Basic Methods in Computational Physics

Lecture Notes

E. Schachinger and D. J. Bonthuis

Lecture notes for the course “Computermethoden der Technischen Physik”,
compiled on October 25, 2023

Contents

Preface

Preface to the first edition

These lecture notes are based in part on the lecture notes *Numerische Methoden in der Physik* by Prof. Dr. Heinrich SORMANN (English translation by Prof. Dr. Lilia BOERI and Julian PILZ) and in part on the book entitled *Basic Concepts in Computational Physics* by Benjamin A. STICKLER and myself [Springer Verlag, Berlin, Heidelberg (2014)]. I am very grateful to Prof. SORMAN who graciously granted permission to make use of all parts of his lecture notes I deem to be of interest for this current set of lecture notes. Last but not least I have to thank my colleague Gerhard DORN for his valuable suggestions.

Ewald SCHACHINGER
Graz, 2017-10-01

Preface to the second edition

My sincere thanks go out to Ewald SCHACHINGER, who gracefully allowed me to use his original text and adapt it to fit my own computational physics lecture. Apart from the first edition of the lecture notes, these notes are based on a number of excellent texts:

- Elementary numerical analysis, S. D. CONTE and Carl DE BOOR, McGraw-Hill Book Company, third edition (1980)
- Numerical linear algebra, Nick TREFETHEN and David BAU, III, ISBN 0-89871-361-7 (1997)
- Numerical methods for ordinary differential equations: Initial Value Problems, David GRIFFITHS and Desmond HIGHAM, ISBN 978-0-85729-147-9 (2010)

These lecture notes are a work in progress. Corrections, suggestions and questions can be submitted by email to bonthuis@tugraz.at.

Douwe Jan BONTHUIS
Graz, 2021-09-18

Chapter 1

Numerical arithmetic

1.1 Motivation

Numerical physics is concerned with the development of fast and reliable algorithms for evaluating solutions to problems in physics. The reason we need numerical algorithms is that most problems in physics cannot be solved in exact arithmetic. In fact, the solutions of most problems with nonlinear terms or derivatives even require infinite combinations of algebraic operations.[1] Examples include solutions to algebraic equations of order 5 and higher, integral and differential equations, quadrature and transcendental equations. That means these equations cannot be solved exactly, even when working in exact arithmetic. In those cases, the process of finding an analytical solution to a problem in physics therefore only defers the inevitable numerical evaluation to the end of the calculation. The use of numerical algortihms is therefore central to all fields of physics.

Numerical algorithms are restricted to using a finite sequence of basic mathematical operations such as addition, subtraction, multiplication and division using finite-digit numbers [3]. We have already specified what properties these numerical algorithms need to have. First, we require the algorithms to be *fast*, which means we care about the number of operations and how this number scales with the size of the problem. Second, *reliable* means we want the algorithms to be stable and to approach the exact solution to the required accuracy.

For infinite algorithms, that implies that repetition of the procedure should approximate the exact result to arbitrary accuracy. Importantly, we need to be able to evaluate the error incurred by approximating the exact solution through a finite number of operations. Linear problems, of the type $A\boldsymbol{x} = \boldsymbol{b}$, can typically be solved by a finite algorithm. That means that these algorithms take a finite number of steps to arrive at the exact solution. However, when performing these steps on a computer – or on paper for that matter – rounding errors are typically unavoidable. Storing numbers on a piece of paper has many of the same issues as those associated with storing numbers in a computer.

The occurring errors can be classified based on the structure every numerical routine follows: we have input-errors, algorithmic-errors, and output-errors as indicated schematically in Fig. 1.1. This structural classification can be refined: input-errors are divided into rounding errors and measurement errors contained in the input data;

[1]The algebraic operations are addition, subtraction, multiplication, division and exponentiation with an integer or fractional power, the latter being equivalent to root extraction.

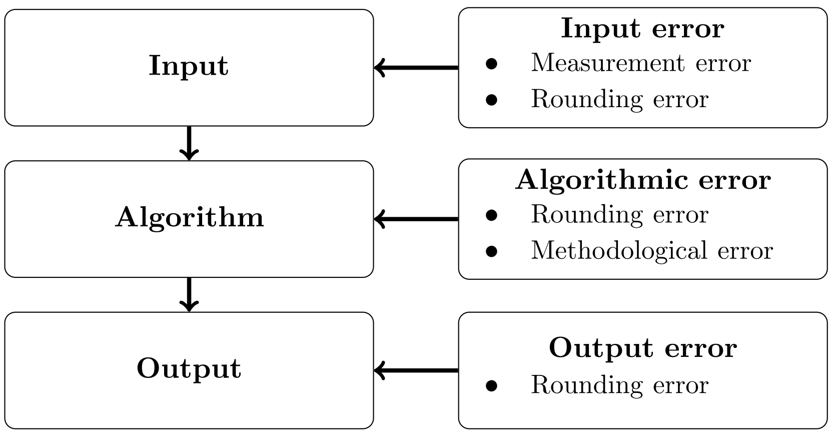

Figure 1.1: Schematic classification of the errors occurring within a numerical procedure.

algorithmic-errors consist of rounding errors during evaluation and of methodological errors due to mathematical approximations; finally, output errors are, in fact, rounding errors. In Secs. 1.2 and 1.3 we will concentrate on rounding errors and methodological errors. Since in most cases measurement errors cannot be influenced by the theoretical physicist concerned with numerical modeling, this particular part will not be discussed here. It is rather a topic lectures in experimental physics will be concerned with. However, we will discuss the stability of numerical routines, *i.e.* the influence of slight modifications of the input parameters or errors incurred throughout the evaluation on the outcome of a particular algorithm in Sec. 1.4.

1.2 Rounding errors

Since every number is stored in a computer using a finite number of digits, we have to truncate every non-terminating number at some point. The error incurred through this truncation depends on the number basis used.

Example 1.1
Consider the number

$$\frac{2}{3} = 0.666666666666\ldots_{10}\,,$$

where we use the decimal basis, indicated by the subscript 10, for the floating-point representation. If the machine allows only ten digits, this will be stored as 0.6666666667_{10}.

Example 1.2
Actually, computers use binary arithmetic. In this case the number

$$0.1_{10} = 0.000110011001100\ldots_2$$

cannot be stored exactly.[a] It becomes, obviously, a periodic binary number. Writing it in a 2-digit floating point representation, the number becomes

Table 1.1: Illustration of the difference between absolute and relative error.

	y	$\overline{y}$	ε_a	ε_r
(1)	0.1	0.09	0.01	0.1
(2)	1000.0	999.99	0.01	0.00001

$0.11_2\, 2^{-3} = 0.09375_{10}$.

[a]A disastrous effect of this binary approximation of 0.1 is discussed by T. Chartier [2].

Example 1.3
In 10-digit floating point format using rounding for the last digit, $\mathrm{Fl}(\sqrt{3}) = 0.1732050808\, 10^1$. This has the consequence that

$$\mathrm{Fl}(\sqrt{3}) \cdot \mathrm{Fl}(\sqrt{3}) = 0.300000000149\, 10^1 \neq \mathrm{Fl}(\sqrt{3} \cdot \sqrt{3}) = 0.3000000000\, 10^1.$$

To quantify the error, we introduce the concepts of the absolute and the relative error. We denote the true value of a quantity by y and its approximate value by $\overline{y}$. Then the absolute error ε_a is defined as

$$\varepsilon_a = |y - \overline{y}|, \tag{1.1}$$

while the relative error ε_r is given by

$$\varepsilon_r = \left| \frac{y - \overline{y}}{y} \right| = \frac{\varepsilon_a}{|y|}, \tag{1.2}$$

provided that $y \neq 0$. In most applications, the relative error is more meaningful. This is illustrated in Tab. 1.1, where it is intuitively obvious that in the second case the approximate value is much better although the absolute error is the same for both examples.

Let us have a look at the relative error of an arbitrary number stored to the k-th digit. We can write an arbitrary number y in the decimal basis in the form $y = 0.d_1 d_2 d_3 \dots d_k d_{k+1} \dots 10^n$ with $d_1 \neq 0$ and $n \in \mathbb{Z}$. Accordingly, we write its approximate value as $\overline{y} = 0.d_1 d_2 d_3 \dots d_k 10^n$, where k is the maximum number of digits stored by the machine and the number is truncated afterward. We obtain for the relative error

$$\begin{aligned}
\varepsilon_r &= \left| \frac{0.d_1 d_2 d_3 \dots d_k d_{k+1} \dots 10^n - 0.d_1 d_2 d_3 \dots d_k 10^n}{0.d_1 d_2 d_3 \dots d_k d_{k+1} \dots 10^n} \right| \\
&= \left| \frac{0.d_{k+1} d_{k+2} \dots 10^{n-k}}{0.d_1 d_2 d_3 \dots 10^n} \right| \\
&= \left| \frac{0.d_{k+1} d_{k+2} \dots}{0.d_1 d_2 d_3 \dots} \right| 10^{-k} \\
&\leq \frac{1}{0.1} 10^{-k} = 10^{-k+1}.
\end{aligned} \tag{1.3}$$

In the last step we employed that, since $d_1 \neq 0$, we have $0.d_1 d_2 d_3 \dots \geq 0.1$ and accordingly $0.d_{k+1} d_{k+2} \dots < 1$. If the last digit would have been rounded to d'_k, in which case we set $d'_k = d_k + 1$ if $d_{k+1} \geq 5$ and $d'_k = d_k$ otherwise instead of a simple truncation, the relative error of a variable y would be $\varepsilon_r \leq 0.5 \cdot 10^{-k+1}$.

Whenever an arithmetic operation is performed, the errors of the variables involved are transferred to the result [8]. This can occur in an advantageous or disadvantageous way, where we understand disadvantageous as an increase in the relative error. Particular care is required when two nearly identical numbers are subtracted (*subtractive cancellation*) or when a large number is divided by a, in comparison, small number. In such cases the rounding error will increase dramatically. We note that it might be necessary to avoid such operations in our aim to design an algorithm which is required to produce reasonable results. An illustrative example and its remedy will be discussed in Sec. 1.3. However, before proceeding to the next section we introduce a lower bound to the accuracy which is achievable with a non-ideal computer, the *machine-number*. The machine-number is the smallest positive number η which can be added to another number, such that a change in the result is observed. In particular,

$$\eta = \min_{\delta} \left\{ \delta > 0 \middle| 1 + \delta > 1 \right\}. \tag{1.4}$$

For a (nonexistent) super-computer, which is capable of saving as much digits as desired, η would be arbitrarily small. A typical value for double-precision in Fortran or C is $\eta \approx 10^{-16}$.

1.3 Methodological errors

For many numerical routines, exact mathematical expressions need to be replaced by a finite set of algebraic operations. Inevitably, a methodological error is introduced into the routine if the exact evaluation requires an infinite set of such operations. In particular, for the numerical evaluation of transcendental functions, the solution of algebraic equations of order 5 and higher as well as integration and differentiation, the mathematical operations need to be replaced by approximate expressions. An intriguing case in point is the numerical differentiation of a function. The first-order approximation of a derivative reads

$$f'(x_0) = \frac{\mathrm{d}}{\mathrm{d}x} f(x)\bigg|_{x=x_0} \approx \frac{f(x_0+h) - f(x_0)}{h}. \tag{1.5}$$

This approximation is referred to as *finite difference* and will be discussed in more detail in Chap. 7.5. One would, in a first guess, expect that the obtained value gets closer to the true value of the derivative $f'(x_0)$ with decreasing values of h. From a calculus point of view, this is correct since by definition

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x)\bigg|_{x=x_0} = \lim_{h\to 0} \frac{f(x_0+h) - f(x_0)}{h}. \tag{1.6}$$

However, this is not the case numerically. In particular, one can find a value $\hat{h}$ for which the relative error is minimal, while for values $h < \hat{h}$ and $h > \hat{h}$ the approximation obtained is worse in comparison. The reason is that for small values of h the rounding errors dominate the result since $f(x_0+h)$ and $f(x_0)$ almost cancel while $1/h$ is very small. Nevertheless, for $h > \hat{h}$, the methodological error, *i.e.* the

Table 1.2: Numerical evaluation of $\bar{f}(x)$ and $\hat{f}(x)$.

x	$\bar{f}(x)$	$\hat{f}(x)$
10^{0}	1.41421354e+00	1.41421354e+00
10^{-1}	1.00125492e+00	1.00125551e+00
10^{-2}	1.00001693e+00	1.00001252e+00
10^{-3}	9.99987185e–01	1.00000012e+00
10^{-4}	9.99570012e–01	1.00000000e+00
10^{-5}	1.00135815e+00	1.00000000e+00
10^{-6}	9.53674436e–01	1.00000000e+00
10^{-7}	5.96046507e–01	1.00000000e+00
10^{-8}	0.00000000e+00	1.00000000e+00

replacement of a derivative by a finite difference, determines the accuracy. We will return to this point in more detail in Sec. 7.5.

Due to the effect of rounding errors, also routines that can be performed exactly using only algebraic operations can incur methodological errors. We give an example in order to illustrate the interplay between methodological errors and rounding errors [1].

Example 1.4
Regard the – apparently nonhazardous – numerical solution of a quadratic equation

$$ax^2 + bx + c = 0, \tag{1.7}$$

where $a, b, c \in \mathbb{R}$, $a \neq 0$. The well known solutions are

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \tag{1.8}$$

Cautious because of the explanation in Sec. 1.2, we immediately diagnose the danger of a subtractive cancellation in the expression of x_1 for $b > 0$ or in x_2 for $b < 0$, and rewrite the above expression for x_1,

$$x_1 = \frac{\left(-b + \sqrt{b^2 - 4ac}\right)}{2a} \frac{\left(-b - \sqrt{b^2 - 4ac}\right)}{\left(-b - \sqrt{b^2 - 4ac}\right)} = \frac{2c}{-b - \sqrt{b^2 - 4ac}}. \tag{1.9}$$

For x_2 we obtain

$$x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}. \tag{1.10}$$

Consequently, if $b > 0$ x_1 should be calculated using Eq. (1.9) and if $b < 0$ Eq. (1.10) should be used to calculate x_2. Moreover, the above expressions can be cast into one expression by setting

$$x_1 = \frac{q}{a} \quad \text{and} \quad x_2 = \frac{c}{q} \quad \text{with} \quad q = -\frac{1}{2}\left[b + \operatorname{sgn}(b)\sqrt{b^2 - 4ac}\right]. \tag{1.11}$$

Thus, Eq. (1.11) can be used to calculate x_1 and x_2 for any sign of b.

Example 1.5
Another intriguing example can be found when we attempt to evaluate numerically the rather simple expression

$$f(x) = \frac{\sqrt{1+x} - \sqrt{1-x}}{x} \tag{1.12}$$

for $x \ll 1$. Since $f(x)$ is well defined by an analytical expression there is no methodological error when $f(x)$ is evaluated numerically. Moreover, we can safely assume that in this particular case there are neither input nor output errors. If we evaluate Eq. (1.12) without further reshaping in floating point format, we obtain the results listed as $\bar{f}(x)$ in the second column of Tab. 1.2. Two features immediately catch our attention in the values of $\bar{f}(x)$ listed in this table:

- The value of the function $\bar{f}(x)$ does not tend to 1 for $x \to 0$, as it must. Instead it begins to decrease monotonically for $x \le 10^{-6}$.
- For values of $x \le 10^{-8}$ $\bar{f}(x)$ is exactly zero.

This behavior is easily explained:

- For small values of x the two square roots $\sqrt{1+x}$ and $\sqrt{1-x}$ represent two almost equal numbers. Subtracting these numbers from each other, and dividing the result by x, the two rounding errors, that are by themselves small, give a large effect, due to the law of error propagation ('subtractive cancellation').
- For $x \ll \tau$, we have that both $1+x = 1$ and $1-x = 1$; thus, the value of $f(x)$ is exactly zero. (Here, $\tau = 1.1920929\mathrm{e}-07$ is the machine number on a standard PC for single precision.)

So far, we only identified the error. The cure in this case is very simple: We rewrite $f(x)$ as

$$f(x) = \frac{2}{\sqrt{1+x} + \sqrt{1-x}}, \tag{1.13}$$

and recover the correct result listed as $\hat{f}(x)$ in the third column of Tab. 1.2. This example shows quite clearly that the rounding error which appears during the evaluation of a mathematical expression can be reduced with the help of an appropriate reformulation of the algorithm after it has been recognized as such.

1.4 Stability and conditioning

When a new numerical method is designed, *stability* is the third crucial point after rounding errors and methodological errors [4]. We give an introductory definition:

> An algorithm is referred to as unstable if small changes in the input (backward error) cause a large change in the output (forward error).

More generally, a mathematical problem expressed by a function $f(x)$ is called *ill-conditioned* if a small perturbation in x leads to a large change in $f(x)$. We illustrate the conditioning of functions by a couple of elucidating examples [5].

Example 1.6
Consider the following system of equations

$$\begin{aligned} x + y &= 2.0, \\ x + 1.01y &= 2.01. \end{aligned} \tag{1.14}$$

The equations are easily solved and give $x = 1.0$ and $y = 1.0$. To make our point we consider now the case in which the right hand side of the second equation of (1.14) is subjected to a small perturbation, i.e. we consider in particular the following system of equations

$$\begin{aligned} x + y &= 2.0, \\ x + 1.01y &= 2.02. \end{aligned} \tag{1.15}$$

The corresponding solution is $x = 0.0$ and $y = 2.0$. We observe that a relative change of 0.05 % on the right hand side of the second equation in (1.14) resulted in a 100 % relative change of the solution. Moreover, if the coefficient of y in the second equation of (1.14) were 1.0 instead of 1.01, which corresponds to a relative change of 1 %, the equations would be unsolvable. This is a behavior typical for ill-conditioned problems.

Example 1.7
Consider the initial value problem

$$\begin{cases} \ddot{y} - 10\dot{y} - 11y = 0, \\ y(0) = 1, \qquad \dot{y}(0) = -1. \end{cases} \tag{1.16}$$

The general solution is readily obtained to be of the form

$$y = A\exp(-t) + B\exp(11t), \tag{1.17}$$

with numerical constants A and B. The initial conditions yield the unique solution

$$y = \exp(-t). \tag{1.18}$$

The initial conditions are now changed by two small parameters $\delta, \varepsilon > 0$ to give:

$$y(0) = 1 + \delta \qquad \text{and} \qquad \dot{y}(0) = -1 + \varepsilon. \tag{1.19}$$

The unique solution which satisfies these initial conditions is:

$$\overline{y} = \left(1 + \frac{11\delta}{12} - \frac{\varepsilon}{12}\right)\exp(-t) + \left(\frac{\delta}{12} + \frac{\varepsilon}{12}\right)\exp(11t). \tag{1.20}$$

We calculate the relative error

$$\begin{aligned}\varepsilon_r &= \left|\frac{y-\overline{y}}{y}\right| \\ &= \left(\frac{11\delta}{12}-\frac{\varepsilon}{12}\right)+\left(\frac{\delta}{12}+\frac{\varepsilon}{12}\right)\exp(12t)\,, \end{aligned} \tag{1.21}$$

which indicates that the problem is ill-conditioned since for large values of t the second term definitely overrules the first one.

To quantify the conditioning of functions, we define the condition number. Let δx denote a small perturbation of x (the backward error) and write $\delta f = f(x+\delta x) - f(x)$ (the forward error). The absolute condition number κ_a of the problem f at x is defined as

$$\kappa_a = \lim_{\delta\to 0}\sup_{|\delta x|<\delta}\frac{|\delta f|}{|\delta x|}. \tag{1.22}$$

If f is differentiable, we can evaluate κ_a by means of the derivative $f'(x)$. In the limit $\delta x \to 0$, $\delta f = f'(x)\delta x$, with which the absolute condition number becomes

$$\kappa_a = |f'(x)|. \tag{1.23}$$

Similarly, the relative condition number is defined as

$$\kappa_r = \lim_{\delta\to 0}\sup_{|\delta x|<\delta}\frac{|\delta f|/|f(x)|}{|\delta x|/|x|}. \tag{1.24}$$

For differentiable functions, the relative condition number becomes

$$\kappa_r = f'(x)\frac{|x|}{|f(x)|}. \tag{1.25}$$

The larger κ_r, the more ill-conditioned a function is said to be. For vector-valued functions, the derivative is replaced by the Jacobian $J(\boldsymbol{x})$, whose entries i, j are the partial derivatives $\partial f_i/\partial x_j$, evaluated in $\boldsymbol{x}$, with f_i and x_j being the components of $\boldsymbol{f}$ and $\boldsymbol{x}$. The absolute values are replaced by vector norms.

Whereas conditioning pertains to the properties of the mathematical problem, stability pertains to the behavior of an algorithm used to solve that problem. Some problems are inherently ill-conditioned, but algorithms to solve well-conditioned problems can typically be designed to be stable. In particular, we can view the algorithm as the sequential evaluation of n functions, leading to a sequence containing the input x_0, a number of intermediate results x_i with $i = 1 \ldots n-1$ and the output x_n. We denote the function that transforms the intermediate result x_i to the final result x_n by f_i,

$$x_n = f_i(x_i), \quad i = 0 \ldots n-1, \tag{1.26}$$

which means that the complete algorithm is written as $f_0(x_0)$. An algorithm is stable if none of the functions f_i is ill-conditioned.[2]

The reason that every function f_i must be well conditioned is that errors occur in every numerical operation of an algorithm, in addition to perturbing the input. If these errors are enhanced during the execution of the rest of the algorithm, this is called an *induced instability*:

[2]Although unstable behavior is typically not desirable, the discovery of unstable systems marked the birth of a specific branch in physics called *Chaos Theory*. We briefly comment on this point at the end of this section.

> A method is referred to as induced unstable if a small error at one point of the calculation induces a large error at some subsequent point.

Induced instability is particularly dangerous since small rounding errors are unavoidable in most calculations. Hence, if some part of the whole algorithm is ill-conditioned, the final output will be dominated by the error induced in such a way. Again, an example will help to illustrate such behavior.

Example 1.8
The definite integral

$$I_n = \int_0^1 \mathrm{d}x\, x^n \exp(x-1) \ , \tag{1.27}$$

is considered. Integration by parts yields

$$I_n = 1 - nI_{n-1} \ . \tag{1.28}$$

This expression can be used to recursively calculate I_n from I_0, where

$$I_0 = 1 - \exp(-1) \ . \tag{1.29}$$

Although the recursion formula (1.28) is exact we will run into massive problems using it. The reason is easily illustrated:

$$\begin{aligned} I_n &= 1 - nI_{n-1} \\ &= 1 - n + n(n-1)I_{n-2} \\ &= 1 - n + n(n-1) - n(n-1)(n-2)I_{n-3} \\ &\ \ \vdots \\ &= 1 + \sum_{k=1}^{n-1}(-1)^k \frac{n!}{(n-k)!} + (-1)^{n-1} n! I_0 \ . \end{aligned} \tag{1.30}$$

Thus, the initial rounding error included in the numerical value of I_0 is multiplied with $n!$. Note that for large n we have according to STIRLING's approximation

$$n! \approx \sqrt{2\pi} n^{n+\frac{1}{2}} \exp(-n) \ , \tag{1.31}$$

i.e. an initial error increases almost as n^n.

However, equation (1.28) can be reformulated to give

$$I_n = \frac{1}{n+1}(1 - I_{n+1}) \ , \tag{1.32}$$

and this opens an alternative method for a recursive calculation of I_n. We can start with some value $N \gg n$ and simply set $I_N = 0$. The error introduced in such a way may in the end not be acceptable, nevertheless, it decreases with every iteration step due to the division by n in Eq. (1.32).

Having discussed some basic features of stability in numerical algorithms we would like to add a few remarks on *Chaos Theory*. Chaos theory investigates dynamical processes which are very sensitive to initial conditions. One of the best known examples for such a behavior is the weather prediction. Although, POINCARÉ already observed chaotic behavior while working on the three body problem, one of the pioneers of chaos theory was E. N. LORENZ [6].[3] In 1961 he ran weather simulations on a computer of restricted capacity. When he tried to reproduce one particular result by restarting the calculation with new parameters calculated the days before, he observed that the outcome was completely different [7]. The reason was that the equations he dealt with were ill-conditioned, and the rounding error he introduced by simply typing in the numbers off the graphical output increased drastically over the course of the calculation, producing a completely different result. Nowadays, various physical systems are known which indeed behave in such a way. Apart from wheather preditions, examples include turbulences in fluids, oscillations in electrical circuits, oscillating chemical reactions, population growth in ecology, the time evolution of the magnetic field of celestial bodies, *etc.*

It is important to note, that chaotic behavior induced in such systems is *deterministic*, yet *unpredictable*. This is due to the impossibility of an exact knowledge of the initial conditions required to predict, for instance, the weather over a reasonably long period. A feature which is referred to as the *butterfly effect*: a hurricane can form because a butterfly flapped its wings several weeks before. However, these effects have nothing to do with intrinsically probabilistic properties which are solely a feature of quantum mechanics. In contrast to this, in chaos theory, the future is uniquely determined by initial conditions yet still unpredictable. This is often referred to as *deterministic chaos*.

It has to be emphasized that chaos in physical systems is a consequence of the equations describing the processes and not a consequence of the numerical method used for modeling. Therefore, it is important to distinguish between the stability of a numerical method and the conditioning of a physical system in general.

Bibliography

[1] Burden, R.L., Faires, J.D.: Numerical Analysis. PWS-Kent Publishing Comp., Boston (1993)

[2] Chartier, T.: Devastating roundoff error. Math. Horizons **13**, 11 (2006). DOI 10.1080/10724117.2006.11974643. URL http://www.jstor.org/stable/25678616

[3] Ernst, H.: Numerische Methoden für Kleincomputer. Luther Verlag (1984)

[4] Higham, N.J.: Accuracy and Stability of Numerical Algorithms, 2nd edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2002)

[5] Jacques, I., Judd, C.: Numerical Analysis. Chapman and Hall, London (1987)

[6] Lorenz, E.N.: Deterministic nonperiodic flow. Journal of the Atmospheric Sciences **20**, 130–141 (1963)

[7] Roulstone, I., Norbury, J.: Invisible in the Storm: The Role of Mathematics in Understanding Weather. Prinecton University Press, Princeton, USA (2013)

[8] Ueberhuber, C.W.: Numerical Computation 1: Methods, Software and Analysis. Springer, Berlin, Heidelberg (1997)

[3]Not to be confused with H. LORENTZ, who introduced the LORENTZ transformation.

Chapter 2

Numerical solution of systems of linear equations

2.1 Setting the stage

We plan to discuss here briefly two of the most important methods to solve non-homogeneous systems of linear equations by numerical methods. We consider a system of n equations of the form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \ , \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \ , \\ \vdots \quad &\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \ , \end{aligned} \tag{2.1}$$

which is usually transformed into a matrix equation,

$$A\mathbf{x} = \mathbf{b}. \tag{2.2}$$

The coefficients of the matrix $A = [a_{ij}]$ as well as the vector $\mathbf{b} = [b_i]$ are assumed to be real valued and, furthermore, if

$$\sum_{i=1}^{n} |b_i| \neq 0 \ , \tag{2.3}$$

the problem (2.2) is referred to as non-homogeneous (in-homogeneous). The solution of such non-homogeneous systems of equations is one of the central problems in numerical analysis, since numerous numerical methods, such as methods for the numerical interpolation (see Chap. 3), least-square methods (see Chap. 4), differential methods, etc. can be reduced to the problem of solving a set of non-homogeneous linear equations.

The solution to (2.2) is well defined as long as the matrix A is non-singular, i.e. as long as

$$\det(A) \neq 0 \ . \tag{2.4}$$

Then the unique solution of (2.2) can be written as

$$\mathbf{x} = A^{-1}\mathbf{b}. \tag{2.5}$$

However, the inversion of matrix A is very complex for $n \geq 4$ and one would prefer methods which are computationally more effective. Basically, one distinguishes between *direct* and *iterative* methods. Since a complete discussion of this huge topic would be too extensive, we will mainly focus on two methods.

In contrast to iterative procedures, direct procedures do not contain any methodological errors and can, therefore, be regarded to be *exact*. However, these methods are often computationally very extensive and roundoff errors are in many cases not negligible. As an example we will discuss the *LU* decomposition. On the other hand, many iterative methods are fast and roundoff errors can be controlled easily. However, it is not guaranteed that an iterative procedure converges, even in cases where the system of equations is known to have unique solutions. Moreover, the result is an approximate solution. As an illustration for an iterative procedure we will discuss the Gauss-Seidel method.

The most important element in this context is the matrix of coefficients A. The properties of this matrix determine dominantly the numerical method which will have to be applied to solve the non-homogeneous system of equations. Thus, it appears to be convenient at this point to review a few basic elements of linear algebra concerning matrices and their properties.

2.2 Matrices in a nutshell

We follow in this chapter quite closely chapter 1 of the book by Y. Saad [4]. For the sake of generality, all vector spaces considered in this chapter are complex, unless otherwise stated. A complex $n \times m$ matrix A is an $n \times m$ array of complex numbers

$$a_{ij}, \quad i = 1, \ldots, n, \; j = 1, \ldots, m.$$

The set of all $n \times m$ matrices is a complex vector space denoted by $\mathbb{C}^{n \times m}$.

2.2.1 Matrix operations

The main operations with matrices are the following:

- Addition: $C = A + B$, where A, B, and C are matrices of size $n \times m$ and

$$c_{ij} = a_{ij} + b_{ij}, \quad i = 1, \ldots, n, \; j = 1, \ldots, m.$$

- Multiplication by a scalar: $C = \alpha A$, where

$$c_{ij} = \alpha a_{ij}, \quad i = 1, \ldots, n, \; j = 1, \ldots, m.$$

- Multiplication by another matrix: $C = AB$ where $A \in \mathbb{C}^{n \times m}$, $B \in \mathbb{C}^{m \times p}$, $C \in \mathbb{C}^{n \times p}$, and

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}.$$

The *transpose* of a matrix $A \in \mathbb{C}^{n \times m}$ is a matrix $C \in \mathbb{C}^{m \times n}$ whose elements are defined by $c_{ij} = a_{ji}, i = 1, \ldots, m, \; j = 1, \ldots, n$. It is denoted by A^T. It is often more relevant to use the *transpose conjugate* matrix denoted by $A^\dagger$ and defined by

$$A^\dagger = (A^\star)^T = \left(A^T\right)^\star,$$

in which the $\star$ denotes the (element-wise) complex conjugation.

Matrices are strongly related to linear mappings between vector spaces of finite dimension. This is because they represent these mappings with respect to two given bases: one for the initial vector space and the other for the image vector space, or *range* of A.

2.2.2 Square matrices and eigenvalues

A matrix is *square* if it has the same number of columns and rows, i.e., if $m = n$. An important square matrix is the *identity matrix*

$$I = [\delta_{ij}]_{i,j=1,\ldots,n},$$

where δ_{ij} is the KRONECKER delta. The identity matrix satisfies the equality $AI = IA = A$ for every matrix A of size n. The inverse of a matrix, when it exists, is a matrix C such that

$$CA = AC = I.$$

The inverse of A is denoted by A^{-1}.

The *determinant* of a matrix may be defined in several ways. For simplicity, the following recursive definition is used here. The determinant of a 1×1 matrix $[a]$ is defined as the scalar a. Then the determinant of an $n \times n$ matrix is given by

$$\det(A) = \sum_{j=1}^{n} (-1)^{j+1} a_{1j} \det(A_{1j}),$$

where A_{1j} is an $(n-1) \times (n-1)$ matrix obtained by deleting the first row and the j-th column of A. (LAPLACE formula.) A matrix is said to be *singular* when $\det(A) = 0$ and *non-singular* otherwise. We have the following simple properties:

- $\det(AB) = \det(A)\det(B)$.
- $\det\left(A^T\right) = \det(A)$.
- $\det(\alpha A) = \alpha^n \det(A)$.
- $\det\left(A^\star\right) = [\det(A)]^\star$.
- $\det(I) = 1$.

From the above definition of determinants it can be shown by induction that the function that maps a given complex value λ to the value $p_A(\lambda) = \det(A - \lambda I)$ is a polynomial of degree n. This is known as the *characteristic polynomial* of the matrix A.

We can demonstrate this quite easily: We concentrate first on a diagonal matrix $A = \operatorname{diag}(a_{11}, a_{22}, \ldots, a_{nn})$. In this case, the matrix elements are also the eigenvalues of the diagonal matrix A. For the characteristic polynomial follows immediately:

$$p_A(\lambda) = \det(A - \lambda I) = (a_{11} - \lambda)(a_{22} - \lambda) \cdots (a_{nn} - \lambda).$$

Consequently, λ is an eigenvalue of a general matrix A if and only if there is an eigenvector $\mathbf{u} \neq 0$ such that

$$A\mathbf{u} = \lambda \mathbf{u},$$

or

$$(A - \lambda I)\mathbf{u} = 0.$$

Since $\mathbf{u}$ is non-zero, the matrix $A - \lambda I$ is singular and this, in turn, means that its determinant is zero. Thus, the roots of the function $\det(A - \lambda I)$ are the eigenvalues of A and it is obvious that this determinant is a polynomial in λ.

Definition 1
A complex scalar λ is called an eigenvalue of the square matrix A if a non-zero vector $\mathbf{u}$ of $\mathbb{C}^n$ exists such that $A\mathbf{u} = \lambda\mathbf{u}$. The vector $\mathbf{u}$ is called an eigenvector of A associated with λ. The set of all the eigenvalues of A is called the spectrum of A and is denoted by $\sigma(A)$.

We emphasize: A scalar λ is an eigenvalue of A if and only if $\det(A - \lambda I) \equiv p_A(\lambda) = 0$. That is true if and only if λ is a root of the characteristic polynomial. In particular, there are at most n distinct eigenvalues.

It is clear that a matrix is singular if and only if it admits zero as an eigenvalue. A well known result in linear algebra is stated in the following proposition.

Proposition 1
A matrix A is non-singular if and only if it admits an inverse.

Thus, the determinant of a matrix determines whether or not the matrix admits an inverse.

The maximum modulus of the eigenvalues is called *spectral radius* and is denoted by $\rho(A)$

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|.$$

The *trace* of a matrix is equal to the sum of all its diagonal elements

$$\mathrm{tr}\,(A) = \sum_{i=1}^{n} a_{ii}.$$

It is easy to prove that the trace of A is also equal to the sum of the eigenvalues of A counted with their multiplicities as roots of the characteristic polynomial.

Proposition 2
If λ is an eigenvalue of A, then $\lambda^\star$ is an eigenvalue of $A^\dagger$. An eigenvector $\mathbf{v}$ of $A^\dagger$ associated with the eigenvalue $\lambda^\star$ is called a left eigenvector of A.

When a distinction is necessary, an eigenvector of A is often called a right eigenvector. Therefore, the eigenvalue λ as well as the right and left eigenvectors, $\mathbf{u}$ and $\mathbf{v}$, satisfy the relations

$$A\mathbf{u} = \lambda\mathbf{u}, \qquad \mathbf{v}^\dagger A = \lambda\mathbf{v}^\dagger,$$

or equivalently,

$$\mathbf{u}^\dagger A^\dagger = \lambda^\star\mathbf{u}^\dagger, \qquad A^\dagger\mathbf{v} = \lambda^\star\mathbf{v}.$$

2.2.3 Types of matrices

The choice of a method for solving linear systems will often depend on the structure of the matrix A. One of the most important properties of matrices is symmetry, because of its impact on the eigenstructure of A. A number of other classes of matrices also have particular eigenstructures. The most important ones are listed below:

- *Symmetric matrices:* $A^T = A$.
- *Hermitian matrices:* $A^\dagger = A$.
- *Skew-symmetric matrices:* $A^T = -A$.
- *Skew-Hermitian matrices:* $A^\dagger = -A$.
- *Normal matrices:* $A^\dagger A = AA^\dagger$.
  A normal matrix is diagonalizable and vice versa.
- *Non-negative matrices:* $a_{ij} \geq 0,\ i, j = 1, \dots, n$ (similar definition for non-positive, positive, and negative matrices).
- *Unitary matrices:* $Q^\dagger Q = I$.

It is worth noting that a unitary matrix Q is a matrix whose inverse is its transpose conjugate $Q^\dagger$, since

$$Q^\dagger Q = I \quad \rightarrow \quad Q^{-1} = Q^\dagger. \tag{2.6}$$

A matrix Q such that $Q^\dagger Q$ is diagonal is often called orthogonal.

Some matrices have particular structures that are often convenient for computational purposes. The following list, though incomplete, gives an idea of these special matrices which play an important role in numerical analysis and scientific computing applications.

- *Diagonal matrices:* $a_{ij} = 0 \quad \text{for} \quad j \neq i$. Notation:

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

- *Upper triangular matrices:* $a_{ij} = 0 \quad \text{for} \quad i > j$.
- *Lower triangular matrices:* $a_{ij} = 0 \quad \text{for} \quad i < j$.
- *Upper bidiagonal matrices:* $a_{ij} = 0 \quad \text{for} \quad j \neq i \text{ or } j \neq i + 1$.
- *Lower bidiagonal matrices:* $a_{ij} = 0 \quad \text{for} \quad j \neq i \text{ or } j \neq i - 1$.
- *Tridiagonal matrices:* $a_{ij} = 0$ for any pair i, j such that $|j - i| > 1$. Notation:

$$A = \text{tridiag}(a_{i,i-1}, a_{ii}, a_{i,i+1}).$$

- *Banded matrices:* $a_{ij} \neq 0$ only if $i - m_\ell \leq j \leq i + m_u$, where m_ℓ and m_u are two non-negative integers. The number $m_\ell + m_u + 1$ is called the bandwidth of A.
- *Upper* HESSENBERG *matrices:* $a_{ij} = 0$ for any pair i, j such that $i > j + 1$. Lower Hessenberg matrices can be defined similarly.
- *Outer product matrices:* $A = \mathbf{u}\mathbf{v}^\dagger$, where both $\mathbf{u}$ and $\mathbf{v}$ are vectors.
- *Permutation matrices:* the columns of A are a permutation of the columns of the identity matrix.
- *Block diagonal matrices:* generalizes the diagonal matrix by replacing each diagonal entry by a matrix. Notation:
$$A = \mathrm{diag}(A_{11}, A_{22}, \ldots, A_{nn}).$$
- *Block tridiagonal matrices:* generalizes the tridiagonal matrix by replacing each nonzero entry by a square matrix. Notation:
$$A = \mathrm{tridiag}(A_{i,i-1}, A_{ii}, A_{i,i+1}).$$

The above properties emphasize structure, i.e., positions of the non-zero elements with respect to the zeros. Also, they assume that there are many zero elements or that the matrix is of low rank. This is in contrast with the classifications listed earlier, such as symmetry or normality.

2.3 Condition of a system of equations

The numerical solution of Eq. (2.2), even when calculated using a direct method, typically deviates from the exact solution, because:

- Rounding errors can occur during the numerical evaluation, and they can accumulate over the course of the algorithm.
- Errors in the input values cause errors in the solution.

If small variations in the input data or in intermediate results cause large variations in the numerical solution, the system is referred to as *ill conditioned* [see Fig. 2.1(b)]. The quality of the numerical solution of Eq. (2.2) depends strongly on the condition of the matrix A. As introduced in Sec. 1.4, the *condition number* offers a practical way to estimate the condition of a function. For linear systems of equations, the condition number can be determined according to the following perturbation analysis [4].

We consider the linear system (2.2) with A an $n \times n$ non-singular matrix. Given any matrix E, the matrix $A(\varepsilon) = A + \varepsilon E$ is non-singular for ε small enough, i.e., for $\varepsilon \leq \alpha$ where α is some small number. We perturb the matrix A by εE and the right hand side $\mathbf{b}$ of Eq. (2.2) by $\varepsilon \mathbf{e}$. The solution $\mathbf{x}(\varepsilon)$ of the perturbed system will then satisfy the equation
$$(A + \varepsilon E)\mathbf{x}(\varepsilon) = \mathbf{b} + \varepsilon \mathbf{e}. \tag{2.7}$$

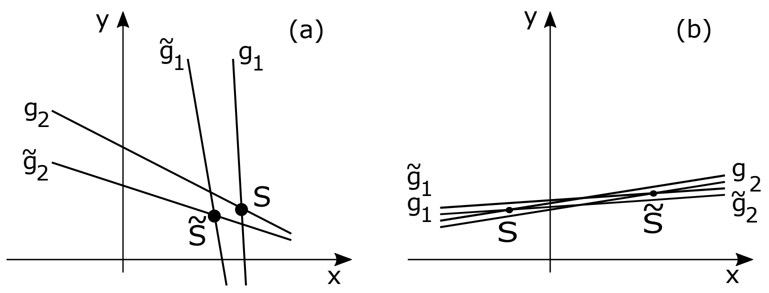

Figure 2.1: Good (a) and ill-conditioned (b) 2x2-System, from [6].

Let $\boldsymbol{\delta}(\varepsilon) = \mathbf{x}(\varepsilon) - \mathbf{x}$, then

$$\begin{aligned}(A+\varepsilon E)\boldsymbol{\delta}(\varepsilon) &= \mathbf{b} + \varepsilon\mathbf{e} - (A+\varepsilon E)\mathbf{x} \\ &= \varepsilon(\mathbf{e} - E\mathbf{x}) \\ \boldsymbol{\delta}(\varepsilon) &= \varepsilon(A+\varepsilon E)^{-1}(\mathbf{e} - E\mathbf{x}).\end{aligned}$$

As an immediate result, the function $\mathbf{x}(\varepsilon)$ is differentiable at $\varepsilon = 0$ and its derivative is given by

$$\mathbf{x}'(0) = \lim_{\varepsilon\to 0} \frac{\boldsymbol{\delta}(\varepsilon)}{\varepsilon} = A^{-1}(\mathbf{e} - E\mathbf{x}). \tag{2.8}$$

The size of the derivative of $\mathbf{x}(\varepsilon)$ is an indication of the variation that the solution $\mathbf{x}(\varepsilon)$ undergoes when the data, the pair $[A, \mathbf{b}]$, is perturbed in the direction $[E, \varepsilon]$. Thus, a small variation $[\varepsilon E, \varepsilon\mathbf{e}]$ will cause the solution to vary by roughly $\varepsilon\mathbf{x}'(0) = \varepsilon A^{-1}(\mathbf{e} - E\mathbf{x})$. The relative variation is given by

$$\frac{||\mathbf{x}(\varepsilon) - \mathbf{x}||}{||\mathbf{x}||} \leq \varepsilon\, ||A^{-1}|| \left(\frac{||\mathbf{e}||}{||\mathbf{x}||} + ||E|| \right) + \sigma(\varepsilon).$$

We use the fact that $||\mathbf{b}|| \leq ||A||\,||\mathbf{x}||$ and transform the above equation into

$$\frac{||\mathbf{x}(\varepsilon) - \mathbf{x}||}{||\mathbf{x}||} \leq \varepsilon\, ||A||\, ||A^{-1}|| \left(\frac{||\mathbf{e}||}{||\mathbf{b}||} + \frac{||E||}{||A||} \right) + \sigma(\varepsilon), \tag{2.9}$$

which relates the relative variation in the solution to the relative sizes of the perturbations. The quantity

$$\kappa(A) = ||A||\, ||A^{-1}||$$

is called the *condition number* of the linear system (2.2) with respect to the norm $||\cdot||$. Thus, the condition number is relative to a norm. When standard norms $||\cdot||_p\,, p = 1, \ldots, \infty$ are used it is customary to label $\kappa(A)$ with the same label as the associated norm. Thus,

$$\kappa(A)_p = ||A||_p\, ||A^{-1}||_p\,.$$

For large matrices, the determinant of a matrix is almost never a good indication of "near" singularity or of the degree of sensitivity of the linear system. The reason is that $\det(A)$ is the product of the eigenvalues which depends very much on a scaling of a matrix, whereas the condition number of a matrix is scaling-invariant. For example, for $A = \alpha I$ the determinant is $\det(A) = \alpha^N$ which can be very small if $|\alpha| < 1$, whereas $\kappa(A) = 1$ for any of the standard norms.

