pp. 91–96, 1956.

Chapter 4

Interpolation and Related Subjects

4.1 The Interpolation Problem

4.1.1 Introduction

We have previously encountered two types of interpolation

- Piecewise linear interpolation that is commonly used in tables, when the requirements of accuracy are modest. A more modern application is in Computer Graphics.

- Interpolation of the values of a function in n equidistant points by a function in $\mathcal{P}_n$. Recall that in Sec. 3.2.2, $\mathcal{P}_n$ was defined as the space of polynomials in one variable of degree *less than* n; n is the number of data required to specify a polynomial in $\mathcal{P}_n$; the dimension of the linear space $\mathcal{P}_n$ is n.[1]

The first type of interpolation will be generalized in Sec. 4.6 where we shall study interpolation by **piecewise polynomials**, in particular by **splines**, which are piecewise polynomials, where a few derivatives are required to be continuous at the knots, i.e., the joints of the pieces.

In the first two sections we shall go deeper into the following **polynomial interpolation problem** for non-equidistant, distinct points: *Find a polynomial* $p^* \in \mathcal{P}_m$ *such that*

$$p^*(x_i) = f(x_i), \quad i = 1 : m, \quad x_i \neq x_j \text{ for } i \neq j. \tag{4.1.1}$$

Recall that, by the corollary of Theorem 3.2.1, $p^*(x)$ *is uniquely determined* for a given **grid**, $(x_1, x_2, \ldots, x_m)$. This theorem is general, although the rest of Sec. 3.2 dealt with interpolation polynomials in the equidistant case only, and their appli-

[1]Some authors use similar notations, e.g., $\mathbf{P}_n$ or Π_n, to denote the $n+1$-dimensional space of polynomials of degree *less than or equal to* n.

cation to numerical differentiation and integration.[2] Also note that the formulation and the solution of this problem are *independent of the ordering of the points x_i.*

A set of polynomials $\{p_1(x), p_2(x), \dots, p_m(x)\}$, such that any polynomial $p \in \mathcal{P}_m$ can be expressed as a linear combination

$$p(x) = c_1p_1(x) + \ldots + c_mp_m(x),$$

is called a **basis** in $\mathcal{P}_m$. The coefficients make a column vector $c = (c_1, c_2, \ldots c_m)^T$ that can be viewed as a *coordinate vector* of p in the space $\mathcal{P}_n$, with respect to this basis. The **power basis**, where $p_j(x) = x^{j-1}$, i.e., $p(x) = \sum_{j=1}^n c_jx^{j-1}$, is the simplest basis, though not always the best.

The interpolation problem (4.1.1) leads to a linear system of equations

$$c_1p_1(x_i) + c_2p_2(x_i) + \ldots + c_mp_m(x_i) = f(x_i), \quad i = 1 : m. \tag{4.1.2}$$

or if we introduce the matrix $M_p = [p_j(x_i)]_{i,j=1}^m$, and the column vector[3] $\tilde{f} = \big(f(x_1), f(x_2), \ldots, f(x_m)\big)^T$,

$$M_pc = \tilde{f}. \tag{4.1.3}$$

The proof of Theorem 3.2.1 was based on the fact that this matrix is non-singular in the case of the power basis; in this case it is the **Vandermonde matrix** $[x_i^{j-1}]_{i,j=1}^n$.

In any basis $\{q_1(x), q_2(x), \ldots, q_n(x)\}$, the q_j must be linear combinations of the p_k, $k = 1 : n$. This can be expressed in vector-matrix form:

$$\big(q_1(x), q_2(x), \ldots, q_n(x)\big) = \big(p_1(x), p_2(x), \ldots, p_n(x)\big)A, \tag{4.1.4}$$

where A is a constant matrix. *A must be non-singular.* For, if it were singular then there would exist a non-trivial vector v such that $Av = 0$, hence

$$(q_1(x), q_2(x), \ldots, q_n(x))v = (p_1(x), p_2(x), \ldots, p_n(x))Av = 0 \quad \forall x,$$

and $(q_1(x), q_2(x), \ldots, q_n(x))$ would thus not be a basis.

Similarly set $M_q = [q_j(x_i)]_{i,j=1}^n$. By putting $x = x_i$, $i = 1 : m$ into (4.1.4), we see that $M_q = M_pA$. M_q is non-singular for every basis. If we set $p(x) = \sum d_jq_j(x)$, the system (4.1.2) becomes for this basis $M_qd = \tilde{f}$. Then $M_pc = \tilde{f} = M_qd = M_pAd$, hence $c = Ad$; the matrix A is thus like a coordinate transformation in Geometry. We collect the recent formulas for convenient reference:

$$M_pc = \tilde{f} = M_qd, \quad c = M_p^{-1}\tilde{f}, \quad M_pA = M_q, \quad c = Ad. \tag{4.1.5}$$

Example 4.1.1. *An application to numerical integration.* We shall find a formula for integrals of the form $I = \int_0^1 x^{-1/2}f(x)\,dx$ that is exact for $f \in \mathcal{P}_m$ and uses the values $f(x_i)$, $i = 1 : m$. Such integrals need a special treatment, due to the integrable singularity at $x = 0$.

[2]It is de facto so, although the polynomials were invisible in the derivations of formulas by operator methods.

[3]We try to make a distinction between f that is an element in some function space and $\tilde{f} \in \mathbf{R}^m$.

Set $\mu_j = \int_0^1 x^{-1/2} p_j(x)\,dx$, and introduce the row vector $\mu^T = (\mu_1, \mu_2, \dots, \mu_n)$. Then

$$I \approx \int_0^1 x^{-1/2} p(x)\,dx = \sum_{j=1}^{m} c_j \mu_j = \mu^T c = \mu^T(p) M_p^{-1} \tilde{f}, \tag{4.1.6}$$

where it is emphasized in the last expression that μ^T depends on the basis. In fact $\mu^T(q) = \mu^T(p)A$, and $M_q^{-1} = A^{-1}M_p^{-1}$; we see that the approximation to I is independent of the basis, as it should, in view of Theorem (3.2.1).

Another approach is **the method of undetermined coefficients**, i.e., to seek a formula $I \approx \sum_{i=1}^{m} b_i f(x_i) \equiv b^T \tilde{f}$, which is exact when $f(x) = p_j(x)$, $j = 1 : m$; then it is exact for all $p \in \mathcal{P}_m$ These conditions lead to the linear system $b^T M_p = \mu^T$.[4] Thus $b^T = \mu^T M_p^{-1}$, the final result of this approach becomes $I \approx b^T \tilde{f} = \mu(p)^T M_p^{-1} \tilde{f}$, which is *the same as* (4.1.6), although interpolation was not mentioned in this approach. In view of Theorem (3.2.1) this is no surprise, since the same values of f are used, and both formulas are exact for all $f \in \mathcal{P}_m$.
See numerical applications (for the power basis) in Problem 2. Evidently these two approaches can be used for any linear functional.

A variant of this interpolation problem is to find a polynomial $p \in \mathcal{P}_n$, where $n < m$ that, in some sense, fits to the data $x_i, f(x_i)$, $i = 1 : m$. We obtain, in this case too, a system of the form $M_p c = \tilde{f}$, but this is said to be **overdetermined**; there are m rows but only n columns. An overdetermined system can typically be satisfied only approximately; see Example 1.2.5, where a straight line could not be made to pass through the five points. This is not interpolation, but it is mentioned here, since it is a natural extension of interpolation, and in some mathematical software packages these two problems can easily be treated by the same commands.

Overdetermination can be used to attain two different types of **smoothing**:

(a) to reduce the effect of random or other irregular errors in the values of the function;

(b) to give the polynomial a smoother behaviour between the grid points.

One of several possible methods for the treatment of overdetermined linear systems is the important **the method of least squares**, see Chapter 8. Its application to the approximation of functions leads to rather simple computations. In many applications it can be motivated by statistical arguments. In this case we shall *find the vector c that minimizes* $S(c)$, where (see Problem 5)

$$S(c) = \sum_{i=1}^{m} \Big(\sum_{j=1}^{n} c_j p_j(x_i) - f(x_i) \Big)^2. \tag{4.1.7}$$

4.1.2 Various bases for $\mathcal{P}_n$

How to choose the basis, depends on the purpose. If the purpose is to compute derivatives or integrals of the interpolation polynomial, the power basis is a conve-

[4]This may be called the *adjoint* to the system $M_p c = \tilde{f}$.

nient choice, and *the shifted power basis*, where $q_j(x) = (x-a)^{j-1}$, is sometimes also convenient. Convenience is, however, not all that matters, when n is large. Mathematically, the choice of basis (for a finite-dimensional space) makes no difference. Computationally, working with *rounded values of the coefficients*, the choice of basis can make a great difference. If a shifted power basis is to be used for polynomial approximation on a certain interval, it is often good to choose a near the midpoint of the interval.

Other bases are often more advantageous.

A **cardinal basis** of $\mathcal{P}_m$ is generated by a polynomial

$$\Phi_m(x) = (x - x_1)(x - x_2)\cdots(x - x_m), \tag{4.1.8}$$

where x_i, $i = 1 : m$ are m distinct real numbers. The basis reads,

$$p_j(x) = \frac{\Phi_m(x)}{(x - x_j)\Phi'(x_j)}, \quad j = 1 : m. \tag{4.1.9}$$

By L'Hospital's rule $p_j(x_i) = \delta_{ij}$, (Kronecker's delta). (This is the property that in a more general context characterizes a cardinal basis.)

It follows that $\sum_{j=1}^{m} p(x_j)p_j(x) = p(x_i)$, for $x = x_i$, $i = 1 : m$, hence this is a basis that directly displays the solution of the interpolation problem for m distinct points. This is **Lagrange's interpolation formula**, but it seems to be relatively hard to compute the value of a basis function. We shall see in §4.2.3 that this basis is more practical than it looks.

A sequence of polynomials $q_1, q_2, q_3, \ldots$ (finite or infinite),

$$\begin{aligned}
q_1(x) &= a_{11}\\
q_2(x) &= a_{12} + a_{22}x\\
q_3(x) &= a_{13} + a_{23}x + a_{33}x^2\\
&\ \ \cdots\\
q_m(x) &= a_{1m} + a_{2m}x + a_{3m}x^2 + \ldots + a_{mm}x^{m-1}\\
&\qquad\qquad \cdots
\end{aligned}$$

where $a_{jj} \neq 0$ for all j, is defined to be a **triangle family** of polynomials, i.e., $q_j(x)$ is of $(j-1)$'th degree with a non-zero leading coefficient.[5].

Conversely, for any j, p_j, where $p_j(x) = x^{j-1}$ can be expressed recursively and uniquely as linear combinations of $q_1(x), q_2(x), \ldots, q_j(x)$, so that we obtain a triangular scheme also for the inverse transformation. So *every triangular family* $\{q_1(x), q_2(x), \ldots, q_m(x)\}$ *is a basis for* $\mathcal{P}_m$.

What we has just seen, is indeed a proof of the well known fact that the inverse of a triangular matrix, (with no zeros in the main diagonal) is also triangular. Among interesting triangle families we can mention the families where $q_{j+1}(x)$ is defined by $(x-c)^j$, $T_j(x)$, and many other families of orthogonal polynomials.

[5]In the terminology used in the previous subsection, this triangular matrix equals A^T; this explains the notation for the elements

Example 4.1.2. *A suitable basis for polynomial interpolation.* Suppose that $x_1, x_2, \dots, x_m$, are m distinct points. The formulas

$$p_1(x) = 1, \quad p_j(x) = (x - x_1)(x - x_2)\dots(x - x_{j-1}), \quad j = 1 : m,$$

define a triangle family with unit leading coefficients, hence a basis. The representation

$$\begin{aligned} p(x) = c_1 + c_2(x - x_1) + c_3(x - x_1)(x - x_2) + \dots \\ + c_m(x - x_1)(x - x_2)\cdots(x - x_{m-1}) \end{aligned} \tag{4.1.10}$$

is often very convenient. If a polynomial is given in this form then $p(x)$ can be evaluated using only m multiplications and $2m$ additions, for a given numerical value x, from the nested form

$$\begin{aligned} p(x) = (\cdots(c_m(x - x_{m-1}) + c_{m-1})(x - x_{m-2}) + \\ \cdots + c_3)(x - x_2) + c_2)(x - x_1) + c_1. \end{aligned}$$

Notice that the case, where all the $x_i = 0$, gives Horner's rule, see Sec. 1.4.2. We have $p(x) = b_1$, where b_1 is computed using the recursion formula:

$$b_m := c_m, \quad b_{i-1} := b_i(x - x_{i-1}) + c_{i-1}, \quad i = m : -1 : 2. \tag{4.1.11}$$

We leave the proof to Problem 4.

Note that, by (4.1.10), $c_1 = p(x_1)$, $(x_2 - x_1)c_2 = p(x_2) - c_1, \dots$. This indicates that the coefficients c_j can be computed recursively, by much less work than the linear system (4.1.3) would require for the power basis, by standard methods for linear algebra. In next section we shall see how this basis leads to Newton's interpolation formula, which belongs to the best interpolation formulas, with respect to computational economy and numerical stability.

The matrix approach described in the previous subsection are convenient; a Vandermonde matrix is easily generated, when you work in a matrix-oriented command language. If you deal with a modest number of polynomials of low degree, (say) less than 10, convenience can be given a larger weight than the optimal number of arithmetical operations and a minimized effect of rounding errors. In the latter respects, the matrix approach is inferior to Newton's interpolation formula and the other formulas and algorithms to be discussed in the next section, in particular if m is large.

The main reason why we started with such non-optimal procedures, is that they are *easily generalizable to other interpolation problems*, e.g., interpolation in other function spaces than $\mathcal{P}_m$ (see Problem 3), or interpolation with other conditions on the function f in addition to function values (see later sections). For such a non-standard interpolation problem—problems like that occur in practice—the matrix approach is helpful also for finding out under what conditions the problem has a unique solution.

Problems

1. Study experimentally interpolation in $\mathcal{P}_m$, $m = 2 : 2 : 16$ for $f(x) = (3 + x)^{-1}$, $x \in [-1, 1]$. Use the linear system $M_p c = \tilde{f}$ and the power basis. Study both m equidistant points and m Chebyshev points, i.e.,
$$x_i = -1 + 2\frac{i-1}{m-1}, \qquad x_i = \cos\left(\frac{2i-1}{m}\frac{\pi}{2}\right),$$
respectively. Plot the error curve, $y = |f(x) - p(x)|$ in semi-logarithmic scale. For the larger values of m, make also experiments to illuminate the effects from random perturbations of the function values to the values of $p(x)$.

   Make also a few experiments with a random vector $\tilde{f}$, for $m = 16$ and $m = 8$, in order to compare the grid data and the order of magnitude of $p(x)$ between the grid points.

2. Write a program for the derivation of a formula for integrals of the form $I = \int_0^1 x^{-1/2} f(x)\,dx$ that is exact for $f \in \mathcal{P}_m$ and uses the values $f(x_i)$, $i = 1 : m$, by means of the power basis, according to Example 4.1.1.

   (a) Compute the coefficients b_i for $m = 6 : 8$ with equidistant points, $x_i = (i - 1)/(m - 1)$, $i = 1 : m$. Apply the formulas to the integrals
$$\int_0^1 x^{-1/2} e^{-x}\,dx; \quad \int_0^1 \frac{dx}{\sin\sqrt{x}}; \quad \int_0^1 (1 - t^3)^{-1/2}\,dt.$$
In the first of the integrals compare with the result obtained by series expansion in Problem 3.1.1. In the last integral a substitution is needed for bringing it to the right form.

   (b) Do the same for the case, where the step size $x_{i+1} - x_i$ grows proportionally to i; $x_1 = 0$; $x_m = 1$. Is the accuracy significantly different compared to (a), for the same number of points?

   (c) Make some very small random perturbations of the x_i, $i = 1 : m$ in (a), (say) of the order of 10^{-13}. Of which order of magnitude are the changes in the coefficients b_i , and the changes in the results for the first of the integrals?

3. *A little about two-dimensional interpolation problems.*

   Much of the theory of the introduction can be generalized to other interpolation problems than problems with polynomials in one variable, but one cannot be sure that there is unconditionally a unique solution to the problem. It may not be enough to require that the points are distinct.

   (a) For the interpolation by a linear function in two variables, $p(x, y; c) = c_1 + c_2 x + c_3 y$, characterize triples of triples (x_i, y_i, f_i), such that the interpolation problem problem $p(x_i, y_i; c) = f_i$, $i = 1 : 3$ has exactly one solution. When does it have infinitely many solutions, and when is there no solution?

   (b) Set
$$c = (c_1, c_2, c_3, c_4, c_5, c_6)^T; \quad p(x, y; c) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2.$$
Consider the interpolation problem: Given x_i, y_i, f_i, $i = 1 : 6$, try to find c, so that $p(x_i, y_i; c) = f_i$, $i = 1 : 6$.

   Choose x_i, y_i, f_i, $i = 1 : 6$ by 18 independent random numbers, solve the linear system $p(x_i, y_i; c) = f_i$, $i = 1 : 6$, look at $\max|c_i|$. Repeat this (say) 25 times. You have a fair chance to avoid singular cases, or cases where $\max|c_i|$ is very large.

(c) Now choose (x_i, y_i) as 6 distinct points on some circle in $\mathbf{R}^2$, and choose f_i at random. This should theoretically lead to a singular matrix. Explain why, and find experimentally the rank (if your software has convenient commands or routines for that). Find a general geometric characterization of the sextuples of points (x_i, y_i), $i = 1 : 6$, that lead to singular interpolation problems. *Hint*: Brush up your knowledge of conic sections.

4. Prove the validity of (4.1.11).

5. *A little about least squares approximation.* The sum $S(c)$ was defined in (4.1.7). Show that

$$\frac{\partial S(c)}{\partial c_k} = \sum_{i=1}^{m} 2p_k(x_i)\Big(\sum_{j=1}^{n} c_j p_j(x_i) - f(x_i)\Big),$$

and show that the equations for the minimization can be written in the form

$$M^T M c = M^T \tilde{f}. \tag{4.1.12}$$

in the notations of §4.1.1, (but $M = M_p$). Show that $M^T M$ is a symmetric $n \times n$ matrix.

COMMENT: This linear system is called the **normal equations**. If $m = n$ it is mathematically equivalent to the system $Mc = \tilde{f}$, but the condition number may be much larger. It will be shown in Ch. 7 that the matrix $M^T M$ is non-singular, and that the system yields the minimum of $S(c)$, unless the columns of M are linearly dependent. The matrix is, however, sometimes very ill-conditioned, and we shall in Chapter 8 see better methods for handling the least squares problems in such situations.

6. *A warning for polynomial extrapolation of empirical functions, or... ?*

(a) Write a program $c = polyapp(x, y, n)$ that finds the coefficient vector c for a polynomial in $p \in \mathcal{P}_n$, in a shifted power basis, such that $y_i \approx p(x_i)$, $i = 1 : m$, $m \geq n$, in the least squares sense, or study a program that does almost this.[6]

(b) The following data shows the development of the Swedish GDP, quoted (with permission) from a table made by a group associated with the Swedish Employer's Confederation. (The data are expressed in prices of 1985 and scaled so that the value for 1950 is 100.) (a) For the upper pairs of data, compute and plot $p(x), x \in$

1950	1955	1960	1965	1970	1975	1980	1985	1990
100.0	117.7	139.3	179.3	219.3	249.1	267.5	291.5	326.4
1952	1957	1962	1967	1972	1977	1982	1987	1992
104.5	124.6	153.5	189.2	226.4	247.7	270.2	307.6	316.6

$[1950, 2000]$ (say). Mark the given data points. Do this for $m = 9$, and for (say) $n = 9$, and then for $n = 8 : -2 : 2$. Store the results, so that comparisons can be made afterwards.

Hint: Subtract 1970 from the years.

(b) Do the same for the lower pairs of data. Organize the plots, so that interesting

[6]The Matlab command polyfit does *almost* this.

comparisons become convenient, e.g. how well were the data points of one of the sets interpolated by the results from the other set?

(c) Make forecasts for 1995 and 2000 with both data sets. Then, use a reduced data set, e.g., for the years 1982 and earlier (so that $m = 7$), and and compare the forecasts for 1987 and 1992 with the given data. (Isn't it a reasonable test for every suggested forecast model to study its ability to predict the present from the past?) See if you obtain better results with the logarithms of the GDP values.

Try to draw general conclusions about this type of prediction, and—if you have time—see if they still hold, when you make similar computations with data from another model.

4.2 Interpolation Formulas and Algorithms

4.2.1 Divided Differences and Newton's Interpolation Formula

Let the unique solution p^* in $\mathcal{P}_m$ of the interpolation problem be expressed with the basis of Eq.(4.1.10). Suppose that

$$\begin{aligned} f(x) = c_1 + c_2(x-x_1) + \ldots + c_m(x-x_1)(x-x_2)\cdots(x-x_{m-1}) \quad (4.2.1)\\ +A_m(x)(x-x_1)(x-x_2)\cdots(x-x_m), \end{aligned}$$

If $f \in \mathcal{P}_m$, we know from Sec. 4.1.2 that such a formula holds with $A_m(x) \equiv 0$, otherwise we do not know yet that this Ansatz is correct, but we shall see that it is so.

For $x = x_1$ we get $c_1 = f(x_1)$. Set

$$[x]f = f(x), \qquad [x_1, x]f = \frac{f(x) - f(x_1)}{x - x_1}.$$

Then

$$\begin{aligned} [x_1, x]f = c_2 + c_3(x-x_2) + \ldots + c_m(x-x_2)\cdots(x-x_{m-1})\\ +A_m(x)(x-x_2)\cdots(x-x_m), \end{aligned}$$

and $c_2 = [x_1, x_2]f$. We now *define* recursively, for $k \geq 1$, **divided differences**[7]

$$[x_1, x_2, \ldots, x_{k-1}, x_k, x]f = \frac{[x_1, x_2, \ldots, x_{k-1}, x]f - [x_1, x_2, \ldots, x_{k-1}, x_k]f}{x - x_k}. \quad (4.2.2)$$

We obtain, for $k = 2$,

$$\begin{aligned} [x_1, x_2, x]f = c_3 + c_4(x-x_3) + \ldots + c_m(x-x_3)\cdots(x-x_{m-1})\\ +A_m(x)(x-x_3)\cdots(x-x_m), \end{aligned}$$

and $c_3 = [x_1, x_2, x_3]f$. By induction it follows that

$$c_k = [x_1, x_2, \ldots, x_{k-1}, x_k]f, \qquad k = 1 : m. \quad (4.2.3)$$

[7]We now prefer the modern notation $[\ldots]f$ to the older notations $f[\ldots]$ or $f(\ldots)$, since it emphasizes that $[\ldots]$ is an operator. Note that the interpretation $[x]f = f(x)$ is consistent with this.

For $k = m$ we obtain, $A_m(x) = [x_1, x_2, \dots x_m, x]f$.

We now introduce a new notation that is convenient in the following. For $k = 1 : m$, set

$$\Phi_0(x) = 1, \quad \Phi_k(x) = \Phi_{k-1}(x)(x - x_k) = (x - x_1)(x - x_2) \cdots (x - x_k). \quad (4.2.4)$$

For $f \in \mathcal{P}_m$ we know that the Ansatz is correct, hence we can trust the coefficients c_k. Moreover, since $p^*(x_j) = f(x_j)$, $j = 1 : m$, it follows that $[x_1, x_2, \dots x_j]p^* = [x_1, x_2, \dots x_j]f$. Hence

$$\begin{aligned}
p(x) &= \sum_{j=1}^{m} [x_1, \dots, x_j]p \, \Phi_{j-1}(x) \\
&\equiv \sum_{j=1}^{m} [x_1, \dots, x_j]p \cdot (x - x_1) \cdots (x - x_{j-1}) \quad \forall p \in \mathcal{P}_m, \\
p^*(x) &= \sum_{j=1}^{m} [x_1, \dots, x_j]f \, \Phi_{j-1}(x) \\
&\equiv \sum_{j=1}^{m} [x_1, \dots, x_j]f \cdot (x - x_1) \cdots (x - x_{j-1}).
\end{aligned}$$

For a general function f we do not yet know that the Ansatz is correct, but after inserting the only possible values of c_k and $A_m(x)$ in (4.2.2), together with the notation with $\Phi_k(x)$, we can conjecture that the following is an identity:

$$f(x) = \sum_{j=1}^{m} [x_1, \dots, x_j]f \, \Phi_{j-1}(x) + [x_1, x_2, \dots x_m, x]f \, \Phi_m(x).$$

We prove this by induction. For $m = 1$, it is true, because the right hand side becomes $f(x_1) + [x_1, x]f \cdot (x - x_1) = f(x)$ =the left hand side. Next suppose that it is true for $m = n$. The difference between the right hand side for $m = n + 1$ and $m = n$ reads

$$\begin{aligned}
&[x_1, \dots, x_{n+1}]f \, \Phi_n(x) - [x_1, x_2, \dots x_n, x]f \, \Phi_n(x) + [x_1, x_2, \dots x_{n+1}, x]f \, \Phi_{n+1}(x) \\
&= \big([x_1, \dots, x_{n+1}]f - [x_1, x_2, \dots x_n, x]f + [x_1, x_2, \dots x_{n+1}, x]f \ (x - x_{n+1})\big)\Phi_n(x) \\
&= \big([x_1, \dots, x_{n+1}, x]f \ (x_{n+1} - x) + [x_1, x_2, \dots x_{n+1}, x]f \ (x - x_{n+1})\big)\Phi_n(x) = 0,
\end{aligned}$$

hence the conjecture is true. We summarize the results in a theorem.

Theorem 4.2.1. *Newton's Interpolation Formula with exact remainder.*

The interpolation problem of determining the polynomial $p^* \in \mathcal{P}_m$ such that

$$p^*(x_i) = f(x_i), \qquad i = 1 : m,$$

where the x_i are distinct points, has the solution

$$p^*(x) = \sum_{j=1}^{m} [x_1, \dots, x_j]f \, \Phi_{j-1}(x), \quad (4.2.5)$$

$$f(x) - p^*(x) = [x_1, x_2, \dots x_m, x]f \, \Phi_m(x). \quad (4.2.6)$$

These formulas are valid also for complex x_i, x.

In particular, if $f \in \mathcal{P}_m$ then, for all x, $[x_1, x_2, \ldots, x_m, x]f = 0$. For $x = x_{m+1}$, this equation is, by Theorem 3.2.1, the only non-trivial relation of the form $\sum_{j=1}^{m+1} a_j f(x_j) = 0$ that holds for all $f \in \mathcal{P}_m$,.

Do not forget the simple Horner-like scheme (4.1.11) for the computation of $p^*(x)$; it is less obvious in the new notation.

Theorem 4.2.2.

For every m, the divided difference $[x_1, x_2, \ldots, x_m]f$ is the coefficient of x^{m-1} in the interpolation polynomial $p^* \in \mathcal{P}_m$. A divided difference is a symmetric function of its arguments.

Proof: The first statement follows from (4.2.3). The second statement then holds, because the interpolation polynomial is independent of how the points are ordered. □

If we use (4.2.2), but call the first point x_{i+1} instead of x_1 we get

$$[x_{i+1}, \ldots, x_{k-1}, x_k, x]f = \frac{[x_{i+1}, \ldots, x_{k-1}, x]f - [x_{i+1}, \ldots, x_{k-1}, x_k]f}{x - x_k}.$$

Now set $x = x_i$ and use the symmetry property (Theorem 4.2.2). We obtain the formula

$$[x_i, x_{i+1}, \ldots, x_{k-1}, x_k]f = \frac{[x_i, x_{i+1}, \ldots, x_{k-1}]f - [x_{i+1}, \ldots, x_{k-1}, x_k]f}{x_i - x_k}. \qquad (4.2.7)$$

This formula can be used recursively to compute the divided differences. The computation is conveniently arranged in a table (recall that $[x_i]f = f(x_i)$).

$$\begin{array}{lllll}
x_1 & [x_1]f & & & \\
 & & [x_1, x_2]f & & \\
x_2 & [x_2]f & & [x_1, x_2, x_3]f & \\
 & & [x_2, x_3]f & & [x_1, x_2, x_3, x_4]f \\
x_3 & [x_3]f & & [x_2, x_3, x_4]f & \\
 & & [x_3, x_4]f & & \\
x_4 & [x_4]f & & &
\end{array}$$

This table is called a **divided-difference table**. *Each entry in the table is computed from the two entries immediately to the left and above and below.* Hence the complete table can be constructed, for example, column by column.

The divided differences which occur in Newton's interpolation formula (4.2.6) are those in the upper diagonal of the table. However, since the points $x_1, x_2, \ldots, x_m$ can be arbitrarily ordered, we can also introduce the points in Newton's interpolation formula in *backward* order $x_m, \ldots, x_1$. This gives the *backward form* for the interpolating polynomial

$$p^*(x) = f(x_m) + \sum_{j=1}^{m-1} [x_m, x_{m-1} \ldots, x_{m-j}]f \cdot (x - x_{m-j+1}) \cdots (x - x_m). \qquad (4.2.8)$$

The divided differences in this formula lie on the *upward* diagonal starting at f_m in the table.

Example 4.2.1. Compute the interpolation polynomial for the following table using (4.2.6).

$$\begin{array}{lcccc}
x_1 = 1 & \mathbf{0} & & & \\
 & & \mathbf{2} & & \\
x_2 = 2 & 2 & & \mathbf{1} & \\
 & & 5 & & \mathbf{0} \\
x_3 = 4 & 12 & & 1 & \\
 & & 8 & & \\
x_4 = 5 & 20 & & &
\end{array}$$

(The entries appearing in the Newton interpolation formula are boldface.) We get two alternative representations

$$\begin{aligned}
p(x) &= 0 + 2(x-1) + 1(x-1)(x-2) + 0(x-1)(x-2)(x-4) \\
&= 20 + 8(x-5) + 1(x-5)(x-4) + 0(x-5)(x-4)(x-2) = x^2 - x,
\end{aligned}$$

where the last is obtained from (4.2.8).

Example 4.2.2. Set $f(x;z) = 1/(z-x)$; x is the variable, z is a parameter; both may be complex. The following elementary, though remarkable, expansion can be proved directly by induction (Problem 3a).

$$\begin{aligned}
\frac{1}{z-x} &= \frac{1}{z-x_1} + \frac{x-x_1}{(z-x_1)(z-x_2)} + \ldots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{m-1})}{(z-x_1)(z-x_2)\cdots(z-x_m)} \\
&\quad + \frac{(x-x_1)\cdots(x-x_m)}{(z-x_1)\cdots(z-x_m)(z-x)} = \sum_{j=1}^{m} \frac{\Phi_{j-1}(x)}{\Phi_j(z)} + \frac{\Phi_m(x)}{\Phi_m(z)(z-x)}. \quad (4.2.9)
\end{aligned}$$

When we match this with Newton's interpolation formula we find that

$$[x_1, x_2, \ldots, x_j]f(x;z) = \frac{1}{\Phi_j(z)}, \qquad [x_1, x_2, \ldots, x_j, x]f(x;z) = \frac{1}{\Phi_j(z)(z-x)}. \quad (4.2.10)$$

These formulas can also be proved by induction, by working algebraically with $1/(z-x)$ in the divided difference table (Problem 3b).

An interesting feature is that these formulas do *not* require that the points x_i are distinct. (They are consistent with the extension to non-distinct points that will be made in Sec. 4.3.) Everything is continuous except if the parameter $z = x_i$, $i = 1 : m$, or, of course if $z = x$, see Sec. 4.3. If all the x_i coincide, we obtain a geometric series with a remainder.

This is more than a particular example. Since $1/(z-x)$ is the kernel of Cauchy's integral (and several other integral representations), this expansion is easily generalized to arbitrary analytic functions, see Sec. 4.4.

Theorem 4.2.3. *Lagrange's Remainder Term for Interpolation*

Let f be a given real function, $f \in C^m[\operatorname{int}(x, x_1, x_2, \dots, x_m)]$. Denote by p^* the solution of the interpolation problem (4.1.1). Then

$$f(x) - p^*(x) = \frac{f^{(m)}(\xi_x)}{m!}\Phi_m(x), \tag{4.2.11}$$

for some point $\xi_x \in \operatorname{int}(x, x_1, x_2, \dots, x_m)$, and

$$[x_1, x_2, \dots x_m, x_{m+1}]f = \frac{f^{(m)}(\xi)}{m!}, \quad \xi \in \operatorname{int}(x_1, \dots, x_{n+1}). \tag{4.2.12}$$

Proof: Introduce a new variable z, and set

$$G(z) = f(z) - p^*(z) - [x_1, x_2, \dots x_m, x]f\, \Phi_m(z).$$

We know by Theorem 4.2.1 that

$$f(x) - p^*(x) = [x_1, x_2, \dots x_m, x]f\, \Phi_m(x). \tag{4.2.13}$$

hence $G(x) = 0$. Then $G(z) = 0$ for $z = x, x_1, x_2 \dots, x_m$. From repeated use of Rolle's theorem it follows that there exists a point $\xi_x \in \operatorname{int}(x, x_1, x_2, \dots, x_m)$, such that $G^{(m)}(\xi_x) = 0$. Since $p^{(m)}(z) = 0$ and $\Phi_m^{(m)}(z) = m!$ for all z, $G^{(m)}(z) = f^{(m)}(z) - [x_1, x_2, \dots x_m, x]f\, m!$. If we now put $z = \xi_x$, we obtain

$$[x_1, x_2, \dots x_m, x]f = \frac{f^{(m)}(\xi_x)}{m!}. \tag{4.2.14}$$

Put this into the definition of $G(z)$, and set $z = x$. Since $G(x) = 0$, the first statement follows. The second statement follows from (4.2.14) for $x = x_{m+1}$. ☐

In this theorem x_i, x, $f(x)$, etc. must be *real*, while (4.2.13), i.e., Newton's interpolation formula with the exact remainder term, is valid also in the complex plane.

Notice the similarity to the remainder term in Taylor's formula. We shall see that this can be considered as a limiting case when all the points x_i coincide. Notice also that the right hand side of (4.2.11) is zero at the grid points—as it should be.

Example 4.2.3. Use the same notations as before. For $f(x) = x^m$ the interpolation error becomes $f(x) - p^*(x) = \Phi_m(x)$, because $f^{(m)}(x)/m! \equiv 1$. Fig. 4.2.1 shows the interpolation error with m equidistant points on $[-1, 1]$ and with m Chebyshev points on the same interval, i.e.,

$$x_i = -1 + 2\frac{i-1}{m-1}, \qquad x_i = \cos\Big(\frac{2i-1}{m}\frac{\pi}{2}\Big),$$

respectively, for $m = 6$ and $m = 12$. The behaviour of the error curves are rather typical for functions where $f^{(m)}(x)$ is slowly varying. Also note that the error increases rapidly, when x leaves the interval $\operatorname{int}(x_1, x_2, \dots, x_m)$. In the equidistant case, the error is quite large also in the outer parts of the interval.

Figure 4.2.1. *Error of interpolation in $\mathcal{P}_m$ for $f(x) = x^{m+1}$. Equidistant points (upper part), Chebyshev points (lower part); $m = 6$ and $m = 12$ (to be made).*

To form the Newton interpolation polynomial we only need one diagonal of the divided-difference table, and it is not necessary to store the entire table.

Algorithm 4.2.1 *Computing the Newton Coefficients*
The following program replaces (overwrites) the function values $f_1, f_2, \dots, f_m$, where $f_i = f(x_i)$, $i = 1 : m$. by the diagonal $f_i = [x_1, x_2, \dots, x_i]f$, $i = 1 : m$ of the divided difference table. At step j the jth column of the table is computed. Note that it is necessary to proceed from the bottom of each column to avoid overwriting data needed later! The algorithm uses $m(m-1)/2$ divisions and $m(m-1)$ subtractions.

```
for $j = 1 : m - 1$
    for $i = m : -1 : j + 1$
        $f_i := (f_i - f_{i-1})/(x_i - x_{i-j})$;
    end
end
```

The Newton interpolation polynomial is then given by (4.2.8), and can be evaluated using the recursion (4.1.11).

Algorithm 4.2.2 *Divided Difference Table*
The following algorithm computes the difference table one diagonal at a time. In the ith step the entries $f_i, [x_{i-1}, x_i]f, \dots, [x_1, x_2, \dots, x_i]f$ on the upward diagonal of the divided-difference table overwrites the function values $f_i, f_{i-1}, \dots, f_1$.

```
for $i = 2 : m$
    for $j = i : -1 : 2$
        $f_j := (f_j - f_{j-1})/(x_i - x_{j-1});$
    end
end
```

Theorem 4.2.4. *If $x_k = x_1 + (k-1)h$, $f_k = f(x_k)$, then*

$$[x_i, x_{i+1}, \ldots, x_{i+j}]f = \frac{\Delta^j f_i}{h^j j!}. \tag{4.2.15}$$

Proof. By induction, with the use of equation (4.2.7). The details are left to the reader. □

We are now in a position to give a short proof of the important formula (3.2.4) that we formulate as a theorem.

Theorem 4.2.5. *Assume that $f \in C^k$. Then*

$$\Delta^k f(x) = h^k f^{(k)}(\zeta), \qquad \zeta \in [x, x+kh]. \tag{4.2.16}$$

If $f \in \mathcal{P}_k$ then $\Delta^k f(x) = 0$. Analogous results hold, mutatis mutandis, for backward and central differences.

Proof: Combine the result in Theorem 4.2.4 with (4.2.12), after appropriate substitutions. □

4.2.2 Scaled Divided Differences

We have noted above that, in the notation for the equidistant case, $\nabla^k f_n \approx h^k f^{(k)}$, while in the divided difference notation $f[x_n, x_{n-1}, \ldots, x_{n-k}] \approx f^{(k)}/k!$. For the basis functions of the interpolation formulas, we have, respectively,

$$\binom{x}{k} = O(1), \qquad (x-x_n)(x-x_{n-1})\cdots(x-x_{n-k+1}) = O(h^k),$$

provided that $x - x_{n-j} = O(h)$, $j = 0, 1, \ldots, k-1$.

For many applications the quantities used in the equidistant case have a more appropriate order of magnitude. F. Krogh[16] introduced a scaling for the divided differences with the same advantage; in the equidistant case these scaled divided differences are identical to $\nabla^k f_n$.[8] The main known applications so far has been to multistep methods for ordinary differential equations.

We shall introduce some new notions and notations. Let $X = (x_0, x_1, x_2, \ldots)$, and $Y = (y_0, y_1, y_2, \ldots)$, be two given sequences, and set

$$\phi(x; n, k) = (x - x_n)(x - x_{n-1}) \cdots (x - x_{n-k}).$$

(Note that the degree of the polynomial ϕ is $k + 1$.)

We first introduce an *operator notation* for the usual divided differences. Let $[X; n, k]Y$ be the k'th order backward divided difference of the sequence Y with respect to the sequence X at x_n, i.e. if $y_j = f(x_j)$ $\forall j$ then, in the notation of Sec. 4.1,

$$[X; n, k]Y = [x_n, x_{n-1}, \ldots, x_{n-k}]f.$$

The argument X can be omitted in the divided difference, if there is no doubt about the sequence. The basic recurrence relation (4.2.7) for divided differences now reads[9]

$$[n, 0]Y = y_n, \quad [n + 1, j + 1]Y = \frac{[n + 1, j]Y - [n, j]Y}{x_{n+1} - x_{n-j}}. \tag{4.2.17}$$

Let $P(x; n, k)Y$ be the interpolation polynomial of degree k that assumes the values y_{n-j} for $x = x_{n-j}$, $j = 0, 1, \ldots, k$. Newton's interpolation formula for the kth degree polynomial then reads

$$P(x; n, k)Y = [n, 0]Y + \sum_{i=1}^{k} \phi(x; n, i - 1)[n, i]Y. \tag{4.2.18}$$

The **scaled divided differences** $\langle n, k\rangle Y$ are defined as follows,

$$\langle n, 0\rangle Y = [n, 0]Y = y_n, \quad \langle n, j\rangle Y = \phi(x_n; n - 1, j - 1)[n, j]Y, \tag{4.2.19}$$

and we set $\phi_s(x; n, j) = \phi(x; n, j)/\phi(x_n; n-1, j)$. The basic recurrence relation now reads,

$$\langle n + 1, j + 1\rangle Y = \langle n + 1, j\rangle Y - \phi_s(x_{n+1}; n, j - 1)\langle n, j\rangle Y. \tag{4.2.20}$$

Note that in the equidistant case, $\langle n, j\rangle Y = \nabla^j y_n$. In this notation Newton's interpolation formula reads

$$P(x; n, k)Y = \langle n, 0\rangle Y + \sum_{i=1}^{k} \phi_s(x; n, i - 1)\langle n, i\rangle Y. \tag{4.2.21}$$

Algorithm 4.2.3 Given a backward (upward))diagonal in the difference scheme, $\langle X; n, i\rangle Y$, $i = 0 : k$, computed by means of (4.2.20). The value of the interpolation

[8]The notation and the algorithm below are due to L. O. Eriksson and G. Dahlquist.

[9]This formula, and several formulas below, can be written as a pure operator relation, without Y.

polynomial $P(x;n,k)Y$ and its derivative can be computed simultaneously by the following algorithm that also shows how to compute $\phi_s(x;n,k-1)$ by a recurrence relation.

```
$\varphi := 1$; $p := \langle n,k\rangle y$; $p1 := 0$;
**for** $i = k-1 : -1 : 0$
    $t := (x - x_{n-i})/(x_n - x_{n-i-1})$;
    $\varphi := \varphi * t$;
    $p := \langle n,i\rangle y + p * t$;
    $p1 := p/(x_n - x_{n-i-1}) + p1 * t$;
**end**
$\phi_s(x;n,k-1) := \varphi$;
$P(x;n,k)Y := p$; $P'(x;n,k)Y := p1$;
```

4.2.3 Lagrange's Interpolation Formula

We here consider an alternative construction of the unique interpolation polynomial.

Theorem 4.2.6. *Lagrange's interpolation formula.*

The unique interpolation polynomial $p^* \in \mathcal{P}_m$ interpolating the function $f(x)$ at the distinct points x_i, $i = 1 : m$ can be written

$$p^*(x) = \sum_{i=1}^{m} f(x_i) L_i(x), \tag{4.2.22}$$

where

$$L_i(x) = \prod_{\substack{j=1\\ j\neq i}}^{m} \frac{(x - x_j)}{(x_i - x_j)}, \qquad i = 1 : m, \tag{4.2.23}$$

are the **Lagrange polynomials**.

Proof: The Lagrange polynomials are exactly of degree $m-1$ and satisfy

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{if } i \neq j. \end{cases}$$

It follows that $p^* \in \mathcal{P}_m$, and that $p^*(x_j) = f_j$, $j = 1 : m$. □

The expression (4.2.22) is not as efficient for the calculation of a sequence of interpolating values as the Newton formula, since the functions $L_i(x)$ have to be recomputed for each new value of x. However, we can write

$$L_i(x) = \mu_i(x)\Phi_m(x), \qquad \mu_i(x) = \frac{\lambda_i}{x - x_i}, \tag{4.2.24}$$

where

$$\lambda_i = 1\Big/\prod_{\substack{j=1\\ j\neq i}}^{m}(x_i - x_j), \qquad \Phi_m(x) = \prod_{j=1}^{m}(x - x_j). \tag{4.2.25}$$

Here the *support coefficients* λ_i depend only on the given points x_i, $i = 1 : m$, and can be computed in $m(m+1)$ operations, or even $\frac{1}{2}m(m+1)$ (see Schwarz [21, Sec. 3.2.1]). Since the interpolation formula is exact for $f(x) \equiv 1$, we have $\sum_{i=1}^{m} L_i(x) \equiv 1$. Then, by (4.2.22) and (4.2.24),

$$p^*(x) = \frac{\sum_{i=1}^{m} f(x_i)L_i(x)}{\sum_{i=1}^{m} L_i(x)} = \frac{\sum_{i=1}^{m} f(x_i)\mu_i(x)}{\sum_{i=1}^{m} \mu_i(x)}. \tag{4.2.26}$$

Now, for each new value of x we only need to evaluate the coefficients $\mu_i(x)$, which requires $m+1$ divisions, and then we compute $p^*(x)$ by means of the last expression.

4.2.4 Neville's and Aitken's algorithms

There are other recursive algorithms for interpolation. Of interest are those based on *successive linear interpolations.* The basic formula is given in the following theorem.

Theorem 4.2.7.

Assume that the two polynomials $p_{m-1}(x)$ and $q_{m-1}(x)$, both in $\mathcal{P}_{m-1}$ interpolate $f(x)$ at the points $x_1, \dots, x_{m-1}$, and $x_2, \dots, x_m$, respectively. If the m points $x_1, x_2, \dots, x_{m-1}, x_m$ are distinct then

$$p_m(x) = \frac{x - x_1}{x_m - x_1} q_{m-1}(x) + \frac{x_m - x}{x_m - x_1} p_{m-1}(x).$$

is the unique polynomial in $\mathcal{P}_m$ that interpolates $f(x)$ at the m points $x_1, x_2, \dots, x_{m-1}, x_m$.

PROOF: Since $q_{m-1}(x)$ and $p_{m-1}(x)$ both interpolate $f(x)$ at the points $x_2, \dots, x_{m-1}$ and

$$\frac{x - x_1}{x_m - x_1} + \frac{x_m - x}{x_m - x_1} = 1,$$

it follows that also $p_m(x)$ interpolates $f(x)$ at these points. Further, $p_m(x_1) = p_{m-1}(x_1)$ and hence interpolates $f(x)$ at x_1. A similar argument shows that, $p_m(x)$ interpolates $f(x)$ at $x = x_m$. Hence $p_m(x)$ is the unique polynomial interpolating $f(x)$ at the distinct points $x_1, x_2, \dots, x_m$. □

Neville's and **Aitken's** algorithms both use Theorem 4.2.7 repeatedly to construct successively higher order interpolation polynomials. In Neville's interpolation algorithm one puts

$$p_{i,0} = f(x_i), \qquad i = 1 : m$$

and defines recursively for $i = 2 : m$

$$p_{i,k+1} = \frac{(x - x_{i-k})p_{i,k} - (x - x_i)p_{i-1,k}}{x_i - x_{i-k}}, \qquad k = 1 : i-1, \tag{4.2.27}$$

where $p_{i,k+1}$ interpolates at the points $x_{i-k},\dots,x_{i-1},x_i$. Note that it is easy to add new interpolation points in this scheme. Aitken's scheme is similar, but more adapted to the case when the number of points is fixed, see Isaacson and Keller [15, 1966, Chapter 6.2]. These formulas are better than Newton's only in the case that $f(x)$ is to be evaluated for the same values of $x-x_i$ for several functions (sequences) f. In this case one should compute $t_{ik} = (x - x_{i-k})/(x_i - x_{i-k})$ once and for all. We saw in Sec. 3.3.5 its application to the extrapolation to $x = 0$ of a polynomial given at a few positive arguments, a typical example, where it is efficient and widely used.

4.2.5 Miscellaneous Questions about Interpolation

We discuss here some miscellaneous questions about interpolation and applications.

Example 4.2.4. *Error in linear interpolation.* Suppose we want to compute by linear interpolation the value $f(x)$ at a point $x = x_0 + \theta h$, $0 \le \theta \le 1$, where we have put $h = x_1 - x_0$. (θ is called the **normalized interpolation argument**.) Assume that $|f''(x)| \le M_2$ $\forall x \in \operatorname{int}(x_1, x_2)$. Then from (4.2.11) it follows that the remainder $R(x) = f(x) - p(x)$ satisfies

$$|R(x)| = \frac{1}{2}|f''(\xi)(x - x_0)(x - x_1)| \approx \frac{h^2}{2}\theta(1-\theta)M_2 \le \frac{h^2}{8}M_2, \tag{4.2.28}$$

where we have used that $\theta(1-\theta)$ takes on its maximum value $1/4$ for $\theta = 1/2$. If the values f_0 and f_1 are given to t correct decimal digits the round-off errors in these values ϵ_0 and ϵ_1 satisfy $|\epsilon_i| \le \frac{1}{2}10^{-t}$. In linear interpolation $p(x) = (1-\theta)f_0 + \theta f_1$, so for $0 \le \theta \le 1$ the round-off error R_T in $p(x)$

$$|R_T| = |(1-\theta)\epsilon_0 + \theta\epsilon_1| \le \frac{1}{2}10^{-t}.$$

Thus if $h^2 M_2 \le 4 \cdot 10^{-t}$, then $|R(x)| \le \frac{1}{2} \cdot 10^{-t}$ and so the total error in $p(x)$ is bounded by 10^{-t}, twice the round-off errors in the given values of f. Hence higher order interpolation is not necessary in this case.

A rule of thumb is that linear interpolation can be used if $|\delta^2 f_n|/8$ is a tolerable error; it is wise to put a safety margin on this to allow for an irregular error component.

Example 4.2.5. *Inverse interpolation.* It often happens that one has a sequence of pairs $\{(x_i, y_i)\}$ and want to determine a point where $y(x) = c$. We saw an example as early as in the simulation of the motion of a ball (Sec. 1.4), when we computed the landing point. We there used linear interpolation.