Very often the condition number of HADAMARD is used to determine the condition of matrix A. This condition number is defined as [1]:

$$\kappa(A)_{\mathrm{H}} = \frac{|\det(A)|}{\alpha_1\alpha_2\cdots\alpha_n}, \quad \text{with} \quad \alpha_i = \sqrt{a_{i1}^2 + a_{i2}^2 + \cdots + a_{in}^2}.$$

The following empirical rules hold:

- $\kappa(A)_{\mathrm{H}} < 0.01$ ill-conditioned.
- $\kappa(A)_{\mathrm{H}} > 0.1$ well-conditioned.
- $0.01 \le \kappa(A)_{\mathrm{H}} \le 0.1$ undefined.

2.4 Direct methods

Direct methods are designed to solve a system of equations using a finite number of algebraic operations. The aim of all direct methods is to reduce the original system of n equations,

$$A\boldsymbol{x} = \boldsymbol{b},$$

to an equivalent one of the form

$$U\boldsymbol{x} = \boldsymbol{y}, \tag{2.10}$$

where U is a so-called *upper triangular* matrix with the property

$$U = [u_{ij}] \quad \text{with} \quad u_{ij} = 0 \quad \text{for} \quad i > j.$$

The motivation for such a substitution is easily explained: systems of the type (2.10) are easily solved in n^2 steps through *back-substitution* – see Eqs. (2.17) and (2.20).

2.4.1 The LU decomposition

The LU decomposition [3, 5] is essentially a numerical realization of GAUSSIAN elimination which is based on a fundamental property of linear systems of equations (2.2). This property states the system (2.2) to remain unchanged when a linear combination of rows is added to one particular row. This property is then employed in order to obtain a matrix in *triangular* form. It was demonstrated by DOOLITTLE and CROUT [3, 5, 8] that the GAUSSIAN elimination can be formulated as a decomposition of the matrix A into two matrices L and U:

$$A = LU. \tag{2.11}$$

Here, U is an *upper triangular* matrix and L is a *lower triangular* matrix. In particular, U is of the form

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & & & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}, \tag{2.12}$$

and L is of the form

$$L = \begin{bmatrix} 1 & 0 & & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & & & & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} . \tag{2.13}$$

The factorization (2.11) is referred to as *LU decomposition*. The corresponding procedure can be easily identified by equating the elements in Eq. (2.11). The following algorithm, due to DOOLITTLE, yields the desired result,

$$\begin{aligned} &\text{For } k = 1, 2, \dots, n : \\ &u_{kj} = a_{kj} - \sum_{m=1}^{k-1} l_{km} u_{mj} \quad \text{for } j = k, k+1, \dots, n \\ &l_{kk} = 1 \\ &l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{m=1}^{k-1} l_{im} u_{mk} \right) \quad \text{for } i = k+1, k+2, \dots, n. \end{aligned} \tag{2.14}$$

The first line in the loop of Eq. (2.14) produces the k^{th} row of U, whereas the second and third lines produce the k^{th} column of L. Note that in this notation we used the convention that the contribution of the sum is equal to zero if the upper boundary is less than the lower boundary. We rewrite Eq. (2.2) with the help of the *LU* decomposition (2.11)

$$A\mathbf{x} = LU\mathbf{x} = \mathbf{b}, \tag{2.15}$$

and by defining $\mathbf{y} = U\mathbf{x}$, we retrieve a system of equations for the vector $\mathbf{y}$:

$$L\mathbf{y} = \mathbf{b}. \tag{2.16}$$

The particular form of L allows to solve the system (2.16) immediately by *forward substitution*. We find the solution

$$\begin{aligned} y_1 &= b_1, \\ y_i &= b_i - \sum_{k=1}^{i-1} l_{ik} y_k , \qquad i = 2, \dots, n, \end{aligned} \tag{2.17}$$

and the equation

$$U\mathbf{x} = \mathbf{y}, \tag{2.18}$$

remains. It can be solved by *backward substitution*:

$$x_n = \frac{y_n}{u_{nn}}, \tag{2.19}$$

$$x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{k=i+1}^{n} u_{ik} x_k \right), \quad i = n-1, \dots, 1 . \tag{2.20}$$

Matrix inversion

We note that this method can also be employed to invert the matrix A. The strategy is based on the relation

$$AX = I \; , \tag{2.21}$$

where $X = A^{-1}$ is to be determined and I is the n-dimensional identity. Eq. (2.21) is equivalent to the following system of equations:

$$\begin{aligned} A\mathbf{x}_1 &= \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} , \\ A\mathbf{x}_2 &= \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} , \\ \vdots \quad & \quad \vdots \\ A\mathbf{x}_n &= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} , \end{aligned} \tag{2.22}$$

where the vectors $[\mathbf{x}_i]$ are the rows of the unknown matrix X, i.e. $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^T$. The n equations of the system (2.22) can be solved with the help of the LU decomposition.

Calculation of the determinant

Furthermore, one can easily calculate the determinant of A using the LU decomposition. We note that

$$\det(A) = \det(LU) = \det(L)\det(U) = \det(U) \; , \tag{2.23}$$

since L and U are triangular matrices, the determinants are equal to the product of the diagonal elements, which yields $\det(L) = 1$. Hence we have

$$\det(A) = \det(U) = \prod_{i=1}^{n} u_{ii} \; . \tag{2.24}$$

2.4.2 Partial pivoting

Let us suppose that we have at our disposal a program which solves a non-homogeneous set of equations by LU-decomposition. As this is a direct method, the numerical results obtained through this program are not affected by methodological errors. All possible deviations from the 'true' results must, therefore, come from roundoff errors.

We want to study this in using a concrete problem: we consider the following set of three equations

$$\begin{aligned} x_1 + 5923181x_2 + 1608x_3 &= 5924790 \\ 5923181x_1 + 337116x_2 - 7x_3 &= 6260290 \\ 6114x_1 + 2x_2 + 9101372x_3 &= 9107488, \end{aligned} \tag{2.25}$$

with the exact solution: $x_1 = x_2 = x_3 = 1$.

We study the numerical solution in single precision (*float*) to emphasize the numerical errors. The program generates the following LU-Matrix together with the eigenvector $\mathbf{x}$:

$$LU\text{-Matrix:} \begin{pmatrix} 1.000000e+00 & 5.923181e+06 & 1.608000e+03 \\ 5.923181e+06 & -3.508407e+13 & -9.524475e+09 \\ 6.114000e+03 & 1.032216e-03 & 9.101371e+06 \end{pmatrix}$$

$$\mathbf{x}: \begin{pmatrix} 9.398398e-01 \\ 1.000000e+00 \\ 9.995983e-01 \end{pmatrix}.$$

Obviously, the effect of the rounding error is considerable (in particular on x_1)!

If we now change the <u>order</u> of the equations (2.25) which, in principle, should have no influence on the eigenvector the same program returns:

$$LU\text{-Matrix:} \begin{pmatrix} 5.923181e+06 & 3.371160e+05 & -7.000000e+00 \\ 1.032216e-03 & -3.459764e+02 & 9.101372e+06 \\ 1.688282e-07 & -1.712019e+04 & 1.558172e+11 \end{pmatrix}$$

$$\mathbf{x}: \begin{pmatrix} 9.999961e-01 \\ 1.000068e+00 \\ 1.000000e+00 \end{pmatrix}.$$

This time the rounding errors are already much smaller. If we re-order the lines in the sequence 2/1/3 we obtain the correct eigenvector up to machine precision:

$$LU\text{-Matrix:} \begin{pmatrix} 5.923181e+06 & 3.371160e+05 & -7.000000e+00 \\ 1.688282e-07 & 5.923181e+06 & 1.608000E+03 \\ 1.032216E-03 & -5.841058E-05 & 9.101372E+06 \end{pmatrix}$$

$$\mathbf{x}: \begin{pmatrix} 1.000000e+00 \\ 1.000000e+00 \\ 1.000000e+00 \end{pmatrix}.$$

We conclude that the order in which the equations are set affects the size of the rounding errors and the quality of the numerical solution in a decisive way!

Now, what is the 'correct' order? If we compare the l_{ik} in the LU-matrices shown above, we immediately recognize, that the rounding error is the smallest when the absolute values of the l_{ik} are as small as possible. In order to keep the l_{ik} as small as possible, the absolute values of u_{kk} must become as large as possible. Looking at Eq. (2.14), this can easily be achieved by calculating the u_{kk} with the largest absolute value using the second line in the loop of Eq. (2.14) and exchange the row in which this element occurs with the k-th row. Only after this swap has been performed we evaluate the third line in the loop of Eqs. (2.14). This strategy is called *LU-decomposition with partial pivoting.* In general, especially for higher-order systems, there is no chance to get a correct result without using some kind of pivoting. There are however also sets of equations (see for example [1], page 56) for which the *LU*-decomposition is stable even without pivoting. This occurs if the matrix of coefficients is symmetric, tridiagonal, cyclically tridiagonal, diagonal-dominant, or generally positive definite.

2.4.3 Direct solution of systems with special coefficient matrices

Since direct methods for linear systems are relatively expensive in terms of computational time and storage, many practical applications employ algorithms that are particularly designed to exploit specific properties of the matrix of coefficients, if present. In particular, several specialized direct methods have been designed for matrices commonly encountered in physical applications. Prominent examples include the Cholesky-decomposition method used for symmetric matrices and a special form of *LU* decomposition for tridiagonal matrices. These specialized methods are usually the first choice for matrices of such a specific form because they are much faster and more stable than methods developed for matrices of more general form. Since a full treatment of these methods is beyond the scope of this lecture, we refer the interested reader to books on numerical linear algebra, for instance Refs. [5, 8]. Here, we will only treat a solution method for tridiagonal matrices.

Tridiagonal coefficient matrix

Many important numerical methods (for example the spline interpolation) lead to linear sets of equations of the form

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & c_{n-1} \\ 0 & 0 & 0 & 0 & \dots & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ \vdots \\ r_n \end{pmatrix}, \tag{2.26}$$

where the *tridiagonal* matrix of coefficients can clearly be rewritten in terms of the three vectors **b** (main diagonal), **a**, and **c** (lower and upper secondary diagonal). In this case, if we use the *LU*-decomposition (without pivoting), we obtain the following structure:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ m_2 & 1 & 0 & \dots 0 & \\ 0 & m_3 & 1 & \dots 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \underbrace{\begin{pmatrix} u_1 & c_1 & 0 & \dots & 0 \\ 0 & u_2 & c_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \dots & c_{n-1} \\ 0 & 0 & 0 & \dots & u_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{pmatrix}}_{\equiv \mathbf{y}} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ \vdots \\ r_n \end{pmatrix},$$

which results the following simple expressions, through Eqs. (2.14) to (2.20):

$$\begin{aligned} u_1 &= b_1, \\ y_1 &= r_1, \\ m_j &= a_j/u_{j-1}, \\ u_j &= b_j - m_j c_{j-1}, \quad j = 2, \ldots, n \\ y_j &= r_j - m_j y_{j-1}. \end{aligned} \tag{2.27}$$

The components of the solution will then be obtained through a back substitution:

$$\begin{aligned} x_n &= y_n/u_n \\ x_j &= (y_j - c_j x_{j+1})/u_j, \quad j = n-1, \ldots, 1. \end{aligned} \tag{2.28}$$

2.4.4 Iterative improvement

It is clear that the accuracy of the numerical solution of a linear set of equations is limited by the machine precision. Unfortunately, we cannot obtain this level of precision in many cases due to the accumulation of rounding errors. But there is still the possibility of iteratively improving the numerical solution. What follows will briefly explain the (very simple) theory underlying this idea. Let us suppose that the numerical solution of the system:

$$A\mathbf{x} = \mathbf{b}$$

is the vector $\bar{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x}$ which is affected by a rounding error. If we now re-insert $\bar{\mathbf{x}}$ in this system of equations, we obtain a different non-homogeneous vector $\mathbf{b} + \delta\mathbf{b}$, due to the rounding error

$$A\bar{\mathbf{x}} = A(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b},$$

and

$$\underbrace{(A\mathbf{x} - \mathbf{b})}_{=0} + A\delta\mathbf{x} = \delta\mathbf{b}.$$

Thus, we have an error vector $\delta\mathbf{x}$ which is the solution of the system

$$A\delta\mathbf{x} = \delta\mathbf{b},$$

where A is the original matrix of coefficients and $\delta\mathbf{b}$ is the so-called *residual vector* $A\bar{\mathbf{x}} - \mathbf{b}$.

Finally, the improved solution $\mathbf{x}$ is found from

$$\mathbf{x} = \bar{\mathbf{x}} - \delta\mathbf{x},$$

and this allows to define a condition to terminate the iteration after some lower boundary on $\delta\mathbf{x}$ has been reached.

A few remarks:

1. Many *LU* decomposition programs destroy the matrix A on return. Thus, it is important to store the corresponding information somewhere before executing the *LU* decomposition algorithm, because, most likely, the matrix A is going to be required in the subsequent iterative improvement.

2. Due to the big risk of a subtractive cancellation the calculation of the residual vector should be performed in double precision, if at all possible.

3. Some software libraries (for example NAG) contain programs for the solution of non-homogeneous linear equations which already include a method for iterative improvement.

4. In ill-conditioned and/or singular systems the linear system of equations can still be numerically satisfied although the solution deviates considerably from the exact solution. In such cases re-iteration doesn't improve the solution. This is one of the most important problems in the use of numerical methods in linear systems. There is no easy, general way to avoid that the components of the solution are affected by rounding errors.

5. Another unpleasant consequence of the rounding error is that in a numerical calculation it is practically impossible to discriminate between singular or almost singular problems. In the program a matrix is considered singular if all the elements of a pivot column are exactly zero. However, since these elements themselves are produced by arithmetic operations, we could have elements that are not exactly zero due to rounding errors, even if the matrix of coefficients is singular. For this reason many programs avoid giving a definition of singularity and leave the choice to the user.

2.5 Iterative methods

2.5.1 The Gauss-Seidel method

The Gauss-Seidel method is an iterative procedure to approximate the solution of non-homogeneous systems of linear equations [3, 2]. The advantage of an iterative procedure, in contrast to a direct approach, is that its formulation is in general much simpler. However, one might have problems with the convergence of the method, even in cases where a solution exists and is unique. We note that the Gauss-Seidel method is of particular interest whenever one has to deal with *sparse* coefficient matrices.[1] This requirement is not too restrictive since most of the matrices encountered in physical applications are indeed sparse.

Again, we use Eq. (2.1) as a starting point for our discussion. It is a requirement of the Gauss-Seidel method that *all* diagonal elements of A are non-zero. We then solve each row of (2.1) for x_i. This creates the following hierarchy

$$\begin{aligned} x_1 &= -\frac{1}{a_{11}}\left(a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n - b_1\right), \\ x_2 &= -\frac{1}{a_{22}}\left(a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n - b_2\right), \\ \vdots \quad & \quad \vdots \\ x_n &= -\frac{1}{a_{nn}}\left(a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1} - b_n\right), \end{aligned} \tag{2.29}$$

[1] A matrix A is referred to as sparse, when the matrix is populated primarily by zeros, see Ref. [4].

or in general for $i = 1, \ldots, n$

$$x_i = -\frac{1}{a_{ii}} \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j - b_i \right). \tag{2.30}$$

We note that Eq. (2.30) can be rewritten as a matrix equation

$$\mathbf{x} = C\mathbf{x} + \mathbf{b}', \tag{2.31}$$

where we defined the matrix $C = [c_{ij}]$ via

$$c_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}} & i \neq j, \\ 0 & i = j, \end{cases} \tag{2.32}$$

and the vector $\mathbf{b}' = [b_i']$ as

$$b_i' = \frac{b_i}{a_{ii}}. \tag{2.33}$$

We recognize that Eq. (2.30) can be transformed into an iterative form with the help of a trivial manipulation

$$x_i = x_i - \left[x_i + \frac{1}{a_{ii}} \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j - b_i \right) \right], \tag{2.34}$$

or

$$x_i^{(t+1)} = x_i^{(t)} - \Delta x_i^{(t)}, \tag{2.35}$$

where

$$\Delta x_i^{(t)} = x_i^{(t)} + \frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(t+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(t)} - b_i \right), \tag{2.36}$$

and t is an iteration index. Eq. (2.35) in combination (2.36) produces a sequence of vectors

$$\mathbf{x}^{(0)} \to \mathbf{x}^{(1)} \to \mathbf{x}^{(2)} \to \ldots \to \mathbf{x}^{(m)}, \tag{2.37}$$

where $\mathbf{x}^{(0)}$ is referred to as the initialization vector or trial vector. One can prove that if this sequence converges, it approaches the exact solution $\mathbf{x}$ arbitrarily close:

$$\lim_{t \to \infty} \mathbf{x}^{(t)} = \mathbf{x}. \tag{2.38}$$

We remark that if the terms $x_i^{(t+1)}$ on the right hand side of Eq. (2.36) are replaced by $x_i^{(t)}$ the method is referred to as the JACOBI method.

To terminate the GAUSS-SEIDEL method, we need an exit condition: One should terminate the iteration whenever:

- The approximate solution $x^{(t)}$ obeys the required accuracy ε or $\tilde{\varepsilon}$, for instance

$$\max\left(\left|x_i^{(t)} - x_i^{(t-1)}\right|\right) \leq \varepsilon, \tag{2.39}$$

  where ε is the absolute error, or

$$\max\left(\frac{\left|x_i^{(t)} - x_i^{(t-1)}\right|}{\left|x_i^{(t)}\right|}\right) \leq \tilde{\varepsilon}, \tag{2.40}$$

  where $\tilde{\varepsilon}$ is the relative error.

- When a maximum number of iterations is reached. This condition may be interpreted as an emergency exit which ensures that the iteration terminates even if the process is not convergent or has still not converged.

Let us discuss one final, however, crucial point of this section: In many cases the convergence of the GAUSS-SEIDEL method can be significantly improved by including a *relaxation parameter* ω to the iterative process. In this case the update routine (2.35) takes on the form

$$x_i^{(t+1)} = x_i^{(t)} - \omega \Delta x_i^{(t)}. \tag{2.41}$$

If the relaxation parameter ω obeys $\omega > 1$ one speaks of *over-relaxation*, if $\omega < 1$ of *under-relaxation* and if $\omega = 1$ the regular GAUSS-SEIDEL method is recovered. An appropriate choice of the relaxation parameter may fasten the convergence of the method significantly. The best result will certainly be obtained if the *ideal* value of ω, ω_i were known. Unfortunately, it is impossible to determine ω_i prior to the iteration in a general case. We remark the following properties:

- The method (2.41) is only convergent for $0 < \omega \leq 2$.

- If the matrix C is positive definite and $0 < \omega < 2$, the GAUSS-SEIDEL method converges for any choice of $x^{(0)}$ (OSTROWSKI-REICH theorem, [7]).

- In many cases, $1 \leq \omega \leq 2$. We note that this inequality holds only under particular restrictions for the matrix C [see Eq. (2.32)]. However, we note without going into detail, that these restrictions are almost always fulfilled when one is confronted with applications in physics.

- If C is positive definite and tridiagonal, the ideal value ω can be calculated using

$$\omega = \frac{2}{1 + \sqrt{1 - \lambda^2}}, \tag{2.42}$$

  where λ is the largest eigenvalue of C, Eq. (2.32).

- Since the calculation of λ is in many cases quite complex, one could employ the following idea: It is possible to prove that

$$\lim_{t\to\infty} \frac{\left|\Delta x^{(t+1)}\right|}{\left|\Delta x^{(t)}\right|} \to \lambda^2. \tag{2.43}$$

Hence, one may start with $\omega = 1$, perform t_0 $(20 < t_0 < 100)$ iterations and then approximate ω_i with the help of Eq. (2.43) and

$$\lambda^2 \approx \frac{\left|\Delta x^{(t_0)}\right|}{\left|\Delta x^{(t_0-1)}\right|}. \tag{2.44}$$

The iteration is then continued with the approximated value of ω_i until convergence is reached.

We have to be careful not to set a too low value for ε or $\tilde{\varepsilon}$. Indeed, if the problem is ill-conditioned, it can happen that we never reach the desired precision due to rounding errors! Nevertheless, one can state that iterative methods exhibit smaller rounding errors in comparison to direct methods. In fact, iterative methods have the desirable property that only the roundoff error of the last iteration affects the results: there is no accumulation of roundoff errors by successive iterations! For this reason, in some cases the GAUSS-SEIDEL method is preferable to direct methods, if one wants to attain a higher precision.

In conclusion we remark that numerous numerical libraries contain sophisticated routines to solve linear systems of equations. In many cases it is, thus, advisable to rely on such routines, in particular because they also provide various routines developed for specific matrices.

Bibliography

[1] Engeln-Müllges, G., Reutter, F.: Formelsammlung zur Numerischen Mathematik mit Standard-FORTRAN-Programmen. BI Mannheim, Mannheim (1984)

[2] Hazewinkel, M. (ed.): Encyclopaedia of Mathematics. Springer, Berlin, Heidelberg (1994)

[3] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[4] Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2003)

[5] Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, 2nd edn. Springer, Berlin, Heidelberg (1993)

[6] Überhuber, C.: Computer Numeric 2. Springer, Berlin, Heidelberg (1995)

[7] Varga, R.S.: Matrix Iterative Analysis, 2nd edn. Springer Series in Computational Mathematics, Vol. 27. Springer, Berlin, Heidelberg (2000)

[8] Westlake, J.R.: A Handbook of Numerical Matrix Inversion and Solution of Linear Equations. Wiley, New York (1968)

Chapter 3

Interpolation and FOURIER transform of point sets

3.1 The problem

Let us assume that we know the coordinates x and y of n data points. The correlation between the coordinates is unique i.e. for every value of x exists one and only one value of y. We also assume the values x to be ordered by increasing magnitude, although they do not have to be equidistant.

There is one further, very important requirement for the point set which is difficult to express mathematically: the sampling points $(x_i|y_i), i = 1, \dots, n$ should express a functional correlation between x and y in a way which is satisfying "phenomenologically". Fig. 3.1 illustrates the significance of this sentence.

Moreover, let us assume a given point set to be representative in an interval $[x_1, x_n]$. We can then define an *interpolation function* $I(x)$ as follows:

- $I(x)$ should exactly pass through all sampling points, or:

$$I(x_i) = y_i, \quad \text{for} \quad i = 1 \dots, n. \tag{3.1}$$

- $I(x)$ should connect the sampling points *as smoothly as possible.*

The following paragraph is cited from [6], p.127: The strategy to construct an interpolation function is to take a finite set of functions $\varphi_k(x)$ as a basis. These basis functions are linearly independent within the interval $\mathcal{I} = [x_1, x_n]$. The interpolation function is then defined as a linear combination of these functions. We thus have:

$$I(x) = \sum_{k=1}^{n} c_k \varphi_k(x), \tag{3.2}$$

with c_k some general coefficient. Typical expansion functions for $I(x)$ are:

- Polynomials: 1; x; x^2; x^3; ....
- Exponential functions: 1; $e^{\pm iax}$; $e^{\pm 2iax}$; $e^{\pm 3iax}$; ....
- Rational functions:

$$I(x) = \frac{p_0 + p_1 x + \dots + p_\mu x^\mu}{q_0 + q_1 x + \dots + q_\nu x^\nu},$$

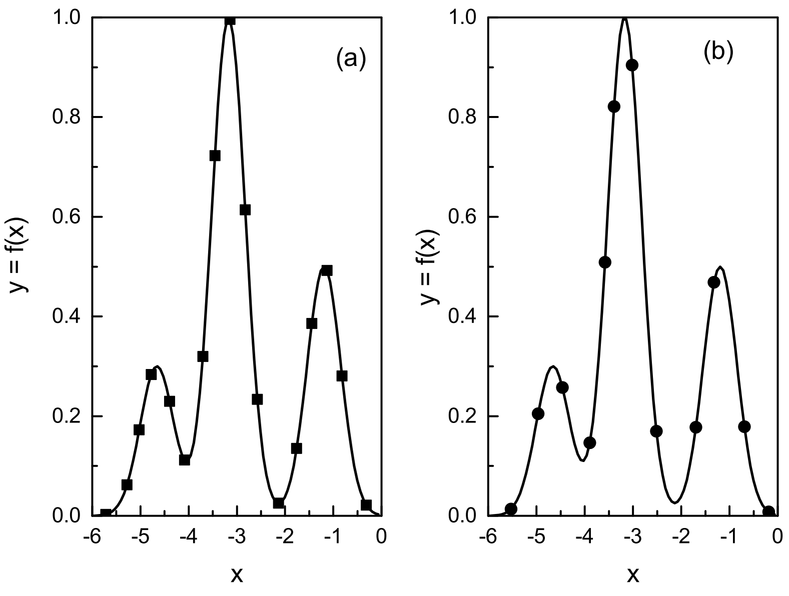

Figure 3.1: (a) The point set (solid squares) is representative and (b) the point set (solid circles) is not representative.

with $\nu + \mu + 1 = n$, the number of sampling points (q_0 is optional!) within the Interval $\mathcal{I}$. See Ref. [7], p. 111ff.

- Multidimensional interpolation. See Ref. [7], p. 123ff.

3.2 Polynomial interpolation

A 'standard' interpolation method is the *polynomial interpolation*. It uses an $(n-1)$ degree polynomial with general coefficients c_k as an ansatz:

$$I(x) = \sum_{k=1}^{n} c_k x^{k-1}. \tag{3.3}$$

The n coefficients c_k are determined by inserting Eq. (3.3) into Eq. (3.1) which results in:

$$I(x_i) = \sum_{k=1}^{n} c_k x_i^{k-1} = y_i \quad \text{for} \quad i = 1, \ldots, n. \tag{3.4}$$

This is a set of linear inhomogeneous equations for the coefficients $c_1, \ldots, c_n$ which is of the form:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \tag{3.5}$$

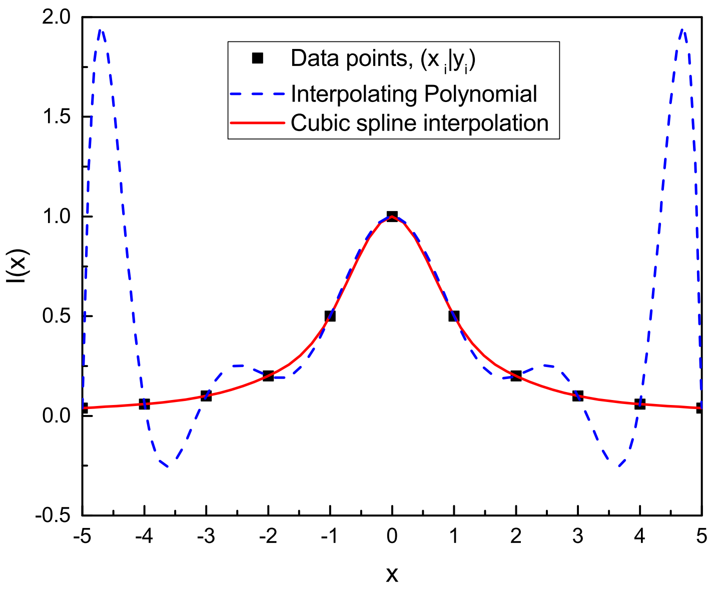

Figure 3.2: Polynomial interpolation (dashed blue line) and piece-wise spline interpolation (solid red line) of the function (3.6)

.

The matrix on the left hand side of this equation is a VANDERMODE *matrix* (see Ref. [7]). The solution of such a system of equations is unique and there exists only one polynomial of degree $n - 1$ which passes exactly through all sampling points. Thus, the requirement (3.1) has been satisfied. Nevertheless, it has to be emphasized that VANDERMODE systems can be quite ill-conditioned.[1] In such a case, *no* numerical method is going to give an accurate answer for the coefficients c_k.

To give an example, we study the function

$$y(x) = \frac{1}{1 + x^2} \tag{3.6}$$

and use eleven data points (black solid squares in Fig. 3.2) at the arguments $x = -5, -4, \ldots, 5$. If we take a tenth degree interpolation polynomial through these sampling points, we obtain the interpolation function $I(x)$ shown as a dashed blue curve in Fig. 3.2.

The problem becomes evident: the polynomial interpolation describes the function $y(x)$ properly only around $x \approx 0$. Sizeable methodological errors occur at the edges of the interpolation interval. If the degree of the interpolating polynomial is too high, compared to the smoothness of the interpolated data set/function, the interpolating polynomial will attempt to use its high-degree coefficients, in large and almost precisely canceling combinations in an attempt to match the tabulated values down to the last possible error margin of accuracy. In general, high degree interpolating polynomials trend to oscillate (often widely) between its constraining

[1]The problem of ill-conditioned matrices is discussed in Subsec. 2.3.

points and these oscillations would be present even if the machine's floating precision were infinitely good.

3.2.1 Piece-wise interpolation with cubic splines

The basic idea of this method is to split the original interval $\mathcal{I}$ into $n-1$ subintervals $\mathcal{I}_1 = [x_1, x_2], \mathcal{I}_2 = [x_2, x_3], \ldots, \mathcal{I}_{n-1} = [x_{n-1}, x_n]$ and to interpolate the data set within these intervals by low degree interpolating polynomials.

In particular, the starting point for the spline interpolation are third degree polynomials

$$P^i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \tag{3.7}$$

in which the index i indicates that we consider the polynomial only in the interval $[x_i, x_{i+1}]$. The interpolation curve $I(x)$ (in case of n sampling points) consists of $n-1$ polynomial pieces:

$$I(x) = \begin{cases} P^1(x) & \text{for} \quad x \in [x_1, x_2] \\ P^2(x) & \text{for} \quad x \in [x_2, x_3] \\ \vdots & \vdots \quad \vdots \\ P^{n-1}(x) & \text{for} \quad x \in [x_{n-1}, x_n]. \end{cases} \tag{3.8}$$

The overall $4(n-1)$ polynomial coefficients a_i, b_i, c_i, and d_i are determined in such a way that:

- $P^i(x)$ passes through the sampling points y_i and y_{i+1} .
- $P^i(x)$ has the same *first and second derivative* at point x_{i+1} as the next polynomial $P^{i+1}(x)$ on the right.

The first of these two conditions defines an interpolation function $I(x)$, the second transforms $I(x)$ into a *cubic spline interpolation.*

In principle we can use polynomials of higher order to make a spline interpolation. But since cubic splines are most frequently used, we simply concentrate on this particular case.

Thus, we get for cubic splines the following conditions:

$$\begin{aligned} P^i(x_i) &= y_i \\ P^i(x_{i+1}) &= y_{i+1}, \quad \text{for} \quad i = 1, \ldots, n-1 \\ \vdots &= \vdots \\ \frac{\mathrm{d}}{\mathrm{d}x} P^i(x_{i+1}) &= \frac{\mathrm{d}}{\mathrm{d}x} P^{i+1}(x_{i+1}) \\ \frac{\mathrm{d}^2}{\mathrm{d}x^2} P^i(x_{i+1}) &= \frac{\mathrm{d}^2}{\mathrm{d}x^2} P^{i+1}(x_{i+1}) \quad \text{for} \quad i = 1, \ldots, n-2 \end{aligned} \tag{3.9}$$

These are $4n-6$ equations for $4n-4$ polynomial coefficients. Thus, two more equations are required to solve the system. These two additional conditions are in principle arbitrary. A very common choice is to require that the second derivatives of the interpolation function at the beginning and the end of the interpolation interval vanish:

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} P^1(x_1) = 0, \quad \text{and} \quad \frac{\mathrm{d}^2}{\mathrm{d}x^2} P^{n-1}(x_n) = 0. \tag{3.10}$$

The interpolation curve we obtain in this case is continuous in the functional value as well as in the first and second derivatives and ensures a smooth connection of the sampling points. In the literature splines with the additional conditions (3.10) are called *natural cubic splines*. In many cases it is desirable to adapt these two additional conditions to accomodate better the actual problem. We shortly describe another form of additional conditions which is preferred in literature (e.g. Ref. [9], p. 450f) over natural splines: the cubic spline function with *not-a-knot* condition. We impose:

$$P^1(x) = P^2(x), \quad \text{for} \quad x_1 \le x \le x_3, \tag{3.11}$$

and

$$P^{n-2}(x) = P^{n-1}(x), \quad \text{for} \quad x_{n-2} \le x \le x_n, \tag{3.12}$$

This means that the first and second sub-interval are approximated by the same polynomial. The same condition is imposed for the second to last and the last sub-interval. Consequently, x_2 and x_{n-1} are no longer 'true' knots of the spline function ('not-a-knot').

In principle, one could determine all spline coefficients for natural cubic splines from the set of equations (3.9) and (3.10). In practice a different approach is used which is based on the following equations:

$$a_i = y_i, \quad i = 1, \dots, n; \quad c_1 = c_n = 0, \tag{3.13}$$

$$h_{i-1}c_{i-1} + 2c_i(h_{i-1} + h_i) + h_i c_{i+1} = \frac{3r_i}{h_i} - \frac{3r_{i-1}}{h_{i-1}}, \quad i = 2, \dots, n-1. \tag{3.14}$$

Here $h_i = x_{i+1} - x_i$ and $r_i = y_{i+1} - y_i$. These equations follow after a rather easy but long calculation from the Eqs. (3.9) and (3.10) which we will skip here.

Eq. (3.14) is a set of linear non-homogeneous equations of degree $n-1$ for the unknowns $c_2, \dots, c_{n-1}$. The matrix of coefficients is tridiagonal and symmetric. With the knowledge of a_i and c_i the *conditional equations* for the remaining spline coefficients result in:

$$\begin{aligned} b_i &= \frac{r_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i), \\ d_i &= \frac{1}{3h_i}(c_{i+1} - c_i), \quad i = 1, \dots, n-1. \end{aligned} \tag{3.15}$$

The effectiveness of interpolation with cubic splines, and in particular for the smoothness of the interpolation curve, can be seen in Fig. 3.2 (solid red curve).

3.3 Fourier analysis of discrete data

In contrast to the previous section we will now use exponential basis functions as expansion functions for the interpolation function $I(x)$. To accomplish this, let us assume a set of n *uniformly* distributed sampling points

$$(x_j = x_0 + j\Delta | y_j), \quad j = 0, 1, \dots, n-1, \tag{3.16}$$

to exist in which the x_i are real numbers and the y_i are (in general) complex. Δ is the distance between two adjacent x-coordinates x_i and x_{i+1}. (See Fig. 3.3.) To simplify the required formalism, we assume $x_0 = 0$ in what follows. This set of

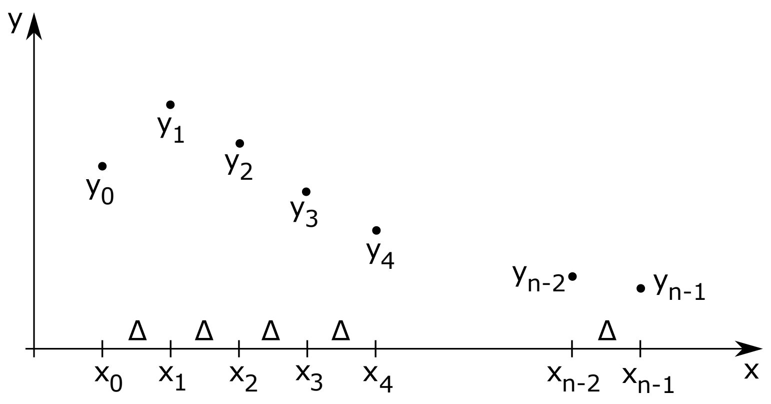

Figure 3.3: The n uniformly distributed sampling points $(x_i|y_i)$ according to Eq. (3.16).

sampling points is to be interpolated by exponential basis functions, described in Sec. 3.1:

$$y(x) = \frac{1}{n}\sum_{k=0}^{n-1} Y_k e^{-i\alpha kx}, \tag{3.17}$$

with the FOURIER coefficients Y_k and a so far undetermined constant α. We assume, furthermore, the n given sampling points to belong to a *periodic* function with period[2]

$$P = n\Delta. \tag{3.18}$$

Consequently, the interpolation function $I(x)$ must also be periodic in P, i.e.:

$$I(x+P) = I(x). \tag{3.19}$$

We can easily show that this condition is equivalent to a special choice of the constants α, which we left unspecified so far:

$$\frac{1}{n}\sum_{k=0}^{n-1} Y_k e^{-i\alpha k(x+P)} = \frac{1}{n}\sum_{k=0}^{n-1} Y_k e^{-i\alpha kx}, \tag{3.20}$$

which is satisfied if

$$e^{i\alpha kP} = e^{i\alpha kn\Delta} = 1, \tag{3.21}$$

or

$$\alpha = \frac{2\pi}{n\Delta}. \tag{3.22}$$

This results in the following expression for the interpolation function

$$I(x) = \frac{1}{n}\sum_{k=0}^{n-1} Y_k \exp\left(-i\frac{2\pi kx}{n\Delta}\right), \tag{3.23}$$

and the interpolation condition (3.1) follows with

$$I(x_j) = \frac{1}{n}\sum_{k=0}^{n-1} Y_k \exp\left(-i\frac{2\pi kj}{n}\right) = y(x_j), \quad \text{for} \quad j = 0,\dots,n-1. \tag{3.24}$$

[2]It does not make a difference if the function values to be interpolated *really* describe a periodic function!

The FOURIER coefficients Y_k are in general complex numbers.

Of course we could determine Y_k directly from the linear system of equations (3.24) but in the case of a FOURIER expansion this task is greatly simplified. We multiply Eq. (3.24) with $\exp(i2\pi k'j/n)$, sum over the index j, and obtain:

$$\sum_{j=0}^{n-1} y_j \exp\left(i\frac{2\pi k'j}{n}\right) = \frac{1}{n}\sum_{k=0}^{n-1} Y_k \underbrace{\sum_{j=0}^{n-1} \exp\left[i\frac{2\pi j}{n}(k'-k)\right]}_{(A)} . \tag{3.25}$$

The expression (A) has in case of *equidistant* sampling points the simple solution:

$$(A) = n\delta_{k,k'}, \quad \text{with} \quad \delta_{k,k'} = \begin{cases} 0 & \text{for} \quad k \neq k' \\ 1 & \text{for} \quad k = k' \end{cases} . \tag{3.26}$$

The symbol $\delta_{k,k'}$ is commonly referred to as the KRONECKER *delta*. The reason for the result (3.26) is that the functions of a FOURIER expansion for a set of equidistant data points form a complete set of orthogonal functions. Thus, Eq. (3.25) together with (3.26) results in:

$$\sum_{j=0}^{n-1} y_j \exp\left(i\frac{2\pi k'j}{n}\right) = \frac{1}{n} Y_k n \delta_{k,k'} = Y_{k'}, \tag{3.27}$$

and we obtain the FOURIER coefficients Y_k from the simple conditional equation

$$Y_k = \sum_{j=0}^{n-1} y_j \exp\left(i\frac{2\pi kj}{n}\right), \quad k = 0, \dots, n-1. \tag{3.28}$$

Eq. (3.28) is commonly referred to as the *Discrete* FOURIER *Transform* (DFT) of y_i because the Y_i are solely determined by the y_i. We rewrite Eq. (3.24) as

$$y_j = \frac{1}{n}\sum_{k=0}^{n-1} Y_k \exp\left(-i\frac{2\pi kj}{n}\right), \quad j = 0, \dots, n-1, \tag{3.29}$$

which is then referred to as the *Inverse Discrete* FOURIER *Transform* (IDFT).

Since the basis functions $\exp(i2\pi kj/n)$ of the FOURIER expansion are complex, the coefficients Y_k are also in general complex. An important special case occurs if the values y_j are real. In this case the FOURIER coefficients have the following property:

$$Y_k = Y^\star_{n-k}, \tag{3.30}$$

where the symbol $\star$ indicates the conjugate complex.

3.3.1 Numerical calculation of the FOURIER coefficients

The numerical evaluation of Eq. (3.28) is a n^2 - order method, i.e. we have to calculate n terms of the sum for each of the n coefficients. For large n this implies a time-intensive calculation.

In terms of computational time the so-called *Fast*-FOURIER *transform* (FFT), based on a recursive calculation of the sum in Eq. (3.28), represents a significant improvement. This algorithm, based on the results of a paper by Danielson and Lanczos (1942) (see also Ref. [5]), is simple in principle, though not so easy to program. Therefore, we illustrate it in principle, using an example with $n = 8$ sampling points. If we split the sum in Eq. (3.28) into two sub-sums, namely:

$$Y_k = \sum_{j=0}^{7} y_j \exp\left(i\frac{2\pi kj}{8}\right) = \sum_{j=0(2)}^{6} y_j \exp\left(i\frac{2\pi kj}{8}\right) + \sum_{j=1(2)}^{7} y_j \exp\left(i\frac{2\pi kj}{8}\right).$$

Here (2) indicates that the index j of the sums is incremented by 2 after each step of the sum. A transformation of the indices of the sum results in the expression:

$$Y_k = \sum_{j=0}^{3} y_{2j} \exp\left(i\frac{2\pi kj}{4}\right) + \exp\left(i\frac{2\pi kj}{8}\right) \sum_{j=0}^{3} y_{2j+1} \exp\left(i\frac{2\pi kj}{4}\right).$$

The abbreviation $W_m^{j,k} = \exp(i2\pi kj/m)$ transforms the above equation into:

$$Y_k = \sum_{j=0}^{7} W_8^{j,k} y_j = \underbrace{\sum_{j=0}^{3} W_4^{j,k} y_{2j}}_{Y_k^0} + W_8^{1,k} \underbrace{\sum_{j=0}^{3} W_4^{j,k} y_{2j+1}}_{Y_k^1}.$$

We can, again, split the two sub-sums Y_k^0 and Y_k^1:

$$\begin{aligned} Y_k^0 &= \underbrace{\sum_{j=0}^{1} W_2^{j,k} y_{4j}}_{Y_k^{00}} + W_4^{1,k} \underbrace{\sum_{j=0}^{1} W_2^{j,k} y_{4j+2}}_{Y_k^{01}} \\ Y_k^1 &= \underbrace{\sum_{j=0}^{1} W_2^{j,k} y_{4j+1}}_{Y_k^{10}} + W_4^{1,k} \underbrace{\sum_{j=0}^{1} W_2^{j,k} y_{4j+3}}_{Y_k^{11}}. \end{aligned}$$

The last splitting of the four sums Y_k^{00}, Y_k^{01}, Y_k^{10}, and Y_k^{11} gives finally:

$$\begin{aligned} Y_k^{00} &= y_0 + W_2^{1,k} y_4, \qquad & Y_k^{01} &= y_2 + W_2^{1,k} y_6, \\ Y_k^{10} &= y_1 + W_2^{1,k} y_5, \qquad & Y_k^{11} &= y_3 + W_2^{1,k} y_7. \end{aligned}$$

Thus, we ‘transformed the Fourier coefficients in the plane of values y’ and we can now, step-by-step, construct the desired coefficient Y_k, as is illustrated schematically in Fig. 3.4. This approach works, obviously, only if the sampling points considered to construct the coefficients in Fig. 3.4 can be doubled with each step. Thus the number of sampling points n has to be a power of two. (Modern FFT codes do no longer enforce this restriction.)

It is also apparent from Fig. 3.4 that the values y_j ($j = 0, \dots, n-1$) at the beginning of the cascade (the left hand side of Fig. 3.4) must be ordered ‘properly’. This particular order of indices can be obtained by *reversing* the bits of the original indexes in their binary notation (see Tab. 3.1).