In general a natural approach is to reverse the roles of x and y, i.e., to compute the inverse function $x(y)$ for $y = c$, by means of Newton's interpolation formula with the divided differences $[y_i, y_{i+1}, \dots y_{i+j}]x$ (unscaled or scaled). It is convenient to order the points so that $\ldots < y_5 < y_3 < y_1 < c < y_2 < y_4 < \ldots$. This approach is successful if the function $x(y)$ is suitable for local approximation by a polynomial.

Sometimes, however, the function $y(x)$ is much better suited for local approximation by a polynomial than the inverse function $x(y)$. Then we can instead, for some m, solve the following equation,

$$y_1 + [x_1, x_2]y \cdot (x - x_1) + \sum_{j=2}^{m-1} [x_1, x_2, \dots x_{j+1}]y \, \Phi_j(x) = c.$$

Again it is convenient to order the points so that the root comes in the middle, e.g., so that $\dots < x_5 < x_3 < x_1 < \text{root} < x_2 < x_4 < \dots$.

We write the equation in the form $x = x_1 + F(x)$, where

$$F(x) \equiv \frac{(c - y_1) - \sum_{j=2}^{m-1} [x_1, x_2, \dots x_{j+1}]y \, \Phi_j(x)}{[x_1, x_2]y}.$$

Then we can use *iteration.* We ignore the sum to get the first guess x^0; this means the same as linear inverse interpolation. We then iterate, $x^i = x_1 + F(x^{i-1})$, until x^i and x^{i-1} are close enough. A more careful termination criterion will be suggested in Ch. 5, where the effect on the result of errors like the interpolation error is also discussed.

Suppose that $x_i - x_1 = O(h)$, $i > 1$, where h is some small parameter in the context (usually some step size), then $\Phi_j(x) = O(h^j)$, $\Phi_j'(x) = O(h^{j-1})$. The divided differences are $O(1)$, and we assume that $[x_1, x_2]y$ is bounded away from zero. Then the terms of the sum decrease like h^j.

By the discussion of iteration in Sec.1.2, the convergence ratio is $F'(x)$, and this is here approximately $\Phi_2'(x)[x_1, x_2, x_3]y/[x_1, x_2]y = O(h)$. So, if h is small enough, the iterations converge rapidly. If more than two iterations are needed, Aitken acceleration (Sec. 3.3.2) may be practical.

Example 4.2.6. *On numerical differentiation* We shall study the computation of $f'(x_0)$ by (3.2.19), i.e., set

$$g_n = \frac{f(x_{n+1}) - f(x_{n-1})}{2h}; \quad f'(x_0) = \Big(1 - \frac{\delta^2}{6} + \frac{\delta^4}{30} - \frac{\delta^6}{140} + \dots\Big) g_0.$$

Suppose that the irregular errors of the function values are bounded by U, and denote their effect on $f'(x_0)$ by R_{XF}. It has been mentioned several times—see, e.g., Example 3.3.15 in connection with the use of Richardson extrapolation for numerical differentiation— that irregular errors in the values of $f(x)$ are of much greater importance in numerical differentiation than in interpolation and integration.

The formula above is used to compute $f'(3)$, where $f(x) = \ln x$. The truncation error (called R_T) is approximately $\mu\delta^{2k+1} f_0 \approx h^{2k+1} f^{(2k+1)}(3) = (2k)!(h/3)^{2k+1}$, where k is the number of terms.

k	1	2	3
R_T	$0.012h^2$	$0.0033h^4$	$0.0024h^6$
R_{XF}	U/h	$1.5U/h$	$11U/(6h)$

In a log-log diagram the plots of R_T and R_{XF} versus h, are six straight lines that illustrate quantitatively the Scylla and Charybdis situation (see explanation in Sec. 3.1.4); the truncation error increases, and the effect of the irregular error decreases with h. One sees how the choice of h, which minimizes the sum of the bounds for the two types of error, depends on U and k, and tells what accuracy can be obtained. The optimal point is a little above and to the left of the intersection of the R_T-line and the R_{XF}-line for the same k. (Why?)

The effect of the pure rounding errors is important, though it should not be exaggerated. When macheps= 10^{-16}, one can obtain the first two derivatives very accurately by the optimal choice of h.

For $U = \frac{1}{2}10^{-6}$ the authors found that for $k = 2$, $h = 0.2$, the error bound is nearly $9\cdot 10^{-6}$, and not much better with the best choice of h for $k = 3$. For $k = 1$, $h = 0.03$ is a good choice—the error bound is about $3\,10^{-5}$. It is left to the user (Problem 8) to check and modify the experiments and conclusions indicated in this example. See also the appendix of Ch. 12, where similar questions are discussed in a more general context, namely differentiation for vectorvalued functions of vector-valued arguments.

4.2.6 The Runge Phenomenon

Even equidistant interpolation can give rise to convergence difficulties, when the number of interpolation points becomes large. This difficulty is often referred to as **Runge's phenomenon**, and we illustrate it in the following example. A more advanced discussion is given in Sec. 4.5.2, by means of complex analysis.

Example 4.2.7. The function f, whose graph is the continuous curve shown in Fig. 4.2.1, is approximated in two different ways by a polynomial of degree 10 in $[-1, 1]$.

The dashed curve has been determined by interpolation on the equidistant grid with 11 points ($m = 10$)

$$x_i = -1 + 2i/m, \qquad i = 0, 1, \ldots, m. \tag{4.2.29}$$

The graph of the polynomial so obtained has—unlike the graph of f—a disturbing course between the grid points. The agreement with f near the ends of the interval is especially bad, while near the center of the interval $[-\frac{1}{5}, \frac{1}{5}]$ the agreement is fairly good. Such behavior is typical of *equidistant interpolation of fairly high degree*, and can be explained theoretically.

The *dotted* curve in Fig. 4.2.6 has been determined by interpolation at the so-called **Chebyshev abscissae**

$$x_i = \cos\frac{2i+1}{m+1}\frac{\pi}{2}, \qquad i = 0, 1, \ldots, m, \tag{4.2.30}$$

($m = 10$). This procedure is studied more closely in the next section. The agreement with f is now much better than with equidistant interpolation, but still not good.

Figure 4.2.2. *Polynomial interpolation of* $1/(1 + 25x^2)$ *in two ways by the use of 11 points: equidistant points (dashed curve), Chebyshev abscissae (dotted curve).*

The function is not at all suited for approximation by one polynomial over the entire interval. Here one would get a much better result using approximation by rational functions (somewhat of a trick, since the curve shown is the graph of $f(x) = 1/(1+25x^2)$), or with piecewise polynomials— see Sec. 4.6 and Chapter 12.

Notice that the difference between the values of the two polynomials is much smaller at the grid points of the equidistant grid than in certain points between the grid points, especially in the outer parts of the interval. This intimates that the values which one gets by equidistant interpolation with a polynomial of high degree can be very sensitive to disturbances in the given values of the function. Put another way, *equidistant interpolation using polynomials of high degree is in some cases an ill-conditioned problem, especially in the outer parts of the interval* $[x_0, x_m]$. The effect is even worse if one extrapolates—i.e., if one computes values of the polynomial outside the grid. However, equidistant interpolation works well near the center of the interval.

Even with equidistant data one can often get a more well-behaved curve by — instead of interpolating—fitting a polynomial of lower degree (e.g., $n = 6$) using the method of least squares. Generally, if one chooses $n < 2\sqrt{m}$, then the polynomial fit is quite well conditioned, but higher values of n should be avoided. In the above example, however, the agreement would still be quite bad, even at the grid points, when the degree is chosen to be so low.

If one intends to approximate a function *in the entire interval* $[-1, 1]$ by a *polynomial* and can choose the points at which the function is computed or measured, then one should choose the Chebyshev abscissae. Using these points, interpolation is a fairly *well-conditioned* problem in the entire interval and one can conveniently fit a polynomial of lower degree than m, if one wishes to smooth errors in measurement; see the next section. The risk of disturbing surprises between the grid points is insignificant.

Example 4.2.7 shows how important it is to study the course of the approximating curve $p^*(x)$ between the points which are used in the calculation before one accepts the approximation. When one uses procedures for approximation for which one does not have a complete theoretical analysis, one should make an *experimental perturbational calculation.* In the above case such a calculation would very probably reveal that the interpolation polynomial reacts quite strongly if the values of the function are disturbed by small amounts, say $\pm 10^{-3}$. This would give a basis for rejecting the unpleasing dashed curve in the example, even if one knew nothing more about the function than its values at the equidistant grid points.

Review Questions

1. Prove the theorem which says that the interpolation problem for polynomials has a unique solution.
2. When is linear interpolation sufficient?
3. Derive Newton's interpolation formula.
4. Derive Newton's interpolation formula for the equidistant case, starting from Newton's general interpolation formula. How is this formula easily remembered?
5. Discuss how various sources of error influence the choice of step length in numerical differentiation.
6. Derive the Lagrange interpolation formula.

Problems

1. (a) Compute $f(3)$ by quadratic interpolation in the following table:

x	1	2	4	5
$f(x)$	0	2	12	21

   Use the points $1, 2$, and 4, and the points $2, 4$, and 5, and compare the results.

   (b) Compute $f(3)$ by cubic interpolation.

2. Compute $f(0)$ using one of the interpolation formulas treated above on the following table:

x	0.1	0.2	0.4	0.8
$f(x)$	0.64987	0.62055	0.56074	0.43609

   The interpolation formula is here used for *extrapolation.* Use also Richardson extrapolation.

3. Work out the details of Example 4.2.2 (about divided differences etc. for $1/(z-x)$).
4. (a) Consider the two polynomials $p(x)$ and $q(x)$, both in $\mathcal{P}_m$, which interpolate $f(x)$ at the points $x_1, \dots, x_m$, and $x_2, \dots, x_{m+1}$, respectively. Assume that $\{x_i\}_{i=1}^{m+1}$ is an increasing sequence, and that $f^{(m)}(x)$ has constant sign in the interval $[x_1, x_{m+1}]$. Show that $f(x)$ is contained between $p(x)$ and $q(x)$ for all $x \in [x_1, x_{m+1}]$.

   (b) Suppose that $f(x) = f_1(x) - f_2(x)$, where both $f_1^{(m)}(x)$ and $f_2^{(m)}(x)$ have the same constant sign in $[x_1, x_{m+1}]$. Formulate and prove a kind of generalization of the result in (a).
5. An example with inverse interpolation. Not complete!!!

6. Derive an approximate formula for $f'(x_0)$ when the values $f(x_{-1}), f(x_0), f(x_1)$ are given at three *nonequidistant* points. Give an approximate remainder term. Check the formula and the error estimate on an example of your own choice.

7. Given a sequence of function values $f_1, f_2, f_3 \dots$ at equidistant points $x_j = x_0 + jh$. $\min f_j$ takes place at $j = n$. Let $p^*(x)$ be the quadratic interpolation polynomial determined by f_{n-1}, f_n, f_{n+1}. Show that

$$\min p^*(x) = a - \frac{(\mu\delta f_n)^2}{2\delta^2 f_n}, \quad \text{is obtained at} \quad x = x_n - h\frac{\mu\delta f_n}{\delta^2 f_n}.$$

Show that the error of the maximum value can be estimated by $\max|\Delta^3 f_j|/\sqrt{243}$, where j is in some neighborhood of n. Why and how is the estimate of x less accurate?

Write a handy program that includes the search all local maxima and minima.

Sketch or work out improvements of this algorithm, perhaps with ideas of inverse interpolation and with cubic interpolation. And perhaps for non-equidistant data.

8. Check the table and the conclusions in Example 4.2.6, and see how the attainable accuracy varies with U. Draw the log-log diagram mentioned in the text.

What is the best accuracy one can hope for (with the optimal choice of h, for these three values of k, if $U = 10^{-16}$. Study also the analogous question for $f''(x_0)$.

9. Prove the validity of Algorithm 4.2.3

10. Given a backwards (upwards) diagonal in the table of divided differences (scaled or unscaled), $\langle X; m, i\rangle Y$, $0, 1, \dots, k$. Find a recurrence formula for the computation of the next diagonal of the difference scheme for the interpolation polynomial $P(x; m, k)Y$, i.e., if $\langle X; m{+}1, k\rangle Y = \langle X; m, k\rangle Y$ (why?), find $\langle X; m{+}1, i\rangle Y$, $i = k{-}1$, $k-2, \dots, 0$.

Hint: Look up the equidistant case in Example 3.2.6.

11. Show that, if the points x_i are distinct,

$$[x_1, x_2, \dots, x_m]f = \sum_{i=0}^{m} \frac{f(x_i)}{\Phi'_m(x_i)},$$

where $\Phi_m(x)$ is defined in (4.2.25).

Hint: Compare the coefficients of x^{m-1} in Newton's and Lagrange's expressions for the interpolation polynomial.

4.3 Interpolation where values of derivatives are used.

4.3.1 Hermite Interpolation

The **osculating polynomial** is a generalization of the interpolation polynomial, where at n different points of interpolation $\{x_i\}_{i=1}^n$, we require agreement also with the first $r_i - 1$ derivatives of $f(x)$; $r_i \geq 1$, $\sum_{i=1}^n r_i = m$. This polynomial arises as the result of passages to the limit in n groups of points, where in a group r_i distinct interpolation points coalesce into one point x_i. We say that the point x_i has **multiplicity** r_i. The polynomial in (4.1.8) becomes

$$\Phi_m(x) = \Pi_{i=1}^n (x - x_i)^{r_i}.$$

This is also described as **Hermite interpolation** (or osculatory interpolation) in $\mathcal{P}_m$ with a sequence of m (non-distinct) interpolation points, where a point of multiplicity r_i occurs r_i times.

For example, the Taylor polynomial in $\mathcal{P}_m$

$$p(x) = \sum_{j=0}^{m-1} \frac{f^{(j)}(x_0)}{j!}(x - x_0)^j \tag{4.3.1}$$

interpolates $f(x)$ at the point x_0 with multiplicity m (or x_0 is repeated m times). We use here, and in the following, the notation $f^{(0)}(x)$ for $f(x)$.

Theorem 4.3.1.

The problem of finding a polynomial $\hat{p} \in \mathcal{P}_m$ that satisfies the Hermite interpolation conditions

$$p^{(j)}(x_i) = f^{(j)}(x_i), \qquad i = 1:n, \quad j = 0:r_i - 1, \quad (r_i \geq 1, \quad \sum r_i = m), \tag{4.3.2}$$

has a unique solution.

Proof: The conditions are expressed by a system of m linear equations for the coefficients of $\hat{p}$, with respect to some basis. This has a unique solution for any right hand side, unless the corresponding homogeneous problem has a non-trivial solution.

Suppose that a polynomial $p \in \mathcal{P}_m$ comes from such a solution of the homogeneous problem, i.e., $p^{(j)}(x_i) = 0$, $i = 1:n$, $j = 0:r_i - 1$. Then, x_i must be a zero of multiplicity r_i of $p(x)$, hence $p(x)$ must have at least $\sum r_i = m$ zeros (counting the multiplicities), but this is impossible, because the degree of p is less than m. This contradiction proves the theorem. □

4.3.2 The Divided Difference Table in Multiple Points

An interpolation problem that contains multiple points is obtained by a passage to the limit from a case with (say) m distinct points. Due to the symmetry property of divided differences, we can permute the arguments, before we go to the limit, so that the arguments that make a group come together. We can therefore without loss of generality assume that equal arguments are placed together, and that the values x_i for different groups are different.

Assume that $f \in C^{r-1}$. By Theorem 4.2.3,

$$[x_1, x_2 \ldots x_r]f = f^{(r-1)}(\xi)/(r-1)!, \quad \xi \in \operatorname{int}(x_1, x_2, \ldots, x_r).$$

Now let $x_i \to x_1$, $i = 2:r$. Then $[x_1, x_2 \ldots x_r]f \to f^{(r-1)}(x_1)/(r-1)!$, so the natural *definition of a divided difference with r equal arguments* reads

$$[x_1, x_1 \ldots x_1]f = f^{(r-1)}(x_1)/(r-1)!, \quad r \text{ equal arguments.} \tag{4.3.3}$$

This definition and the usual recurrence formula for the divided differences are, under the above assumptions, sufficient for the construction of a table of divided differences in the case of multiple points, e.g.,

$$
\begin{aligned}
[x_0, x_0]f &= \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(x_0),\\
[x_0, x_0, x_1]f &= \frac{[x_0, x_0]f - [x_0, x_1]f}{x_0 - x_1} = \frac{f'(x_0) - [x_0, x_1]f}{x_0 - x_1}.
\end{aligned}
$$

It can be shown that if $f \in C^k$, the divided differences belong to $C^{k+1-\max r_i}$, and that the interpolation polynomial has this kind of differentiability with respect to the x_i, nota bene if the "groups" do not coalesce further.

Example 4.3.1. To interpolate the function f and its first derivative f' at the two points x_0 and x_1, and also its second derivative at x_0 we construct the generalized divided-difference table, where $x_1 \neq x_0$.

$$
\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & f_0' & & & \\
x_0 & f_0 & & \frac{1}{2}f_0'' & & \\
 & & f_0' & & [x_0, x_0, x_0, x_1]f & \\
x_0 & f_0 & & [x_0, x_0, x_1]f & & [x_0, x_0, x_0, x_1, x_1]f \\
 & & [x_0, x_1]f & & [x_0, x_0, x_1, x_1]f & \\
x_1 & f_1 & & [x_0, x_1, x_1]f & & \\
 & & f_1' & & & \\
x_1 & f_1 & & & &
\end{array}
$$

The interpolating polynomial now reads

$$
\begin{aligned}
p^*(x) &= f_0 + f_0'(x - x_0) + \frac{1}{2} f_0''(x - x_0)^2\\
&\qquad + [x_0, x_0, x_0, x_1]f(x - x_0)^3 + [x_0, x_0, x_0, x_1, x_1]f(x - x_0)^3(x - x_1).\\
f(x) - p^*(x) &= [x_0, x_0, x_0, x_1, x_1, x]f(x - x_0)^3(x - x_1)^2\\
&= f^{(5)}(\xi_x)(x - x_0)^3(x - x_1)^2/5!
\end{aligned}
$$

For the simplest Hermite interpolation problem, i.e., **cubic interpolation** and interpolation in $\mathcal{P}_4$, the given data are $f_i = f(x_i)$, $f_i' = f'(x_i)$, $i = 0, 1; x_1 = x_0 + h$. Set $x = x_0 + th$, and denote the remainder $f(x) - p^*(x)$ by R_T. One can show (Problem 1) that

$$
\begin{aligned}
p^*(x_0 + th) &= f_0 + \Delta f_0 + t(1 - t)(\Delta f_0 - hf'f_0) - t(1 - t)t(hf_1' - 2\Delta f_0 + hf_0')\\
&= (1 - t)f_0 + tf_1 + t(1 - t)\Big[(1 - t)(hf_0' - \Delta f_0) - t(hf_1' - \Delta f_0)\Big],\\
R_T &= \frac{h^4}{4!} t^2(t - 1)^2 f^{(4)}(\xi),
\end{aligned}
$$

We get the error bound

$$|R_T| \le \frac{h^4}{384} \max_{x\in[x_0,x_1]} |f^{(4)}(x)|. \tag{4.3.4}$$

In particular, putting $p = 1/2$, we get the useful formula

$$f_{1/2} = \frac{1}{2}(f_0 + f_1) + \frac{1}{8}h(f_0' - f_1') + R_T. \tag{4.3.5}$$

4.3.3 Other Interpolation Problems with Derivatives

Sometimes there are gaps in the sequence of derivatives that are numerically known at a point. The problem is then called **Birkhoff interpolation** or lacunary interpolation. We illustrate by two examples that such problems can either have a unique solution or lead to a singular system of linear equations. See also Problems. We use the notation of Sec. 4.1.1.

Example 4.3.2. Given $\tilde{f} = \big(f(-1), f'(0), f(1)\big)^T$. Try to find a polynomial $p \in \mathcal{P}_3$ that satisfies such data. The new feature is that $f(0)$ is missing.

Set up (4.1.3), i.e., $M_p c = \tilde{f}$, in the power basis.

$$M_p = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

The determinant is evidently zero, so *there is no solution* for most data. An explanation is that $hf' = \mu\delta f$ for all $f \in \mathcal{P}_3$.

Example 4.3.3. Given $\tilde{f} = \big(f(1), f(-1), f''(1), f''(-1)\big)^T$. Try to find a polynomial $p^* \in \mathcal{P}_4$ that satisfies such data. The new feature is that there are no first derivatives. In this case, we obtain for the power basis,

$$M_p = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 2 & -6 \end{pmatrix}.$$

The determinant is 48, and *this interpolation problem is uniquely solvable.* The coefficient vector of p^* is $c = M_p^{-1}\tilde{f}$.

$e_1^T c$, i.e., the 1st coordinate of c, is an approximation to $f(0)$; this is also a linear functional of f. Denote by Rf the remainder functional for this approximation, i.e.,

$$Rf = f(0) - e_1^T M_p^{-1} \tilde{f}. \tag{4.3.6}$$

Example 4.3.4. *Asymptotic error estimation for an approximation to a linear functional.* We shall now find an *approximate* estimate of the remainder functional

Rf, defined in (4.3.6). The main purpose of this example is to present a technique that is simple and generally applicable to all linear functionals Rf such that $Rp = 0, \forall p \in \mathcal{P}_m$. There exist also methods to obtain *rigorous* bounds for Rf, see Chapter 12, but they are more complicated.

In this example $m = 4$. Denote by $T_4 f$ the sum of the first four terms of the Maclaurin series of $f(x)$ into powers of x, and insert the basis notations $p_5(x) \equiv x^4$, $p_6(x) \equiv x^5$ etc. By Taylor's formula

$$f(x) = T_4 f(x) + \frac{f^{(4)}(0)}{4!} p_5(x) + \frac{f^{(5)}(0)}{5!} p_6(x) + \dots.$$

Here $RT_4 f = 0$, because $T_4 f \in \mathcal{P}_4$. Hence

$$Rf = \frac{f^{(4)}(0)}{4!} Rp_5 + \frac{f^{(5)}(0)}{5!} Rp_6 + \dots,$$

By (4.3.6),

$$Rp_5 = e_1^T M_p^{-1} \tilde{p}_5 = (1,0,0,0) M_p^{-1} (1,1,12,12)^T = 5.$$

after the solution of a linear system. (Rp_6 can be computed similarly, but we do not need this below.)

We now modify the problem a little, so that it becomes more like typical applications. Set $f(x) = F(hx)$, where h is some small step size parameter. Then the given data are $F(h)$, $f(-h, h^2 F''(h)$, $h^2 F''(-h)$, and $f^{(4)}(0) = h^4 F^{(4)}(0)$, $Rf = R_h F$, etc. and we finally obtain the **asymptotic error estimate** (recall that $Rp_5 = 5$),

$$R_h F \sim \frac{5}{24} h^4 F^{(4)}(0), \qquad (h \to 0).$$

4.3.4 Leibniz Formula for Differences

Another property of divided differences will be needed in developing spline functions in Sec. **??**.

Theorem 4.3.2. *A Difference Analogue to the Leibniz formula*
Let $f(x) = g(x)h(x)$, and $x_i \le x_{i+1} \le \ldots \le x_{i+k}$. Then

$$[x_i, \dots, x_{i+k}]f = \sum_{r=i}^{i+k} [x_i, \dots, x_r]g \cdot [x_r, \dots, x_{i+k}]h, \tag{4.3.7}$$

provided that $g(x)$ and $f(x)$ are sufficiently many times differentiable so that the divided differences on the right hand side are defined for any coinciding points x_j.

Proof: Note that the product polynomial

$$
\begin{aligned}
P(x) = &\sum_{r=i}^{i+k} (x-x_i)\cdots(x-x_{r-1})[x_i,\dots,x_r]g \\
&\cdot \sum_{s=i}^{i+k} (x-x_{s+1})\cdots(x-x_{i+k})[x_s,\dots,x_{i+k}]h
\end{aligned}
$$

agrees with $f(x)$ at $x_i,\dots,x_{i+k}$ since by Newton's interpolation formula the first factor agrees with $g(x)$ and the second with $h(x)$ there. If we multiply out we can write $P(x)$ as a sum of two polynomials

$$P(x) = \sum_{r,s=i}^{i+k} \dots = \sum_{r\le s} \dots + \sum_{r>s} \dots = P_1(x) + P_2(x).$$

Since in $P_2(x)$ each term in the sum has $\prod_{j=i}^{i+k}(x-x_j)$ as a factor it follows that $P_1(x)$ will also interpolate $f(x)$ at $x_i,\dots,x_{i+k}$. The theorem now follows since the leading coefficient of $P_1(x)$, which equals $\sum_{r=i}^{i+k}[x_i,\dots,x_r]g\cdots[x_r,\dots,x_{i+k}]h$, must equal the leading coefficient of the unique interpolation polynomial of degree k, which is $[x_i,\dots,x_{i+k}]f$. ◻

Review Questions

1. What is meant by Hermite interpolation (osculatory interpolation) ? Prove the uniqueness result for the Hermite interpolation problem.

Problems

1. (a) Construct the divided difference scheme (unscaled or scaled) for the simplest Hermite interpolation problem, where the given data are $f(x_i)$, $f'(x_i)$, $i = 0, 1$; $x_1 = x_0 + h$. Prove all the formulas concerning this problem that are stated at the end of Sec. 4.3.2.

   (b) For $f(x) = (1+x)^{-1})$, $x_0 = 1$, $x_1 = 1.5$, compute by Hermite interpolation $f(1.25)$. Compare the error bound and the actual error.

   (c) Show that

$$|f'(x) - \hat{p}'(x)| \le \frac{h^3}{72\sqrt{3}}\Big(\max_{x\in[x_0,x_1]} |f^{(iv)}(x)| + O(h|f^{(v)}(x)|\Big).$$

Hint: $\frac{d}{dx}[x_0,x_0,x_1,x_1,x]f = [x_0,x_0,x_1,x_1,x,x]f \le \dots$.

2. Given $x_i, y(x_i), y'(x_i)$, $x_i = x_0 + ih$, $i = 1, 2, 3$. Let $\hat{p} \in \mathcal{P}_6$ be the Hermite interpolation polynomial to these data.

(a) Find the remainder term, and show that the interpolation error for $x \in [x_1, x_3]$ does not exceed $\frac{h^6 \max|f^{(6)}(x)|}{4860}$ in magnitude.

(b) Write a program that computes $\hat{p}(x_1 + 2jh/k)$, $j = 0 : k$.

COMMENT: This is one of several possible procedures for starting a multistep method for an ordinary differential equation $y' = f(x, y)$. Two steps with an accurate one-step method, provide values of y, y', and this program then produces starting values (y only) for the multistep method.

3. Derive the usual formula of Leibniz for the k'th derivative from (4.3.7) by a passage to the limit.

4. Give a short and complete proof of the uniqueness of the interpolation polynomial for distinct points, by the use of the ideas of the proof of Theorem 4.3.1.

5. Modify the integration formula in Example 4.1.1 to a formula for $\int_0^h x^{-1/2} f(x)\,dx$, and derive an asymptotic error estimate ($h \to 0$), by means of the technique of Example 4.3.4.

6. (a) Derive an *asymptotic error estimate* for one step of length h with the midpoint rule, $\int_{-h/2}^{h/2} f(x)\,dx \approx hf(0)$. Derive also a *strict local error bound*, by integrating a Taylor expansion of $f(x)$ with remainder, on the assumption that $|f''(x)| < M$.

(b) Derive an *asymptotic global error estimate* for the trapezoidal rule over the interval $[a, b]$, with step size $h = (b - a)/n$, $n \to \infty$.

Hint: $\sum_{i=1}^n hf''(x_i) \to \int_a^b f''(x)\,dx$, etc.

(c) Derive also a *strict global error bound* on the assumption that $|f''(x)| < M$ for $x \in [a, b]$. Compare these results with results that can be derived from the analysis of the Euler-Maclaurin formula.

Hint: Recall the relation of the midpoint rule (rectangle rule) to the trapezoidal rule that was mentioned in Sec.3.3.

4.4 Spline Functions

4.4.1 Introduction

Before the computer age ship builders and others in engineering design used a **spline** to draw smooth curves. A spline is a thin elastic ruler, which can be bent so that it passes trough a given set of points, see Fig. 4.4.1.

The curvature of a spline $y = s(x)$, $x \in [a, b]$ in the plane is given by

$$\kappa(x) = \frac{s''(x)}{(1 + (s'(x))^2)^{3/2}}.$$

By Hamilton's principle the shape the spline will take is such that the strain energy

$$\int_a^b \kappa(x)^2\,dx \approx const \cdot \int_a^b s''(x)^2\,dx,$$

is minimized, where the approximation holds for slowly varying deflections, i.e., when $(s'(x))^2$ is approximately constant. Under this assumption, according to

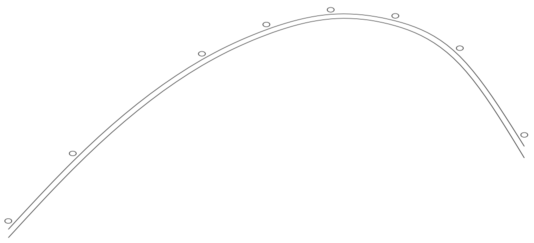


Figure 4.4.1. *The original spline.*

elasticity theory, $s(x)$ is built up of **piecewise** third degree polynomials (cubic polynomials) in such a way that $s(x)$ *and its two first derivatives are everywhere continuous.* Let x_i, $i = 0 : m$ be the points the spline is forced to interpolate. Then the third derivative can have discontinuities at the points x_i. Such a function is called a **cubic spline function**, or shorter, a **cubic spline** The points x_i, $i = 0 : m$, are called **breakpoints** or **knots**.

The mathematical concept of spline functions was introduced by Schoenberg [19] in 1946. In computer aided design (CAD), where curves and surfaces have to be represented mathematically, so that they can be manipulated and visualized easily, spline functions are now used extensively. Important applications occur in computer-aided design, analysis and manufacturing. in the aircraft and automotive industries.

With the use of splines, there is no reason to fear equidistant data, as opposed to the situation with higher-degree polynomials. Also, if the function to be approximated is badly behaved somewhere then, using spline approximation with properly chosen knots, the effect of this can be confined locally, allowing good approximation elsewhere in the interval.

Spline functions can also be used in the numerical treatment of boundary-value problems for differential equations, and for **surface fitting**. In the following we restrict ourself to consider curves in the plane. For more information on spline approximations of curves and surfaces the reader is referred to de Boor [3], where also FORTRAN programs for computations with spline functions can be found, and and Dierckx [8]. A more geometric view is taken in Farin [10]. Several packages exist for computing with splines; see, e.g., the spline toolbox in MATLAB and FITPACK Dierckx [6]–[7] available through netlib.

4.4.2 Piecewise Affine and Cubic Interpolation

We have seen that it is often not efficient to approximate a given function by a single polynomial over its entire range. On the other hand, polynomials of low degree can give good approximations *locally* in a small interval. Therefore it is natural to consider approximations by piecewise polynomials of different degrees of global continuity.

We first consider the simplest case of interpolating given values $y_i = f(x_i)$ on a grid

$$\Delta = \{a = x_0 < x_1 < \cdots < x_m = b\}$$

in the interval $[a, b]$ by a **piecewise affine function** $s(x)$. This interpolating function is uniquely determined and equal to the broken line

$$q_i(x) = y_{i-1} + d_i(x - x_i), \quad x \in [x_{i-1}, x_i), \quad i = 1 : m, \tag{4.4.1}$$

where

$$h_i = x_i - x_{i-1}, \qquad d_i = [x_{i-1}, x_i]f(x) = (y_i - y_{i-1})/h_i. \tag{4.4.2}$$

i.e. d_i is the divided difference of f at $[x_{i-1}, x_i]$. If $f \in C^2[a, b]$ then the error satisfies (see Example 4.2.4)

$$|f(x) - s(x)| \le \frac{1}{8} \max_i \Big(h_i^2 \max_{x \in [x_{i-1}, x_i]} |f''(x)|\Big). \tag{4.4.3}$$

Hence, we can make the error arbitrary small by decreasing $\max_i h_i$. An important property of interpolation with a piecewise affine function is that it preserves monotonicity and convexity of the interpolated function.

The broken line interpolating function has a discontinuous first derivative at the knots, which makes it unsuitable for many applications. To get better smoothness piecewise polynomials of higher degree need to be used. Although piecewise quadratic approximation is sometimes useful, piecewise *cubic* polynomials with continuous second derivatives are by far the more important (see Figure 4.4.2.

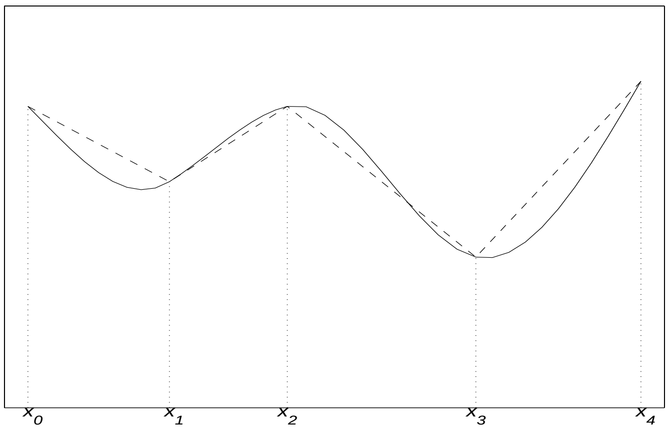


Figure 4.4.2. *Broken line and cubic spline interpolation.*

A cubic polynomial $q_i(x)$ on the interval $[x_{i-1}, x_i)$ is uniquely determined by the values of the function and its first derivative at the end points of the interval. This follows from the more general result on Hermite interpolation in Theorem 4.3.1. By (4.3.4), translated to the notation in (4.4.2), the cubic $q_i(x)$ can be written in the form

$$q_i(x) = ty_i + (1-t)y_{i-1} + h_i t(1-t)\left[(k_{i-1}-d_i)(1-t) - (k_i - d_i)t\right], \quad i = 1:m, \tag{4.4.4}$$

where h_i, d_i are as in (4.4.2), t is a local variable

$$t = \frac{x - x_{i-1}}{h_i} \in [0,1), \quad x \in [x_{i-1}, x_i), \tag{4.4.5}$$

and $k_j = q_i'(x_j)$, $j = i-1, i$, is the derivative of the spline function.

If the interpolating spline $s(x)$ is to be evaluated at many points, a form more efficient to use than (4.4.4) is the piecewise polynomial (pp) form

$$q_i(x) = y_{i-1} + a_{1i}(x - x_{i-1}) + a_{2i}(x - x_{i-1})^2 + a_{3i}(x - x_{i-1})^3. \tag{4.4.6}$$

From (4.4.4) we obtain after some calculation

$$\begin{aligned} a_{1i} &= q_i'(x_{i-1}) = k_{i-1}, \quad a_{2i} = \tfrac{1}{2}q_i''(x_{i-1}) = (3d_i - 2k_{i-1} - k_i)/h_i, \\ a_{3i} &= \tfrac{1}{6}q_i'''(x_{i-1}) = (k_{i-1} + k_i - 2d_i)/h_i^2. \end{aligned} \tag{4.4.7}$$

Using Horner's scheme $q_i(x)$ can be evaluated from (4.4.6) using only four multiplication.

With piecewise cubic polynomials we can interpolate given function values and first derivatives on the grid Δ. By construction the interpolating piecewise cubic function $s(x)$ will have continuous first derivatives. If $f \in C^4[a,b]$ then it follows from the remainder term (4.3.4) that the error satisfies

$$|f(x) - s(x)| \le \frac{1}{384} \max_i \Big(h_i^4 \max_{x \in [x_{i-1}, x_i]} |f^{(iv)}(x)| \Big). \tag{4.4.8}$$

It can be shown (Problem 1c of Sec. 4.3) that also the first derivative of $s(x)$ is a good approximation to $f'(x)$. If $f \in C^5[a,b]$ we have

$$|f'(x) - s'(x)| \le \frac{1}{72\sqrt{3}} \max_i \Big(h_i^3 \max_{x \in [x_i, x_{i+1}]} |f^{(iv)}(x) + O(h_i f^{(5)}(x))| \Big). \tag{4.4.9}$$

Sometimes it is useful to consider the values k_i, $i = 0:m$, as parameters which are used to give the interpolating function the desired shape. In the next section we show that it is possible to choose these parameters such that the interpolating function $s(x)$ also has a continuous *second* derivative.

We shall now formally define a spline function of order $k \ge 1$.

Definition 4.4.1.

Let $\Delta = \{a = x_0 < x_1 < \cdots < x_m = b\}$ be a subdivision of the interval $[a,b]$. A spline function on Δ of order $k \ge 1$ (degree $k-1 \ge 0$), is a real function s with properties:

(a) *For $x \in [x_i, x_{i+1}]$, $i = 0:m-1$, $s(x)$ is a polynomial of degree $< k$.*

(b) *For $k = 1$, $s(x)$ is a piecewise constant function. For $k \geq 2$, $s(x)$ and its first $k-2$ derivatives are continuous on $[a, b]$, i.e., $s(x) \in C^{k-2}[a, b]$.*

We denote by $S_{\Delta,k}$ the set of all spline functions of order k on Δ. From the definition it follows that if $s_1(x)$ and $s_2(x)$ are spline functions of the same degree, so is $c_1 s_1(x) + c_2 s_2(x)$. Thus $S_{\Delta,k}$ is a *linear space.*

Examples of elements of $S_{\Delta,k}$ are the **truncated power** functions

$$(x - x_j)_+^{k-1}, \quad j = 1 : m-1,$$

and all their linear combinations. Moreover, $\mathcal{P}_k$ is linear subspace of $S_{\Delta,k}$.[10] Conversely, together these functions span $S_{\Delta,k}$. All we need for the first subinterval is a basis of $\mathcal{P}_k$, e.g., the power basis $\{1, x, \ldots, x^{k-1}\}$. Further, all we need for each additional subinterval $[x_j, x_{j+1})$, $j = 1 : m-1$, is the new basis function $(x - x_j)_+^{k-1}$. One can show that these $k + m - 1$ functions are linearly independent. The dimension of the linear space thus is $k + m - 1$.

4.4.3 Cubic Spline Interpolation

In the following we shall first study cubic spline functions which *interpolate* a given function $f(x)$ at the grid Δ, i.e., the space $S_{\Delta,4}$. By definition a cubic spline consists of cubic polynomials pieced together in such a way that their values and first *two* derivatives coincide at the knots. In contrast to Hermite interpolation, the cubic polynomial in each subinterval will now depend on *all data points.*

Theorem 4.4.2.

Every cubic spline function, with knots $a = x_0 < x_1 < \cdots < x_m = b$, which interpolates the function $y = f(x)$,

$$s(x_i) = f(x_i) = y_i, \quad i = 0 : m,$$

equals for $x \in [x_{i-1}, x_i)$, $i = 1 : m$ a third degree polynomial of the form (4.4.4). *The $m+1$ parameters k_i, $i = 0 : m$, satisfy $m-1$ linear equations*

$$h_{i+1} k_{i-1} + 2(h_i + h_{i+1}) k_i + h_i k_{i+1} = 3(h_i d_{i+1} + h_{i+1} d_i), \qquad (4.4.10)$$
$$i = 1 : m-1,$$

where $h_i = x_i - x_{i-1}$, $d_i = (y_i - y_{i-1})/h_i$.

Proof: We require the second derivative of the spline $s(x)$ to be continuous at x_i, $i = 1 : m-1$. We have

$$s(x) = \begin{cases} q_i(x), & x \in [x_{i-1}, x_i), \\ q_{i+1}(x), & x \in [x_i, x_{i+1}), \end{cases}$$

[10] Recall the notation $(x - u)_+^j = \max\{x - u, 0\}$ that was introduced in Sec. 3.2.3 in connection with the Peano kernel.

where $q_i(x)$ is given by (4.4.6)–(4.4.7). Differentiating twice we get $\frac{1}{2}q_i''(x) = a_{2,i} + 3a_{3,i}(x - x_{i-1})$, and letting $x \to x_i$

$$\tfrac{1}{2}q_i''(x_i) = a_{2,i} + 3a_{3,i}h_i = (k_{i-1} + 2k_i - 3d_i)/h_i.$$

Replacing i by $i+1$ we get

$$\tfrac{1}{2}q_{i+1}''(x_i) = a_{2,i+1} = (3d_i - 2k_i - k_{i+1})h_{i+1}.$$

These last two expressions must be equal, which gives the conditions

$$\frac{1}{h_i}(k_{i-1} + 2k_i - 3d_i) = \frac{1}{h_{i+1}}(3d_{i+1} - 2k_i - k_{i+1}), \quad i = 1 : m-1. \tag{4.4.11}$$

Multiplying both sides by $h_i h_{i+1}$ we get (4.4.10). □

The conditions (4.4.10) are $(m-1)$ linearly independent[11] equations for the $(m+1)$ unknowns k_i, $i = 0 : m$. Two additional conditions are therefore needed to uniquely determine the interpolating spline. The four most important choices are discussed below.

(i) If the derivatives at the end points are known we can take

$$k_0 = f'(a), \qquad k_m = f'(b). \tag{4.4.12}$$

The corresponding spline function $s(x)$ is called the **complete cubic spline interpolant**. If k_0 and k_m are determined by numerical differentiation with a truncation error that is $O(h^4)$, we call the spline interpolant **almost complete**. For example, k_m may be the sum of (at least) four terms of the expansion $Df(x_m) = -\frac{1}{h}\ln(1-\nabla)y_m$ into powers of the operator ∇; see Example 3.2.6. Similarly k_0 may be the sum of four terms of the expansion $Df(x_0) = \frac{1}{h}\ln(1+\Delta)y_0$ into powers of the operator Δ; see (3.2.20).[12]

(ii) A physical spline is straight outside the interval $[a, b]$, i.e. $s''(x) = 0$ for $x \le a$ or $x \ge b$. Thus $q_1''(x_0) = q_m''(x_m) = 0$. From (4.4.7) we obtain

$$\tfrac{1}{2}q_i''(x_{i-1}) = a_{2i} = (3d_i - 2k_{i-1} - k_i)/h_i.$$

Setting $i = 1, m$ gives the two conditions

$$\begin{aligned} 2k_0 + k_1 &= 3d_1 \\ k_{m-1} + 2k_m &= 3d_m. \end{aligned} \tag{4.4.13}$$

The corresponding approximating spline is called the **natural spline interpolant**. It should be stressed that when a cubic spline is used for the approximation of a smooth function, these boundary conditions are *not natural*!

[11] The equations are strictly row diagonally dominant (see Sec. 7.4.1) and therefore linearly independent

[12] Two terms of the *central* difference expansion in (3.2.43) or one *Richardson* extrapolation, see (3.3.20), give higher accuracy, but need extra function values outside the grid Δ.

(iii) If the end point derivatives are not known, a convenient condition is to require that $s'''(x)$ be continuous across the first and last interior knots x_1 and x_{m-1}. Hence $q_1(x) = q_2(x)$ and $q_{m-1}(x) = q_m(x)$. Then x_1 and x_{m-1} are no longer knots, and these conditions are known as "**not a knot**" conditions. From (4.4.7) we obtain,

$$\tfrac{1}{6}q_i'''(x) = a_{3i} = (k_{i-1} + k_i - 2d_i)/h_i^2, \quad x \in [x_{i-1}, x_i), \quad i = 1 : m.$$

Hence the condition $q_1''' = q_2'''$ gives $(k_0 + k_1 - 2d_1)/h_1^2 = (k_1 + k_2 - 2d_2)/h_2^2$, or

$$h_2^2 k_0 + (h_2^2 - h_1^2)k_1 - h_1^2 k_2 = 2(h_2^2 d_1 - h_1^2 d_2).$$

Since this equation would destroy the tridiagonal form of the system, we use (4.4.10), for $i = 1$ to eliminate k_2. This gives the equation

$$h_2 k_0 + (h_2 + h_1)k_1 = 2h_2 d_1 + \frac{h_1(h_2 d_1 + h_1 d_2)}{h_2 + h_1}. \tag{4.4.14}$$

If the right boundary condition is treated similarly we get

$$(h_{m-1} + h_m)k_{m-1} + h_{m-1}k_m = 2h_{m-1}d_m + \frac{h_m(h_{m-1}d_m + h_m d_{m-1})}{h_{m-1} + h_m}. \tag{4.4.15}$$

(iv) If the spline is used to represent a periodic function, then $y_0 = y_m$ and the boundary conditions

$$s'(a) = s'(b), \qquad s''(a) = s''(b), \tag{4.4.16}$$

suffice to determine the spline uniquely. From the first condition it follows that $k_0 = k_m$, which can be used to eliminate k_0 in the equation (4.4.10) for $k = 1$. The second condition in (4.4.16) gives using (4.4.11) $(k_0 + 2k_1 - 3d_1)/h_1 = -(2k_{m-1} + k_m - 3d_m)/h_m$, or after eliminating k_0,

$$2h_m k_1 + 2h_1 k_{m-1} + (h_1 + h_m)k_m = 3(h_m d_1 + h_1 d_m),$$

The spline interpolant has the following best approximation property.

Theorem 4.4.3.

Among all functions g which are twice continuously differentiable on $[a, b]$ and which interpolate f at the points $a = x_0 < x_1 < \cdots < x_m = b$, the natural spline function minimizes

$$\int_a^b \big(s''(t)\big)^2 dt.$$

The same minimum property holds for the complete spline interpolant, if the functions g satisfy $g'(a) = f'(a)$, and $g'(b) = f'(b)$.

PROOF: See de Boor [3, 1978, Chapter 5]. □

Due to this property spline functions yield smooth interpolation curves, except for rather thin oscillatory layers near the boundaries if the "natural" boundary conditions $s''(a) = s''(b) = 0$ are far from being satisfied. For the complete or almost complete cubic spline and for cubic splines determined by the "not-a-knot" conditions, these oscillations are much smaller; see Sec. 4.4.4.[13]

Equations (4.4.10) together with any of these boundary conditions give rise to a well-conditioned system of linear equations for determining the derivatives k_i. For the first three boundary conditions the system is **tridiagonal** Such systems can be easily solved by Gaussian elimination without pivoting. (Methods for solving general banded linear systems will be studied in more detail in Sec. 6.4.)

Consider the tridiagonal system of linear equations $Tx = g$, where

$$T = \begin{pmatrix} b_1 & c_1 & & & \\ a_1 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & a_{n-1} & b_n \end{pmatrix}. \tag{4.4.17}$$

The following algorithm performs Gaussian elimination without pivoting on T and computes the solution x:

Put $\beta_1 = b_1$, and compute recursively

$$\gamma_k = a_k/\beta_k, \qquad \beta_{k+1} = b_{k+1} - \gamma_k c_k, \quad k = 1 : n-1.$$

Compute the vector y and the solution x by

$$\begin{aligned} &y_1 = g_1, \qquad y_{k+1} = g_{k+1} - \gamma_k y_k, \quad k = 1 : n-1, \\ &x_n = y_n/\beta_n, \qquad x_k = (y_k - c_k x_{k+1})/\beta_k, \quad k = n-1 : -1 : 1. \end{aligned}$$

This algorithm requires only about $5n$ multiplications and additions.

It can be proved that a sufficient condition for this algorithm to be stable is that T is **diagonally dominant**, i.e.,

$$|b_1| > |c_1|, \quad |b_k| > |a_{k-1}| + |c_k|, \quad k = 2 : n-1, \quad |b_n| > |a_{n-1}|.$$

It is also stable for the system resulting from the not-a-knot boundary condition although this is not diagonally dominant in the first and last row; see Problem 2b.

Example 4.4.1. In the case of spline interpolation with constant stepsize $h_i = h$ equation (4.4.10) becomes

$$k_{i-1} + 4k_i + k_{i+1} = 3(d_i + d_{i+1}), \quad i = 1 : m-1. \tag{4.4.18}$$

The "not a knot" boundary conditions (4.4.14)–(4.4.15) become

$$k_0 + 2k_1 = \tfrac{1}{2}(5d_1 + d_2), \qquad 2k_{m-1} + k_m = \tfrac{1}{2}(d_{m-1} + 5d_m). \tag{4.4.19}$$

[13]When a spline is to be used for the approximate representation of a smooth function, the natural spline is *not* a natural choice.

We obtain a tridiagonal system $Tk = g$, where,

$$\begin{pmatrix} 1 & 2 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 2 & 1 \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_{m-1} \\ k_m \end{pmatrix} = 3 \begin{pmatrix} (5d_1 + d_2)/6 \\ d_1 + d_2 \\ \vdots \\ d_{m-1} + d_m \\ (d_{m-1} + 5d_m)/6 \end{pmatrix}.$$

except for the first and last row, the elements of T are constant along the diagonals. The condition number of T increases very slowly with m; for example, $\kappa(T) < 16$ for $m = 100$.

Consider now the periodic boundary conditions in (iv). Setting $k_m = k_0$ in the last equation we obtain a linear system of equations $Tk = g$ for $k_1, \dots, k_{m-1}$ where

$$T = \left(\begin{array}{cccccc|c} b_1 & c_1 & & & & & a_m \\ a_1 & b_2 & c_2 & & & & 0 \\ & \ddots & \ddots & \ddots & & & \vdots \\ & & a_{m-3} & b_{m-2} & c_{m-2} & & 0 \\ & & & a_{m-2} & b_{m-1} & & c_{m-1} \\ \hline c_m & 0 & \cdots & 0 & a_{m-1} & & b_m \end{array}\right). \tag{4.4.20}$$

Here T is tridiagonal except for its last row and last column, where an extra nonzero elements occur. Such systems, called **arrowhead system**, can be solved with about twice the work of a tridiagonal system; see further Chapter 7.

In some applications one wants to smoothly interpolate given points (x_j, y_j), $j = 0 : m$, where a representation of the form $y = f(x)$ is not suitable. Then we can use a **parametric spline** representation $x = x(\theta)$, $y = y(\theta)$, where the parameter values $0 = \theta_0 \le \theta_1 \le \cdots \le \theta_m$ correspond to the given points. Using the approach described previously two spline functions $s_x(t)$ and $s_y(t)$ can then be determined, that interpolate the points (θ_i, x_i) and (θ_i, y_i), $i = 0 : m$, respectively. The parametrization is usually chosen as $\theta_i = d_i/d$, $i = 1 : m$, where $d_0 = 0$,

$$d_i = d_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = 1 : m.$$

are the the cumulative distance and $d = \sum_{j=1}^m d_j$.

For boundary conditions we have the same choices as mentioned previously. In particular, using periodic boundary conditions for $s_x(t)$ and $s_y(t)$ allows the representation of *closed curves* (see Problem 5).

4.4.4 Error in Cubic Spline Interpolation

We will now derive estimates of the error in cubic spline interpolation of a function with good smoothness properties, $f \in C^5$ (say). Let $x \in I_i = [x_{i-1}, x_i]$, and set

$$t = (x - x_{i-1})/h_i, \quad y_i = f(x_i), \quad y_i' = f'(x_i).$$

The error can be expressed as the sum of two components:

i. The error $E_H(x)$ due to Hermite interpolation with correct values of $f'(x_{i-1})$, $f'(x_i)$.

ii. The error $E_S(x)$ due to the errors of the slopes $e_i = k_i - y'_i$, $i = 0 : m$.

We shall see that the first part is typically the dominant part. For the error $E_H(x)$ we have from equations (4.4.8)–(4.4.9)

$$\max_{x\in I_i} |E_H(x)| \le \frac{1}{384} \max_{x\in I_i} |h_i^4 f^{(iv)}(x)|, \tag{4.4.21}$$

By (4.4.4) the second part of the error is

$$E_S(x) = h_i t(1-t)\big[e_{i-1}(1-t) - e_i t\big], \quad x = x_{i-1} + th_i, \quad t \in [0,1).$$

Since $|1-t| + |t| = 1$, it follows easily that

$$|E_S(x)| \le \frac{1}{4} \max_{1\le i\le m} |h_i e_j|, \quad j = i-1, i. \tag{4.4.22}$$

We shall estimate $|e_j|$ in the case of *constant step size.* Set

$$l_i = 3(d_i + d_{i+1}) - (y'_{i-1} + 4y'_i + y'_{i+1}), \qquad i = 1 : m-1.$$

Then by (4.4.10) $(e_1, \dots, e_{m-1})$ satisfies

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{m-2} \\ e_{m-1} \end{pmatrix} = \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_{m-2} \\ l_{m-1} \end{pmatrix} - \begin{pmatrix} e_0 \\ 0 \\ \vdots \\ 0 \\ e_m \end{pmatrix},$$

or $Ae = l - b$. We write $e = e_I - e_B$, where $Ae_I = l$, $Ae_B = b$. These two systems will be treated differently.

We first estimate e_I and note that, since the matrix A is diagonally dominant, we can use Lemma 6.4.1 to obtain[14]

$$\max_{1\le i<m} |e_{I,i}| \le \frac{1}{\alpha} \max_{1\le i<m} |l_i|, \quad \text{where} \quad \alpha = \min_i \Big(|a_{ii}| - \sum_{j\ne i} |a_{ij}|\Big) = 2.$$

In order to estimate $\max_{1\le i<m} |l_i|$, note that the defining relation for l_i can be rewritten as

$$\frac{h}{3} l_i = y_{i+1} - y_{i-1} - \frac{h}{3}(y'_{i-1} + 4y'_i + y'_{i+1}).$$

The right hand side here equals the *local error of Simpson's formula* for computing the integral of y' over the interval $[x_{i-1}, x_{i+1}]$, which according to Problem 3.2.12 approximately is $h^5 f^{(5)}(x_i)/90$. It follows that[15]

$$\max_{1\le i<m} |e_{I,i}| \le \frac{1}{60} \max_i h_i^4 |f^{(5)}(x_i)|.$$

[14]This is typically an overestimate, almost by a factor of 3, see Problem 3.3.37.

[15]Notice that, if $f \in \mathcal{P}_5$, the slopes k_i becomes exact in complete cubic splines interpolation.

By (4.4.22) this shows that the contribution of e_I to E_S is $O(h^5)$ if $f \in C^5$, while $E_H(x)$ is $O(h^4)$. For complete splines $e_0 = e_m = 0$, and for almost complete splines $e_0 = O(h^4)$, $e_m = O(h^4)$. Hence $e_{B,i} = O(h^4)$, and its contribution to E_S is $O(h^5)$. So if h is sufficiently small, *the Hermite interpolation error is, in the whole interval* $[a, b]$*, asymptotically, the dominant source of error for complete and almost complete splines.*[16]

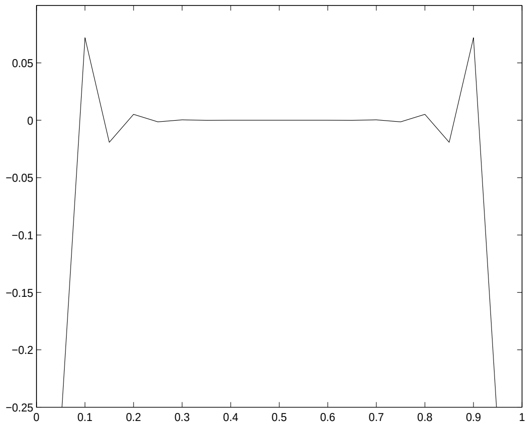


Figure 4.4.3. *Boundary slope errors* $e_{B,i}$ *for a cubic spline,* $e_0 = e_m = -1$*;* $m = 20$.

Similar conclusions seem to hold also in the case of variable step size, under the reasonable assumption that $h_{n+1} - h_n = O(h_n^2)$, see Sec. 13.1 (in particular Problem 11), where variable step size is discussed in the context of ordinary differential equations.

Finally we discuss the *effect of the boundary slope errors* for other boundary conditions. The equation $Ae_B = b$ can be written as a difference equation

$$e_{B,i+1} + 4e_{B,i} + e_{B,i-1} = 0, \quad i = 1 : m - 1.$$

see Sec. 3.4. One can show (Problem 3.4.5) that, for any boundary condition,

$$e_{B,i} \approx u^i e_0 + u^{m-i} e_m, \quad u = \sqrt{3} - 2 \approx -0.268,$$

if u^m is negligible. (Here u and u^{-1} are the roots of the characteristic equation $u^2 + 4u + 1 = 0$.)

Fig. 4.4.3 shows (for $m = 20$, $e_0 = e_m = -1$) how rapidly this error component dies out, e.g., $u^4 = 0.005$. At the midpoint $x = 0.5$ the error is $0.3816 \cdot 10^{-5}$.

If $m \gg 1$, $e_0 \neq 0$, and $e_m \neq 0$ it follows that e_B is negligible outside thin *oscillatory boundary layers* near $x = x_0$ and $x = x_m$. The height and thickness

[16]In the literature the usual (rigorous) error bound for a perfect spline, due to Hall and Meyer, is five times as large as the bound for the Hermite error. It is valid with $h = \max h_i$, independent of the position of the knots, for all $f \in C^4$, while we require $f \in C^5$.

of the layers depend on e_0 and e_m. We discuss the left boundary; the right one is analogous. Assume that

$$e_0 = e_{B,0} \neq 0, \qquad e_1 \approx e_{B,1} \approx ue_{B,0} \approx ue_0.$$

We then estimate e_0 by putting

$$k_0 = y_0' + e_0, \qquad k_1 = y_1' + e_1 \approx y_1' + ue_0,$$

into the boundary condition at x_0, i.e. the first equation of (4.4.13) for the natural splines and (4.4.14) for the "not a knot" splines. (Complete splines have no oscillatory boundary layers; $e_B = 0$.) The peak of the contribution of e_B to the spline interpolation error is then obtained by (4.4.4) for $i = 1$, and equals

$$he_0 \max_{0\le t\le 1} |t(1 - t(1 - t - ut)| \approx 0.17he_0. \tag{4.4.23}$$

For the *natural splines*, this procedure leads to

$$\begin{aligned}(2+u)e_0 &= 3\frac{1}{h}(y_1 - y_0) - 2y_0' - y_1' - O(h^4)\\ &= 3\Big(y_0' + \frac{1}{2}hy_0'' + \dots\Big) - 3y_0' - hy_0'' + \dots \sim \frac{1}{2}hy_0''.\end{aligned}$$

Since $2 + u = \sqrt{3}$, we obtain $e_0 \approx 0.29hy_0''$, and, by (4.4.23), the peak near $x = x_0$ becomes approximately $0.049h^2|y''|$, i.e. 40% of the *linear* interpolation error (instead of cubic), often clearly visible in a graph of $s(x)$.

For the "not a knot"-splines the procedure leads to

$$(1+2u)e_0 = \frac{5}{2h}(y_1 - y_0) + \frac{1}{2h}(y_2 - y_1) - y_0' - 2y_1' - O(h^4) \approx \frac{h^3}{12}y^{(4)};$$

see Problem 3.2.10. We thus obtain $e_0 \sim 0.180h^3y^{(4)}$, and hence by (4.4.23) the peak near x_0 becomes $0.031h^4y^{(4)}$, typically very much smaller than we found for natural splines. Still it is about 11.5 times as large as the Hermite interpolation error, but since the oscillations die out by the factor $u = 0.29$ in each step, we conclude that *the Hermite interpolation is the dominant error source in cubic "not a knot"-spline interpolation in (say) the interval* $[a + 3h, b - 3h]$.

For natural splines the boundary layers are much thicker, because the peaks are much higher.

Example 4.4.2.

For the function $f(x) = 1/(1 + 25x^2)$, $x \in [-1, 1]$, the maximum norm of the error is 0.022, in interpolation with a natural cubic spline function at the eleven equidistant points $x_i = -1 + 0.2i$, $i = 0 : 10$. This good result contrasts sharply with the unpleasant experience near the boundaries of interpolation with a tenth-degree polynomial shown in Fig. 4.2.6. An (almost) perfect cubic spline or a "not a knot"-spline gives even better results near the boundaries.

4.4.5 Approximation with B-Splines

It was shown in the beginning of Sec. 4.4.3 that the set of spline functions of order k, $S_{\Delta,k}$, on the grid

$$\Delta = \{a = x_0 < x_1 < \cdots < x_m = b\}$$

is a linear space of dimension $k + m - 1$. A basis was shown to be

$$\{1, x, \ldots, x^{k-1}\} \cup \{(x - x_1)_+^{k-1}, (x - x_2)_+^{k-1}, \ldots, (x - x_{m-1})_+^{k-1}\}, \qquad (4.4.24)$$

which is the **truncated power basis**.

Example 4.4.3. For $k = 2$ the space $S_{\Delta,k}$ consists of continuous piecewise affine (linear) functions also called linear splines. Then a basis is

$$\{1, x\} \cup \{l_1(x), \ldots, l_{m-1}(x)\}, \quad l_i(x) = (x - x_i)_+.$$

Another basis for $S_{\Delta,2}$ is obtained by introducing an artificial exterior knot $x_{-1} \leq x_0$. Then it is easy to see that using the functions $l_i(x)$, $i = -1 : m - 1$ every linear spline on $[x_0, x_m]$ can also be written as

$$s(x) = \sum_{i=-1}^{m-1} c_i l_i(x).$$

The truncated power basis has several disadvantages. The basis functions are not local; e.g., the monomial basis functions $1, x, \ldots, x^{k-1}$ are nonzero on the whole interval $[a, b]$. Also the basis functions (4.4.24) are almost linearly dependent when the knots are close. Therefore this basis yields an ill-conditioned linear systems for various tasks and is not suited for numerical computations. In the following we will construct a more satisfactory basis for $S_{\Delta,k}$.

In anticipation of the fact that it may be desirable to interpolate at other points than the knots we consider from now on the sequence of knots

$$\Delta = \{\tau_0 \leq \tau_1 \leq \cdots \leq \tau_m\}. \qquad (4.4.25)$$

where $\tau_i \leq \tau_{i+k}$, $i = 0 : m - k$, i.e., at most k successive knots are allowed to coincide.

We start by considering $k = 1$. The space $S_{\Delta,1}$ consists of piecewise constant functions. As a basis for $S_{\Delta,2}$ we can simply take the functions

$$N_{i,1}(x) = \begin{cases} 1 & x \in [\tau_i, \tau_{i+1}); \\ 0 & \text{otherwise}. \end{cases}, \quad i = 0 : m - 1. \qquad (4.4.26)$$

The functions $N_{i,1}(x)$ are arbitrarily chosen to be continuous from the right, i.e., $N_{i,1}(\tau_i) = N_{i,1}(\tau_i + 0)$.

For $k = 2$ we define the **hat functions**[17] by

$$N_{i,2}(x) = \begin{cases} (x - \tau_i)/(\tau_{i+1} - \tau_i), & x \in [\tau_i, \tau_{i+1}], \\ (\tau_{i+2} - x)/(\tau_{i+2} - \tau_{i+1}), & x \in [\tau_{i+1}, \tau_{i+2}), \\ 0, & x \notin (\tau_i, \tau_{i+2}), \end{cases} \quad i = -1 : m-1. \tag{4.4.27}$$

where we have introduced two **exterior knots** $\tau_{-1} < \tau_0$ and $\tau_{m+1} > \tau_m$ at the boundaries. (In the following we refer to the knots $\tau_0, \dots, \tau_m$ as **interior knots**.) Note that for $x \in (\tau_i, \tau_{i+1})$ we have $N_{j,2}(x) = 0$, $j \neq i-1, i$. Hence, for a fixed value of x *at most two* hat functions will be nonzero. The exterior knots can be put arbitrarily near the end points τ_0 and τ_m; Indeed they are usually taken to coincide with the boundary so that $\tau_{-1} = \tau_0$ and $\tau_{m+1} = \tau_m$; see Fig. 4.4.5. In this case $N_{-1,1}$ and $N_{m-1,1}$ become "half-hats" with a singularity at τ_0 and τ_m, respectively.

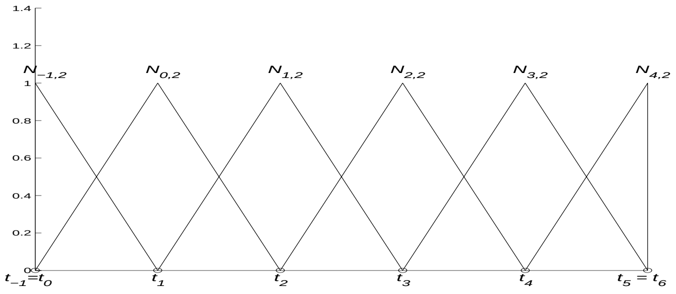


Figure 4.4.4. *The six hat functions $N_{i,2}(x)$, $i = -1 : 4$ ($m + k - 1 = 6$).*

The $(m+1)$ functions $N_{i,2}(x)$, $i = -1 : m-1$, are **B-splines** of order two (degree one). At a distinct knot τ_i just one hat function is nonzero, $N_{i+1}(x) = 1$. It follows that the spline function of order $k = 2$ interpolating the points (τ_i, y_i), $i = 0 : m$, can uniquely be written as

$$s(x) = \sum_{i=-1}^{m-1} c_i N_{i,2}(x). \tag{4.4.28}$$

(4.4.28) with $c_i = y_{i+1}$. This shows that the restriction of the functions $N_{i,2}(x)$, $i = -1 : m-1$, to the interval $[\tau_0, \tau_m]$ are $(m+1)$ linearly independent functions in $S_{\Delta,2}$ and form a basis for $S_{\Delta,2}$.

If we allow two interior knots coalesce, $\tau_i = \tau_{i+1}$, $0 < i < m-1$, then $N_{i-1,2}(x)$ and $N_{i,2}(x)$ will have a discontinuity at τ_i. This generalizes the concept of a B-spline of order 2 given in Definition 4.4.1 and allows us to model functions with discontinuities at certain knots.

It is easily verified that the functions $N_{i,2}(x)$ can be written as a linear combination of the basis function

$$l_i(x) = (x - \tau_i)_+, \quad i = 1 : m+1,$$

[17]The generalization of hat function to two dimensions is often called tent function. This concept is very important in, e.g., in finite element methods;, see Chap. 14.

and it holds that

$$\begin{aligned} N_{i,2}(x) &= \big((x-\tau_{i+2})_+ - (x-\tau_{i+1})_+\big)/(\tau_{i+2}-\tau_{i+1}) \\ &\quad -\big((x-\tau_{i+1})_+ - (x-\tau_i)_+\big)/(\tau_{i+1}-\tau_i) \\ &= \big([\tau_{i+1},\tau_{i+2}]_t(t-x)_+ - [\tau_i,\tau_{i+1}]_t(t-x)_+ \\ &= (\tau_{i+2}-\tau_i)[\tau_i,\tau_{i+1},\tau_{i+2}]_t(t-x)_+, \quad i=1:m. \end{aligned} \tag{4.4.29}$$

Here $[\tau_i,\tau_{i+1},\tau_{i+2}]_t$ means the second order divided difference functional[18] operating on a function of t, i.e., the values τ_i etc. are to be substituted for t not for x. Recall that divided differences are defined also for *coincident values* of the argument, see Sec. 4.2.

From the definition of the Peano kernel and its basic properties, given in Sec. 3.2.3 it follows that the last expression in (4.4.30) tells us that $N_{i,2}$ is the Peano kernel of a second order divided difference functional multiplied by the constant $\tau_{i+2}-\tau_i$. This observation suggests a definition of B-splines of arbitrary order k and a B-spline basis for the space $S_{\Delta,k}$.

Definition 4.4.4. *Let $\Delta=\{\tau_0\le\tau_1\le\cdots\le\tau_m\}$ be an arbitrary sequence of knots such that $\tau_i<\tau_{i+k}$, $i=0:m-k$. Then a B-spline of order k equals (apart from a stepsize factor) the Peano kernel of a k-th order divided difference functional; more precisely we define (with the notations used in this chapter)*

$$N_{i,k}(x) = (\tau_{i+k}-\tau_i)[\tau_i,\tau_{i+1},\ldots,\tau_{i+k}]_t(t-x)_+^{k-1}, \tag{4.4.30}$$

where $[\tau_i,\tau_{i+1},\ldots,\tau_{i+k}]l_x^{k-1}$ denotes the k-th divided difference of the function $l_x^{k-1}(\cdot)$ with respect to the set of points $\tau_i,\tau_{i+1},\ldots,\tau_{i+k}$.

Since divided differences are defined also for coalescing points (see Sec. 4.3), Definition 4.4.4 remains valid for knots that are not distinct.

Example 4.4.4. For $k=1$ (4.4.30) gives ($\tau_i\ne\tau_{i+1}$)

$$N_{i,1}(x) = (\tau_{i+1}-\tau_i)[\tau_i,\tau_{i+1}]_t(t-x)_+^0.$$

If $\tau_i<x<\tau_{i+1}$, then $(\tau_{i+1}-x)_+^0=1$ and $(\tau_i-x)_+^0=0$ and hence $N_{i,1}=1$; otherwise $N_{i,1}=0$. This coincides with the piecewise constant functions in (4.4.26).

It can be shown that $N_{i,k}(x)$ is defined for all x and is a linear combination of functions $(\tau_j-x)_+^{k-1}$. If the knots are distinct then by Problem 4.2.11,

$$N_{i,k}(x) = (\tau_{i+k}-\tau_i)\sum_{j=i}^{i+k}\frac{(\tau_j-x)_+^{k-1}}{\Phi'_{i,k}(\tau_j)}, \quad \Phi_{i,k}(x)=\prod_{j=i}^{i+k}(x-\tau_j). \tag{4.4.31}$$

This shows that $N_{i,k}$ is a linear combination of functions $(\tau_j-x)_+^{k-1}$, $j=i:i+k$, and thus a spline of order k (as anticipated in the terminology).

[18]The notation is defined in Sec. 4.2.1

The B-spline for equidistant knots is related to the probability density of the sum of k uniformly distributed random variables on $[-\frac{1}{2}, \frac{1}{2}]$. This was known already to Laplace.[19] Their importance for spline approximation was first appreciated by Schoenberg [19].

Theorem 4.4.5. *The B-splines of order k has the following properties:*

(i) Positivity: $N_{i,k}(x) > 0, \quad x \in (\tau_i, \tau_{i+k})$.

(ii) Compact support: $N_{i,k}(x) = 0, \quad x \notin [\tau_i, \tau_{i+k}]$.

(iii) Summation property: $\sum_i N_{i,k}(x) = 1, \quad \forall x \in [\tau_0, \tau_m]$.

Proof. A proof can be based on the general facts concerning Peano kernels found in Sec. 3.2.3, where also an expression for the B-spline ($k = 3$) is calculated for the equidistant case. (Unfortunately the symbol x means opposite things here and in Sec. 3.2.3.)

(i) By (4.2.12) $Rf = [\tau_i, \tau_{i+1}, \dots, \tau_{i+k}]f = f^{(k)}(\xi)/k!$, $\xi \in (\tau_i, \tau_{i+k})$, and $Rp = 0$, for $p \in \mathcal{P}_k$. It then follows from the corollary of Peano's remainder theorem that the Peano kernel does not change sign in $[\tau_i, \tau_{i+k}]$. It must then have the same sign as $\int K(u)\,du = R(x-a)^k/k! = 1$. This proves a somewhat weaker statement than (i) ($N_{i,k}(x) \geq 0$ instead of $N_{i,k}(x) > 0$).

(ii) This property follows since a Peano kernel always vanishes outside its interval of support of the functional; in this case $[\tau_i, \tau_{i+k}]$. (A more general result concerning the number of zeros is found, e.g., in Powell [17, Theorem 19.1]. Among other things this theorem implies that the jth derivative of a B-spline, $j \leq k-2$, changes sign exactly j times. This explains the "bell-shape" of B-splines.)

(iii) For a sketch of a proof of the summation property [20], see Problem 8. □

To get a basis of B-splines for the space $S_{\Delta,k}$, $\Delta = \{\tau_0 \leq \tau_1 \leq \cdots \leq \tau_m\}$, $(m+k-1)$ B-splines of order k are needed. We therefore choose $2(k-1)$ additional knots $\tau_{-k+1} \leq \cdots \leq \tau_{-1} \leq \tau_0$, and $\tau_{m+k-1} \geq \cdots \geq \tau_{m+1} \geq \tau_m$, and B-splines $N_{i,k}(x)$, $i = -k+1 : m-1$.

It is convenient to let the exterior knots coincide with the end points,

$$\tau_{-k+1} = \cdots = \tau_{-1} = \tau_0, \qquad \tau_m = \tau_{m+1} = \cdots = \tau_{m+k-1}.$$

It can be shown that this choice tends to optimize the conditioning of the B-spline basis. Figure 4.6.4 shows the first four cubic B-splines for $k = 4$ (the four last B-splines are a mirror image of these). We note that $N_{-3,4}$ is discontinuous, $N_{-2,4}$ has a non-zero first derivative, and $N_{-2,4}$ a non-zero second derivative at the left boundary.

[19] Pierre Simon Laplace (1749–1827), French mathematician and astronomer, has also given important contributions to, e.g., mathematical physics and probability theory.

[20] The B-splines $M_{i,k}$ originally introduced by Curry and Schoenberg were normalized so that $\int_{-\infty}^{\infty} M_{i,k}\,dx = 1$.

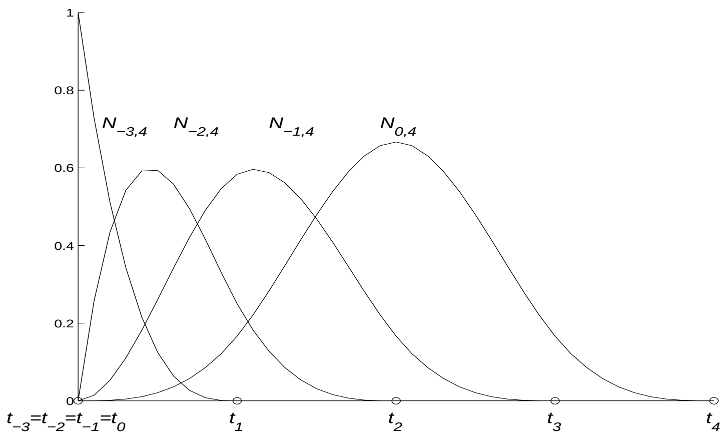


Figure 4.4.5. *The four cubic B-splines nonzero for $x \in (t_0, t_1)$ with coalescing exterior knots $t_{-3} = t_{-2} = t_{-1} = t_0$.*

Interior knots of multiplicity $r > 1$ are useful when we want to model a function, which has less than $k-2$ continuous derivatives at a particular knot. If $r \le k$ interior knots coalesce then the spline will only have $k-1-r$ continuous derivatives at this knot.

Lemma 4.4.6. *Let τ_i be a knot of multiplicity $r \le k$, i.e.,*

$$\tau_{i-1} < \tau_i = \cdots = \tau_{i+r-1} < \tau_{i+r}.$$

Then $N_{i,k}$ is at least $(k-r-1)$ times continuously differentiable at τ_i. For $r = k$, the B-spline becomes discontinuous.

Proof. The truncated power $(t-\tau_i)_+^{k-1}$ is $(k-2)$ times continuously differentiable and $[\tau_i, \dots, \tau_{i+k}]g$ contains at most the $(r-1)$st derivative of g. Hence the lemma follows. □

Consider the spline function

$$s(x) = \sum_{i=-k+1}^{m-1} c_i N_{i,k}(x). \tag{4.4.32}$$

If $s(x) = 0$, $x \in [\tau_0, \tau_m]$, then $s(\tau_0) = s'(\tau_0) = \cdots = s^{(k-1)}(\tau_0) = 0$, and $s(\tau_i) = 0$, $i = 1 : m-1$. From this it can be deduced by induction that in (4.4.32) $c_i = 0$, $i = -k+1 : m-1$. This shows that the $(m+k-1)$ B-splines $N_{i,k}(x)$, $i = -k+1 : m-1$, are linearly independent and form a basis for the space $S_{\Delta,k}$. (A more general result is given in de Boor [3, Theorem IX.1].) Thus any spline function $s(x)$ of order k (degree $k-1$) on Δ can be uniquely written in the form (4.4.32). Note that from the compact support property it follows that for any fixed value of $x \in [\tau_0, \tau_m]$ at most k terms will be nonzero in the sum in (4.4.32), so we have

$$s(x) = \sum_{i=j-k+1}^{j} c_i N_{i,k}(x), \qquad x \in [\tau_j, \tau_{j+1}). \tag{4.4.33}$$

B-splines were not used in practical calculations for general knot sequences until the early seventies, when a stable recurrence relation was established independently by de Boor [1] and Cox [4].

Theorem 4.4.7. *The B-splines satisfy the recurrence relation*

$$N_{i,k}(x) = \frac{x-\tau_i}{\tau_{i+k-1}-\tau_i} N_{i,k-1}(x) + \frac{\tau_{i+k}-x}{\tau_{i+k}-\tau_{i+1}} N_{i+1,k-1}(x). \tag{4.4.34}$$

Proof. (de Boor [3, pp. 130–131]) The recurrence is derived by applying Leibniz' formula for the k-th divided difference (Theorem 4.3.2) to the product

$$(t-x)_+^{k-1} = (t-x)(t-x)_+^{k-2}.$$

This gives

$$\begin{aligned}[\tau_i,\dots,\tau_{i+k}]_t(t-x)_+^{k-1} &= (\tau_i - x)[\tau_i,\dots,\tau_{i+k}]_t(t-x)_+^{k-2} \\ &\quad + 1\cdot[\tau_{i+1},\dots,\tau_{i+k}]_t(t-x)_+^{k-2}. \end{aligned} \tag{4.4.35}$$

since $[\tau_i]_t(t-x) = (\tau_i - x)$, $[\tau_i,\tau_{i+1}]_t(t-x) = 1$, and $[\tau_i,\dots,\tau_j]_t(t-x) = 0$ for $j > i+1$. By the definition of a divided difference

$$(\tau_i - x)[\tau_i,\dots,\tau_{i+k}]_t = \frac{\tau_i - x}{\tau_{i+k}-\tau_i}\left([\tau_{i+1},\dots,\tau_{i+k}]_t - [\tau_i,\dots,\tau_{i+k-1}]_t\right).$$

Substitute this in (4.4.35), simplify and apply the definition of B-splines. This yields (4.4.34). □

Note that with k multiple knots at the boundaries the denominators in (4.4.34) can become zero. In this case the corresponding nominator also is zero and the term should be set equal to zero.

From Property (ii) in Theorem 4.4.5 we conclude that only k B-splines of order k may be nonzero on a particular interval $[\tau_j, \tau_{j+1}]$. Starting from $N_{i,1}(x) = 1$, $x \in [\tau_i, \tau_{i+1})$ and 0 otherwise, cf. (4.4.26), these B-splines of order k can be *simultaneously* evaluated using this recurrence by forming successively their values for order $1 : k$ in only about $\frac{3}{2}k^2$ flops. This recurrence is extremely stable, since it consists of taking positive (nonnegative) combinations of positive (nonnegative) numbers.

Suppose that $x \in [\tau_i, \tau_{i+1}]$, and $\tau_i \neq \tau_{i+1}$. Then the B-splines of order $k = 1, 2, 3, \dots$, nonzero at x can be simultaneously evaluated by computing the triangular array

$$\begin{array}{ccccc}
 & & & 0 & \\
 & & 0 & & \dots \\
 & 0 & & N_{i-3,4} & \\
0 & & N_{i-2,3} & & \dots \\
 & N_{i-1,2} & & N_{i-2,4} & \\
N_{i,1} & & N_{i-1,3} & & \dots \\
 & N_{i,2} & & N_{i-1,4} & \\
0 & & N_{i,3} & & \dots \\
 & 0 & & N_{i,4} & \\
 & & 0 & & \dots \\
 & & & 0 &
\end{array} \tag{4.4.36}$$

The boundary of zeros in the array is due to the fact that all other B-splines not mentioned explicitly vanish at x. This array can be generated column by column. The first column is known from (4.4.26), and each entry in a subsequent column can be computed as a linear combination with nonnegative coefficients of its two neighbors using (4.4.34). Note that if this is arranged in a suitable order the elements in the new column can overwrite the elements in the old column.

To evaluate $s(x)$, we first determine the index i such that $x \in [\tau_i, \tau_{i+1})$ using, e.g., a linear search or bisection (see Sec. 6.1). The recurrence above is then used to generate the triangular array (4.4.36), which provides $N_{j,k}(x)$, $j = i-k+1 : i$. in the sum (4.4.33).

Using the B-spline basis we can formulate a more general interpolation problem, where the $n = m+k-1$ interpolation points, or nodes, x_j do not necessarily coincide with the knots τ_i. We consider determining a spline function $s(x) \in S_{\Delta,k}$, such that

$$s(x_j) = f_j, \quad j = 1 : m+k-1.$$

Since any spline $s(x) \in S_{\Delta,k}$ can be written as a linear combination of B-splines, the interpolation problem can equivalently be written

$$\sum_{i=-k+1}^{m-1} c_i N_{i,k}(x_j) = f_j, \quad j = 1 : m+k-1. \tag{4.4.37}$$

These equations form a linear system $Ac = f$ for the coefficients, where

$$a_{ij} = N_{i-k,k}(x_j), \quad i,j = 1 : m+k-1, \tag{4.4.38}$$

and

$$c = (c_{-k+1}, \dots, c_{m-1})^T, \qquad f = (f_1, \dots, f_{m+k-1})^T.$$

The elements $a_{ij} = N_{i-k,k}(x_j)$ of the matrix A can be evaluated by the recurrence (4.4.34). The matrix A will have a banded structure since $a_{ij} = 0$ unless $x_j \in [\tau_i, \tau_{i+k}]$. Hence at most k elements are nonzero in each row of A. (Note that if $x_j = \tau_i$ for some i only $k-1$ elements will be nonzero, which explains why tridiagonal systems were encountered in cubic spline interpolation in earlier sections.)

Schoenberg and Whitney [20, 1953] showed that *the matrix A is nonsingular if and only if its diagonal elements are nonzero*,

$$a_{jj} = N_{j-k,k}(x_j) \neq 0, \quad j = 1 : n,$$

or equivalently if the nodes x_j satisfy

$$\tau_{j-k} < x_j < \tau_j, \quad j = 1 : n. \tag{4.4.39}$$

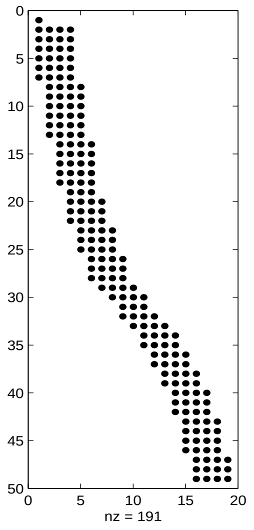
Further, the matrix can be shown to be **totally nonnegative**, i.e., the determinant of every submatrix is nonnegative. For such systems, if Gaussian elimination is carried out*without pivoting*, the error bound is particularly favorable. This will also preserve the banded structure of A during the elimination.

When the B-spline representation (4.4.32) of the interpolant has been determined it can be evaluated at a given point using the recursion formula (4.4.34). If it has to be evaluated at many points it is more efficient to first convert the spline to its polynomial representation (4.4.6). For hints on how to do that see Problem 9 (b) and (c).

Unless the Schoenberg–Whitney condition (4.4.39) is well-satisfied the system may become ill-conditioned. For splines of even order k the interior nodes the

$$\tau_0 = x_0, \quad \tau_{j+1} = x_{j+k/2}, \quad j = 0 : n-k-1, \quad \tau_m = x_n,$$

is a good choice in this respect. In the important case of cubic splines this means that knots are positioned at each data point except the second and next last (cf. the "not a knot" condition in Sec. 4.4.3.



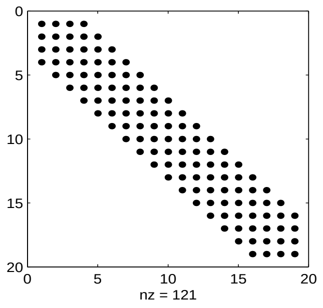


Figure 4.4.6. *Structure of the matrices A and A^TA arising in cubic spline approximation of Titanium data.(nonzero elements showed).*

In some application we are given function values $f_j = f(x_j)$, $j = 1 : n$, that we want to approximate with a spline functions with *much fewer knots* so that $m + k - 1 \leq n$. Then (4.4.37) is an *overdetermined linear system* and the interpolation conditions cannot be satisfied exactly. We therefore consider the linear **least squares spline approximation** problem

$$\min \sum_{j=1}^{n} \left(\sum_{i=-k+1}^{m-1} c_i N_{i,k}(x_j) - f_j \right)^2 . \tag{4.4.40}$$

Using the same notation as above this can be written in matrix form

$$\min_{c} \|Ac - f\|_2^2. \tag{4.4.41}$$

The matrix A will have full column rank equal to $m + k - 1$ if and only if there is a subset of points τ_j satisfying the Schoenberg–Whitney conditions (4.4.39). If A has full column rank then the least squares solution c is unique and is uniquely determined by the normal equations $A^TAc = A^Tf$. The matrix A^TA is symmetric and positive definite and hence the normal equations can be solved using Cholesky factorization of A^TA. A will have at most k nonzero elements in each row; see Fig. 4.4.5. Advantage should be taken of the banded form of the matrix A^TA. More stable methods for solving linear least squares problems (4.4.41) will be introduced in Sec. 8.5.7.

Example 4.4.5. (de Boor [3]) Consider experimental data describing a property of titanium as a function of temperature. Experimental values for $t_i = 585 + 10i$, $i = 1 : 49$, are given. We want to fit this data using a least squares cubic spline Figure 4.4.7 shows results from using a least squares fitted cubic spline with 9 and 17 knots, respectively. The spline with 9 knots shows oscillations near the points where the curve flattens out and the top of the peak is not well matched. Increasing the number of knots to 17 we get a very good fit.

We have in the treatment above assumed that the set of (interior) knots $\{\tau_0 \leq \tau_1 \leq \cdots \leq \tau_m\}$ is given. In many spline approximation problems it is more realistic to consider the location of knots to be free and try to determine a small set of knots such that the given data can be approximated to a some preassigned accuracy. Several schemes have been developed to treat this problem.

One class of algorithms start with only a few knots and iteratively add more knots guided by some measure of the error; see de Boor [1, Chapter XII]. The placement of the knots are chosen so that the Schoenberg–Whitney conditions are always satisfied. The iterations are stopped when the approximation is deemed satisfactory. If a node $\tilde{\tau} \in [\tau_j, \tau_{j+1})$ is inserted then the B-spline series with respect to the enlarged set of nodes can cheaply and stably be computed from the old one (see Dierckx [8]).

Other algorithms starts with many knots and successively *remove* knots, which are not contributing much to the quality of the approximation. In these two classes of algorithms one does not seek an optimal knot placement at each step. This is done in a more recent algorithms; see Schwetlick and Schütze [22].

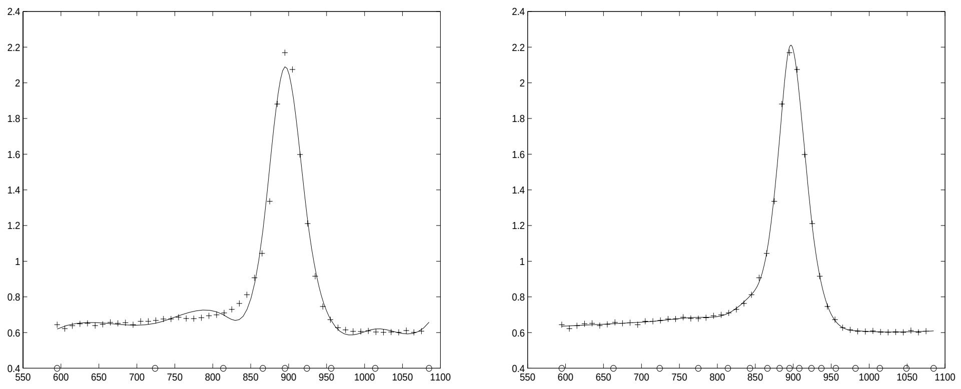


Figure 4.4.7. *Least squares cubic spline approximation of Titanium data; the knots are marked on the axes by a "o"; left: 9 knots;left: 17 knots.*

Review Questions

1. What is meant by a cubic spline function? Give an example where such a function is better suited than a polynomial for approximation over the whole interval.
2. (a) What is the dimension of the space $S_{\Delta,k}$ of spline functions of order k on a grid $\Delta = \{x_0, x_1, \dots, x_m\}$? Give a basis for this space.

   (b) Set up the linear system for cubic spline interpolation in the equidistant case for some common boundary conditions. What does the unknown quantities mean, and what conditions are expressed by the equations? About how many operations are required to interpolate a cubic spline function to $m + 1$, $m \gg 1$, given values of a function?
3. What error sources have influence on the results of cubic spline interpolation? How fast do the boundary errors die out? How do the results in the interior of the interval depend on the step size (asymptotically)? One of the common types of boundary conditions yield much larger error than the others. Which one? Compare it quantitatively with one of the others.
4. Approximately how many arithmetic operations are required to evaluate the function values of all cubic B-splines that are nonzero at a given point?
5. Express the restrictions of $f(x) = 1$ and $f(x) = x$ to the interval $[x_0, x_m]$ as linear combinations of the hat functions defined by (4.4.27).
6. The Schoenberg–Whitney conditions give necessary and sufficient conditions for a certain interpolation problem with B-splines of order k. What is the interpolation problem and what are the conditions?

Problems and Computer Exercises

1. Suppose that $f(x)$ and the grid Δ are symmetric around the midpoint of the interval $[a, b]$. You can then considerably reduce the the amount of computation needed for

the construction of the cubic spline interpolant by replacing the boundary condition at $x = b$ by an adequate condition at the midpoint. Which?

(a) Set up the matrix and right hand side for this in the case of constant step size h.

(b) Do the same for a general case of variable step size.

2. (a) Write a program for solving a tridiagonal linear system by Gaussian elimination without pivoting. Assume that the nonzero diagonals are stored in three vectors. Adapt it to cubic spline interpolation with equidistant knots with several types of boundary conditions.

(b) Consider the triadiagonal system resulting from the not-a-knot boundary conditions. Show that after eliminating k_0 between the first two equations and k_m between the last two eqiuations the remaining tridiagonal system for $k_1, \dots, k_{m-1}$ is diagonally dominant.

(c) Interpolate a cubic spline $s(x)$ through the points $(x_i, f(x_i))$, where

$$f(x) = (1 + 25x^2)^{-1}, \quad x_i = -1 + \frac{2}{10}(i-1), \quad i = 1 : 11.$$

Compute a natural spline, a complete spline (here $f'(x_1)$ and $f'(x_{11})$ are needed) and a “not a knot” spline. Compute and compare error curves (natural and logarithmic).

(c) Similar runs as in (b), though for $f(x) = 1/x$, $1 \le x \le 2$, with $h = 0.1$ and $h = 0.05$. Compare the “almost complete”, as described in the text, with the complete and the natural boundary condition.

3. If f'' is known at the boundary points, then the boundary conditions can be chosen so that $f'' = s''$ at the boundary points. Show that this leads to the conditions

$$\begin{aligned} 2k_0 + k_1 &= 3d_1 - h_1 f''(x_0), \\ k_{m-1} + 2k_m &= 3d_m + h_m f''(x_m). \end{aligned}$$

4. Show that the formula

$$\int_{x_0}^{x_m} s(x)dx = \sum_{i=1}^{m} \Big(\frac{1}{2} h_i (y_{i-1} + y_i) + \frac{1}{12}(k_{i-1} - k_i) h_i^2 \Big),$$

is exact for all cubic spline functions $s(x)$. How does the formula simplify if all $h_i = h$?

Hint: Integrate (4.4.4) from x_{i-1} to x_i.

5. In (4.4.4) the cubic spline $q_i(x)$ on the interval $[x_{i-1}, x_i)$ is expressed in terms of function values y_{i-1}, y_i, and the first derivatives k_{i-1}, k_i.

(a) Show that if $M_i = s''(x_i)$, $i = 0, m$, are the *second derivatives* (also called **moments**) of the spline function then

$$k_i - d_i = \frac{h_i}{6}(2M_i + M_{i-1}), \qquad k_{i-1} - d_i = -\frac{h_i}{6}(M_i + 2M_{i-1}).$$

Hence $q_i(x)$ can also be uniquely expressed in terms of y_{i-1}, y_i and M_{i-1}, M_i.

(b) Show that, using the parametrization in (a), the continuity of the *first* derivative of the spline function at an interior point x_i gives the equation

$$h_i M_{i-1} + 2(h_i + h_{i+1}) M_i + h_{i+1} M_{i+1} = 6(d_{i+1} - d_i).$$

6. (a) Develop an algorithm for solving the arrowhead linear system $Tk = g$ (4.4.20), using Gaussian elimination without pivoting. Show that about twice the number of arithmetic operations are needed compared to a tridiagonal system.

(b) At the end of Sec. 4.4.3 parametric spline interpolation to given points (x_i, y_i), $i = 0 : m$, is briefly mentioned. Work out the details on how to use this to represent a closed curve. Try it out on a boomerang, an elephant, or what have you?

7. (a) Compute and plot a B-spline basis of order $k = 3$ (locally quadratic) and $m = 6$ subintervals of equal length.

Hint: In the equidistant case there is some translation invariance and symmetry, so you do not really need more than essentially three different B-splines. You need one spline with triple knot at x_0 and a single knot at x_1 (very easy to construct), and two more splines.

(b) Set up a scheme to determine a locally quadratic B-spline, which interpolates given values at the *midpoints* $x_i = (\tau_{i+1} + \tau_i)/2$ $(\tau_{i+1} \neq \tau_i)$, $i = 0 : m-1$, and the boundary points τ_0, τ_m. Show that the spline is uniquely determined by these interpolation conditions.

8. Use the recurrence (4.4.34)

$$N_{i,k}(x) = \frac{x - \tau_i}{\tau_{i+k-1} - \tau_i} N_{i,k-1}(x) + \frac{\tau_{i+k} - x}{\tau_{i+k} - \tau_{i+1}} N_{i+1,k-1}(x)$$

to show that

$$\sum_i N_{i,k}(x) = \sum_i N_{i,k-1}(x), \qquad \tau_0 \le x \le \tau_m,$$

where the sum is taken over all nonzero values. Use this to give an induction proof of the summation property in Theorem 4.4.5.

9. (a) Using the result $\dfrac{d}{dx}(t-x)_+^{k-1} = -(k-1)(t-k)_+^{k-2}$, $k \ge 1$, show the formula for differentiating a B-spline

$$\frac{d}{dx} N_{i,k}(x) = (k-1)\left(\frac{N_{i,k-1}(x)}{\tau_{i+k-1} - \tau_i} - \frac{N_{i+1,k-1}(x)}{\tau_{i+k} - \tau_{i+1}} \right).$$

Then use the relation (4.4.34) to show

$$\frac{d}{dx} \sum_{i=r}^{s} c_i N_{i,k}(x) = (k-1) \sum_{i=r}^{s+1} \frac{c_i - c_{i-1}}{\tau_{i+k-1} - \tau_i} N_{i,k-1}(x),$$

where $c_{r-1} := c_{s+1} := 0$.

(b) Given the B-spline representation of a cubic spline function $s(x)$. Show how to find its polynomial representation (4.4.6) by computing the function values and first derivatives $s(\tau_i), s'(\tau_i)$, $i = 0 : m$.

(c) Apply the idea in (a) recursively to show how to compute all derivatives of $s(x)$ up to order $k-1$. Use this to develop a method for computing the polynomial representation of a spline of arbitrary order k from its B-spline representation.

4.5 Polynomial Interpolation of Analytic Functions

In this section we make a more detailed theoretical and experimental study of interpolation of an analytic function $f(z)$ on a real interval. including an analysis of the **Runge phenomenon**. We then study interpolation at an infinite equidistant point set from the point of view of Complex Analysis. This interpolation problem, which was studied by Whittaker and others at the beginning of the century, became revived at the middle of the century under the name of the **Shannon sampling theorem**, with important applications to Communication Theory.

We shall encounter multi-valued functions: the logarithm and the square root. For each of these we choose that branch, which is positive for large positive values of the argument. They will appear in such contexts that we can then keep them non-ambiguous by forbidding z to pass the interval $[-1,1]$. (We can, however, allow z to approach that interval.)

4.5.1 Chebyshev Interpolation

We first consider the general problem of polynomial interpolation of an analytic function, at an arbitrary sequence of distinct points in $\mathbf{C}$. We let

$$\Phi(z) = (z-x_1)(z-x_2)...(z-x_n), \quad z \in \mathbf{C}, \quad x_j \in \mathbf{C},$$

where $x_i \neq x_j$, $i \neq j$. The following theorem is valid, when the interpolation points x_j are in the complex plane, although we shall here mainly apply it to the case, when they are located in the interval $[-1,1]$.

Theorem 4.5.1.

Assume that $f(z)$ is analytic in a domain $\mathcal{D}$ that contains the points $x_1, x_2, \dots x_n$, as well as the point $u \in \mathbf{C}$. Let $L_n f$ be the solution of the interpolation problem $(L_n f)(x_j) = f(x_j)$, $j = 1,2,...,n$. Then the interpolation error can be expressed as a complex integral, $f(u) - (L_n f)(u) = I_n(u)$, where

$$I_n(u) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}} \frac{\Phi(u)f(z)}{\Phi(z)(z-u)}\,dz.$$

Proof: By the residue theorem,

$$I_n(u) = \sum_{j=1}^{n} \frac{\Phi(u)f(x_j)}{(u-x_j)\Phi'(x_j)} + f(u),$$

where the sum, with reversed sign, is Lagrange's form of the interpolation polynomial. □

(Note the relation between the Lagrange interpolation formula and the expansion of $f(z)/\Phi(z)$ into partial fractions, when, e.g., $f(z)$ is a polynomial).

We now proceed to *Chebyshev interpolation*, i.e., interpolation at the the zeros of the Chebyshev polynomials. We shall show that it is almost as efficient as the truncation of a Chebyshev expansion. In this case, $\Phi(z) = 2^{1-n}T_n(z)$. Let $\mathcal{D} = \mathcal{E}_R$, $x \in [-1,1]$, $z \in \partial\mathcal{E}_R$. Consider the integral in Theorem 4.5.1 and assume that

$|f(z)| \le M$ for $z \in \partial\mathcal{E}_R$. It can be shown (Problem 2) that $|T_n(x)| \le 1$ and

$$|T_n(z)| \ge \tfrac{1}{2}(R^n - R^{-n}), \quad |z-x| \ge a-1, \quad \int_{\partial\mathcal{E}_R} |dz| \le 2\pi a,$$

where a is the major semi-axis of $\mathcal{E}_R$, i.e. $a = \frac{1}{2}(R + R^{-1})$. Then, by a straightforward calculation,

$$|f(x) - (L_n f)(x)| \le \frac{2MR^{-n}a}{(1-R^{-2n})(a-1)}. \tag{4.5.1}$$

This is somewhat less sharp than the result obtained by the Chebyshev expansion, in particular when $R \approx 1$. The details are left for Problem 2.