The huge advantage of FFT compared to the direct evaluation of Eq. (3.28) is the considerable reduction in computing time. While for the calculation of the Fourier

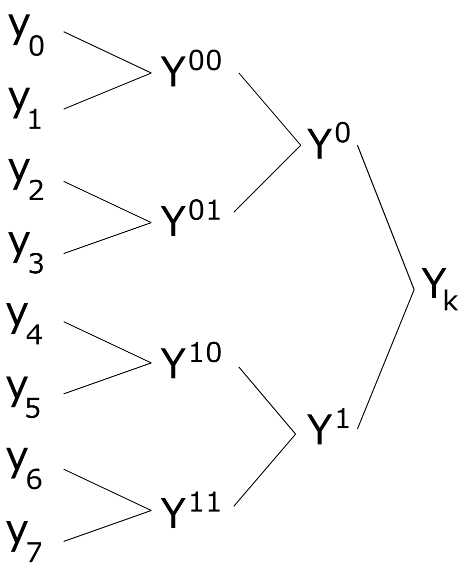

Figure 3.4: Basic principle of the 'Fast FOURIER Transform'.

Table 3.1: Reordering of the y_j indices by bit reversal.

primary order (index)	binary notation	BIT REVERSAL	new order (index)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

coefficients using Eq. (3.28) n^2 terms are required in the sum, this number reduces to $n\ln(n)$ when FFT is employed. This is documented in Tab. 3.2. Consequently, FFT is the only possible way to handle Fourier transforms on a PC!

A very good FFT code can be found in Ref. [7]. The probably fastest FFT code can be downloaded from Ref. [2].

3.3.2 Application of DFT to solve convolution integrals

This application certainly goes beyond a pure interpolation application. Figure 3.5 shows schematically a typical experimental setup. The sample emits a signal $s(t)$ (input signal) which is processed by the instrument. The result of this process is the signal $m(t)$ (output signal) which will then be registered in the recorder. For the analysis of the recorded data it is in general very important to eliminate the instrument's influence on the input signal as much as possible. The *correlation* between the input and output signal is assumed to be described by the convolution

Table 3.2: Number of steps required to calculate the FOURIER coefficients.

number of data points	DFT	FFT
n	n^2	$nln(n)$
128	16384	896
1024	1048576	10240
$\vdots$	$\vdots$	$\vdots$
1048576	$\approx 10^{12}$	$\approx 2.1 \cdot 10^7$

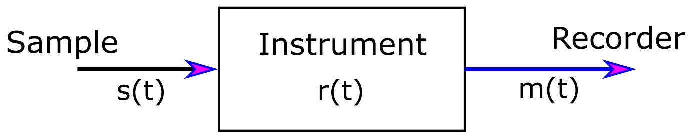

Figure 3.5: A typical experimental setup.

integral

$$m(t) = \int_{-\infty}^{\infty} \mathrm{d}t' \, r(t') s(t - t'), \tag{3.31}$$

where $r(t)$ defines the instrument's influence on the original signal. $r(t)$ is very often called the *device* or *resolution* function.

The functions $m(t)$ and $s(t)$ are now assumed to be periodic with period P and the device function is assumed to be bounded on this period, i.e.:

$$r(t) = 0 \quad \text{outside} \quad [0, P].$$

Under these conditions the range of integration in Eq. (3.31) can be limited to $t' \in [0, P]$:

$$m(t) = \int_0^P \mathrm{d}t' \, r(t') s(t - t').$$

This integral can be approximated with the help of the trapezoidal formula (see Sec. 8.3). We assume now the t-coordinates of the data points to be equidistant within the interval $[0, P]$ and introduce

$$t_j = j\Delta, \quad \Delta = \frac{P}{n}, \quad j = 0, \ldots, n - 1,$$

with n the number of data points within the interval. Thus, we obtain

$$m(t) \approx \Delta \sum_{j=0}^{n-1} r(t_j) s(t - t_j).$$

If we are interested in the particular argument

$$t_\ell = \ell\Delta$$

then

$$m(t_\ell) \approx \Delta \sum_{j=0}^{n-1} r(t_j) s(t_\ell - t_j),$$

or in a more compact notation:

$$m_\ell \approx \Delta \sum_{j=0}^{n-1} r_j s_{\ell-j}. \tag{3.32}$$

Note that the above sum contains negative indices for s. This is, however, not a problem, since

$$s_{-i} = s_{n-i},$$

because of the assumed periodicity of $r(t)$. Now DFT comes into play: we apply the FOURIER transform defined in Eq. (3.24) to the three values m, r, and s as they appear in Eq. (3.32) and obtain:

$$\begin{aligned}
\frac{1}{n}\sum_{k=0}^{n-1} M_k e^{-i2\pi k\ell/n} &\approx \Delta \sum_{j=0}^{n-1} \frac{1}{n}\sum_{k=0}^{n-1} S_k e^{-i2\pi k(\ell-j)/n} \frac{1}{n}\sum_{k'=0}^{n-1} R_{k'} e^{-i2\pi k' j/n} \\
&= \frac{\Delta}{n^2}\sum_{k=0}^{n-1}\sum_{k'=0}^{n-1} S_k R_{k'} e^{-2\pi k\ell/n} \underbrace{\sum_{j=0}^{n-1} e^{i2\pi(k-k')j/n}}_{n\delta_{k,k'}} \\
&= \frac{\Delta}{n}\sum_{k=0}^{n-1} S_k R_k e^{-i2\pi k\ell/n}.
\end{aligned}$$

This implies the very important relation:

$$M_k = \Delta S_k R_k. \tag{3.33}$$

Apart from the constant Δ, the DFT of the convolution $m(t)$ is obtained easily from the product of the DFTs of $r(t)$ and $s(t)$. The practical importance of Eq. (3.33) arises from the fact that we can solve this equation for the DFT of the input signal without any problems:

$$S_k = \frac{M_k}{\Delta R_k}. \tag{3.34}$$

Thus, if we know the output signal $m(t)$ and the device function $r(t)$ in real space, we can proceed as follows:

1. DFT of $m(t)$ $\rightarrow$ M_k.
2. DFT of $r(t)$ $\rightarrow$ R_k.
3. Calculate the S_k with the help of Eq. (3.34).
4. Inverse DFT of S_k $\rightarrow$ $s(t)$.

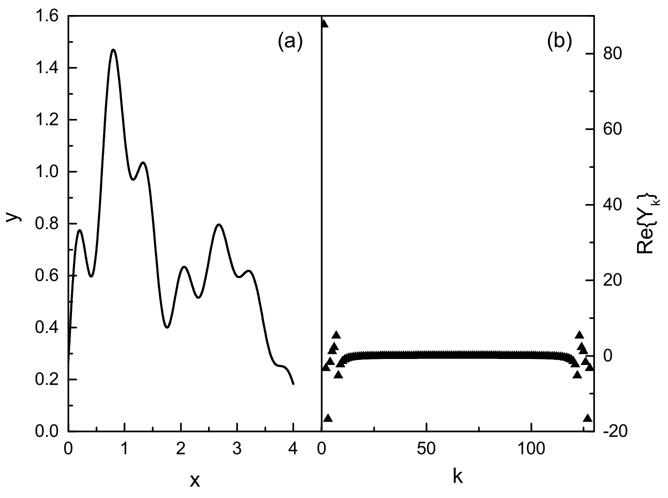

Figure 3.6: Demonstration of a data smoothing by DFT: (a) undisturbed signal according to Eq. (3.35), (b) DFT of the curve in (a): the FOURIER coefficients $\Re\text{e}\{Y_k\}$ vs k (solid black up-triangles).

3.3.3 Data smoothing

Data smoothing is derived from data interpolation but in this particular case the sampling points are assumed to have some statistical error and, thus, a 'true' reproduction of the data points by the interpolating function is not required. (This particular situation will be discussed in more detail in Chap. 4 when we will speak about least squares minimalization problems.)

This application of DFT is demonstrated using the test function shown in Fig. 3.6(a)

$$y(x) = e^{-x/2}\left[2\,e^{-2(x-1)^2} + 3\,e^{-(x-3)^2} + \frac{1}{3}\sin(10x)\right] \tag{3.35}$$

within the interval $0 \leq x \leq 4$. In Fig. 3.7(a) we overlaid $y(x)$ of Eq. (3.35) with statistical errors of a standard deviation with $\sigma = 0.1$ [see Eq. (4.3)], just like they appear unavoidably as a result of experiments.

The corresponding values of the FOURIER coefficients for $n = 128$ are presented in Figs. 3.6(b) and 3.7(b), respectively. We can clearly see how the statistical errors affect the FOURIER coefficients: in the 'middle range' of k, FOURIER coefficients appear in Fig. 3.7(b) which are different from zero [in contrast to the results for the smooth curve, Fig. 3.6(b)]. Therefore, it seems to be quite natural to ascribe these coefficients to *noise* and, consequently, to suppress them.

This is most easily done using the rule of DIRICHLET, defined by

$$\hat{Y}_k = Y_k, \quad \text{for} \quad 0 \leq k \leq k_0 \quad \text{and} \quad n - k_0 < k < n,$$

$$\hat{Y}_k = 0 \quad \text{for} \quad k_0 \leq k \leq n - k_0.$$

The index k_0 can be chosen arbitrarily.

It is obvious that the quality of the noise suppression is essentially dependent on an appropriate choice of k_0. In our particular example $k_0 = 9$ turned out to be a

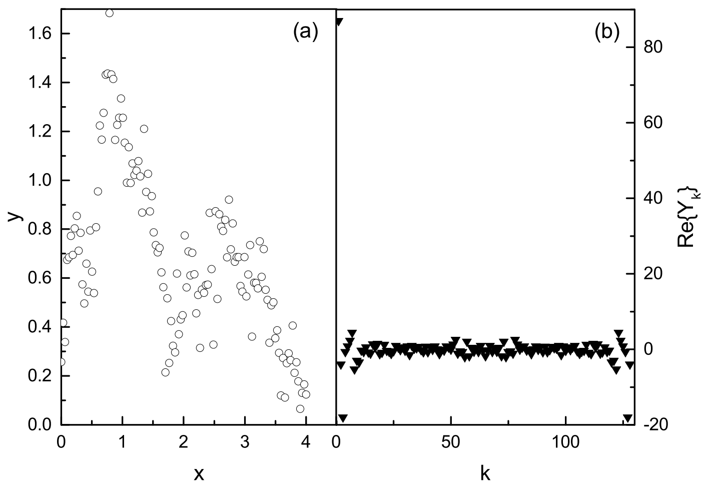

Figure 3.7: Demonstration of a data smoothing by DFT: (a) Signal according to Eq. (3.35) plus statistical errors of standard deviation $\sigma = 0.1$, (b) DFT of the curve in (a): the FOURIER coefficients $\Re\mathrm{e}\{Y_k\}$ vs k (solid black down-triangles).

good choice. Nevertheless, we have to admit that the knowledge of the ideal curve makes such a choice much easier! Otherwise, only trial and error is going to help. The high quality of the DIRICHLET smoothing is demonstrated in Fig. 3.8.

3.3.4 Frequency analysis

This is an important application in which a particular signal is decomposed into its frequency components. A typical application is the sound analysis of the human voice which even has its place as forensic evidence. Nevertheless, it has to be emphasized that 'frequency analysis' is a byproduct of an interpolation between a given set of sampling points with periodicity P.

We start with the interpolation function Eq. (3.23) written in the form

$$I(t) = \frac{1}{n}\sum_{k=0}^{n-1} Y_k \exp\left(-i\frac{2\pi kt}{P}\right) = \frac{1}{n}\sum_{k=0}^{n-1} Y_k \exp\left(-i2\pi\nu_k t\right), \tag{3.36}$$

with frequencies

$$\nu_k = \frac{k}{P}, \quad k = 0, \ldots, n-1.$$

Here, n is the number of data points and P is the period of the registered signal.

From this it becomes quite clear why this particular FOURIER transform is called a transform from time-space to frequency-space. (A transform from position space to wave-number space is absolutely equivalent from a mathematic's point of view).

We now assume not only the registered values y_j, but also the function $f(t)$ which 'lies behind these values', to be real. In this case we are mainly interested in

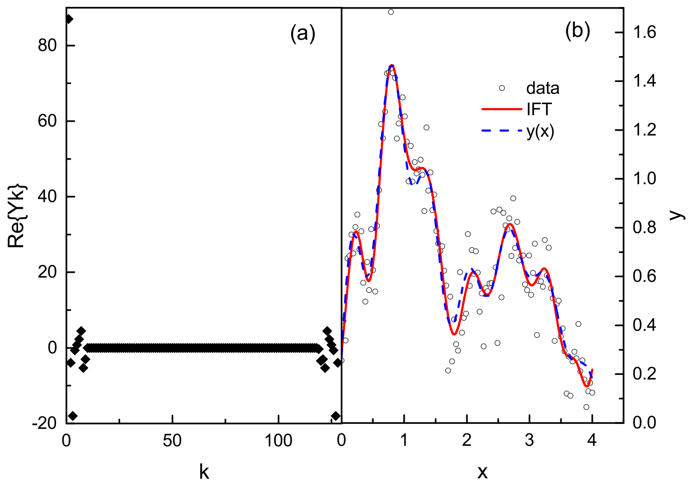

Figure 3.8: Demonstration of a data smoothing process by DFT: (a) Execution of the Dirichlet attenuation in Fourier space for $k_0 = 9$. This is to be compared with Fig. 3.7(b). (b) Effect of this Dirichlet attenuation in the (real) space of x: blue dashed curve: undisturbed signal; red solid curve: Dirichlet attenuated signal generated by inverse Fourier transform (IDFT). The disturbed signal is presented by black open circles.

the real part of the interpolation function, namely

$$\Re\mathrm{e}\left\{I(t)\right\} = \sum_{k=0}^{n-1}\left[\Re\mathrm{e}\left\{\frac{Y_k}{n}\right\}\cos\left(\frac{2\pi kt}{P}\right) + \Im\mathrm{m}\left\{\frac{Y_k}{n}\right\}\sin\left(\frac{2\pi kt}{P}\right)\right]. \tag{3.37}$$

Thus, the function $f(t) \approx \Re\mathrm{e}\left\{I(t)\right\}$ consists of *cosine* parts with amplitudes equal to the 'real part of Y_k/n' and of *sine* parts with amplitudes equal to the 'imaginary part of Y_k/n'.

It is an obvious question whether or not the interpolation described by Eq. (3.37) is the best possible, i.e. whether or not this function really connects the data points as smoothly as possible as we discussed it in Sec. 3.2. The answer is *no* and the reason for this is illustrated in Fig. 3.9. Because of the periodicity of the interpolation function, the Fourier coefficients Y_k are also periodic, with period n:

$$Y_k = Y_{k+\mu n}, \tag{3.38}$$

for any integer μ.

We can now demonstrate, that each connected group of n Fourier coefficients and frequencies gives an interpolation function which passes through the sampling points, regardless of the range of the frequency axis in which this group lies. Naturally, we obtain different interpolation functions for different ranges!

In particular, we want to know which of these possible interpolation functions is the one with the smallest tendency for oscillations between sampling points. The answer is easily found: It is the one that includes frequencies of smallest absolute values. If n coefficients are required, the ideal frequency range will then, necessarily, lie around zero frequency and, thus, within the range $-n/2 < k \leq +n/2$.

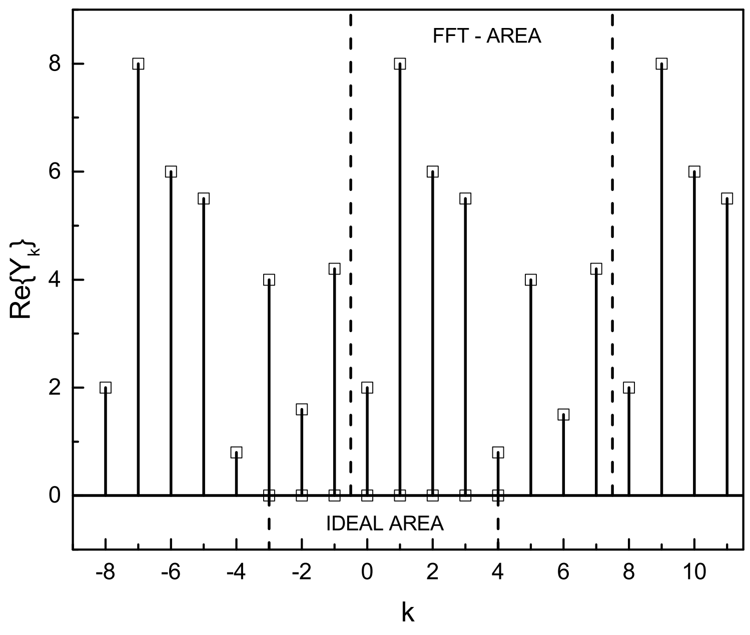

Figure 3.9: Periodicity of the real part of the FOURIER coefficients in k-space for $n = 8$. Two areas are marked: 'FFT - AREA' indicates the area of the FOURIER coefficients as it is spanned by FFT programs ($k = 0, \ldots, 7$). The 'IDEAL AREA', on the other hand, is the area where $k = -3, -2, -1, 0, 1, 2, 3, 4$ which yields the "smoothest possible" interpolation function.

We find this *ideal* interpolation function by slightly rearranging Eq. (3.37):

$$\begin{aligned}\Re\mathrm{e}\left\{I(t)\right\} &= \sum_{k=-(n/2)+1}^{-1}\left[\Re\mathrm{e}\left\{\frac{Y_k}{n}\right\}\cos\left(\frac{2\pi kt}{P}\right)+\Im\mathrm{m}\left\{\frac{Y_k}{n}\right\}\sin\left(\frac{2\pi kt}{P}\right)\right]\\ &\quad+\sum_{k=0}^{n/2}\left[\Re\mathrm{e}\left\{\frac{Y_k}{n}\right\}\cos\left(\frac{2\pi kt}{P}\right)+\Im\mathrm{m}\left\{\frac{Y_k}{n}\right\}\sin\left(\frac{2\pi kt}{P}\right)\right].\end{aligned}$$

We perform in the first term of this equation a transformation of indices $k' = n + k$ and get:

$$\begin{aligned}\Re\mathrm{e}\left\{I(t)\right\} &= \sum_{k'=(n/2)+1}^{n-1}\left[\Re\mathrm{e}\left\{\frac{Y_{k'-n}}{n}\right\}\cos\left(\frac{2\pi(k'-n)t}{P}\right)\right.\\ &\quad\left.+\Im\mathrm{m}\left\{\frac{Y_{k'-n}}{n}\right\}\sin\left(\frac{2\pi(k'-n)t}{P}\right)\right]\\ &\quad+\sum_{k=0}^{n/2}\left[\Re\mathrm{e}\left\{\frac{Y_k}{n}\right\}\cos\left(\frac{2\pi kt}{P}\right)+\Im\mathrm{m}\left\{\frac{Y_k}{n}\right\}\sin\left(\frac{2\pi kt}{P}\right)\right].\end{aligned}$$

We take into account the periodicity of Y_k according to Eq. (3.38)

$$Y_{k'-n} = Y_{k'},$$

and set $k' \equiv k$ which results in

$$\Re\mathrm{e}\left\{I(t)\right\}_{\mathrm{ideal}} = \sum_{k=0}^{n-1}\left[\Re\mathrm{e}\left\{\frac{Y_k}{n}\right\}\cos\left(2\pi\nu_k t\right) + \Im\mathrm{m}\left\{\frac{Y_k}{n}\right\}\sin\left(2\pi\nu_k t\right)\right], \tag{3.39}$$

with

$$\nu_k = \begin{cases} k/P & \text{for} \quad k = 0,\dots,n/2 \\ (k-n)/P & \text{for} \quad k = n/2+1,\dots,n-1. \end{cases} \tag{3.40}$$

This does, of course, not imply that Eqs. (3.39) and (3.40) represent the (unknown) function $f(t)$ 'which lies behind the sampling points' *in an ideal way*. Nevertheless, it is the interpolation function with the least possible oscillations between the sampling points. This stems from the fact that only *discrete* frequencies with a resolution of

$$\Delta\nu = \frac{1}{P} \tag{3.41}$$

within the frequency interval

$$-\frac{n}{2P} < \nu \le \frac{n}{2P} \tag{3.42}$$

are covered. The 'limiting frequency' $\nu_{\mathrm{Ny}} = n/(2P)$ is called 'NYQUIST frequency' in literature.

We can draw the following conclusions concerning the quality of the frequency analysis:

- A large period P results in a high frequency resolution.
- A large number of sampling points n results in a large frequency range.

The aliasing effect

If the signal to be analyzed contains a frequency component $\nu_0 = k_0/P$ which is outside the range (3.42) a disturbing effect is introduced. It projects the corresponding amplitude with the index k_0, Y_{k_0}, into the NYQUIST range, i.e. a frequency component

$$\nu' = \frac{n-k_0}{P} \tag{3.43}$$

is displayed, that actually does not exist. This aliasing effect is shown schematically in Fig. 3.10. Here $F(\nu)$ is the DFT of $f(t)$.

A simple example

In an attempt to elucidate some of the points presented in the previous subsections we study a simple example of frequency analysis: We define an analytical function $f(t)$, that only consists of cosines and sines and generate from this function a set of sampling points which is then used for frequency analysis by DFT. The DFT analysis will return the amplitudes and frequencies of the test function. We will use the following test function:

$$f(t) = 2\cos(2\pi\,2t) - 3\sin(2\pi\,4t) - \cos(2\pi\,4t) + 2\sin(2\pi\,7t). \tag{3.44}$$

The corresponding frequencies and amplitudes are listed in Table 3.3. The period

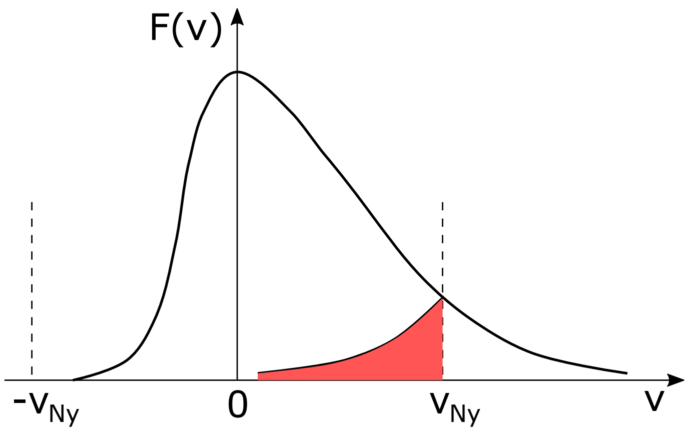

Figure 3.10: The aliasing effect in frequency analysis (schematic). The shaded area indicates how regions of $F(\nu)$ for $\nu > \nu_{\mathrm{Ny}}$ are projected inside the region $-\nu_{\mathrm{Ny}} < \nu \leq \nu_{\mathrm{Ny}}$.

Table 3.3: Frequencies and amplitudes of the test function $f(t)$ Eq. (3.44).

frequency (Hz)	cosine ampl.	sine ampl.
2	+2	–
4	-1	-3
7	–	+2

of the test function $P = 1\,\mathrm{s}$ and $n = 64$ sampling points were generated. For the frequency analysis this means a frequency resolution of $\Delta\nu = 1\,\mathrm{Hz}$ and a frequency range of –31 to +32 Hz [see Eqs. (3.41) and (3.42)] for the ideal interpolation function $I(t)$. The DFT of the test function $f(t)$ results in the FOURIER coefficients Y_k for the unoptimized frequencies ν_k and $k \in [0, 63]$. The result is shown in Fig. 3.11(b) and analyzed in Table 3.4. The first three frequencies are correct, the last three

Table 3.4: Analysis of Fig. 3.11(b).

k	ν_k (Hz)	$\Re\mathrm{e}\{Y_k/n\}$	$\Im\mathrm{m}\{Y_k/n\}$
		(cosine)	(sine)
2	2.0	1.0	0.0
4	4.0	-0.5	-1.5
7	7.0	0.0	1.0
57	57.0	0.0	-1.0
60	60.0	-0.5	1.5
62	62.0	1.0	0.0

frequencies are far away from the optimal region (3.42). The corresponding sine-cosine curve is nevertheless one possible *interpolation function* $I(t)$ as demonstrated in Fig. 3.12(a), although heavy oscillations occur between the sampling points.

If we perform a frequency optimization on the basis of Eqs. (3.39) to (3.40), we obtain the frequencies and FOURIER coefficients listed in Table 3.5. This result is consistent with the test requirements for both frequencies and amplitudes. For instance we find two cosine terms, each with an amplitude of 1.0 and with frequencies

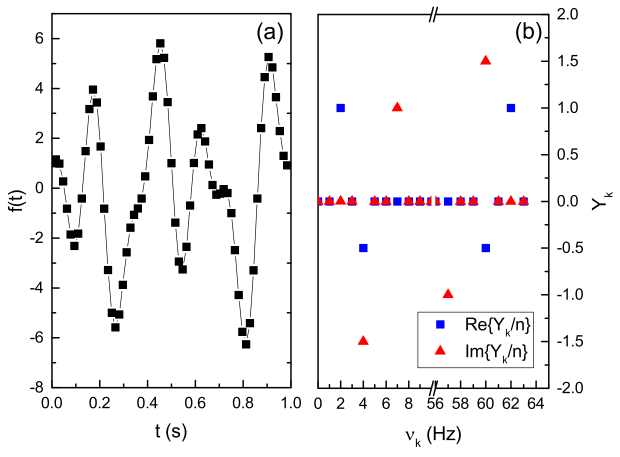

Figure 3.11: (a) The test function $f(t)$ according to Eq. (3.44). The solid squares indicate the 64 sampling points chosen for analysis. (b) The complex FOURIER coefficients Y_k as a function of frequency ν_k. The solid, blue squares correspond to the real part of Y_k while the solid, red up-triangles correspond to the imaginary part of Y_k.

Table 3.5: Analysis of Fig. 3.12(b) with frequency optimization.

k	ν_k (Hz)	$\Re\text{e}\{Y_k/n\}$ (cosine)	$\Im\text{m}\{Y_k/n\}$ (sine)
2	2.0	1.0	0.0
4	4.0	-0.5	-1.5
7	7.0	0.0	1.0
-7	-7.0	0.0	-1.0
-4	-4.0	-0.5	1.5
-2	-2.0	1.0	0.0

+2 and −2. This means:

$$\cos(2\pi\, 2t) + \cos(-2\pi\, 2t) = 2\cos(2\pi\, 2t),$$

i.e. the cosine term with frequency 2 Hz has a total amplitude of 2 which is consistent with the first term of the test function (3.44). We also have two sine terms, one with a frequency of +4 Hz and an amplitude of −1.5 and a second one with a frequency of −4 Hz and an amplitude of +1.5. This results in:

$$-1.5\sin(2\pi\, 4t) + 1.5\sin(-2\pi\, 4t) = -3\sin(2\pi\, 4t),$$

which is exactly the second term of the test function (3.44) and so on. The corresponding interpolation curve $I(t)$ is shown in Fig. 3.12(b) and follows exactly the test function $f(t)$.

Finally we want to demonstrate the aliasing effect. Let us assume the test function $f(t)$ to be defined as:

$$f(t) = 2\cos(2\pi\, 2t) - 3\sin(2\pi\, 4t) - \cos(2\pi\, 4t) + 2\sin(2\pi\, 55t). \tag{3.45}$$

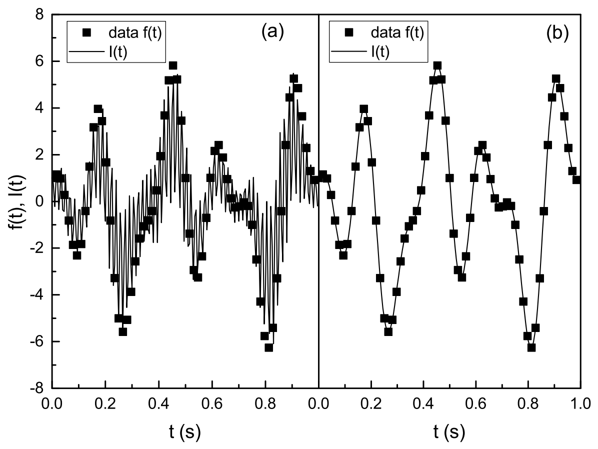

Figure 3.12: (a) Interpolation function $I(t)$ vs t after DFT frequency analysis without frequency optimization, (b) Interpolation function $I(t)$ after DFT frequency analysis with frequency optimization. The solid squares represent the sampling points.

Thus, except for the fourth term, which has a frequency of 55 Hz, this test function is identical to previous one, Eq. (3.44). Since the period is 1 s and $n = 64$, the NYQUIST frequency has a value of 32 Hz. Consequently, the frequency of the fourth term lies outside the NYQUIST range. The DFT result including the frequency optimization is quoted in Table 3.6. We thus find the first three terms of the test function, while

Table 3.6: DFT analysis of Eq. (3.45) with frequency optimization.

k	ν_k (Hz)	$\Re\text{e}\{Y_k/n\}$ (cosine)	$\Im\text{m}\{Y_k/n\}$ (sine)
2	2.0	1.0	0.0
4	4.0	-0.5	-1.5
7	7.0	0.0	1.0
9	9.0	0.0	-1.0
-9	-9.0	0.0	1.0
-7	-7.0	0.0	-1.0
-4	-4.0	-0.5	1.5
-2	-2.0	1.0	0.0

the last term with 55 Hz does not appear. Instead we find a frequency of ±9 Hz that cannot be found in the original function. This is a typical aliasing effect [see Eq. (3.43)] because:

$$64 - 55 = 9.$$

The total amplitude of this aliasing term is $-1 - (1) = -2$, as we can see from Table 3.6. In contrast, the term with a frequency of 55 Hz in the corresponding test function (3.45) has an amplitude of +2. The reason for this discrepancy can be found in the fact that when parts of $f(t)$ are projected back into the NYQUIST range by aliasing, cosine terms retain their sign while sine terms do change their sign.

There are numerous important properties, examples and applications of descrete FOURIER transforms beyond those already discussed above for which we refer to the literature [8, 4, 3, 1].

Bibliography

[1] Bernatz, R.: Fourier Series and Numerical Methods for Partial Differential Equations. Wiley, New York (2010)

[2] Frigo, M., Johnson, S.G.: FFTW, The Fastest Fourier Transform in the West,Vers. 3.3.4 (2015). URL http://www.fftw.org

[3] Hansen, E.W.: Fourier Transforms: Principles and Applications. Wiley, New York (2014)

[4] Kammler, D.W.: A First Course in Fourier Analysis. Cambridge Univerity Press, Cambridge, UK (2008)

[5] Nussbaumer, H.J.: Fast Fourier Transform and Convolution Algorithms. Springer Series in Inormation Sciences. Springer, Berlin, Heidelberg (1982)

[6] Paulin, G., Griepentrog, E.: Numerische Verfahren der Programmiertechnik. VEB Verlag Berlin, Berlin (1975)

[7] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[8] Schücker, T.: Distributions, Fourier Transforms and Some of Their Applications to Physics. Lecture Notes in Physics: Vol. 37. World Scientific, New Jersey (1991)

[9] Überhuber, C.: Computer Numeric 1. Springer, Berlin, Heidelberg (1995)

Chapter 4

Approximation of point sets: least squares minimization

4.1 The problem

Although the method of interpolation is very powerful in many cases, there are many other problems for which it is not possible to present the data points as a smooth function. This happens in particular when the data points are affected by statistical errors as it is the case whenever they represent experimental data.

Let us imagine we measure the resistance R of a metallic conductor using the experimental setup depicted in Fig. 4.1. We obtain a current-voltage diagram, shown in Fig. 4.2. The relation between voltage U and current I is linear and follows OHM's law

$$U = RI,$$

with R the electrical resistance. Statistical deviations and possible measurement errors do not change the linear relation between U and I! Thus, the approximating function **must** be a straight line (a polynomial of first order). Clearly, it doesn't make sense to approximate such data with an interpolation function that passes exactly through all measured points. In fact, what we rather wish to achieve is a smoothing of the data, without giving measurement errors too much weight.

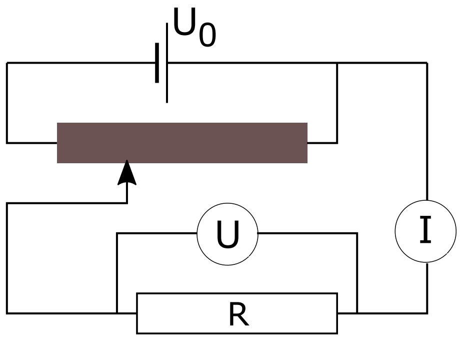

Figure 4.1: Experimental setup which allows to measure the electrical resistance R.

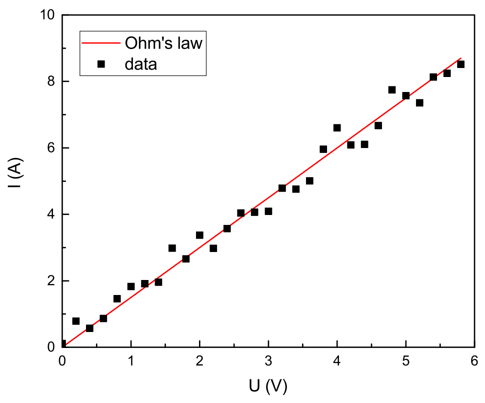

Figure 4.2: A current vs voltage diagram.

4.1.1 Mathematical formulation of the problem

Given a set of n points, $(x_i|y_i)$, we wish to find a curve $f(x)$ which approximates as closely as possible the points. At the same time, we want to take into account possible uncertainties due to measurement errors. We also wish to assign different weights to the points through suitable weighting factors. These requirements can be formulated mathematically as

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k; \mathbf{a})\right]^2 \quad \rightarrow \quad \text{min!} \tag{4.1}$$

This is the basic equation of the least squares (LSQ) approximation. Here χ^2 is the weighted error sum, $g_k > 0$ is the weighting factor of the k-th data point, and $f(x; \mathbf{a})$ is the *model function* with q model or fitting parameters $\mathbf{a} = (a_1, a_2, \ldots, a_q)^T$.

The sum of the weighted square errors between the 'real' function and its approximate value should be minimal. The significance of the weighting factor is quite obvious, if we think in terms of a statistical analysis of the data. We will explain this in more detail in the rest of the chapter.

4.2 Statistical analysis of the least squares problem

4.2.1 Expectation value and standard devation of a measurement

The LSQ method is based on the comparison between the experimental quantities y_k and the corresponding approximate values of the model function $f(x_k; \mathbf{a})$. It is obvious that the quality of the results (fit parameters) will depend strongly on the statistical quality of the y_k values. Let us assume we repeat n times a measurement under exactly the same conditions. The measured values at $x = x_k$ will correspond to (in general) *different* values $y_k^{(1)}, y_k^{(2)}, \ldots, y_k^{(n)}$. If we plot all these values in a diagram, as in Fig. 4.3, we obtain a set of points distributed around the *expectation*

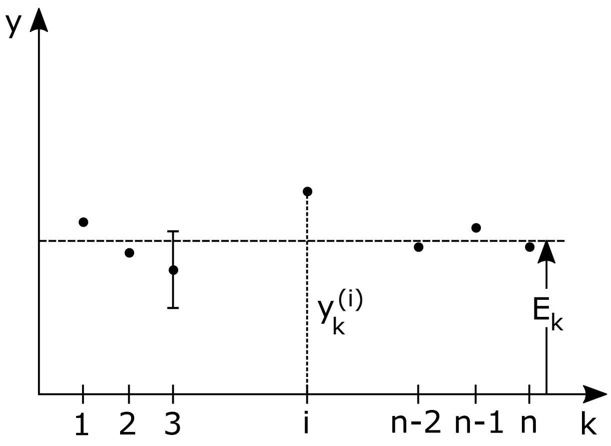

Figure 4.3: Expectation value of n measured data points.

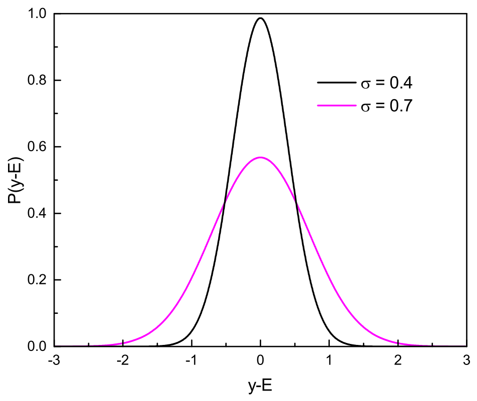

Figure 4.4: The normal Gaussian distribution.

value (or *mean value*) E_k:

$$E_k = \frac{1}{n}\sum_{j=1}^{n} y_k^{(j)}. \tag{4.2}$$

A measure of the *spread* of the y_k values around their expaction value is the *standard deviation*:

$$\sigma_k = \left\{ \frac{\sum_{j=1}^{n}\left[y_k^{(j)} - E_k\right]^2}{n-1} \right\}^{1/2}. \tag{4.3}$$

In many practical cases the probability P with which a specific distance between the measured value and the expectation value occurs is given by the following *distribution*:

$$P(y-E) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{(y-E)^2}{2\sigma^2}\right\}. \tag{4.4}$$

This is the very common *normal Gaussian distribution* (Fig. 4.4). It is important to stress in this case that $P(y-E)$ has inflection points exactly at $y-E=\pm\sigma$.

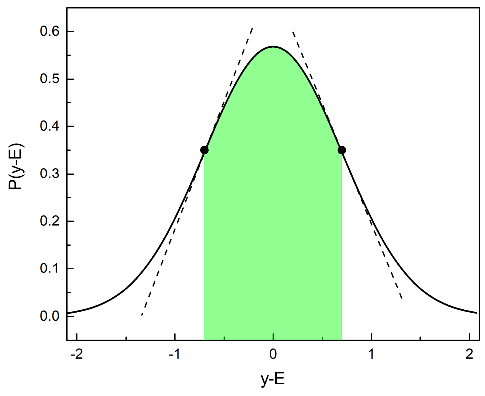

Figure 4.5: A geometrical explanation of the standard deviation σ.

The green shaded area in Fig. 4.5 represents the probability for a measured value to lie within the interval $[E-\sigma, E+\sigma]$. This probability is:

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\sigma}^{\sigma} \mathrm{d}(y-E) \, \exp\left[-\frac{(y-E)^2}{2\sigma^2}\right] = 0.685.$$

Thus, in a normal distribution, around 69 %($\approx$ 2/3) of all values lie within the interval $[E-\sigma, E+\sigma]$.

Normal distributions are typical for experiments in which the measured quantity can assume all possible values within a given interval; these are called *analogue* measurements. In the next section we will discuss the POISSON *distribution* which is typical for *digital* (counting) experiments.

4.2.2 Statistics in the LSQ method

This is not a course on statistics. Thus, we will limit ourselves to a few useful concepts without proof.

- In the LSQ method, the statistical uncertainty on the y_k's is taken into account by assuming the weighting factors in Eq. (4.1) to be the squares of the reciprocal of the corresponding standard deviations:

$$g_k \equiv \frac{1}{\sigma_k^2}. \tag{4.5}$$

- The LSQ method gives the statistical expectation value of the fitting parameters. The corresponding standard deviations can be obtained by the following procedure:

  We first calculate the so-called *normal matrix* N with elements:

$$[N]_{ij} = \frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_i \partial a_j}. \tag{4.6}$$

We then *invert* the matrix N to construct the *covariance* matrix C:

$$C = N^{-1}. \tag{4.7}$$

The covariance matrix contains two important pieces of information:

1. The diagonal elements c_{ii} of C are the standard deviations of the fitting parameters, i.e.
$$\sigma_{a_i} = \sqrt{c_{ii}}. \tag{4.8}$$
2. The *correlation coefficients* of the fitting parameters, r_{ij}, can also be extracted from C:
$$r_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}}. \tag{4.9}$$
The r_{ij}'s which always lie within the interval $[-1, +1]$ indicate how strongly the i-th and j-th fitting parameters influence each other.

- The variance V and the standard deviation are, finally, determined from:

$$V = \frac{\chi^2}{n-q}, \quad \text{and} \quad \sigma_V = \sqrt{\frac{\chi^2}{n-q}}. \tag{4.10}$$

In case of an ideal model and for $n \gg 1$, V follows approximately from a normal distribution with mean value 1 and standard deviation σ_V . The quantity $n-q$ (number of data points minus number of fitting parameters) is called the number of *degrees of freedom*. The variance can be used as an indicator for the quality of the model function employed in the fit: if V lies significantly outside the interval $[1-\sigma_V, 1+\sigma_V]$, the model is not a good one for the given set of points.

4.2.3 The standard deviation of data points

The statistical evaluation of an LSQ problem requires the knowledge of the standard deviations σ_k of the measured data. How can we determine these σ_ks?

There are two important cases:

- The measured data follow a normal distribution centered around their expectation value: this is very common in experiments. In this case it is a fairly good approximation to assume the σs of all measured data to be the same, i.e. that
$$\sigma_1 \approx \sigma_2 \approx \cdots \approx \sigma_n.$$
This hypothesis can be tested by repeating the same experiment several times for a given value of x and by calculating the corresponding σs using Eq. (4.3).

- The measured data follow another important statistical distribution, typical of counting experiments (digital measurements), namely the Poisson distribution. Such an experimental setup is shown schematically in Fig. 4.6. Without entering into the details of Poisson statistics, we will illustrate here only its most important properties:

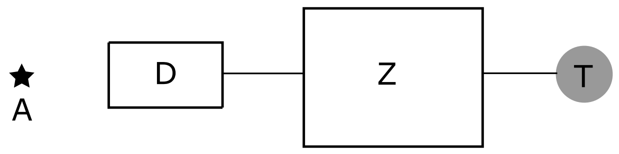

Figure 4.6: Experimental setup of a counting experiment. For instance: A is some radioactive substance, D is the radiation detector, Z is a digital counter, and T is the clock.

In a Poisson distribution the relation between the expectation value E of a measured quantity and the corresponding standard deviation σ is:

$$\sigma = \sqrt{E}.$$

In practice, the (unknown) expectation value is approximated by the measured value, giving

$$\sigma \approx \sqrt{y},$$

and the weighting factors of the LSQ method are given by:

$$g_k \approx \frac{1}{y_k}. \tag{4.11}$$

4.3 Model functions with linear parameters

The term model function with linear parameters usually indicates a function of the form

$$f(x;\mathbf{a}) = \sum_{j=1}^{m} a_j \varphi_j(x), \tag{4.12}$$

with arbitrary (linearly independent) basis functions $\varphi_j(x)$. The definition *linear model functions*, often used in practice, is misleading: $f(x;\mathbf{a})$ need not at all be linear in x but only in the fitting parameters. In *linear regression* the model function is a straight line:

$$f(x;a_1,a_2) = a_1 + a_2 x.$$

This is a special case of a model function with linear parameters.

We insert a function of the form (4.12) in the LSQ basic equation (4.1) and obtain:

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - \sum_{j=1}^{m} a_j \varphi_j(x_k) \right]^2 \quad \rightarrow \quad \text{min!}$$

This minimization of the square error is obtained by setting the partial derivatives of χ^2 with respect to the model parameters $\mathbf{a}$ to zero:

$$\frac{\partial}{\partial a_i}\chi^2 = \sum_{k=1}^{n} g_k \varphi_i(x_k) \left[y_k - \sum_{j=1}^{m} a_j \varphi_j(x_k) \right] \overset{!}{=} 0, \quad \forall i \in [1,m].$$

This is a linear, inhomogeneous set of m-th order equations for the a_is:

$$\sum_{j=1}^{m} a_j \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k) = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k), \quad \forall i \in [1,m].$$

This constitutes the linear problem

$$A\mathbf{a} = \beta,$$

with the symmetric, positive definite matrix A of coefficients

$$A \equiv [\alpha_{ij}] \quad \text{with} \quad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k) \tag{4.13}$$

and the inhomogeneous vector

$$\beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k). \tag{4.14}$$

For linear model functions (4.12) it is a rather easy task to construct the elements of the normal matrix (4.6). We obtain:

$$\frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_i \partial a_j} = \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k) = \alpha_{ij}. \tag{4.15}$$

4.3.1 Standard deviation of the fitting parameters

We discussed (without proof) in Sec. 4.2.2 that the standard deviation of the fitting parameters can be derived from the diagonal elements of the covariance matrix (4.8). We will demonstrate now that this is indeed so using the simple example of a linear regression with the model function:

$$f(x; a, b) = a + bx.$$

The LSQ formula

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - a - bx_k\right] \quad \rightarrow \quad \text{min!}$$

is fulfilled for $a = a_{\text{opt}}$ and $b = b_{\text{opt}}$. Furthermore, following Eq. (4.6) we write the normal matrix as

$$N = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

with

$$a_{11} = \sum_{k=1}^{n} g_k, \quad a_{12} = a_{21} = \sum_{k=1}^{n} g_k x_k, \quad a_{22} = \sum_{k=1}^{n} g_k x_k^2.$$

The optimized parameters are obtained as the solution of the inhomogeneous, linear system of equations:

$$N \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix},$$

with

$$\beta_1 = \sum_{k=1}^{n} g_k y_k, \quad \beta_2 = \sum_{k=1}^{n} g_k x_k y_k.$$

The solution of this system of equations is

$$a_{\text{opt}} = \frac{\beta_1\alpha_{22} - \beta_2\alpha_{12}}{D}, \quad b_{\text{opt}} = \frac{\beta_2\alpha_{11} - \beta_1\alpha_{12}}{D}, \tag{4.16}$$

with D the determinant of the normal matrix N:

$$D = \alpha_{12}\alpha_{22} - \alpha_{12}^2. \tag{4.17}$$

The covariance matrix C is the inverse of the normal matrix N, i.e.

$$C = N^{-1} = \frac{1}{D}\begin{pmatrix} \alpha_{22} & -\alpha_{12} \\ -\alpha 12 & \alpha_{11} \end{pmatrix}.$$

Consequently, the standard deviation of the fitting parameters is given by

$$\sigma_a = \sqrt{\frac{\alpha_{22}}{D}}, \quad \sigma_b = \sqrt{\frac{\alpha_{11}}{D}}. \tag{4.18}$$

These standard deviations can, of course, be obtained by another method, namely the error propagation rule (EPR). It is quite clear that the optimized parameters a_{opt} and b_{opt} are necessarily functions of all the x and y components of the given data points:

$$a_{\text{opt}} = a_{\text{opt}}(x_1, \ldots, x_n; y_1, \ldots, y_n), \quad b_{\text{opt}} = b_{\text{opt}}(x_1, \ldots, x_n; y_1, \ldots, y_n).$$

Accoring to the EPR the standard deviations of a_{opt} and b_{opt} follow from:

$$\begin{aligned} \sigma_{a_{\text{opt}}}^2 &= \sum_{\ell=1}^{n}\left[\left(\frac{\partial a_{\text{opt}}}{\partial x_\ell}\right)^2 \sigma(x_\ell)^2 + \left(\frac{\partial a_{\text{opt}}}{\partial y_\ell}\right)^2 \sigma(y_\ell)^2\right], \\ \sigma_{b_{\text{opt}}}^2 &= \sum_{\ell=1}^{n}\left[\left(\frac{\partial b_{\text{opt}}}{\partial x_\ell}\right)^2 \sigma(x_\ell)^2 + \left(\frac{\partial b_{\text{opt}}}{\partial y_\ell}\right)^2 \sigma(y_\ell)^2\right]. \end{aligned}$$

Here, the $\sigma(x_\ell)$ and $\sigma(y_\ell)$ are the standard deviations (errors) of the corresponding x and y values. If we assume the x values to be exact, $\sigma(x_\ell) = 0$, and the errors on the y's to follow from Eq. (4.5)

$$g_\ell = \frac{1}{\sigma^2(y_\ell)},$$

then we obtain:

$$\sigma_{a_{\text{opt}}}^2 = \sum_{\ell=1}^{n}\frac{1}{g_\ell}\left(\frac{\partial a_{\text{opt}}}{\partial y_\ell}\right)^2, \quad \text{and} \quad \sigma_{b_{\text{opt}}}^2 = \sum_{\ell=1}^{n}\frac{1}{g_\ell}\left(\frac{\partial b_{\text{opt}}}{\partial y_\ell}\right)^2.$$

The partial derivatives in these two equations can be calculated using Eq. (4.16) and this results in:

$$\begin{aligned} \sigma_{a_{\text{opt}}}^2 &= \sum_{\ell=1}^{n}\frac{1}{g_\ell}\left[\frac{\alpha_{22}g_\ell - \alpha_{12}g_\ell x_\ell}{D}\right]^2, \\ \sigma_{b_{\text{opt}}}^2 &= \sum_{\ell=1}^{n}\frac{1}{g_\ell}\left[\frac{-\alpha_{12}g_\ell - \alpha_{11}g_\ell x_\ell}{D}\right]^2. \end{aligned}$$

We gather all previous results and find

$$\begin{aligned}
\sigma^2_{a_{\rm opt}} &= \frac{1}{D^2}\sum_{\ell=1}^{n}\frac{1}{g_\ell}\left[\alpha_{22}g_\ell-\alpha_{12}g_\ell x_\ell\right]^2 \\
&= \frac{1}{D^2}\sum_{\ell=1}^{n}\sum_{k=1}^{n}\sum_{k'=1}^{n} g_\ell g_k g_{k'}\left[x_k^2x_{k'}^2-2x_\ell x_k^2 x_{k'}+x_\ell^2 x_k x_{k'}\right],
\end{aligned}$$

or by renaming the indices

$$\begin{aligned}
\sigma^2_{a_{\rm opt}} &= \frac{1}{D^2}\sum_{\ell,k,k'=1}^{n} g_\ell g_k g_{k'} x_k^2\left[x_{k'}^2-x_{k'}x_\ell\right] \\
&= \frac{1}{D^2}\sum_{k=1}^{n} g_k x_k^2\left[\sum_{\ell=1}^{n} g_\ell\sum_{k'=1}^{n} g_{k'}x_{k'}^2-\left(\sum_{\ell=1}^{n} g_\ell x_\ell\right)^2\right].
\end{aligned}$$

Obviously, the expression in the square bracket in the last line is the determinant D of the normal matrix N [see Eqs. (4.16) and (4.17)] and we find, finally:

$$\sigma^2_{a_{\rm opt}} = \frac{1}{D^2}D\sum_{k=1}^{n} g_k x_k^2 = \frac{\alpha_{22}}{D}.$$

This is just the result (4.18) we derived from the covariance matrix C, **q.e.d.**

4.3.2 A test problem

We consider 100 data points normally distributed around their expectation values; we will assume that σ is the same for all points. The expectation values of all points lie exactly on the third degree polynomial:

$$y(x) = 0.5 - x - 0.2x^2 + 0.1x^3. \tag{4.19}$$

Thus, the ideal model function for this problem is a third-degree polynomial (q = 4) with parameters: $a_1, \ldots, a_4$:

$$f(x;\mathbf{a}) = \sum_{j=1}^{4} a_j x^{j-1}.$$

We use the subroutine *lfit* of Ref. [6] and study first the case of $\sigma = 0.1$. Results for model functions built from polynomials with various numbers q of fitting parameters are summarized in Table 4.1. Depending on the number q of terms in the model function, the number of degrees of freedom is between 95 and 98. We should expect a variance between 0.86 and 1.14 for a good model [see Eq. (4.10)].

Due to the relatively large spread of the data points [see Fig. 4.7(a)] we cannot decide whether the best model is a second or third degree polynomial. In fact, for $q = 4$ the σ of some of the fitting parameters are close to their absolute values. These fitting parameters match very poorly the original expression, even though the 3rd degree polynomial was the correct fitting function.

The situation definitely improves if the data points have a better statistics. To demonstrate this, we consider a second data set with $\sigma = 0.025$. The results are

Table 4.1: Results of an LSQ fit to data generated from Eq. (4.19) with $\sigma = 0.1$.

q	Variance	Optimized parameters	correct	Remarks
2	1.306998	$a_1 = 0.497451 \pm 0.021365$	$a_1 = 0.5$	bad model
		$a_2 = -0.999120 \pm 0.017321$	$a_2 = -1.0$	
3	1.314428	$a_1 = 0.475493 \pm 0.035762$	$a_1 = 0.5$	
		$a_2 = -0.943120 \pm 0.075161$	$a_2 = -1.0$	bad model
		$a_3 = -0.225688 \pm 0.033549$	$a_3 = -0.2$	
4	1.325881	$a_1 = 0.455818 \pm 0.055498$	$a_1 = 0.5$	
		$a_2 = -0.852151 \pm 0.210119$	$a_2 = -1.0$	good model
		$a_3 = -0.326034 \pm 0.219026$	$a_3 = -0.2$	bad statistics
		$a_4 = 0.130687 \pm 0.066190$	$a_4 = 0.1$	see Fig. 4.7(a)
5	1.339045	$a_1 = 0.438528 \pm 0.083959$	$a_1 = 0.5$	
		$a_2 = -0.732662 \pm 0.483449$	$a_2 = -1.0$	reasonable model
		$a_3 = -0.552281 \pm 0.853013$	$a_3 = -0.2$	very bad statistics
		$a_4 = 0.287968 \pm 0.576919$	$a_4 = 0.1$	
		$a_5 = -0.036074 \pm 0.131447$	$a_5 = 0$	

presented in Table 4.2 and in Fig. 4.7(b). In this case it is much easier to decide which is the optimal model. Still, the variance of all polynomials is too large. Nevertheless, the model functions can be considered good starting from $q \leq 4$. Since, however, for $q > 4$ the absolute values of some of the fitting parameters 'fall below their σ', a third degree polynomial is, obviously, the best model function.

4.4 Nonlinear least squares fit

Before we discuss the most general case of a completely arbitrary model function $f(x; \mathbf{a})$ we want to point out that it is in most cases of advantage to linearize the model function if at all possible. For instance, if the model function is an exponential function, it may be linearized by taking the data points $\ln(y_k)$ instead of y_k.

However, if this is not possible there are numerous alternatives to find a solution of the problem. For instance, the GAUSS - NEWTON method can be employed if the model function $f(x; \mathbf{a})$ and its derivatives with respect to the parameters a_j are known analytically. Another possibility is offered by the application of an deterministic optimization algorithm, like *steepest descent* or *conjugate gradients* [1]. If even these methods are not applicable, the methods of *stochastic optimization* [2, 5, 3] might provide a last resort.

We discuss now the GAUSS - NEWTON method which is essentially a generalization of the NEWTON method which will be discussed in Subsec. 6.2.3. The GAUSS - NEWTON method is a method developed to minimize the expression (4.1) iteratively. The derivatives

$$\frac{\partial f(x; \mathbf{a})}{\partial a_\ell} \tag{4.20}$$

are assumed to be known analytically. This is an iterative algorithm and, thus, an iteration index is introduced and indicated as a superscript index a_j^n. The algorithm is described by the following steps:

1. Choose a set of initial values $\{a_j^0\}$ for the iteration.

Table 4.2: Results of an LSQ fit to data generated from Eq. (4.19) with $\sigma = 0.025$.

q	Variance	Optimized parameters	correct	Remarks
2	1.306998	$a_1 = 0.499363 \pm 0.005341$	$a_1 = 0.5$	bad model
		$a_2 = -0.999780 \pm 0.004330$	$a_2 = -1.0$	
3	1.314428	$a_1 = 0.493873 \pm 0.008940$	$a_1 = 0.5$	
		$a_2 = -0.985780 \pm 0.018790$	$a_2 = -1.0$	bad model
		$a_3 = -0.206422 \pm 0.008387$	$a_3 = -0.2$	
4	1.325881	$a_1 = 0.488954 \pm 0.013875$	$a_1 = 0.5$	good model
		$a_2 = -0.963038 \pm 0.052530$	$a_2 = -1.0$	the fitting parameters have
		$a_3 = -0.231508 \pm 0.054757$	$a_3 = -0.2$	convenient σ
		$a_4 = 0.107672 \pm 0.016548$	$a_4 = 0.1$	see Fig. 4.7(b)
5	1.339045	$a_1 = 0.484632 \pm 0.020990$	$a_1 = 0.5$	good model
		$a_2 = -0.933166 \pm 0.120862$	$a_2 = -1.0$	but some fitting parameters
		$a_3 = -0.288070 \pm 0.213253$	$a_3 = -0.2$	have very bad statistics
		$a_4 = 0.146992 \pm 0.144230$	$a_4 = 0.1$	'mixing of parameters'
		$a_5 = -0.009018 \pm 0.032862$	$a_5 = 0$	

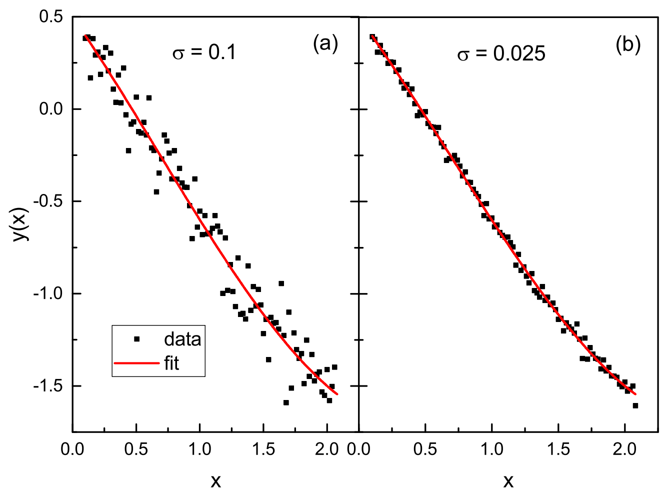

Figure 4.7: Linear LSQ calculation of simulated data: (a) $\sigma = 0.1$, (b) $\sigma = 0.025$. The black solid squares correspond to the data [Eq. (4.19) plus σ] and the red solid line reproduces the linear LSQ fit for $q = 4$.

2. Linearize the function $f(x_\ell; \{\alpha_j^n\})$ and insert the result into Eq. (4.1):

$$\chi^2 \approx \chi_k^2 = \sum_k g_k \left\{ y_k - f(x_k; \{a_j^n\}) - \sum_\ell \left[\frac{\partial f(x_k; \{a_j\})}{\partial a_\ell} \right]_{\{a_j\}=\{a_j^n\}} (a_\ell - a_\ell^n) \right\}^2 . \tag{4.21}$$

   We introduce the following abbreviations for a more compact notation:

$$\mathrm{d}f_{k,\ell}^n = \left[\frac{\partial f(x_k; \{a_j\})}{\partial a_\ell} \right]_{\{a_j\}=\{a_j^n\}} , \tag{4.22}$$

   and

$$f_k^n = f(x_k; \{a_j^n\}). \tag{4.23}$$

3. We have to solve

$$\frac{\partial \chi^2}{\partial a_i} = -2 \sum_k g_k \mathrm{d}f_{k,i}^n \left[y_k - f_k^n - \sum_\ell \mathrm{d}f_{k,\ell}^n (a_\ell - a_\ell^n) \right] \stackrel{!}{=} 0, \tag{4.24}$$

   for all parameters $\{a_j\}$. Therefore, we introduce in addition to the vector $\mathbf{a} = (a_1, a_2, \ldots)^T$ the vector $\beta = (\beta_1, \beta_2, \ldots)^T$ with elements

$$\beta_i = \sum_k g_k (y_k - f_k^n) \mathrm{d}f_{k,i}^n, \tag{4.25}$$

   and the matrix M with elements:

$$M = [M_{ij}], \quad M_{ij} = \sum_k g_k \mathrm{d}f_{k,i}^n \mathrm{d}f_{k,j}^n . \tag{4.26}$$

   This transforms Eq. (4.24) into a linear system of equations

$$M(\mathbf{a} - \mathbf{a^{(n)}}) = \beta, \tag{4.27}$$

   which is solved for $\Delta\mathbf{a^{(n)}} = \mathbf{a} - \mathbf{a^{(n)}}$. Please note that $\mathbf{a^{(n)}}$ denotes the vector $\mathbf{a}$ after n iterations. The vector $\mathbf{a^{(n+1)}}$ for the next iteration step is guessed from:

$$\mathbf{a^{(n+1)}} = \mathbf{a^{(n)}} + \Delta\mathbf{a^{(n)}}. \tag{4.28}$$

4. The iteration is terminated if for all parameters the desired accuracy was achieved. For instance, the condition $|a_j^{n+1} - a_j^n| \le \varepsilon$ can be used with ε a small parameter. A criterion for the relative error can be formulated in analogue.

Some comments concerning the covariance matrix are in order: It is more complicated in the nonlinear case because we also have to consider the second partial derivatives of the model function $f(x; \mathbf{a})$. However, if these can for some reason be neglected we obtain, again, that $N_{ij} = M_{ij}$, as in Sec. 4.3. Another, more serious problem is found in the fact that the GAUSS - NEWTON method suffers from severe instability problems. However, a possible remedy was formulated by D. MARQUART [4] who suggested to multiply the diagonal elements of the matrix M with a factor $(1 + \lambda)$ where $\lambda > 0$. A detailed analysis shows that one can choose λ

sufficiently large and in such a way that the value of χ_n^2 decreases monotonically, i.e. $\chi_{n+1}^2 \leq \chi_n^2$ for all iteration steps n. However, an increase of λ decreases the convergence rate and more iterations are necessary until the required accuracy was obtained. It is therefore desirable to choose λ values in such a way that the error decreases monotonically but that, at the same time, a convergence rate is maintained which is as large as possible. A possible strategy is to start with some given value of λ and to reduce it after every iteration step by a constant rate. However, if at some point the error χ^2 increases, i.e. $\chi_{n+1}^2 > \chi_n^2$, then λ has again to be increased.

Bibliography

[1] Avriel, M.: Nonlinear Programming: Analysis and Methods. Dover Publications, New York (2003)

[2] Hartmann, A.H., Rieger, H.: Optimization Algorithms in Physics. Wiley - VCH, Berlin (2002)

[3] Locatelli, M., Schoen, F.: Global Optimization. MPS-SIAM Series on Optimization. Cambridge University Press, Cambridge, UK (2013)

[4] Marquart, D.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics **11**, 431–441 (1963). DOI 10.1137/0111030

[5] Panos, M.P., Resende, M.G.C. (eds.): Handbook of Applied Optimization. Oxford University Press, New York (2002)

[6] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

Chapter 5

Eigenvalues and eigenvectors of real matrices

5.1 Introduction

We considered in Chap. 2 non-homogeneous problems of the form:

$$A\mathbf{x} = \mathbf{b}, \qquad A: \text{ real matrix.}$$

Now, we will treat homogeneous problems, i.e. problems for which $\mathbf{b}$ is a zero vector:

$$A\mathbf{x} = \mathbf{0}. \tag{5.1}$$

Theory tells us that we have to distinguish two cases:

- The determinant of the matrix of coefficients A is different from zero: in this case the solution $\mathbf{x}$ is, necessarily, the zero vector.
- The determinant of A is zero: in this case, apart from the *trivial solution* $\mathbf{x} = \mathbf{0}$, there is also a non-trivial one $\mathbf{x} \neq \mathbf{0}$. For example, the homogeneous system:
$$\begin{pmatrix} 1 & 2 & 3 \\ -1 & 13 & 2 \\ 0 & 3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$
admits the non-trivial solution $\mathbf{x} = (-7, -1, +3)^T$. It is clear that this solution is defined only up to a multiplicative constant c: in fact, every vector $c\mathbf{x}$ is still a solution of the system.

In the following, however, we do not want to consider systems of type (5.1). In fact, the most important systems for practical applications are homogeneous systems of the type

$$A(\lambda)\mathbf{x} = 0, \quad \text{with} \quad A(\lambda) = [a_{ij}(\lambda)], \tag{5.2}$$

in which the elements of the matrix A depend on a variable λ. The solutions of Eq. (5.2) are:

- The trivial solution $\mathbf{x} = \mathbf{0}$.
- Non-trivial solutions $\mathbf{x}_1, \mathbf{x}_2, \ldots$ when λ assumes one of the values $\lambda_1, \lambda_2, \ldots$ for which the matrix A is singular.

Thus, the condition for *non-trivial solutions* is:

$$\det[A(\lambda)] = 0. \tag{5.3}$$

The (complex) solutions of Eq. (5.3), λ_1, λ_2, ... and the corresponding vectors $\mathbf{x}_1$, $\mathbf{x}_2$, ... are the *eigenvalues* and *eigenvectors* of the system (5.2). Solving Eq. (5.2) for its eigenvalues and eigenvectors is called the solution of the *generalized eigenvalue* problem. Usually, there is essentially only one possibility to solve this kind of problems, namely to determine the roots of Eq. (5.3) numerically. This, in turn, requires the evaluation of many determinants and, therefore, this method is numerically very expensive.

A very important sub-case of the generalized eigenvalue problem (5.2) is that of *regular eigenvalue problems*, with

$$A(\lambda) = A_0 - \lambda\, I. \tag{5.4}$$

In this particular case the matrix A_0 doesn't depend on λ. This kind of problems allows for more effective methods to find the solution.

For regular eigenvalue problems

$$(A_0 - \lambda\, I)\mathbf{x} = 0, \quad \text{or} \quad A_0\mathbf{x} = \lambda\mathbf{x}, \tag{5.5}$$

the condition for non-trivial solutions is given by:

$$\det(A_0 - \lambda\, I) = 0. \tag{5.6}$$

We will now consider only regular eigenvalue problems and omit, for brevity, the subscript 0 in A_0.

The evaluation of Eq. (5.6) results in an n-th degree polynomial in λ where n is the order of the linear system (5.5). This polynomial is called the *characteristic polynomial* of the matrix A:

$$\det(A - \lambda\, I) \equiv P_n(\lambda) = \lambda^n + \sum_{i=1}^{n} p_i \lambda^{n-i}. \tag{5.7}$$

The p_1, ..., p_n are the (real) coefficients of the characteristic polynomial. This polynomial has exactly n (not necessarily different) roots which are either real or complex-conjugate. Due to Eq. (5.6) the n roots are the eigenvalues of the matrix A and, thus, the characteristic polynomial has multiple roots:

$$P_n(\lambda) = 0, \quad \text{for} \quad \lambda: \ \lambda_1, \lambda_2, \dots \lambda_k = \lambda_{k+1}, \dots = \lambda_{k+q-1}, \lambda_{k+q}, \dots \lambda_n.$$

Here, the eigenvalue λ_k is q times *degenerate*. Furthermore, following Eq. (5.5) each eigenvalue λ_i is associated with an eigenvector $\mathbf{x}_i$.

5.1.1 Eigenvalues and eigenvectors of specific matrices

Following the classification presented in Subsec. 2.2.3 it is quite obvious that *all the eigenvalues of symmetrical or hermitian matrices are real.* It is also obvious that symmetrical and hermitian matrices are normal matrices.

A very important question is to know whether a given matrix is *diagonalizable* or not. Diagonalizable means that there exists a non-singular matrix U which can transform the matrix A into a diagonal matrix D:

$$U^{-1} A\, U = D. \tag{5.8}$$

This type of transformation is a *similarity operation* which does not change the spectrum of the matrix and this means that the eigenvalues of D are also the eigenvalues of A.

- The eigenvalue problem of matrix D is easy to solve (the eigenvalues are the diagonal elements of D), the similarity operation (5.8) is equivalent to the evaluation of the eigenvalues of A:

$$\lambda_i = d_{ii}. \tag{5.9}$$

- We can easily demonstrate that the column vectors of U coincide with the eigenvectors of A: We multiply the transformation (5.8) from left with U, and obtain:

$$U\, U^{-1} A\, U = U D \quad \text{and} \quad A U = U D. \tag{5.10}$$

  The matrix U can also be represented as a system of n column vectors:

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nn} \end{pmatrix} \equiv (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n)\,.$$

  The component-by-component evaluation of the right side of Eq. (5.10) gives:

$$[UD]_{ij} = \sum_{\ell=1}^{n} u_{i\ell} d_{\ell j} = u_{ij} d_{jj} = \lambda_j u_{ij}.$$

  On the other hand, the matrix UD can be written in terms of vectors using Eq. (5.9) as:

$$UD \equiv (\lambda_1 \mathbf{u}_1, \lambda_2 \mathbf{u}_2, \ldots, \lambda_n \mathbf{u}_n)\,.$$

  Thus, Eq. (5.10) gives

$$A\,(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n) = (\lambda_1 \mathbf{u}_1, \lambda_2 \mathbf{u}_2, \ldots, \lambda_n \mathbf{u}_n)$$

  with the consequence that each column vector $\mathbf{u}_j$ satisfies the eigenvalue condition and we can state: *The j-th column of the transformation matrix U represents the j-th eigenvector of the original matrix A.*

Thus, if we are able to find the transformation matrix U we are able to solve the eigenvalue problem for the matrix A!

Which matrices are diagonalizable?

- All those matrices whose eigenvectors form a *linearly independent* system.

- It is possible to prove that, in particular, the eigenvectors of normal matrices have this property. Moreover, the eigenvectors of such matrices are orthonormal, i.e. they obey the relation
$$\mathbf{x}_i \mathbf{x}_j^\star = \delta_{ij}.$$
We conclude since the eigenvectors of A are the column vectors of the transformation matrix U that:
$$UU^T = I \quad \text{and} \quad UU^\dagger = I.$$

Normal matrices have the following important property: *Normal matrices can be reduced into diagonal form through an orthogonal and unitary transformation:*
$$U^T AU = D \quad \text{and} \quad U^\dagger AU = D. \tag{5.11}$$

5.2 Numerical solution of regular eigenvalue problems

5.2.1 General remarks

The numerical solution of regular eigenvalue problems is a sub-field of numerical mathematics and is of extraordinary importance. Over the years, numerous specialised methods have been developed. A possible application of these methods requires the potential user to know in advance exactly about his/her requirements in order to be able to choose the most suitable method for the given problem.

It is the aim of this section to illustrate the basic ideas of the most important methods which are an example of simple, yet effective algorithms. A long and exhaustive review of all possible methods, albeit very helpful, is outside the scope of this lecture. The interested reader is referred to the literature, for instance Refs. [4, 3, 2, 1].

In the following we will restrict our analysis to real matrices and describe:

- VON MISES' method (method of vector iteration).
- JACOBI's method.

5.2.2 VON MISES' method (Power method)

This iterative method returns the eigenvalue which has the largest (or, as we will see in the following, the smallest) absolute value for a given real matrix A.[1]

The application of this method requires the matrix A to be diagonalizable (i.e. its eigenvectors must form a linearly independent system) and to possess one dominant eigenvalue:
$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_{n-1}| \geq |\lambda_n|. \tag{5.12}$$
As the eigenvectors of A are linearly independent, any other vector $\mathbf{v}^{(0)} \neq \mathbf{0}$ can be expressed as a linear combination of the eigenvectors $\mathbf{x}_i$
$$\mathbf{v}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i, \quad \text{with} \quad |\alpha_1| + |\alpha_2| + \cdots + |\alpha_n| \neq 0.$$

[1]For brevity, we will speak about the *smallest* or *largest* eigenvalue in the understanding that this is meant to be the eigenvalue with the smallest or largest absolute value.

We then choose $\mathbf{v}^{(0)}$ as the starting vector for the following iteration

1. $\mathbf{v}^{(0)}$ is multiplied from the left with the matrix A:

$$A\mathbf{v}^{(0)} \equiv \mathbf{v}^{(1)} = \sum_{i=1}^{n} \alpha_i A\mathbf{x}_i = \sum_{i=1}^{n} \alpha_i \lambda_i \mathbf{x}_i,$$

   where the last identity follows from the eigenvalue equation (5.5).

2. The vector $\mathbf{v}^{(1)}$ is then multiplied from the left with matrix A (*continued vector multiplication*)

$$A\mathbf{v}^{(1)} \equiv \mathbf{v}^{(2)} = \sum_{i=1}^{n} \alpha_i \lambda_i A\mathbf{x}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^2 \mathbf{x}_i.$$

We repeate these steps and generate a sequence of vectors $\mathbf{v}^{(0)}$, $\mathbf{v}^{(1)}$, ..., $\mathbf{v}^{(t)}$, $\mathbf{v}^{(t+1)}$, ... with

$$\mathbf{v}^{(t)} = \sum_{i=1}^{n} \alpha_i \lambda_i^t \mathbf{x}_i, \qquad t = 0, 1, \dots . \tag{5.13}$$

If there is one dominant eigenvalue as suggested by the relation (5.12) the first terms of the sum (5.13) will become more and more dominant in successive iterations as the powers of λ increase. For not too small values of t we can write in a good approximation:

$$\mathbf{v}^{(t)} \approx \alpha_1 \lambda_1^t \mathbf{x}_1, \quad \text{and} \quad \mathbf{v}^{(t+1)} \approx \alpha_1 \lambda_1^{t+1} \mathbf{x}_1. \tag{5.14}$$

This are vector equations and they have to be satisfied for each of the ℓ components. Thus, we have:

$$\frac{v_\ell^{(t+1)}}{v_\ell^{(t)}} \approx \lambda_1, \quad \forall\, \ell = 1, 2, \dots, n. \tag{5.15}$$

Consequently, the ratio between the same component of two subsequent vectors tends towards the largest eigenvalue if the iteration converges. We find:

$$\lim_{t\to\infty} \frac{v_\ell^{(t+1)}}{v_\ell^{(t)}} \to \lambda_1 \tag{5.16}$$

Furthermore, Eq. (5.14) results in:

$$\lim_{t\to\infty} \mathbf{v}^{(t)} \quad \to \quad \propto \mathbf{x}_1. \tag{5.17}$$

In principle we could choose an arbitrary value ℓ to evaluate Eq. (5.15). However, we cannot *a priori* exclude that during the iteration one of the $v_\ell^{(t)}$ becomes extremely small, or even zero. This can be avoided if we evaluate

$$\frac{1}{n'} \sum_{\mu} \frac{v_\mu^{(t+1)}}{v_\mu^{(t)}} \approx \lambda_1 \tag{5.18}$$

instead of Eq. (5.15). Here the sum goes over all indices μ for which

$$\left| v_\mu^{(t)} \right| > \varepsilon \tag{5.19}$$

and n' is the number of terms in the sum which satisfy this condition.

Calculation of the smallest eigenvector

The smallest eigenvalue of a real matrix is for many practical applications often more interesting than its largest eigenvalue. VON MISES' method can also be applied to solve this problem.

We start from the eigenvalue equation (5.5)

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i$$

and multiply it with the matrix A^{-1}

$$\mathbf{x}_i = \lambda_i A^{-1}\mathbf{x}_i.$$

This results in the *eigenvalue equation for the inverse matrix*

$$A^{-1}\mathbf{x}_i = \frac{1}{\lambda_i}\mathbf{x}_i \tag{5.20}$$

and we find:

- The eigenvalues of a matrix A^{-1} are simply the inverse of the eigenvalues of A.
- The matrices A^{-1} and A have the same eigenvectors.

Since the inverse of the smallest eigenvectors of A is obviously the largest eigenvalue of A^{-1}. Thus, if we want to determine the smallest possible eigenvalue of A we have to apply VON MISES' method to its inverse.

There is also an alternative to obtain the smallest eigenvalue with the VON MISES iteration. For this we apply the iteration procedure used previously but apply it to the inverse of A:

$$\mathbf{v}^{(t+1)} = A^{-1}\mathbf{v}^{(t)}.$$

We then multiply this equation from the left by A

$$A\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} \tag{5.21}$$

and interpret this as a linear system of equations with the matrix of coefficients A with the non-homogeneous vector $\mathbf{v}^{(t)}$ and the solution $\mathbf{v}^{(t+1)}$. If we apply the LU decomposition, see Chap. 2.4.1, to determine the solution of this system, we only need to compute the LU decomposition of A once when we calculate the inverse of A before the beginning of the VON MISES iteration. In this way we obtain a series of vectors with the property

$$\lim_{t\to\infty} \frac{v_\ell^{(t)}}{v_\ell^{(t+1)}} = \lambda_n, \tag{5.22}$$

with λ_n the smallest eigenvalue of A.

Improvement through spectral shift

Even without mathematical proof it becomes immediately evident that the convergence of the iteration (5.22) will be the better the larger the ratio

$$\frac{1/\lambda_n}{1/\lambda_{n-1}} = \frac{\lambda_{n-1}}{\lambda_n}$$

becomes. This can be exploited in the following way: Let us assume we know λ_0 to be a good approximation for the smallest eigenvalue λ_n. If we consider the matrix

$$A' = A - \lambda_0 I$$

of the matrix A all eigenvalues will be shifted by λ_0. The same holds for the eigenvalues λ_{n-1} and λ_n and

$$\frac{\lambda_{n-1} - \lambda_0}{\lambda_n - \lambda_0} > \frac{\lambda_{n-1}}{\lambda_n}.$$

Thus, we expect the limiting value

$$\lim_{t\to\infty} \frac{v_\ell^{(t)}}{v_\ell^{(t+1)}} + \lambda_0 = \lambda_n. \tag{5.23}$$

to converge more rapidly to the value (5.22).

5.2.3 Jacobi's method

This method permits to *solve the full eigenvalue problem for symmetrical matrices.* Let us consider a real matrix

$$A = [a_{ij}], \quad \text{with} \quad a_{ij} = a_{ji}.$$

It is a general property of real symmetrical matrices to have *only real eigenvalues.* The Jacobi method is based on an orthogonal transformation of the matrix A into a diagonal matrix D (see Subchap. 5.1.1):

$$U^T A U = D. \tag{5.24}$$

We are also aware of the fact that once the matrix U is known, the eigenvalue problem for A is completely solved:

- The diagonal elements of D are the eigenvalues of A:

$$\lambda_i = d_{ii}, \qquad i = 1, \dots, n.$$

- The columns of the transformation matrix U are the eigenvectors of A (up to a renormalization factor).

Iterative approximation of the transformation matrix U

We have the problem of finding the transformation matrix U. The Jacobi method approximates the transformation (5.24) by a sequence of similarity operations. Obviously, the transformation matrices U_t, $(t = 0, 1, 2, \dots)$ have to be orthogonal. We have instead of Eq. (5.24):

$$\begin{aligned}
U_0^T A U_0 &\equiv A^{(1)} \\
U_1^T A^{(1)} U_1 &\equiv A^{(2)} = U_1^T U_0^T A U_0 U_1 \\
&\vdots \\
U_t^T A^{(t)} U_t &\equiv A^{(t+1)} = U_t^T U_{t-1}^T \cdots U_0^T A U_0 \cdots U_{t-1} U_t,
\end{aligned}$$

where the single transformations have to be chosen in such a way that the sequence of matrices converges to:

$$\begin{aligned} \lim_{t\to\infty} A^{(t)} &\to D \\ \lim_{t\to\infty} U_0 U_1 \cdots U_{t-1} U_t &\to U. \end{aligned}$$

The matrices U_t used in the JACOBI method have the general form of *orthogonal rotation matrices*:

$$U_t(i,j,\varphi) = \begin{array}{c} \left(\begin{array}{ccccccccc} 1 & & \vdots & & & & \vdots & & \\ & \ddots & \vdots & & & & \vdots & & \\ \cdots & \cdots & \cos\varphi & \cdots & \cdots & \cdots & -\sin\varphi & \cdots & \cdots \\ & & \vdots & \ddots & & & \vdots & & \\ & & \vdots & & 1 & & \vdots & & \\ & & \vdots & & & \ddots & \vdots & & \\ \cdots & \cdots & \sin\varphi & \cdots & \cdots & \cdots & \cos\varphi & \cdots & \cdots \\ & & \vdots & & & & \vdots & \ddots & \\ & & \vdots & & & & \vdots & & 1 \end{array}\right) \begin{array}{l} \\ \\ i^{th} \text{ row} \\ \\ \\ \\ j^{th} \text{ row} \\ \\ \\ \end{array} \\ \begin{array}{cc} i^{th} \text{ column} & \qquad j^{th} \text{ column} \end{array} \end{array} \tag{5.25}$$

U_t is specified by three parameters: i and j indicate the rows and columns of the matrix for which the rotation matrix differs from the identity matrix. φ is the rotation angle. It is easy to show that $U(i,j,\varphi)$ is always orthogonal, independently of its arguments

$$U_t^T(i,j,\varphi)U_t(i,j,\varphi) = I.$$

We analyze in a first step the transformation

$$U^T(i_{t-1}, j_{t-1}, \varphi_{t-1}) A^{(t-1)} U(i_{t-1}, j_{t-1}, \varphi_{t-1}) \quad \to \quad A^{(t)} \tag{5.26}$$

which results, written in components, in:

$$a_{k\ell}^{(t)} = \sum_{m=1}^{n} \sum_{m'=1}^{n} u_{mk} u_{m'\ell} a_{mm'}^{(t-1)}. \tag{5.27}$$

It follows from the above equation that the matrix $A^{(t-1)}$ retains its symmetry during the transformation and we have

$$a_{k\ell}^{(t)} = a_{\ell k}^{(t)}.$$

We now evaluate Eq. (5.27) for values of k and ℓ which are are neither equal to i nor to j. Under these conditions the components of the rotation matrix (5.25) are KRONECKER deltas and we obtain

$$a_{kl}^{(t)} = \sum_{m=1}^{n} \sum_{m'=1}^{n} \delta_{mk} \delta_{m'\ell} a_{mm'}^{(t-1)} = a_{kl}^{(t-1)}.$$

We conclude: *The transformation leaves all the elements of A unchanged which do not contain the indices i or j.*

The so far undetermined elements in the i^{th} and j^{th} row and column can be determined from Eq. (5.27):

$$\begin{aligned} \ell = i, k = 1, \dots, n \text{ with } k \neq i, j : a_{ki}^{(t)} &= a_{ik}^{(t)} = a_{ki}^{(t-1)} \cos\varphi + a_{kj}^{(t-1)} \sin\varphi \quad (5.28) \\ \ell = j, k = 1, \dots, n \text{ with } k \neq i, j : a_{kj}^{(t)} &= a_{jk}^{(t)} = a_{kj}^{(t-1)} \cos\varphi + a_{ki}^{(t-1)} \sin\varphi \quad (5.29) \end{aligned}$$

Finally, we find for the elements $a_{ii}^{(t)}$, $a_{jj}^{(t)}$, and $a_{ij}^{(t)} = a_{ji}^{(t)}$ the following transformations:

$$a_{ii}^{(t)} = a_{ii}^{(t-1)} \cos^2\varphi + 2a_{ij}^{(t-1)} \cos\varphi \sin\varphi + a_{jj}^{(t-1)} \sin^2\varphi, \tag{5.30}$$

$$a_{jj}^{(t)} = a_{jj}^{(t-1)} \cos^2\varphi - 2a_{ij}^{(t-1)} \cos\varphi \sin\varphi + a_{ii}^{(t-1)} \sin^2\varphi, \tag{5.31}$$

$$a_{ij}^{(t)} = a_{ij}^{(t-1)} \left(\cos^2\varphi - sin^2\varphi\right) + \left[a_{jj}^{(t-1)} - a_{ii}^{(t-1)}\right] \cos\varphi \sin\varphi. \tag{5.32}$$

Choice of the parameters i, j, and φ

It is the purpose of each of these orthogonal transformations of matrix A that the sequence of matrices

$$A^{(1)}, A^{(2)}, \dots, A^{(t-1)}, A^{(t)}, A^{(t+1)}, \dots$$

approaches iteratively the diagonal form. A good measure for the convergence of this iteration is clearly the sum S of the squares of the non-diagonal elements of the matrix $A^{(t)}$:

$$S^{(t)} = 2 \sum_{m=1}^{n-1} \sum_{m'=m+1}^{n} \left[a_{mm'}^{(t)}\right]^2 .$$

Thus, if we perform during a Jacobi iteration the transformation

$$U_{t-1}^T A^{(t-1)} U_{t-1} \quad \rightarrow \quad A^{(t)}$$

we can consider it to be successful if

$$S^{(t-1)} > S^{(t)},$$

where $S^{(t-1)}$ and $S^{(t)}$ are the sums of the squares of the non-diagonal elements of the matrices $A^{(t-1)}$ and $A^{(t)}$, respectively.

It is possible to prove that the difference between $S^{(t-1)}$ and $S^{(t)}$ is given by

$$S^{(t-1)} - S^{(t)} = 2\left\{\left[a_{ij}^{(t-1)}\right]^2 - \left[a_{ij}^{(t)}\right]^2\right\}. \tag{5.33}$$

Thus, this difference is only determined by the matrix elements a_{ij} before and after the transformation.

Equation (5.33) suggests a strategy for the best choice of the free parameters i, j, and φ during the iteration: *The decrease of S is stronger*

- *if the absolute value of $a_{ij}^{(t-1)}$ is as large as possible.*
- *if the transformation reduces $a_{ij}^{(t)}$ to zero.*

If i and j are chosen as the indices of the largest off-diagonal elements of A before each iteration,[2] the second condition can be satisfied through an appropriate choice of the rotation angle φ. Setting Eq. (5.32) to zero results in:

$$\tan 2\varphi = \frac{2a_{ij}^{(t-1)}}{a_{ii}^{(t-1)} - a_{jj}^{(t-1)}}. \tag{5.34}$$

For $a_{ii}^{(t-1)} = a_{jj}^{(t-1)}$ the rotation angle is $\varphi = \pi/4$.

If we choose the parameters of the rotation matrix (5.25) before each iteration according to the rules outlined above we will obtain a sequence of monotonously decreasing sums $S^{(t)}$

$$S^{(0)} > S^{(1)} > S^{(2)} > \cdots > S^{(t)} > \cdots$$

and this guarantees a faster approach toward the required diagonal form.

After having determined the eigenvalues we have to find the eigenvectors for a complete solution of the eigenvalue problem. They are contained in the product

$$B^{(t-1)} = U_0 U_1 U_2 \cdots U_{t-2} U_{t-1} \tag{5.35}$$

of the single transformation matrices. It is, again, possible to prove that the multiplication of the matrix B with a rotation matrix $U_t(i,j,\varphi)$ changes only the elements of the i^{th} and j^{th} column of B:

$$\left[B^{(t-1)}U_t(i,j,\varphi)\right]_{ki} = b_{ki}\cos\varphi + b_{kj}\sin\varphi, \tag{5.36}$$

$$\left[B^{(t-1)}U_t(i,j,\varphi)\right]_{kj} = b_{kj}\cos\varphi - b_{ki}\sin\varphi, \tag{5.37}$$

for $k = 1, \ldots, n$. Two remarks:

1. Jacobi iteration programs usually calculate normalized eigenvectors which are, obviously, defined only up to a (multiplicative) constant (–1).

2. The effect of the non-uniqueness of the eigenvectors is even stronger in case of degenerate eigenvalues. For instance, in the case of a two times degenerate eigenvalue this means that each linear combination of the two eigenvectors returned by a Jacobi iteration program for this eigenvalue is still an eigenvector to the same eigenvalue.

5.2.4 An application

Let us consider a system of n point masses m_i, $i = 1, \ldots, n$ coupled by springs, as is shown schematically in Fig. 5.1. We will make the following assumptions:

- The restoring forces are so large (the masses so small) that the gravitational force can be neglected.

- The oscillation amplitudes are small compared to the distance between the particles at rest.

[2] Indeed, finding the largest off-diagonal element before each matrix transformation is very time consuming. For this reason, most programs employ a simplified method.

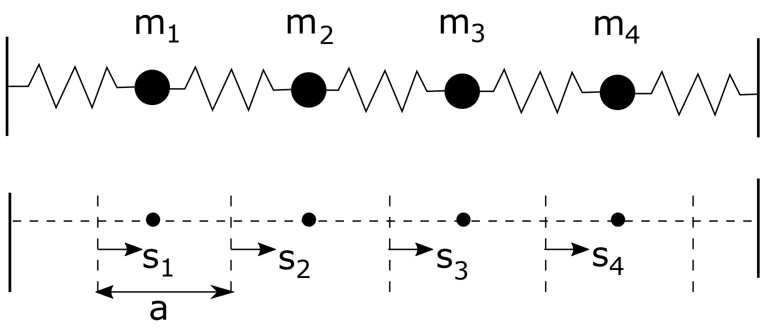

Figure 5.1: A system of point masses.