Note, however, that $f(z)$ is allowed to have a singularity arbitrarily close to the interval $[-1,1]$, and the convergence of Chebyshev interpolation will still be exponential. Of course, the exponential rate will be very poor, when $R \approx 1$.

4.5.2 Analysis of The Runge Phenomenon

It is well known that the Taylor series of an analytic function converges at an exponential rate inside its circle of convergence, while it diverges at an exponential rate outside. We shall see that a similar result holds for certain interpolation processes. In general, the domains of convergence are not disks but bounded by level curves of a logarithmic potential, related to the asymptotic distribution of the interpolation points.

For the sake of simplicity, we now confine the discussion to the case, when the points of interpolation are located in the standard interval $[-1,1]$, but we are still interested in the evaluation of the polynomials in the complex domain. Part of the discussion can, however, be generalized to a case, when the interpolation points are on an arc in the complex plane.

Let $q : [a,b] \mapsto [-1,1]$ be an increasing and continuously differentiable function. Set $t_{n,j} = a + (b-a)j/n$, $j = 0,1,2,\dots,n$, and let the interpolation points be $x_{n,j}$, $j = 1,2,\dots,n$, where $q(t_{n,j-1}) < x_{n,j} \le q(t_{n,j})$, i.e. one interpolation point in each of n subintervals of $[-1,1]$. In the definition of Φ, we now write $x_{n,j}$, Φ_n instead of x_j, Φ. Note that, as $n \to \infty$,

$$\frac{1}{n}\ln\Phi_n(z) = \frac{1}{n}\sum_{j=1}^{n} \ln(z - x_{n,j}) \;\to\; \psi(z) := \frac{1}{b-a}\int_a^b \ln(z - q(t))\,dt, \quad z \notin [-1,1]. \tag{4.5.2}$$

Put $x = q(t)$, and introduce a density function $w(x), x \in]-1,1[$ that is the derivative of the inverse function of q, i.e., $w(x) = 1/\big(q'(t(x))(b-a)\big) > 0$. Then

$$\psi(z) = \int_{-1}^{1} \ln(z-x)w(x)\,dx, \quad w(x) > 0, \quad \int_{-1}^{1} w(x)\,dx = 1, \tag{4.5.3}$$

and $\psi(z)$ is analytic in the whole plane outside the interval $[-1,1]$. Its real part $P(z)$ is the **logarithmic potential** of a weight distribution,

$$P(z) = \Re\psi(z) = \int_{-1}^{1} \ln|z-x|w(x)\,dx, \quad w(x) > 0. \tag{4.5.4}$$

Figure 4.5.1. *Figure to be made.*

The function $\frac{1}{n}\ln|\Phi_n(z)|$ is itself the logarithmic potential of a discrete distribution of equal weights $\frac{1}{n}$, at the interpolation points $x_{j,n}$. This function is less pleasant to deal with than $P(z)$, since it becomes $-\infty$ at the interpolation points while, according to classical results of potential theory, $P(z)$ *is continuous everywhere, also on the interval* $[-1,1]$. If we set $z = x+iy$, $\partial P(z)/\partial x$ is also continuous for $z \in]-1,1[$, while $\partial P(z)/\partial y$ has a jump there. We write it thus,

$$\psi'(x-0i) - \psi'(x+0i) = 2\pi i w(x). \tag{4.5.5}$$

By (4.5.3), $\psi(z) = \ln z + O(z^{-1})$, $|z| \to \infty$, or even $O(z^{-2})$, if the weight distribution is symmetric around the origin.

We make the definition

$$\mathcal{D}(v) = \{z \in \mathbf{C} : P(z) < P(v)\}.$$

and set $P^* = \max_{x\in[-1,1]} P(x)$. It can be shown that $\mathcal{D}(v)$ is a simply connected domain if $P(v) > P^*$. The level curve $\partial\mathcal{D}(v) = \{z : P(z) = P(v)\}$ then encloses $[-1,1]$. A level curve $\{z : P(z) = a\}$ is strictly inside the level curve $\{z : P(z) = a'\}$ if $a \le P^* < a'$. (The proof of these statements essentially utilizes the minimum principle for harmonic functions and the fact that $P(z)$ is a regular harmonic function outside $[-1,1]$ that grows to ∞ with $|z|$.)

We now consider two examples.

Example 4.5.1. *Equidistant interpolation* In this case we may take $q(t) = t$, $t \in [-1,1]$, hence $w(x) = 1/2$. For the **equidistant case** we have if $z \notin [-1,1]$,

$$P(z) = \tfrac{1}{2}\Re\int_{-1}^{1} \ln(z-x)\,dx = \tfrac{1}{2}\Re\big((1-z)\ln(z-1) + (1+z)\ln(z+1)\big) - 1.$$

The upper half of the level curves may look something like Fig. 4.4.1.:

On the imaginary axis,

$$P(iy) = \tfrac{1}{2}\ln(1+y^2) + y(\tfrac{1}{2}\pi - \arctan y) - 1.$$

When $z \to x \in [-1,1]$, from any direction, $P(z)$ tends to

$$P(x) = \tfrac{1}{2}\big((1-x)\ln(1-x) + (1+x)\ln(1+x)\big) - 1. \tag{4.5.6}$$

$P'(x)$ is continuous in the interior, but becomes infinite at $x = \pm 1$. The imaginary part of $\psi(z)$ has, however, different limits, when the interval is approached from above and below: $\Im(\psi(x \pm 0i)) = \pm\pi(1-x)$.

The level curve of $P(z)$ that passes through the points ± 1, intersects the imaginary axis at the points $\pm iy$, determined by the equation $P(iy) = P(1) = \ln 2 - 1$, with the root $y = 0.5255$. Theorem 4.5.2 (below) will tell us that $L_n f(x) \to f(x)$, $\forall x \in]-1,1[$, if $f(z)$ is analytic inside and on this contour.

In the classical example of Runge, $f(z) = 1/(1+25z^2)$ has poles inside this contour at $z = \pm 0.2i$. Proposition 4.5.3 will tell us that the level curve of $P(z)$ that passes through these poles will separate between the points, where the interpolation process converges and diverges. Its intersections with the real axis is determined by the equation $P(x) = P(0.2i) = -1.41142$. The roots are $x = \pm 0.72668$.

Example 4.5.2. *Chebyshev interpolation*
In this example we have

$$q(t) = \cos(\pi(1-t)), \quad t \in [0,1], \qquad w(x) = \frac{1}{\pi}(1-x^2)^{-\frac{1}{2}}.$$

Moreover (see Sec. 3.5.1) substitute s for w,

$$\Phi_n(z) = 2^{1-n}T_n(z) = 2^{-n}(s^n + s^{-n}),$$

where $z = \frac{1}{2}(s + s^{-1})$, $s = z + \sqrt{z^2-1}$. Note that $|s| \geq 1$, according to our convention about the choice of branch for the square root. Hence,

$$P(z) = \lim \frac{1}{n}\ln|\Phi_n(z)| - \ln 2 = \ln\frac{|s|}{2} = \ln|z + \sqrt{z^2-1}| - \ln 2.$$

Therefore, the family of confocal ellipses $\partial\mathcal{E}_R$ introduced in Sec. 3.5.1 (see eq. (3.1.30) are, in this example, the level curves of $P(z)$. In fact, by Eq. (1.3') and the formula for $P(z)$, the interior of $\mathcal{E}_R$ equals $\mathcal{D}(\ln R - \ln 2)$. The family includes, as a limit case ($R = 1$), the interval $[-1,1]$, in which $P(z) = -\ln 2$.

Our problem is related to a more conventional application of potential theory, namely the problem of finding the electrical charge distribution of a long insulated charged metallic plate in the strip

$$\{(x,y) \in \mathbf{R}^2 : \quad -1 < x < 1, \quad -L < y < L, \quad L \gg 1\}.$$

Such a plate will be equipotential. The charge density at the point (x,y) is then proportional to $w(x) = \frac{1}{\pi}(1-x^2)^{-1/2}$; *a fascinating relationship between electricity and approximation.*

Note that if $z \notin [-1,1]$, we can, by the definition of $P(z)$ as a Riemann sum (see (4.1)) find a sequence $\{\epsilon_n\}$ that decreases monotonically to zero, such that

$$\frac{1}{n}|\ln\Phi_n(z) - \psi(z)| < \epsilon_n, \quad z \notin [-1,1]. \tag{4.5.7}$$

It is conceivable that the same sequence can be used for all z on a curve that does not touch the interval $[-1,1]$. (The proof is omitted.)

We can only claim a *one-sided inequality*, if we allow that $u \in [-1,1]$.

$$\frac{1}{n}(\Re \ln \Phi_n(u) - \psi(u)) < \epsilon_n, \quad u \in \mathbf{C}. \tag{4.5.8}$$

(Recall that $\Re \ln \Phi_n(u) = -\infty$ at the interpolation points.) We can use the same sequence for z and u. We can also say that $|\Phi_n(u)|$ *behaves like* $\exp\big((P(u) \pm \delta)n\big)$ *outside the immediate vicinity of the interpolation points.*

Theorem 4.5.2.

Assume that $[-1,1]$ *is strictly inside a simply connected domain* $\mathcal{D} \supseteq \mathcal{D}(v)$. *If* $f(\zeta)$ *is analytic in the closure of* $\mathcal{D}$, *then the interpolation error* $(L_n f)(u) - f(u)$ *converges like an exponential to* 0 *for any* $u \in \mathcal{D}(v)$.

Proof: By Theorem 4.5.1, $f(u) - (L_n f)(u) = I_n(u)$, where

$$I_n(u) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}} \frac{\Phi_n(u) f(z)}{\Phi_n(z)(z-u)}\, dz. \tag{4.5.9}$$

Note that $P(z) \ge P(v)$, because $\mathcal{D} \supseteq \mathcal{D}(v)$. Then, by (4.5.7) and (4.5.8), $|\Phi_n(u)/\Phi_n(z)| < \exp n\big(P(u) - P(v) + 2\epsilon_n\big)$ Let $|f(z)| \le M$. For any $u \in \mathcal{D}(v)$, we can choose $\delta > 0$, such that $P(u) < P(v) - 3\delta$, $|z-u| > \delta$. Next, choose n large enough so that $\epsilon_n < \delta$. Then

$$|f(u) - (L_n f)(u)| < \frac{1}{2\pi} M \exp n(-3\delta + 2\delta) \int_{\partial\mathcal{D}(v)} \frac{|dz|}{\delta} \le \frac{K\exp(-n\delta)}{\delta}.$$

where $K = K(v)$ does not depend on n, δ and u, hence the convergence is exponential. ☐

Remark If u is away from the boundary $\partial\mathcal{D}(v)$, more realistic estimates of the interpolation error and the speed of convergence is for $n \gg 1$ given by

$$K_1(v)|\Phi_n(u)|e^{(-P(v)+\delta)n} \le K_1(v)e^{(P(u)-P(v)+\delta)n},$$

where $K_1(v)$ is another constant. The latter estimate is realistic outside the immediate vicinity of the interpolation points.

It seems, as if one should choose $|v|$ as large as possible, in order to increase $P(v)$. A bound for $P(v)$ is usually set by the singularities of $f(z)$. If $f(z)$ is an entire function, the growth of the maximum modulus of $|f(z)|$, $|z| \in \mathcal{D}(v)$, hidden in $K_1(v)$, sets a bound for $P(v)$ that usually increases with n.

We shall now derive a complement and a kind of converse to Theorem 4.5.2, for functions $f(z)$ that have simple poles in $\mathcal{D}(v)$.

Proposition 4.5.3.

Assume that $[-1,1]$ *is strictly inside a domain* $\mathcal{D} \supset \mathcal{D}(v)$, *and that* $f(\zeta)$ *is analytic in the closure of* $\mathcal{D}$, *except for a finite number of simple poles* p *in the interior, all with the same value of* $P(p)$.

Outside the interval $[-1,1]$, *the curve* $\partial\mathcal{D}(p)$ *then separates the points, where the sequence* $\{(L_nf)(u)\}$ *converges, from the points, where it diverges. The behavior of* $|(L_nf)(u) - f(u)|$, *when* $u \in \mathcal{D}(v)$, $n \gg 1$ *is roughly described by the formula,*

$$|(f - L_nf)(u)| \approx K|\Phi_n(u)|e^{(-P(p)\pm\delta)n}/\max_p(1/|p-u|). \tag{4.5.10}$$

This can be further simplified, if u *is not in immediate vicinity of the interpolation points, see* (4.5.11).

Proof sketch: At the application of the residue theorem to the integral $I_n(u)$, see (4.5.9), we must this time also conider the poles of $f(z)$. We obtain

$$\frac{I_n(u)}{\Phi_n(u)} = \frac{(f - L_nf)(u)}{\Phi_n(u)} + \sum_p \frac{\mathrm{res}_f(p)}{\Phi_n(p)(p-u)},$$

where $\mathrm{res}_f(p)$ is the residue of f at the pole $p \in \mathcal{D}(v)$. Roughly speaking, for $n \gg 1$,

$$I_n(u) = O\big(e^{-P(v)+\delta)n}\big), \qquad \sum_p = O\big(e^{(-P(p)\pm\delta)n}/\max_p(1/|p-u|), \quad P(v) > P(p).$$

It follows that $|I_n| \ll \sum$, unless there is a cancellation of terms in $\sum$. For fixed n, $\sum = 0$ is equivalent to an algebraic equation of degree less than the number of poles, and the roots will depend on n. We conclude that the case of cancellation can be ignored, and hence we obtain Eq. (4.5.10), and the following simplified version, valid if u is not in the immediate vicinity of the interpolation points. $|\Phi_n(u)|$ behaves like $\exp(P(u) \pm \delta)n$.

$$|(f - L_nf)(u)| \approx Ke^{(P(u)-P(p)\pm\delta)n}/\max_p(1/|p-u|). \tag{4.5.11}$$

The separation statement follows from this. $\square$

There are several interpolation processes with interpolation points in $[-1,1]$ that converge for all $u \in [-1,1]$, when the condition of analyticity is replaced by a more modest smoothness assumption, e.g., $f \in C^p$. This is the case, when the sequence of interpolation points are the zeros of the orthogonal polynomials which belong to a density function that is continuous and strictly positive in $]-1,1[$. We shall prove the following result.

Proposition 4.5.4.

Consider an interpolation process where the interpolation points has a (perhaps unknown) asymptotic density function $w(x)$, $x \in [-1,1]$. *Assume that*

$$(L_nf - f)(x) \to 0, \quad \forall x \in [-1,1], \quad \forall f \in C^k[-1,1],$$

as $n \to \infty$, for some $k \geq 1$. Then the logarithmic potential $P(x)$ must be constant in $[-1, 1]$, and the density function must be the same as for Chebyshev interpolation, i.e., $w(x) = \frac{1}{\pi}(1 - x^2)^{-1/2}$.

Proof: Let $f(z)$ be analytic in some neighborhood of $[-1, 1]$, e.g. any function with a pole at a point p (arbitrarily) close to this interval. A fortiori, for such a function our interpolation process must converge at all points u in some neighborhood of the interval $[-1, 1]$.

Suppose that $P(x)$ is not constant, and let x_1, x_2 be points, such that $P(x_1) < P(x_2)$. We can then choose the pole p so that $P(x_1) + \delta < P(p) < P(x_2) - \delta$. By Proposition 4.5.3, the process would then diverge at some points u arbitrarily close to x_2. This contradiction shows that $P(x)$ must be constant in $[-1, 1]$, $P(x) = a$, (say).

This gives a Dirichlet problem for the harmonic function $P(z)$, $z \notin [-1, 1]$. This has a unique solution, and one can verify that the harmonic function $P(z) = a + \Re \ln(z + \sqrt{z^2 - 1})$ satisfies the boundary condition. We must also determine a. This is done by means of the behaviour as $z \to \infty$. We find that

$$\begin{aligned} P(z) = a + \Re \ln(z + z(1 - z^{-2})^{1/2}) &= a + \Re \ln(2z - O(z^{-1}) \\ &= a + \Re \ln z + \ln 2 - O(z^{-2}). \end{aligned}$$

This is to be matched with the result of the discussion of the general logarithmic potential in the beginning of Sec. 4.5.2. In our case, where we have a symmetric distribution, and $\int_1^1 w(x)dx = 1$, we obtain $P(z) = \Re\psi(z) = \Re \ln z + O(z^{-2})$. The matching yields $a = -\ln 2$.

Finally, by (4.5.5), we obtain after some calculation, $w(x) = (1 - x^2)^{-1/2}$. The details are left for Problem 3. □

Compare the above discussion with the derivations and results concerning the asymptotic distribution of the zeros of orthogonal polynomials, given in the standard monograph G. Szegö [24].

Related ideas were applied around 1980 in the NADA thesis of L. Reichel. His problem was the Helmholtz equation in 2D, with a regionally constant complex coefficient, with potential applications (excuse my pun!) to the microwave heating of cheeseburgers etc.. Since one can find nice bases of particular solutions of the Helmholtz equation in different regions, i.e. bread, meat and cheese, one may try **boundary collocation**, to express the appropriate continuity conditions at the interfaces between meat and cheese etc. in a finite number of points. It turned out that a good allocation of these collocation points is crucial. It depends on the geometry, and this question became, for some choices of bases, similar to polynomial interpolation.

4.5.3 The Sampling Theorem

The ideas of this paper can be applied to other interpolation problems than polynomial interpolation. We shall apply them to a derivation of the celebrated sampling theorem which is an interpolation formula that expresses a function that is **band-limited** to the frequency interval $[-W, W]$, i.e., a function that has a Fourier representation of the following form (see also Strang [23, p. 325].

$$f(z) = \frac{1}{2\pi}\int_{-W}^{W} \hat{f}(k)e^{ikz}\,dk, \quad |\hat{f}(k)| \le M, \tag{4.5.12}$$

in terms of its values at all integer points. The **Shannon Sampling Theorem** reads,

$$f(z) = \sum_{j=-\infty}^{\infty} f\left(\frac{j\pi}{W}\right)\frac{\sin(Wz - j\pi)}{(Wz - j\pi)}. \tag{4.5.13}$$

This is, like Lagrange's interpolation formula, a so-called *cardinal interpolation formula.* As Wz/π tends to an integer m, all terms except one on the right hand side become zero; for $j = m$ the term becomes $f(m\pi/W)$.

We shall sketch a derivation of this for $W = \pi$. The details are left for Problem 9. We first note that Eq. (4.5.12) shows that $f(z)$ is analytic for all z. Then we consider the same Cauchy integral as many times before,

$$I_n(u) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}_n} \frac{\Phi(u)f(z)}{\Phi(z)(z-u)}\,dz, \quad u \in \mathcal{D}_n.$$

Here $\Phi(z) = \sin \pi z$, which vanishes at all integer points, and $\mathcal{D}_n$ is the *open* rectangle with vertices at $\pm(n + 1/2) \pm bi$. By the residue theorem, we obtain after a short calculation,

$$I_n(u) = f(u) + \sum_{j=-n}^{n} \frac{\Phi(u)f(j)}{\Phi'(j)(j-u)} = f(u) - \sum_{j=-n}^{n} \frac{f(j)\sin \pi(j-u)}{\pi(j-u)}.$$

Set $z = x + iy$. Note that

$$|f(z)| \le \frac{1}{2\pi}\int_{-\pi}^{\pi} Me^{-ky}dk \le \frac{M(e^{|\pi y|} - e^{-|\pi y|})}{|2\pi y|}, \quad |\Phi(z)| \ge e^{|\pi y|}.$$

These inequalities, applied for $y = b$, allow us to let $b \to \infty$; ($2b$ is the height of the symmetric rectangular contour). Then it can be shown that $I_n(u) \to 0$ as $n \to \infty$, which establishes the sampling theorem for $W = \pi$. The general result is then obtained by "regula de tri", but it is sometimes hard to get it right, see Problem 9. ☐Strang, loc.cit., gives an entirely different derivation, based on Fourier analysis.

Problems

1. We use the notations and assumptions of Theorem 4.5.1.

(a) *Representation of the interpolation operator as an integral operator.* Show that

$$(L_nf)(x) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}} K(x,z)\frac{f(z)}{\Phi(z)}\,dz, \quad K(x,z) = \frac{\Phi(x) - \Phi(z)}{(x-z)},$$

also if $x \notin \mathcal{D}$. Note that $K(x, z)$ is a polynomial, symmetric in the two variables x, z.

(b) *A formula for the divided difference.* Show that

$$[x_1, x_2, \ldots, x_n]f = \frac{1}{2\pi i} \int_{\partial\mathcal{D}} \frac{f(z)}{\Phi(z)}\, dz.$$

Hint: Look at the leading term of the polynomial $(L_n f)(x)$.

2. Check the omitted details of the derivations in Sec. 4.5.1.

3. Check the validity of Eq. (4.5.5) on the Chebyshev and the equidistant cases. Also show that $\int_{-1}^{1} w(x)\, dx = 1$, and check the statements about the behaviour of $P(z)$ for $|z| \gg 1$.

4. *Literature search.* Find the articles, where L. Reichel analyzes boundary collocation methods, by techniques similar to those used in Sec. 4.5.2. There are both green NADA reports and publications in journals or conference proceedings.

5. (a) Work out the details of the proof of the Sampling Theorem.

(b) The formulation of the Sampling Theorem with a general W in Strang, loc cit., does not agree with our Eq, (4.5.13). Who is right?

Computer Exercises

1. (a) Write a program for solving equations of the form $\psi(z) = c$, where c runs through a rectangular grid in a complex plane, not necessarily equidistant. You may assume that ψ is defined in such a way, that that its derivative is rather easily computed. When applicable, compute also the intersections of two families of level curves, i.e., with constant $\Re c$ and constant $\Im c$, with the real axis.

(b) Apply your program(s) to the plotting of these level curves in the two cases of the text, (related to, respectively, equidistant and Chebyshev interpolation). Due to the symmetry it is sufficient to draw the curves in a quarter-plane. Think of the "aspect" of the plotting so that the conformality of the mappings becomes visible. If the scanning of the grid (of c) leads z to cross the forbidden interval $[-1, 1]$, or to some other exceptional situation, the program should return a nice message, and continue the scanning without interrupt, with more fruitful values of c, so that nothing is lost. By the way, find out, how the system you work with, handles the logarithm and square root in the complex domain. It may not be entirely according to our conventions, but it almost certainly produces some value that your own program can modify appropriately.

(c) If $\Re c \gg 1$, the level curve for the real part is, close to a circle (why?). Use equidistant values of $\Im c \in [0, \frac{1}{2}\pi]$. The values of $\Re c$ are to be chosen so that the drawings become intellectually interesting and/or visually pleasing. You are then likely to find that the density of the level curves for the imaginary part, when they approach the interval $[0, 1]$ is different for the Chebyshev case and the equidistant case. Explain theoretically how this is related to the density function w in the text.

(d) The level curves of the imaginary part intersect the interval $[0, 1]$, at different angles in the Chebyshev and the equidistant cases. Give a theoretical analysis of this.

2. (a) (After Meray (1884) and Cheney, p. 65.) Let L_nf be the polynomial of degree $< n$ which interpolates to the function $f(z) = 1/z$ at the n'th roots of unity. Show that $(L_nf)(z) = z^{n-1}$. Show that $\lim_{n\to\infty} \max_{|u|=1} |(L_nf - f)(u)| > 0$.

Hint: Solve this directly, without the use of the previous theory.

(b) Modify the theory of Sec. 4.5.2 to this case with equidistant interpolation points on the unit circle, and make an application to $f(z) = 1/(z-a)$, $a > 0$, $a \neq 1$. Here, $\Phi_n(z) = z^n - 1$. What is $\psi(z)$, $P(z)$? The density function? (The integral for $\psi(z)$ is a little tricky, but you may find it in a table. There are, however, simpler alternatives to the integral, see the end of Sec. 4.5.2. Check your result by thinking like Faraday.) Find out for which values of a, u, $(|u| \neq 1, |u| \neq a)$, $(L_nf - f)(u) \to 0$, and estimate the speed of convergence (divergence).

(c) What can be said about the cases excluded above, i.e., $|u| = 1$, $|u| = a$? Also look at the case, when $|a| = 1$, $(a \neq 1)$.

(d) Is the equidistant interpolation on the unit circle identical to the Cauchy FFT method (with $a = 0$, $R = 1$) for the approximate computation of the coefficients in a power series ? See, in particular Eq. (3.1.10).

(e) We saw in b) that the equidistant interpolation on the unit circle gives no good polynomial approximation when the pole is inside the unit circle. The coefficients computed by the Cauchy FFT are however useful with a different interpretation, namely as coefficients in an interpolation polynomial $p(z^{-1})$ for $f(z) = 1/(z-a)$, or as approximate coefficients in a Laurent series $1/(z-a) = \sum_{j=1}^{\infty} c_j z^{-j}$, that converges for $|z > a|$. Note that the Cauchy integral, Eq. (3.1.8), is valid also for the coefficients of a Laurent expansion.
Now consider

$$f(z) = \frac{-5}{(3-z)(1-2z)} = \frac{1}{3-z} - \frac{2}{1-2z}.$$

This has three Laurent expansions, i.e., an ordinary Taylor series for $|z| < \frac{1}{2}$, an expansion into negative powers for $|z| > 3$, and a mixed expansion for the annulus $\frac{1}{2} < |z| < 3$. It is conceivable that the the first two expansions can be found by FFT, with different interpretations of the results, but what about the annulus case ? It is easily seen from the above partial fraction form of $f(z)$ what the Laurent expansion should be. When the FFT is applied to $f(z)$, it does therefore, in principle, find a coefficient by adding a coefficient of a *negative* power of z from the first term of the partial fraction decomposition, to the coefficient of a *positive* power of z from the second term. *Can this really work ?*
Explain, why things go so well with a careful treatment, in spite that I almost tried to convince you above that it would not work. Also try to formulate what "careful treatment" means in this case.

Hint: Generalize to the case of a Laurent expansion the relation between the FFT output and the series coefficients given (for a Taylor series) in Eq. (3.1.11). Also read in Strang [23] or somewhere else about "aliasing".

4.6 Trigonometric Interpolation and FFT.

Not included in this version. Part of it is found in Chapter 12.

4.6.1 Multidimensional Interpolation

4.6.2 Repeated One-Dimensional Interpolation

The simplest way to generalize interpolation to functions of several variables is to use repeated one-dimensional interpolation, i.e., to work with one variable at a time. The following formula for **bilinear interpolation**,

$$\begin{aligned} f(x_0+ph, y_0+qh) &\approx (1-q)\phi(y_0) + q\phi(y_0+k), \\ \phi(y) &= (1-p)f(x_0,y) + pf(x_0+h,y). \end{aligned}$$

is the simplest example. After simplification it can written as

$$\begin{aligned} f(x_0+ph, y_0+qh) \approx{}& (1-p)(1-q)f_{0,0} + p(1-q)f_{1,0} \\ &+ (1-p)qf_{0,1} + pqf_{1,1}, \end{aligned} \tag{4.6.1}$$

where we have used the notation $f_{ij} = f(x_0+ih, y_0+jk)$, $i,j \in \{0,1\}$. This formula is exact for functions of the form $f(x,y) = a + bx + cy + dxy$, and from equation 4.2.7 we obtain the error bound,

$$\max_{(x,y)\in R} \frac{1}{2}\big(p(1-p)h^2|f_{xx}| + q(1-q)h^2|f_{yy}|\big), \qquad 0 \le p,q \le 1,$$

where $R = \{(x,y) : x_0 \le x \le x_0+h, y_0 \le y \le y_0+k\}$. The formula for bilinear interpolation can easily be generalized by using higher order interpolation in the x and/or y direction.

In the following we consider explicitly only the case of two dimension, since corresponding formulas for three and more dimensions are analogous.

4.6.3 Rectangular Grids

A **rectangular grid** in the (x,y)-plane with grid spacings h,k in the x and y directions, respectively, consists of points $x_i = x_0 + ih$, $y_i = y_0 + ik$. In the following we use the notation $f(x_i,y_j) = f_{ij}$. (For interpolation formulas valid for irregular triangular grids, see Sec. 8.4.3)

Central difference approximations for partial derivatives using function values can be obtained by working with one variable at a time,

$$\frac{\partial f}{\partial x} = \frac{1}{2h}(f_{i+1,j} - f_{i-1,j}) + O(h^2), \qquad \frac{\partial f}{\partial y} = \frac{1}{2k}(f_{i,j+1} - f_{i,j-1}) + O(k^2).$$

For second order derivatives

$$\frac{\partial^2 f}{\partial x^2} = \frac{1}{h^2}(f_{i+1,j} - 2f_{ij} + f_{i-1,j}),$$

and a similar formula holds for $\partial^2 f/\partial y^2$.

Formulas of higher accuracy can also be obtained by operator techniques, based on an operator formulation of Taylor's expansion (see Theorem 4.6.6,

$$f(x_0+h, y_0+k) = \exp\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) f(x_0,y_0) \tag{4.6.2}$$

From this we obtain

$$f(x_0+h,y_0+k) = f_{0,0} + \Big(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\Big)f_{0,0}$$
$$+ \Big(h^2\frac{\partial^2}{\partial x^2} + 2hk\frac{\partial^2}{\partial x\partial y} + k^2\frac{\partial^2}{\partial y^2}\Big)f_{0,0} + O(h^2+k^2).$$

An interpolation formula valid for all quadratic functions can be obtained by replacing in Taylor's formula the derivatives by difference approximations valid for quadratic polynomials,

$$\begin{aligned} f(x_0+ph,y_0+qh) \approx\; & f_{0,0} + \frac{1}{2}p(f_{1,0}-f_{-1,0}) + \frac{1}{2}q(f_{0,1}-f_{0,-1}) \\ & + \frac{1}{2}p^2(f_{1,0}-2f_{0,0}+f_{-1,0}) \\ & + \frac{1}{4}pq(f_{1,1}-f_{1,-1}-f_{-1,1}+f_{-1,-1}) \\ & + \frac{1}{2}q^2(f_{0,1}-2f_{0,0}+f_{0,-1}). \end{aligned} \qquad (4.6.3)$$

This formula uses function values in nine points. (The proof of the expression for approximating the mixed derivative $\dfrac{\partial^2}{\partial x\partial y}f_{0,0}$ is left as an exercise, Problem 2.

Cubic Hermite interpolation treating one variable at a time (see Example 4.6.3):

$$\begin{aligned} f(x_0+ph,y_0+qh) &= (1-q)\phi(y_0) + q\phi(y_1) \\ &+q(1-q)\Big[(1-q)(h\phi_y'(y_0)-\Delta\phi(y_0)) - q(h\phi_y'(y_1)-\Delta\phi(y_0))\Big]. \end{aligned}$$

$$\begin{aligned} \phi(y) &= (1-p)f(x_0,y) + pf(x_1,y) \\ &+p(1-p)\Big[(1-p)(hf_x'(x_0,y)-\Delta f(x_0,y)) - p(hf_x'(x_1,y)-\Delta f(x_0,y))\Big]. \end{aligned}$$

This formula requires that the quantities $f, \partial f/\partial x, \partial f/\partial y$, and $\partial^2 f/\partial x\partial y$ are given at the four points (x_i,y_j) for $0\le i,j\le 1$. Hence 16 quantities are needed to specify the bicubic polynomial.

Review Questions

1. How is bilinear interpolation performed? What is the order of accuracy?

Problems

1. Compute by bilinear interpolation $f(0.5, 0.25)$ when

$$f(0,0)=1, \quad f(1,0)=2, \quad f(0,1)=3, \quad f(1,1)=5.$$

2. Derive a formula for $f''_{xy}(0,0)$ using f_{ij}, $|i| \le 1, |j| \le 1$, which is exact for all quadratic functions.

4.7 Examples of Interpolation in Nonlinear Function Spaces

A little about interpolation by rational functions and by a sum of exponentials. Not yet written.

Notes and References

The basic idea of scaled divided differences is due to F. Krogh [16]. The notation and presentation used in Sec. 4.2.2 is due to L. O. Eriksson and G. Dahlquist [9]

Runge's example is from 1901 (?), see the bibliography in Cheney [12].

[1] C. de Boor. On calculating with B-splines. *J. Approx. Theory*, 6:50–62, 1972.

[2] C. de Boor. *Spline Toolbox for use with* MATLAB. The Math. Works, South Natick, 1990.

[3] C. de Boor. *A Practical Guide to Splines.* Revised ed., Springer-Verlag, Berlin, 1991.

[4] M. G. Cox. The numerical evaluation of B-splines. *J. Inst. Math. Applics.*, 10:134–149, 1972.

[5] M. G. Cox. Algorithms for spline curves and surfaces. Tech. Report DITC 166/90, National Physical Laboratory, Teddington, U.K., 1990.

[6] P. Dierckx. *FITPACK User Guide part I: Curve Fitting Routines.* TW Report 89, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium.

[7] P. Dierckx. *FITPACK User Guide part II: Surface Fitting Routines.* TW Report 122, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium.

[8] P. Dierckx. Curve and Surface Fitting with Splines. Clarendon Press, New York, 1993.

[9] L. O. Eriksson and G. Dahlquist.

[10] G. Farin. Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide. Academic Press, New York, 1988.

[11] C.-E. Fröberg. *Numerical Mathematics. Theory and Computer Applications.* The Benjamin/Cummings, Menlo Park, CA, 1985.

[12] E. W. Cheney. *Introduction to Approximation Theory.* McGraw-Hill, New York, NY, 1966.

[13] H. H. Goldstine. *The Computer from Pascal to von Neumann.* Princeton, 1972.

[14] P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1962.

[15] E. Isaacson and H.B. Keller. *Analysis of Numerical Methods*, Dover, New York Corrected reprint of 1966 original.

[16] F. T. Krogh. A variable step variable order multistep method for the numerical solution of ordinary differential equations. In A. J. Morell, editor, *Proceedings of the IFIP Congress 1968*, pages 194–199. North-Holland, Amsterdam, 1969.

[17] M. J. D. Powell. *Approximation Theory and Methods.* Cambridge University Press, Cambridge, UK, 1981.

[18] H. Prautzsch, W. Boehm, M. Paluszny. Bézier and B-spline techniques Springer-Verlag, Berlin, 2002.

[19] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4:45–99 and 112–141., 1946.

[20] I. J. Schoenberg and A. Whitney, On Pólya frequency functions III: The positivity of translation determinants with an application to the interpolation problem by spline curves. Trans. Amer. Math. Soc., 74, 246–259, 1953.

[21] H. R. Schwarz. *Numerical Analysis. A Comprehensive Introduction.* John Wiley, New York, Ny, 1989.

[22] H. Schwetlick and T. Schütze. Least squares approximation by splines with free knots. BIT, 35 (1995), 361–384.

[23] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, USA, 1986.

[24] G. Szegö. *Orthogonal Polynomials*, 4th ed.. Amer. Math. Soc. Colloq. Publ., vol. 23, Amer. Math. Soc., Providence, RI.

Chapter 5

Approximate Computation of Linear Functionals

5.1 Introduction

The problem considered in this chapter is to compute an approximation to a definite integral

$$I = \int_a^b f(x)\,dx \tag{5.1.1}$$

by using values of f, and possibly some its derivatives. As is well known, even many relatively simple integrals cannot be expressed in terms of elementary functions, and must be evaluated by numerical methods. An important classical example is the error function defined by

$$\mathrm{erf}(b) = \frac{2}{\sqrt{\pi}} \int_0^b e^{-x^2}\,dx,$$

and used in many branches of mathematics (see Problem 7, Sec. 3.1).

The above problem is known as numerical **quadrature**, which relates to the ancient problem, the quadrature of the circle, i.e., constructing a square with equal area to that of a circle. The name was then also used for more general problems of finding areas and volumes.

Frequently f is assumed to be known on a grid of equidistant points. Several classical formulas dating back to Newton (1642–1727) are described in Section 5.2. By a careful choice of grid points the order of accuracy of the quadrature rule can substantially improved. The construction of such Gaussian quadrature rules are based on the theory of orthogonal polynomials, and the treatment of these are therefore delayed until Section 9.4. Some special techniques for integrals with singularities of different kinds are given in Section 5.3. Finally, methods for approximating multiple integrals, where f is known on a rectangular or triangular grid, are treated in Section 5.4.

5.2 Classical Quadrature Rules

5.2.1 The Trapezoidal and Midpoint Rules

One obvious way to solve the above problem is to approximate $f(x)$ by a simple, easily integrated function. We now study two methods which are based on approximating the function $f(x)$ by piecewise constants and linear functions respectively. We assume f to be known on a grid of equidistant points

$$x_0 = a, \quad x_i = x_0 + ih, \quad x_n = b. \tag{5.2.1}$$

where $h = (b-a)/n$ is the step length. We will use the notation $f_i = f(x_i)$.

In the **midpoint rule** approximate the integral (5.1.1) by

$$\hat{R} = h \sum_{i=0}^{n-1} f_{i+1/2}, \tag{5.2.2}$$

see Fig. 5.2.1. (Compare the above approximation with the Riemann sum in the *definition* of a definite integral.) Notice that (5.2.2) uses the values of f at the *midpoints* of each subinterval (x_{i-1}, x_i). This is essential for good accuracy, as can be seen from Fig. 5.2.1.

The **trapezoidal rule** was mentioned already in Sec. 1.2. Here we use the approximation

$$\hat{T} = h \sum_{i=1}^{n} \frac{1}{2}(f_{i-1} + f_i) = \frac{h}{2} f_0 + h \sum_{i=2}^{n-1} f_i + \frac{h}{2} f_n, \tag{5.2.3}$$

see Fig. 5.2.1. This formula is based on piecewise linear interpolation in the intervals (x_{i-1}, x_i), $i = 1, 2, \ldots, n$.

The trapezoidal rule is called a **closed rule** because values of f at both endpoints are used. It is not uncommon that f has an integrable singularity at an endpoint. In that case an **open rule**, like the midpoint rule, still can be applied.

Figure 5.2.1.

It is easy to get strict estimates for the discretization error of both the midpoint and the trapezoidal rule. Suppose that f'' is continuous in $[a, b]$. For one subinterval of the trapezoidal rule, we have from the remainder term in interpolation in (4.2.6)

$$\epsilon_i = \hat{T}(h) - \int_{x_{i-1}}^{x_i} f(x)\,dx = -\int_{x_{i-1}}^{x_i} \frac{f''(\xi)}{2}(x - x_{i-1})(x - x_i)\,dx,$$

where $\xi \in [x_{i-1}, x_i]$ depends on x. Since $(x - x_{i-1})(x - x_i) < 0$ for $x \in [x_{i-1}, x_i]$ we can apply a generalized form of the mean-value theorem of integral calculus, and it follows that

$$\epsilon_i = -\frac{1}{2} f''(\xi_i) \int_{x_{i-1}}^{x_i} (x - x_{i-1})(x - x_i)\,dx, \qquad \xi_i \in [x_{i-1}, x_i].$$

Setting $x = x_{i-1} + ht$, we get

$$\epsilon_i = -\frac{1}{2} f''(\xi_i) \int_0^1 hth(t-1)h\,dt = -\frac{h^3 f''(\xi_i)}{12}.$$

The **global** truncation error is just the sum of these **local** truncation errors. Hence for the trapezoidal rule

$$R_T = \sum_{i=1}^{n} \epsilon_i = -n\frac{h^3}{12} f''(\xi) = -\frac{(b-a)h^2}{12} f''(\xi), \quad \xi \in [a,b]. \tag{5.2.4}$$

For the midpoint rule using the remainder term in Taylor's formula we have

$$f(x) = f_{i-\frac{1}{2}} + (x - x_{i-\frac{1}{2}}) f'_{i-\frac{1}{2}} + \frac{1}{2}(x - x_{i-\frac{1}{2}})^2 f''(\xi_i), \qquad \xi_i \in [x_{i-1}, x_i].$$

Noting that the integral over $[x_{i-1}, x_i]$ of the second term vanishes we get analogously

$$\epsilon_i = \frac{1}{2} f''(\xi_i) \int_{-\frac{1}{2}}^{\frac{1}{2}} h^3 t^2\,dt = \frac{h^3 f''(\xi_i)}{24},$$

and thus

$$R_T = \frac{(b-a)h^2}{24} f''(\xi), \quad \xi \in [a,b]. \tag{5.2.5}$$

Hence also the midpoint rule is exact when $f(x)$ is a linear function. The midpoint rule can also be interpreted as the area of the trapezium defined by the tangent of f at the midpoint $x_{i-\frac{1}{2}}$. Note that the error is roughly half the size of the error in the trapezoidal rule and of opposite sign.

Note that the error in the midpoint rule is half as large as that for the trapezoidal rule and has the opposite sign. However, the trapezoidal rule is more economical to use when a sequence of approximations for $h, h/2, h/4, \dots$ is to be computed, since about half of the values needed for $h/2$ were already computed and used for h, etc. indeed, it is easy to verify the following useful relation between the trapezoidal and midpoint rules:

$$\hat{T}(h/2) = \frac{1}{2}(\hat{T}(h) + \hat{R}(h)). \tag{5.2.6}$$

If U denotes one unit in the last decimal place of the values of $f(x)$ then for both formulas the resulting rounding error R_A in the approximations satisfies

$$|R_A| \leq hn\frac{1}{2}U = (b-a)\frac{1}{2}U. \tag{5.2.7}$$

If the rounding error is negligible and h sufficiently small, then it follows from (5.2.4) that the error in $T(h/2)$ is about 1/4-th of that in $T(h)$. Hence the magnitude of the error in $T(h/2)$ can be estimated by $\frac{1}{3}|T(h/2) - T(h)|$, or more conservatively by $|T(h/2) - T(h)|$.

Example 5.2.1.

Compute approximately $\int_0^{0.8} \frac{\sin x}{x}\,dx$. As an exercise the reader should check some of the midpoint and trapezoidal sums given below, which are correct to ten decimals. (Use (5.2.6).)

h	$\hat{R}(h)$	$\hat{T}(h)$
0.8	0.77883 66846	0.75867 80454
0.4	0.77376 69772	0.76875 73650
0.2	0.77251 27162	0.77126 21711
0.1		0.77188 74437

The correct value, to six decimals, is 0.772096. Verify that in this example the error is approximately proportional to h^2 for both $\hat{R}(h)$ and $\hat{T}(h)$. We estimate the error in $T(0.1)$ to be $\frac{1}{3}6.26 \cdot 10^{-4} \leq 2.1 \cdot 10^{-4}$.

5.2.2 Simpson's Rule

One way to get a more accurate approximation of the integral (5.1.1) is to approximate $f(x)$ by a piecewise polynomial of higher degree. Consider the integral

$$\int_{x_{i-1}}^{x_{i+1}} f(x)\,dx.$$

According to Taylor's formula we have

$$f(x) = f_i + (x-x_i)f_i' + \frac{(x-x_i)^2}{2}f_i'' + \frac{(x-x_i)^3}{3!}f_i''' + O(h^4),$$

where the remainder is zero for all polynomials of degree 3 or less. Integrating term by term, the integrals of the second and fourth term vanishes and we obtain

$$\int_{x_{i-1}}^{x_{i+1}} f(x)\,dx = 2hf_i + 0 + \frac{1}{3}h^3 f_i'' + 0 + O(h^5).$$

Using $h^2 f_i'' = (f_{i-1} - 2f_i + f_{i+1}) + O(h^4)$ (see (4.7.5)) we have that

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+1}} f(x)\,dx &= 2hf_i + \frac{1}{3}h(f_{i-1} - 2f_i + f_{i+1}) + O(h^5) \qquad (5.2.8)\\ &= \frac{1}{3}h(f_{i-1} + 4f_i + f_{i+1}) + O(h^5),\end{aligned}$$

where the remainder term is zero for all third-degree polynomials. The above formula, **Simpson's rule**[1] and is one of the most famous classical formulas for numerical integration.

To determine the truncation error, we first determine R_T for $f(x) = (x-x_i)^4$:

$$R_T = \frac{1}{3}h(h^4 + 0 + h^4) - \int_{x_{i-1}}^{x_{i+1}} x^4\,dx = h^5(\frac{2}{3} - \frac{2}{5}) = h^5\frac{4}{15},$$

from which follows the *asymptotic* error estimate

$$R_T = h^5\frac{4}{15}\frac{f^{(4)}(x_i)}{4!} + O(h^6) = \frac{h^5}{90}f^{(4)}(x_i) + O(h^6).$$

One can prove the following strict error estimate $R_T = \frac{h^5}{90}f^{(4)}(\xi_i)$, $\xi \in [x_{i-1}, x_{i+1}]$.

In the practical use of Simpson's formula for computing the integral (5.1.1), one divides the interval $[a,b]$ into an even number $n = 2m$ steps of length h, and use the formula (5.2.8) on each of m double steps. This compounded form of Simpson's rule can then be written

$$\int_a^b f(x)\,dx = \frac{h}{3}(f_0 + 4U + 2E + f_n) + R_T,$$

where

$$U = f_1 + f_3 + \cdots + f_{n-1}, \qquad E = f_2 + f_4 + \cdots + f_{n-2},$$

and the remainder is

$$R_T = \sum_{i=0}^{m-1} \frac{h^5}{90}f^{(4)}(\xi_i) = \frac{(b-a)}{180}h^4 f^{(4)}(\xi), \qquad \xi \in [a,b]. \tag{5.2.9}$$

5.2.3 Newton-Cotes' Formulas

Let $w(x)$ be a given nonnegative integrable function. We want to approximately compute the definite integral

$$I = \int_a^b \omega(x)f(x)\,dx \tag{5.2.10}$$

when $m+1$ function values $f_i = f(x_i)$ are given, $a \le x_0 < x_1 < \cdots < x_m \le b$. The unique polynomial $p(x)$ interpolating $f(x)$ at these points is given by the Lagrange interpolation formula (Theorem 4.3.4),

$$p(x) = \sum_{i=0}^{m} f(x_i)L_i(x), \qquad L_i(x) = \prod_{\substack{j=0\\ j\neq i}}^{m} \frac{(x-x_j)}{(x_i-x_j)}.$$

[1]Due to Thomas Simpson (1710–1761).

If we then approximate the integral I by $\int_a^b \omega(x)p(x)\,dx$ we obtain the quadrature rule

$$\int_a^b f(x)\omega(x)\,dx \approx C_0f_0 + C_1f_1 + \ldots + C_mf_m, \qquad C_i = \int_a^b L_i(x)\omega(x)\,dx. \quad (5.2.11)$$

By construction *this formula is exact when $f(x)$ is a polynomial of degree m.* The truncation error is obtained by integrating the remainder (see Theorem 4.2.2)

$$R = \frac{1}{(m+1)!}\int_a^b \Phi(x)f^{(m+1)}(\xi_x)\omega(x)\,dx, \quad \Phi(x) = (x-x_0)(x-x_1)\cdots(x-x_m).$$

The coefficients C_i depend only on the weight function $\omega(x)$ and on the distribution of the points $\{x_i\}_{i=0}^m$. In practice, the coefficients are more easily computed using the method of undetermined coefficients rather than by integrating $L_i(x)$.

Example 5.2.2.

Derive a formula

$$\frac{1}{\sqrt{2h}}\int_0^{2h} x^{-1/2}f(x)\,dx \approx C_0f(0) + C_1f(h) + C_2f(2h)$$

which is exact when $f(x)$ is any second-degree polynomial. Equating the left and right hand sides for $f(x) = 1, x, x^2$ we obtain

$$2 = C_0 + C_1 + C_2, \qquad \frac{2}{3} = \frac{1}{2}C_1 + C_2, \qquad \frac{2}{5} = \frac{1}{4}C_1 + C_2.$$

Hence $C_0 = \frac{12}{15}$, $C_1 = \frac{16}{15}$, and $C_2 = \frac{2}{15}$.

In particular, for $\omega(x) = 1$ and equidistant points $x_i = a + ih$, $i = 0, 1, \ldots, n$, $h = (b-a)/n$, we obtain rules called the **closed Newton-Cotes' quadrature rules**.[2] With $x = a + sh$ we obtain for the closed rules

$$L_i = \frac{s(s-1)\cdots(s-i+1)(s-i-1)\cdots(s-m)}{i(i-1)\cdots1(-1)\cdots(i-m)},$$

and

$$\int_{x_0}^{x_m} p(x)\,dx = h\sum_{i=0}^{m} f_i\int_0^m L_i(s)ds.$$

Example 5.2.3.

Taking $n = 1$ we we find again the trapezoidal formula; for $n = 2$ we have

$$C_0 = \int_0^2 \frac{(s-1)(s-2)}{(-1)(-2)}ds = \frac{1}{3} = C_2, \qquad C_1 = \int_0^2 \frac{s(s-2)}{1(-1)}ds = \frac{4}{3},$$

[2]Named after Roger Cotes (1682–1716)

giving Simpson's rule. As shown above, because of symmetry, this is exact for all polynomials of degree three, although an interpolation polynomial of only degree two was used to derive it.

It can be shown in general that the closed Newton-Cotes formulas are exact for polynomials of degree $m+1$ for m even and degree m for m odd, see Isaacson and Keller [1966, Sec. 7.1.1]. Newton-Cotes' formulas up to order $m = 10$ with error terms are listed in Abramowitz and Stegun [1971, pp.886–887]. For $m \leq 7$ the coefficients C_i are positive; however for $m = 8$ and $m \geq 10$ negative coefficients appear. Such formulas may some times be useful, but are less robust with respect to rounding errors.

The **open Newton-Cotes rules** use equidistantly distributed points $x_1, \ldots, x_{m-1}$ in the *interior* of $[a,b]$. The simplest open Newton-Cotes rule for $m = 2$ is equivalent to the midpoint rule with step length $2h$. The open formulas have polynomial order $m-1$ for m even and $m-2$ for m odd. For the open formulas negative coefficients occur already for $m = 4$.

5.2.4 Adaptive Quadrature Methods

The quadrature methods considered so far all use the same step length over the interval of integration. This may not be efficient when, for example, $f(x)$ has greatly different magnitude in different parts of the interval $[a,b]$. Adaptive quadrature methods automatically adapts step sizes so that the approximation I satisfies a prescribed error tolerance

$$|I - \int_a^b f(x)\,dx| \leq \epsilon. \tag{5.2.12}$$

We first remark that evaluation of the integral (5.1.1) is equivalent to solving

$$\frac{dy}{dx} = f(x), \qquad y(a) = 0, \tag{5.2.13}$$

and taking $I = y(b)$. This is a special case of an initial value problem for an ordinary differential equation, and the methods described in Chapter 13 can be used to solve the problem (5.2.13). These algorithms have been developed to include sophisticated techniques for adaptively choosing step size and order in the integration (see Sec. 13.2), and may therefore be a good choice for handling difficult cases.

For simplicity we consider here only a *fixed order* adaptive method, which is based on Romberg's method. For a subinterval $[a_j, b_j]$, put $h_j = (b_j - a_j)$ and compute the trapezoidal approximations

$$T_{00} = \hat{T}(h_j), \quad T_{10} = \hat{T}(h_j/2), \quad T_{20} = \hat{T}(h_j/4),$$

and the extrapolated values T_{11} and T_{21}, which are equivalent to Simpson's rule see Sec. 5.2.2. For the approximation $I_j = T_{21}$ we have the error estimate $R_j = |T_{21} - T_{11}|$. We *accept* the approximation I_j if

$$|T_{21} - T_{11}| < \frac{h_j \epsilon}{b-a}, \tag{5.2.14}$$

that is we require the error to be *less than $\epsilon/(b-a)$ per unit step.* Otherwise we *reject* the approximation, and subdivide the interval in two intervals $[a_j, \frac{1}{2}(a_j+b_j)]$, $[\frac{1}{2}(a_j+b_j), b_j]$. The same rule is now applied to these two subintervals. Note that if we have saved the function values computed previously for T_{20} these can be reused. We start with one interval $[a,b]$ and carry on subdivisions until the error criterion in (5.2.14) is satisfied for all intervals. Since the total error is the sum of errors for all subintervals we then have the error estimate

$$R_T < \sum_j \frac{h_j \epsilon}{b-a} = \epsilon$$

as required.

In many situations it might be preferable to specify a *relative error tolerance*

$$\big|I - \int_a^b f(x)\,dx\big| / \big|\int_a^b f(x)\,dx\big| \le \epsilon.$$

This is complicated by the fact that the denominator might be zero. Hence sometimes a combination of relative and absolute criterion is used.

Many variations on the simple scheme outlined above are possible. For example, we could base the method on a higher order Romberg scheme, or even try to choose an optimal order for each subinterval. Adaptive methods work even when the integrand $f(x)$ is badly behaved. However, if f has singularities or unbounded derivatives, the error criterion may never be satisfied. For such cases it is necessary to include some bound of the number of subdivisions allowed. It should be kept in mind that although adaptive quadrature algorithms are convenient to use they are in general less efficient than methods which have been specially adapted for a particular problem.

A collection of computer subroutines for adaptive quadrature is given by Piessens et al. [2]. We finally warn the reader that *no automatic quadrature routine can be guaranteed always to work.* The integrand $f(x)$ may, for example, be nonzero only on a small subset of $[a,b]$. Since any quadrature rule only samples $f(x)$ in a finite number of points an automatic routine theoretically may return the value zero in such a case!

Review Questions

1. Give an account of the theoretical background of Romberg's method and its use.

Problems

1. (a) Derive the closed Newton-Cotes rule for $m = 3$,

$$I = \frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3) + R_T, \qquad h = (b-a)/3,$$

also known as Simpson's (3/8)-rule.

(b) Derive the open Newton-Cotes rule for $m = 4$,

$$I = \frac{4h}{3}(2f_1 - f_2 + 2f_3) + R_T, \qquad h = (b-a)/4.$$

(c) Find asymptotic error estimates for the formulas in (a) and (b) by applying them to suitable polynomials.

2. Use Romberg's method to compute the integral $\int_0^4 f(x)\,dx$, using the following (correctly rounded) values of $f(x)$. Need all the values be used?

x	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
$f(x)$	−4271	−2522	−499	1795	4358	7187	10279	13633	17247

3. Compute the integral

$$\frac{1}{2\pi}\int_0^{2\pi} e^{\frac{1}{\sqrt{2}}\sin x}\,dx$$

by the trapezoidal rule, using $h = \pi/2$ and $h = \pi/4$.

4. Gregory's quadrature formula (James Gregory (1638–1675)) has the form

$$\int_{x_0}^{x_0+nh} y\,dx = h\frac{E^n - 1}{hD}f_0 = h\left(\frac{f_n}{-\ln(1-\nabla)} - \frac{f_0}{\ln(1+\Delta)}\right)$$
$$= h\Big(T - c_2(\nabla^2 f_n + \Delta^2 f_0) - c_4(\nabla^4 f_n + \Delta^4 f_0) - c_6(\nabla^6 f_n + \Delta^6 f_0) - \cdots\Big),$$

where T is the trapezoidal sum. Determine the coefficients c_2, c_4, and c_6 so that the formula is exact for polynomials of as high degree as possible.

5. (a) Show that the trapezoidal rule, with $h = 2\pi/(n+1)$, is exact for all trigonometric polynomials of period 2π—i.e., for functions of the type

$$\sum_{k=-n}^{n} c_k e^{ikt}, \qquad i^2 = -1.$$

—when it is used for integration over a whole period.

(b) Show that if $f(t)$ can be approximated by a trigonometric polynomial of degree n so that the magnitude of the error is less than ϵ, in the interval $(0, 2\pi)$, then the error with the use of the trapezoidal rule with $h = 2\pi/(n+1)$ on the integral $(2\pi)^{-1}\int_0^{2\pi} f(t)\,dt$ is less than 2ϵ.

(c) Use the above to explain the sensationally good result in Problem 2 above, when $h = \pi/4$.

Hint: First use Theorem 5.4.1 to determine how well the function $g(x) = e^{x/\sqrt{2}}$ can be approximated by a seventh-degree algebraic polynomial for $x \in [-1, 1]$.

6. Euler's constant is defined by $\gamma = \lim_{N\to\infty} F(N)$, where

$$F(N) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N-1} + \frac{1}{2N} - \log N.$$

(a) Use the Euler-Maclaurin formula with $f(x) = x^{-1}$, $h = 1$, to show that, for any integer M

$$\gamma = F(M) + \frac{1}{12}M^{-2} - \frac{6}{720}M^{-4} + \frac{120}{30240}M^{-6} - \ldots,$$

where every other partial sum is larger than γ, and every other is smaller.

(b) Compute γ to seven decimal places, using $M = 10$, $\sum_{n=1}^{10} n^{-1} = 2.92896825$, $\ln 10 = 2.30258509$.

(c) Show how Richardson extrapolation can be used to compute γ from the following values:

M	1	2	4	8
$F(M)$	0.5	0.55685	0.57204	0.57592

5.3 Special Transformations

The extent to which Romberg's method is successful depends on how well the function can be locally approximated by a polynomial. It is often profitable to investigate whether or not one can transform or modify the given problem in some way so that the resulting integrand is more suitable for numerical integration.

5.3.1 Integrals with Singularities

If the integrand becomes infinite at a point, such a modification is *necessary.* Even if some low-order derivative of the function is infinite at some point in or near the interval of integration, one *should* make such a modification. It is not uncommon that a single step taken close to a point where, for example, the derivative of the integrand is infinite gives a larger error than all other steps combined, when using a constant step-size.

We first consider a few cases where the integrand has a **singularity** or is "almost singular".

Example 5.3.1. (Substitution)

Consider $I = \displaystyle\int_0^1 \frac{1}{\sqrt{x}} e^x\, dx$. The integrand is infinite at the origin. By the substitution $x = t^2$ we get $I = \int_0^1 e^{t^2} 2\,dt$. This integral can be treated without difficulty by, for example Romberg's method.

The integral above could also be evaluated by using a special Gauss quadrature rule for the weight function $\omega(x) = 1/\sqrt{x}$. With the midpoint rule and Romberg's method, a sufficient condition that the method converges as $h \to 0$ is that the integrand be continuous, but to get rapid convergence more is required.

Example 5.3.2. (Integration by parts)

$$\begin{aligned} I &= \int_0^1 x^{-1/2} e^x\, dx = 2x^{1/2}e^x\big|_0^1 - 2\int_0^1 x^{1/2} e^x\, dx \\ &= 2e - 2\frac{2}{3}x^{3/2}e^x\big|_0^1 + \frac{4}{3}\int_0^1 x^{3/2}e^x\, dx = \frac{2}{3}e + \frac{4}{3}\int_0^1 x^{3/2} e^x\, dx. \end{aligned}$$

The last integral has a mild singularity at the origin. If one wants high accuracy, then it is advisable to integrate by parts a few more times before the numerical treatment.

Example 5.3.3. (Simple Comparison Problem)

In $I = \int_{0.1}^{1} x^{-3}e^x\,dx$ the integrand is infinite near the left end point. If we write

$$I = \int_{0.1}^{1} x^{-3}\Big(1 + x + \frac{x^2}{2}\Big)\,dx + \int_{0.1}^{1} x^{-3}\Big(e^x - 1 - x - \frac{x^2}{2}\Big)\,dx$$

the first integral can be computed analytically. The second integrand can be treated numerically. The integrand and its derivatives are of moderate size. Note, however, the cancellation in the evaluation of the integrand.

For integrands of the form $\omega(x)f(x)$, where $f(x)$ is well suited for local approximation by a polynomial, we can derive special quadrature methods of Newton-Cotes or Gauss types for such integrals (see Sec. 5.2.3 and Sec. 5.3.1).

If the integrand is **oscillating**, then with ordinary integration methods one must choose a step size which is small with respect to the wave length; this is often an irritating limitation in many applications. The techniques previously mentioned (simple comparison problem, special integration formula, etc.) are sometimes effective in such situations. In addition, the following method can be used on integrals of the form

$$I = \int_0^\infty f(x)\sin(g(x))\,dx,$$

where $g(x)$ is an increasing function, and both $f(x)$ and $g(x)$ can be approximated by a polynomial. Set

$$I = \sum_{n=0}^{\infty}(-1)^n u_n, \qquad u_n = \int_{x_n}^{x_{n+1}} f(x)|\sin(g(x))|\,dx,$$

where $x_0, x_1, x_2, \ldots$ are the successive zeros of $\sin(g(x))$. The convergence of this alternating series can then be improved with the help of repeated averaging, see Sec. 3.2.1.

5.3.2 Infinite Intervals

Infinite intervals of integration occur often in practical problems. For integrals of the form $\int_{-\infty}^{\infty} f(x)\,dx$, *the trapezoidal rule or the midpoint rule often give surprisingly good accuracy* if one integrates over the interval $[-R_1, R_2]$, assuming that $f(x)$ and its lower derivatives are small for $x \le -R_1$ and $x \ge R_2$.

Example 5.3.4.

Compute $\int_{-\infty}^{\infty} e^{-x^2}\,dx$. For $|x| \ge 4$, the integrand is less than $\frac{1}{2}\cdot 10^{-6}$. Using the trapezoidal rule for the integral over $[-4, 4]$ we get the estimate 1.772636 with $h = 1$ and 1.772453 with $h = 0.5$. (The values for the function are correct up to six decimal places.) The exact value of the integral is $\pi^{1/2} = 1.772454$. The truncation error in the value of the integral is less than 10^{-5} times the truncation error in the largest term of the trapezoidal sum—a superb example of "cancellation of truncation errors"!

The error which is committed when we replace ∞ by 4 can be estimated in the following way:

$$\begin{aligned}|R| &= 2\int_4^\infty e^{-x^2}\,dx = 2\int_{16}^\infty e^{-t}\frac{1}{2}t^{-1/2}\,dt\\ &< 2\cdot\frac{1}{2}16^{-1/2}\int_{16}^\infty e^{-t}\,dt = \frac{1}{4}e^{-16} < 10^{-7}.\end{aligned}$$

One can also try some substitution which maps the interval $(0,\infty)$ to $(0,1)$—e.g., $t = e^{-x}$ of $t = 1/(1+x)$. However, in such cases one must be careful not to introduce an unpleasant singularity into the integrand instead. Take, for example, $I = \int_0^\infty (1+x^2)^{-4/3}\,dx$ and make the substitution $t = 1/(1+x)$. Then we get $I = \int_0^1 (t^2+(1-t)^2)^{-4/3}t^{2/3}\,dt$. he integrand now has an infinite derivative at the origin. This can be eliminated by making the substitution $t = u^3$, to get

$$I = \int_0^1 (u^6 + (1-u^3)^2)^{-4/3}3u^4du,$$

which can be computed with, for example, Romberg's method.

Review Questions

1. Give an account of Gauss quadrature formulas: accuracy, how the points are determined, some important property of the weights.

Problems

1. Compute the integral $\displaystyle\int_0^\infty (1+x^2)^{-4/3}\,dx$ with five correct decimals. Expand the integrand in powers of x^{-1} and integrate term-wise over the interval $[R,\infty]$, for a suitable value of R. Then use Romberg's method on the remaining interval $[0,R]$.
2. Propose a suitable plan (using a computer) for computing the following integrals, for $s = 0.5, 0.6, 0.7, \dots, 3.0$:

   (a) $\displaystyle\int_0^\infty (x^3+sx)^{-1/2}\,dx$; (b) $\displaystyle\int_0^\infty (x^2+1)^{-1/2}e^{-sx}\,dx$, error $< 10^{-6}$;

   (c) $\displaystyle\int_\pi^\infty (s+x)^{-1/3}\sin x\,dx$.

3. For expressing integrals appearing in the solution of certain integral equations the following modification of the midpoint rule is often used:

$$\int_{x_0}^{x_n} K(x_j, t)y(t)\,dt = \sum_{i=0}^{n-1} m_{ij} y_{i+1/2},$$

where $y_{i+1/2} = y(\frac{1}{2}(t_i + t_{i+1}))$ and m_{ij} is the moment integral

$$m_{ij} = \int_{t_i}^{t_{i+1}} K(x_j, t)\,dt.$$

Derive an error estimate for this formula.

5.4 Multiple Integrals

5.4.1 Introduction

The ideas of numerical quadrature can be generalized to multiple integrals, but the amount of work will increase rapidly with the number of dimensions. It is therefore advisable to try to reduce the number of dimensions by applying analytic techniques to parts of the task.

Example 5.4.1.

The following triple integral can be reduced to a single integral:

$$\int_0^\infty \int_0^\infty \int_0^\infty e^{-(x+y+z)} \sin(xz)\sin(yz)\,dxdydz$$

$$\int_0^\infty e^{-x}\,dx \int_0^\infty e^{-y}\sin(yx)dy \int_0^\infty e^{-z}\sin(zx) = \int_0^\infty \left(\frac{x}{1+x^2}\right)^2 e^{-x}\,dx,$$

because

$$\int_0^\infty e^{-z}\sin(zx)dz = \int_0^\infty e^{-y}\sin(yx)dz = \frac{x}{1+x^2}.$$

The remaining single integral is simply evaluated by the techniques previously studied.

Often a transformation of variable is needed for such a reduction (see Problem 1 at the end of this section), but sometimes that does not help either. Several approaches are then possible:

(a) numerical integration in one direction at a time—see Sec. 5.4.2;

(b) the use of a rectangular grid, mainly if the boundary of the region is composed of straight lines–see Sec. 5.4.3.

(c) the use of an irregular triangular grid—possible for more general boundaries—see Sec. 5.4.4.

(d) Monte-Carlo methods, mainly for problems with complicated boundaries and a large number of dimensions—see Sec. 5.4.5.

5.4.2 Successive One-Dimensional Quadrature

For simplicity we restrict ourselves below to the two-dimensional case, although the ideas are more general. Consider the integral

$$I = \int_D f(x,y)\,dxdy \tag{5.4.1}$$

where D is a domain in the x-y plane. The simplest way to compute an approximation to I is by repeated use of one dimensional quadrature rules. If lines parallel with the x-axis have at most one segment in common with D, then I can be written in the form

$$I = \int_a^b \Big(\int_{c(x)}^{d(x)} f(x,y)dy\Big)\,dx,$$

or

$$I = \int_a^b \phi(x)\,dx, \qquad \phi(x) = \int_{c(x)}^{d(x)} f(x,y)dy. \tag{5.4.2}$$

For a sequence of values x_i, $i = 1,\dots,n$ we can evaluate the function $\phi(x)$ by the one-dimensional quadrature methods described previously. These function values are then used in another one-dimensional quadrature rule to evaluate I. Note that if D is a more general domain, it might be possible to decompose D into the union of simpler domains on which these methods can be used.

Figure 5.4.1. *Region D of integration.*

Example 5.4.2.
Compute

$$I = \int\int_D \sin^2 y \sin^2 x (1 + x^2 + y^2)^{-1/2}\,dxdy,$$

where

$$D = \{(x,y) \mid x^2 + y^2 \le 1\} \cup \{(x,y) \mid 1 \le x \le 3, |y| \le 0.5\}.$$

is the composite region shown in Fig. 8.4.1. Then

$$I = \int_{-1}^{3} \phi(x) \sin^2 x\,dx, \tag{5.4.3}$$

$$\phi(x) = \int_{-c(x)}^{c(x)} \sin^2 y (1 + x^2 + y^2)^{-1/2} dy, \tag{5.4.4}$$

where

$$c(x) = \begin{cases} (1-x^2)^{1/2}, & x \le \frac{1}{2}\sqrt{3}; \\ \frac{1}{2}, & x \ge \frac{1}{2}\sqrt{3}. \end{cases}$$

Values of $\phi(x)$ were obtained by the application of Romberg's method to (5.4.4) and numerical integration applied to the integral (5.4.3) yielded the value of $I = 0.13202 \pm 10^{-5}$. Ninety-six values of x were needed, and for each value of x, twenty function evaluations used, on the average. The grid is chosen so that $x = \frac{1}{2}\sqrt{3}$, where $\phi'(x)$ is discontinuous, is a grid point.

5.4.3 Product Rules

Consider a double integral over a rectangular region $D = \{(x,y) \mid a \le x \le b, c \le y \le d\}$. Decomposing the integral as in (5.4.2) and using one-dimensional quadrature rules we can write

$$I \approx \sum_{i=1}^{n} u_i \phi(x_i), \qquad \phi(x_i) \approx \sum_{j=1}^{n} v_j f(x_i, y_j),$$

or, combining the rules

$$I \approx \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} f(x_i, y_j), \qquad w_{ij} = u_i v_j. \tag{5.4.5}$$

This is called a **product rule** for the double integral I, and it uses mn function values $f_{ij} = f(x_i, y_j)$.

In particular we can use values of f and an equidistant **rectangular grid** in the (x,y)-plane with grid spacings h and k in the x and y directions, respectively. Let $x_0 = a$, $h = (b-a)/n$, $y_0 = c$, $k = (d-c)/m$, and use the notation $x_i = x_0 + ih$, $y_j = y_0 + jk$. Then the following formulas can be used, generalizing the compound rectangle rule and trapezoidal rule, respectively:

$$I \approx hk \sum_{i=1}^{M}\sum_{j=1}^{N} f_{i-\frac{1}{2}, j-\frac{1}{2}}, \tag{5.4.6}$$

$$I \approx hk \sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij} f_{ij} \tag{5.4.7}$$

Here, for the trapezoidal rule $w_{ij} = 1$ for the interior grid points—i.e., when $0 < i < M$ and $0 < j < N$, $w_{ij} = \frac{1}{4}$ for the four corner points, while $w_{ij} = \frac{1}{2}$ for the other boundary points. Both formulas are exact for bilinear functions, and the error can be expanded in even powers of h, k so that repeated Richardson extrapolation can be used.

Formulas of higher accuracy can also be obtained by using Gaussian quadrature rules in the x and y direction. Note that if the one-dimensional formulas are exact for polynomials of degree d_1 and d_2, respectively, then the product rule will be exact for bivariate polynomials $x^p y^q$ where $p \le d_1$ and $q \le d_2$.

Higher accuracy formulas can also be derived by **operator** techniques, based on an operator formulation of Taylor's expansion, see equation (4.8.2),

$$u(x_0 + h, y_0 + k) = e^{(hD_x + kD_y)} u(x_0, y_0). \qquad (5.4.8)$$

It is possible to use product rules on nonrectangular regions, if these can be mapped into a rectangle. This can be done, e.g., for a triangle. For nonrectangular regions, the rectangular lattice may also be bordered by triangles or "triangles" with one curved side, which may be treated with the techniques outlined in the next section.

5.4.4 Irregular Triangular Grids

A grid of triangles of arbitrary form is a convenient means for approximating a complicated plane region. It is fairly easy to program a computer to refine a coarse triangular grid automatically; see Fig. 8.4.2. It is also easy to adapt the density of points to the behavior of the function.

Triangular grids are thus more flexible than rectangular ones. On the other hand, the administration of a rectangular grid requires less storage and a simpler program. Sometimes the approximation formulas are also a little simpler. Triangular grids have an important application in the **finite element method** (FEM) for problems in continuum mechanics and other applications of partial differential equations; see Chapter 14.

Figure 5.4.2. *Refinement of a triangular grid.*

Let $P_i = (x_i, y_i)$, $i = 1, 2, 3$, be the vertices of a triangle T. Then any point $P = (x, y)$ in the plane can be uniquely expressed by the vector equation

$$P = \theta_1 P_1 + \theta_2 P_2 + \theta_3 P_3, \qquad \theta_1 + \theta_1 + \theta_1 = 1. \qquad (5.4.9)$$

In fact, the θ_i, which are called **barycentric coordinates** of P, are determined from the following nonsingular set of equations:

$$\begin{aligned} \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 &= x, \qquad (5.4.10)\\ \theta_1 y_1 + \theta_2 y_2 + \theta_3 y_3 &= y,\\ \theta_1 + \theta_2 + \theta_3 &= 1, \end{aligned}$$

The interior of the triangle is characterized by the inequalities $\theta_i > 0$, $i = 1, 2, 3$. In this case P is the center of mass (centroid) of the three masses $\theta_1, \theta_2, \theta_3$ located at the vertices of the triangle (see Fig. 8.4.3). This explains the term "barycentric coordinates". $\theta_1 = 0$ is the equation for the side P_2P_3, and similarly for the other sides.

Figure 5.4.3. *Center of mass of a triangle.*

If f is a *nonhomogeneous linear function* of P, i.e., if $f(P) = a^T P + b$, then the reader can verify that

$$f(P) = \theta_1 f(P_1) + \theta_2 f(P_2) + \theta_3 f(P_3). \tag{5.4.11}$$

this is a form of *linear interpolation on triangular grids.* In order to obtain *quadratic interpolation*, we define

$$\Delta'' = f(P_i) + f(P_j) - 2f\big(\frac{1}{2}(P_i + P_j)\big), \qquad i \neq j. \tag{5.4.12}$$

Theorem 5.4.1.
The interpolation formula

$$f(P) = \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 - 2(\theta_2\theta_3\Delta''_{23} + \theta_3\theta_1\Delta''_{31} + \theta_1\theta_2\Delta''_{12})$$

where $f_i = f(P_i)$, is exact for all quadratic functions.

Proof. The right-hand is a quadratic function of P, since it follows from (5.4.10) that the θ_i are (nonhomogeneous) linear functions of x, y. (See also Problem 8.) It remains to show that the right hand side is equal to $f(P)$ for $P = P_i$, and $P = (P_i + P_j)/2$, $i, j = 1, 2, 3$.

For $P = P_i$, $\theta_i = 1$, $\theta_j = 0$, $i \neq j$, hence the right hand side equals f_i. For $P = (P_i + P_j)/2$,

$$\theta_i = \theta_j = \frac{1}{2}, \quad \theta_k = 0, \qquad k \neq i, k \neq j,$$

and hence the right hand side becomes

$$\frac{1}{2}f_i + \frac{1}{2}f_j + -2 \cdot \frac{1}{2}\Big(f_i + f_j - 2u\big(\frac{1}{2}(P_i + P_j)\big)\Big) = f\big(\frac{1}{2}(P_i + P_j)\big).$$

□

The following theorem is equivalent to a rule which has been used in mechanics for the computation of moments of inertia since the nineteenth century:

Theorem 5.4.2.

Let A be the area of a triangle T, with vertices P_1, P_2, P_3. Then the quadrature formula

$$\int\int_T f(x,y)\,dxdy \tag{5.4.13}$$
$$= \frac{1}{3}A\left(f\Big(\frac{1}{2}(P_1+P_2)\Big) + f\Big(\frac{1}{2}(P_2+P_3)\Big) + f\Big(\frac{1}{2}(P_3+P_1)\Big)\right)$$

is exact for all quadratic functions.

PROOF: By symmetry, $\int_T \int \theta_i\,dxdy$ is the same for $i = 1, 2, 3$. Similarly $\int_T \int \theta_i\theta_j\,dxdy$ is the same for all three (i,j)-combinations. Hence for the quadratic function

$$\begin{aligned}\int_T \int f(x,y)\,dxdy &= a(f_1+f_2+f_3) - 2b(\Delta''_{23} + \Delta''_{31} + \Delta''_{12})\\ &= (a-4b)(f_1+f_2+f_3)\\ &+ 4b\Big(f\big(\frac{1}{2}(P_1+P_2)\big) + f\big(\frac{1}{2}(P_2+P_3)\big) + f\big(\frac{1}{2}(P_3+P_1)\big)\Big),\end{aligned}$$

where

$$a = \int_T \int \theta_1\,dxdy, \qquad b = \int_T \int \theta_1\theta_2\,dxdy.$$

Using θ_1, θ_2 as new variables of integration, we get by (5.4.10) and the relation $\theta_3 = 1 - \theta_1 - \theta_2$,

$$\begin{aligned}x &= \theta_1(x_1 - x_3) + \theta_1(x_1 - x_3) + x_3,\\ y &= \theta_1(y_1 - y_3) + \theta_1(y_1 - y_3) + y_3.\end{aligned}$$

Hence the functional determinant is equal to

$$\begin{vmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{vmatrix} = 2A,$$

and (check the limits of integration!)

$$a = \int_{\theta_1=0}^{1}\int_{\theta_2=0}^{1-\theta_1} 2\theta_1 d\theta_1 d\theta_2 = 2A\int_0^1 \theta_1(1-\theta_1)d\theta_1 = \frac{A}{3},$$

$$b = \int_{\theta_1=0}^{1}\int_{\theta_2=0}^{1-\theta_1} 2\theta_1\theta_2 d\theta_1 d\theta_2 = 2A\int_0^1 \theta_1\frac{(1-\theta_1)^2}{2}d\theta_1 = \frac{A}{3}.$$

The results now follows by insertion of this into (5.4.13). □

Figure 5.4.4. *Correction for curved boundary segment.*

A numerical method can be based on Theorem 5.4.1, by covering the domain D by triangles. For each curved boundary segment (Fig. 8.4.4) the correction

$$\frac{4}{3}f(S)A(PRQ) \tag{5.4.14}$$

is to be added, where $A(PRQ)$ is the area of the triangle with vertices P, R, Q. The error of the correction can be shown to be $O(\|Q-P\|^5)$ for each segment, if R is close to the midpoint of the arc PQ. If the boundary is given in parametric form, $x = x(t)$, $y = y(t)$, where x and y are twice differentiable on the arc PQ, then one should choose $t_R = \frac{1}{2}(t_P + t_Q)$. Richardson extrapolation can be used to increase the accuracy, see the examples.

Figure 5.4.5. *The grids for I_4 and I_{16}.*

Example 5.4.3.

Consider the integral

$$I = \int\int_D (x^2+y^2)^k \, dxdy$$

where D is the region shown in Fig. 8.4.5. Let I_n be the result obtained with n triangles. The grids for I_4 and I_{16} are shown in Fig. 8.4.5. Put

$$R'_n = I_{4n} + \frac{1}{15}(I_{4n} - I_n), \qquad R''_n = R'_{4n} + \frac{1}{63}(R'_{4n} - R'_n).$$

The following results were obtained. In this case the work could be reduced by a factor of 4, because of symmetry.

It is seen that R'-values have full accuracy for $k = 2$ and the R''-values have high accuracy even for $k = 4$. In fact, it can be shown that R'-values are ex-

k	I_4	I_{16}	I_{64}	R'_4	R'_{16}	R''_4	Exact
2	0.250000	0.307291	0.310872	0.311111	0.311111	0.311111	0.311111
3	0.104167	0.161784	0.170741	0.165625	0.171338	0.171429	0.171429
4	0.046875	0.090678	0.104094	0.093598	0.104988	0.105169	0.105397

act for any fourth-degree polynomial and R''-values are exact for any sixth-degree polynomial, when the region is covered exactly by the triangles.

Example 5.4.4.

The integral

$$a\int\int (a^2 - y^2)^{-1/2}\,dxdy$$

over a quarter of the unit circle is computed with the grids shown in Fig. 8.4.2, and with boundary corrections according to (5.4.9). The following results, using the notation of the previous example, were obtained and compared with the exact values:

a	I_8	I_{32}	R'_8	Exact
2	0.351995	0.352077	0.352082	0.352082
4	0.337492	0.337608	0.337615	0.337616
6	0.335084	0.335200	0.335207	0.335208
8	0.334259	0.334374	0.334382	0.334382

Note, however, that Richardson extrapolation may not always give improvement, e.g., in cases, where the rate of convergence of the basic method is more rapid than usual.

We mention also that some progress has been made in developing quadrature rules of optimal order for rectangles and triangles. In one dimension this led to Gaussian quadrature rules. In two dimensions the problem is much more difficult. Non-product rules for simple regions like a circle, equilateral triangle, regular hexagon, etc., can be found in Abramowitz and Stegun [1, pp. 891–895]. For a thorough treatment of multiple integrals the reader is referred to the book by Stroud [3].

5.4.5 Monte Carlo Methods

One of the most important application of the Monte Carlo method described in Section 1.4.2 is in the numerical calculation of multiple integrals. If we use product rules to evaluate a multiple integral in d dimensions the work will depend exponentially on d. This means that the problem may quickly becomes intractable when d increases. On the other hand, for the Monte Carlo method the complexity always is proportional to $1/\epsilon$, where ϵ is the required tolerance *independent of the dimension* d. Hence the Monte Carlo method can be said to break "the curse of dimension" inherent in other approaches!

We shall briefly describe some ideas used in integration by the Monte Carlo method. For simplicity, we first consider integrals in *one* dimension, even though the Monte Carlo method cannot really compete with traditional numerical methods for this problem.

Let $R_1, R_2, \dots, R_n$ be a sequence of random numbers rectangularly distributed on $[0,1]$, and set

$$I = \int_0^1 f(x)\,dx \approx I_1 = \frac{1}{n}\sum_{i=1}^n f(R_i).$$

This generalizes to multiple integrals. For example, to approximate a two dimensional integral over the domain $0 \le x, y \le 1$, we sample the integrand $f(x,y)$ in points (R_{2i-1}, R_{2i}), for $i = 1, 2, \dots, n$. The technique can be applied to an integral ovee a general region D, provided that we can sample the integrand f randomly over D.

One can show that the expectation of the variable I_1 is I and that the standard deviation of this estimate decreases in proportion to $n^{-1/2}$. This is very slow even compared to the trapezoidal rule—where the error decreases as n^{-2}. To get one extra decimal place of accuracy we must increase the number of points by a factor of 100. To get three digit accuracy the order of one million points may be required! However, if we consider, e.g., a six-dimensional integral this is not exorbitant. Using a product rule with 10 subdivisions in each dimension would also require 10^6 points.

The above estimate is a special case of a more general one. Suppose X_i $i = 1, 2, \dots, n$, has density function $g(x)$. Then

$$I_2 = \frac{1}{n}\sum_{i=1}^n \frac{f(X_i)}{g(X_i)}$$

has expected value I, since

$$E\left(\frac{f(X_i)}{g(X_i)}\right) = \int_0^1 \frac{f(x)}{g(x)} f(x)\,dx = \int_0^1 f(x)\,dx = I.$$

If one can find a frequency function $g(x)$ such that $f(x)/g(x)$ fluctuates less than $f(x)$, then I_2 will have smaller variance than I_1. This procedure is called **importance sampling**; it has proved very useful in particle-physics problems, where important phenomena (e.g., dangerous radiation which penetrates a shield) are associated with certain events of low probability.

In Sec. 5.4.5 we mentioned the method of using a simple comparison problem. The Monte Carlo variant of this method is called the **control variate method**. Suppose that $\phi(x)$ is a function whose integral has a known value K, and suppose that $f(x) - \phi(x)$ fluctuates much less than $f(x)$. Then

$$I = K + \int_0^1 (f(x) - \phi(x))\,dx \approx K + I_3, \qquad I_3 = \frac{1}{n}\sum_{i=1}^n (f(R_i) - \phi(R_i)),$$

where I_3 has less variance than I_1.

Review Questions

1. How is bilinear interpolation performed? What is the order of accuracy?
2. Define barycentric coordinates, and give the formula for linear interpolation on a triangular grid.
3. Describe the methods for numerical integration with rectangular or triangular grids.

Problems

1. Let D be the unit circle. Introduce polar coordinates in the integral

$$I = \int\int_D \frac{y \sin(ky)}{x^2 + y^2}\, dxdy$$

and reduce it analytically to a single integral.

2. Let E be the ellipse $\{(x, y) \mid (x/a)^2 + (y/b)^2 \le 1\}$. Transform

$$I = \int\int_E f(x, y)\, dxdy$$

into an integral over a rectangle in the (r, t)-plane with the transformation $x = ar\cos t$, $y = br \sin t$.

3. Compute by bilinear interpolation $u(0.5, 0.25)$ when

$$u(0, 0) = 1, \quad u(1, 0) = 2, \quad u(0, 1) = 3, \quad u(1, 1) = 5.$$

4. Show that, using the notation for equidistant rectangular grids, the formula

$$\int_{x_0-h}^{x_0+h} \int_{y_0-k}^{y_0+k} f(x, y)\, dxdy = \frac{4hk}{6}(f_{1,0} + f_{0,1} + f_{-1,0} + f_{0,-1} + 2f_{0,0})$$

is exact for all cubic polynomials.

5. Is a quadratic polynomial uniquely determined, given six functions values at the vertices and midpoints of the sides of a triangle?
6. Show that the boundary correction of (5.4.9) is exact if $f \equiv 1$, and if the arc is a parabola where the tangent at R is parallel to PQ.
7. Formulate generalizations to several dimensions of the integral formula of Theorem 5.4.1, and convince yourself of their validity.

   Hint: The formula is most simply expressed in terms of the values in the vertices and in the centroid of a simplex.

8. (a) Write a program which uses the Monte Carlo method to compute $\int_0^1 e^x\, dx$. Take 25, 100, 225, 400 and 635 points. How does the error depend (approximately) on the number of points?

   (b) Compute the integral in (a) using the control variate method. Take $\phi(x) = 1 + x + x^2/2$. Use the same number of points as in (a).

Notes and References

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* National Bureau of Standards, Dover Publications, New York, NY, 1965.

[2] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner. *QUADPACK, A Subroutine Package for Automatic Integration.* Springer-Verlag, Berlin, 1983.

[3] A. Stroud. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, NJ, 1972.

Chapter 6

Solving Scalar Nonlinear Equations

6.1 Introduction

In this chapter we study numerical methods for computing the roots of a nonlinear equation

$$f(x) = 0, \tag{6.1.1}$$

where $f(x)$ is a real- or complex-valued function of one variable. We also briefly consider the related problem of finding the minimum or maximum of a real-valued function $f(x)$. These are problems that has occupied mathematicians for many centuries.

The roots of (6.1.1) cannot in general be expressed in closed form. Even when an explicit solution is available, this is often so complicated that numerical methods are more useful. Such methods are iterative in nature, that is, starting from one or more initial approximations to the root, they produce a sequence $\{x_n\}$ which presumably converges to the desired root.

The function $f(x)$ need not be known by a closed analytical expression. For the methods to be applicable it suffices that we can evaluate $f(x)$ and possibly some of its derivatives for given numerical values of x. The object is often to use as few function evaluations as possible in order to approximate the root with a prescribed accuracy. It is not uncommon with applications where each function value is obtained by a complicated computation, e.g., by the numerical solution of a differential equation.

With certain methods it is sufficient for convergence to know an interval [a,b] which contains the desired root (and no other root). An important such method is the bisection method described in Section 6.1.1. Other methods, which make more use of regularity assumptions about $f(x)$, may require an initial approximation which is close to the desired root; in return, these methods converge more quickly. Such methods are often special cases of fixed point iteration methods, which are covered in Section 6.2, Newton's method and its modifications are the most important of these; see Section 6.3. The method in Section 6.4 based on interpolation can achieve similar efficiency without using derivatives. It is often suitable to use, e.g., the bisection method to roughly locate the root and then change to a more rapidly convergent method in the final stage.

Since an iterative method has to be truncated after a finite number of steps we can only hope to obtain approximations to the roots of (6.1.1). However, in practice the round-off errors occurring in the evaluation of computed function values of $f(x)$ is the dominant source of error. The effect of such errors on the attainable accuracy of computed roots is discussed in Section 6.5. For the most part we limit ourselves to the problem of determining *simple root* α, i.e., we assume that $f'(\alpha) \neq 0$. Multiple roots are discussed in Section 6.5.3. Special methods for the problem of determining real and complex roots of a polynomial equation $P(x) = 0$ are taken up in Section 6.6.

Solving a *system of nonlinear equations* can be a far more difficult problem. Even though many of the methods for a single equation, such as Newton's method, are easily generalized, they presume that good approximations to the roots are known. To obtain convergence from less accurate initial approximations requires several modifications. Hence the discussion of such methods are deferred to Chapter 11.

6.1.1 The Bisection Method

Rough initial approximations to the real roots of an equation $f(x) = 0$ can often be obtained by graphing the function $f(x)$. In simple cases this can be a useful tool for determining the number of roots and intervals containing each root.

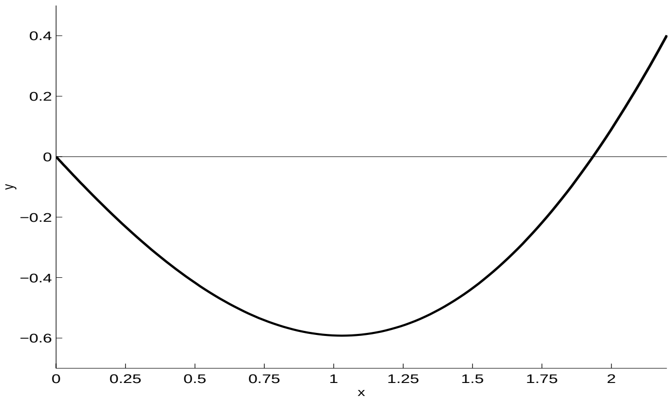


Figure 6.1.1. *Graph of curve* $y = (x/2)^2 - \sin x$.

By the intermediate-value theorem if a function $f(x)$ is continuous for $a \leq x \leq b$, $f(a) \neq f(b)$, and k is between $f(a)$ and $f(b)$, then there is a point $\xi \in (a, b)$ such that $f(\xi) = k$. In particular, if $f(a)f(b) < 0$ then the equation $f(x) = 0$ has at least one root in the interval (a, b).

Example 6.1.1.

We obtain a rough estimate of the root α of the equation

$$f(x) = (x/2)^2 - \sin x = 0.$$

from the graph in Figure 6.1.1. It is clear that $\alpha \in (1.8, 2)$, probably close to $\alpha \approx x_0 = 1.9$. (Alternatively we could look for the intersection of the two curves $y_1 = (x/2)^2$, $y_2 = \sin x$.)

Another possibility is to make a table of some values of the function $f(x)$. We can easily set up the following table:

x	$(x/2)^2$	$\sin x$	$f(x)$
1.6	0.64	0.9996	< 0
1.8	0.81	0.974	< 0
2.0	1.00	0.909	> 0

From this table we can conclude that there is a root in the interval $(1.8, 2.0)$.

A more systematic use of the above tabulation method is made in the **bisection method**. Assume that $f(x)$ is continuous in the interval (a_0, b_0) and that $f(a_0)f(b_0) < 0$. We shall determine a sequence of intervals $I_k = (a_k, b_k)$, $k = 1, 2, 3, \ldots$, such that

$$(a_1, b_1) \supset (a_2, b_2) \supset (a_3, b_3) \subset \cdots$$

and which all contain a root of the equation $f(x) = 0$. The intervals are determined recursively as follows. Given $I_k = (a_k, b_k)$ compute the midpoint

$$m_k = \frac{1}{2}(a_k + b_k). \tag{6.1.2}$$

and $f(m_k)$. We can assume that $f(m_k) \neq 0$, since otherwise we have found a root. The new interval $I_{k+1} = (a_{k+1}, b_{k+1})$ is then determined by the rule

$$(a_{k+1}, b_{k+1}) = \begin{cases} (m_k, b_k), & \text{if } f(m_k)f(a_k) > 0; \\ (a_k, m_k), & \text{if } f(m_k)f(a_k) < 0. \end{cases} \tag{6.1.3}$$

From the construction it follows immediately that $f(a_{k+1})f(b_{k+1}) < 0$ (see also Figure 6.1.1) and therefore the interval I_{k+1} also contains a root of $f(x) = 0$.

After n bisection steps we have contained a root in the interval (a_n, b_n) of length $2^{-n}(b_0 - a_0)$. If we take m_n as an estimate of the root α, we have the error estimate

$$|\alpha - m_n| < 2^{-(n+1)}(b_0 - a_0). \tag{6.1.4}$$

At each step we gain one binary digit in accuracy or, since $10^{-1} \approx 2^{-3.3}$, on the average one decimal digit per 3.3 steps, which may suffice for many practical purposes. To find an interval of length δ which includes a root will take about $\log_2((b-a)/\delta)$ evaluations of f.

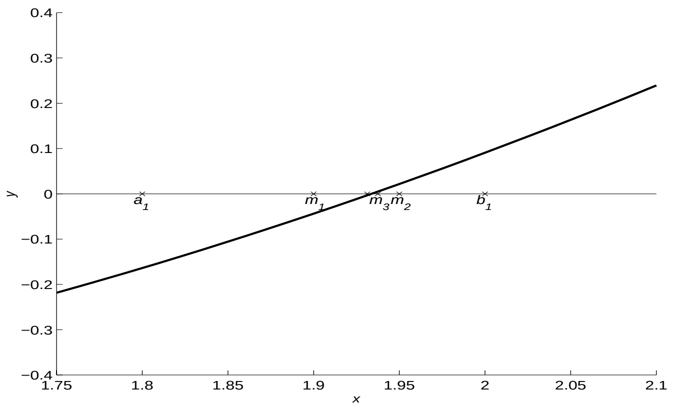


Figure 6.1.2. *The bisection method.*

Example 6.1.2.

The bisection method applied to the equation $(x/2)^2 - \sin x = 0$, with $I_0 = (1.8, 2)$ gives the sequence of intervals $[a_n, b_n], where$:

n	a_n	b_n	m_n	$f(m_n)$
1	1.8	2	1.9	<0
2	1.9	2	1.95	>0
3	1.9	1.95	1.925	<0
4	1.925	1.95	1.9375	>0
5	1.925	1.9375	1.93125	<0
6	1.93125	1.9375	1.934375	>0

Here after six function evaluations we have $\alpha \in (1.93125, 1.934375)$ an interval of length $0.2 \cdot 2^{-6} = 0.003125$.

When implementing the bisection method the midpoint should be computed as $m = a + \frac{1}{2}(b - a)$. This has the advantage that no round-off occurs in the subtraction (see Theorem 2.2.2).

Example 6.1.3.

The inequalities $a \leq \frac{1}{2}(a + b) \leq b$, where a and b are floating point numbers with $a \leq b$ can be violated in base 10 arithmetic. For example, assume that floating point arithmetic with six decimal digits is used. Taking $a = 0.742531$ and $b = 0.742533$ we obatin $fl(a+b) = 1.48506$ (rounded) and $\frac{1}{2}(a+b) = 0.742530$. On the other hand the inequalities $a \leq a + \frac{1}{2}(b - a) \leq b$ are true in base β arithmetic, for any β. With a and p as given we get the correct value 0.742532.

An algorithmic description of the bisection method is given below. In this the tolerance δ is increased by the amount $u \max(|a|, |b|)$, where u is the machine precision. This is to guard against the possibility that δ has been chosen smaller than the spacing between the floating point numbers between a and b.

Algorithm 6.1.1 *The Bisection Method.*

Let f be a given function and $I = [a, b]$ an interval such that $b > a$ and $f(a)f(b) \leq 0$. The algorithm bisect attempts to compute an approximation to a root $m \in I$ of $f(x) = 0$, with an error less than a specified tolerance $\delta > 0$.

```
root = bisect($f, a, b, \delta$);
$\delta = \delta + u \cdot \max((|a|, |b|)$;
$fa = f(a)$;
while $|b - a| > \delta$
    $m = a + (b - a)/2$;
    $fm = f(m)$;
    if $fm \cdot fa \leq 0$
      $b = m$;
    else
      $a = m$;   $fa = fm$;
    end;
end;
$root = a + (b - a)/2$;
```

The bisection algorithm is deceptively simple and it has be stressed that it may not return an approximation with an error close to δ. This is because *the proper choice of interval will only be made as long as the sign of $f(m)$ is correctly evaluated.* In Section 6.5 we show that there is a limiting accuracy with which a root can be determined, which is independent of the particular method used.

The bisection algorithm makes no quantitative use of the magnitude of the function values. As long as the sign of the computed function value $f(m)$ is correctly determined the correct subinterval will be chosen. If the tolerance δ is chosen too "small" or the root is ill-conditioned this may fail to be true in the later steps (cf. Sec. 6.2), but even then the computed midpoints will stay within the domain of uncertainty; cf. Sec. 6.5.1.

If the initial interval (a, b) contains several roots of $f(x) = 0$, then the bisection method will converge to just one of these. If it is known that the minimum distance between two zeros is less than d and we want to compute *all* roots in (a, b), then we can start by computing the function values of $f(x)$ at equidistant points $a, a+d, a+2d, \ldots$. The bisection method is then applied to each interval $I_k = [a+kd, a+(k+1)d]$ in which $f(x)$ changes sign.

If we only know (say) a lower bound $a < \alpha$ for the root to be determined we can proceed as follows. We choose an initial steplength d and in the first **hunting phase** compute successively function values $f(a+d)$, $f(a+2d)$, $f(a+4d), \ldots$, until a function value is found such that $f(a)f(a + 2^k d) < 0$. At this point we have bracketed a root and can start the bisection algorithm.

In the bisection method the interval of interest is in each step split into *two* subintervals. An obvious generalization is to partition instead into k subintervals, for $k \geq 2$. In such a **multi-section method** of order k the interval $I = [a, b]$ is divided into k subintervals $I_i = [x_i, x_{i+1}]$, where

$$x_i = a + i[(b-a)/k], \quad i = 0 : k.$$

Bisection is multi-section of order 2. If there exists only one root in the interval I and we wish to compute it with an absolute error ϵ, then it is necessary to perform

$$n_k = \log_2\left(\frac{b-a}{2\epsilon}\right) \Big/ \log_2(k)$$

multi-sections of order k. Thus, the efficiency of multi-section of order k compared to bisection is

$$n_1/(kn_k) = \log_2(k)/k.$$

Hence if there is a single root in the interval bisection is always preferable. If there are several roots in the interval multi-section may perform better if subintervals can be processed in parallel.

In Section 4.4.5 we considered evaluating the nonzero B-splines for a given argument x. Then we first have to search an ordered sequence of knots $\tau_0, \ldots, \tau_m$ to find the interval such that $\tau_j \leq x < \tau_{j+1}$. This can be solved by a slight modification of the bisection method. An similar problem important in computer science is *searching in an ordered register*, e.g., a register of employees ordered according to increasing Social Security number. If the nth number in the register is denoted by $f(n)$, then searching for a certain number a means that an equation $f(n) = a$ is to be solved (here f is an increasing, discontinuous function). The bisection method can also be used in searching an *alphabetically* ordered register.

The bisection method has the advantage that, for any function $f(x)$ continuous in the interval $[a, b]$, it reduces by one half the width of an interval bracketing a root $\alpha \in (a, b)$. This is true provided that the sign of computed functions values are correctly determined. The rate of convergence is slow, but *independent of the regularity of $f(x)$*. In the rest of this chapter we will consider methods which, assuming more regularity of $f(x)$ can achieve much faster rate of convergence.