In the following, a is the distance between two point masses at rest; s_i is the instantaneous displacement of the masses from their equilibrium position. We assume that the forces between the masses are proportional to their distance – *harmonic approximation.* The factor of proportionality between the distance and the force is the *force constant* D. If such a system is put into oscillation and left alone, the equation of motion for each mass m_i is given by

$$m_i\ddot{s}_i + D[a - s_{i-1}(t) + s_i(t)] - D[a - s_i(t) + s_{i+1}(t)] = 0.$$

Here, m_i is the mass of the i^{th} particle and $s_i(t)$, $s_{i-1}(t)$, and $s_{i+1}(t)$ are the instanteneous elongations of the i^{th} point and of its left and right neighbors. In the end, we obtain a system of n coupled ordinary differential equations of second order:

$$\begin{aligned} m_1\ddot{s}_1 + 2Ds_1 - Ds_2 &= 0, \\ m_i\ddot{s}_i - Ds_{i-1} + 2Ds_i - Ds_{i+1} &= 0, \quad i = 2,3,\ldots,n-1, \\ m_n\ddot{s}_n - Ds_{n-1} + 2Ds_n &= 0. \end{aligned} \tag{5.38}$$

We restrict ourselves to small elongations and the system (5.38) can be reduced to a homogeneous, linear system of equations. The ansatz

$$s_i(t) = \frac{b_i}{\sqrt{m_i}} e^{i\omega t}$$

results in

$$\begin{aligned} \left(\frac{2D}{m_1} - \omega^2\right) b_1 - \frac{D}{\sqrt{m_1 m_2}} b_2 &= 0 \\ -\frac{D}{\sqrt{m_i m_{i-1}}} b_{i-1} + \left(\frac{2D}{m_i} - \omega^2\right) b_i - \frac{D}{\sqrt{m_i m_{i+1}}} b_2 &= 0, \quad i = 2,3,\ldots,n-1, \\ -\frac{D}{\sqrt{m_n m_{n-1}}} b_{n-1} + \left(\frac{2D}{m_n} - \omega^2\right) b_n &= 0, \end{aligned}$$

which can be written as a *regular eigenvalue problem*:

$$\begin{pmatrix} \frac{2D}{m_1} & -\frac{D}{\sqrt{m_1 m_2}} & \\ \ddots & \ddots & \ddots \\ -\frac{D}{\sqrt{m_i m_{i-1}}} & \frac{2D}{m_i} & -\frac{D}{\sqrt{m_i m_{i+1}}} \\ & \ddots & \ddots \\ & -\frac{D}{\sqrt{m_n m_{n-1}}} & \frac{2D}{m_n} \end{pmatrix} \begin{pmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{pmatrix} = \lambda \begin{pmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{pmatrix} \tag{5.39}$$

with $\omega^2 = \lambda$. The (real) eigenvalues of the above *symmetrical, tridiagonal matrix* (see page 30) are the squares of the eigenfrequencies of the oscillating system (smallest frequency= zero harmonics plus $n-1$ over-tunes). The eigenvalues of this system can now be rather easily calculated using a JACOBI iteration method. Nevertheless, as the matrix is tridiagonal, one would be well advised to look for a numerical routine tuned for tridiagonal matrices because of faster execution and most likely better accuracy.

5.2.5 Extended symmetrical eigenvalue problems

Many interesting problems in the field of technical applications cannot be represented as an eigenvalue problem of the type (5.5) but rather as a homogeneous linear system of equations of the type

$$(A - \lambda S)\mathbf{x} = 0, \tag{5.40}$$

where A is a symmetrical matrix and S is also a symmetrical matrix which is also positive definite (in some applications S is called a *structure matrix*). Equation (5.40) establishes a so-called *extended eigenvalue problem*. The obvious question is now how to reduce Eq. (5.40) to a form like

$$(A' - \lambda I)\mathbf{x} = 0.$$

This can quite easily be accomplished by multiplying Eq. (5.40) from the left by the inverse of S, namely S^{-1}

$$\left(S^{-1}A - \lambda I\right)\mathbf{x} = 0,$$

with the new matrix of coefficients $A' = S^{-1}A$.

Unfortunately, this straight forward idea is not very useful in practice, because the new matrix of coefficients A' has lost the important property of being symmetric because

$$[A']_{ij} = \sum_{k=1}^{n} s_{ik}^{-1} a_{kj},$$

is in general not identical to

$$[A']_{ji} = \sum_{k=1}^{n} s_{jk}^{-1} a_{ki}.$$

A better, albeit more complicated method is to reformulate the problem. The starting point is the so-called CHOLESKY decomposition of the symmetrical, positive definite matrix S:

$$S = LL^T. \tag{5.41}$$

Please note that this decomposition is very similar to the LU-decomposition described in Chap. 2, Eq. (2.11). The difference is that in the present case a symmetrical, positive definite matrix is represented as a product of a lower triangular matrix L and an upper triangular matrix L^T which is its transpose of L about the main diagonal. If we carry out the CHOLESKY decomposition, the rest of the procedure is relatively straight forward: Eq. (5.41) is inserted in the extended eigenvalue equation (5.40) and this results in

$$\left(A - \lambda LL^T\right)\mathbf{x} = 0,$$

or

$$\left[A\left(L^T\right)^{-1} - \lambda L\right]\left(L^T\mathbf{x}\right) = 0,$$

and

$$\left[\underbrace{L^{-1}A\left(L^T\right)^{-1}}_{C} - \lambda I\right]\left(L^T\mathbf{x}\right) = 0.$$

We get in a compact notation[3]:

$$(C - \lambda I)\mathbf{y} = 0, \quad \text{with} \quad C = L^{-1}A(L^{-1})^T \quad \text{and} \quad \mathbf{y} = L^T\mathbf{x}. \tag{5.42}$$

The eigenvalues of the matrix C are identical to those of A in Eq. (5.40) and it is easy to prove that this matrix is symmetrical. From the eigenvectors $\mathbf{y}$ of the system (5.42) the eigenvectors $\mathbf{x}$ of Eq. (5.40) can be constructed using

$$\mathbf{x} = \left(L^{-1}\right)^T\mathbf{y}. \tag{5.43}$$

We are now in a position to derive the formulas required for programming the Cholesky decomposition. We start from Eq. (5.41) in its component representation:

$$s_{ij} = \sum_{k=1}^{n} \ell_{ik}\ell_{jk} = \sum_{k=1}^{j} \ell_{ik}\ell_{jk},$$

i.e. the evaluation of the elements s_{ij} is carried out column by column (index j) under the condition $i \le j$ because L is lower triangular matrix. We obtain immediately for the first column ($j = 1$)

$$s_{11} = \ell_{11}^2 \to \ell_{11} = \sqrt{s_{11}},$$

and

$$s_{i1} = \ell_{i1}\ell_{11} \to \ell_{i1} = \frac{s_{i1}}{\ell_{11}} \quad \text{for} \quad i = 2, \dots, n.$$

We obtain in analogue for the second column ($j = 2$)

$$s_{22} = \ell_{21}^2 + \ell_{22}^2 \to \ell_{22} = \sqrt{s_{22} - \ell_{21}^2},$$

and

$$s_{i2} = \ell_{i1}\ell_{21} + \ell_{i2}\ell_{22} \to \ell_{i2} = \frac{s_{i2} - \ell_{i1}\ell_{i2}}{\ell_{22}} \quad \text{for} \quad i = 3, \dots, n.$$

These equations lead us, without further work, to the general Cholesky formulas:

$$\begin{aligned} \ell_{jj} &= \sqrt{s_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2}, \\ \ell_{ij} &= \frac{s_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk}}{\ell_{jj}}, \quad \text{for} \quad i = j+1, \dots, n. \end{aligned} \tag{5.44}$$

Two remarks:

[3] We employed here the identity $(L^T)^{-1} = (L^{-1})^T$

- Problems will occur, obviously, whenever the argument of the square root becomes zero or negative during the CHOLESKY decomposition. This can a priori be prevented by checking whether or not the matrix S is positive definite. Thus, we will have to check in a program before the square root is calculated whether or not
$$\left(s_{jj} - \sum_{k=1}^{j-1} \ell_{jk}^2\right) > 0.$$
If this cannot be established, the matrix S is not positive definite and the calculation must be interrupted!

- An analysis of Eqs. (5.44) reveals that the s_{ij} values required to calculate the ℓ_{ij} are used only once in the course of the calculation: this opens the possibility to store the values of the L matrix (column by column) in the memory locations occupied by the S matrix with the disadvantage that the original S matrix gets lost.

5.2.6 More advanced algorithms

The JACOBI method works in principle extremely well for all real symmetrical matrices. Furthermore, most professional libraries offer even more performant programs which are clearly faster for matrices of higher order (> 20)[7]. The basic idea of these programs is to make use of a two-step method:

1. The symmetrical matrix A is reduced to a form which can be better handled using a *final* (i.e. non-iterative) method. This is typically a tridiagonal symmetrical form. The HOUSEHOLDER ALGORITHM is very often applied to accomplish this.

2. The eigenvectors and eigenvalues of A are calculated from this tridiagonal matrix using the so-called QR-algorithm or the QL-algorithm.

The interested reader is referred to Refs. [2, 1].

Bibliography

[1] Hermann, M.: Numerische Mathematik. de Gruyter, München, Wien (2011)

[2] Kressner, D.: Numerical Methods for General and Structured Eigenvalue Problems. Springer, Berlin, Heidelberg (2005)

[3] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[4] Varga, R.S.: Matrix Iterative Analysis, 2nd edn. Springer Series in Computational Mathematics, Vol. 27. Springer, Berlin, Heidelberg (2000)

Chapter 6

Numerical solution of transcendental equations

6.1 The problem

Let us consider a real-valued function $F(x)$. We seek the real values of x for which:

$$F(x) = 0. \tag{6.1}$$

The values of x that obey this relation are called *solutions*, *zeroes*, or *roots* of Eq. (6.1). If no analytical solution can be found, there are many methods available to treat this problem numerically. In this chapter we discuss a number of iterative solution methods, which can be split into the following two categories:

- Open-ended methods, where knowledge of the approximate location of the root is not required;
- Bracketing methods, where an upper and lower limit are used to restrict the algorithm and to guarantee convergence.

Note that we put the emphasis on *transcendental* equations in the title. This does not mean, however, that we wish to exclude *algebraic* equations of the type

$$F(x) = P_m(x) = \sum_{j=1}^{m} \alpha_j{}^{j-1} = 0.$$

Algebraic equations are considered to be a special case of transcendental equations. Many specialized methods exist to numerically determine the zeroes of algebraic equations. Examples include the methods of LOBATSCHEWSKI and GRAEFFE [4], p.60ff, and the Weierstrass method [3]. We will not discuss these methods within this lecture.

6.2 Open-ended iterative methods

6.2.1 Fixed point iteration

The equation $F(x) = 0$ can almost always be reduced to the form:

$$x = f(x), \tag{6.2}$$

in which case x is referred to as a *fixed point* of the function $f(x)$. From Eq. 6.2 we obtain a typical iteration algorithm for finding the roots:

$$\begin{aligned} x_0 & \quad \text{starting value} \\ x_1 &= f(x_0), \\ x_2 &= f(x_1), \\ &\vdots \\ x_{t+1} &= f(x_t) \quad \text{for } t = 0, 1, 2, \dots \end{aligned} \tag{6.3}$$

If this method converges, the iteration will approach the exact solution of the problem arbitrarily close (within roundoff errors). We have in fact:

$$\lim_{t\to\infty} x_t = x_{\text{exact}},$$

but it is not guaranteed that this method will converge. In fact, the convergence depends strongly on how Eq. (6.1) was reformulated into Eq. (6.2), as we will demonstrate in the following example.

Example 6.2.1. Let us imagine that we are looking for the roots of the function $F(x) = x^3 - x - 5$ in the vicinity of $x = 2$. We consider three possible reformulations of $F(x)$ in the form of Eq. (6.2):

$$\text{a)} \quad x = x^3 - 5, \quad \text{b)} \quad x = \frac{5}{x^2 - 1}, \quad \text{c)} \quad x = \sqrt[3]{x + 5}. \tag{6.4}$$

Results for the starting point $x_0 = 2$ are presented in the following table.

t	a)	b)	c)
0	2	2	2
1	3	1.6667	1.9129
2	22	2.8125	1.9050
3	10643	0.7236	1.9042
4	.	-10.4944	1.9042
⋮	⋮	⋮	⋮
	divergent	divergent	convergent

6.2.2 Convergence criteria and error estimates

In order to obtain the relation that ensures the iteration's convergence we start from Eq. (6.2) for the exact solution

$$x_{\text{exact}} = f(x_{\text{exact}}). \tag{6.5}$$

We obtain after the $(t+1)^{\text{th}}$ iteration:

$$x_{t+1} = f(x_t) - \delta_t, \tag{6.6}$$

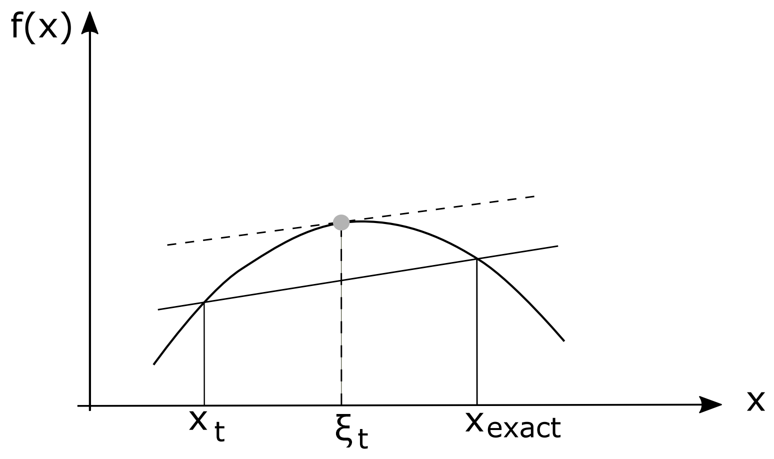

Figure 6.1: Illustration of the mean-value theorem of Eq. (6.8).

where δ_t represents the roundoff error in the numerical evaluation of the function $f(x)$ for $x = x_t$. The minus sign is an arbitrary convention. Subtracting Eq. (6.5) from Eq. (6.6) gives:

$$x_{\text{exact}} - x_{t+1} = f(x_{\text{exact}}) - f(x_t) + \delta_t. \tag{6.7}$$

Using the mean-value theorem (see Fig. 6.1) we can write:

$$\frac{f(x_{\text{exact}}) - f(x_t)}{x_{\text{exact}} - x_t} = f'(\xi_t), \quad \text{with} \quad \xi_t \in [x_t, x_{\text{exact}}]. \tag{6.8}$$

Inserting Eq. (6.8) into Eq. (6.7), the error at step $t+1$ can be related to the error at step t:

$$x_{\text{exact}} - x_{t+1} = f'(\xi_t)(x_{\text{exact}} - x_t) + \delta_t. \tag{6.9}$$

This equation can be reduced, step by step, into:

$$\begin{aligned} x_{\text{exact}} - x_{t+1} = {} & f'(\xi_t)f'(\xi_{t-1})\cdots f'(\xi_0)(x_{\text{exact}} - x_0) + \delta_t \\ & + f'(\xi_t)\delta_{t-1} \\ & + f'(\xi_t)f'(\xi_{t-1})\delta_{t-2} \\ & + \cdots \\ & + f'(\xi_t)f'(\xi_{t-1})\cdots f'(\xi_1)\delta_0. \end{aligned}$$

In order to *estimate the error* we will now use the following definition and assumption.

1. We consider the interval $\mathcal{I}$ comprising all values of x_t obtained during the iteration and x_{exact}, that is $x_0, x_1, \ldots, x_{t+1}, \ldots, x_{\text{exact}} \in \mathcal{I}$. The maximum value of the first derivative of the function within the interval $\mathcal{I}$ is denoted m:

$$m = \max_{\mathcal{I}} |f'(x)|. \tag{6.10}$$

2. We assume that the roundoff error does not depend on the index of the iteration and therefore:

$$\delta_t \approx \delta \quad \text{for } t = 0, 1, \ldots.$$

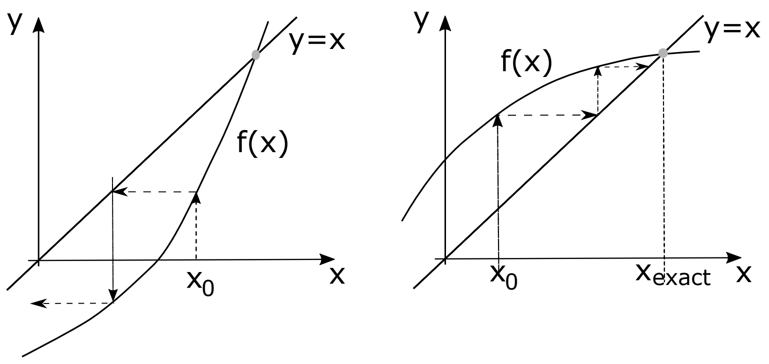

Figure 6.2: Convergence behavior of the iteration (6.3).

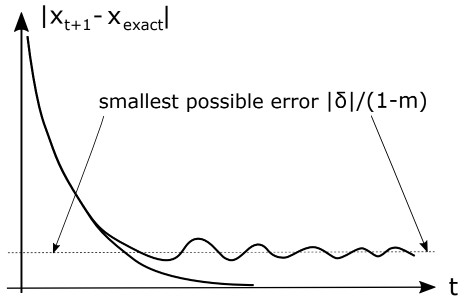

Figure 6.3: Error evolution of the iteration (6.3).

The consequence of this is:

$$|x_{\text{exact}} - x_{t+1}| \;\leq\; m^{t+1}|x_{\text{exact}} - x_0| + |\delta|(1 + m + m^2 + \cdots + m^t).$$

Taking the limit for $t \to \infty$ leads to:

$$\lim_{t\to\infty} |x_{\text{exact}} - x_{t+1}| \;\leq\; \underbrace{|x_{\text{exact}} - x_0| \lim_{t\to\infty} m^{t+1}}_{\text{methodological error}} + \underbrace{\lim_{t\to\infty} \frac{|\delta|}{1-m}}_{\text{roundoff error}}.$$

Both terms diverge for $m \geq 1$. Therefore, the iteration's convergence criterion is:

$$0 \leq m < 1. \tag{6.11}$$

If this relation is satisfied, we have:

$$\lim_{t\to\infty} |x_{\text{exact}} - x_{t+1}| \;\leq\; \frac{|\delta|}{1-m}. \tag{6.12}$$

This is a very important result, as it demonstrates that the iteration method is numerically stable: the size of the roundoff errors converges to a limiting value. These relations are illustrated in Figs. 6.2 and 6.3.

We seek an *estimate* of the error. In order to accomplish this we start from Eq. (6.9) with some modifications:

$$\begin{aligned}
x_{\text{exact}} - x_{t+1} &= f'(\xi_t)(x_{\text{exact}} - x_t) + \delta_t \\
&= f'(\xi_t)(x_{\text{exact}} - x_t + x_{t+1} - x_{t+1}) + \delta_t \\
(x_{\text{exact}} - x_{t+1})[1 - f'(\xi_t)] &= f'(\xi_t)(x_{x+t} - x_t) + \delta_t \\
(x_{\text{exact}} - x_{t+1}) &= \frac{f'(\xi_t)}{1 - f'(\xi_t)}(x_{t+1} - x_t) + \frac{\delta_t}{1 - f'(\xi_t)}.
\end{aligned}$$

If the iteration converges we obtain:

$$|x_{\text{exact}} - x_{t+1}| \;\leq\; \frac{m}{1-m}\,|x_{t+1} - x_t| + \frac{|\delta|}{1-m}. \tag{6.13}$$

If we were able to calculate the maximum value of the first derivative of the function $f(x)$ within the interval $\mathcal{I}$ and to estimate the value of δ accordingly, Eq. (6.13) could serve as a basis for an error control of the iterative method. We could iterate until the expression on the right hand side of Eq. (6.13) becomes smaller than a desired precision ε. We would only have to make sure that ε was never set to a value smaller than the limiting precision given by $|\delta|/(1-m)$. In practice it is most of the times impossible or too complicated to determine m and δ. In this case we can employ the more trivial criterion:

$$|x_{\text{exact}} - x_{t+1}| \;\leq\; |x_{t+1} - x_t|\,. \tag{6.14}$$

However, we have to keep in mind that Eq. (6.14) only holds for $m \leq 1/2$, and can be grossly wrong for $1/2 < m < 1$! Otherwise, Eq. (6.14) is only useful as long as the methodological error is larger than the roundoff error!

It is the purpose of the following example to illustrate some of the relations derived before: We want to determine numerically the zeroes of the function:

$$F(x) = a - x + (1-a)x^2.$$

It can easily be verified that this quadratic function has two roots:

$$x_1 = 1, \quad \text{and} \quad x_2 = \frac{a}{1-a}.$$

We approximate now the root x_1 with the help of the iteration

$$x_{t+1} = a + (1-a)x_t^2.$$

The first derivative of $f(x)$ at $x = 1$ is $2(1-a)$. This is also the maximum value of the derivative within the iteration interval, i.e. we have

$$m = |2(1-a)|.$$

On the basis of Eqs. (6.11) and (6.13) we expect that

- The iteration diverges for $m \geq 1$, i.e. for $a \leq 0.5$ or $a \geq 1.5$.
- The simplified error estimate (6.14) breaks down for $1/2 < m < 1$, i.e. for $a < 0.75$.

This can quite easily be verified by doing the necessary numerics.

6.2.3 The Newton-Raphson method

For differentiable functions $F(x)$, a more advanced technique is the so-called Newton-Raphson method [5, 6].[1] It is based on the definition of $f(x)$ as

$$f(x) = x - \frac{F(x)}{F'(x)}\,, \tag{6.15}$$

[1]This method is also referred to as the Newton method.

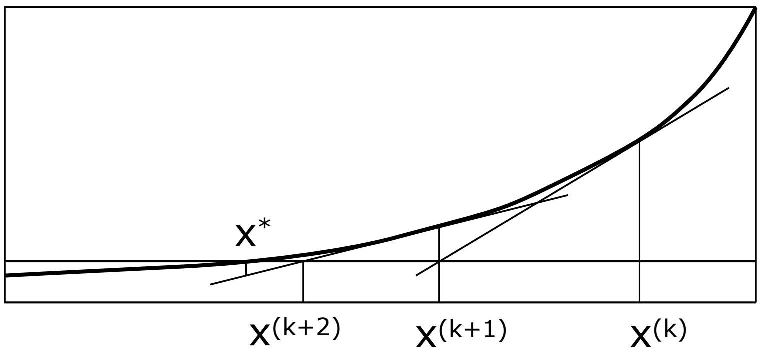

Figure 6.4: Graphical interpretation of the NEWTON-RAPHSON iteration.

which clearly satisfies $f(x) = x$ for $F(x) = 0$ as long as $F'(x) \neq 0$. Here $F'(x)$ denotes the derivative of $F(x)$ with respect to x. Eq. (6.15) allows the iteration

$$x_{t+1} = x_t - \frac{F(x_t)}{F'(x_t)} \; . \tag{6.16}$$

The graphical interpretation of this formula ('tangent method') is shown in Fig. 6.4. The convergence behavior of the iteration (6.16) highly depends on the form of the function $F(x)$ and on the choice of the starting point x_0. The routine can be regarded as converged if $|x_{t+1} - x_t| < \varepsilon$, where ε is the required accuracy.

We can again use Eq. (6.13) for an error estimate. Here, due to Eqs. (6.12) and (6.15), m is determined from:

$$m = \max_{\mathcal{I}} \frac{\mathrm{d}}{\mathrm{d}x}\left(x - \frac{F(x)}{F'(x)}\right) = \max_{[x_0, x_{\text{exact}}]} \frac{F(x)F''(x)}{[F'(x)]^2}.$$

According to the convergence criterion (6.11) that m must be smaller than one we can also state: It is difficult to apply the NEWTON-RAPHSON iteration if in the vicinity of the zero the slope of $F(x)$ is small. This can happen, for example, if two zeroes lie close to each other.

However, when the NEWTON-RAPHSON method works, its convergence is very fast, as can be demonstrated by the following example. We expand the function $F(x)$ around x_t in a TAYLOR series up to second order:

$$F(x) = F(x_t) + F'(x_t)(x - x_t) + \frac{1}{2}F''(x_t)(x - x_t)^2.$$

If x happens to be the exact solution we obtain by dividing by $F'(x_t)$:

$$0 = -\frac{F(x_t)}{F'(x_t)} - (x_{\text{exact}} - x_t) - \frac{1}{2}\frac{F''(x_t)}{F'(x_t)}(x_{\text{exact}} - x_t)^2.$$

The first term on the right hand side represents the correction term in the NEWTON-RAPHSON formula, i.e. we have:

$$0 = x_{t+1} - x_t - x_{\text{exact}} + x_t - \frac{1}{2}\frac{F''(x_t)}{F'(x_t)}(x_{\text{exact}} - x_t)^2,$$

and

$$(x_{\text{exact}} - x_{t+1}) = -\frac{1}{2}\frac{F''(x_t)}{F'(x_t)}(x_{\text{exact}} - x_t)^2.$$

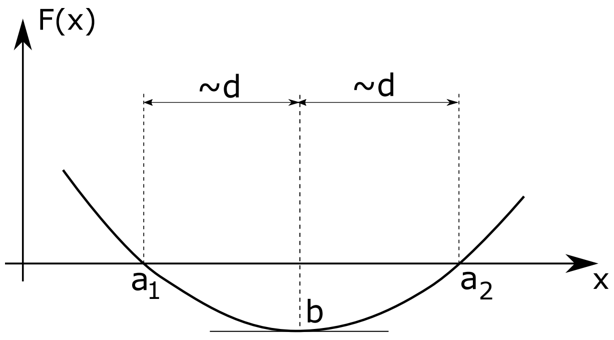

Figure 6.5: MACON's method.

Consequently, the absolute error of the solution in the $(t+1)^{\text{th}}$ iteration step is proportional to the square of the absolute error in the t^{th} iteration step. Thus, we can state: *the* NEWTON-RAPHSON *method converges quadratically.*

We briefly introduce the case of a non-linear system of equations of the form $F(x) = 0$ where $F(x) \in \mathbb{R}^N$ and $x \in \mathbb{R}^N$. In this case the iteration scheme is given by

$$x_{t+1} = x_t - J^{-1}(x_t)F(x_t)\ , \tag{6.17}$$

where

$$J(x) = \nabla_x F(x) = \begin{pmatrix} \frac{\partial F_1(x)}{\partial x_1} & \frac{\partial F_1(x)}{\partial x_2} & \cdots & \frac{\partial F_1(x)}{\partial x_N} \\ \frac{\partial F_2(x)}{\partial x_1} & \frac{\partial F_2(x)}{\partial x_2} & \cdots & \frac{\partial F_2(x)}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_N(x)}{\partial x_1} & \frac{\partial F_N(x)}{\partial x_2} & \cdots & \frac{\partial F_N(x)}{\partial x_N} \end{pmatrix} . \tag{6.18}$$

is the JACOBI matrix of $F(x)$.

If $F(x)$ is not differentiable, the *secant method* (see Sec. 6.2.5), *regula falsi* (see Sec. 6.3.2) or *stochastic methods* can be employed. We can also make use of the methods discussed in Chap. 7 to numerically calculate the derivatives in Eqs. (6.16) or (6.18). A more detailed discussion of other methods to solve transcendental equations numerically can be found in any textbook on numerical methods, see for instance Refs. [2, 1].

6.2.4 MACON's method

This method is employed when, for instance, pairs of zeroes sit very close to each other. In general localizing the minimum or maximum of the function between the two zeroes a_1 and a_2 is not really a problem. If we assume the value of the abscissas of the extremum b to be known with enough precision, we can expand $F(x)$ in a TAYLOR series around b:

$$F(x) = F(b) + (x-b)F'(b) + \frac{1}{2}(x-b)^2F''(b) + \cdots \tag{6.19}$$

We can further assume that, to a first approximation, the two zeroes a_1 and a_2 to be located symmetrically around b and have therefore a distance d from the extremum, such as:

$$F(b+d) \approx 0 \quad \text{and} \quad F(b-d) \approx 0.$$

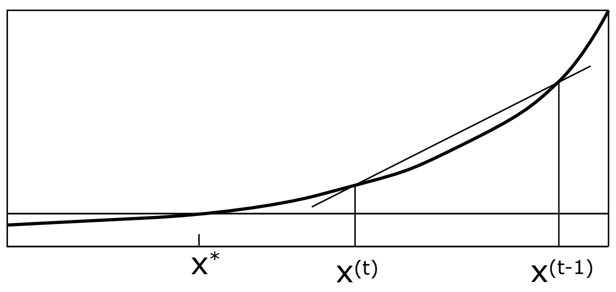

Figure 6.6: Graphical interpretation of the secant method.

We use this in Eq. (6.19) and get

$$0 \approx F(b) + \frac{1}{2}(+d)^2 F''(b), \qquad 0 \approx F(b) + \frac{1}{2}(-d)^2 F''(b).$$

These two equations are equivalent and, thus, d can be estimated as:

$$d = \pm\sqrt{-\frac{2F(b)}{F''(b)}}. \tag{6.20}$$

If we now start the Newton-Raphson iteration using $b - d$ or $b + d$ as starting points, we can most of the times avoid convergence problems.

6.2.5 The secant method

This method is closely related to the Newton-Raphson method and is often used when the analytical calculation of the derivative of the function $F(x)$ is not possible. In this case the differential ratio in Eq. (6.16) is replaced by the difference ratio:

$$x_{t+1} = x_t - F(x_t)\frac{x_t - x_{t-1}}{F(x_t) - F(x_{t-1})}. \tag{6.21}$$

Graphically this means that the Newton-Raphson tangent method is replaced by a *secant method* (see Fig. 6.6).

Note that instead of a single initial value, the secant method requires two initial values, x_0 and x_1. The method can be used as an open-ended iteration, which means that the root does not need to be located between the two initial values. If the root is located outside the interval $[x_0, x_1]$, however, the method may not converge, as is the case for the open-ended iteration methods with single initial values.

6.3 Bracketing methods

One way to avoid the possible convergence problems that are inherent to open-ended iteration algorithms is to use a bracketed method, which means that the root is located in the interval bounded by the initial values. If the bracketing is maintained during the iteration and the initial interval contains a single root, the algorithm is guaranteed to converge. Obviously, choosing such initial values required either knowledge about the approximate location of the root, or a separate algorithm for selecting the initial interval.

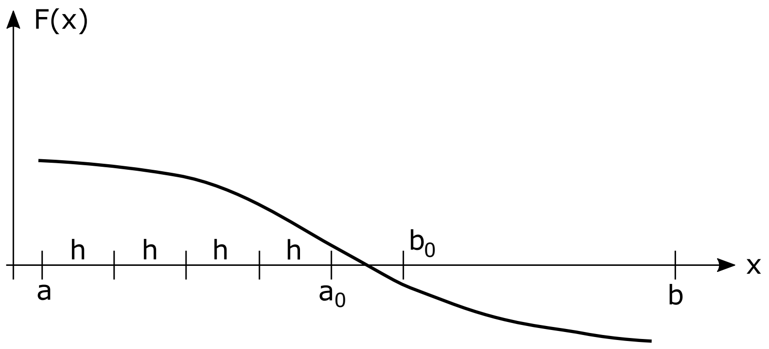

Figure 6.7: Gross search for zeroes, bisection method.

6.3.1 The bisection method

In order to identify the real zeroes of the function $F(x)$ within a given interval $[a, b]$, a *gross localization* of the zeroes within a given interval width h is carried out, starting from a.

Let us assume the first zero to lie within the interval $[a_0, b_0]$ (see Fig. 6.7). This means that inside this interval the function $F(x)$ changes sign, i.e. we have:

$$F(a_0)F(b_0) < 0.$$

Once an interval with this property was identified, the actual interval bisection takes place. First of all, the interval $[a_0, b_0]$ is divided by two:

$$x_0 = \frac{a_0 + b_0}{2}.$$

This leaves us with three possibilities:

1. $F(x_0) = 0$: x_0 is (exactly) the desired zero. (This case will rarely occur due to the roundoff error in the evaluation of the function).

2. $F(a_0)F(x_0) < 0$. In this case the desired zero lies within the *left* half interval $[a_0, x_0]$.

3. $F(a_0)F(x_0) > 0$. In this case the desired zero lies within the *right* half interval $[x_0, b_0]$.

In case (1) the zero was found; in cases (2) and (3) the left or the right half interval will be divided further by two. This process will be iterated, until the zero has been determined with required accuracy, i.e. until the current interval width h after t divisions ($[a_t, b_t]$) is smaller than the precision with which we wish to determine the zero.

As at least one root is always located between the values a_t and b_t, this method is guaranteed to converge. That means that if a zero is located within the interval $[a_0, b_0]$, it can be calculated with the desired accuracy, down to machine precision. Therefore, the only source of uncertainty lies in the choice of the interval width h for the gross search. In particular, a sign change does not always signify the presence of a single root, and neither does the absence of a sign change necessarily mean that no roots are located in the interval, as is demonstrated in Fig. 6.8:

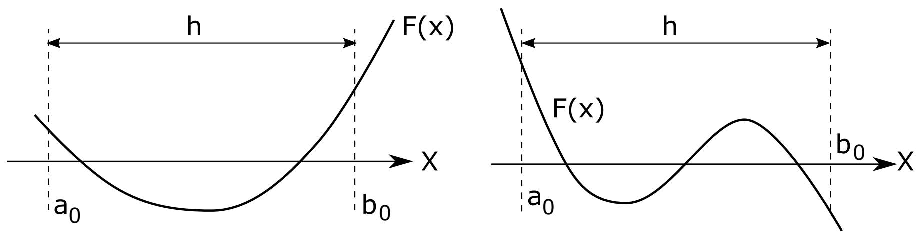

Figure 6.8: Problems typical of the bisection method.

- The left-hand panel of Fig. 6.8 shows a situation when there is no sign change within the interval $[a_0, b_0]$ because it contains two zeroes. This will happen whenever an interval $[a_0, b_0]$ contains an *even* number of zeroes.
- The right-hand panel of Fig. 6.8 shows a situation when the sign change is the result of an *odd* number of zeroes.

In both cases, the interval width h of the gross search has been chosen too large. This example demonstrates that h acts as a resolution limit for this method: *In general we can determine only those zeroes, which are separated by a distance which is larger than the initial width h.*

6.3.2 Regula Falsi

Using a procedure similar to the bisection method, the secant method can be modified in order to guarantee convergence. Choosing the values a_0 and b_0 as the initial values of the secant method, we know that at least one root is located in the initial interval if $F(a_0)F(b_0) < 0$. In the Regula Falsi algorithm, the secant method is applied not to subsequent estimates x_{t-1} and x_t, but to a_t and b_t,

$$x_{t+1} = b_t - F(b_t)\frac{b_t - a_t}{F(b_t) - F(a_t)}. \tag{6.22}$$

To maintain the bracketing, the subsequent values a_{t+1} and b_{t+1} are chosen such that

$$a_{t+1} = \begin{cases} x_{t+1} & \text{if } F(x_{t+1})F(a_t) > 0 \text{ and } F(x_{t+1})F(b_t) < 0 \\ a_t & \text{if } F(x_{t+1})F(a_t) < 0 \text{ and } F(x_{t+1})F(b_t) > 0, \end{cases} \tag{6.23}$$

and

$$b_{t+1} = \begin{cases} x_{t+1} & \text{if } F(x_{t+1})F(a_t) < 0 \text{ and } F(x_{t+1})F(b_t) > 0 \\ b_t & \text{if } F(x_{t+1})F(a_t) > 0 \text{ and } F(x_{t+1})F(b_t) < 0. \end{cases} \tag{6.24}$$

This procedure guarantees that at least one root is located in the interval $[a_{t+1}, b_{t+1}]$, which is the root that the algorithm will converge to.

6.3.3 Higher-order interpolation methods

Another way to interpret the Regula Falsi method is that the function between the values a_t and b_t is interpolated using a linear function. The root of the function $F(x)$ can also be obtained by interpolating the function around the root with a higher order function, as long as an analytical expression for the root of the interpolation function is available. For example, using a parabolic interpolation leads to the Muller method. Expanding the function around x_t to second order leads to the parabolic function $F(x) \approx A(x - x_t)^2 + B(x - x_t) + C$. To calculate the coefficients, we pick two points x_{t-2} and x_{t-1} on opposite sides of x_t where we equate the function F to the parabola, leading to the equations

$$\begin{aligned} F(x_{t-2}) &= A(x_{t-2} - x_t)^2 + B(x_{t-2} - x_t) + C \\ F(x_{t-1}) &= A(x_{t-1} - x_t)^2 + B(x_{t-1} - x_t) + C \\ F(x_t) &= c. \end{aligned} \tag{6.25}$$

Solving the equations for A, B and C gives

$$\begin{aligned} A &= \frac{(x_{t-1} - x_t)\left(F(x_{t-2}) - F(x_{t-1})\right) - (x_{t-2} - x_t)\left(F(x_{t-1}) - F(x_t)\right)}{(x_{t-2} - x_t)(x_{t-1} - x_t)(x_{t-2} - x_{t-1})} \\ B &= \frac{(x_{t-2} - x_t)^2\left(F(x_{t-1}) - F(x_t)\right) - (x_{t-1} - x_t)^2\left(F(x_{t-2}) - F(x_t)\right)}{(x_{t-2} - x_t)(x_{t-1} - x_t)(x_{t-2} - x_{t-1})} \\ C &= F(x_t). \end{aligned} \tag{6.26}$$

The roots of the parabolic interpolation is the given by

$$x_{t+1} - x_t = \frac{2C}{-B \pm \sqrt{B^2 - 4AC}}. \tag{6.27}$$

In order to stay as close as possible to the previous estimate x_t, we pick the sign in the denominator to minimize the difference,

$$x_{t+1} = x_t - \frac{2C}{B + \operatorname{sign}(B)\sqrt{B^2 - 4AC}}. \tag{6.28}$$

One of the advantages of using the Muller method is that it can also be used for finding roots in the complex plane.

Bibliography

[1] Burden, R.L., Faires, J.D.: Numerical Analysis. PWS-Kent Publishing Comp., Boston (1993)

[2] Jacques, I., Judd, C.: Numerical Analysis. Chapman and Hall, London (1987)

[3] Kerner, I.: Ein gesamtschrittverfahren zur berechnung der nullstellen von polynomen. Numer. Math. **8**, 290–294 (1966). DOI 10.1007/BF02162564

[4] Poloshi, G.N.: Mathematisches Praktikum. Teubner Verlag, Leipzig (1963)

[5] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[6] Ralston, A., Rabinowitz, P.: A First Course in Numerical Analysis, 2nd edn. Dover, New York (2001)

Chapter 7

Numerical differentiation

7.1 Introduction

This chapter is the first of two systematic introductions to the numerical treatment of differential equations. Differential equations and, thus, derivatives and integrals are of eminent importance in the modern formulation of natural sciences and in particular of physics. Very often the complexity of the expressions involved does not allow an analytical approach, although modern *symbolic* software can ease a physicists life significantly. Thus, in many cases a numerical treatment is unavoidable and one should be prepared.

We introduce here the notion of finite differences as a basic concept of numerical differentiation [4, 7, 2]. In contrast, the next chapter will deal with the concepts of numerical quadrature. Together, these two chapters will set the stage for a comprehensive discussion of algorithms designed to solve numerically differential equations. In particular, the solution of ordinary differential equations will always be based on an integration.

This chapter is composed of four sections. The first repeats some basic concepts of calculus and introduces formally finite differences. The second formulates approximates to derivatives based on finite differences, while the third section includes a more systematic approach based on an operator technique. It allows an arbitrarily close approximation of derivatives with the advantage that the expressions discussed in this section can immediately be applied to the problems at hand. The chapter is concluded with a discussion of some additional aspects.

7.2 Finite differences

Let us consider a smooth function $f(x)$ on the finite interval $[a, b] \subset \mathbb{R}$ of the real axis. The interval $[a, b]$ is divided into $N - 1 \in \mathbb{N}$ equally spaced sub-intervals of the form $[x_i, x_{i+1}]$ where $x_1 = a$, $x_N = b$. Obviously, x_i is then given by

$$x_i = a + (i-1)\frac{b-a}{N-1}, \quad i = 1, \ldots, N \ . \tag{7.1}$$

We introduce the distance h between two grid-points x_i by:

$$h = x_{i+1} - x_i = \frac{b-a}{N-1}, \quad \forall i = 1, \ldots, N-1 \ . \tag{7.2}$$

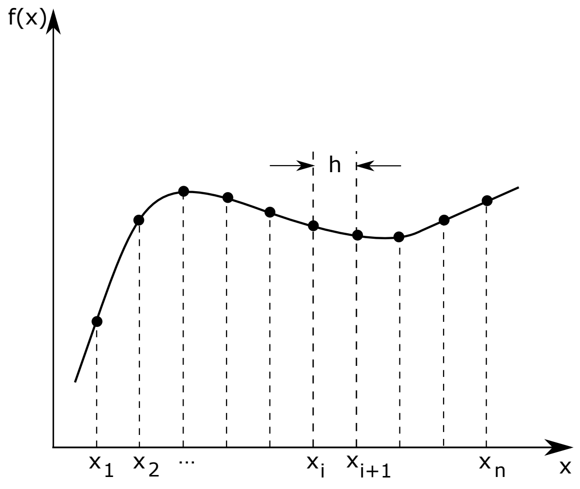

Figure 7.1: We define equally spaced grid-points x_i on a finite interval on the real axis in such a way that the function $f(x)$ is sufficiently well approximated by its functional values $f(x_i)$ at these grid-points.

For the sake of a more compact notation we restrict our discussion to equally spaced grid-points keeping in mind that the extension to arbitrarily spaced grid-points by replacing h by h_i is straight forward and leaves the discussion essentially unchanged.

Note that the number of grid-points and, thus, their distance h, has to be chosen in such a way that the function $f(x)$ can be sufficiently well approximated by its function values $f(x_i)$ as indicated in Fig. 7.1. We understand by *sufficiently well approximated* that some interpolation scheme in the interval $[x_i, x_{i+1}]$ will reproduce the function $f(x)$ within a required accuracy. In cases where the function is strongly varying within some sub-interval $[c, d] \subset [a, b]$ and is slowly varying within $[a, b] \setminus [c, d]$ it might be advisable to use variable grid-spacing in order to reduce the computational cost of the procedure.

We introduce the following notation: The function value of $f(x)$ at the grid-point x_i will be denoted by $f_i \equiv f(x_i)$ and its n-th derivative:

$$f_i^{(n)} \equiv f^{(n)}(x_i) = \left.\frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n}\right|_{x=x_i} . \tag{7.3}$$

Furthermore, we define for arbitrary $\xi \in [x_i, x_{i+1})$

$$f_{i+\varepsilon}^{(n)} = f^{(n)}(\xi) , \tag{7.4}$$

where $f_{i+\varepsilon}^{(0)} \equiv f_{i+\varepsilon}$ and ε is chosen to give:

$$\xi = x_i + \varepsilon h , \qquad \varepsilon \in [0, 1) . \tag{7.5}$$

Let us remember some basics from calculus. The first derivative, denoted $f'(x)$ of a function $f(x)$ which is smooth within the interval $[a, b]$, i.e. $f(x) \in \mathcal{C}^\infty[a, b]$ for arbitrary $x \in [a, b]$, is defined as

$$\begin{aligned} f'(x) &:= \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} \\ &= \lim_{h\to 0} \frac{f(x)-f(x-h)}{h} \\ &= \lim_{h\to 0} \frac{f(x+h)-f(x-h)}{2h} \; . \end{aligned} \tag{7.6}$$

However, it is impossible to draw numerically the limit $h \to 0$ as discussed in Sec. 1.3, Eq. (1.6). This manifests itself in a non-negligible error due to subtractive cancellation.