Review Questions

1. What does limit the final accuracy of a root computed by the bisection algorithm? Discuss suitable termination criteria.

Problems

1. Use graphic representation to determine the zeros of the following functions to one correct decimal:

(a) $4\sin x + 1 - x$; (b) $1 - x - e^{-2x}$; (c) $(x+1)e^{x-1} - 1$;
(d) $x^4 - 4x^3 + 2x^2 - 8$; (e) $e^x + x^2 + x$; (f) $e^x - x^2 - 2x - 2$; (g) $3x^2 + \tan x$.

2. Show analytically that the equation $xe^{-x} = \gamma$ has exactly two real roots when $\gamma < e^{-1}$.

3. Plot the functions $f(x) = \cosh x$ and $g(x) = 1/\cos x$ and deduce that the equation $\cosh x \cos x = 1$ has its smallest positive root in the interval $(3\pi/2, 2\pi)$. Determine this root using the bisection method.

4. The following equations all have a root in the interval $(0, 1.6)$ Determine these with an error less than 10^{-6} using the bisection method.
(a) $x\cos x = \ln x$; (b) $2x = e^{-x}$; (c) $e^{-2x} = 1 - x$.

5. Let k be a given non-negative number and consider the equation $\sin x = -k\cos x$. This equation has infinitely many roots. Separate the roots, i.e., partition the real axis into intervals which contain exactly one root.

6. The choice of m_k as the *arithmetic* mean of a_{k-1} and b_{k-1} in the bisection method minimizes the worst case maximum *absolute* error. If in the case that $ab > 0$ we take instead

$$m_k = \sqrt{a_k b_k}$$

i.e., the *geometric* mean, then the worst case *relative* error is minimized. Do Example 6.1.2 using this variation of the bisection method.

6.2 Fixed-Point Iteration

We now introduce a very general class of iteration methods, which includes many important root-finding methods as special cases.

Let ϕ be a continuous function and $\{x_n\}$ the sequence generated by

$$x_{n+1} = \phi(x_n), \quad n = 0, 1, 2, \dots . \tag{6.2.1}$$

for some initial value x_0. Assuming that $\lim_{n\to\infty} x_n = \alpha$ we have that

$$\alpha = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \phi(x_n) = \phi(\alpha), \tag{6.2.2}$$

i.e., the limiting value α is a root of the equation $x = \phi(x)$. We call α a **fixed point** of the mapping $x \to \phi(x)$, and the iteration (6.2.1) a **fixed-point iteration**.

An iterative method for solving an equation $f(x) = 0$ can be constructed by rewriting it in the form $x = \phi(x)$, which then defines a fixed point iteration (6.2.1). Clearly this can be done in many ways.

Example 6.2.1.

The equation $x + \ln x = 0$ can, for example, be written as: (i) $x = -\ln x$; (ii) $x = e^{-x}$; (iii) $x = (x + e^{-x})/2$. Each of these give rise to a different fixed-point iteration.

In Figure 6.2.1 the results from the first eight iterations

$$x_{n+1} = e^{-x_n}, \quad x_0 = 0.3,$$

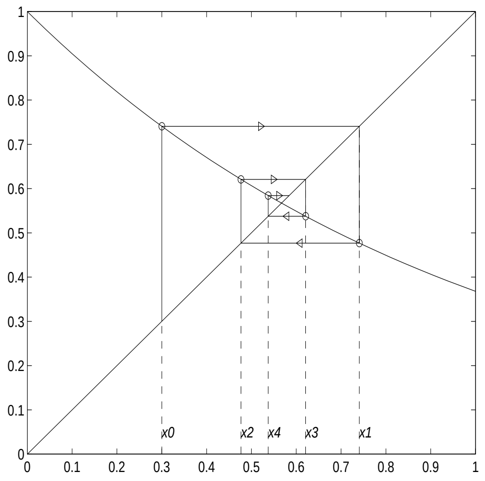


Figure 6.2.1. *The fix-point iteration* $x_{k+1} = e^{-x_k}$, $x_0 = 0.3$.

are pictured. We get $x_9 = 0.5641$ (correct value 0.567143).

As was shown already in Section 1.2, the iteration (6.2.1) may not converge even if the initial value x_0 is chosen arbitrarily close to a root. If (6.2.2) holds for all x_0 in a sufficiently close neighborhood of α the α is called a point of **attraction** otherwise α is a point of **repulsion**. We now develop a general theory for fixed point iteration methods.

A sufficient condition for (6.2.1) to generate a convergenct sequence is given in the following theorem.

Theorem 6.2.1.

Suppose that the function $\phi(x)$ has a real fixed point α, and that in the closed interval

$$J = \{x \mid |x-\alpha| \le \rho\}$$

$x \to \phi(x)$ is a **contraction mapping**, *i.e.,*

$$|\phi(x) - \phi(y)| \le m|x-y|, \quad m < 1, \tag{6.2.3}$$

Then the iteration method $x_n = \phi(x_{n-1})$, $n = 1, 2, 3 \ldots$, generates a sequence $\{x_n\}$ such that for all $x_0 \in J$:

(a) $x_n \in J$, $n = 1, 2, \ldots$;

(b) $\lim_{n\to\infty} x_n = \alpha$,

(c) α is the only root in J of $x = \phi(x)$.

Proof. We first prove assertion (a), by induction. Suppose that $x_{n-1} \in J$. Then by (6.2.3) it follows that

$$|x_n - \alpha| = |\phi(x_{n-1}) - \phi(\alpha)| \le m|x_{n-1} - \alpha| \le m\rho.$$

Hence $x_n \in J$ and (a) is proved. Repeated use of the inequality above gives

$$|x_n - \alpha| \le m|x_{n-1} - \alpha| \le \cdots \le m^n|x_0 - \alpha|,$$

and since $m < 1$, the result (b) follows. Suppose, finally, that $x = \phi(x)$ has another root $\beta \in J$, $\beta \neq \alpha$. Then, by (6.2.3)

$$|\alpha - \beta| = |\phi(\alpha) - \phi(\beta)| < |\alpha - \beta|,$$

a contradiction; thus (c) follows. □

Note that if $\phi'(x)$ exists, then a sufficient condition for (6.2.3) to hold is that

$$|\phi'(x)| \le m < 1, \quad x \in J. \tag{6.2.4}$$

since then by the mean value theorem we have for $x, y \in J$ that

$$|\phi(x) - \phi(y)| = |\phi'(\zeta_n)||x - y| < |x - y|, \quad \zeta_n \in J.$$

On the other hand if $|\phi'(\alpha)| > 1$ then the iterative method (6.2.1) diverges. The four different cases that occur, depending on the sign and magnitude of $\phi'(\alpha)$ were illustrated in Figures 1.2.1a–d.

In Theorem 6.2.1 *we assumed the existence of a fixed point* α of $\phi(x)$. It is remarkable that the theorem can be modified so that it can be used to *prove the existence of a fixed point*, and hence of a root of the equation $x = \phi(x)$.

Theorem 6.2.2.

Let x_0 be a starting point, and $x_{n+1} = \phi(x_n)$, $n = 1, 2, \ldots$. Further, let m be a constant, $0 < m < 1$, and J be a closed interval with $x_0 \in J$ such that

$$|\phi(x) - \phi(y)| \le m|x - y|, \quad x, y \in J$$

and $x_1 + \dfrac{m}{1-m}(x_1 - x_0) \in J$. Then (a), (b) and (c) of Theorem 6.2.1 are true.

Proof. The theorem will be proved in a more general setting in Chapter 12 (see Theorem 12.2.1). □

Assume that $\phi'(x)$ exists and is continuous in a neighborhood of a root α. Then it follows from the proof of Theorem 6.2.1, that if x_0 is chosen sufficiently close to α, then the sequence x_n generated by (6.2.1) converges, and it holds that

$$\lim_{n\to\infty} \frac{x_n - \alpha}{x_{n-1} - \alpha} = \phi'(\alpha).$$

If $\phi'(\alpha) \neq 0$ then we say that convergence is **linear** with **rate** $|\phi'(\alpha)|$. The iteration method $x_{n+1} = \phi(x_n)$ is then a first order method.

We now formally define the important concept of convergence order for a convergent sequence.

Definition 6.2.3.

A convergent sequence $\{x_k\}$ *with* $\lim_{k\to\infty} x_k = \alpha$, *is said to have* **convergence order** *equal to p if it holds*

$$\lim_{k\to\infty} \frac{|x_{k+1}-\alpha|}{|x_k-\alpha|^p} = C \neq 0. \tag{6.2.5}$$

C is called the **asymptotic error constant**. *For* $p=1$ *it is necessary that* $|C| \le 1$ *and C is called the rate of linear convergence. Convergence is* **superlinear** *if* $C=0$ *for some* $p \ge 1$. *For* $p = 1, 2, 3$ *the convergence is called* linear, quadratic, *and* cubic, *respectively.*

Note that p need not be an integer in (6.2.5). The number of accurate decimal places in the approximation x_n equals $\delta_n = -\log_{10}|\epsilon_n|$. From equation (6.2.5) follows

$$\delta_{n+1} \approx p\delta_n - \log_{10}|C|.$$

Hence for linear convergence ($p=1$) as $n \to \infty$ each iteration gives a fixed number of additional decimal places. For a method with convergence of order $p > 1$ each iteration increases the number of decimal places p-fold as $n \to \infty$.

If each iteration requires m units of work (usually the work involved in computing a function value or a value of one of its derivatives) then the **efficiency index** of the iteration is defined as

$$E = p^{1/m}. \tag{6.2.6}$$

The efficiency index gives a basis for comparing the efficiency of iterative methods of different order of superlinear convergence. (Methods that converge linearly all have $E=1$.)

The order of the iteration method (6.2.1) can be determined if $\phi(x)$ is sufficiently many times continuously differentiable in a neighborhood of α.

Theorem 6.2.4. *The iteration method* $x_{n+1} = \phi(x_n)$ *is of order p for the root* α *if and only if*

$$\phi^{(j)}(\alpha) = 0, \quad j = 1:p-1, \quad \phi^{(p)}(\alpha) \neq 0. \tag{6.2.7}$$

Proof. If equation (6.2.7) holds, then according to Taylor's theorem we have

$$x_{n+1} = \phi(x_n) = \alpha + \frac{1}{p!}\phi^{(p)}(\zeta_n)(x_n-\alpha)^p, \quad \zeta_n \in \operatorname{int}(x_n, \alpha).$$

Hence for a convergent sequence x_n the error $\epsilon_n = x_n - \alpha$ satisfies

$$\lim_{n\to\infty} |\epsilon_{n+1}|/|\epsilon_n|^p = |\phi^{(p)}(\alpha)|/p! \neq 0,$$

and the order of convergence equals p. It also follows that if $\phi^{(j)}(\alpha) \neq 0$ for some j, $1 \le j < p$, or if $\phi^{(p)}(\alpha) = 0$, then the iteration cannot be of order p. $\square$

For a linearly convergent fixed point iteration we have

$$\frac{x_n - \alpha}{x_{n-1} - \alpha} \approx C \neq 0,$$

that is, the sequence $\{x_n - \alpha\}$ approximately forms a geometric series. A more rapidly convergent sequence $\{x'_n\}$ can then be obtained by **Aitken extrapolation** (see Section 3.2.3)

$$x'_n = x_n - (\Delta x_{n-1})^2/\Delta^2 x_{n-2}. \tag{6.2.8}$$

Note that if the convergence is *not* linear, then the sequence $\{x'_n\}$ will usually converge *slower* than $\{x_n\}$!

Example 6.2.2.
The equation $x = e^{-x}$ has one root $\alpha \approx 0.567$. Using the fixed-point iteration $x_{n+1} = e^{-x_n}$ combined with Aitken extrapolation we obtain the result shown in the table below.

n	x_n	Δx_{n-1}	$\Delta^2 x_{n-2}$	x'_n
0	**0.567**00 00000			
1	**0.567**22 45624	2245624		
2	**0.567**09 71994	-1273630	-3519254	**0.56714 329**25
3	**0.5671**6 94312	722318	1995948	**0.56714 329**11
4	**0.5671**2 84650	-409662	-1131980	**0.56714 32906**

It is seen that in this example the extrapolated sequence $\{x'_n\}$ converges much more rapidly, and nine correct decimals are obtained.

In the above example Aitken extrapolation was used in a *passive* way to transform the sequence $\{x_n\}$ into $\{x'_n\}$. It is also possible to use Aitken extrapolation in an *active way* as follows. We start as before by computing $x_1 = \phi(x_0)$, $x_2 = \phi(x_1)$ and apply the formula (6.3.17) to compute x'_2. Next we continue the iterations from x'_2, i.e., compute $x_3 = \phi(x'_2)$, $x_4 = \phi(x_3)$. We can now extrapolate from x'_2, x_3 and x_4 to get x'_4, etc. It can easily be shown that the sequence $z_n = x'_{2n}$ satisfies $z_{n+1} = \psi(z_n)$, where

$$\psi(z) = z - \frac{(\phi(z) - z)^2}{(\phi(\phi(z)) - \phi(z)) - (\phi(z) - z)}.$$

Active Aitken extrapolation may lead to a convergent method even when the basic iteration $x_{n+1} = \phi(x_n)$ diverges!

Problems

1. In Example 6.2.1 three different fixed point iterations were suggested for solving the equation $x + \ln x = 0$. (a) Which of the formulas *can* be used?

(b) Which of the formulas *should* be used?

(c) Give an even better formula!

2. Investigate if and to what limit the iteration $x_{n+1} = 2^{x_n - 1}$ sequence converges for various choices of x_0.

3. (a) Consider the fix point iteration $x_{n+1} = \phi(x_n)$, where $\phi(x) = x + (x-1)^2$. Show that this has a fix point for $\alpha = 1$ and that $\phi'(\alpha) = 1$.

(b) Show that the iteration in (a) is convergent for $x_0 < 1$.

4. In order to determine a root of the equation $x = \phi(x)$ one has computed $x_4 = 0.43789$, $x_5 = 0.43814$, and knows that $|\phi'(x)| \leq 0.4$. How many more iterations are needed to be sure to attain an error less than $0.5 \cdot 10^{-4}$?

5. Let the function $f(x)$ be four times continuously differentiable and have a simple zero α. Successive approximation are computed by $x_{n+1} = (x'_{n+1} + x''_{n+1})/2$, where

$$x'_{n+1} = x_n - u(x_n), \qquad x''_{n+1} = x_n - u(x_n)/u'(x_n),$$

and $u(x) = f(x)/f'(x)$. Show that if the sequence x_n converges to α, then the convergence is *cubic*.

6. Use active Aitken extrapolation on the (divergent) iterative method $x_{n+1} = 5 \ln x_n$ to compute the smallest root of the equation $x = 5 \ln x$. Start with $x_0 = 1.3$.

6.3 Newton's Method

6.3.1 Introduction

In this and the following sections we will study methods which make more efficient use of the computed function values than the bisection method and also use values of derivatives of $f(x)$. If $f(x)$ is sufficiently regular such methods achieve significantly faster convergence.

Newton's method[1] for solving an equation $f(x) = 0$ is based on approximating the curve $y = f(x)$ by its tangent at the point $(x_n, f(x_n))$, where x_n is the current approximation to the root. Thus the next approximation x_{n+1} is determined as the abscissa of the point of intersection of the tangent with the x-axis (see Figure 1.2.3). Clearly we have to assume that $f'(x_n) \neq 0$. This is equivalent to replacing the equation $f(x) = 0$ by

$$T(x) = f(x_n) + (x - x_n)f'(x_n) = 0, \tag{6.3.1}$$

where $T(x)$ is obtained by truncating the Taylor expansion of $f(x)$ at x_n after the first two terms. Hence in **Newton's method** x_{n+1} is determined from

$$x_{n+1} = x_n + h_n, \quad h_n = -f(x_n)/f'(x_n). \tag{6.3.2}$$

If the iterations are broken off when $|h_n| < \delta$ it can be shown (see the error analysis below) that the truncation error is less than δ, provided that $|Kh_n| \leq 1/2$, where K is an upper bound for $|f''/f'|$ in the neighborhood of the root. This restriction is seldom of practical importance. However, one should also take account of rounding errors made in computing h_n, see Section 6.5.

[1]Newton's method is occasionally called Newton–Raphson's method; for a historical account of the development of the method see Ypma [14, 1995].

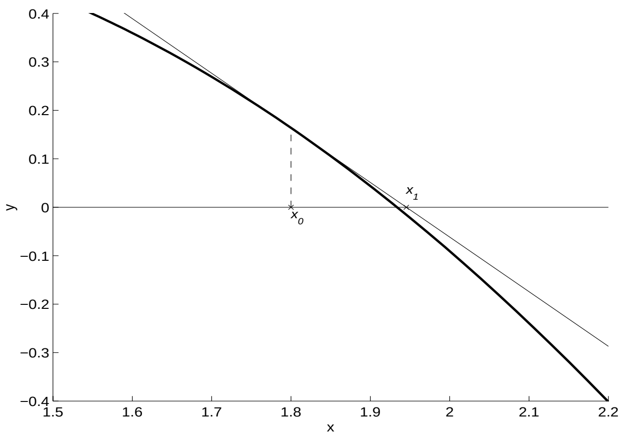


Figure 6.3.1. *Newton's method for the equation* $f(x) = (x/2)^2 - \sin x = 0$.

Example 6.3.1.

Given the equation $f(x) = (x/2)^2 - \sin x = 0$ and $f'(x) = x/2 - \cos x$, we want to compute its positive root; cf. Example 6.1.1. In Figure 6.3.1 the first step starting from $x_0 = 1.8$ is illustrated. Continuing Newton's method generates the following approximations (correct digits in x_n in bold):

n	x_n	$f(x_n)$	$f'(x_n)$	h_n
0	**1**.8	−0.163847 630878	1.127202 094693	−0.145357 812631
1	**1.9**45357 812631	0.015436 106659	1.338543 359427	0.011532 018406
2	**1.933**825 794225	0.000095 223283	1.322020 778469	0.000072 028582
3	**1.933753 76**5643	0.000000 003722	1.3219174 29113	0.000000 002816
4	**1.933753 762827**			

The number of correct digits approximately double in each iteration until the limiting precision is reached. Although the initial approximation is not very good, already x_4 is correct to twelve decimals!

Note that when we approach the root the *relative* precision in the computed values of $f(x_n)$ becomes less and less. Since $f'(x_n)$ is only used for computing h_n. it need not be computed to much greater relative precision than $f(x_n)$, In the above example we could have used $f'(x_2)$ also for $n > 2$ without affecting the convergence. Such a simplification can be of great importance when Newton's method is used on *systems* of nonlinear equations; see Section 12.2.3.

Newton's method applied to the equation $f(x) = x^p - c = 0$ can be used to compute $c^{1/p}$, $p = \pm 1, \pm 2, \ldots$. The sequence $x_1, x_2, x_3, \ldots$, is then computed recursively from

$$x_{n+1} = x_n - \frac{x_n^p - c}{p x_n^{p-1}},$$

which can be written as

$$x_{n+1} = \frac{1}{p}\left((p-1)x_n + \frac{c}{x_n^{p-1}}\right) = \frac{x_n}{(-p)}[(1-p) - cx_n^{-p}]. \tag{6.3.3}$$

It is convenient to use the first expression in (6.3.3) when $p > 0$ and the second when $p < 0$. This iteration formula is often used for calculating, e.g., $\sqrt{c}$, $\sqrt[3]{c}$, and $1/\sqrt{c}$, corresponding to $p = 2, 3, -2$ respectively. Note also that $1/c$, corresponding to $p = -1$ can be computed by the iteration

$$x_{n+1} = x_n + x_n(1 - cx_n),$$

using only multiplications and addition no divisions.

Example 6.3.2.

Suppose we want to construct an algorithm based on Newton's method for efficiently computing the square root of a given number $z > 0$. The first problem is to find an initial approximation. This task is greatly simplified by writing

$$a = c \cdot 4^e, \quad 1/4 \le c < 1,$$

where e is an integer. Then we have $\sqrt{a} = \sqrt{c} \cdot 2^e$, and we need only consider the reduced range $c \in [1/4, 1)$. An initial approximation x_0 can now be obtained by linear interpolation of $x = \sqrt{c}$ in the endpoints $1/4, 1$, giving the iteration $x_0 = (1 + 2c)/3$,

$$x_{n+1} = x_n + h_n, \quad h_n = \frac{1}{2}\Big(x_n - \frac{c}{x_n}\Big), \quad n = 0, 1, 2, \dots. \tag{6.3.4}$$

For $c = 0.5$ the result is (correct digits in boldface)

$$x_0 = 0.66666666666667, \; x_1 = \mathbf{0.70}833333333333, \; x_2 = \mathbf{0.70710}784313725,$$
$$x_3 = \mathbf{0.70710678118}734, \; x_4 = \mathbf{0.70710678118655}.$$

The quadratic rate of convergence is apparent. For a slightly more refined variant of this method see Problem 10.

6.3.2 Convergence Results

We first consider the **local** convergence of Newton's method, that is the convergence in a neighborhood of a root α.

Theorem 6.3.1. *Assume that α is a simple root of $f(x) = 0$, i.e., $f'(\alpha) \neq 0$, and that f' exists and is continuous. Then the convergence order of Newton's method is at least equal to two.*

Proof. Newton's method can be written as a fixed point iteration $x_{k+1} = \phi(x_k)$ where

$$\phi(x) = x - u(x), \quad u(x) = \frac{f(x)}{f'(x)}, \tag{6.3.5}$$

where $f'(x) \neq 0$ for all x in a neighborhood of α. If $f(x)$ is twice differentiable then

$$\phi'(x) = 1 - u'(x) = \frac{f(x)f''(x)}{f'(x)^2}. \tag{6.3.6}$$

Hence $\phi'(\alpha) = 0$, and the result follows from Theorem 6.2.4. □

Assume that $f'(x)$ is continuous in a neighbourhood of α and that $f'(\alpha) \neq 0$. Expanding f in a Taylor series about x_0 we get

$$0 = f(\alpha) = f(x_n) + (\alpha - x_n)f'(\xi_n), \quad \xi_n \in \operatorname{int}(x_n, \alpha).$$

We let $\epsilon_n = x_n - \alpha$ denote the error in the approximation x_n. Subtracting (6.3.1) with $x = x_{n+1}$ and using $x_{n+1} - x_n = \epsilon_{n+1} - \epsilon_n$, we have

$$-\epsilon_n f'(\xi_n) = (\epsilon_{n+1} - \epsilon_n)f'(x_n),$$

or after dividing by $f'(x_n)$

$$\epsilon_{n+1} = \left(1 - \frac{f'(\xi_n)}{f'(x_n)}\right)\epsilon_n, \quad n = 0, 1, 2, \ldots.$$

Hence, if x_0 is sufficiently close to α, then $\lim_{n\to\infty} x_n = \alpha$. In other words α is a point of attraction of the Newton iteration and *Newton's method always converges (to a simple root) from a sufficiently good starting approximation.*

For convergence f need only have *one* continuous derivative. To get a more precise relation between ϵ_{n+1} and ϵ_n we assume in what follows that f has *two* continuous derivatives. Taking one more term in the Taylor series of f yields

$$0 = f(\alpha) = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{1}{2}(\alpha - x_n)^2 f''(\zeta_n), \quad \zeta_n \in \operatorname{int}(x_n, \alpha).$$

After subtracting equation (6.3.1) and dividing by $f'(x_n)$ we have

$$\epsilon_{n+1} = \epsilon_n^2 \frac{1}{2}\frac{f''(\zeta_n)}{f'(x_n)}, \quad \zeta_n \in \operatorname{int}(x_n, \alpha). \tag{6.3.7}$$

Provided that $f'(\alpha) \neq 0$, it follows that (6.2.5) is satisfied with $p = 2$ and we have

$$C = \frac{1}{2}\left|\frac{f''(\alpha)}{f'(\alpha)}\right|. \tag{6.3.8}$$

If $f''(\alpha) \neq 0$, then $C > 0$ and the rate of convergence is quadratic.

Note that the relation of equation (6.3.7) *between the errors only holds as long as the round-off errors in the calculations can be ignored.* The limiting factor for the accuracy which can be achieved in calculating the root is the accuracy of the computed values of $f(x)$. This will be discussed more fully in Section 6.5.

The following theorem gives a condition on x_0, which is sufficient for the global convergence of Newton's method.

Theorem 6.3.2. *Assume that $I = [a,b]$ is an interval containing the root α such that*

$$\frac{1}{2}\left|\frac{f''(y)}{f'(x)}\right| \le m, \quad \forall \;\; x,y \in I.$$

Let $x_0 = \alpha + \epsilon_0$ be chosen so that

$$[\alpha - |\epsilon_0|, \alpha + |\epsilon_0|] \subset I, \quad \textit{and} \quad |m\epsilon_0| = m|x_0 - \alpha| < 1.$$

Then $x_n \in I$ for all $n > 0$, and

$$|\epsilon_n| \le \frac{1}{m}(m\epsilon_0)^{2^n}. \tag{6.3.9}$$

Proof. If $x_n \in I$, then from equation (6.3.7) it follows that $|\epsilon_{n+1}| \le m\epsilon_n^2$, or equivalently $|m\epsilon_{n+1}| \le |m\epsilon_n|^2$. Since $x_0 \in I$, the proof follows by induction. $\square$

In practice, since α is of course unknown, the criterion in Theorem 6.3.2 cannot directly be applied. Below we state a sometimes more practical criterion for convergence.

Theorem 6.3.3. *Let x_0 be a given initial approximation and define x_n and h_n according to* (5.3.2). *Let $I_0 = \operatorname{int}(x_0, x_0 + 2h_0)$ and assume that*

$$2|h_0|M_2 \le |f'(x_0)|, \quad M_2 = \max_{x\in I_0} |f''(x)|.$$

Then $x_n \in I_0$, $n = 1,2,3,\ldots$, $\lim_{n\to\infty} x_n = \alpha$, and α is the only root of $f(x) = 0$ in I_0.

Proof. See Ostrowski [1973, Chap. 4]. $\square$

When a bound for the magnitude of the second derivative is not available the convergence criterion given in the following theorem may be easier to apply.

Theorem 6.3.4. *Suppose that $f'(x) \neq 0$, that $f''(x)$ does not change sign in the interval $[a,b]$ and that $f(a)f(b) < 0$. If*

$$\left|\frac{f(a)}{f'(a)}\right| < b - a, \qquad \left|\frac{f(b)}{f'(b)}\right| < b - a,$$

then Newton's method converges from an arbitrary $x_0 \in [a,b]$.

Proof. That the theorem is true can be easily seen from Figure 6.3.2. $\square$

In many practical problems one has a sufficiently good initial approximation for the desired root and convergence is not a problem. If this is not the case one can use a starting method with guaranteed convergence. Hence in practice it is most often neither possible nor necessary to analyze the convergence a priori.

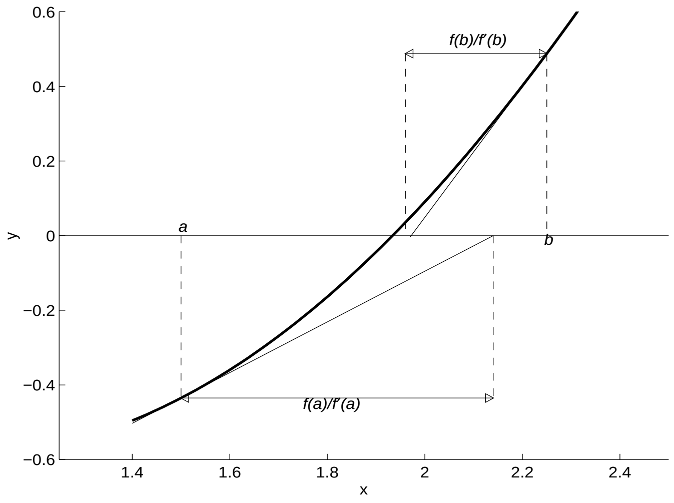


Figure 6.3.2. *A situation where Newton's method converges from any* $x_0 \in [a,b]$.

6.3.3 Safeguarded Newton Method

Newton's method is very efficient if started from an intial approximation close to simple zero. If this is not the case Newton's method may converge slowly or even diverge. Hence a more robust algorithm is needed, which retains the good asymptotic convergence of Newton's method. We now give an example of how such a method can be constructed by combining Newton's method with bisection.

Assume that a and b are known such that $a < b$ and $f(a)f(b) < 0$. At each step a new approximation x is computed and a, b are updated to a', b'. We take either $a' = x$, $b' = b$ or $a' = a$, $b' = x$, where the choice is made so that $f(a')f(b') \le 0$. To determine x we first check if the Newton iterate $z = x - f(x)/f'(x)$ lies in (a,b). If it does, we take $x = z$, otherwise we take a bisection step, i.e. set $x = (a+b)/2$.

We want to check if $z \in [a,b]$, or equivalently if

$$b - z = b - x + f(x)/f'(x) \ge 0 \quad \text{and} \quad z - a = x - a - f(x)/f'(x) \ge 0.$$

We want to perform the check in a way which avoids division by $f'(x)$, which may be close to zero. First assume that $f'(x) > 0$. Then these two inequalities are equivalent to

$$(b-x)f'(x) \ge -f(x) \quad \text{and} \quad (x-a)f'(x) \ge f(x).$$

(Only one of these will be nontrivial depending on whether $f(x) > 0$ or not.) The case when $f'(x) < 0$ is analyzed similarly and gives

$$(b-x)f'(x) \le -f(x) \quad \text{and} \quad (x-a)f'(x) \le f(x).$$

Algorithm 6.3.1 *Safeguarded Newton Method*

Let f be a given function and a and b are two initial approximations to a root in $[a, b]$ such that $f(a)f(b) \leq 0$. The following algorithm takes the next approximation as the Newton iterate if it lies in the interval $[a, b]$; otherwise the bisection point is chosen. The first step is always a bisection step. It terminates either when an interval $[a, b]$ with $|b - a| \leq tolx$ containing the root has been found or when $|f(x)| \leq tolf$ or when maxit iterations have been carried out.

```
$[x, fx] = \mathrm{safenewt}(f, fp, a, b, tolx, tolf, maxit);$
$fa = f(a);$   $fb = f(b);$
% It is assumed that $a < b$ **and** $fa \cdot fb <= 0$.
**if** $|fa| < |fb|$
    $x = a;$   $fx = fa;$ **else** $x = b;$   $fx = fb;$
**end**;
$it = 0;$
**while** $(|fx| > tolf)$ & $(|b - a| > tolx)$ & $(it < maxit)$
  $fpx = fp(x);$
% Is next Newton iterate strictly in $[a, b]$?
  **if** $sign(fx) = sign(fpx)$
    $xinab = |fx| < (x - a) \cdot |fpx|$
  **else**
    $xinab = |fx| < (b - x) \cdot |fpx|$
  **end**;
  **if** $xinab$
% Take Newton step from $(x, fx)$
    $x = x - fx/fpx;$   **else**
% Take bisection step
    $x = a + (b - a)/2;$
  **end**;
  $fx = f(x);$   $it = it + 1;$
  **if** $fa \cdot fx < 0$
    $b = x;$   $fb = fx;$   **else**
    $a = x;$   $fa = fx;$
  **end**;
**end**;
```

This algorithm is still not quite acceptable because the number of steps needed can in some cases be an arbitrarily factor larger than for the bisection method. A crude remedy would be to force it to take a bisection step at least every pth iteration for some fixed number $p > 1$. A more sophisticated improvement would be to accept the Newton iterate only if it is "sufficiently far away" from the current interval $[a, b]$ We return to the discussion of how that can be specified more precisely in Section 6.4.4

6.3.4 Higher Order Methods

A method of quadratic convergence will eventually converge very rapidly. As a rule of thumb one can say that the number of significant digits in each iteration will approximately double as was illustrated in Example 6.3.1. Similarly for a method of cubic convergence the significant digits will approximately triple. There is rarely any need for methods of higher order of convergence than two or three!

We first briefly review some famous methods with cubic convergence. Newton's method was derived by approximating the function $f(x)$ with its linear Taylor approximation. Taking one more term in the Taylor expansion of $f(x)$ at x_n we get the following approximation of the equation $f(x_n + h) = 0$:

$$T(h) = f(x_n) + hf'(x_n) + \frac{h^2}{2} f''(x_n) = 0, \quad h = x - x_n, \tag{6.3.10}$$

Assuming that $f'(x_n)^2 \geq 2f(x_n)f''(x_n)$ this quadratic equation has real solutions

$$h_n = -\frac{f'(x_n)}{f''(x_n)} \left(1 \pm \sqrt{1 - 2\frac{f(x_n)f''(x_n)}{(f'(x_n))^2}} \right).$$

Rearranging this and taking the solution of the smallest absolute value we get the iteration

$$x_{n+1} = x_n - u(x_n) \cdot \frac{2}{1 + \sqrt{1 - 2t(x_n)}}, \tag{6.3.11}$$

where

$$u(x) = \frac{f(x)}{f'(x)}, \qquad t(x) = \frac{f(x)f''(x)}{f'(x)^2}. \tag{6.3.12}$$

This is **Euler's iteration method**. Using the approximation

$$\frac{2}{1 + \sqrt{1 - 2t(x_n)}} \approx 1 + \tfrac{1}{2}t(x_n), \tag{6.3.13}$$

valid for $|t| \ll 1$, we obtain the third order iteration method

$$x_{n+1} = x_n - u(x_n)\big(1 + \tfrac{1}{2}t(x_n)\big), \tag{6.3.14}$$

usually also attributed to Euler.

A different third order iterative method is obtained by instead of (6.3.13) using a rational approximation

$$x_{n+1} = x_n - u(x_n) \cdot \frac{1}{1 - \frac{1}{2}t(x_n)}, \tag{6.3.15}$$

This is the famous **Halley's iteration method** [3][2], which also has the distinction of being the most frequently rediscovered iteration method. It can also be derived as follows. Starting from (6.3.10) we get

$$h_n = -f(x_n) \Big/ \Big(f'(x_n) + \frac{h_n}{2} f''(x_n) \Big).$$

[2]Edmund Halley (1656–1742), an English astronomer, who predicted the periodic reappearance (c:a 75 years) of a comet named after him.

Replacing h_n in the denominator by the Newton correction $-f(x_n)/f'(x_n)$, again gives Halley's method.

The methods (6.3.14) and (6.3.15) are related to the (1,0) and (0,1) Padé approximations. We now show that both are of third order for simple zeros. Consider an iteration function of the general form

$$\phi(x) = x - u(x)H(t(x)), \tag{6.3.16}$$

where $u(x)$ and $t(x)$ are defined by (6.3.5) and (6.3.6). Differentiating (6.3.16) and using $u'(x) = 1 - t(x)$ we get

$$\phi'(x) = 1 - (1 - t(x))H(t) - u(x)H'(t)t'(x).$$

Since $u(\alpha) = t(\alpha) = 0$ it follows that $\phi'(\alpha) = 1 - H(0)$, and so $\phi'(\alpha) = 0$ and the iteration function (6.3.16) of at least second order if $H(0) = 1$. Differentiating once more and putting $x = \alpha$ we get

$$\phi''(\alpha) = t'(\alpha)H(0) - 2u'(\alpha)H'(0)t'(\alpha) = t'(\alpha)(H(0) - 2H'(0)).$$

Hence $\phi'(\alpha) = \phi''(\alpha) = 0$ and the method (6.3.16) of at least third order if the conditions

$$H(0) = 1, \qquad H'(0) = 1/2 \tag{6.3.17}$$

are satisfied. For Euler's and Halley's method we have

$$H_E(t) = 2(1 + \sqrt{1 - 2t})^{-1}, \qquad H_H(t) = (1 - t/2)^{-1},$$

respectively, and both these methods satisfy the conditions in (6.3.17).

Example 6.3.3.

Using (6.3.12)–(6.3.15) a short calculation shows that Halley's method for solving the equation $f(x) = x^2 - c = 0$ can be written

$$x_{n+1} = x_n - 2x_n \frac{x_n - c/x_n}{3x_n + c/x}, \quad n = 0, 1, 2, \ldots. \tag{6.3.18}$$

(see Problem 8). Assuming that $1/4 < c \leq 1$, the initial approximation $x_0 = (1 + 2c)/3$ obtained by linear interpolation in Example 6.3.2 can be used. For $c = 0.5$ we obtain the following result (correct digits in boldface)

$$x_0 = 0.66666666666667, \quad x_1 = \mathbf{0.707}07070707071, \quad x_2 = \mathbf{0.7071067811865}2,$$
$$x_3 = \mathbf{0.70710678118655}.$$

Already two iterations give a result correct to 13 digits. Compared to Newton's method we have gained almost two iterations.

There are several ways to construct iteration methods of higher order for solving the equation $f(x) = 0$. The earlist derivation of a basic family of methods of arbitrary order is due to E. Schröder [10]. By Taylors formula we have

$$f(x + h) = f(x) + hf'(x) + \sum_{k=2}^{p-1} \frac{h^k}{k!} f^{(k)}(x) + O(h^p). \tag{6.3.19}$$

Assume that $f'(x) \neq 0$ and set $f(x+h) = 0$. Neglecting the $O(h^p)$-term we obtain

$$-u = h + \sum_{k=2}^{p-1} a_k h^k, \tag{6.3.20}$$

where

$$u = \frac{f(x)}{f'(x)}, \qquad a_k = \frac{f^{(k)}(x)}{k! f'(x)}, \quad k \geq 2. \tag{6.3.21}$$

Reversing the power series (6.3.20) (cf. Sec. 3.1.3) we obtain

$$h = -u - \sum_{k=2}^{p-1} c_k u^k. \tag{6.3.22}$$

This defines a series of iteration functions $\phi_p(x) = x + h(x)$ of order p. In particular we have

$$c_2 = a_2, \qquad c_3 = 2a_2^2 - a_3, \qquad c_4 = 5a_2^3 - 5a_2 a_3 + a_4,$$

Using the first two terms in the sum we get Euler's method (6.3.14). Variants of these high order methods can be derived from the family (6.3.21) by computing different approximants in a Padé table. However, as pointed out before, methods of order $p > 3$ are of practical interest only in special cases.

Kalantari [6] has given a compact determinantal formulation of a related basic family of iteration fucntion. Set

$$D_p(x) = \det \begin{pmatrix} f'(x) & \frac{f''(x)}{2!} & \dots & \frac{f^{(p-1)}(x)}{(p-1)!} & \frac{f^{(p)}(x)}{(p)!} \\ f(x) & f'(x) & \ddots & \ddots & \frac{f^{(p-1)}(x)}{(p-1)!} \\ 0 & f(x) & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \frac{f''(x)}{2!} \\ 0 & 0 & \dots & f(x) & f'(x) \end{pmatrix}.$$

Note that $D_p(x)$ corresponds to the determinant of a Toeplitz matrix defined with respect to the normalized derivatives of $f(x)$. (Recall that a square matrix is called Toeplitz if its elements are identical along each diagonal.) For each $p \geq 2$, define

$$B_p(x) \equiv x - f(x) \frac{D_{p-2}(x)}{D_{p-1}(x)}. \tag{6.3.23}$$

A root θ of f is a fixed-point of B_p, i.e. $f(\alpha) = 0$ implies $B_p(\alpha) = \alpha$. It can be shown that B_p defines an iteration method of order p. The third order method in this family reads

$$B_p(x) = x - f(x) \frac{f'(x)}{\det \begin{pmatrix} f'(x) & \frac{f''(x)}{2!} \\ f(x) & f'(x) \end{pmatrix}},$$

which is Halley's method (6.3.15).

Review Questions

1. Define the order of convergence and the asymptotic error constant for a convergent sequence $\{x_k\}$. What is meant by quadratic convergence, and superlinear convergence?

2. (a) Under what assumptions is convergence of Newton's method quadratic?

(b) Device an example where Newton's method diverges, even though the equation has real roots.

3. The equation $f(x) = \sin x = 0$ has one trivial root $x = 0$ in the interval $(-\pi/2, \pi/2)$. Show that for an initial approximation x_0 chosen so that $\tan x_0 = 2x_0$ Newton's method cycles, and $x_{2k} = x_0$ for all $k \geq 0$!

Problems

1. (a) Compute $\epsilon_{n+1}/\epsilon_n^2$ for $n = 0, 1, 2$, and the limit, as $n \to \infty$, in Example 6.3.1.

(b) Treat the equation in Example 6.3.1 making use $f'(x_2)$ also for $n > 2$. Compare the convergence with this simplification.

2. The equation $x^3 - 2x - 5 = 0$ is of historical interest because it was the one used by Wallis to exhibit Newton's method when he first published it. Determine the real root of this equation.

3. For $p = 2$, the iteration (6.3.3) is the same algorithm for computing square roots which was given in Example 1.2.1. Show that in this case

$$x_{n+1} - \sqrt{c} = \frac{1}{2x_n}(x_n - \sqrt{c})^2.$$

Use this relation to show that for all $x_0 > 0$, we have that $x_1 \geq x_2 \geq x_3 \geq \cdots \geq \sqrt{c}$ and that $\lim_{n\to\infty} = \sqrt{c}$

4. Describe an iteration for the division-free computation of the reciprocal of a positive number c. Determine the largest set of starting values x_0 such that the iterates converge to c.

5. Determine p, q and r so that the order of the iterative method

$$x_{n+1} = px_n + qc/x_n^2 + rc^2/x_n^5$$

for computing $\sqrt[3]{c}$ becomes as high as possible. For this choice of p, q and r, give a relation between the error in x_{n+1} and the error in x_n.

6. Use Newton's method to determine the positive root of the equation to six correct decimals: (a) $x = 1 - e^{-2x}$; (b) $x \ln x - 1 = 0$

7. Apply Newton's method to determine one of the complex roots of the equation $z^2 + 1 = 0$. Start with $z_0 = 1 + i$.

8. (a) Show that Halley's method can be derived by applying Newton's method to the equation $f(x)(f'(x))^{-1/2} = 0$.

(b) Show that Halley's method applied to $f(x) = x^2 - c = 0$ gives rise to the iteration

$$x_{n+1} = \frac{x_n^3 + 3cx_n}{3x_n^2 + c} = x_n - \frac{2x_n(x_n^2 - c)}{3x_n^2 + c}.$$

9. To compute the square root of the normalized floating point number $a = m \cdot 2^q$, $1/2 \le m < 1$ by Newton's method we first shift the mantissa so that the exponent becomes even, $a = c \cdot 2^{2e}$, and $1/2 \le c < 2$. Then its square root is

$$\sqrt{a} = \sqrt{c} \cdot 2^e,$$

and thus we only need to compute $\sqrt{c}$ for $1/2 \le c \le 1$. (For $1 < c < 2$ we compute $\sqrt{1/c}$ and invert.)

(a) What starting value minimizes the maximum initial error over all c in the interval $[1/2, 1]$.

(b) Use the expression for the error in Problem 4 to determine the minimum number of iterations that suffices to give $\sqrt{c}$ with an error less than 10^{-9} for *all* c in $[1/2, 1]$ using Newton's method and the initial approximation from (a). Round-off errors may be neglected.

Computer Exercises

1. (a) Given $\sigma_1 \ge \sigma_1 \ge \ldots \ge \sigma_n > 0$, $c_1, c_2, \ldots c_n$, and $\alpha > 0$, consider the secular equation $\phi(\lambda) = \alpha$, where

$$\phi^2(\lambda) = \sum_{i=1}^{n} \left(\frac{\sigma_i c_i}{\sigma_i^2 + \lambda} \right)^2 .$$

Show that $\phi(\lambda)$ is a convex and strictly decreasing function of λ. Conclude that if $\phi(0) > \alpha$, this equation has a unique root $\lambda > 0$ of smallest magnitude.

(b) Newton's method for solving $\phi(\lambda) - \alpha = 0$ is

$$\lambda_{k+1} = \lambda_k - h_k, \quad h_k = \frac{\phi(\lambda_k) - \alpha}{\phi'(\lambda_k)}.$$

Show that with $\lambda_0 = 0$ this method produces a strictly increasing sequence λ_k converging to the solution. Derive an explicit expression for h_k.

(c) The Newton iteration in (b) often converges very slowly. A more efficient method is obtained by instead applying Newton's method to the equation $h(\lambda) = 1/\phi(\lambda) = 1/\alpha$. Show that this iteration can be written

$$\lambda_{k+1} = \lambda_k - h_k \frac{\phi(\lambda_k)}{\alpha},$$

where h_k is the Newton step in (b).

(d) Let $\sigma_i = 1/i^2$, $c_i = 1/i^2 + 0.001$, $i = 1, 2, \ldots, 20$. Plot the function $f(\lambda)$ for $\lambda \in (0, 0.0005)$. Solve the equation $\phi(\lambda) = \alpha = 2.5$, using $\lambda_0 = 0$, comparing the two methods in (b) and (c).

6.4 Methods Based on Interpolation

6.4.1 The Secant Method

The necessity to calculate $f'(x)$ in Newton's method is sometimes a disadvantage, since $f'(x)$ may not be available or require a considerable computational effort.

In the **secant method** the derivative at x_n is approximated by the difference quotient,[3]

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

The secant method requires two initial approximations x_0 and x_1. The sequence $x_2, x_3, \dots$ is then computed by

$$x_{n+1} = x_n + h_n, \quad h_n = -f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}, \quad n \geq 1, \tag{6.4.1}$$

assuming that $f(x_n) \neq f(x_{n-1})$. Note that this iteration is of the form $x_{n+1} = \phi(x_n; x_{n-1})$. Hence, in contrast to Newton's method it is not a fixed point iteration of the form studied in Section 6.2. Sometimes such iteration methods are called fixed point iterations with memory, since it also uses the old information x_{n_1}.

The geometrical interpretation of the secant method is that x_{n+1} is the abscissa of the point of intersection between the secant through $(x_{n-1}, f(x_{n-1}))$ and $(x_n, f(x_n))$ and the x-axis, see Figure 6.4.1. Notice that the secant method only requires *one function evaluation per step.*

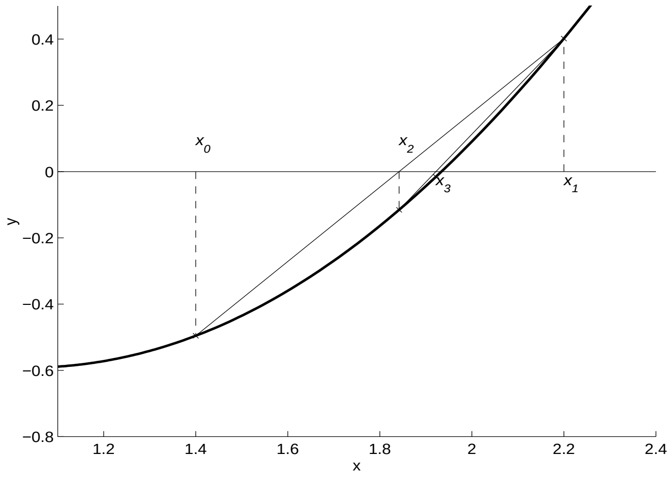


Figure 6.4.1. *The secant method.*

When $|x_n - x_{n-1}|$ is small, the quotient $(x_n - x_{n-1})/(f(x_n) - f(x_{n-1}))$ will in general be determined with poor relative accuracy. If by accident we get approximations x_n and x_{n-1} which are very close to the root α, and not bracketing α, the resulting rounding error in x_{n+1} can become very large. However, from the error analysis below it follows that the secant method in general gives a sequence such that $|x_n - x_{n-1}| \gg |x_n - \alpha|$. In this case the dominant contribution to the roundoff error in h_n comes from the error in $f(x_n)$, and is not larger than for Newton's

[3]Historically the secant method preceded Newton's method.

method. Note that (6.4.1) should *not* be rewritten in the form

$$x_{n+1} = \frac{x_{n-1}f_n - x_n f_{n-1}}{f_n - f_{n-1}},$$

since this formula can give rise to severe difficulties with *cancellation* when $x_n \approx x_{n-1}$ and $f_n f_{n-1} > 0$.

Example 6.4.1.

We use the same equation $f(x) = (x/2)^2 - \sin x = 0$ as in previous examples to examine the convergence of the secant method, and take $x_0 = 1.5, x_1 = 2$.

n	x_n	$f(x_n)$	h_n
0	1.5	−0.434994 986604	
1	2.0	+0.090702 573174	−0.086268 778965
2	**1.9**13731 221035	−0.026180 060742	+0.019322 989205
3	**1.933**054 210240	−0.000924 399645	+0.000707 253882
4	**1.9337**61 464122	+0.000010 180519	−0.000007 704220
5	**1.933753 75**9902	−0.000000 003867	+0.000000 002925
6	**1.933753 762827**		

In this example, the secant method converges almost as fast as Newton–Raphson's method, and x_5 is again correct to eight decimals. This is because $x_1 = 2$ is quite a good initial approximation. However, in the example in Figure 6.4.1 we see that the iterate x_3 will lie outside the initial interval $[x_0, x_1]$.

We remark that even (6.4.1) is not always safe to use. We must take care to avoid overflow or division by zero. Without restriction we can assume that $|f_{n-1}| \geq |f_n| > 0$ (otherwise renumber the two points). Then, s_n can be computed without risk of overflow from

$$h_n = \frac{s_n}{1 - s_n}(x_n - x_{n-1}), \quad s_n = \frac{f_n}{f_{n-1}},$$

where the division with $1 - s_n$ is only carried out if $1 - s_n$ is large enough.