This problem is circumvented by the use of TAYLOR's theorem. It states that if there is a function which is $(n+1)$-times continuously differentiable on the interval $[a, b]$ then $f(x)$ can be expressed in terms of a series expansion at point $x_0 \in [a, b]$:

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k + \frac{f^{(n+1)}[\zeta(x)]}{(n+1)!}(x-x_0)^{n+1}, \quad \forall x \in [a,b] \; . \tag{7.7}$$

Here, $\zeta(x)$ takes on a value between x and x_0.[1] The last term on the right hand side of Eq. (7.7) is commonly referred to as *truncation error*. (A more general definition of this error was given in Sec. 1.1.)

We introduce now the finite difference operators

$$\Delta_+ f_i = f_{i+1} - f_i \; , \tag{7.8a}$$

as the *forward difference*,

$$\Delta_- f_i = f_i - f_{i-1} \; , \tag{7.8b}$$

as the *backward difference*, and

$$\Delta_c f_i = f_{i+1} - f_{i-1} \; , \tag{7.8c}$$

as the *central difference*.[2] The derivative of $f(x)$ can now be approximated with the help of TAYLOR's theorem (7.7). In a first step we consider (restricting to third order in h)

$$\begin{aligned} f_{i+1} &= f(x_i) + hf'(x_i) + \frac{h^2}{2}f''(x_i) + \frac{h^3}{6}f'''[\zeta(x_i+h)] \\ &= f_i + hf_i' + \frac{h^2}{2}f_i'' + \frac{h^3}{6}f_{i+\varepsilon_\zeta}''' \; , \end{aligned} \tag{7.9a}$$

with $f_{i+1} \equiv f(x_i + h)$. Here ε_ζ is the fractional part ε which has to be determined according to $\zeta(x_i + h)$. In analogue we find for f_{i-1}

$$f_{i-1} = f_i - hf_i' + \frac{h^2}{2}f_i'' - \frac{h^3}{6}f_{i+\varepsilon_\zeta}''' \; . \tag{7.9b}$$

Solving Eqs. (7.9) for the derivative f_i' leads directly to the definition of finite difference derivatives.

[1] Note that for $x_0 = 0$ the series expansion (7.7) is referred to as MCLAURIN series.

[2] Please note that the symbols Δ_+, Δ_-, and Δ_c in Eqs. (7.8) are linear operators acting on f_i. For a basic introduction to the theory of linear operators see for instance [9, 1].

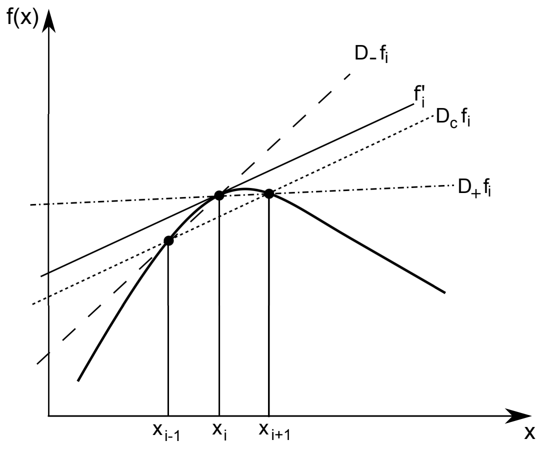

Figure 7.2: Graphical illustration of different finite difference derivatives. The solid line labeled f'_i represents the real derivative for comparison.

7.3 Finite difference derivatives

We define the *finite difference derivative* or difference approximations

$$D_+f_i = \frac{\Delta_+f_i}{h} = \frac{f_{i+1}-f_i}{h}\ , \tag{7.10a}$$

as the *forward difference derivative*,

$$D_-f_i = \frac{\Delta_-f_i}{h} = \frac{f_i-f_{i-1}}{h}\ , \tag{7.10b}$$

as the *backward difference derivative*, and

$$D_cf_i = \frac{\Delta_cf_i}{2h} = \frac{f_{i+1}-f_{i-1}}{2h}\ , \tag{7.10c}$$

as the *central difference derivative*.[3] A graphical interpretation of these expressions is straight forward and is presented in Fig. 7.2.

Using the above definitions (7.10) together with the expansions (7.9) we obtain

$$\begin{aligned} f'_i &= D_+f_i - \frac{h}{2}f''_i - \frac{h^2}{6}f'''_{i+\varepsilon\zeta} \\ &= D_-f_i + \frac{h}{2}f''_i - \frac{h^2}{6}f'''_{i+\varepsilon\zeta} \\ &= D_cf_i - \frac{h^2}{6}f'''_{i+\varepsilon\zeta}\ . \end{aligned} \tag{7.11}$$

We observe that in the central difference approximation of f'_i the truncation error scales like h^2 while it scales like h in the other two approximations; thus the central difference approximation should have the smallest methodological error. Note that

[3]The central difference derivative is related to the forward and backward difference derivatives via:

$$D_c = \frac{1}{2}(D_+ + D_-).$$

the error is usually not dominated by the derivatives of $f(x)$ since we assumed that $f(x)$ is a smooth function and sufficiently well approximated on the grid within $[a, b]$. Furthermore we have to emphasize that the central difference approximation is essentially a *three point* approximation, including f_{i-1}, f_i and f_{i+1}, although f_i cancels. Thus, we can improve our approximation by taking even more grid-points into account. For instance, we could combine the above finite difference derivatives. Let us prepare this step by expanding Eqs. (7.9) to higher order derivatives. We then obtain for the forward difference derivative

$$D_+ f_i = f_i' + \frac{h}{2} f_i'' + \frac{h^2}{6} f_i''' + \frac{h^3}{24} f_i^{IV} + \frac{h^4}{120} f_i^V + \dots , \tag{7.12}$$

for the backward difference derivative

$$D_- f_i = f_i' - \frac{h}{2} f_i'' + \frac{h^2}{6} f_i''' - \frac{h^3}{24} f_i^{IV} + \frac{h^4}{120} f_i^V \mp \dots , \tag{7.13}$$

and, finally, for the central difference derivative

$$D_c f_i = f_i' + \frac{h^2}{6} f_i''' + \frac{h^4}{120} f_i^V + \dots . \tag{7.14}$$

In order to improve the method we have to combine $D_+ f_i$, $D_- f_i$ and $D_c f_i$ from different grid-points in such a way that at least the terms proportional to h^2 cancel. This can be achieved by observing that[4]

$$8 D_c f_i - D_c f_{i-1} - D_c f_{i+1} = 6 f_i' - \frac{h^4}{5} f_{i+\varepsilon\zeta}^V , \tag{7.15}$$

which gives

$$\begin{aligned} f_i' &= \frac{1}{6} (8 D_c f_i - D_c f_{i+1} - D_c f_{i-1}) + \frac{h^4}{30} f_i^V \\ &= \frac{1}{12h} (f_{i-2} - 8 f_{i-1} + 8 f_{i+1} - f_{i+2}) + \frac{h^4}{30} f_i^V . \end{aligned} \tag{7.16}$$

Note that this simple combination yields an improvement of two orders in h ! One can even improve the approximation in a similar fashion by simply calculating the derivative from even more points, for instance $f_{i\pm 3}$.

7.4 A systematic approach - The operator technique

We would like to obtain a general expression which will allow to calculate the finite difference derivatives of arbitrary order up to arbitrary order of h in the truncation error. We achieve this goal by introducing the shift operator T and its inverse operator T^{-1} as[5]

$$T f_i = f_{i+1} , \tag{7.18}$$

[4]Please note that the TAYLOR expansion of $(D_c f_{i-1} + D_c f_{i+1})/2 = (f_{i+2} - f_{i-2})/(4h)$ is equivalent to the expansion (7.14) of $D_c f_i$ with h replaced by $2h$.

[5]We note in passing that the shift operators form the discrete translational group, a very important group in theoretical physics. Let $T(n) = T^n$ denote the shift by $n \in \mathbb{N}$ grid-points. We then have

$$T(n)T(m) = T(n+m) , \tag{7.17a}$$

$$T(0) = \mathbb{1} , \tag{7.17b}$$

and

$$T^{-1} f_i = f_{i-1} \,, \tag{7.19}$$

where $TT^{-1} = \mathbb{1}$ is the unity operator. We can write these operators in terms of the forward and backward difference operators Δ_+ and Δ_- of Eqs. (7.8), in particular

$$T = \mathbb{1} + \Delta_+ \,, \tag{7.20}$$

and

$$T^{-1} = \mathbb{1} - \Delta_- \,. \tag{7.21}$$

Moreover, if $D \equiv \mathrm{d}/\mathrm{d}x$ denotes the derivative operator and if the n-th power of this operator D is understood as the n-th successive application of it, we can rewrite the Taylor expansions (7.9) as

$$\begin{aligned} f_{i+1} &= \left[\mathbb{1} + hD + \frac{1}{2}h^2D^2 + \frac{1}{3!}h^3D^3 + \dots\right] f_i \\ &\equiv \exp(hD)\, f_i \,, \end{aligned} \tag{7.22}$$

and

$$\begin{aligned} f_{i-1} &= \left[\mathbb{1} - hD + \frac{1}{2}h^2D^2 - \frac{1}{3!}h^3D^3 \pm \dots\right] f_i \\ &\equiv \exp(-hD)\, f_i \,, \end{aligned} \tag{7.23}$$

Hence, we find that [5][6]

$$T = \mathbb{1} + \Delta_+ \equiv \exp(hD) \,, \tag{7.24}$$

and, accordingly, that

$$T^{-1} = \mathbb{1} - \Delta_- \equiv \exp(-hD) \,. \tag{7.25}$$

Finally, we obtain the central difference operator:

$$\Delta_c = T - T^{-1} = \exp(hD) - \exp(-hD) \equiv 2\sinh(hD) \,. \tag{7.26}$$

Equations (7.24), (7.25) and (7.26) can be inverted for hD:

$$hD = \begin{cases} \ln(\mathbb{1} + \Delta_+) = \Delta_+ - \frac{1}{2}\Delta_+^2 + \frac{1}{3}\Delta_+^3 \mp \dots \,, \\ -\ln(\mathbb{1} - \Delta_-) = \Delta_- + \frac{1}{2}\Delta_-^2 + \frac{1}{3}\Delta_-^3 + \dots \,, \\ \sinh^{-1}\left(\frac{\Delta_c}{2}\right) = \frac{\Delta_c}{2} - \frac{1}{3!}\left(\frac{\Delta_c}{2}\right)^3 + \frac{3^2}{5!}\left(\frac{\Delta_c}{2}\right)^5 \mp \dots \,. \end{cases} \tag{7.27}$$

Again, the n-th power of an operator K (with $K = \Delta_+, \Delta_-, \Delta_c$) $K^n f_i$ is understood as the n-th successive action of the operator K on f_i, i.e. $K^{n-1}(K f_i)$. Expression

and

$$T(n)^{-1} = T(-n) \,, \tag{7.17c}$$

which are the properties required to form a group. Here $\mathbb{1}$ denotes the unity element. Moreover, we have

$$T(n)T(m) = T(m)T(n) \,, \tag{7.17d}$$

i.e. it is an Abelian group. The group of discrete translations is usually denoted by $\mathbb{T}^d$ [8].

[6]This representation of the shift operator T explains why the derivative operator D is frequently referred to as the *infinitesimal generator of translations* [8].

(7.27) allows to approximate the derivatives up to arbitrary order using finite differences. Furthermore, we can take the k-th power of Eq. (7.27) in order to get an approximate k-th derivative, $(hD)^k$ [5].

However, it turns out that the expansion (7.27) in terms of the central difference Δ_c does not optimally use the grid because it contains only odd powers of Δ_c. For instance, the third power $\Delta_c^3 f_i$ includes the function values $f_{i\pm3}$ and $f_{i\pm1}$ at 'odd' grid-points but ignores the function values f_i and $f_{i\pm2}$ at 'even' grid-points. Since this is true for all odd powers of Δ_c we observe that the expansion (2.27) uses only half of the grid. On the other hand, if one computes the square $(hD)^2$ of (2.27) only 'even' grid-points are used, while the 'odd' grid-points are ignored. This reduces the accuracy of the method and an improvement is required. The easiest remedy is to formally introduce function values $T^{\pm\frac{1}{2}} f_i \overset{!}{=} f_{i\pm1/2}$ at *intermediate* grid-points[7] $x_{i\pm1/2} = x_i \pm h/2$. This definition allows to introduce the central difference operator δ_c of intermediate grid-points,

$$\delta_c = T^{\frac{1}{2}} - T^{-\frac{1}{2}} = 2\sinh\left(\frac{hD}{2}\right), \tag{7.28}$$

and the average operator:

$$\mu = \frac{1}{2}\left(T^{\frac{1}{2}} + T^{-\frac{1}{2}}\right) = \cosh\left(\frac{hD}{2}\right). \tag{7.29}$$

The central difference operator Δ_c on the grid is connected to the central difference operator δ_c of intermediate grid-points by:

$$\Delta_c = 2\mu\delta_c. \tag{7.30}$$

To avoid the problem of Eq. (7.27) that only odd or even grid-points are accounted for we replace all shift operators $\Delta_c/2$ by δ_c and then multiply the right hand side of Eq. (2.27) by μ. This ensures that function values at intermediate grid-points will not appear in the final expression. Hence, we obtain for the first order derivative operator:

$$D = \frac{1}{h}\begin{cases} \Delta_+ - \frac{1}{2}\Delta_+^2 + \frac{1}{3}\Delta_+^3 \mp \ldots, \\ \Delta_- + \frac{1}{2}\Delta_-^2 + \frac{1}{3}\Delta_-^3 + \ldots, \\ \mu\delta_c - \frac{1}{3!}\mu\delta_c^3 + \frac{3^2}{5!}\mu\delta_c^5 \mp \ldots \end{cases} \tag{7.31}$$

When higher order derivatives are calculated, we replace, again, $\Delta_c/2$ by δ_c and multiply odd powers of δ_c by μ. This procedure results, for instance, in the second order derivative operator:

$$D^2 = \frac{1}{h^2}\begin{cases} \Delta_+^2 - \Delta_+^3 + \frac{11}{12}\Delta_+^4 \mp \ldots, \\ \Delta_-^2 + \Delta_-^3 + \frac{11}{12}\Delta_-^4 + \ldots, \\ \delta_c^2 - \frac{1}{3}\delta_c^4 + \frac{8}{45}\delta_c^6 \mp \ldots. \end{cases} \tag{7.32}$$

[7] These intermediate grid-points are virtual, auxilliary grid-ponts which will be eliminated in due course.

In particular, we obtain for the central difference derivative

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + \mathcal{O}(h^2)\ , \tag{7.33}$$

and

$$f_i'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + \mathcal{O}(h^2)\ . \tag{7.34}$$

Here, $\mathcal{O}(h^2)$ indicates that this term is of the order of h^2 and we get the important result that the truncation error is of the order $\mathcal{O}(h^2)$.[8]

7.5 Error analysis

We discussed already that the basic principle of a numerical differentiation is that of replacing the required differential with a suitable finite difference expression:

$$\left.\frac{\mathrm{d}}{\mathrm{d}x} f(x)\right|_{x=x_0} \approx \frac{f(x_0 + h) - f(x_0)}{h}. \tag{7.35}$$

The differential ratio (7.35) converges to the first derivative of the function $f(x)$ at x_0 when the increment $h \to 0$. A finite increment (or stepsize) h introduces a *methodological* error ε_V which gets smaller and smaller with decreasing h.

Theory shows that ε_V must have the form

$$\varepsilon_V = hC_V(h), \tag{7.36}$$

where the quantity $C_V(h)$ depends in many cases only weakly on h and can in practice be replaced by a constant:

$$C_V(h) \approx C_V.$$

If we plot ε_V vs h in a *log-log scale*, the error curve satisfies the linear relation

$$\log \varepsilon_V = \log C_V + \log h \tag{7.37}$$

with slope +1. The curve (a) in Fig. 7.3 shows this relation for the concrete example

$$f(x) = \ln\ x \sin(5x),$$

with

$$\left.\frac{\mathrm{d}}{\mathrm{d}x} f(x)\right|_{x_0=3} = \frac{\sin(15)}{3} + 5\ln\ 3\ \cos(15).$$

Clearly the relation (7.36) is well fulfilled within the interval $10^{-7} < h < 10^{-1}$. For increments that are larger or smaller than these values, the 'real' behaviour of the error deviates substantially because of the following reasons:

- The deviation for $h > 10^{-1}$ is due to the fact that C_V is no longer a constant if the increment h becomes too large.

[8]The leading order of the truncation error can be determined by inserting the dominant contribution of Eqs. (7.28) and (7.29) into the remainder of Eqs. (7.31) and (7.32), respectively. For instance, it follows from Eq. (7.29) that $\mu \sim \mathcal{O}(1)$ and from Eq. (7.28) that $\delta_c \sim \mathcal{O}(h)$ and, hence, we find with the help of Eq. (7.31) that $\mu\delta_c^3/h \sim \mathcal{O}(h^2)$. In analogue, we obtain from Eq. (7.32) that $\delta_c^4/h^2 \sim \mathcal{O}(h^2)$.

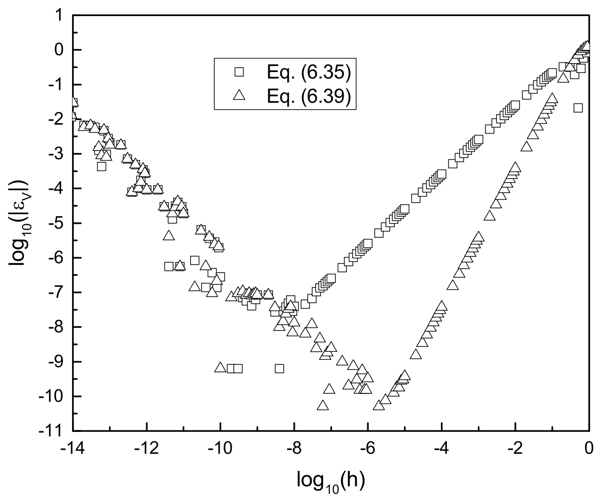

Figure 7.3: Error diagram for numerical differentiation: (a) Eq. (7.35) open squares, (b) Formula (7.39) open up-triangles.

- It is more interesting to analyze the behaviour of the error for $h < 10^{-7}$ where the error does no longer follow the predicted linear behaviour as a function of h, see Eq. (7.36). Instead, the error acquires a strong functional dependence and this is due to a dominant "subtractive cancellation" of the roundoff errors in Eq. (7.35).

The dependence of the roundoff error on h does not follow a smooth curve, but tends to the expression:

$$\varepsilon_R = \frac{C_R}{h}. \tag{7.38}$$

From what we learnt so far, we can conclude:

- Using an optimal increment step h_{opt} we can reduce the error to a minimal value which cannot be further reduced - in our concrete examples this error is around 5×10^{-7}.

- Using an increment step $h < h_{\text{opt}}$ does not improve the numerical result. On the contrary, it makes it worse!

There are two possible strategies to further reduce this minimal error:

- The roundoff error can be reduced by increasing the precision with which real numbers are stored in memory. Unfortunately, in standard computers this is in most cases not possible.

- The methodological error can be reduced using an *improved algorithm*, for example using the more efficient formula:

$$\left.\frac{\mathrm{d}}{\mathrm{d}x} f(x)\right|_{x=x_0} \approx \frac{f(x_0+h) - f(x_0-h)}{2h} \tag{7.39}$$

for the differential ratio. The error ε_V decreases much faster with decreasing values of h when this formula is applied. We get

$$\varepsilon_V = \bar{C}_V(h)\, h^2. \tag{7.40}$$

Nevertheless, the roundoff error has the same dependence on h as before and the minimal error can be reduced by roughly three orders of magntidude - see curve (b) in Fig. 7.3.

We conclude: The error diagnostics in this example (and in many other areas of numerical mathematics) is largely simplified by the fact that under certain assumptions (in this case, within a particular range of h) the methodological error is much larger than the roundoff error.

7.6 Concluding discussion

First of all, although Eq. (7.27) allows to approximate a derivative of any order k arbitrarily close, it is still an infinite series which leaves us with the decision at which order to truncate. This choice will highly depend on the choice of h which in turn depends on the function we would like to differentiate. Consider, for instance, the periodic function

$$f(x) = \exp(i\omega x) \ , \tag{7.41}$$

where $\omega, x \in \mathbb{R}$ and i is the imaginary unit with $i^2 = -1$. Its first derivative is

$$f'(x) = i\omega \exp(i\omega x) \ . \tag{7.42}$$

We now introduce grid-points by

$$x_k = x_0 + kh \ , \tag{7.43}$$

where h is the grid-spacing and x_0 is some finite starting point on the real axis. Accordingly,

$$f_k = \exp\left[i\omega(x_0 + kh)\right] \ , \tag{7.44}$$

and the exact value of the first derivative is

$$f'_k = i\omega \exp\left[i\omega(x_0 + kh)\right] = i\omega f_k \ . \tag{7.45}$$

We calculate the forward, backward, and central difference derivatives according to Eqs. (7.10) and obtain

$$D_+ f_k = i\omega f_k \exp\left(\frac{i\omega}{2}\right) \operatorname{sinc}\left(\frac{h\omega}{2}\right) \ , \tag{7.46a}$$

with $\operatorname{sinc}(x) = \sin(x)/x$ and

$$D_- f_k = i\omega f_k \exp\left(-\frac{i\omega}{2}\right) \operatorname{sinc}\left(\frac{h\omega}{2}\right) \ , \tag{7.46b}$$

and

$$D_c f_k = i\omega f_k \operatorname{sinc}(h\omega) \ . \tag{7.46c}$$

We divide the approximate derivatives by the true value (7.45) and take the modulus. We get

$$\left|\frac{D_+ f_k}{f'_k}\right| = \left|\frac{D_- f_k}{f'_k}\right| = \operatorname{sinc}\left(\frac{h\omega}{2}\right) , \tag{7.47}$$

and

$$\left|\frac{D_c f_k}{f'_k}\right| = \operatorname{sinc}(h\omega) . \tag{7.48}$$

Since $|\sin(x)| \leq |x|$, $\forall x \in \mathbb{R}$ we obtain that in all three cases this ratio is less than one independent of h, unless $\omega = 0$. (Please keep in mind that $\operatorname{sinc}(x) \to 1$ as $x \to 0$.) Hence, the first order finite difference approximations underestimate the true value of the derivative. The reason is easily found: $f(x)$ oscillates with frequency ω while the finite difference derivatives applied here approximate the derivative linearly. Higher order corrections will, of course, improve the approximation significantly. Furthermore, we observe that the one-sided finite difference derivatives (7.46a) and (7.46b) are exactly zero if $h\omega = 2n\pi$, $n \in \mathbb{N}$, i.e. if the grid-spacing h matches a multiple of the frequency $2\pi\omega$ of the function $f(x)$. The same occurs when central derivatives (7.46c) are used, but now for $h\omega = \pi n$. This is not really a problem in our example because we choose the grid-spacing $h \ll 2\pi/\omega$ in order to approximate the function $f(x)$ sufficiently well. However, in many cases the analytic form of the function is unknown and we only have its representation on the grid. In this case one has to check carefully by changing h whether the function is periodic or not.

We discuss, finally, how to approximate partial derivatives of functions which depend on more than one variable. Basically this can be achieved by independently discretisizing the function of interest in each particular variable and then by defining the corresponding finite difference derivatives. We will briefly discuss the case of two variables and the extension to even more variables is straight forward. We regard a function $g(x, y)$ where $(x, y) \in [a, b]\times[c, d]$. We denote the grid-spacing in x-direction by h_x and in y-direction by h_y. The evaluation of derivatives of the form $\frac{\partial^n}{\partial x^n} g(x, y)$ or $\frac{\partial^n}{\partial y^n} g(x, y)$ for arbitrary n are approximated with the help of the schemes discussed above, only the respective grid-spacing has to be accounted for. We will now briefly discuss mixed partial derivatives, in particular the derivative $\frac{\partial^2}{\partial x \partial y} g(x, y)$. Higher orders can be easily obtained in the same fashion. Here, we will restrict to the case of the central difference derivative. Again, the extension to the other two forms of derivatives is straight forward. We would like to approximate the derivative at the point $(a + ih_x, c + jh_y)$, which will be abbreviated by (i, j). Hence, we compute

$$\begin{aligned}
\frac{\partial}{\partial y}\frac{\partial}{\partial x} g(x,y)\Big|_{(i,j)} &= \frac{1}{2h_x}\left[\frac{\partial}{\partial y} g(x,y)\Big|_{(i+1,j)} - \frac{\partial}{\partial y} g(x,y)\Big|_{(i-1,j)}\right] + \mathcal{O}(h_x^2) \\
&= \frac{1}{2h_x}\left[\frac{g_{i+1,j+1} - g_{i+1,j-1}}{2h_y} + \mathcal{O}(h_y^2)\right. \\
&\quad \left. - \frac{g_{i-1,j+1} - g_{i-1,j-1}}{2h_y} - \mathcal{O}(h_y^2)\right] + \mathcal{O}(h_x^2) ,
\end{aligned} \tag{7.49}$$

where we made use of the notation $g_{i,j} \equiv g(x_i, y_j)$. Neglecting higher order contributions yields

$$\frac{\partial}{\partial y}\frac{\partial}{\partial x} g(x,y)\Big|_{(i,j)} \approx \frac{1}{2h_x}\frac{g_{i+1,j+1} - g_{i+1,j-1} - g_{i-1,j+1} + g_{i-1,j-1}}{2h_y} . \tag{7.50}$$

This simple approximation is easily improved with the help of methods developed in the previous sections.

It should be noted that there are also other methods to approximate derivatives. One of the most powerful methods, is the method of *finite elements* [3]. The conceptual difference to the method of finite differences is that one divides the domain in finite sub-domains (elements) rather than by replacing these by sets of discrete grid-points. The function of interest, say $g(x, y)$, is then replaced within each element by an interpolating polynomial. However, this method is quite complex and definitely beyond the scope of this book. Another interesting method, which is particularly useful for the solution of hyperbolic differential equations, is the method of *finite volumes*. The interested reader is referred to the book by R. J. LeVeque [6].

Bibliography

[1] Davies, E.B.: Linear Operators and their Spectra. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, UK (2007)

[2] Gautschi, W.: Numerical Analysis. Springer, Berlin, Heidelberg (2012)

[3] Gockenbach, M.S.: Understanding and Implementing the Finite Element Method. Cambridge University Press, Cambridge, UK (2006)

[4] Jordan, C.: Calculus of Finite Differences, 3rd edn. AMS Chelsea Publishing, Providence (1965)

[5] Lapidus, L., Pinder, G.F.: Numerical Solution of Partial Differential Equations. Wiley, New York (1982)

[6] LeVeque, R.J.: Finite Volume Methods for Hyperbolic Problems. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, UK (2002)

[7] Süli, E., Mayers, D.: An Introduction to Numerical Analysis. Cambridge University Press, Cambridge, UK (2003)

[8] Tung, W.K.: Group Theory in Physics. World Scientific, New Jersey (2003)

[9] Weidmann, J.: Lineare Operatoren in Hilberträumen, vol. I: Grundlagen. Springer, Berlin, Heidelberg (2000)

Chapter 8

Numerical integration

8.1 Introduction

Numerical integration is certainly one of the most important concepts in computational analysis since it plays a major role in the numerical treatment of differential equations. Given a function $f(x)$ which is continuous on the interval $[a, b]$, one wishes to approximate the integral by a discrete sum of the form

$$\int_a^b \mathrm{d}x f(x) \approx \sum_{i=1}^{N} \omega_i f(x_i), \tag{8.1}$$

where the ω_i are referred to as weights and x_i are the grid-points at which the function needs to be evaluated. Such methods are commonly referred to as *quadrature* [5, 11].

We will mainly discuss two different approaches to the numerical integration of arbitrary functions. We start with a rather simple approach, the *rectangular* rule. The search of an improvement of this method will lead us first to the *trapezoidal* rule, then to the SIMPSON rule and, finally, to a general formulation of the method, the NEWTON - COTES quadrature. This will be followed by a more advanced technique, the GAUSS - LEGENDRE quadrature. At the end of the chapter we will discuss an elucidating example and briefly sketch extensions of all methods to more general problems, such as integration of non-differentiable functions or the evaluation of multiple integrals.

8.2 Rectangular rule

The straight forward approach to numerical integration is to employ the concept of finite differences developed in Sec. 7.2. We regard a smooth function $f(x)$ within the interval $[a, b]$, i.e. $f(x) \in \mathcal{C}^\infty[a, b]$. The RIEMANN definition of the proper integral of $f(x)$ from a to b states that:

$$\int_a^b \mathrm{d}x\, f(x) = \lim_{N\to\infty} \frac{b-a}{N} \sum_{i=0}^{N} f\left(a + i\frac{b-a}{N}\right). \tag{8.2}$$

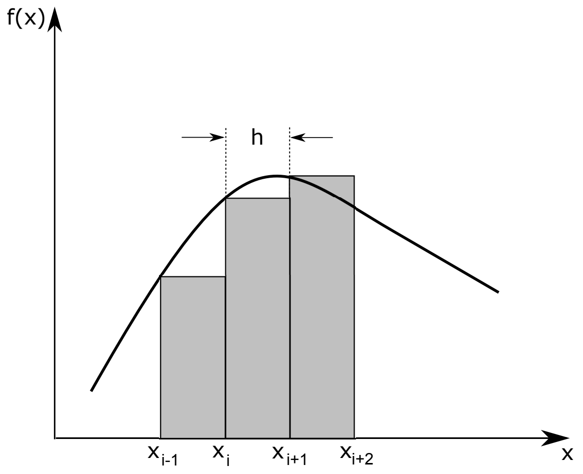

Figure 8.1: Illustration of the numerical approximation of a proper integral according to Eq. (8.3).

We approximate the right hand side of this relation using equally spaced grid-points $x_i \in [a, b]$ according to Eq. (7.1) and find

$$\int_a^b \mathrm{d}x f(x) \approx h \sum_{i=1}^{N-1} f_i. \tag{8.3}$$

It is clear that the quality of this approach strongly depends on the discretization chosen, i.e. on the values of x_i as illustrated schematically in Fig. 8.1. Again, a non-uniform grid may be of advantage. We can estimate the error of this approximation by expanding $f(x)$ into a TAYLOR series.

We note that

$$\int_a^b \mathrm{d}x f(x) = \sum_{i=1}^{N-1} \int_{x_i}^{x_{i+1}} \mathrm{d}x f(x), \tag{8.4}$$

hence, the approximation (8.3) is equivalent to an estimate of the area in the unit interval, *the elemental area*:

$$\int_{x_i}^{x_{i+1}} \mathrm{d}x\, f(x) \approx h f_i. \tag{8.5}$$

Furthermore, we find following Eq. (7.9a):

$$\begin{aligned}\int_{x_i}^{x_{i+1}} \mathrm{d}x f(x) &= \int_{x_i}^{x_{i+1}} \mathrm{d}x \left[f_i + (x - x_i) f'_{i+\varepsilon_\zeta}\right] \\ &= f_i h + \mathcal{O}(h^2). \end{aligned} \tag{8.6}$$

In this last step we applied the first mean value theorem for integration which states that if $f(x)$ is continuous in $[a, b]$, then there exists a $\zeta \in [a, b]$ such that

$$\int_a^b \mathrm{d}x f(x) = (b - a) f(\zeta). \tag{8.7}$$

Consequently, the error we make with approximation (8.3) can be seen from Eq. (8.6) to be of the order $\mathcal{O}(h^2)$.

This procedure corresponds to a forward difference approach and, equivalently, backward differences can be used. This results in:

$$\int_a^b \mathrm{d}x f(x) = h \sum_{i=2}^{N} f_i + \mathcal{O}(h^2). \tag{8.8}$$

Let us now define the forward and backward rectangular rule by

$$_iI_{i+1}^{+} = hf_i \, , \tag{8.9}$$

and

$$_iI_{i+1}^{-} = hf_{i+1}, \tag{8.10}$$

respectively. Thus, we obtain from TAYLOR's expansion that:

$$\begin{aligned} \int_{x_i}^{x_{i+1}} \mathrm{d}x f(x) &= {}_iI_{i+1}^{+} + \frac{h^2}{2} f_i' + \frac{h^3}{3!} f_i'' + \ldots \\ &= {}_iI_{i+1}^{-} - \frac{h^2}{2} f_{i+1}' + \frac{h^3}{3!} f_{i+1}'' \mp \ldots . \end{aligned} \tag{8.11}$$

However, the use of central differences gives more accurate results as has already been observed in Chap. 7 in which the differential operator was approximated. We make use of the concept of intermediate grid-points (see Sec. 7.4) and consider the integral

$$\int_{x_i}^{x_{i+1}} \mathrm{d}x f(x), \tag{8.12}$$

expand $f(x)$ in a TAYLOR series around the midpoint $x_{i+\frac{1}{2}}$, and obtain:

$$\begin{aligned} \int_{x_i}^{x_{i+1}} \mathrm{d}x f(x) &= \int_{x_i}^{x_{i+1}} \mathrm{d}x \left\{ f_{i+\frac{1}{2}} + \left(x - x_{i+\frac{1}{2}}\right) f_{i+\frac{1}{2}}' \right. \\ &\quad \left. + \frac{\left(x - x_{i+\frac{1}{2}}\right)^2}{2} f_{i+\frac{1}{2}}'' + \mathcal{O}\left[\left(x - x_{i+\frac{1}{2}}\right)^3\right] \right\} \\ &= hf_{i+\frac{1}{2}} + \frac{h^3}{24} f_{i+\varepsilon_\zeta}'' \\ &= {}_iI_{i+1} + \frac{h^3}{24} f_{i+\varepsilon_\zeta}'' . \end{aligned} \tag{8.13}$$

Thus, the error generated by this method, the central rectangular rule, scales as $\mathcal{O}(h^3)$ which is a significant improvement in comparison to Eqs. (8.3) and (8.8).[1] We obtain

$$\int_a^b \mathrm{d}x f(x) = h \sum_{i=1}^{N-1} f_{i+\frac{1}{2}} + \mathcal{O}(h^3). \tag{8.14}$$

This approximation is known as the *rectangular rule*. It is illustrated in Fig. 8.2. Note that the boundary points $x_1 = a$ and $x_N = b$ do not enter Eq. (8.14). Such a procedure is commonly referred to as an *open integration rule*. On the other hand, if the end-points are taken into account by the method it is referred to as a *closed integration rule*.

[1]In this context the intermediate position $x_{i+1/2}$ is understood as a true grid-point. If, on the other hand, the function value $f_{i+1/2}$ is approximated by $\mu f_{i+1/2}$, Eq. (7.29), the method is referred to as the *trapezoidal rule*

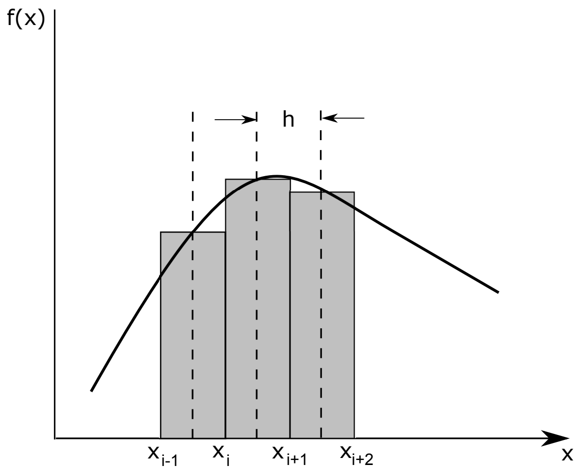

Figure 8.2: Scheme of the rectangular integration rule according to Eq. (8.14). Note that boundary points do not enter the evaluation of the elemental areas.

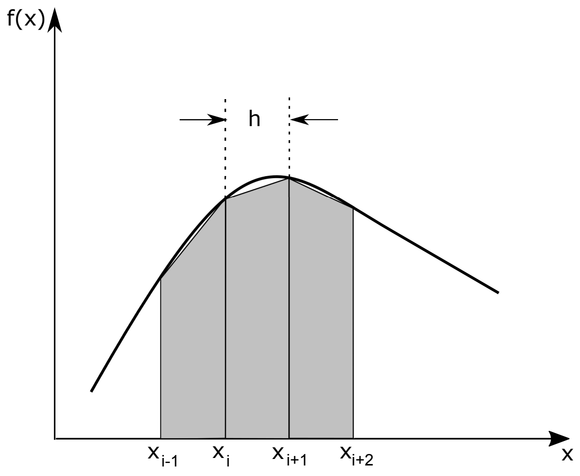

Figure 8.3: Sketch of how the elemental areas under the curve $f(x)$ are approximated by trapezoids.

8.3 Trapezoidal rule

An elegant alternative to the rectangular rule is found when the area between two grid-points is approximated by a trapezoid as is shown schematically in Fig. 8.3. The trapezoidal rule is obtained when the function values $f_{i+1/2}$ at intermediate grid-points on the right hand side of the central rectangular rule (8.13) are approximated with the help of $\mu f_{i+1/2}$, Eq. (7.29). Thus, the elemental area is calculated from

$$\int_{x_i}^{x_{i+1}} \mathrm{d}x f(x) \approx \frac{h}{2}\left(f_i + f_{i+1}\right). \tag{8.15}$$

and we obtain:

$$\begin{aligned}\int_a^b \mathrm{d}x f(x) &\approx \frac{h}{2}\sum_{i=1}^{N-1}(f_i+f_{i+1})\\ &= h\left(\frac{f_1}{2}+f_2+\ldots+f_{N-1}+\frac{f_N}{2}\right)\\ &= \frac{h}{2}(f_1+f_N)+h\sum_{i=2}^{N-1}f_i\\ &= {}_1I_N^T. \end{aligned}\tag{8.16}$$

Note that this integration rule is closed, although the boundary points f_1 and f_N enter the summation (8.16) only with half the weight in comparison to all other function values f_i. This stems from the fact that the function values f_1 and f_N contribute only to one elemental area, the first and the last one. Another noticeable feature of the trapezoidal rule is that, in contrast to the rectangular rule (8.14), only function values at grid-points enter the summation, which can be desirable in some cases.

The error of this method can be estimated by inserting expansion (7.9a) into Eq. (8.16). One obtains for an elemental area:

$$\begin{aligned}{}_iI_{i+1}^T &= \frac{h}{2}(f_i+f_{i+1})\\ &= hf_i+\frac{h^2}{2}f_i'+\frac{h^3}{4}f_i''+\ldots. \end{aligned}\tag{8.17}$$

On the other hand, we know from Eq. (8.6) that

$$hf_i=\int_{x_i}^{x_{i+1}}\mathrm{d}x f(x)-\frac{h^2}{2}f_i'-\frac{h^3}{3!}f_i''-\ldots,\tag{8.18}$$

which, when inserted into (8.17), yields

$${}_iI_{i+1}^T=\int_{x_i}^{x_{i+1}}\mathrm{d}x f(x)+\frac{h^3}{12}f_i''+\mathcal{O}(h^4).\tag{8.19}$$

Hence, we observe that the error induced by the trapezoidal rule is comparable to the error of the rectangular rule, namely $\mathcal{O}(h^3)$. However, since we do not have to compute function values at intermediate grid-points, this rule may be advantageous in many cases.

We remember from Chap. 7 that a more accurate estimate of a derivative was achieved by increasing the number of grid-points involved which in the case of integration leads us to the SIMPSON rule.

8.4 The SIMPSON rule

The basic idea of the SIMPSON rule is to include higher order derivatives into the expansion of the integrand. These higher order derivatives, which are primarily unknown, are then approximated by expressions we obtained within the context of finite difference derivatives. Let us discuss this procedure in greater detail. To this purpose we will study the integral of $f(x)$ within the interval $[x_{i-1}, x_{i+1}]$ and expand the integrand around the midpoint x_i:

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+1}} \mathrm{d}x f(x) &= \int_{x_{i-1}}^{x_{i+1}} \mathrm{d}x \left[f_i + (x - x_i) f_i' + \frac{(x-x_i)^2}{2!} f_i'' \right. \\ &\quad \left. + \frac{(x-x_i)^3}{3!} f_i''' + \ldots \right] \\ &= 2hf_i + \frac{h^3}{3} f_i'' + \mathcal{O}(h^5). \end{aligned} \tag{8.20}$$

Inserting Eq. (7.34) for f_i'' yields

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+1}} \mathrm{d}x f(x) &= 2hf_i + \frac{h}{3}(f_{i+1} - 2f_i + f_{i-1}) + \mathcal{O}(h^5) \\ &= h\left(\frac{1}{3}f_{i-1} + \frac{4}{3}f_i + \frac{1}{3}f_{i+1}\right) + \mathcal{O}(h^5). \end{aligned} \tag{8.21}$$

Note that in contrast to the trapezoidal rule, the procedure described here is a three point method since the function values at three different points enter the expression. We can immediately write down the resulting integral from a to b. Since,

$$\int_a^b \mathrm{d}x f(x) = \int_{x_0}^{x_2} \mathrm{d}x f(x) + \int_{x_2}^{x_4} \mathrm{d}x f(x) + \ldots + \int_{x_{N-2}}^{x_N} \mathrm{d}x f(x), \tag{8.22}$$

where we assumed that N is even and employed the discretization $x_i = x_0 + ih$ with $x_0 = a$ and $x_N = b$. We obtain:

$$\int_a^b \mathrm{d}x f(x) = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 2f_{N-2} + 4f_{N-1} + f_N) + \mathcal{O}(h^5). \tag{8.23}$$

This expression is exact for polynomials of degree $n \leq 3$ since the first term in the error expansion involves the fourth derivative. Hence, whenever the integrand is satisfactorily reproduceable by a polynomial of degree three or less, the SIMPSON rule might give almost exact estimates, independent of the discretization h.

The arguments applied above allow for a straightforward extension to four- or even more-point rules. We find, for instance,

$$\int_{x_i}^{x_{i+3}} \mathrm{d}x f(x) = \frac{3h}{8}(f_i + 3f_{i+1} + 3f_{i+2} + f_{i+3}) + \mathcal{O}(h^5), \tag{8.24}$$

which is usually called SIMPSON's three-eight rule.

It is important to note that all the methods discussed so far are special cases of a more general formulation, the NEWTON - COTES rules [11] which will be discussed in the next section.

8.5 General formulation - The Newton - Cotes rules

We define the Lagrange interpolating polynomial $p_{n-1}(x)$ of degree $n-1$ [6, 2, 4] to a function $f(x)$ as[2]

$$p_{n-1}(x) = \sum_{j=1}^{n} f_j L_j^{(n-1)}(x), \tag{8.25}$$

where

$$L_j^{(n-1)}(x) = \prod_{\substack{k=1\\k\neq j}}^{n} \frac{x-x_k}{x_j-x_k}. \tag{8.26}$$

An arbitrary smooth function $f(x)$ can then be expressed with the help of a Lagrange polynomial of degree n by

$$f(x) = p_{n-1}(x) + \frac{f^{(n)}[\zeta(x)]}{n!}(x-x_1)(x-x_2)\dots(x-x_n). \tag{8.27}$$

If we neglect the second term on the right hand side of this equation and integrate the Lagrange polynomial of degree $n-1$ over the n grid-points from x_1 to x_n we obtain the closed n-point Newton - Cotes formulas. For instance, if we set $n=2$, then

$$\begin{aligned}
p_1(x) &= f_1 L_1^{(1)}(x) + f_2 L_2^{(1)}(x)\\
&= f_1\frac{x-x_2}{x_1-x_2} + f_2\frac{x-x_1}{x_2-x_1}\\
&= \frac{1}{h}\left[x(f_2-f_1) - x_1 f_2 + x_2 f_1\right],
\end{aligned} \tag{8.28}$$

with $f_1 \equiv f(x_1)$ and $f_2 \equiv f(x_2)$. Integration over the respective interval yields

$$\begin{aligned}
\int_{x_1}^{x_2} \mathrm{d}x\, p_1(x) &= \frac{1}{h}\left[\frac{x^2}{2}(f_2-f_1) + x(x_2 f_1 - x_1 f_2)\right]\Bigg|_{x_1}^{x_2}\\
&= \frac{h}{2}\left[f_2+f_1\right],
\end{aligned} \tag{8.29}$$

which is exactly the trapezoidal rule. By setting $n=3$ one obtains Simpson's rule and setting $n=4$ gives the Simpson's three-eight rule.