6.4.2 Local Convergence of the Secant Method

We shall now derive an asymptotic formula for the errors in the successive approximations produced by the secant method. According to Newton's interpolation formula with error term, Theorem 4.3.1, we have

$$f(x) = f(x_n) + (x - x_n)f[x_{n-1}, x_n] + (x - x_{n-1})(x - x_n)\frac{f''(\zeta_n)}{2}, \tag{6.4.2}$$

where $\zeta_n \in \operatorname{int}(x, x_{n-1}, x_n)$ and

$$f[x_{n-1}, x_n] = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

If we ignore the error term, we get the equation of the secant. Thus x_{n+1} satisfies the equation

$$0 = f(x_n) + (x_{n+1} - x_n)f[x_{n-1}, x_n].$$

Now put $x = \alpha$ in (6.4.2) and subtract the equation of the secant. Since $f(\alpha) = 0$ we get

$$(\alpha - x_{n+1})f[x_{n-1}, x_n] + (\alpha - x_{n-1})(\alpha - x_n)f''(\zeta_n)/2 = 0.$$

According to the mean-value theorem, we have

$$f[x_{n-1}, x_n] = f'(\zeta_n'), \quad \zeta_n' \in \operatorname{int}(x_{n-1}, x_n),$$

and it follows that

$$\epsilon_{n+1} = \frac{f''(\zeta_n)}{2f'(\zeta_n')}\epsilon_n\epsilon_{n-1}. \tag{6.4.3}$$

Notice that if we let $x_{n-1} \to x_n$, then the error formula (6.4.3) becomes identical with the error formula (6.3.7) for Newton's method.

Example 6.4.2.

The ratios $\epsilon_{n+1}/(\epsilon_n\epsilon_{n-1})$ in Example 6.4.1 are equal to

$$0.697,\ 0.527,\ 0.550, \quad n = 1, 2, 3,$$

which compares well with the limiting value 0.543 of $f''(\alpha)/(2f'(\alpha))$

It is easy to show that if the first derivative $f'(x)$ is continuous, then for the secant method it holds that

$$\epsilon_{n+1} = \left(1 - \frac{f'(\xi_n)}{f'(\zeta_n)}\right)\epsilon_n, \quad \xi_n \in \operatorname{int}(x_{n-1}, \alpha), \quad \zeta_n \in \operatorname{int}(x_n, x_{n-1}).$$

From this it can be deduced that the secant method always converges from sufficiently good starting values x_0, x_1.

The following theorem gives the order of convergence for the secant method.

Theorem 6.4.1. *Suppose that in a neighborhood I of the root α we have*

$$\frac{1}{2}\left|\frac{f''(y)}{f'(x)}\right| \le m, \quad x, y \in I.$$

Let $p = (1 + \sqrt(5))/2 = 1.618\ldots$ be the positive root of the equation $\mu^2 - \mu - 1 = 0$. Then it holds that

$$|\epsilon_n| \le \frac{1}{m}K^{p^n}, \quad K = \max\left(m|\epsilon_0|, (m|\epsilon_1|)^{1/p}\right), \quad n = 0, 1, 2, \ldots. \tag{6.4.4}$$

Proof. The proof is by induction. From the choice of K it follows that equation (6.4.4) is trivially true for $n = 0, 1$. From the assumption and (6.4.3) we have

$$|\epsilon_{n+1}| \le m|\epsilon_n||\epsilon_{n-1}| \tag{6.4.5}$$

Since $p^2 = p + 1$ it follows that if (6.4.4) holds for n and $n - 1$ then

$$|\epsilon_{n+1}| \leq \frac{1}{m} K^{p^n} K^{p^{n-1}} = \frac{1}{m} K^{p^{n+1}}.$$

□

If each iteration requires m units of work (usually the work involved in computing a function value or a value of one of its derivatives) then the of the iteration may be defined as

To compare the efficiency of the secant method and Newton's method we assume that the work to compute $f'(x)$ is θ times the amount of work required to compute $f(x)$. Then, with the same amount of work we can perform $k(1+\theta)$ iterations with the secant method and k iterations with Newton's method. Equating the errors we get $(m\epsilon_0)^{2^k} = (m\epsilon_0)^{p^{k(1+\theta)}}$, where $p = 1.618...$ Hence the errors are the same for both methods when $p^{k(1+\theta)} = 2^k$ or

$$(1+\theta)\log\left(\tfrac{1}{2}(1+\sqrt{5})\right) = \log 2,$$

which gives $\theta = 0.4404\ldots$. Thus, from this analysis we conclude that if $\theta > 0.44$, then the secant method is asymptotically more efficient than Newton's method.

In Example 6.4.1 we can observe that the error $\epsilon_n = x_n - \alpha_n$ (and also the value $f(x_n)$) changes sign at every third step. Hence in this example

$$\alpha \in \operatorname{int}(x_{n+1} - x_n), \quad n = 0, 1, 3, 4, \ldots,$$

i.e., *the root α is bracketed by x_{n+1} and x_n except for every third step.* We shall show that this is no coincidence. Assume that $x_n \in (a, b)$, $n = 0, 1, 2, \ldots$, and that $f'(x)$ and $f''(x)$ have constant sign in (a, b). Then from (6.4.3) it follows that

$$\epsilon_{n+1}/(\epsilon_n \epsilon_{n-1})$$

has constant sign for all n. If $\alpha \in \operatorname{int}(x_0, x_1)$ then $\epsilon_0\epsilon_1 < 0$, and it follows that the sign of ϵ_n must change every third step (verify this!). Hence convergence occurs in a waltz rhythm! If on the other hand the initial approximations to the secant method are chosen so that $f(x_0)f(x_1) > 0$, then convergence will be monotone. Since Newton's method can be viewed as the limit when both interpolation points in the secant method coincides, it will converge monotonically under the same assumptions.

6.4.3 False Position Method

For the secant method we noted that the last two iterates x_{n+1} and x_n will not always bracket the root. Therefore there is no guarantee that the computed approximations are contained in $[x_0, x_1]$ and, like Newton's method, the secant method may diverge even when the initial approximations x_0 and x_1 bracket the root.

Suppose that at a certain step we have two approximations x_n and x_{n-1} to a root for which $f_n f_{n-1} < 0$ and that x_{n+1} has been computed by a secant step according to (6.4.1). In the **false-position method** (Latin **regula falsi**)[4] one takes in the next step the secant through the points x_{n+1} and whichever of x_n and x_{n-1} for which the function value is of opposite sign to f_{n+1}. The advantage of this method is that convergence to a root is guaranteed for a continuous function $f(x)$.

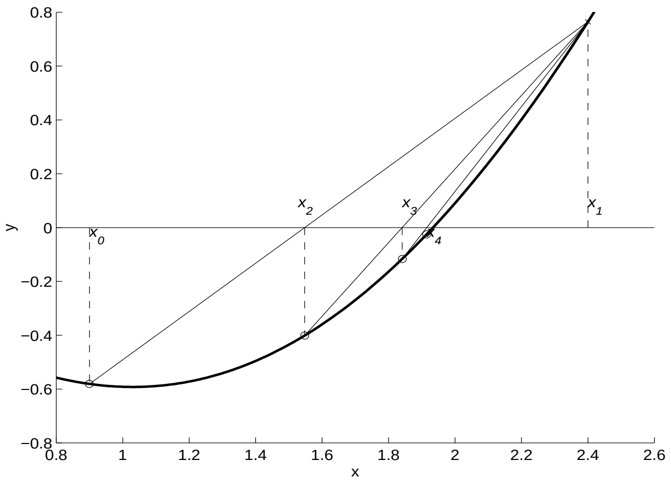


Figure 6.4.2. *The false-position method.*

A drawback is that in contrast to the secant method regula falsi only has a *linear rate of convergence.* This is because once an interval has been reached on which the $f(x)$ is convex or concave, one of the endpoints is always retained. In Figure 6.4.3 successive secants will *all* pass through the point $(x_0, f(x_0))$. Using $f(x_n) = f'(\zeta_n)(x_n - \alpha)$ we obtain

$$\epsilon_{n+1} = \epsilon_n - \frac{f'(\zeta_n)\epsilon_n}{(f_0 - f_n)/(x_0 - x_n)}.$$

If $\lim_{n\to\infty} = \alpha$, it follows that

$$\lim_{n\to\infty} \frac{|\epsilon_{n+1}|}{|\epsilon_n|} = 1 - \frac{f'(\alpha)}{f_0/\epsilon_0} = C, \tag{6.4.6}$$

which shows that convergence is linear if $C \neq 0$. If f'' is continuous, the same conclusion follows also directly from (6.4.3).

Example 6.4.3.

We apply the regula falsi to the same equation as in Example 6.4.1 using again the initial approximations $x_0 = 1.5, x_1 = 2$.

[4]The method of regula falsi is very old, originating in 5th century Indian texts.

n	x_n	f_n	h_n
0	1.5	−0.434994986604	
1	2.0	+0.090702573174	+0.086268778965
2	**1.9**13731 221035	−0.026180060742	−0.019322989205
3	**1.933**054 210240	−0.000924399645	−0.000675397892
4	**1.9337**29 608132	−0.000031930094	−0.000023321005
5	**1.933752** 929137	−0.000001102069	−0.000000804916
6	**1.933753 7**34053		

Note that the approximations x_2 and x_3 are the same as for the secant method. In all following iterations the point $x_1 = 2$ is kept and convergence becomes linear with rate $C \approx 0.034$.

6.4.4 Safeguarded Secant Method

Although the above algorithm is quite efficient and reliable it can converge much slower than the bisection method, e.g., when α is an ill-conditioned root. Efficient and robust root finders can be constructed by combining the secant method, or some higher order interpolation method with bisection, cf. the safeguarded Newton method discussed in Section 6.3.3. A simple variant of such a method can proceed as follows.

Let a and b such that $f(a)f(b) < 0$ be given and set $c = a$. In the following b denotes the last iterate, a the one preceding it and c the most recent iterate such that $f(c)f(b) < 0$. At each step we update a, b, c to a', b', c'. If $f(a)f(b) < 0$ the new iterate b' is computed from a secant step using a and b. If $f(a)f(b) > 0$ we check implicitly (to avoid possible division by zero) if the result of a secant step will lie in $\operatorname{int}[b, c]$. If it does we take a secant step, otherwise the new iterate is computed from a bisection step using the points bracketing the root, $b' = (b + c)/2$. Finally, we set $c' = c$ or $c' = a$

A particularly elegant combination of bisection and the secant method was developed in the 1960th by van Wijngaarden, Dekker and others at the Mathematical Center in Amsterdam. A related algorithm, called ZEROIN, which combines bisection and inverse quadratic interpolation, was published by Brent [1]. The Matlab function "fzero", which finds a zero near a given approximation x_0, is based on the Fortran implementation of ZEROIN in Forsythe, Malcolm, and Moler [2, pp.161–166].

6.4.5 Higher Order Interpolating Methods

In the secant method we used linear interpolation through (x_{n-1}, f_{n-1}) and (x_n, f_n) to determine the next approximation to the root. A natural generalization is to use $r+1$ different approximations $x_n, x_{n-1} \dots, x_{n-r}$ to determine an interpolating polynomial $P(x)$ of degree r and choose the root closest to x_n as the new approximation x_{n+1}. In practice only the case $r = 2$ is of interest, since for $r > 2$ there are no useful formulas for determining the roots of the interpolating polynomial $P(x)$. For $r = 2$ we get the **Muller–Traub method**, which we describe below.

By Newton's interpolating formula (4.3.4) the interpolating polynomial $P(x)$ can be written

$$P(x) = f_n + f[x_n, x_{n-1}](x - x_n) + f[x_n, x_{n-1}, x_{n-2}](x - x_n)(x - x_{n-1}).$$

Setting $h_n = (x - x_n)$ the equation $P(x) = 0$ becomes

$$f[x_n, x_{n-1}, x_{n-2}]h_n^2 + \omega h_n + f_n = 0, \tag{6.4.7}$$

where

$$\omega = f[x_n, x_{n-1}] + (x_n - x_{n-1})f[x_n, x_{n-1}, x_{n-2}]. \tag{6.4.8}$$

Since we want the root closest to x_n, that means we want the smallest root h_n to the equation (6.4.7). To express the smallest root in a numerically stable way the standard formula for the roots of a quadratic equation should be multiplied by its conjugate quantity (see Example 2.3.3). Using this formula we get

$$x_{n+1} = x_n + h_n, \quad h_n = -\frac{2f_n}{\omega \pm \sqrt{\omega^2 - 4f_n f[x_n, x_{n-1}, x_{n-2}]}}. \tag{6.4.9}$$

Here the sign in the denominator should be chose so as to minimize the $|h_n|$. Note that equation (6.4.7) may have complex roots even if the zero being sought is real. On the other hand, the Muller–Traub method has the useful property that complex roots can be found from real starting approximations.

It can be shown, using a similar proof as for the secant method, that the Muller–Traub method is at least of order $p = 1.839\ldots$, where p is the largest root of the equation $\mu^3 - \mu^2 - \mu - 1 = 0$. Hence this method does not quite achieve quadratic convergence. In fact, it can be shown under very weak restrictions that no iterative method using only one function evaluation can have $p \geq 2$.

A different way to extend the secant method is to use *inverse interpolation.* We then consider $x_n, x_{n-1}, \ldots, x_{n-r}$ as given values of the *inverse function* $g(f)$ of $f(x)$ at distinct points $f_n, f_{n-1}, \ldots, f_{n-r}$. Since $x = g(f)$ it follows that $\alpha = g(0)$ is the desired root. We now fit a polynomial $Q(f)$ to to the points and obtain the next approximation from $x_{n+1} = Q(0)$. For $r = 1$ this is again the secant method. For $r > 1$ this is a fundamentally different procedure from before and as a rule gives different results. We consider now the special case $r = 2$.

Using Newton's general interpolation formula we get

$$Q(f) = g(f_n) + (f - f_n)g[f_n, f_{n-1}] + (f - f_n)(f - f_{n-1})g[f_n, f_{n-1}, f_{n-2}].$$

If we note that $g(f_n) = x_n$ and put $f = 0$ we get

$$x_{n+1} = x_n - f_n g[f_n, f_{n-1}] + f_n f_{n-1} g[f_n, f_{n-1}, f_{n-2}]. \tag{6.4.10}$$

Inverse quadratic interpolation has the same order of convergence as the Muller–Traub method and it has the advantage of not requiring the solution of a quadratic equation. (For other ways of avoiding this see Problems 2 and 4.) It is used together with the bisection method in the algorithm ZEROIN by Brent [1], who claims that on the average this saves 0.5 function evaluations over using the secant method.

It should be noted that it is not always safe to use the formula in (6.4.10). Care has to be taken in order to avoid overflow and possibly division by zero. If we assume that $0 \neq |f_n| \leq |f_{n-1}| \leq |f_{n-2}|$ then it is safe to compute

$$s_n = f_n/f_{n-1}, \quad s_{n-1} = f_{n-1}/f_{n-2}, \quad r_n = f_n/f_{n-2} = s_n s_{n-1}.$$

We can rewrite (6.4.10) in the form $x_{n+1} = x_n + p_n/q_n$, where

$$\begin{aligned} p_n &= s_n[(1-r_n)(x_n - x_{n-1}) - s_{n-1}(s_{n-1} - r_n)(x_n - x_{n-2})], \\ q_n &= (1-s_n)(1-s_{n-1})(1-r_n). \end{aligned}$$

The final division p_n/q_n is only carried out if the correction is sufficiently small, see Brent [1] for a detailed discussion.

Review Questions

1. How does regula falsi differ from the secant method? Why does it in general only have linear convergence?

2. Outline how the secant method can be safeguarded by combining it with the bisection method.

Problems

1. Use the secant method to determine the roots of the following equations to six correct decimals

(a) $2x = e^{-x}$; (b) $\tan x + \cosh x = 0$.

2. Another modification of the secant method can be derived by estimating $f'(x_n)$ in Newton's method by quadratic interpolation through the points x_n, x_{n-1}, x_{n-2}. Show that the resulting method can be written $x_{n+1} = x_n - f(x_n)/\omega$, where

$$\omega = f[x_n, x_{n-1}] + (x_n - x_{n-1}) f[x_n, x_{n-1}, x_{n-2}].$$

3. Assume that we have $f_n f_{n-1} < 0$, and have computed x_{n+1}. If $f_{n+1} f_n < 0$ then in the next step we compute x_{n+2} using a secant through (x_{n+1}, f_{n+1}) and (x_n, f_n). Otherwise, if $f_{n+1} f_n > 0$, we use a line through (x_{n+1}, f_{n+1}) and $(x_{n-1}, \theta f_{n-1})$, where $0 < \theta < 1$. Clearly, $\theta = 1$ would correspond to a regula falsi step and usually give $f_{n+2} f_{n+1} > 0$. On the other hand, $\theta = 0$ gives $x_{n+1} = x_n$, and thus $f_{n+1} f_n < 0$. Hence a suitable choice of θ will always give $f_{n+2} f_{n+1} < 0$.
Show that with $\theta = 0.5$ in a modified step it holds asymptotically $\epsilon_{n+1} \approx -\epsilon_n$. Deduce that the resulting algorithm gives cubic convergence with three function evaluations and hence has efficiency index $E = 3^{1/3} = 1.4422\ldots$. [5]

[5]The resulting modified rule of false position is often called after its origin the **Illinois method**. It is due originally to the staff of the computer center at the University of Illinois in the early 1950's.

4. The Muller–Traub method uses three points to determine the coefficient of an interpolating parabola. The same points can also be interpolated by a rational function of the form

$$g(x) = \frac{x-a}{bx+c}.$$

An iterative method is devised by taking x_{n+1} equal to the root a of $g(x) = 0$.

(a) Show that this is equivalent to calculating x_{n+1} from the "modified secant formula"

$$x_{n+1} = x_n - f_n \frac{x_n - x_{n-2}}{f_n - \tilde{f}_{n-2}}, \qquad \tilde{f}_{n-2} = f_{n-2}\frac{f[x_n, x_{n-1}]}{f[x_{n-1}, x_{n-2}]}.$$

Hint: Use a theorem in projective geometry, according to which the cross ratio of any four values of x is equal to the cross ratio of the corresponding values of $g(x)$ (see Householder [1970, p.159]). Hence

$$\frac{(0-f_n)/(0-f_{n-2})}{(f_{n-1}-f_n)/(f_{n-1}-f_{n-2})} = \frac{(x_{n+1}-x_n)/(x_{n+1}-x_{n-2})}{(x_{n-1}-x_n)/(x_{n-1}-x_{n-2})}.$$

(b) Use the result in (a) to show that $x_{n-1} \in \text{int}(x_{n-2}, x_n)$ if

$$\text{sign}(f_n) = -\text{sign}(f_{n-2}), \qquad \text{sign}(f[x_n, x_{n-1}]) = \text{sign}(f[x_{n-1}, x_{n-2}]).$$

Computer Exercises

1. Implement a safeguarded version of the secant method based on the outline at the end of Sec. 6.4.3 and the safeguarded Newton algorithm in Sec. 6.3.3.

2. The result in Problem 4 suggest that the Illinois method is modified by taking

$$\beta = f[x_{n+1}, x_n]/f[x_n, x_{n-1}], \qquad \theta = \begin{cases} \beta, & \text{if } \beta > 0; \\ \frac{1}{2}, & \text{if } \beta \le 0. \end{cases}.$$

Implement this modified method. Compare it with the unmodified Illinois method and with the safeguarded secant algorithm. As test equations use the following:

(a) $f(x) = x^2 - (1-x)^n$, $a = 25$, $b = 1$, $n = 2, 5, 10$; one inflection point on $[0, 1]$.

(b) $f(x) = e^{-nx}(x-1) + x^n$, $a = 0.25$, $b = 1$, $n = 5, 10, 15$; A family of curves which lie increasingly close to the x-axis for large n.

(c) $f(x) = (nx-1)/((n-1)x)$, $a = 0.01$, $b = 1$, $n = 2, 5, 10$; A family of curves with the y-axis asymptotic.

6.5 Attainable Accuracy and Multiple Roots

6.5.1 Error Estimation

In the previous analysis of iteration methods we studied the asymptotic convergence rate of x_n to a root as $n \to \infty$, disregarding round-off errors. However, in practice computed function values of f will be affected by errors, e.g., roundoff errors in floating point computation. In other words, we are dealing not with the true function $f(x)$ but with a contaminated function $\bar{f}(x) = f(x) + \delta(x)$, where $\delta(x)$ denotes the error in evaluating $f(x)$. Note that $\bar{f}$ in general is not even continuous on $[a,b]$. Even if $\bar{f}(a)\bar{f}(b) < 0$, the equation $\bar{f}(x) = 0$ may not have a zero in $[a,b]$!

We now derive an error estimate, which takes into account that the computed values of $f(x)$ are subject to errors. Let α be a *simple* root of the equation $f(x) = 0$. Assume that for any x in a neighborhood of α the *computed value* $\bar{f}(x)$ of $f(x)$ satisfies

$$|\bar{f}(x) - f(x)| \le \delta. \tag{6.5.1}$$

Here δ is an upper bound for roundoff and other errors in computed function values of $f(x)$. We assume that $f'(x)$ is continuous in a neighborhood J of α containing an approximation x_n to a simple root α. Then, by the mean-value theorem

$$f(x_n) = (x_n - \alpha)f'(\zeta_n), \quad \zeta_n \in \operatorname{int}(x_n, \alpha),$$

from which we obtain the estimate

$$|x_n - \alpha| \le |f(x_n)|/M_1, \quad |f'(x)| \ge M_1, \quad x \in J. \tag{6.5.2}$$

Using (6.5.2) we get the error estimate

$$|x_n - \alpha| \le (|\bar{f}(x_n)| + \delta)/M_1. \tag{6.5.3}$$

The best we can hope for is to find an approximation x_n such that the computed function value $\bar{f}(x_n)$ is zero. It follows that independent on which method is used the accuracy with which a simple root α can be determined is limited by δ/M_1. If $f'(x)$ does not vary much near $x_n = \alpha$, then we have the approximate error bound

$$|x_n - \alpha| \le \delta/M_1 \approx \epsilon_\alpha, \quad \epsilon_\alpha = \delta/|f'(\alpha)|. \tag{6.5.4}$$

Since this is the best error bound for *any* method we call ϵ_α, the **attainable accuracy** for the simple root α. Note that We call the interval $[\alpha - \epsilon_\alpha, \alpha + \epsilon_\alpha]$ the **domain of uncertainty** for the root α. if $|f'(\alpha)|$ is small, then ϵ_α is large and the problem of computing the root α is ill-conditioned (see Figure 6.5.1).

Example 6.5.1.

Suppose we have computed the approximation $x = 1.93375$ to the positive root to the equation $f(x) = \sin x - (x/2)^2$. We have $f'(x) = \cos x - x/2$ and it is easily verified that $|f(x)| > 1.31 = M_1$, $x \in [1.93, 1.94]$. Further, using six decimals we have $\sin 1.93375 = 0.934852 \pm 0.510^{-6}$, and $(x/2)^2 = 0.966875)^2 = 0.934847 \pm 0.510^{-6}$. Then from (6.5.3) follows the strict error estimate

$$|x - \alpha| < 6 \cdot 10^{-6}/1.31 < 5.6 \cdot 10^{-6}.$$

6.5.2 Termination Criteria

Suppose that we want to compute an approximation to the root α to a prescribed absolute accuracy. In hand computation, it may be possible to interrupt the iterations on the basis of the error estimate (6.5.3), provided that the necessary derivative is

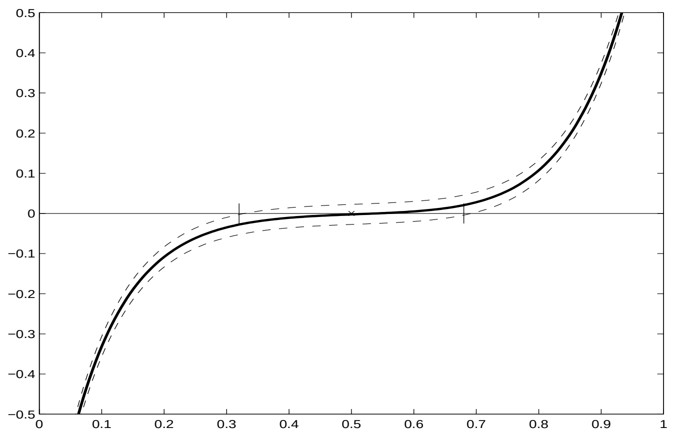


Figure 6.5.1. *An ill-conditioned root.*

easy to estimate. On a computer it is usually better to iterate a few extra times rather than make the effort to use a special formula for error estimation.

In subroutines for solving a nonlinear equation it is common practice to use a termination criterion of the following form. Assuming that $f(a)f(b) \le 0$ the iterations are terminated if

$$|b-a| \le 2u|x_n| + \tau, \tag{6.5.5}$$

where τ is a user specified absolute tolerance and u is the rounding unit (see Section 2.2).

We must also deal with the possibility that the user specified tolerance is too small and cannot be attained. If this is the case, then from some n onwards rounding errors will dominate in the evaluation of $f(x_n)$ and the computed values of $f(x)$ may vary randomly in an interval $(-\delta,\delta)$. If we are using a method like the bisection method, the iterations will continue until the criterion (6.5.5) is satisfied, but this, of course, may *not* mean that the root actually has been determined to this precision!

If we are using Newton's method, quadratic convergence will eventually be lost and the computed corrections h_n will tend to vary randomly in some interval $(-\epsilon,\epsilon)$. This observation suggests the following alternative termination criterion:[6]

Accept the approximation x_n when for the first time the following two conditions are satisfied

$$|x_{n+1}-x_n| \ge |x_n - x_{n-1}|, \qquad |x_n - x_{n-1}| < tol. \tag{6.5.6}$$

Here *tol* is a coarse tolerance, used only to prevent the iterations from being terminated before x_n even has come close to α. When (6.5.6) is satisfied the quantity $|x_{n+1}-x_n|$ is usually a good estimate of $|x_n - \alpha|$. With this termination criterion

[6]This termination criterion has been suggested by Jan Garwick.

the risk of not terminating the iterations in time for ill-conditioned roots is quite small. Note also that for iteration methods of superlinear convergence ultimately converge so fast that the cost of always iterating until the attainable accuracy is obtained may be small, even if the user specified tolerance is much larger than ϵ_α.

6.5.3 Multiple Roots

In the previous discussion we have assumed that the root α to be computed is a simple root, i.e., $f'(\alpha) \neq 0$. We make the following definition:

Definition 6.5.1.

A root α to the equation $f(x) = 0$ is said to have **multiplicity** *q if*

$$0 \neq \lim_{x \to \alpha} |f(x)/(x-\alpha)^q| < \infty. \qquad (6.5.7)$$

Suppose that $f(x)$ is q times continuously differentiable in a neighborhood of a root α of multiplicity q. By (6.5.7) we have $f^{(j)}(\alpha) = 0$, $j < q$ and by Taylor's formula

$$f(x) = \frac{1}{q!}(x-\alpha)^q f^{(q)}(\xi), \quad \xi \in \text{int}(x, \alpha). \qquad (6.5.8)$$

Multiple roots are inherently ill-conditioned. Proceeding as in Section 6.5.1, if $|f(x_n)| \le \delta$, then using (6.5.8) with $x = x_n$, we find that the best corresponding error bound for a root α of multiplicity q if is

$$|x_n - \alpha| \le (\delta\, p!/M_q)^{1/q}, \quad |f^{(q)}(x)| \ge M_q, \quad x \in J. \qquad (6.5.9)$$

Comparing this with (6.2.3)) we see that because of the exponent $1/q$ multiple roots in general are very ill-conditioned. If there are several distinct but pathologically close roots we can expect a similar behavior.

Example 6.5.2.

The equation $f(x) = (x-2)x + 1 = 0$ has a double root $x = 1$. The (exact) value of the function at $x = 1 + \epsilon$ is

$$f(x+\epsilon) = (\epsilon - 1)(1+\epsilon) + 1 = -(1-\epsilon^2) + 1 = \epsilon^2.$$

Now, suppose that we use a floating point arithmetic with eight decimal digits in the mantissa. Then

$$fl(1-\epsilon^2) = 1, \quad |\epsilon| < \frac{1}{2}\sqrt{2}\cdot 10^{-4},$$

and for $0.99992929 \le x \le 1.0000707$, the computed value of $f(x)$ will be zero when $f(x)$ is evaluated using Horner's rule. Hence the root can only be computed with about four correct digits, that is, with a relative error equal to the *square root of the machine precision.*

If $f(x)$ is q times continuously differentiable in a neighborhood of a root α of multiplicity q we have by Taylor's formula (cf. (6.5.8))

$$f'(x) = \frac{1}{(q-1)!}(x-\alpha)^{q-1} f^{(q)}(\xi'), \quad \xi' \in \text{int}(x, \alpha).$$

It follows that if x_n is close to α, then the Newton correction satisfies

$$h_n = \frac{f(x_n)}{f'(x_n)} \approx \frac{1}{q}(x_n - \alpha) = \epsilon_n/q.$$

and for the corresponding errors we have

$$\epsilon_{n+1} = \epsilon_n - \epsilon_n/q = (1 - 1/q)\epsilon_n.$$

This shows that for a root of multiplicity $q > 1$ Newton's method only converges *linearly* with rate $C = 1 - 1/q$. (The same is true of other methods which have quadratic or higher rate of convergence for simple roots.) Note also that when $x_n \to \alpha$ both $f(x_n) \to 0$ and $f'(x_n) \to 0$. Therefore rounding errors may seriously affect the Newton correction when evaluated close to α, and some safeguarding is essential; see Section 6.5.4.

We now consider the case when *the multiplicity q is known a priori.* Then the modified Newton's method

$$x_{n+1} = x_n + q\frac{f(x_n)}{f'(x_n)}, \tag{6.5.10}$$

is easily shown to have quadratic convergence for roots of multiplicity q.

From (6.5.8) it follows that the equation $u(x) = 0$, where

$$u(x) = f(x)/f'(x), \tag{6.5.11}$$

always has *a simple root at $x = \alpha$.* Hence any method applied to this equation will retain its order of convergence independent of the multiplicity of α as a root to $f(x) = 0$. In particular Newton's method applied to the equation (6.5.11) gives

$$x_{n+1} = x_n - \frac{u(x_n)}{u'(x_n)}, \qquad u'(x_n) = 1 - \frac{f''(x_n)}{f'(x_n)}u(x_n). \tag{6.5.12}$$

or

$$x_{n+1} = x_n - u(x_n)\frac{1}{1 - t(x_n)}, \qquad t(x_n) = \frac{f(x_n)f''(x_n)}{(f'(x_n))^2}.$$

which requires the evaluation of the second derivative of $f(x)$. Note that this is similar to Halley's method (6.3.15). One can also use the secant method

$$x_{n+1} = x_n - u(x_n)\frac{x_n - x_{n-1}}{u(x_n) - u(x_{n-1})}.$$

The transformation (6.5.11) is most useful if an *analytical simplification* can be done such that $u(x)$ can be evaluated accurately also in a neighborhood of α.

Review Questions

1. What two quantities determines the attainable accuracy of a simple root α to the equation $f(x) = 0$. Give an example of an ill-conditioned root.

2. Discuss the choice of termination criteria for iterative methods.

3. (a) Assume that f is continuously differentiable in a neighborhood of a double root α of the equation $f(x) = 0$. Describe how the equation can be converted to one with a simple root α.

(b) Discuss the case when $f(x) = 0$ has two distinct roots which *nearly* coincide.

Computer Exercises

1. Determine the multiple root $\alpha = 1$ of the equation $p(x) = (1 - x)^5 = 0$, when the function is evaluated using Horner's scheme, i.e.,

$$p(x) = (((((x - 5)x + 10)x - 10)x + 5)x - 1 = 0.$$

(a) Use bisection (cf. Algorithm 6.1.1) with initial interval $(0.9, 1.1)$ and tolerance $\delta = 10^{-8}$. What final accuracy is achieved?

(b) Use Newton's method, starting from $x_0 = 1.1$ and evaluating $p'(x)$ using Horner's scheme. Terminate the iterations when for the first time $|x_{n+1} - 1| > |x_n - 1|$. How many iterations are performed before termination? Repeat with a couple of other starting values!

(c) Same as (b), but perform one step of the modified Newton's method (6.5.10) with $x_0 = 1.1$ and $q = 5$. How do you explain the achieved accuracy which is much better than predicted by (6.5.9)?

6.6 Zeros of Polynomials

6.6.1 Introduction

Consider the algebraic polynomial equation of degree n

$$p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_n = 0, \quad a_0 \neq 0. \tag{6.6.1}$$

Such equations are special enough that information about the location of the roots can be obtained and many methods have been devised specifically tailored to them.

The fundamental theorem of algebra states for any algebraic equation $p(z) = 0$ of degree $n > 0$, there exists at least one complex number z_1 such that $p(z_1) = 0$. Hence we can write $p(z) = (z - z_1)p_1(z)$, where $p_1(z)$ is a polynomial of degree $n - 1$. Applying this reasoning recursively it follows that, counting multiplicities, the equation (6.6.1) has exactly n (real or complex) roots $\alpha_1, \alpha_2, \dots, \alpha_n$, and

$$p(z) = a_0(z - \alpha_1)(z - \alpha_2) \cdots (z - \alpha_n). \tag{6.6.2}$$

By this representation it follows that if the coefficients $a_0, a_1, \dots, a_n$ are real, then eventual complex roots must occur in conjugate pairs.

Solution of algebraic equations is a topic that has been extensively studied. The long history of investigations started with algebraic expressions for the zeros of equations of degree less than five. The case of roots of a quadratic is elementary. Cardano published formulas for the roots of a cubic equation in the 16th century and formulas for the roots when $n = 4$ are also known. In 1826 Abel gave a proof that it is not possible to find algebraic expressions for the roots when $n > 4$. However the existing formulas for $n \leq 4$ are not in general suitable for numerical evaluation of the roots. As discussed in Section 2.3.2, already in the quadratic case care must be taken to ensure numerical stability (see also Problem 2)!

Although a problem may appear in the form (6.6.1) it may be that $p(z)$ is the characteristic polynomial of some matrix A and hence the roots to be determined are the eigenvalues of A. It is very important that such equations, which are eigenvalue problems in disguise, are solved by a method for solving eigenvalue problems; consult Chapter 9 and references therein! This is because, even when the eigenvalues are well determined by the elements of the matrix A, the roots of $p(z) = 0$ can be *extremely sensitive to perturbations in the coefficients* $a_0, \dots, a_n$ *of* $p(z)$. A striking example is given in Sec. 6.6.5. Computing the characteristic polynomial is therefore not, as is sometimes thought, a simplification of the eigenvalue problem if a numerical method has to be used!

On the other hand algorithms for solving the eigenvalue problem are now highly developed and can be used also for solving polynomial equations. The characteristic polynomial of the matrix

$$K_n = \begin{pmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

equals

$$p(\lambda) = \det(K - \lambda I) = (-1)^n(\lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n).$$

Hence the roots of $p(\lambda) = 0$ are the eigenvalues of K_n, which is called the **companion matrix** of $p(\lambda)$. (Sometimes the companion matrix is defined slightly differently, e.g., with the coefficients of the polynomial in the last row or in the last column.

6.6.2 Synthetic Division

Function values and derivatives of a polynomial $p(z)$ at a (real or complex) point z_k can conveniently be computed by repeated synthetic division, see Section 1.4.2. If we define the sequence $\{b_i\}_{i=0}^n$ by the recursion

$$b_0 = a_0, \qquad b_i = b_{i-1}z_k + a_i, \quad i = 1 : n, \tag{6.6.3}$$

then it holds that

$$\begin{aligned} p(z) &= (z - z_k)q(z) + b_n, \\ q(z) &= b_0 z^{n-1} + b_1 z^{n-2} + \ldots + b_{n-1}. \end{aligned} \tag{6.6.4}$$

Hence $q(z)$ is the quotient polynomial when dividing out the linear factor $z - z_k$ and we immediately have $p(z_k) = b_n$. Differentiating (6.6.4) we get

$$p'(z) = (z - z_k)q' + q(z),$$

and taking $z = z_k$ we obtain $p'(z_k) = q(z_k) = c_{n-1}$, where

$$c_0 = b_0, \qquad c_i = c_{i-1}z_k + b_i, \quad i = 1 : n - 1.$$

Higher derivatives can be computed in the same fashion. Differentiating once more we have

$$p''(z) = (z - z_k)q''(z) + 2q'(z),$$

and so $p''(z_k) = 2q'(z_k) = 2d_{n-2}$, where

$$d_0 = c_0, \qquad d_i = d_{i-1}z_k + c_i, \quad i = 1 : n - 2.$$

To compute $p^{(i)}(z_k)$ using these formulas requires $n-i$ additions and multiplications.

In the important special case where all the coefficients $a_0, a_1, \dots, a_n$ are real, the above formulas are somewhat inefficient, and one can save operations by performing synthetic division with the quadratic factor

$$(z - z_k)(z - \bar{z}_k) = z^2 - 2zRe(z_k) + |z_k|^2,$$

which has real coefficients (see Problem 1 at the end of this section).

6.6.3 Laguerre's Iteration Method

If we have sufficiently good initial approximations to a (real or complex) zero of $p(z)$, this can be computed by Newton's method. If $p(z)$ has real coefficients, then $p(z)$ and $p'(z)$ are real for real values of z. This means that Newton's method cannot converge to a *complex* root from a *real* initial approximation. The same holds for the secant method and its variants. The Muller–Traub method (see Section 6.4.5), which can converge to complex roots from real approximations and also requires only one evaluation of $p(z)$ per step, has therefore become popular. In many applications no good a priori information about the location of the roots is available. Then a method with good **global** convergence properties has to be used. We also have to avoid converging to the same root more than once. In the following we discuss these and other topics in more detail.

Laguerre's method is a method with very good global convergence properties for polynomial equations, and with *cubic* convergence for simple roots (real or complex). In this method the polynomial $p(z)$ of degree n is approximated in the neighborhood of the point z_k by a special polynomial of the form

$$r(z) = a(z - w_1)(z - w_2)^{n-1},$$

where the parameters a, w_1 and w_2 are determined so that

$$p(z_k) = r(z_k), \qquad p'(z_k) = r'(z_k), \qquad p''(z_k) = r''(z_k). \tag{6.6.5}$$

If z_k is an approximation to a simple zero, α then the simple zero w_1 of $r(z)$ is taken as the new approximation z_{k+1} of α.

In order to derive Laguerre's method we note that the logarithmic derivative of $p(z) = (z-\alpha_1)\cdots(z-\alpha_n)$ is

$$S_1(z) = \frac{p'(z)}{p(z)} = \sum_{i=1}^{n} \frac{1}{z-\alpha_i}.$$

Taking the derivative of this expression we obtain

$$-\frac{dS_1(z)}{dz} = S_2(z) = \left(\frac{p'(z)}{p(z)}\right)^2 - \frac{p''(z)}{p(z)} = \sum_{i=1}^{n} \frac{1}{(z-\alpha_i)^2}.$$

Using (6.6.5) to determine the parameters of the approximating polynomial r(z) we obtain the equations

$$S_1(z_k) = \frac{1}{z_k - w_1} + \frac{(n-1)}{z_k - w_2}, \qquad S_2(z_k) = \frac{1}{(z_k-w_1)^2} + \frac{(n-1)}{(z_k-w_2)^2}.$$

Eliminating $z_k - w_2$ gives a quadratic equation for the correction $z_k - w_1 = z_k - z_{k+1}$. After some algebra we obtain (check this!)

$$z_{k+1} = z_k - \frac{np(z_k)}{p'(z_k) \pm \sqrt{H(z_k)}}, \tag{6.6.6}$$

where

$$H(z_k) = (n-1)^2[p'(z_k)]^2 - n(n-1)p(z_k)p''(z_k).$$

The sign in the denominator in (6.6.6) should be chosen so that the magnitude of the correction $|z_{k+1} - z_k|$ becomes as small as possible.

For polynomial equations with only real roots, Laguerre's method is globally convergent, i.e., it converges for *every* choice of real initial estimate z_0. Suppose the roots are ordered such that $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_n$. If $z_0 \in (\alpha_{j-1}, \alpha_j)$, $j = 2 : n$, then Laguerre's method converges to one of the roots α_{j-1}, α_j; if $z_0 < \alpha_1$ or $z_0 > \alpha_n$ then convergence is to α_1 or α_n respectively.

For polynomial equations with complex roots, Laguerre's method no longer converges for every choice of initial estimate. However, experience has shown that the global convergence properties are good also in this case. In particular, if we take $z_0 = 0$, then Laguerre's method will usually converge to the root of smallest modulus. We finally remark that for multiple roots convergence of Laguerre's method is only linear.

Consider the polynomial equation (6.6.1) and assume that $a_n \ne 0$ so that $\alpha = 0$ is not a root. Now suppose that $a_{n-2}a_{n-1} \ne 0$, and take $z_0 = 0$ in Laguerre's method. A simple calculation gives

$$z_1 = \frac{-na_n}{a_{n-1} \pm \sqrt{H(z_0)}}, \quad H(z_0) = (n-1)^2 a_{n-1}^2 - 2n(n-1)a_n a_{n-2}, \tag{6.6.7}$$

where the sign is now chosen so the the $|z_1|$ is minimized. In particular, for $n = 2$, $H(z_0)$ is the discriminant of $p(z)$ and z_1 is the root of smallest modulus.

Example 6.6.1.

If there are complex roots, then there may be several distinct roots of smallest modulus. For example, the equation

$$p(z) = z^3 - 2z^2 + z - 2$$

has roots $\pm i$ and 2. Using the above formula (6.6.7) for z_1 with $n = 3$, we get

$$z_1 = \frac{6}{1 \pm 2i\sqrt{11}} = \frac{2}{15} \pm i\frac{4\sqrt{11}}{15} = 0.06666666667 \pm 0.88443327743i.$$

Continuing the iterations with Newton's method we get convergence to one of the two roots $\pm i$,

$$z_2 = -0.00849761051 + 1.01435422762i,\ z_3 = -0.00011503062 + 1.00018804502i$$
$$z_4 = -0.00000002143 + 1.00000003279i,\ z_5 = -0.00000000000 + 1.00000000000i$$

6.6.4 Deflation and Zero Suppression

Suppose we have found a root α to the equation $p(z) = 0$. Then taking $z_k = \alpha$ in (6.6.3)–(6.6.4) we have $b_n = p(\alpha) = 0$ and the remaining roots of $p(z)$ are also roots of the polynomial equation

$$q(z) = \frac{p(z)}{z - \alpha} = 0$$

Hence we can continue the iterations with the quotient polynomial $q(z)$ of degree $n - 1$. This process is called **deflation** and can be repeated; as soon as a root has been found it is factored out. In this fashion, all roots are eventually found. Since we work with polynomials of lower and lower degree, deflation saves arithmetic operations. More important is that it prevents the iterations to converge to the same simple root more than once.

So far we have ignored that roots which are factored out are only known with finite accuracy. Also rounding errors occur in the computation of the coefficients of the quotient polynomial $q(x)$. Clearly there is a risk that both these types of errors can have the effect that the zeros of the successive quotient polynomials deviate more and more from those of $p(z)$. Indeed, deflation is not unconditionally a stable numerical process. A closer analysis performed by Wilkinson [13, 1963] shows that if the coefficients of the quotient polynomials are computed by the recursion (6.6.3), then errors resulting from deflation are negligible provided that:

1. the roots are determined in order of *increasing* magnitude;

2. every root is determined to its limiting accuracy.

Note that if the coefficients of the polynomial $p(z)$ are reversed, then we obtain a new polynomial, whose zeros are the reciprocals of the zeros of $p(z)$. Hence if the above procedure is applied to this new polynomial we obtain the zeros of $p(z)$ in order of *decreasing* magnitude.

With Laguerre's method it is quite probable that we get convergence to the root of smallest magnitude from the initial value $z_0 = 0$. However, this cannot be guaranteed and one often proceeds in two steps. First, all n roots are determined using deflation in the process. Next, each root from the first step is refined by doing one or several iterations using the *original* polynomial $p(z)$.

Deflation can be avoided by using a **zero suppression** technique suggested by Maehly [1954]. He notes that the derivative of the reduced polynomial $q(z) = p(z)/(z - \xi_1)$ can be expressed as

$$q'(z) = \frac{p'(z)}{z - \xi_1} - \frac{p(z)}{(z - \xi_1)^2}.$$

More generally, assume that we have determined approximations $\xi_1, \dots, \xi_j$ to j roots of $p(z) = 0$. Then the the first derivative of the reduced polynomial $q_j(z) = p(z)/[(z - \xi_1) \cdots (z - \xi_j)]$ can be expressed as

$$q_j'(z) = \frac{p'(z)}{(z - \xi_1) \cdots (z - \xi_j)} - \frac{p(z)}{(z - \xi_1) \cdots (z - \xi_j)} \sum_{i=1}^{j} \frac{1}{z - \xi_i}.$$

Hence Newton's method applied to $q_j(z)$ can be written

$$z_{k+1} = z_k - \frac{p(z_k)}{p'(z_k) - \sum_{i=1}^{j} p(z_k)/(z_k - \xi_i)}, \tag{6.6.8}$$

which is the **Newton–Maehly method**. This iteration has the advantage that it is not sensitive to the accuracy in the approximations to the previous roots $\xi_1, \dots, \xi_j$. Indeed, the iteration (6.6.8) is locally quadratically convergent to simple zeros of $p(z)$ for *arbitrary* values of $\xi_1, \dots, \xi_j$.

6.6.5 Ill-Conditioned Polynomial Roots

We have previously noted that multiple roots in general are ill-conditioned, i.e. sensitive to perturbations from, for example, round-off errors in the evaluation of function values. Thus, it seems natural that if $p(z) = 0$ has some roots which are very close to each other (*nearly* multiple roots) then these roots should be very sensitive to perturbations. That roots which, at first sight, appear to be well separated can be extraordinary sensitive to small relative perturbations in the coefficients of $p(z)$ is more surprising but nonetheless true! In the following we give a famous example, due to Wilkinson [13, 1984]. This paper contains an extensive discussion of numerical problems in determining roots of polynomial equations.

Example 6.6.2.

Consider the Wilkinson polynomial

$$p(z) = (z-1)(z-2)\cdots(z-20) = z^{20} - 210z^{19} + \ldots + 20!,$$

with zeros $1, 2, \ldots, 20$. Let $\bar{p}(z)$ be the polynomial which is obtained when the coefficient $a_1 = 210$ in $p(z)$ is replaced by

$$-(210 + 2^{-23}) = -210.000000119\ldots,$$

while the rest of the coefficients remain unchanged. Even though the relative perturbation in a_1 is of order 10^{-10}, many of the zeros of the perturbed polynomial $\bar{p}(z)$ deviate greatly from those of $p(z)$. In fact, correct to nine decimal places, the perturbed zeroes are

1.000000000	$10.095266145 \pm 0.643500904i$
2.000000000	
3.000000000	$11.793633881 \pm 1.652329728i$
4.000000000	
4.999999928	$13.992358137 \pm 2.518830070i$
6.000006944	
6.999697234	$16.730737466 \pm 2.812624894i$
8.007267603	
8.917250249	$19.502439400 \pm 1.940330347i$
20.846908101	

For example, the two zeros $16, 17$ have not only changed substantially, but have become a complex pair. It should be emphasized that this behavior is quite typical of polynomials with real coefficients and real roots. Indeed, many polynomials which arise in practice behave much worse than this.

If we assume that the coefficients a_i of a polynomial are given with full machine accuracy, then the error in computed values of $p(x)$ (for real x) can be as large as

$$\delta = 1.06u \sum |(2i+1)a_{n-i}x^i|,$$

see Section 2.4. Hence by (6.5.4) the attainable accuracy of a zero α is equal to

$$\epsilon_\alpha = \frac{\delta}{|p'(\alpha)|} = \frac{\sum |(2i+1)a_{n-i}\alpha^i|}{|p'(\alpha)|}.$$

In particular for the root $\alpha = 14$ in the above example we get $\epsilon_\alpha = 1.89 \cdot 10^{16}$. However, the changes in this example are so large that this linearized perturbation theory does not apply!

If the coefficients are known (and stored) exactly, then by using multiple-precision arithmetic the accuracy in the zeros can be increased. It is generally true that the solution of polynomial equations of high degree requires the use of multiple precision floating-point arithmetic in order to achieve high accuracy. Therefore, if the coefficients of $p(z)$ are not given as the original data, it may be better to avoid computing them. An important example of this is the determination of eigenvalues of matrices; the eigenvalues are zeros of the characteristic equation, $p(\lambda) = \det(A - \lambda I) = 0$. Here the original data are the elements of the matrix A. Numerical values of $p(\lambda)$ can in general be evaluated much more accurately directly from the matrix elements, see Chapter 9. Another example is given below.

Example 6.6.3.