The *open* Newton - Cotes rule can be obtained by integrating the polynomial $p_{n-1}(x)$ of degree $n-1$ which includes the grid-points $x_1,\dots,x_n$ from x_0 to x_{n+1}. The fact that these relations are open means that the function values at the boundary points $x_0 = x_1 - h$ and $x_{n+1} = x_n + h$ do not enter the final expressions. The simplest open Newton - Cotes formula is the central integral approximation, which we encountered as the rectangular rule (8.14). A second order approximation is easily found with help of the two-point Lagrange polynomial (8.28)

$$\begin{aligned}
\int_{x_0}^{x_3} \mathrm{d}x p_1(x) &= \frac{1}{h}\left[\frac{x^2}{2}(f_2-f_1) + x(x_2 f_1 - x_1 f_2)\right]\Bigg|_{x_0}^{x_3}\\
&= \frac{3h}{2}\left[f_2+f_1\right].
\end{aligned} \tag{8.30}$$

[2]The Lagrange polynomial $p_{n-1}(x)$ to the function $f(x)$ is the polynomial of degree $n-1$ that satisfies the n equations $p_{n-1}(x_j) = f(x_j)$ for $j = 1,\dots,n$, where x_j denotes arbitrary but distinct grid-points.

Higher order approximations can be obtained in a similar fashion. To conclude this section let us briefly discuss an idea which is referred to as ROMBERG's method [10].

So far, we approximated all integrals by expressions of the form

$$I = \mathcal{I}^N + \mathcal{O}(h^m), \tag{8.31}$$

where I is the exact, unknown, value of the integral, $\mathcal{I}^N$ is the estimate obtained from an integration scheme using N grid-points and m is the leading order of the error. Let us review the error of the trapezoidal approximation: we learned that the error for the integral over the interval $[x_i, x_{i+1}]$ scales like h^3. Since we have N such intervals, we conclude that the total error behaves like $(b-a)h^2$. Similarly, the error of the three-point SIMPSON rule is for each sub-interval proportional to h^5 and this gives in total $(b-a)h^4$. We assume that this trend can be generalized and conclude that the error of an n-point method with the estimate $\mathcal{I}_n$ behaves like h^{2n-2}. Since, $h \propto N^{-1}$ we have

$$I = \mathcal{I}_n^N + \frac{C_N}{N^{2n-2}}, \tag{8.32}$$

where C_N depends on the number of grid-points N. Let us double the amount of grid-points and we obtain:

$$I = \mathcal{I}_n^{2N} + \frac{C_{2N}}{(2N)^{2n-2}}. \tag{8.33}$$

Obviously, Eqs. (8.32) and (8.33) can be regarded as a linear system of equations in I and C if $C_N \approx C_{2N} \approx C$. Solving Eqs. (8.32) and (8.33) for I yields

$$I \approx \frac{1}{4^{n-1}-1}\left(4^{n-1}\mathcal{I}_n^{2N} - \mathcal{I}_n^N\right). \tag{8.34}$$

It has to be emphasized that in the above expression I is no longer the exact value because of the approximation $C_N \approx C$. However, it is an improvement of the solution and it is possible to demonstrate that this new estimate is exactly the value one would have obtained with an integral approximation of order $n+1$ and $2N$ grid-points! Thus

$$\mathcal{I}_{n+1}^{2N} = \frac{1}{4^{n-1}-1}\left(4^{n-1}\mathcal{I}_n^{2N} - \mathcal{I}_n^N\right). \tag{8.35}$$

This suggests a very elegant and rapid procedure: We simply calculate the integrals using two point rules and add the results according to Eq. (8.35) to obtain more-point results. For instance, calculate $\mathcal{I}_2^2$ and $\mathcal{I}_2^4$, add these according to Eq. (8.35) and get $\mathcal{I}_3^4$. Now calculate $\mathcal{I}_2^8$, add $\mathcal{I}_2^4$, get $\mathcal{I}_3^8$, add $\mathcal{I}_3^4$ and get $\mathcal{I}_4^8$. This pyramid-like procedure can be continued until convergence is achieved, that is $|\mathcal{I}_m^N - \mathcal{I}_{m+1}^N| < \varepsilon$ where $\varepsilon > 0$ can be chosen arbitrarily. An illustration of this method is given in Fig. 8.4.

8.6 GAUSS - LEGENDRE quadrature

In preparation for the GAUSS - LEGENDRE quadrature we introduce a set of orthogonal LEGENDRE polynomials $P_\ell(x)$ [6, 2, 1, 7] which are solutions of the LEGENDRE differential equation

$$\left(1-x^2\right) P_\ell''(x) - 2xP_\ell'(x) + \ell(\ell+1)P_\ell(x) = 0. \tag{8.36}$$

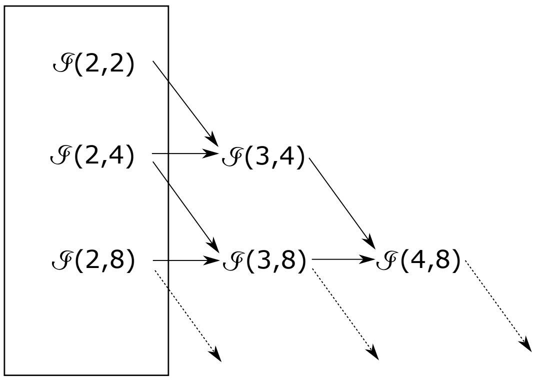

Figure 8.4: Illustration of the ROMBERG method. Here, the $\mathcal{I}(m,n)$ are synonyms for integrals $\mathcal{I}_m^n$ where the first index m refers to the order of the quadrature while the second index n refers to the number of grid-points used. Note that we only have to use a second order integration scheme (left row inside the box), all other values are determined via Eq. (8.35) as indicated by the arrows.

This equation occurs, for instance, when the LAPLACE equation $\Delta f(x) = 0$ is transformed to spherical coordinates. Here, we will introduce the most important properties of LEGENDRE polynomials which will be required for an understanding of the GAUSS - LEGENDRE quadrature.

LEGENDRE polynomials are given by

$$P_\ell(x) = \sum_{k=0}^{\infty} a_{k,\ell} x^k, \tag{8.37}$$

where the coefficients $a_{k,\ell}$ can be determined recursively:

$$a_{k+2,\ell} = \frac{k(k+1) - \ell(\ell+1)}{(k+1)(k+2)} a_{k,\ell}. \tag{8.38}$$

Hence, for even values of ℓ the LEGENDRE polynomial involves only even powers of x and for odd ℓ only odd powers of x. Note also that according to Eq. (8.38) for $k \geq \ell$ the coefficients are equal to zero and, thus, the $P_\ell(x)$ are according to Eq. (8.37) polynomials of degree ℓ. Furthermore, the LEGENDRE polynomials fulfill the orthonomality condition

$$\int_{-1}^{1} \mathrm{d}x P_\ell(x) P_{\ell'}(x) = \frac{2}{2\ell' + 1} \delta_{\ell\ell'}, \tag{8.39}$$

where δ_{ij} is KRONECKER's delta. One obtains, in particular,

$$P_0(x) = 1, \tag{8.40}$$

and

$$P_1(x) = x\ . \tag{8.41}$$

Another convenient way to calculate LEGENDRE polynomials is based on RODRIGUES' formula

$$P_\ell(x) = \frac{1}{2^\ell \ell!} \frac{\mathrm{d}^\ell}{\mathrm{d}x^\ell} \left(x^2 - 1\right)^\ell. \tag{8.42}$$

We focus now on the core of the GAUSS - LEGENDRE quadrature and introduce the function $F(x)$ as a transform of the function $f(x)$

$$F(x) = \frac{b-a}{2} f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right), \tag{8.43}$$

in such a way that we can rewrite the integral of interest as:

$$\int_a^b \mathrm{d}x f(x) = \int_{-1}^1 \mathrm{d}x F(x). \tag{8.44}$$

If the function $F(x)$ can be well approximated by some polynomial of degree $2n-1$, like

$$F(x) \approx p_{2n-1}(x) \ , \tag{8.45}$$

then this means that according to TAYLOR's theorem (7.7) the error introduced by this approximation is proportional to $F^{(2n)}(x)$. If the polynomial $p_{2n-1}(x)$ is explicitly given then we can apply the methods discussed in the previous sections to approximate the integral (8.44). However, even if the polynomial is not explicitly given we write the integral (8.44) as

$$\int_{-1}^1 \mathrm{d}x F(x) = \sum_{i=1}^n \omega_i F(x_i) \ , \tag{8.46}$$

with weights ω_i and grid-points x_i, $i = 1, \dots, n$ which are yet undetermined! Therefore, we will determine the weights ω_i and grid-points x_i in such a way, that the integral is well approximated *even* if the polynomial p_{2n-1} in Eq. (8.45) is unknown. For this purpose we decompose $p_{2n-1}(x)$ into

$$p_{2n-1}(x) = p_{n-1}(x) P_n(x) + q_{n-1}(x) \ , \tag{8.47}$$

where $P_n(x)$ is the LEGENDRE polynomial of degree n and $p_{n-1}(x)$ and $q_{n-1}(x)$ are polynomials of degree $n-1$. Since $p_{n-1}(x)$ itself is a polynomial of degree $n-1$, it can also be expanded in LEGENDRE polynomials of degrees up to $n-1$ by

$$p_{n-1}(x) = \sum_{i=0}^{n-1} a_i P_i(x) \ . \tag{8.48}$$

Using Eq. (8.48) in (8.47) we obtain together with normalization relation (8.39)

$$\int_{-1}^1 \mathrm{d}x \, p_{2n-1}(x) = \sum_{i=0}^{n-1} a_i \int_{-1}^1 \mathrm{d}x P_i(x) P_n(x) + \int_{-1}^1 \mathrm{d}x \, q_{n-1}(x) = \int_{-1}^1 \mathrm{d}x \, q_{n-1}(x) \ . \tag{8.49}$$

Moreover, since $P_n(x)$ is a LEGENDRE polynomial of degree n it has n-zeros in the interval $[-1, 1]$ and Eq. (8.47) results in

$$p_{2n-1}(x_i) = q_{n-1}(x_i) \ , \tag{8.50}$$

where $x_1, x_2, \dots, x_n$ denote the zeros of $P_n(x)$ and these zeros determine the grid-points of our integration routine. It is interesting to note, that these zeros are independent of the function $F(x)$ we want to integrate. We also expand $q_{n-1}(x)$ in terms of LEGENDRE polynomials

$$q_{n-1}(x) = \sum_{i=0}^{n-1} b_i P_i(x) , \tag{8.51}$$

and use it in Eq. (8.50) to obtain

$$p_{2n-1}(x_i) = \sum_{k=0}^{n-1} b_k P_k(x_i) , \quad i = 1, \dots, n , \tag{8.52}$$

which can be written in a more compact form by defining $p_i \equiv p_{2n-1}(x_i)$ and $P_{ki} \equiv P_k(x_i)$:

$$p_i = \sum_{k=0}^{n-1} b_k P_{ki} , \quad i = 1, \dots, n . \tag{8.53}$$

It has to be emphasized again that the grid-points x_i are independent of the polynomial $p_{2n-1}(x)$ and, therefore, independent of $F(x)$. Furthermore, we can replace $p_i \approx F(x_i) \equiv F_i$ according to Eq. (8.45). We recognize that Eq. (8.53) corresponds to a system of linear equations which can be solved for the weights b_k. We obtain

$$b_k = \sum_{i=1}^{n} F_i \left[\mathbf{P}^{-1}\right]_{ik} , \tag{8.54}$$

where $\mathbf{P}$ is the matrix $\mathbf{P} = \{P_{ij}\}$, which is known to be non-singular. We can now rewrite the integral (8.44) with the help of Eqs. (8.45), (8.49), (8.51) together with the properties of the zeros of LEGENDRE polynomials [1, 7] as

$$\int_{-1}^{1} \mathrm{d}x \, F(x) \approx \int_{-1}^{1} \mathrm{d}x p_{2n-1}(x) = \sum_{k=0}^{n-1} b_k \int_{-1}^{1} \mathrm{d}x \, P_k(x) . \tag{8.55}$$

Since $P_0(x) = 1$ according to Eq. (8.40), we deduce from Eq. (8.39)

$$\int_{-1}^{1} \mathrm{d}x P_k(x) = \int_{-1}^{1} \mathrm{d}x P_k(x) P_0(x) = \frac{2}{2k+1}\delta_{k0} = 2\delta_{k0} . \tag{8.56}$$

Hence, Eq. (8.55) reads

$$\int_{-1}^{1} \mathrm{d}x \, F(x) \approx 2b_0 = 2\sum_{i=1}^{n} F_i \left[\mathbf{P}^{-1}\right]_{i0} . \tag{8.57}$$

By defining

$$\omega_i = 2\left[\mathbf{P}^{-1}\right]_{i0} , \tag{8.58}$$

we arrive at the desired expansion

$$\int_{-1}^{1} \mathrm{d}x F(x) \approx \sum_{i=1}^{n} \omega_i F_i . \tag{8.59}$$

Moreover, since we approximated $F(x)$ by a polynomial of degree $2n-1$, the GAUSS - LEGENDRE quadrature is exact for polynomials of degree $2n-1$, i.e. the error is proportional to a derivative of $F(x)$ of order $2n$. Furthermore, expression (8.58) can be put in a more convenient form. One can show that

$$\omega_i = \frac{2}{(1-x_i^2)\,[P_n'(x_i)]^2} \; , \tag{8.60}$$

where

$$P_n'(x_i) = \frac{\mathrm{d}}{\mathrm{d}x} P_n(x)\bigg|_{x=x_i} \; . \tag{8.61}$$

Let us make some concluding remarks. The grid-points x_i as well as the weights ω_i are independent of the actual function $F(x)$ we want to integrate. This means, that one can table these values once and for all [1, 7] and use them for different types of problems. The grid-points x_i are symmetrically distributed around the point $x = 0$, i.e. for every x_j there is a $-x_j$. Furthermore, these two grid-points have the same weight ω_j. The density of grid-points increases approaching the boundary, however, the boundary points themselves are not included, which means that the GAUSS - LEGENDRE quadrature is an *open method*. Furthermore, it has to be emphasized that low order GAUSS - LEGENDRE parameters can easily be calculated by employing relation (8.42). This makes the GAUSS - LEGENDRE quadrature the predominant integration method. In comparison to the trapezoidal rule or even the ROMBERG method, it needs in many cases a smaller number of grid-points, is simpler to implement, converges faster and yields more accurate results. One drawback of this method is that one has to compute the function $F(x)$ at the zeros of the LEGENDRE polynomial x_i. This can be a problem if the integrand at hand is not known analytically.

It is important to note at this point that comparable procedures exist which use other types of orthogonal polynomials, such as HERMITE polynomials. This procedure is known as the GAUSS - HERMITE quadrature.

Tab. 8.1 lists the methods, discussed in the previous sections, which allow to calculate numerically an estimate of integrals of the form:

$$\int_a^b \mathrm{d}x f(x) \; . \tag{8.62}$$

Equal grid-spacing h is assumed, with the GAUSS - LEGENDRE method as the only exception. The particular value of h depends on the order of the method employed and is given in table 8.1.

8.7 An example

Let us discuss as an example the following proper integral:

$$I = \int_{-1}^{1} \frac{\mathrm{d}x}{x+2} = \ln(3) - \ln(1) \approx 1.09861 \; . \tag{8.63}$$

We will now apply the various methods of Tab. 8.1 to approximate Eq. (8.63). Note that these methods could give better results if a finer grid had been chosen. However, since this is only an illustrative example, we wanted to keep it as simple as possible. The rectangular rule gives

$$\mathcal{I}_R = 1 \cdot \frac{1}{2} = 0.5 \; , \tag{8.64}$$

Table 8.1: Summary of the quadrature methods discussed in this chapter applied to the integral $\int_a^b \mathrm{d}x f(x)$. For a detailed description consult the corresponding sections. Equal grid-spacing is assumed for all methods except for the GAUSS - LEGENDRE quadrature. The explicit values of h depend on the order of the method and are listed in the table. Furthermore, we use $x_i = a + ih$ and denote $f(x_i) = f_i$. The function $P^{(m)}(x)$ which appears in the description of the NEWTON - COTES rules denotes the m-th order LAGRANGE interpolating polynomial and $P_m(x)$ is the m-th degree LEGENDRE polynomial.

n	h	$\mathcal{I}$	method	comment
1	$\frac{b-a}{2}$	hf_1	rectangular	open
2	$b-a$	$\frac{h}{2}(f_0+f_1)$	trapezoidal	closed
3	$\frac{b-a}{2}$	$\frac{h}{3}(f_0+4f_1+f_2)$	SIMPSON	closed
4	$\frac{b-a}{3}$	$\frac{3h}{8}(f_0+3f_1+3f_2+f_3)$	SIMPSON $\frac{3}{8}$	closed
m	$\frac{b-a}{m-1}$	$\int_{x_0}^{x_{m-1}} \mathrm{d}x P^{(m)}(x)$	NEWTON - COTES	closed
m	$\frac{b-a}{m+1}$	$\int_{x_0}^{x_{m+1}} \mathrm{d}x P^{(m)}(x)$	NEWTON - COTES	open
m	$P_m(x_j) = 0$	$\frac{b-a}{2}\sum_{j=1}^{m}\omega_j f(z_j)$ $z_j = \frac{a+b}{2} + \frac{a-b}{2}x_j$ $\omega_j = \frac{2}{(1-x_j)^2[P'_m(x_j)]^2}$	GAUSS - LEGENDRE	open

the trapezoidal rule

$$\mathcal{I}_T = \frac{2}{2}\left(\frac{1}{1}+\frac{1}{3}\right) = \frac{4}{3} = 1.333\ldots\ , \tag{8.65}$$

and an application of the SIMPSON rule yields

$$\mathcal{I}_S = \frac{1}{3}\left(\frac{1}{1}+\frac{4}{2}+\frac{1}{3}\right) = \frac{10}{9} = 1.111\ldots\ . \tag{8.66}$$

Finally, we apply the GAUSS - LEGENDRE quadrature in a second order approximation. We could look up the parameters in [1, 7], however, for illustrative reasons we will calculate those in this simple case. For a second order approximation we need the LEGENDRE polynomial of second degree. It can be obtained from RODRIGUES' formula (8.42):

$$\begin{aligned} P_2(x) &= \frac{1}{2^2 2!}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(x^2-1\right)^2 \\ &= \frac{1}{2}\left(3x^2-1\right)\ . \end{aligned} \tag{8.67}$$

In a next step the zeros x_1 and x_2 of $P_2(x)$ are determined from Eq. (8.67) which results immediately in:

$$x_{1,2} = \pm\frac{1}{\sqrt{3}} \approx \pm 0.57735\ . \tag{8.68}$$

The weights ω_1 and ω_2 can now be evaluated according to Eq. (8.60):

$$\omega_i = \frac{2}{(1-x_i^2)\,[P_2'(x_i)]^2} \; . \tag{8.69}$$

It follows from Eq. (8.67) that

$$P_2'(x) = 3x \; , \tag{8.70}$$

and, thus,

$$P_2'(x_1) = -\sqrt{3} \qquad \text{and} \qquad P_2'(x_2) = \sqrt{3} \; . \tag{8.71}$$

This is used to calculate the weights from Eq. (8.69):

$$\omega_1 = \omega_2 = 1 \; . \tag{8.72}$$

We combine the results (8.68) and (8.72) to arrive at the GAUSS - LEGENDRE estimate of the integral (8.63):

$$\mathcal{I}_{GL} = \frac{1}{-\frac{1}{\sqrt{3}}+2} + \frac{1}{\frac{1}{\sqrt{3}}+2} = 1.090909\ldots \; . \tag{8.73}$$

Obviously, a second order GAUSS - LEGENDRE approximation results already in a much better estimate of the integral (8.63) than the trapezoidal rule which is also of second order. It is also better than the estimate by the SIMPSON rule which is of third order.

8.8 Concluding discussion

Let us briefly discuss some further aspects of numerical integration. In many cases one is confronted with improper integrals of the form

$$\int_a^\infty dx f(x), \qquad \int_{-\infty}^a dx f(x), \qquad \text{or} \qquad \int_{-\infty}^\infty dx f(x) \; . \tag{8.74}$$

The question arises whether or not we can treat such an integral with the methods discussed so far. The answer is yes, it is possible as we will demonstrate using the integral

$$I = \int_a^\infty dx f(x) \; . \tag{8.75}$$

as an example; other integrals can be treated in a similar fashion. We rewrite Eq. (8.75) as

$$I = \lim_{b\to\infty} \int_a^b dx f(x) = \lim_{b\to\infty} I(b) \; . \tag{8.76}$$

One now calculates $I(b_1)$ for some $b_1 > a$ and $I(b_2)$ for some $b_2 > b_1$. If $|I(b_2) - I(b_1)| < \varepsilon$, where $\varepsilon > 0$ is the required accuracy, the resulting value $I(b_2)$ can be regarded as the appropriate estimate to I.[3] However, in many cases it is easier to perform an integral transform in order to map the infinite interval onto a finite interval. For instance, consider [9]

$$I = \int_0^\infty dx \frac{1}{(1+x^2)^{\frac{4}{3}}} \; . \tag{8.77}$$

[3]Particular care is required when dealing with periodic functions!

The transformation

$$t = \frac{1}{1+x} \tag{8.78}$$

gives

$$I = \int_0^1 \mathrm{d}t \frac{t^{\frac{2}{3}}}{[t^2 + (1-t)^2]^{\frac{4}{3}}} \ . \tag{8.79}$$

Thus, we mapped the interval $[0,\infty) \to [0,1]$. Integral (8.79) can now be approximated with help of the methods discussed in the previous sections. These can also be applied to approximate convergent integrals whose integrand shows singular behavior within $[a,b]$.

If the integrand $f(x)$ is not smooth within the interval $I : x \in [a,b]$ we can split the total integral into a sum over sub-intervals. For instance, if we consider the function

$$f(x) = \begin{cases} x\ \cos(x), & x < 0\ , \\ x\ \sin(x), & x \geq 0\ , \end{cases}$$

we can calculate the integral over the interval $I : x \in [-10, 10]$ as

$$\int_{-10}^{10} \mathrm{d}x\, f(x) = \int_{-10}^{0} \mathrm{d}x\, x\cos(x) + \int_0^{10} \mathrm{d}x\, x\sin(x)\ .$$

We generalize this result and write

$$\int_I \mathrm{d}x\, f(x) = \sum_k \int_{I_k} \mathrm{d}x\, f(x)\ , \tag{8.80}$$

with sub-intervals $I_k \in I, \forall k$ and the integrand $f(x)$ is assumed to be smooth within each sub-interval I_k but not necessarily within the interval I. We can then apply one of the methods discussed in this chapter to calculate an estimate of the integral over any of the sub-intervals I_k.

Similar to the discussion in Sec. 7.6 about the approximation of partial derivatives on the basis of finite differences, one can apply the rules of quadrature developed here for different dimensions to obtain an estimate of multi-dimensional integrals. However, the complexity of the problem is significantly increased if the integration boundaries are functions of the variables rather than constants. For instance,

$$\int_a^b \mathrm{d}x \int_{\varphi_1(x)}^{\varphi_2(x)} \mathrm{d}y f(x,y)\ . \tag{8.81}$$

Such cases are rather difficult to handle and the method to choose depends highly on the form of the functions $\varphi_1(x)$, $\varphi_2(x)$ and $f(x,y)$. We will not deal with integrals of this kind because this is beyond the scope of this book. The interested reader is referred to books by DAHLQUIST and BJÖRK [3] and by PRESS *et al.* [8].

In a final remark we would like to point out that it can be of advantage to utilize the properties of FOURIER transforms when integrals of the convolution type are to be approximated numerically (see Subsec. 3.3.2).

Bibliography

[1] Abramovitz, M., Stegun, I.A. (eds.): Handbook of Mathemathical Functions. Dover, New York (1965)

[2] Beals, R., Wong, R.: Special Functions. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, UK (2010)

[3] Dahlquist, G., Björk, Å.: Numerical Methods in Scientific Computing. Cambridge University Press, Cambridge, UK (2008)

[4] Fornberg, B.: A Practical Guide to Pseudospectral Methods. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, UK (1999)

[5] Jordan, C.: Calculus of Finite Differences, 3rd edn. AMS Chelsea Publishing, Providence (1965)

[6] Mathai, A.M., Haubold, H.J.: Special Functions for Applied Scientists. Springer, Berlin, Heidelberg (2008)

[7] Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W.: NIST Handbook of Mathematical Functions. Cambridge University Press, Cambridge, UK (2010)

[8] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[9] Sormann, H.: Numerische Methoden in der Physik. Institute of Theoretical and Computational Physics, Graz University of Technology, Austria (2011). Lecture Notes

[10] Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, 2nd edn. Springer, Berlin, Heidelberg (1993)

[11] Ueberhuber, C.W.: Numerical Computation 2: Methods, Software and Analysis. Springer, Berlin, Heidelberg (1997)

Chapter 9

The Kepler problem

9.1 Introduction

The Kepler problem [1, 3, 7, 5, 4, 6] is certainly one of the most important problems in the history of physics and natural sciences in general. We will study this problem for several reasons: (i) it is a nice demonstration of the applicability of the methods introduced in the previous chapters, (ii) important concepts of the numerical treatment of ordinary differential equations can be introduced quite naturally, and (iii) it allows to revisit some of the most important aspects of classical mechanics.

The Kepler problem is a special case of the two-body problem. Let us summarize the main results. We consider two point particles interacting via the rotationally symmetric two body potential U which is solely a function of the distance between the particles. The symmetries of this problem allow several simplifications: (i) The problem can be reduced to the two dimensional motion of a point particle with reduced mass m in the central potential U. (ii) By construction, the total energy E is conserved. (iii) The length ℓ of the angular momentum vector is also conserved because of the symmetry of the potential U. Due to this rotational symmetry it is a natural choice to describe the particle's motion in polar coordinates (ρ, φ).

The final differential equations which have to be solved are of the form

$$\dot{\varphi} = \frac{\ell}{m\rho^2} , \tag{9.1}$$

and

$$\dot{\rho} = \pm\sqrt{\frac{2}{m}\left[E - U(\rho) - \frac{\ell^2}{2m\rho^2}\right]} . \tag{9.2}$$

Here, one usually defines the effective potential

$$U_{\text{eff}}(\rho) = U(\rho) + \frac{\ell^2}{2m\rho^2}, \tag{9.3}$$

as the sum of the interaction potential and the centrifugal barrier $U_{\text{mom}}(\rho) = \ell^2/2m\rho^2$. Equation (9.2) can be transformed into an implicit equation for ρ

$$t = t_0 \pm \int_{\rho_0}^{\rho} \mathrm{d}\rho' \left\{\frac{2}{m}\left[E - U_{\text{eff}}(\rho')\right]\right\}^{-\frac{1}{2}} , \tag{9.4}$$

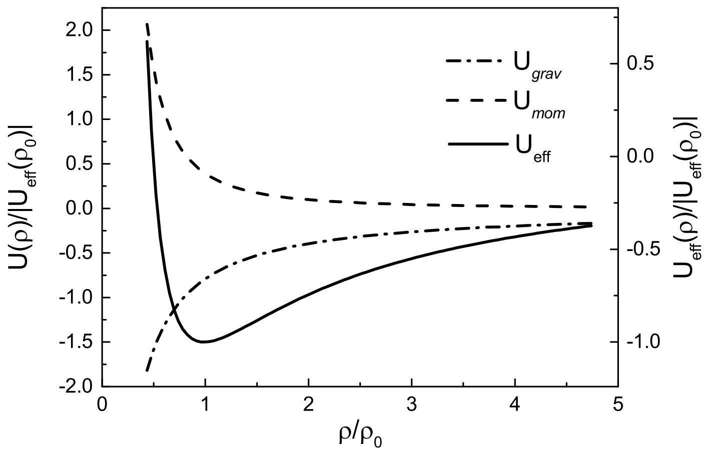

Figure 9.1: Schematic illustration of the effective potential $U_{\text{eff}}(\rho)/U_{\text{eff}}(\rho_0)$ vs ρ/ρ_0 (solid line, right hand scale). Here, ρ_0 is the distance of the minimum in $U_{\text{eff}}(\rho)$. $U_{\text{grav}}(\rho)$ (dashed-dotted line) denotes the gravitational contribution while $U_{\text{mom}}(\rho)$ (dashed line) denotes the centrifugal barrier. Both potentials are normalized to $U_{\text{eff}}(\rho_0)$. (Left hand scale applies.)

with $\rho_0 \equiv \rho(t_0)$ the initial condition at time t_0. Furthermore, the angle φ is related to the radius ρ by

$$\varphi = \varphi_0 \pm \int_{\rho_0}^{\rho} \mathrm{d}\rho' \frac{\ell}{m\rho'^2} \left\{ \frac{2}{m} \left[E - U_{\text{eff}}(\rho') \right] \right\}^{-\frac{1}{2}}, \tag{9.5}$$

with the initial condition $\varphi_0 \equiv \varphi(t_0)$.

The KEPLER problem is defined by the gravitational interaction potential

$$U(\rho) = -\frac{\alpha}{\rho}, \quad \alpha > 0. \tag{9.6}$$

Fig. 9.1 presents schematically the effective potential (9.3) (solid black line, together with the gravitational potential $U(\rho)$ (dashed-dotted line) and the centrifugal barrier U_{mom} (dashed line). The gravitational potential (9.6) is now inserted into Eq. (9.5):

$$\varphi = \varphi_0 \pm \int_{\rho_0}^{\rho} \mathrm{d}\rho' \frac{\ell}{m\rho'^2} \left[\frac{2}{m} \left(E + \frac{\alpha}{\rho'} - \frac{\ell^2}{2m\rho'^2} \right) \right]^{-\frac{1}{2}}. \tag{9.7}$$

The substitution $u = 1/\rho$ simplifies Eq. (9.7) to

$$\varphi = \varphi_0 \mp \int_{u_1}^{u_2} \mathrm{d}u \left[\frac{2mE}{\ell^2} + \frac{2m\alpha}{\ell^2} u - u^2 \right]^{-\frac{1}{2}}, \tag{9.8}$$

where the integration boundaries u_1 and u_2 are $1/\rho_0$ and $1/\rho$, respectively. The integral can now be evaluated with the help of a simple substitution[1] and we obtain

[1]In particular, we substitute

$$w = \left(u - \frac{m\alpha}{\ell^2} \right) \left(\frac{2mE}{\ell^2} + \frac{n^2\alpha^2}{\ell^4} \right)^{-\frac{1}{2}}.$$

the angle φ as a function of ρ:

$$\varphi = \varphi_0 \pm \cos^{-1}\left(\frac{\frac{\ell}{\rho} - \frac{m\alpha}{\ell}}{\sqrt{2mE + \frac{m^2\alpha^2}{\ell^2}}}\right) + \text{const} . \tag{9.9}$$

This solution can conveniently be characterized by the introduction of two parameters, namely

$$a = \frac{\ell^2}{m\alpha} \tag{9.10}$$

and the *eccentricity* e

$$e = \sqrt{1 + \frac{2E\ell^2}{m\alpha^2}} . \tag{9.11}$$

Hence, by neglecting the integration constant and setting $\varphi_0 = 0$ we arrive at

$$\frac{a}{\rho} = 1 + e\cos(\varphi) \tag{9.12}$$

as the final form of Eq. (9.9). It describes for $e > 1$ a hyperbola, for $e = 1$ a parabola and for $e < 1$ an ellipse. The case $e = 0$ is a special case of the ellipse and describes a circle with radius $\rho = a$. A more detailed discussion of this result,in particular the derivation of KEPLER's laws can be found in any textbook on classical mechanics [1, 3, 7, 5, 4, 6]. We discuss now some numerical aspects.

9.2 Numerical treatment

We aim at solving the integral equation (9.4) numerically with the help of the methods discussed in the previous chapter. Remember that Eq. (9.4) expresses the time t as a function of the radius ρ. This equation has to be inverted, in order to obtain $\rho(t)$, which, in turn, is then inserted into Eq. (9.1) in order to determine the angle $\varphi(t)$ as a function of time. This discussion will lead us in a natural way to the most common techniques applied to solve ordinary differential equations, which is of no surprise since Eq. (9.4) is the integral representation of Eq. (9.2).

We give a short outline of what we plan to do. We discretize the time axis in equally spaced time steps Δt, i.e. $t_n = t_0 + n\Delta t$. Accordingly, we define the radius ρ at time $t = t_n$ as $\rho(t_n) \equiv \rho_n$. We can use the methods introduced in Chap. 8 to approximate the integral (9.4) from some ρ_n to ρ_{n+1}. According to this chapter the absolute error introduced will behave like $\delta = |\rho_n - \rho_{n+1}|^K$ where the explicit value of K depends on the method used. However, since the radius ρ changes continuously with time t we know that for sufficiently small values of Δt the error δ will also become arbitrarily small. If we start from some initial values t_0 and ρ_0, we can successively calculate the values ρ_1, ρ_2 , . . . , by applying a small time step Δt.

Let us start by rewriting Eq. (9.4) as:

$$t - t_0 = \int_{\rho_0}^{\rho} d\rho' f(\rho') . \tag{9.13}$$

As we discretized the time axis in equally spaced increments $t_n = t_0 + n\Delta t$ and defined $\rho_n \equiv \rho(t_n)$, we can rewrite (9.13) as

$$\Delta t = t_n - t_{n-1} = \int_{\rho_n}^{\rho_{n+1}} d\rho' f(\rho') . \tag{9.14}$$

The forward rectangular rule, (8.9) results in the approximation

$$\Delta t = (\rho_{n+1} - \rho_n)\, f(\rho_n)\ . \tag{9.15}$$

We solve this equation for ρ_{n+1} and obtain

$$\rho_{n+1} = h(\rho_n)\Delta t + \rho_n\ , \tag{9.16}$$

where we defined

$$h(\rho) = \frac{1}{f(\rho)} = \sqrt{\frac{2}{m}\left[E - U_{\text{eff}}(\rho)\right]}\ , \tag{9.17}$$

following Eqs. (9.2) and (9.3). As Eq. (9.4) is the integral representation of the ordinary differential equation (9.2), approximation (9.16) corresponds to the approximation

$$D_+\rho_n = h(\rho_n)\ , \tag{9.18}$$

where $D_+\rho_n$ is the forward difference derivative (7.10a). Since the left hand side of the discretized differential equation (9.18) is independent of ρ_{n+1}, this method is referred to as an *explicit* method. In particular, consider an ordinary differential equation of the form

$$\dot{y} = F(y)\ . \tag{9.19}$$

Then the approximation method is referred to as an *explicit* EULER *method* if it is of the form

$$y_{n+1} = y_n + F(y_n)\Delta t\ . \tag{9.20}$$

Note that y might be a vector.

Let us use the backward rectangular rule (8.10) to solve Eq. (9.14). We obtain

$$t_{n+1} - t_n = (\rho_{n+1} - \rho_n)\, f(\rho_{n+1})\ , \tag{9.21}$$

or equivalently

$$\rho_{n+1} = \rho_n + h(\rho_{n+1})\Delta t\ . \tag{9.22}$$

Again, this corresponds to an approximation of the differential equation (9.2) by

$$D_-\rho_{n+1} = h(\rho_{n+1})\ , \tag{9.23}$$

where $D_-(\rho_{n+1})$ is the backward difference derivative (7.10b). In this case the quantity of interest ρ_{n+1} still appears in the argument of the function $h(\rho)$ and Eq. (9.22) is an *implicit* equation for ρ_{n+1} which has to be solved. In general, if the problem (9.19) is approximated by an algorithm of the form

$$y_{n+1} = y_n + F(y_{n+1})\Delta t\ , \tag{9.24}$$

it is referred to as an *implicit* EULER *method.* Note that the implicit equation (9.24) might be be analytically unsolvable. Hence, one has to employ a numerical method to solve (9.24) which will also imply a numerical error. However, in the particular case of Eq. (9.22) we can solve it analytically since it is a fourth order polynomial in ρ_{n+1} of the form

$$\rho_{n+1}^4 - 2\rho_n\rho_{n+1}^3 + \rho_n^2\rho_{n+1}^2 - \frac{2\alpha\Delta t^2}{m}\rho_{n+1} + \left(\ell^2 + \frac{2E}{m}\right)\Delta t^2 = 0\ . \tag{9.25}$$

The solution of this equation is quite tedious and will not be discussed here, however, the method one employs is referred to as FERRARI's method [2].

A natural way to proceed is to regard the central rectangular rule (8.13) in a next step. Within this approximation we obtain for Eq. (9.13)

$$\Delta t = (\rho_{n+1} - \rho_n) f\left(\frac{\rho_{n+1} + \rho_n}{2}\right) , \tag{9.26}$$

which is equivalent to the implicit equation

$$\rho_{n+1} = \rho_n + h\left(\frac{\rho_{n+1} + \rho_n}{2}\right) \Delta t . \tag{9.27}$$

It can be written as an approximation to Eq. (9.2) with help of the central difference derivative $D_c \rho_{n+\frac{1}{2}}$:

$$D_c \rho_{n+\frac{1}{2}} = h\left(\frac{\rho_{n+1} + \rho_n}{2}\right) . \tag{9.28}$$

In general, for a problem of the form (9.19) a method of the form

$$y_{n+1} = y_n + F\left(\frac{y_{n+1} + y_n}{2}\right) \Delta t , \tag{9.29}$$

is referred to as the *implicit midpoint rule*. We note that this method might be more accurate since the error of the rectangular rule scales like $\mathcal{O}(\Delta t^2)$ while the error of the forward and backward rectangular rules scale like $\mathcal{O}(\Delta t)$. Nevertheless, in case of the KEPLER problem, one can solve the implicit equation (9.27) analytically for ρ_{n+1} which is certainly of advantage.

In this chapter the KEPLER problem was instrumental in introducing three common methods which can be employed to solve numerically ordinary differential equations of the form (9.19). More general and advanced methods to solve ordinary differential equations and a more systematic description of these methods will be offered in the next chapter.

However, let us discuss another point before proceeding to the chapter on the numerics of ordinary differential equations. As demonstrated in Sec. 1.3 the approximation of the integral (9.4) involves a numerical error. What will be the consequence of this error? Since we demonstrated that the approximations we discussed result in a differential equation in finite difference form, i.e. Eqs. (9.18), (9.23), and (9.27), we know that the derivative $\dot{\rho}$ will exhibit an error. Consequently, energy conservation will be violated with the implication that deviations from the trajectory (9.12) can be expected. This is definitely not desirable.

A solution is provided by a special class of methods, known as *symplectic integrators* which were specifically designed for such cases. They are based on a formulation of the problem using HAMILTON's equations of motion. (See, for instance, Refs. [1, 3, 7, 5, 4].) In the particular case of the KEPLER problem the HAMILTON function is equivalent to the total energy of the system and reads (in some scaled units):

$$H(\mathbf{p}, \mathbf{q}) = \frac{1}{2}\left(p_1^2 + p_2^2\right) - \frac{1}{\sqrt{q_1^2 + q_2^2}} . \tag{9.30}$$

Here $\mathbf{p} = (p_1, p_2)^T$ are the generalized momentum coordinates of the point particle in the two-dimensional plane and $\mathbf{q} = (q_1, q_2)^T$ are the generalized position coordinates.

From this HAMILTON's equations of motion

$$\begin{pmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{p}} H(\mathbf{p}, \mathbf{q}) \\ -\nabla_{\mathbf{q}} H(\mathbf{p}, \mathbf{q}) \end{pmatrix} = \begin{pmatrix} a(\mathbf{q}, \mathbf{p}) \\ b(\mathbf{q}, \mathbf{p}) \end{pmatrix} \tag{9.31}$$

follow, where the functions $a(\mathbf{q}, \mathbf{p})$ and $b(\mathbf{q}, \mathbf{p})$ have been introduced for a more convenient notation. Note that these functions are two dimensional vectors in the case of KEPLER's problem. The so called *symplectic* EULER *method* is given by

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + a(\mathbf{q}_n, \mathbf{p}_{n+1})\Delta t, \\ \mathbf{p}_{n+1} &= \mathbf{p}_n + b(\mathbf{q}_n, \mathbf{p}_{n+1})\Delta t. \end{aligned} \tag{9.32}$$

Obviously, the first equation is explicit while the second is implicit. An alternative formulation reads

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + a(\mathbf{q}_{n+1}, \mathbf{p}_n)\Delta t, \\ \mathbf{p}_{n+1} &= \mathbf{p}_n + b(\mathbf{q}_{n+1}, \mathbf{p}_n)\Delta t, \end{aligned} \tag{9.33}$$

where the first equation is implicit and the second equation is explicit. Of course, Eq. (9.31) may be solved with the help of the explicit EULER method (9.20), the implicit EULER method (9.24) or the implicit midpoint rule (9.29). The solution should be equivalent to solving equation (9.4) with the respective method and then calculating (9.1) successively. Again, a more systematic discussion of symplectic integrators can be found in the following chapter.

Let us conclude this chapter with a final remark. We decided to solve Eqs. (9.4) and (9.1) because we wanted to reproduce the dynamics of the system, i.e. we wanted to obtain $\rho(t)$ and $\varphi(t)$. This directed us to the numerical solution of two integrals. If we wanted to employ symplectic methods, which provide several advantages, we would have to solve four differential equations (9.31) instead of two integrals. Moreover, if we are not interested in the time evolution of the system but in the form of the trajectory in general, we could simply evaluate the integral (9.5) analytically or, if an analytical solution is not feasible for the potential $U(\rho)$ one is interested in, numerically. Methods to approximate such an integral were extensively discussed in Chap. 8.

Bibliography

[1] Arnol'd, V.I.: Mathematical Methods of Classical Mechanics, 2nd edn. Graduate Texts in Mathematics, Vol. 60. Springer, Berlin, Heidelberg (1989)

[2] Clark, A.: Elements of Abstract Algebra. Dover, New York (1971)

[3] Fetter, A.L., Walecka, J.D.: Theoretical Mechanics of Particles and Continua. Dover Publications, New York (2004)

[4] Fließbach, T.: Mechanik, 7th edn. Lehrbuch zur Theoretischen Physik I. Springer, Berlin, Heidelberg (2015)

[5] Goldstein, H., Poole, C., Safko, J.: Classical Mechanics, 3rd edn. Addison-Wesley, Menlo Park (2013)

[6] Ó'Mathúna, D.: Integrable Systems in Celestial Mechanics. Progress in Mathematical Physics, Vol. 51. Birkhäuser Basel, Basel (2008)

[7] Scheck, F.: Mechanics, 5th edn. Springer, Berlin, Heidelberg (2010)

Chapter 10

Ordinary Differential Equations - Initial Value Problems

10.1 Introduction

This chapter introduces common numeric methods designed to solve *initial value problems*. The discussion of the KEPLER problem in the previous chapter allowed the introduction of three concepts, namely the implicit EULER method, the explicit EULER method, and the implicit midpoint rule. Furthermore, we mentioned the symplectic EULER method. In this chapter we plan to put these methods into a more general context and to discuss more advanced techniques.