The largest positive root of the equation

$$p(x) = (x+2)(x^2-1)^6 - 3 \cdot 10^{-6} \cdot x^{11} = 0$$

is to be computed. Here $p(z)$ is a polynomial of degree 13. If the coefficients are computed using decimal floating point arithmetic with seven digits, then the coefficient of x^{11} which is $(12 - 3 \cdot 10^{-6})$ will be rounded to 12.00000. Thus the machine will treat the equation $(x+2)(x^2-1)^6 = 0$, whose exact positive root is 1.

This is a poor result. One can get the root $\alpha = 1.053416973823$ to full accuracy for example by writing the equation in the form

$$x = \phi(x), \qquad \phi(x) = 1 + \frac{0.1}{x+1}\left(\frac{3x^{11}}{x+2}\right)^{1/6},$$

and solving this by the iteration $x_0 = 1$, $x_{k+1} = \phi(x_k)$. Hence the relative error in the previous result is greater then 5%.

6.6.6 Simultaneous Determination of Roots

We now consider iterative methods that, under appropriate separation assumptions. allows for the *simultaneous* determination of all the roots of a polynomial equation. Suppose that the numbers $\xi_i^{(k)}$ are a set of n distinct approximations to the zeros α_i, $i = 1 : n$ of $p(z)$. In **Weierstrass' method** one computes a new set of approximations using the iteration formula

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)}) \Big/ \Big[a_0 \prod_{\substack{j=1 \\ j \neq i}}^{n} (\xi_i^{(k)} - \xi_j^{(k)})\Big], \quad i = 1 : n. \qquad (6.6.9)$$

With $q(z) = (z - \xi_1^{(k)})(z - \xi_2^{(k)}) \cdots (z - \xi_n^{(k)})$ the formula may also be written

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)})/q'(\xi_i^{(k)}),$$

which shows that to first approximation the method is identical to Newton's method. This relation can be used to prove that the asymptotic order of convergence of the Weierstrass method is equal to 2 for simple zeros. For multiple zeros the method will only converge linearly, as does Newton's method.

Note that it is possible to use the new approximations of the roots in (6.6.9) as they become available, i.e.

$$\xi_i^{(k+1)} = \xi_i^{(k)} - p(\xi_i^{(k)}) \Big/ \Big[a_0 \prod_{j<i} (\xi_i^{(k)} - \xi_j^{(k+1)}) \prod_{j>i} (\xi_i^{(k)} - \xi_j^{(k)})\Big], \quad i = 1 : n.$$

This *serial* version of the Weierstrass method can be shown to have an order of convergence which lies between 2 and 3.

If no a priori information about the roots is available then the initial approximations $\xi_i^{(0)}$ can be chosen equidistantly on a circle $|z| = \rho$, centered at the origin, which encloses all the zeros of $p(z)$. Such a circle can be found using the following result:

Theorem 6.6.1. *All the roots $\alpha_1, \dots, \alpha_n$ of the polynomial $p(z) = a_0 z^n + \cdots + a_n, a_0 \neq 0$ lie in the disk $|z| \leq \rho$, where*

$$\rho = \min \left(n \frac{|a_n|}{|a_{n-1}|}, \sqrt[n]{\frac{|a_n|}{|a_0|}} \right).$$

Review Questions

1. Discuss the ill-conditioning of roots of polynomial equations. What famous polynomial did J. H. Wilkinson use as an example?
2. Suppose that all roots of a polynomial equation are to be determined. Describe two methods which avoids the problem of repeatedly converging to roots already found.

Problems

1. Consider a polynomial with real coefficients

$$p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_n, \quad a_i \neq 0, \quad i = 0 : n.$$

(a) Count the number of (real) additions and multiplications needed to compute a value $p(z_0)$ by synthetic division of $p(z)$ by $(z - z_0)$, when z_0 is a real and complex number, respectively.

(b) For a complex number $z_0 = x_0 + iy_0$, $p(z_0)$ can also be computed by performing the synthetic division of p(z) with the real quadratic factor

$$d(z) = (z - z_0)(z - \bar{z}_0) = z^2 - 2x_0 z + (x_0^2 + y_0^2).$$

Derive a recursion for computing the quotient polynomial $q(z)$ and $p(z_0) = b_{n-1} z_0 + b_n$, where

$$q(z) = b_0 z^{n-2} + b_1 z^{n-3} + \ldots + b_{n-2},$$
$$p(z) = q(z) d(z) + b_{n-1} z + b_n.$$

Count the number of real additions and multiplications needed to compute $p(z_0)$ and also show how to compute $p'(z_0)$.

2. (a) Using Cardano's formula the real root α to the equation $x^3 = x + 4$ can be written in the form

$$\alpha = \sqrt[3]{2 + \frac{1}{9}\sqrt{321}} + \sqrt[3]{2 - \frac{1}{9}\sqrt{321}}.$$

Use this expression to compute α and discuss the loss of accuracy due to cancellation.

(b) Compute α to the same accuracy by Newton's method starting from $x_0 = 2$.

3. In Graeffes root-squaring method one separates even and odd powers of the polynomial $p(z)$ and squares the equation as follows

$$(a_0 z^n + a_2 z^{n-2} + a_4 z^{n-4} + \cdots)^2 = (a_1 z^{n-1} + a_3 z^{n-3} + a_5 z^{n-5} + \cdots)^2.$$

Putting $u = z^2$ the resulting equation becomes

$$q(u) = p(-z)p(z) = b_0 u^n + b_1 u^{n-1} + \cdots + b_n = 0.$$

This has roots equal to the squares of the roots of the original equation.

(a) Show that the coefficients b_k of $q(u)$ can be computed from

$$b_0 = a_0^2, \qquad (-1)^k b_k = a_k^2 + \sum_{j=1}^{k} (-1)^j 2 a_{k-j} a_{k+j}, \quad k = 1, 2, \ldots, n.$$

(b) After squaring m times we obtain (after normalizing $A_0 = 1$) an equation in $u = z^{2m}$

$$u^n + A_1 u^{n-1} + A_2 u^{n-2} + \cdots + A_n = 0,$$

with roots $\beta_k = \alpha_k^{2m}$. Assume that the roots α_k of the original equation $p(z) = 0$ are real and distinct. Use the relations between coefficients and roots of an algebraic equation to show that for m large enough we have

$$\beta_1 \approx -A_1, \qquad \beta_2 \approx -A_2/A_1, \qquad \beta_3 \approx -A_3/A_2, \ldots.$$

(c) Square the polynomial $z^3 - 8z^2 + 17z - 10$ $m = 3$ times, and then use the relations in (b) to compute approximations to its three real roots.

4. Consider the iteration $z_{n+1} = z_n^2 + c$, where $c = p + iq$ is a fixed complex number. For a given z_0 the sequence of iterates $z_n = x_n + iy_n$, $n = 0, 1, 2, \ldots$ may either converge to one of the two roots of the quadratic equation $z^2 - z + c = 0$ or diverge to infinity. Consider z_0 chosen, e.g., in the unit squares of the complex plane. The boundary separating the region of convergence from other points in the plane is a very complex **fractal curve** know as the **Julia set**. The **Mandelbrot set** is obtained by fixing $z_0 = 0$ and sweeping over values of c in a region of the complex plane.

(a) Picture the Julia set as follows. Set $c = 0.27334 + 0.000742i$. Sweep over points of z_0 in the region $-1 \le \Re z_0 \le 1$, $-1.3 \le \Im z_0 \le 1.3$. If $|z_N| < R$, for $N = 100$ and $R = 10$ color the point z_0 black. otherwise color the point from hot (red) to cool (blue) according to how fast the iteration is diverging, i.e. according to how fast the inequality $|z_n| > R$ becomes satisfied.

(b) Picture the Mandelbrot set in a similar way. Sweep over values of c in the region $-2.25 \le \Re c \le 0.75$, $-1.5 \le \Im c \le 1.5$.

6.7 Minimizing a Scalar Function

We consider here the problem of finding an approximation to the minimum or maximum of a real-valued function f of one variable belonging to some interval $[a, b]$. We will give methods which determine an approximate local minimum of f by evaluating f and/or f' at a small number of points.

One-dimensional minimization is an important subproblem in methods for optimization and for solving systems of nonlinear equation. For example, if x_k is the current approximation and d_k a search direction, the next approximation in such methods is often found by minimizing a function

$$f(\lambda) = \phi(x_k + \lambda d_k),$$

where λ is a steplength.

If f is differentiable in $I = [a, b]$, a **necessary condition** for f to have a local minimum at an interior point of $x^* \in I$ is

$$f'(x^*) = 0.$$

It is also possible that the minimum is at a or b. This is the case if f' does not change sign on I. If this is checked for separately, then it is possible to reduce the problem to a zero-finding problem for f' by some of the methods described earlier in this chapter. Since f' also vanishes at a point of maximum and inflection, it is necessary to check if the point found is a minimum, and continue if it is not. If it is difficult or impossible to compute f' directly, then f' can be determined by numerical differentiation. However, methods described below, which use function values of f more directly are usually to be preferred.

In any minimization method, which only uses computed values of f, there is a fundamental limitation in the accuracy of the computed minimum. If f is twice differentiable in a neighborhood of a minimum point x^* then

$$f(x^* + \delta) \approx f(x^*) + \tfrac{1}{2}\delta^2 f''(x^*).$$

Hence, unless $\frac{1}{2}\delta^2 > u|f(x^*)|/|f''(x^*)|$, where u is the unit roundoff, the computed value will not be different from $f(x^*)$. This means that there is no difference in the floating point representation of $f(x^* + \delta)$ unless δ is larger than a constant times the *square root* $\sqrt{u}$. Rounding and other errors in the computed function values $\bar{f}(x) = \mathrm{fl}\,(f(x))$ may also contribute to the uncertainty. We can in general not expect the relative error to be less than $\sqrt{u}$ unless we can also use values of f' or the function has some special form; see Example **??**.

Most algorithms for minimizing a nonlinear function of one (or more) variables find, at best, a local minimum. For a function with several local minima, there is no guarantee that the global (lowest) minimum will be found. One remedy is to try several different starting points and hope that the lowest local minimum found is also the global minimum. This approach is neither efficient or safe. In practice we have to be content with algorithms which nearly always give correct results in most practical applications.

We introduce a condition which ensures that a function f has a unique global minimum x^* in an interval $[a, b]$.

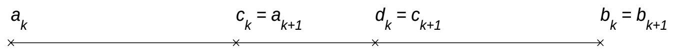


Figure 6.7.1. *One step of interval reduction, $f(c_k) \geq f(d_k)$.*

Definition 6.7.1.

The function $f(x)$ is said to be **unimodal** *in $[a, b]$ if there exists a unique $x^* \in [a, b]$ such that, given any $c, d \in [a, b]$ for which $c < d$*

$$d < x^* \Rightarrow f(c) > f(d); \qquad c > x^* \Rightarrow f(c) < f(d).$$

Using this definition we need not assume that the function f is smooth or even continuous.

We now consider the **interval reduction method**, which can be thought of as being analogue of the bisection method. Assume that f is unimodal in $[a, b]$. It is possible to find a reduced interval on which f is unimodal by comparing values of $f(x)$ at two interior points c and d such that $c < d$. It holds

$$x^* \in \begin{cases} [c, b], & \text{if } f(c) \geq f(d); \\ [a, d], & \text{if } f(c) < f(d). \end{cases}$$

(If $f(c) = f(d)$ we could take the smaller interval $[c, d]$, but we ignore this possibility.) Hence by performing two function evaluations we can enclose x^* in an interval of length at most equal to $\max(b - c, d - a)$. To minimize this length one should take c and d so that $b - c = d - a$. Hence $c + d = a + b$, and we can write

$$c = a + t(b - a), \qquad d = b - t(b - a), \quad 0 < t < 1/2.$$

Then $d - a = b - c = (1 - t)(b - a)$, and by choosing $t \approx 1/2$ we can almost reduce the length of the interval by a factor 1/2. However, $d - c = (1 - 2t)(b - a)$ must not be too small for the available precision in evaluating $f(x)$.

If we only consider one step the above choice would be optimal. Note that this step requires **two** function evaluations. A clever way to save function evaluations is to arrange it so that if $[c, b]$ is the new interval then d can be used as one of the points in the next step; similarly if $[a, d]$ is the new interval then c. Suppose this can be achieved with a fixed value of t. Because of symmetry we need only consider the the first case. Then t must satisfy the following relation (cf. above and Figure 6.7.1)

$$d - c = (1 - 2t)(b - a) = (1 - t)t(b - a).$$

Hence t should equal the root in the interval $(0, 1/2)$ of the quadratic equation $1 - 3t + t^2 = 0$, which is $t = (3 - \sqrt{5})/2$. With this choice the length of the interval will be reduced by the factor

$$1 - t = 2/(\sqrt{5} + 1) = 0.6180...$$

at each step, which is the the **golden section** ratio. For example, $n = 20$ gives a reduction of the interval with a factor about 10^{-4}.

If we know in advance that we are allowed to evaluate $f(x)$ at a fixed number of of points $x_n, x_{n-1}, \dots, x_2, x_1$, then one can show (see [2, Section 8.1]) that the optimal choice of points are related to the **Fibonacci sequence** τ_k, where

$$\tau_0 = \tau_1 = 1, \quad \tau_{k+1} = \tau_k + \tau_{k-1}, \quad k > 1. \tag{6.7.1}$$

At successive steps the length of the interval will be reduced by the factors

$$\tau_{n-1}/\tau_n, \tau_{n-1}/\tau_n, \dots, \tau_1/\tau_2(1 + \epsilon),$$

where $\epsilon \approx \sqrt{u}$. Hence the length of the final interval will be $(1 + \epsilon)(b - a)/\tau_n$. For $n \to \infty$ it holds that

$$t_{k+1}/t_k \to 2/(1 + \sqrt{5}) = 0.618...,$$

and we recover the golden section ratio.

Algorithm 6.7.1 *Golden Section Search.*

Let f be a given continuous function and $I = [a, b]$ an interval. The following algorithm computes an approximation $m \in I$ to a local minimum of $f(x)$, with an error less than a specified tolerance *tau*.

```
$x_{\min} = \text{goldsec}(f, a, b, \delta);$
$t = 2/(3 + \sqrt{5});$
$c = a + t \cdot (b - a);$
$d = b - t \cdot (b - a);$
$fc = f(c); \quad fd = f(d);$
while $(d - c) > \delta \cdot \max(|c|, |d|)$
   if $fc \geq fd$
$\%$Keep right endpoint $b$
      $a = c; \quad c = d;$
      $d = b - t \cdot (b - a);$
      $fc = fd; \quad fd = f(d);$
   else
$\%$Keep left endpoint $a$
      $b = d; \quad d = c;$
```

```
            $c = a + t \cdot (b - a)$;
            $fd = fc$;   $fc = f(c)$;
        end;
    end;
    $x_{\min} = (c + d)/2$;
```

For smooth functions f more efficient methods can be devised where computed function values of f are better utilized. For example, f can be interpolated by a polynomial or rational function, the minimum of which is easy to determine. Since a linear function in general has no minimum the simplest choice is to use a second degree polynomial. In the following algorithm, due to Powell, one seeks $x_k \in [a, b]$ such that

$$f(x_k) \le f(a) \quad \text{and} \quad f(x_k) \le \min_{x\in[a,b]} f(x) + \delta,$$

where δ is a tolerance. In each step we have three points x, y, z, which determine a second degree polynomial $q(x)$. We start with $x = a$, $y = c$, and

$$z = \begin{cases} -c, & \text{if } f(c) > f(a); \\ 2c, & \text{if } f(c) \le f(a). \end{cases}$$

Then the critical point to $q(x)$ is

$$x' = \frac{1}{2}\Big((x + y)(1 - \theta) + (y + z)\theta\Big), \quad \theta = \frac{f[x, y]}{f[x, y] - f[y, z]}.$$

Suppose first that $f[x, y, z] > 0$. Then x' is a minimum point of $q(x)$, and if

$$|f(x') - \min\{f(x), f(y), f(z)\}| \le \delta \text{ or } |f(x') - q(x')| \le \delta,$$

stop with $x^* = x'$. Else, replace x, y, or z with x' and repeat. Usually the point with the largest function value is discarded.

Often a combination of interpolation and golden section search is used; see Brent [1, 1973, Ch. 5]. Note the analogy with robust methods for solving a nonlinear equation, where a combination of inverse interpolation and bisection can be used, see Section 6.4.3. If the first derivative $f'(x)$ is known, then one can use cubic interpolation based on $f(x)$, $f'(x)$, $f(y)$, $f'(y)$. From Taylor's formula we have

$$\hat{f}(x) = a + b(x - x_n) + c(x - x_n)^2/2, \quad a = \hat{f}_n, \quad b = \hat{f}'_n, \quad c = \hat{f}''_n.$$

If $c \ne 0$, then $\hat{f}$ has a stationary point at x^* where $b + c(x - x_n) = 0$, or

$$\min f(x^*) \approx a - b^2/(2c), \quad x^* \approx x_n - b/c.$$

The Matlab function “fmin” is based on the Fortran implementation FMIN of Brent's algorithm given in Forsythe, Malcolm, and Moler [2, pp.184–187].

Review Questions

1. Suppose the twice differentiable function $f(x)$ has a local minimum at a point x^*. What approximate limiting accuracy can you expect in a method for computing x^* which uses only function values?
2. What property should the function $f(x)$ have to be unimodal on the interval $[a, b]$?
3. How many steps in needed in golden section search to reduce an initial interval $[a, b]$ by a factor of 10^{-6}?

Problems

1. Use the algorithm goldsec to find the minimum of the quadratic function $f(x) = (x - 1/2)^2$ starting from $a = 0.25$, $b = 1$. Plot the successive inclusion intervals.

Computer Exercises

1. Modify the algorithm goldsec so that it performs a given number of n steps where the points of evaluation of f are based on the Fibonacci sequence (6.7.1). What is the gain in efficiency compared to *goldsec* for $n = 10$ and $n = 100$, respectively?

Notes and References

Many of the methods for solving a nonlinear equation date back a long time. The regula falsi originated in old Indian texts and was used in medieval Arabic mathematics. It got its name from the Italian mathematician Leonardi Pisano in the 13th century.

Also the secant method predates Newton's method. The current form of Newton's method did not originate with Newton, but is more related to an algorithm published by Raphson about 1690. This is why the method often is called the Newton–Raphson method. The first to give a description of Newton's method using derivatives seems to have been Thomas Simpson in 1740. An interesting historical account of Newton's method is given in Ypma [14]. The third order method of Halley [3] was published more than 300 years ago! For some recent generalizations see Kalantari

One of the best algorithms to combine bisection and interpolation was developed by van Wijngaarden and Dekker at Mathematical Center in Amsterdam in the 1960s; see [?]. It was taken up and improved by Brent [1], see also [2], Section 7.2.

Wilkinson [13] received the Chauvenet Prize of the Mathematical Association of America for his exposition of the ill-conditioning of polynomial zeros in [13].

A more detailed discussion of measures for the asymptotic speed of convergence is found in Ortega and Rheinboldt [7, Chapter 9].

Several comprehensive monographs dealing with methods for solving scalar nonlinear equations are available. Traub [12, 1964] gives an exhaustive enumeration of iteration methods with and without memory, with their order of convergence and efficiency index. Householder [5, 1970] contains much material on classical results and on algebraic equations as does Ostrowski [8, 1973].

The literature on methods for solving algebraic equations is vast. A good survey is given by Sendov et al. in [11].

[1] R. P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[2] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Method for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[3] E. Halley. A new and general method of finding the root equations. Philos. Trans. Roy. Soc. London, 18 (1694).

[4] P. Henrici. *Applied and Computational Complex Analysis*, Vol. 1. Wiley Classics Library, New York, 1996.

[5] A. S. Householder. *The Numerical Treatment of a Single Nonlinear Equation.* McGraw-Hill, New York, 1970.

[6] B. Kalantari, I. Kalantari, and R. Zaare-Nahandi, *A basic family of iteration functions for polynomial root finding and its characterizations*, J. Comput. Appl. Math., 80 (1997), pp. 209–226.

[7] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[8] A. M. Ostrowski. *Solution of Equations in Euclidian and Banach Spaces.* Academic Press, New York, third edition, 1973.

[9] P. Rabinowitz. *Numerical Methods for Non-Linear Algebraic Equations.* Gordon and Breach, London, UK, 1970.

[10] E. Schröder, *On infinitely many algorithms for solving equations* (in German), Math. Ann. 2 (1870), pp. 317–365. English translation by G. W. Stewart, TR-92-121, Institute for Advanced Computer Studies, University of Maryland, College Park, MD, 1992.

[11] Bl. Sendov, A. Andreev, and N. Kjurkchiev. Numerical solution of polynomial equations. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume III, pages 629–778. Elsevier Science, Cambridge, UK, 1994.

[12] J. F. Traub. *Iterative Methods for the Solution of Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1964.

[13] J. H Wilkinson. The perfidious polynomial. In G. H. Golub, editor, *Studies in Numerical Analysis*, pages 1–28. American Mathematical Society, Providence, RI, 1984.

[14] T. J. Ypma. Development of the Newton–Raphson's method. *SIAM Review*, 37:531–551, 1995.

accept single floats (not expressions) as input data and convert them to normalized mulprec numbers by means of the command *npr*, or *npc* for complex floats, see below.[8] Mulprec distinguishes between floats and mulprec numbers by the length, which is equal to 1 or larger than 1, respectively.

For a **complex**, normalized mulprec number these conditions typically hold for both the real and the imaginary part. The exponent and the length are common for both parts; an exception: $x(2)$ may thus be zero for one of the parts.

It is fundamental for Mulprec that P can be squared without overflow with some margin. In fact, $2^{53} > 90P^2$. Hence, if the shorter one of two positive normalized mulprec numbers has at most 90 gytes, we can obtain their product by the multiplication of gytes and addition of integers, so that the sums do not exceed 2^{53}. Typically, there is only one normalization in a multiplication.

The normalized representation of x is *unique* (if $x \neq 0$). For example, note that, if you subtract two positive normalized mulprec numbers, the gytes of the result may have varying signs, unless you **normalize** the result by the mulprec operation *nize* (or the simpler operation *rnize* if the number is real). Since the operation *rnize* is not fast compared to the operations *add* and *sub*, there is as a rule no normalization in *add* and *sub*.

For such reasons we now introduce a more general concept: the **legal mulprec number**; $\mathrm{val}(x)$ has the same value and the same form as the normalized mulprec number, but all the $x(j)$ need not have the same sign and they have a looser bound: $|x(j)| < 45\,P^2$.[9] Evidently such a representation of a number is not unique.

Allowing this more general type of mulprec number in additions and subtractions, makes it *unnecessary to transport carry digits inside these operations*; this is typically done later, if a normalization is needed.[10] A typical suboperation of the normalization is to subtract a multiple cP from one of the $x(j)$; this is typically compensated for by adding c to $x(j-1)$, in order to keep $\mathrm{val}(x)$ constant. (Is not that how we learned to handle the carry in addition in elementary school?)

Multiplication, division, elementary functions etc., do include normalization, both of the operands and of the results. Normalized numbers only should be printed.

The Mulprec Library

The mfiles for about 60 mulprec functions are packed together in the text file mulprec.lib, which can be downloaded from the books homepage. The numbers in the beginning of the lines of the

The mfiles for the following mulprec functions are packed together in the text file mulprec.lib, which can be downloaded from the books homepage. The numbers in the beginning of the lines of the lists below are only for making references in the text more convenient. They are thus *not* to be used in the codes and your commands.

[8]Expressions with mulprec operations are, however, allowed as input data.

[9]The addition of two legal numbers does not cause overflow, but the sum can be illegal at first and must be immediately normalized, see the next footnote.

[10]An exception: if the result of an *add* or a *sub* has become illegal, then it becomes acceptable after an automatic call of nize inside *add.m* (or *sub.m*).

Since the condensed comments in the table below, may be unclear, you are recommended to study the codes a little before you use the system. x, y, z are typically mulprec numbers. As mentioned above, most of the commands accept also floats as input if it makes sense.

In a command like $z = mul(x, y, s)$, the parameter s means the number of gytes wanted in the result (including the exponent; hence it equals the length in the Matlab sense). It is optional; if s is omitted the *exact* product is computed and normalized (not chopped).

An asterisk means that the code is longer than 500 bytes. The absence of an asterisk usually indicates, e.g., that the code is a relatively short combination of other library codes. The number in the beginning of the lines of the following table are not used in the computations; they are just for easy reference to the table and to mulprec.lib.

Basic arithmetic operations.

Addition, subtraction were commented above. Multiplication is performed as in elementary school—the amount of work is approximately proportional to the product of the sizes of the factors. Perhaps one of the fast algorithms presented in Knuth [2, Sec. 4.3.3], in the binary case, will be adapted to the gyte system in the future.

In the table below mul.m, the shorter of the operands is chosen to be the multiplier. In order to avoid overflow (in the additions inside the multiplication), the multiplier is chopped to 90 gytes (at most 623 decimal places). An operation that can handle a multiplier by partitioning it into 90-gyte pieces and calling mul.m once for each piece, is tentatively called mullong.m. It has not yet been implemented. *At present* there are bounds also for the accuracy for division, square root, elementary functions etc., since multiplication is used in their codes.

1/x and 1/sqrt(x) are implemented by Newton's iteration method, (with variable precision) that (roughly) doubles the number of gytes in each iteration. The initial approximation is obtained by the ordinary Matlab operations (giving approximately 16 correct decimals). See more details in mulprec.lib. The square root algorithm is division-free.

At present, some limitations of Mulprec are set by the restriction of the length of the shorter operand of a multiplication to at most 90 gytes). It does not seem to be very difficult to remove these bounds, or at least to widen them considerably.

1.*	add.m	$z = add(x, y)$	$z = x + y$
2a*.	sub.m	$z = sub(x, y)$	$z = x - y$
2b.	subb.m	$z = subb(x, y)$	$z = x + mi(y)$, shorter but slower than 2a
3*.	mul.m	$z = mul(x, y, s)$	$z = x \cdot y$, s optional, see above
3b.	mullong.m	$z = mul(x, y, s)$	$z = x \cdot y$. Unrestricted multiplication. Not yet implemented.
4*.	recip.m	$z = recip(x, s)$	$z = 1/x$

5.	div.m	$z = div(x, y, s)$	$z = x/y$
6a.	mi.m	$z = mi(x)$	$z = -x$; all components change sign *except* the exponent
6b.	muabs.m	$z = \|x\|$	Absolute value of a real or complex mulprec number Not yet implemented.
7*.	musqrt.m	$[y, iny] = musqrt(x, s)$	Returns sqrt(x) and optionally 1/sqrt(x)

Some special mulprec operations.

The operation chop.m is more general than just chopping to a desired length. See the code in mulprec.lib.The normalization code *rnize* still has a bug (?) that violates the uniqueness. It can happen, e.g., that the last two gytes of a positive number read −1 9999634 (say). Such nine-sequences may also occur at other places in the vector. Sometimes such a representation is more easily interpreted than a strictly normalized normalized mulprec number. I have therefore not yet tried to eliminate this "bug".

8.	npr.m	$xx = npr(x)$	Converts real float to normalized mulprec number
9.	npc.m	$xx = npc(x)$	Converts complex float to normalized mulprec number
10.	flo.m	$y = flo(x)$	Approximates mulprec number by float
11*.	chop.m	$y = chop(x, k)$	Returns approximately equivalent mulprec number, length k
12*.	rnize.m	$y = rnize(x)$	Normalizes real mulprec number
13.	nize.m	$y = nize(x)$	Normalizes complex mulprec number
14.	elizer.m	$y = elizer(x)$	Eliminates zero gytes in mulprec number, left and right.
15.	muzero.m	$y = muzero(x)$	If $x == 0$, $y = 1$, else $y = 0$.

Elementary functions.

In the computation of e^x, x real, we first seek $\bar{x}$ and an integer n, such that

$$e^x = e^{\bar{x}} P^n, \quad \text{and} \quad |\bar{x}| < \tfrac{1}{2} \ln P.$$

Then, for an appropriate integer m, $e^{\bar{x}/2^m}$ is computed by the $k-1$-term Maclaurin expansion, a Horner scheme with variable precision. $e^{\bar{x}}$ is then obtained by squaring the sum of the Maclaurin expansion m times. Suppose that the volume of computation is proportional to $m + ck$; the value $c = 0.4$ has been found by a combination of heuristic theory and experiment. In the code, the parameters m and k are obtained from an approximate formula for finding the minimum of $m + ck$ with the constraint that the bound for the relative error of $e^{\bar{x}/2^m}$, due to the Maclaurin truncation and the squarings of the Maclaurin sum does not exceed P^{-s}.

A similar idea is applied for e^{ix}. Now $\bar{x} \in [-8\pi, 8\pi]$, and $k - 1$ terms of the Taylor expansion into powers of $x/2^m$ are used. These methods are inspired from ideas developed by Napier and Briggs, when they computed the first tables of logarithms. See Goldstine [12].

The algorithms in lnr.m and muat2.m are based on Newton's method for the equations $e^y = x$ and $\tan y = x$, respectively, with initial approximations from the Matlab operations $\ln x$ and atan2(y,x). The commands muat2, lnc and mulog do not yet allow floats as input, and the codes are not well tested.

16*.	expo.m	$y = expo(x,s)$	$y = e^x$, x real,
17*.	expi.m	$[cox, six, eix] = expi(x,s)$	$\cos x$ and optionally $\sin x$, e^{ix}, x real
18.	muexp.m	$w = muexp(z,s)$	$y = e^z$, z complex
19*.	lnr.m	$y = lnr(x,s)$	$x > 0$
20*.	muat2.m	$v = muat2(y,x,s)$	adapted from atan2(y,x); not yet with float input
21a.	lnc.m	$w = lnc(z,s)$	$w = ln\ z,\ z \neq 0$; not yet with float input
21b.	mulog.m	$w = mulog(z,s)$	A better(?) name for 21a

A library for mulprec vector algorithms.

A **mulprec column vector** is represented by a (Matlab) rectangular matrix. A **mulprec row vector** is a row of mulprec numbers (where each mulprec number is a row of gytes). In a **rectangular mulprec matrix**, each column is a mulprec column vector, and each row is a mulprec row vector. So, we can say that a mulprec matrix is a row of rectangular (Matlab) matrices, all of the same size. The following set of operations is very preliminary. It was worked out for an application to repeated Richardson h^2 extrapolation, see the m-file rich3.m.

30*.	fixcom.m	$y = fixcom(x,a)$	x,y mulprec vectors. Returns $y \approx x$, $y(1) = a(1)$, length$(y(i))$ =length(a)
31*.	musv.m	$y = musv(sca, vec)$,	sca is mulprec scalar, vec is mulprec vector $y = sca \cdot vec$
32*.	scalp.m	$y = scalp(vec1, vec2)$,	scalar prod. in n-dim Eucl. space, vec_1, vec_2 mulprec column vectors
33.	adv.m	$z = adv(x,y)$	$z = x + y$; x, y, z mulprec vectors
34.	rnizev.m	$y = rnizev(x)$	Normalizes real mulprec vector
35.	chopv.m	$y = chopv(x,s)$	Chops components of mulprec vector to length s
36.	chonizv.m	$y = chonizv(x,s)$	Normalizes and chops a mulprec vector

Miscellaneous.

50*.	intro.m	Starting routine for Mulprec. See below.
51*.	rich3.m	Mulprec algorithm for repeated Richardson h^2 extr.
52*.	polygons.m	Compute circumference for a sequence of polygons. Calls rich3.m
53.	why.m	

There are also *edited diaries* of a few test experiments (comparisons of computations with different precision), e.g., pippi2.dia (π computed by polygons.m and rich3.m), muat2est.dia ($\pi = 4 \arctan 1$, etest.dia (e computed by expo.m).

How to start Mulprec.

Change directory to the seat of the Mulprec files.
Run intro.m. (*If you forget this*, you are likely to obtain confusing error messages. Ignore them and run intro.m!)
Then intro.m brings down the file *const.mat* from the disk. The file const.mat contains, e.g., 50 gytes mulprec approximations to π (called pilong), and to $\ln P$ (called LP), and the default values of some other global variables. Now Matlab is ready for your Mulprec adventures.

More Subprojects.

A mulprec analog to the matlab command rat for finding accurate (or exact) rational approximations to floating point results. In connection with this the basic operations of exact rational arithmetic and continued fractions, including gcd and lcm. (See Knuth [2, vol.II, sec. 4.5.2], in particular p. 327). Mulprec can, of course, not compete with Maple and similar systems for rational arithmetic. Minor tasks of this type may, however, appear in a context where Mulprec is used.

Interesting specific examples: difference schemes, the generalized Euler Transformation, the Euler–Maclaurin Formula, and other methods of convergence acceleration. Illconditioned power series, transformation of a moment sequence to the three-term recurrence coefficients for the orthogonal polynomials to the same weight distribution or, equivalently, transformation of a power series to a continued fraction. Gaussian elimination, Gram–Schmidt orthogonalization.

Theoretical analysis, if possible applied to built-in error estimation and control, e.g., chopping strategies for the construction of the m-files, both for the Mulprec library, and for suggestions to the Mulprec users.

Documentation, both comments in the codes, a detailed report, and (in particular) a clear, short and attractive booklet with a user's manual.

At present, some limitations of Mulprec are set by the restriction of the length of the shorter operand of a multiplication to at most 90 gytes. It does not seem to be very difficult to remove these bounds, or at least to widen them considerably.

Computer Exercises

1. As is well known $f(x) = (1 + x)^{1/x}$ has the limit $e = 2.71828\ 18284\ 59045\ldots$, when $x \to \infty$. Study the sequences $f(x_n)$ for $x_n = 10^{-n}$ and $x_n = 2^{-n}$, for $n = 1, 2, 3, \ldots$. Stop when $x_n < 10^{-10}$ (or when $x_n < 10^{-20}$ if you are using double precision). Give your results as a table of n, x_n, and the relative error $g_n = (f(x_n) - e)/e$. Also plot $\log(|g_n|)$ against $\log(|x_n)|$. Comment on and explain your observations.

Hint: The Maclaurin expansion of $\ln(1 + x)$ is useful. Both truncation and roundoff errors occur.

2. Make up and run some simple examples with several choices of the parameter s, such that you can easily check the accuracy of the result. For example: $1/7$, $\sqrt{0.75}$, $\sin(\pi/3)$, e, $4\arctan 1$. (Compare also the calculations in the dia files.)

3. The ancient Greeks computed approximate values of the circumference of the unit circle, 2π, by inscribing a regular polygon and computing its perimeter. Archimedes considered the inscribed 96-sided regular polygon, whose perimeter is 6.2821. In general, a regular n-sided polygon inscribed in a circle with radius 1 has circumference $c_n = 2n\sin\frac{\pi}{n}$. If we put $h = 1/n$, then

$$c(h) = c_{1/h} = \frac{2}{h}\sin\pi h = 2\pi - \frac{\pi^3}{3}h^2 + \frac{\pi^5}{60}h^4 - \ldots,$$

so $c(h)$ satisfies the assumptions for repeated Richardson extrapolation with $p_k = 2k$. One can prove that the following recursion formula leads from c_n to c_{2n}:

$$c_{2n} = 4n\sin\frac{\pi}{2n} = 4n\sqrt{\frac{1}{2}\left(1-\cos\frac{\pi}{n}\right)} = \sqrt{2-\sqrt{4-\left(c_n/n\right)^2}},$$

$$c_{2n}/(2n) = \frac{(c_n/n)}{\sqrt{2+\sqrt{4-(c_n/n)^2}}}.$$

(The last transformation is made to avoid cancellation and consequential round-off errors.)

The script file polygons.m uses this recursion after the substitutions

$$n = 6*2^{m-1}, \quad m = 1:M,\ M \le 36, \quad c_n = p(m+1), \quad q = p/n.$$

The script polygons.m then calls the function rich3.m that performs Richardson extrapolations until the list of M polygons is exhausted or the sequence of estimates of the limit 2π ceases to be monotonic.

Choose a suitable M, $M \le 36$, and call polygons.m. Compare with the diary file pippi2.dia that contains previous runs of this. Study the elapsed time.

4. Write a code for the summation of an infinite series $\sum f(n)$ by Euler–Maclaurin's Summation Formula, assuming that convenient algorithms exist for the integral and for derivatives of arbitrary order. Consider also how to handle generalized cases where a *limit* is asked for, rather than a sum, e.g., Stirling's asymptotic expansion for $\ln\Gamma(z)$ or the Euler constant γ.

The numerators and denominators of some Bernoulli numbers B_{2n}, $n = 1:17$, are found in the file const.mat, in the vectors $B2nN$ and $B2nD$, respectively. $B0 = 1$, $B1 = -1/2$, are given separately.

4. An interesting table of mathematical constants (40 decimal places) is given in Knuth [2, Appendix A, p. 659]. Compute a few of them to much higher accuracy. For some of them an estimate of the accuracy may be most easily obtained by comparing results obtained using different values of the parameter s. (Compare also the diary files given in the directory of Mulprec.[11]) Some of the constants may require some version of the Euler–Maclaurin formula, see Example 3. Incorporate them to your const.mat, if they are interesting. $\Gamma(1/3)$ and $-\zeta'(2)$ (the derivative of Riemann's ζ-function) seem to be relatively advanced tasks.

[11] We suspect that one digit is wrong in $\sqrt{5}$. Are we right? By the way, Knuth denotes $(\sqrt{5}+1)/2$ by ϕ.

5. Write and test a code for the product of a mulprec matrix by a mulprec vector. Incorporate into your mulprec library.

6. Implement mullong.m according to the indications given above in "Basic arithmetic operations", or in some different way. Do something about the consequences of this for expo.m, if you want to treat the next exercise.

7. Poisson's Summation Formula reads, in the case $f(t) = e^{-t^2h^2}$ with the Fourier Transform $\hat{f}(\omega) = (\sqrt{\pi}/h)e^{-\omega^2/(4h^2)}$,

$$h\sum_{n=-N}^{N} e^{-n^2h^2} = \sqrt{\pi}\sum_{k=-K+1}^{K-1} e^{-\pi^2k^2/h^2} + R_{h,N,K,choppings}.$$

This particular case is also known as the Theta Transformation Formula.

8. Suppose that you want to compute $\sqrt{\pi}$ to an extreme accuracy, by letting a computer that didn't cost more than \$2000 (say) work over a weekend with the use of Mulprec (with a few amendments). For a given (appprox.) bound for $R_{h,N,K,choppings}$, determine a good choice of the parameters h, N, K, and the parameter s in the various terms. Estimate roughly the relation of computing time to error.
• Problem 6 must have been treated, at least in principle, before you can solve this.
• Note that the function evaluation can be arranged as a set of recursion formulas with basic arithmetic operations only. I believe that only two or three evaluations of the exponential will be needed in the whole computation.
• Leave the door open for the use of variable precision, but I am not sure that it will reduce the computing time by a terrific amount in this exercise.
• Note that π appears in several places in the equation. Think of the computation as an iterative process (although in practice one iteration is perhaps enough).
• Before you make a full scale experiment, make sure that neither your computer—nor your office—will be a ruin, when you return after weekend.

[1] H. H. Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century.* Springer-Verlag, New York, 1977.

[2] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* Addison-Wesley, Reading, MA, second edition, 1981.

Appendix. Guide to Literature

For many readers numerical analysis is studied as an important applied subject. Since the subject is still in a dynamic stage of development, it is important to keep track of recent literature. Therefore we give in the following a more complete overview of the literature than is usually given in textbooks. We restrict ourselves to books written in English. The selection presented is by no means complete and reflects a subjective choice. However, we hope it can serve as a guide for a reader who out of interest (or necessity!) wishes to deepen his knowledge. Both more recent textbooks and older classics are included. A valuable source book to the literature before 1956 is Parke [29, 1958]. An interesting account of the history of numerical analysis from the 16th through the 19th century can be found in Goldstine [12, 1977]

Monographs specialized in linear algebra, approximation, ordinary and partial differential equations, and other various areas, will be listed together with a short introductory commentary in later chapters of this book. Reviews of most books of interest can be found in *Mathematical Reviews* as well as in *SIAM Review* and *Mathematics of Computation*

Textbooks in Numerical Analysis

Textbooks which can be read as a complement to this book include Van Loan [25], Conte and de Boor [6, 1980], Deuflhard and Hohmann [7, 1995], and Stewart [37, 38]. Gautschi [11, 1997] is a carefully written introductory text with a wealth of computer exercises and much valuable information in notes after each chapter. Several books contain listings of algorithms, or even comes with a disk containing software, for example, the introductory book by Forsythe, Malcolm, and Moler [8, 1977] and its successor Kahaner, Moler and Nash [2, 1988].

Press et al. [30, 1993] gives an unsurpassed survey of contemporary numerical methods for the applied scientist together with software available on-line. The book by Gander and Hřebiček [10, 1997] contains worked through examples on how to use modern software tools like Matlab and Maple in solving a selection of applied problems.

More advanced classical texts include Isaacson and Keller [22, 1966], Hamming [17, 1974], Ralston and Rabinowitz [32, 1978], and Schwarz [36, 1989]. The book by Stoer and Bulirsch [39, 2002], now in anew edition is particularly suitable for a reader with has a good mathematical background. Strang [40, 1986] gives an excellent modern introduction to applied mathematics.

[1] F. S Acton. *Numerical Methods That (Usually) Work.* Math. Assoc. of America, New York, second edition, 1990.

[2] D. Kahaner anf C. B. Moler and S. Nash. *Numerical Methods and Software.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[3] K. E. Atkinson. *An Introduction to Numerical Analysis.* Wiley, New York, second edition, 1989.

[4] E. K. Blum. *Numerical Analysis and Computation: Theory and Practice.* Addison-Wesley, Reading, MA, 1972.

[5] E. W. Cheney and D. Kincaid. *Numerical Mathematics and Computing.* Brooks and Cole, Pacific Grove, CA, third edition, 1994.

[6] S. D. Conte and C. de Boor. *Elementary Numerical Analysis. An Algorithmic Approach.* McGraw-Hill, New York, third edition, 1980.

[7] P. Deuflhard and A. Hohmann. *Numerical Analysis. A First Course in Scientific Computing.* de Gruyter, Berlin, 1995.

[8] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ, 1977.

[9] C.-E. Fröberg. *Numerical Mathematics. Theory and Computer Applications.* Benjamin/Cummings, Menlo Park, CA, 1985.

[10] W. Gander and J. Hřebiček. *Solving Problems in Scientific Computing using Maple and Matlab.* Springer-Verlag, Berlin, third edition, 1997.

[11] W. Gautschi. *Numerical Analysis.* Birkhäuser, Boston, MA, 1997.

[12] H. H. Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century.* Springer-Verlag, New York, 1977.

[13] G. H. Golub, editor. *Studies in Numerical Analysis.* The Math. Assoc. of America, 1984.

[14] G. H. Golub and J. M. Ortega. *Scientific Computing and Differential Equations. An Introduction to Numerical Methods.* Academic Press, San Diego, CA, 1992.

[15] G. H. Golub and J. M. Ortega. *Scientific Computing. An Introduction with Parallel Computing.* Academic Press, 1993.

[16] G. Hämmerlin and K.-H. Hoffmann. *Numerical Mathematics.* Springer-Verlag, Berlin, 1991.

[17] R. W. Hamming. *Numerical Methods for Scientists and Engineers.* McGraw-Hill, New York, second edition, 1974.

[18] M. T. Heath. *Scientific Computing. An Introductory Survey.* McGraw-Hill, Boston, MA, second edition, 2002.

[19] P. Henrici. *Elements of Numerical Analysis.* Wiley, New York, 1964.

[20] F. B. Hildebrand. *Introduction to Numerical Analysis.* McGraw-Hill, New York, 1974.

[21] A. S. Householder. *Principles of Numerical Analysis.* McGraw-Hill, New York, 1953.

[22] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods.* John Wiley, New York, 1966. Corrected reprint of 1966 original, 1994, Dover, New York.

[23] D. Kincaid and W. Cheney. *Numerical Analysis.* Brooks/Cole, Pacific Grove, CA, second edition, 1996.

[24] C. Lanczos. *Applied Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1956.

[25] C. F. Van Loan. *Introduction to Scientific Computing.* Prentice-Hall, Upper Saddle River, NJ, 1997.

[26] G. I. Marchuk. *Methods in Numerical Mathematics.* Springer-Verlag, Berlin, second edition, 1982.

[27] J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation.* American Institute of Physics, New York, second edition, 1990.

[28] J. Ortega. *Numerical Analysis: A Second Course.* Academic Press, New York, 1972.

[29] N. G. Parke. *Guide to the Literature of Mathematics and Physics.* Dover Publications, New York, second edition, 1958.

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran 77; The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, second edition, 1993.

[31] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics.* Springer-Verlag, New York, 2000.

[32] A. Ralston and P. Rabinowitz. *A First Course in Numerical Analysis.* McGraw-Hill, New York, second edition, 1978.

[33] J. R. Rice. *Mathematical Software.* Academic Press, New York, 1971.

[34] J. R. Rice. *Numerical Methods, Software, and Analysis.* Academic Press, New York, 1983.

[35] H. Rutishauser. *Lectures on Numerical mathematics.* Birkhäuser, Boston, MA, 1990.

[36] H. R. Schwarz. *Numerical Analysis. A Comprehensive Introduction.* Wiley, New York, 1989.

[37] G. W. Stewart. *Afternotes on Numerical Analysis.* SIAM, Philadelphia, PA, 1996.

[38] G. W. Stewart. *Afternotes Goes to Graduate School.* SIAM, Philadelphia, PA, 1997.

[39] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, third edition, 2002.

[40] G. Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA, 1986.

[41] J. Todd, editor. *A Survey of Numerical Analysis.* McGraw-Hill, New York, 1962.

[42] C. W. Ueberhuber. *Numerical Computation. 1 & 2.* Springer-Verlag, Berlin, 1997.

[43] J. S. Vandergraft. *Introduction to Numerical Computations.* Academic Press, New York, 1983.

[44] D. M. Young and R. T. Gregory. *A Survey of Numerical Analysis. Vol. 1.* Addison-Wesley, Reading, MA, 1973.

[45] D. M. Young and R. T. Gregory. *A Survey of Numerical Analysis. Vol. 2.* Addison-Wesley, Reading, MA, 1973.

Handbooks, Tables and Formulas

Some principal questions in the production of software for mathematical computation are discussed in Rice [33, 1971],

Mathematical tables are no longer as important for numerical calculations as they were in the pre-computer days. However, tables can often be an important aid in checking a calculation or planning calculations on a computer. Detailed advice about the use and choice of tables is given in Todd [41, 1962, pp. 93–106], The classical six-figure tables of A most comprehensive source of information on mathematical functions and formulas is Abramowitz and Stegun [1, 1965],

The three monographs edited by Jacobs [9, 1977], Iserles and Powell [8, 1987], and Duff and Watson [5, 1997] give exellent surveys of the development of "state of the art" methods in many different areas of numerical analysis during the last decades. The Handbook of Numerical Analysis [4], edited by P. G. Ciarlet and J. L. Lions, is a multivolume sequence that offers comprehensive coverage in all areas of numerical analysis as well as many actual problems of contemporary interest. Very useful surveys articles are to be found in ACTA Numerica, a Cambridge University Press Annual started in 1992.

Two general mathematics dictionaries, which are useful to have at hand are [11] and [16].

[1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions.* National Bureau of Standards, Dover, New York, 1965.

[2] Yu. A. Brychkov, A. P. Prudnikov, and O. I. Marichev. *Integrals and Series. Vol. 1: Elementary Functions.* Gordon and Breach, New York, 1986.

[3] R. Churchhouse, editor. *Handbook of Applicable Mathematics*, volume III. Numerical Methods. Wiley-Interscience, New York, 1981.

[4] P. G. Ciarlet and J. L. Lions. *Handbook of Numerical Analysis*, volume I–VII. North-Holland, Amsterdam, 1990–2000.

[5] I. S. Duff and G. A. Watson, editors. *The State of the Art in Numerical Analysis.* Clarendon Press, Oxford, 1997.

[6] B. Engquist and W. Schmid, editors. *Mathematics Unlimited—2001 and Beyond.* Springer-Verlag, Berlin, 2001.

[7] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series and Products.* Academic Press, London, UK, fifth edition, 1993.

[8] A. Iserles and M. J. D. Powell, editors. *The State of the Art in Numerical Analysis.* Clarendon Press, Oxford, 1987.

[9] D. A. H. Jacobs, editor. *The State of the Art in Numerical Analysis.* Clarendon Press, Oxford, 1977.

[10] E. Jahnke, F. Emde, and F. Lösh. *Tables of Higher Functions.* McGraw-Hill, New York, sixth edition, 1960.

[11] R. C. James and E. F. Beckenbach, editors. *James & James Mathematics Dictionary.* Van Nostrand, Princeton, NJ, third edition, 1968.

[12] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* Addison-Wesley, Reading, MA, third edition, 1998.

[13] A. V. Lebedev and R. M. Federova. *A Guide to Mathematical Tables.* Van Nostrand, New York, 1960.

[14] A. P. Prudnikov, Yu. A. Brychkov, and O. I. Marichev. *Integrals and Series. Vol. 2: Special Functions.* Gordon and Breach, New York, 1986.

[15] J. Spanler and K. B. Oldham. *An Atlas of Functions.* Springer-Verlag, Berlin, 1987.

[16] E. W. Weisstein, editor. *CRC Concise Encyclopedia of Mathematics.* CRC Press, Boca Raton, FL, 2000.

Index