Let us define the problem: We consider initial value problems of the form

$$\begin{cases} \dot{\mathbf{y}}(t) = f(\mathbf{y}, t), \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \tag{10.1}$$

where $\mathbf{y}(t) \equiv \mathbf{y}$ is an n-dimensional vector and $\mathbf{y}_0$ is referred to as the *initial value* of $\mathbf{y}$. Some remarks about the form of Eq. (10.1) are required:

(i) We note that by posing Eq. (10.1) we assume the differential equation to be *explicit* in $\dot{\mathbf{y}}$, i.e. initial value problems of the form

$$\begin{cases} G(\dot{\mathbf{y}}) = f(\mathbf{y}, t), \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \tag{10.2}$$

are only considered if $G(\dot{\mathbf{y}})$ is analytically invertible. For instance, we will not deal with differential equations of the form

$$\dot{\mathbf{y}} + \log(\dot{\mathbf{y}}) = 1. \tag{10.3}$$

(ii) We note that Eq. (10.1) is a *first order* differential equation in $\mathbf{y}$. However, this is in fact not a restriction since we can transform every explicit differential equation of order n into a coupled set of explicit first order differential equations. Let us demonstrate this. We regard an explicit differential equation of the form

$$\mathbf{y}^{(n)} = f(t; \mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, \ldots, \mathbf{y}^{(n-1)}), \tag{10.4}$$

where we defined $\mathbf{y}^{(k)} \equiv \frac{\mathrm{d}^k}{\mathrm{d}t^k}\mathbf{y}$. This equation is equivalent to the set

$$\begin{aligned}
\dot{y}_1 &= y_2, \\
\dot{y}_2 &= y_3, \\
\vdots \quad & \quad \vdots \\
\dot{y}_{n-1} &= y_n, \\
\dot{y}_n &= f(t, y_1, y_2, \dots, y_n),
\end{aligned} \tag{10.5}$$

which can be written as Eq. (10.1). Hence, we can attenuate the criterion discussed in point (i), that the differential equation has to be explicit in $\dot{\mathbf{y}}$, to the criterion that the differential equation of order n has to be explicit in the n-th derivative of $\mathbf{y}$, namely $\mathbf{y}^{(n)}$.

There is another point required to be discussed before moving on. The numerical treatment of initial value problems is of eminent importance in physics because many differential equations, which appear unspectacular at first glance, cannot be solved analytically. For instance, consider a first order differential equation of the type:

$$\dot{\mathbf{y}} = t^2 + \mathbf{y}^2. \tag{10.6}$$

Although this equation appears to be simple, one has to rely on numerical methods to obtain a solution. However, Eq. (10.6) is not *well posed* since the solution is ambiguous as long as no initial values are given. A numerical solution is only possible if the problem is completely defined. In many cases, one uses numerical methods although the problem is solvable with the help of analytic methods, simply because the solution would be too complicated. A numerical approach might be justified, however, one should always remember that, quote [5]:

Numerical methods are no excuse for poor analysis.

10.2 Simple Integrators

We start by reintroducing the methods already discussed in the previous chapter. Again, we discretize the time coordinate t via the relation $t_n = t_0 + n\Delta t$ and define $f_n \equiv f(t_n)$ accordingly. In the following we will refrain from noting the initial condition explicitly for a more compact notation. We investigate Eq. (10.1) at some particular time t_n:

$$\dot{\mathbf{y}}_n = f(\mathbf{y}_n, t_n). \tag{10.7}$$

Integrating both sides of (10.7) over the interval $[t_n, t_{n+1}]$ gives

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \int_{t_n}^{t_{n+1}} \mathrm{d}t' f[\mathbf{y}(t'), t']. \tag{10.8}$$

Note that Eq. (10.8) is exact and it will be our starting point in the discussion of several paths to a numeric solution of initial value problems. These solutions will be based on an approximation of the integral on the right hand side of Eq. (10.8) with the help of the methods already discussed in Chap. 8.

In the following we list four of the best known simple integration methods for initial value problems:

(1)

Applying the forward rectangular rule (8.9) to Eq. (10.8) yields

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f(\mathbf{y}_n, t_n)\Delta t + \mathcal{O}(\Delta t^2), \tag{10.9}$$

which is the explicit EULER method we encountered already in Sec. 9.2. This method is also referred to as the *forward* EULER *method*. In accordance to the forward rectangular rule, the leading term of the error of this method is proportional to Δt^2 as was pointed out in Sec. 8.2.

(2)

We use the backward rectangular rule (8.10) in Eq. (10.8) and obtain

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f(\mathbf{y}_{n+1}, t_{n+1})\Delta t + \mathcal{O}(\Delta t^2), \tag{10.10}$$

which is the implicit EULER method, also referred to as *backward* EULER *method*. As already highlighted in Sec. 9.2, it may be necessary to solve Eq. (10.10) numerically for $\mathbf{y}_{n+1}$. (Some notes on the numeric solution of non-linear equations can be found in Subsec. 6.2.3.)

(3)

The central rectangular rule (8.13) approximates Eq. (10.8) by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f(\mathbf{y}_{n+\frac{1}{2}}, t_{n+\frac{1}{2}})\Delta t + \mathcal{O}(\Delta t^3), \tag{10.11}$$

and we rewrite this equation in the form:

$$\mathbf{y}_{n+1} = \mathbf{y}_{n-1} + 2f(\mathbf{y}_n, t_n)\Delta t + \mathcal{O}(\Delta t^3). \tag{10.12}$$

This method is sometimes referred to as the *leap-frog* routine or STÖRMER - VERLET method. Note that the approximation

$$\mathbf{y}_{n+\frac{1}{2}} \approx \frac{\mathbf{y}_n + \mathbf{y}_{n+1}}{2}, \tag{10.13}$$

in Eq. (10.11) gives the implicit midpoint rule as it was introduced in Sec. 9.2.

(4)

Employing the trapezoidal rule (8.15) in an approximation to Eq. (10.8) yields

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}\left[f(\mathbf{y}_n, t_n) + f(\mathbf{y}_{n+1}, t_{n+1})\right] + \mathcal{O}(\Delta t^3). \tag{10.14}$$

This is an implicit method which has to be solved for y_{n+1}. It is generally known as the CRANK - NICOLSON *method* [4] or simply as *trapezoidal method*.

Methods (1), (2), and (4) are also known as one-step methods, since only function values at times t_n and t_{n+1} are used to propagate in time. In contrast, the leap-frog method is already a *multi-step* method since three different times appear in the expression. Basically, there are three different strategies to improve these rather simple methods:

- TAYLOR series methods: Use more terms in the TAYLOR expansion of $\mathbf{y}_{n+1}$.
- Linear Multi-Step methods: Use data from previous time steps $\mathbf{y}_k$, $k < n$ in order to cancel terms in the truncation error.
- RUNGE - KUTTA method: Use intermediate points within one time step.

We will briefly discuss the first two alternatives and then turn our attention to the RUNGE - KUTTA methods in the next section.

TAYLOR Series Methods

Because of Chap. 7 we are already familiar with the TAYLOR expansion (7.7) of the function y_{n+1} around the point y_n,

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \dot{\mathbf{y}}_n + \frac{\Delta t^2}{2} \ddot{\mathbf{y}}_n + \mathcal{O}(\Delta t^3). \tag{10.15}$$

We insert Eq. (10.7) into Eq. (10.15) and obtain

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t f(\mathbf{y}_n, t_n) + \frac{\Delta t^2}{2} \ddot{\mathbf{y}}_n + \mathcal{O}(\Delta t^3). \tag{10.16}$$

So far nothing has been gained since the truncation error is still proportional to Δt^2. However, calculating $\ddot{\mathbf{y}}_n$ with the help of Eq. (10.7) gives

$$\ddot{\mathbf{y}}_n = \frac{\mathrm{d}}{\mathrm{d}t} f(\mathbf{y}_n, t_n) = \dot{f}(\mathbf{y}_n, t_n) + f'(\mathbf{y}_n, t_n)\dot{\mathbf{y}}_n = \dot{f}(\mathbf{y}_n, t_n) + f'(\mathbf{y}_n, t_n) f(\mathbf{y}_n, t_n), \tag{10.17}$$

and this results together with Eq. (10.16) in:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t f(\mathbf{y}_n, t_n) + \frac{\Delta t^2}{2}\left[\dot{f}(\mathbf{y}_n, t_n) + f'(\mathbf{y}_n, t_n) f(\mathbf{y}_n, t_n)\right] + \mathcal{O}(\Delta t^3). \tag{10.18}$$

This manipulation reduced the local truncation error to orders of Δt^3. The derivatives of $f(\mathbf{y}_n, t_n)$, $f'(\mathbf{y}_n, t_n)$ and $\dot{f}(\mathbf{y}_n, t_n)$ can be approximated with the help of the methods discussed in Chap. 7, if an analytic differentiation is not feasible. The above procedure can be repeated up to arbitrary order in the TAYLOR expansion (10.15).

Linear Multi-Step Methods

A k-th order linear multi-step method is defined by the approximation

$$\mathbf{y}_{n+1} = \sum_{j=0}^{k} a_j \mathbf{y}_{n-j} + \Delta t \sum_{j=0}^{k+1} b_j f(\mathbf{y}_{n+1-j}, t_{n+1-j}), \tag{10.19}$$

of Eq. (10.8). The coefficients a_j and b_j have to be determined in such a way that the local truncation error is reduced. Two of the best known techniques are the so called second order ADAMS - BASHFORD methods

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{\Delta t}{2}\left[3f(\mathbf{y}_n, t_n) - f(\mathbf{y}_{n-1}, t_{n-1})\right], \tag{10.20}$$

and the second order rule (*backward differentiation formula*)

$$\mathbf{y}_{n+1} = \frac{1}{3}\left[4\mathbf{y}_n - \mathbf{y}_{n-1} + \frac{\Delta t}{2} f(\mathbf{y}_{n+1}, t_{n+1})\right] . \tag{10.21}$$

(For details please consult Refs. [13, 12, 18].)

We note in passing that the backward differentiation formula of arbitrary order can easily be obtained with the help of the operator technique introduced in Sec. 7.4, Eq. (7.27). One simply inserts the backward difference series (7.27) to arbitrary order into the right hand side of the differential equation (10.7).

In many cases, multi-step methods are based on the interpolation of previously computed values $\mathbf{y}_k$ by LAGRANGE polynomials. This interpolation is then inserted into Eq. (10.8) and integrated. However, a detailed discussion of such procedures is beyond the scope of these lecture notes. The interested reader is referred to Refs. [3, 19].

Nevertheless, let us make one last point. We note that Eq. (10.19) is explicit for $b_0 = 0$ and implicit for $b_0 \neq 0$. In many numerical realizations one combines implicit and explicit multi-step methods in such a way that the explicit result [solve Eq. (10.19) with $b_0 = 0$] is used as a guess to solve the implicit equation [solve Eq. (10.19) with $b_0 \neq 0$]. Hence, the explicit method *predicts* the value $\mathbf{y}_{n+1}$ and the implicit method *corrects* it. Such methods yield very good results and are commonly referred to as *predictor - corrector* methods [2].

10.3 RUNGE - KUTTA Methods

In contrast to linear multi-step methods, the idea in RUNGE - KUTTA methods (see, for instance, Ref. [3]) is to improve the accuracy by calculating intermediate grid-points within the interval $[t_n, t_{n+1}]$. We note that the approximation (10.11) resulting from the central rectangular rule is already such a method since the function value $\mathbf{y}_{n+1/2}$ at the grid-point $t_{n+1/2} = t_n + \Delta t/2$ is taken into account. We investigate this in more detail and rewrite Eq. (10.11):

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f(\mathbf{y}_{n+\frac{1}{2}}, t_{n+\frac{1}{2}})\Delta t + \mathcal{O}(\Delta t^3). \tag{10.22}$$

We now have to find appropriate approximations to $y_{n+1/2}$ which will increase the accuracy of Eq. (10.11). Our first choice is to replace $y_{n+1/2}$ with the help of the explicit EULER method, Eq. (10.9),

$$\mathbf{y}_{n+\frac{1}{2}} = \mathbf{y}_n + \frac{\Delta t}{2}\dot{\mathbf{y}}_n = \mathbf{y}_n + \frac{\Delta t}{2} f(\mathbf{y}_n, t_n), \tag{10.23}$$

which, inserted into Eq. (10.22) yields

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f\left[\mathbf{y}_n + \frac{\Delta t}{2} f(\mathbf{y}_n, t_n), t_n + \frac{\Delta t}{2}\right]\Delta t + \mathcal{O}(\Delta t^2). \tag{10.24}$$

We note that Eq. (10.24) is referred to as the *explicit midpoint rule*. In analogy we could have approximated $\mathbf{y}_{n+1/2}$ with the help of the averaged function value $\mu\mathbf{y}_{n+1/2}$ which results in

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f\left(\frac{\mathbf{y}_n + \mathbf{y}_{n+1}}{2}, t_n + \frac{\Delta t}{2}\right)\Delta t + \mathcal{O}(\Delta t^2). \tag{10.25}$$

This equation is referred to as the *implicit midpoint rule.* Let us explain how we obtain an estimate for the error in Eqs. (10.24) and (10.25). In case of Eq. (10.24) we investigate the term

$$\mathbf{y}_{n+1} - \mathbf{y}_n - f\left[\mathbf{y}_n + \frac{\Delta t}{2} f(\mathbf{y}_n, t_n), t_n + \frac{\Delta t}{2}\right] \Delta t.$$

The Taylor expansion of $\mathbf{y}_{n+1}$ and $f(\cdot)$ around the point $\Delta t = 0$ yields

$$\Delta t \left[\dot{\mathbf{y}}_n - f(\mathbf{y}_n, t_n)\right] + \frac{\Delta t^2}{2}\left[\ddot{\mathbf{y}} - \dot{f}(\mathbf{y}_n, t_n) - f'(\mathbf{y}_n, t_n)\dot{\mathbf{y}}_n\right] + \dots \quad (10.26)$$

We observe that the first term cancels because of Eq. (10.7). Consequently, the error is of order Δt^2. A similar argument holds for Eq. (10.25).

Let us introduce a more convenient notation for the above examples before we concentrate on a more general topic. It is presented in algorithmic form, i.e. it defines the sequence in which one should calculate the various terms. This is convenient for two reasons, first of all it increases the readability of complex methods such as Eq. (10.25) and, secondly, it is easy to identify which part of the method involves an implicit step and which part has to be solved separately for the corresponding variable. For this purpose let us introduce vectors $\mathbf{Y}_i$ of some index $i \geq 1$ and we use a simple example to illustrate this notation. Consider the explicit Euler method (10.9). It can be written as

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{y}_n, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + f(\mathbf{Y}_1, t_n)\Delta t. \end{aligned} \quad (10.27)$$

In a similar fashion we write the implicit Euler method (10.10) as

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{y}_n + f(\mathbf{Y}_1, t_{n+1})\Delta t, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + f(\mathbf{Y}_1, t_{n+1})\Delta t. \end{aligned} \quad (10.28)$$

It is understood that the first equation of (10.28) has to be solved for $\mathbf{Y}_1$ first and this result is then plugged into the second equation in order to obtain $\mathbf{y}_{n+1}$. One further example: the Crank - Nicolson (10.14) method can be rewritten as

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{y}_n\ , \\ \mathbf{Y}_2 &= \mathbf{y}_n + \frac{\Delta t}{2}\left[f(\mathbf{Y}_1, t_n) + f(\mathbf{Y}_2, t_{n+1})\right], \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{\Delta t}{2}\left[f(\mathbf{Y}_1, t_n) + f(\mathbf{Y}_2, t_{n+1})\right], \end{aligned} \quad (10.29)$$

where the second equation is to be solved for $\mathbf{Y}_2$ in the second step.

In analogy, the algorithmic form of the explicit midpoint rule (10.24) is defined as

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{y}_n, \\ \mathbf{Y}_2 &= \mathbf{y}_n + \frac{\Delta t}{2} f\left(\mathbf{Y}_1, t_n\right), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \Delta t f\left(\mathbf{Y}_2, t_n + \frac{\Delta t}{2}\right), \end{aligned} \quad (10.30)$$

and we find for the implicit midpoint rule (10.25):

$$\begin{aligned}\mathbf{Y}_1 &= \mathbf{y}_n + \frac{\Delta t}{2} f\left(\mathbf{Y}_1, t_n + \frac{\Delta t}{2}\right), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \Delta t f\left(\mathbf{Y}_1, t_n + \frac{\Delta t}{2}\right).\end{aligned} \tag{10.31}$$

The above algorithms are all examples of the so called RUNGE - KUTTA methods. We introduce the general representation of a d-stage RUNGE - KUTTA method:

$$\begin{aligned}\mathbf{Y}_i &= \mathbf{y}_n + \Delta t \sum_{j=1}^{d} a_{ij} f\left(\mathbf{Y}_j, t_n + c_j \Delta t\right), \qquad i = 1, \ldots, d, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \Delta t \sum_{j=1}^{d} b_j f\left(\mathbf{Y}_j, t_n + c_j \Delta t\right).\end{aligned} \tag{10.32}$$

We note that Eq. (10.32) it is completely determined by the coefficients a_{ij}, b_j and c_j. In particular $a = \{a_{ij}\}$ is a $d \times d$ matrix, while $\mathbf{b} = \{b_j\}$ and $\mathbf{c} = \{c_j\}$ are d dimensional vectors.

BUTCHER tableaus are a very useful tool to characterize such methods. They provide a structured representation of the coefficient matrix a and the coefficient vectors b and c:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1d} \\ c_2 & a_{21} & a_{22} & \ldots & a_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_d & a_{d1} & a_{d2} & \ldots & a_{dd} \\ \hline & b_1 & b_2 & \ldots & b_d \end{array} \tag{10.33}$$

We note that the RUNGE - KUTTA method (10.32) or (10.33) is explicit if the matrix a is zero on and above the diagonal, i.e. $a_{ij} = 0$ for $j \geq i$. Let us rewrite all the methods described here in the form of BUTCHER tableaus:

Explicit EULER:

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \tag{10.34}$$

Implicit EULER:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \tag{10.35}$$

CRANK - NICOLSON:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{10.36}$$

Explicit Midpoint:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array} \tag{10.37}$$

Implicit Midpoint:

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \tag{10.38}$$

With the help of Runge - Kutta methods of the general form (10.32) one can develop methods of arbitrary accuracy. One of the most popular methods is the explicit four stage method (we will call it *e-RK-4*) which is defined by the algorithm:

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{y}_n, \\ \mathbf{Y}_2 &= \mathbf{y}_n + \frac{\Delta t}{2} f(\mathbf{Y}_1, t_n), \\ \mathbf{Y}_3 &= \mathbf{y}_n + \frac{\Delta t}{2} f\left(\mathbf{Y}_2, t_n + \frac{\Delta t}{2}\right), \\ \mathbf{Y}_4 &= \mathbf{y}_n + \Delta t f\left(\mathbf{Y}_3, t_n + \frac{\Delta t}{2}\right), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{\Delta t}{6}\left[f(\mathbf{Y}_1, t_n) + 2f\left(\mathbf{Y}_2, t_n + \frac{\Delta t}{2}\right)\right. \\ &\quad \left. +2f\left(\mathbf{Y}_3, t_n + \frac{\Delta t}{2}\right) + f(\mathbf{Y}_4, t_{n+1})\right]. \end{aligned} \tag{10.39}$$

This method is an analogue to the Simpson rule of numerical integration as discussed in Sec. 8.4. However, a detailed compilation of the coefficient array a and coefficient vectors b, and c is quite complicated. A closer inspection reveals that the methodological error of this method behaves as Δt^5. The algorithm *e-RK-4*, Eq. (10.39), is represented by a Butcher tableau of the form

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{10.40}$$

Another quite popular method is given by the Butcher tableau

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{10.41}$$

We note that this method is implicit and mention that it corresponds to the two point Gauss - Legendre quadrature of Sec. 8.6.

A further improvement of implicit Runge - Kutta methods can be achieved by *choosing* the $\mathbf{Y}_i$ in such a way that they correspond to solutions of the differential equation (10.7) at intermediate time steps. The intermediate time steps at which one wants to reproduce the function are referred to as *collocation points*. At these points the functions are approximated by interpolation on the basis of LAGRANGE polynomials, which can easily be integrated analytically. However, the discussion of such collocation methods [2] is far beyond the scope of these lecture notes.

In general RUNGE - KUTTA methods are very useful. However one always has to keep in mind that there could be better methods for the problem at hand. Let us close this section with a quote from the book by PRESS *et al.* [16]:

> *For many scientific users, fourth-order Runge-Kutta is not just the first word on ODE integrators, but the last word as well. In fact, you can get pretty far on this old workhorse, especially if you combine it with an adaptive step-size algorithm. Keep in mind, however, that the old workhorse's last trip may well take you to the poorhouse: Bulirsch-Stoer or predictor-corrector methods can be very much more efficient for problems where high accuracy is a requirement. Those methods are the high-strung racehorses. Runge-Kutta is for ploughing the fields.*

10.3.1 Error estimate and stepsize adaptation

We rewrite Eq. (10.1) as:

$$\begin{cases} \mathbf{y}'(x) = f(x;\mathbf{y}), \\ \mathbf{y}(x_0) = \mathbf{y}_0, \end{cases} \tag{10.42}$$

The starting point for all further arguments is the expansion of the i^{th} solution of the system (10.42) in a power series:

$$y_i(x) = \sum_{\nu=0}^{p} \frac{(x-x_0)^\nu}{\nu!} \left[\frac{\mathrm{d}^\nu}{\mathrm{d}x^\nu} y_i(x)\right]_{x_0,\mathbf{y}_0} + R_i, \tag{10.43}$$

with the LAGRANGE remainder

$$R_i = \frac{(x-x_0)^{p+1}}{(p+1)!} \left[\frac{\mathrm{d}^{p+1}}{\mathrm{d}x^{p+1}} y_i(x)\right]_{x=\xi}, \quad x_0 \le \xi < x, \tag{10.44}$$

and p is the *order of the method.*

As the RUNGE - KUTTA methods are strongly related to the TAYLOR series expansion (10.43), a natural choice is to use the corresponding LAGRANGE remainder (10.44) for an estimate of the methodological error. This gives for the classical RUNGE - KUTTA method (order $p = 4$)

$$\varepsilon_V = \frac{h^5}{120} \left[y_i^{(5)}(x)\right]_{x=\xi}, \quad x_0 \le \xi \le x_0 + h,$$

and

$$\varepsilon_V = C(h)h^5, \tag{10.45}$$

where h symbolizes the step size. ε_V represents the error between the exact value of the solution $y_i(x)$ in the point $x_0 + h$ and the corresponding approximate solution $\hat{y}_i(x_0 + h)$ and we have:

$$y_i(x_0 + h) = \hat{y}_i(x_0 + h) + C(h)h^5. \tag{10.46}$$

We get the equivalent result if move from x_0 to $x_0 + h$ with two successive RUNGE - KUTTA moves of width $h/2$:

$$y_i(x_0 + h) = \hat{y}\left(x_0 + 2\frac{h}{2}\right) + 2C\left(\frac{h}{2}\right)\left(\frac{h}{2}\right)^5 . \tag{10.47}$$

For easier notation we drop the index i of y_i, use the abbreviations

$$\hat{y}(x_0 + h) \equiv \hat{y}(h), \quad \hat{y}\left(x_0 + 2\frac{h}{2}\right) \equiv \hat{y}(h/2),$$

and assume $C(h)$ not to depend "too strongly" on h:

$$C(h) \approx C\left(\frac{h}{2}\right) = C.$$

The constant C and the corresponding methodological error $\varepsilon_V(h) = Ch^5$ can be calculated by setting Eq. (10.46) equal to Eq. (10.47)

$$\varepsilon_V(h) = \frac{16}{15}\left[\hat{y}(h/2) - \hat{y}(h)\right] \approx \hat{y}(h/2) - \hat{y}(h), \tag{10.48}$$

with the result that we can estimate the (absolute) methodological error from the difference between the two RUNGE - KUTTA solutions $\hat{y}(h/2)$ (= approximate solution after two half moves) and $\hat{y}(h)$ (= approximate solution after one full move).

Furthermore, the combination of Eqs. (10.47) and (10.48) results in a relation that permits to improve the approximate value $\hat{y}$ in a simple way:

$$\hat{y}_{\text{imp}} = \hat{y}(h/2) + \frac{\hat{y}(h/2) - \hat{y}(h)}{16}. \tag{10.49}$$

Hovever, there is a problem connected with this method. Clearly, we obtain a better approximation for the solution, but we have little information on the quality of this corrected value $\hat{y}$! This is the reason why formula is rarely used in programs.

We can, finally, ask how large the ideal stepsize h should be, in order to ensure that ε_V does not exceed a given error threshold ε. As the methodological error is proportional to the fifth power of h the answer to this question is easily found. We have in fact:

$$\frac{\varepsilon}{\varepsilon_V(h)} = \left(\frac{h_{\text{ideal}}}{h}\right)^5$$

from which we immediately obtain a definition for h_{ideal}:

$$h_{\text{ideal}} = h\left[\frac{\varepsilon}{\hat{y}(h/2) - \hat{y}(h)}\right]^{1/5} . \tag{10.50}$$

This formula is the basis for a *stepsize adaptation* scheme employed by some programs.

10.4 Hamiltonian Systems - Symplectic Integrators

Let us define a symplectic integrator as a numerical integration in which the mapping

$$\Phi_{\Delta t} : \mathbf{y}_n \mapsto \mathbf{y}_{n+1}, \tag{10.51}$$

is symplectic. Here $\Phi_{\Delta t}$ is referred to as the *numerical flow* of the method. If we regard the initial value problem (10.1) we can define in an analogous way the *flow of the system* φ_t as

$$\varphi_t(\mathbf{y}_0) = \mathbf{y}(t). \tag{10.52}$$

For instance, if we consider the initial value problem

$$\begin{cases} \dot{\mathbf{y}} = A\mathbf{y}, \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \tag{10.53}$$

where $\mathbf{y} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$, then the flow φ_t of the system is given by:

$$\varphi_t(\mathbf{y}_0) = \exp(At)\mathbf{y}_0. \tag{10.54}$$

On the other hand, if we regard two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^2$, we can express the area ω of the parallelogram spanned by these vectors as

$$\omega(\mathbf{v},\mathbf{w}) = \det(\mathbf{v}\mathbf{w}) = \mathbf{v}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\mathbf{w} = ad - bc, \tag{10.55}$$

where we put $\mathbf{v} = (a,b)^T$ and $\mathbf{w} = (c,d)^T$. More generally, if $\mathbf{v},\mathbf{w} \in \mathbb{R}^{2d}$, we have

$$\omega(\mathbf{v},\mathbf{w}) = \mathbf{v}\begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}\mathbf{w} \equiv \mathbf{v}J\mathbf{w}, \tag{10.56}$$

where I is the $d \times d$ dimensional unity matrix. Hence (10.56) represents the sum of the projected areas of the form

$$\det\begin{pmatrix} v_i & w_i \\ v_{i+d} & w_{i+d} \end{pmatrix}. \tag{10.57}$$

If we regard a mapping $M : \mathbb{R}^{2d} \mapsto \mathbb{R}^{2d}$ and require that

$$\omega(M\mathbf{v}, M\mathbf{w}) = \omega(\mathbf{v},\mathbf{w}), \tag{10.58}$$

i.e. the area is preserved, we obtain the condition that

$$M^TJM = J, \tag{10.59}$$

which is equivalent to $\det(M) = 1$. Finally, a differentiable mapping $f : \mathbb{R}^{2d} \mapsto \mathbb{R}^{2d}$ is referred to as *symplectic* if the linear mapping $f'(\mathbf{x})$ (Jacobi matrix) conserves ω for all $\mathbf{x} \in \mathbb{R}^{2d}$. One can easily prove that the flow of Hamiltonian systems (energy conserving) is symplectic, i.e. area preserving in phase space. Every Hamiltonian system is characterized by its Hamilton function $H(\mathbf{p},\mathbf{q})$ and the corresponding Hamilton equations of motion [1, 7, 17, 9, 8]:

$$\dot{\mathbf{p}} = -\nabla_{\mathbf{q}}H(\mathbf{p},\mathbf{q}) \quad \text{and} \quad \dot{\mathbf{q}} = \nabla_{\mathbf{p}}H(\mathbf{p},\mathbf{q}). \tag{10.60}$$

We define the flow of the system via

$$\varphi_t(\mathbf{x}_0) = \mathbf{x}(t), \tag{10.61}$$

where

$$\mathbf{x}_0 = \begin{pmatrix} \mathbf{p}_0 \\ \mathbf{q}_0 \end{pmatrix} \quad \text{and} \quad \mathbf{x}(t) = \begin{pmatrix} \mathbf{p}(t) \\ \mathbf{q}(t) \end{pmatrix}. \tag{10.62}$$

Hence we rewrite (10.60) as

$$\dot{\mathbf{x}} = J^{-1}\nabla_{\mathbf{x}} H(\mathbf{x}), \tag{10.63}$$

and note that $\mathbf{x} \equiv \mathbf{x}(t, \mathbf{x}_0)$ is a function of time and initial conditions. In a next step we define the Jacobian of the flow via

$$P_t(\mathbf{x}_0) = \nabla_{\mathbf{x}_0}\varphi_t(\mathbf{x}_0) \ , \tag{10.64}$$

and calculate

$$\begin{aligned}
\dot{P}_t(\mathbf{x}_0) &= \nabla_{\mathbf{x}_0}\dot{\mathbf{x}} \\
&= J^{-1}\nabla_{\mathbf{x}_0}\nabla_{\mathbf{x}} H(\mathbf{x}) \\
&= J^{-1}\Delta_{\mathbf{x}} H(\mathbf{x})\nabla_{\mathbf{x}_0}\mathbf{x} \\
&= J^{-1}\Delta_{\mathbf{x}} H(\mathbf{x}) P_t(\mathbf{x}_0) \\
&= \begin{pmatrix} -\nabla_{\mathbf{qp}} H(\mathbf{p},\mathbf{q}) & -\nabla_{\mathbf{qq}} H(\mathbf{p},\mathbf{q}) \\ \nabla_{\mathbf{pp}} H(\mathbf{p},\mathbf{q}) & \nabla_{\mathbf{pq}} H(\mathbf{p},\mathbf{q}) \end{pmatrix} P_t(\mathbf{x}_0).
\end{aligned} \tag{10.65}$$

Hence, P_t is given by the solution of the equation

$$\dot{P}_t = J^{-1}\Delta_{\mathbf{x}} H(\mathbf{x}) P_t. \tag{10.66}$$

Symplecticity ensures that the area

$$P_t^T J P_t = \text{const}, \tag{10.67}$$

which can be verified by calculating $\frac{\mathrm{d}}{\mathrm{d}t}\left(P_t^T J P_t\right)$ where we keep in mind that $J^T = -J$. Hence,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} P_t^T J P_t &= \dot{P}_t^T J P_t + P_t^T J \dot{P}_t \\
&= P_t^T \Delta_{\mathbf{x}} H(\mathbf{x}) (J^{-1})^T J P_t + P_t^T J J^{-1} \Delta_{\mathbf{x}} H(\mathbf{x}) P_t \\
&= 0,
\end{aligned} \tag{10.68}$$

even if the Hamilton function is not conserved. This means that the flow of a Hamiltonian system is symplectic, i.e. area preserving in phase space [1, 7, 9].

Since this conservation law is violated by methods like *e-RK-4* or explicit Euler, one introduces so called *symplectic integrators*, which have been particularly designed as a remedy to this shortcoming. A detailed investigation of these techniques is far too engaged for these lecture notes. The interested reader is referred to Refs. [17, 10, 11, 14].

However, we provide a list of the most important integrators.

Symplectic Euler

$$q_{n+1} = q_n + a(q_n, p_{n+1})\Delta t \ , \tag{10.69a}$$
$$p_{n+1} = p_n + b(q_n, p_{n+1})\Delta t \ . \tag{10.69b}$$

Here $a(p,q) = \nabla_p H(p,q)$ and $b(p,q) = -\nabla_q H(p,q)$ have already been defined in Sec. 9.2.

Symplectic Runge - Kutta

It can be demonstrated that a Runge - Kutta method is symplectic if the coefficients fulfill

$$b_i a_{ij} + b_j a_{ji} = b_i b_j \;, \tag{10.70}$$

for all i, j [11, 6]. This is a property of the collocation methods based on Gauss points c_i.

10.5 An Example - The Kepler Problem, Revisited

It has already been discussed in Sec. 9.2 that the Hamilton function of this system takes on the form [15]

$$H(\mathbf{p}, \mathbf{q}) = \frac{1}{2}\left(p_1^2 + p_2^2\right) - \frac{1}{\sqrt{q_1^2 + q_2^2}}, \tag{10.71}$$

and Hamilton's equations of motion read

$$\begin{aligned}
\dot{p}_1 &= -\nabla_{q_1} H(\mathbf{p}, \mathbf{q}) = -\frac{q_1}{(q_1^2 + q_2^2)^{\frac{3}{2}}}, && (10.72\text{a})\\
\dot{p}_2 &= -\nabla_{q_2} H(\mathbf{p}, \mathbf{q}) = -\frac{q_2}{(q_1^2 + q_2^2)^{\frac{3}{2}}}, && (10.72\text{b})\\
\dot{q}_1 &= \nabla_{p_1} H(\mathbf{p}, \mathbf{q}) = p_1, && (10.72\text{c})\\
\dot{q}_2 &= \nabla_{p_2} H(\mathbf{p}, \mathbf{q}) = p_2 \;. && (10.72\text{d})
\end{aligned}$$

We now introduce the time instances $t_n = t_0 + n\Delta t$ and define $q_i^n \equiv q_i(t_n)$ and $p_i^n \equiv p_i(t_n)$ for $i = 1, 2$. In the following we give the discretized recursion relation for three different methods, namely explicit Euler, implicit Euler, and symplectic Euler.

Explicit Euler

In case of the explicit Euler method we have simple recursion relations

$$\begin{aligned}
p_1^{n+1} &= p_1^n - \frac{q_1^n \Delta t}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} \;, && (10.73\text{a})\\
p_2^{n+1} &= p_2^n - \frac{q_2^n \Delta t}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} \;, && (10.73\text{b})\\
q_1^{n+1} &= q_1^n + p_1^n \Delta t \;, && (10.73\text{c})\\
q_2^{n+1} &= q_2^n + p_2^n \Delta t \;. && (10.73\text{d})
\end{aligned}$$

Implicit Euler

We obtain the implicit equations

$$\begin{aligned}
p_1^{n+1} &= p_1^n - \frac{q_1^{n+1}\Delta t}{[(q_1^{n+1})^2 + (q_2^{n+1})^2]^{\frac{3}{2}}} , && (10.74a)\\
p_2^{n+1} &= p_2^n - \frac{q_2^{n+1}\Delta t}{[(q_1^{n+1})^2 + (q_2^{n+1})^2]^{\frac{3}{2}}} , && (10.74b)\\
q_1^{n+1} &= q_1^n + p_1^{n+1}\Delta t , && (10.74c)\\
q_2^{n+1} &= q_2^n + p_2^{n+1}\Delta t . && (10.74d)
\end{aligned}$$

These implicit equations can be solved, for instance, by the use of the Newton method discussed in Subsec. 6.2.3.

Symplectic Euler

Employing Eqs. (10.69) gives

$$\begin{aligned}
p_1^{n+1} &= p_1^n - \frac{q_1^n\Delta t}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} , && (10.75a)\\
p_2^{n+1} &= p_2^n - \frac{q_2^n\Delta t}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} , && (10.75b)\\
q_1^{n+1} &= q_1^n + p_1^{n+1}\Delta t , && (10.75c)\\
q_2^{n+1} &= q_2^n + p_2^{n+1}\Delta t . && (10.75d)
\end{aligned}$$

These implicit equations can be solved analytically and we obtain

$$\begin{aligned}
p_1^{n+1} &= p_1^n - \frac{q_1^n\Delta t}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} , && (10.76a)\\
p_2^{n+1} &= p_2^n - \frac{q_2^n\Delta t}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} , && (10.76b)\\
q_1^{n+1} &= q_1^n + p_1^n\Delta t - \frac{q_1^n\Delta t^2}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} , && (10.76c)\\
q_2^{n+1} &= q_2^n + p_2^n\Delta t - \frac{q_2^n\Delta t^2}{[(q_1^n)^2 + (q_2^n)^2]^{\frac{3}{2}}} . && (10.76d)
\end{aligned}$$

A second possibility of the symplectic Euler is given by Eq. (9.33). It reads

$$\begin{aligned}
p_1^{n+1} &= p_1^n - \frac{q_1^{n+1}\Delta t}{[(q_1^{n+1})^2 + (q_2^{n+1})^2]^{\frac{3}{2}}} , && (10.77a)\\
p_2^{n+1} &= p_2^n - \frac{q_2^{n+1}\Delta t}{[(q_1^{n+1})^2 + (q_2^{n+1})^2]^{\frac{3}{2}}} , && (10.77b)\\
q_1^{n+1} &= q_1^n + p_1^n\Delta t , && (10.77c)\\
q_2^{n+1} &= q_2^n + p_2^n\Delta t . && (10.77d)
\end{aligned}$$

The trajectories calculated using these four methods are presented in Figs. 10.1 and 10.2, the time evolution of the total energy of the system is plotted in Fig. 10.3.

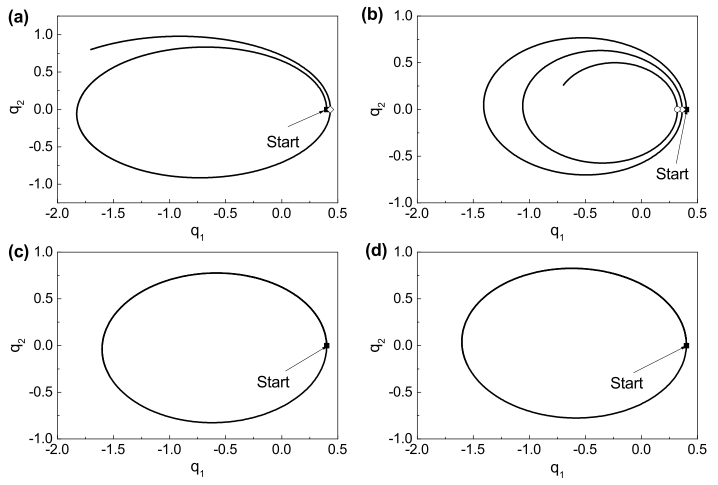

Figure 10.1: KEPLER trajectories in position space for the initial values defined in Eqs. (10.78) and (10.79). They are indicated by a solid square. Solutions have been generated **(a)** by the explicit EULER method (10.73), **(b)** by the implicit EULER method (10.74), **(c)** by the symplectic EULER method (10.76), and **(d)** by the symplectic EULER method (10.77).

The initial conditions were [11]

$$p_1(0) = 0, \qquad q_1(0) = 1 - e \, , \tag{10.78}$$

and

$$p_2(0) = \sqrt{\frac{1+e}{1-e}}, \qquad q_2(0) = 0 \, , \tag{10.79}$$

with $e = 0.6$ which gives $H = -1/2$. Furthermore, we set $\Delta t = 0.01$ for the symplectic EULER methods and $\Delta t = 0.005$ for the forward and backward EULER methods in order to reduce the methodological error. The implicit equations were solved with help of the NEWTON method as discussed in Subsec. 6.2.3. The JACOBI matrix was calculated analytically, hence no methodological error enters because approximations of derivatives were unnecessary.

According to theory [15] the q-space and p-space projections of the phase space trajectory are ellipses. Furthermore, energy and angular momentum are conserved. Thus, the numerical solutions of HAMILTON's equations of motion (10.72) should reflect these properties. Figs. 10.1(a,b) and 10.2(a,b) present the results of the explicit EULER method, Eqs. (10.73), and the implicit EULER method, Eqs. (10.74), respectively. Obviously, the result does not agree with the theoretical expectation and the trajectories are open instead of closed. The reason for this behavior is the methodological error of the method which is accumulative and, thus, causes a violation of energy conservation. This violation becomes apparent in Fig. 10.3 where the total energy $H(t)$ is plotted vs time t. Neither the explicit EULER method (dashed line) nor the implicit EULER method (short dashed line) conform to the requirement of energy conservation. We also see step-like structures of $H(t)$. At the center of these steps an open diamond symbol and in the case of the implicit EULER method an additional open circle indicate the position in time of the perihelion of

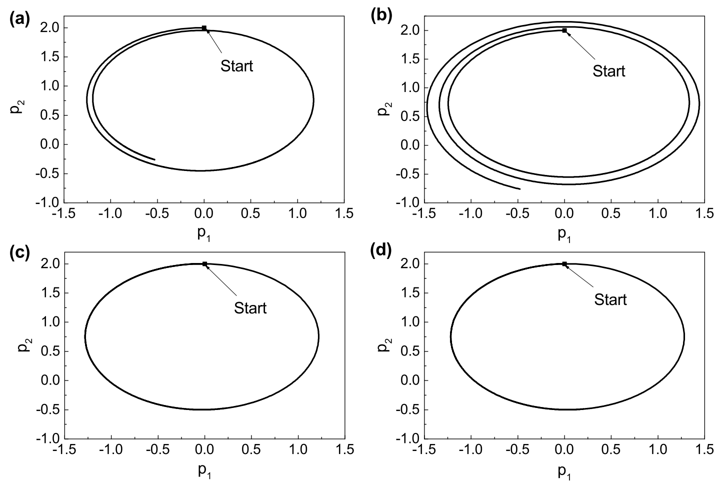

Figure 10.2: Kepler trajectories in momentum space for the initial values defined in Eqs. (10.78) and (10.79). They are indicated by a solid square.Solutions have been generated **(a)** by the explicit Euler method (10.73), **(b)** by the implicit Euler method (10.74), **(c)** by the symplectic Euler method (10.76), and **(d)** by the symplectic Euler method (10.77).

the point-mass (point of closest approach to the center of attraction). It is indicated by the same symbols in Figs. 10.1(a,b). At this point the point-mass reaches its maximum velocity, the pericenter velocity, and it covers the biggest distances along its trajectory per time interval Δt. Consequently, the methodological error is biggest in this part of the trajectory which manifests itself in those steps in $H(t)$. As the point-mass moves 'faster' when the implicit Euler method is applied, again, the distances covered per time interval are greater than those covered by the point-mass in the explicit Euler method. Thus, it is not surprising that the error of the implicit Euler method is bigger as well when $H(t)$ is determined.

These results are in strong contrast to the numerical solutions of Eqs. (10.72) obtained with the help of symplectic Euler methods which are presented in Figs. 10.1(c,d) and 10.2(c,d). The trajectories are almost perfect ellipses for both symplectic methods which follow Eqs. (10.76) and (10.77). Moreover, the total energy $H(t)$ (solid and dashed-dotted lines in Fig. 10.3) varies very little as a function of t. Deviations from the mean value can only be observed around the perihelion which is indicated by a solid square. Moreover, these deviations compensate because of the symplectic nature of the method. This proves that symplectic integrators are the appropriate technique to solve the equations of motion of Hamiltonian systems.

Bibliography

[1] Arnol'd, V.I.: Mathematical Methods of Classical Mechanics, 2nd edn. Graduate Texts in Mathematics, Vol. 60. Springer, Berlin, Heidelberg (1989)

[2] Ascher, U.M., Petzold, L.R.: Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. Society for Industrial and Applied Mathematics, Philadelphia (1998)

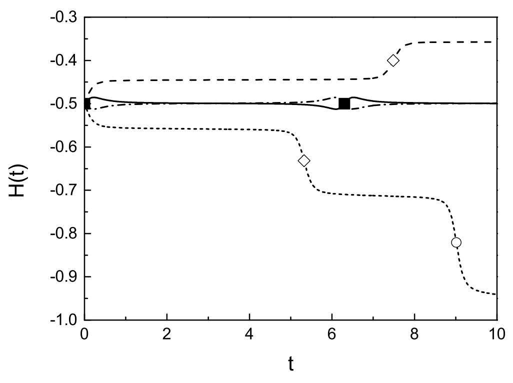

Figure 10.3: Time evolution of the total energy H calculated with the help of the four methods discussed in the text. The initial values are given by Eqs. (10.78) and (10.79).Solutions have been generated (i) by the explicit Euler method (10.73) (*dashed line*), (ii) by the implicit Euler method (10.74) (*dotted line*), (iii) by the symplectic Euler method (10.76) (*solid line*), and (iv) by the symplectic Euler method (10.77) (*dashed-dotted line*).

[3] Collatz, L.: The Numerical Treatment of Differential Equations. Springer, Berlin, Heidelberg (1960)

[4] Crank, J., Nicolson, P.: A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. Proc. Cambridge Philos. Soc. **43**, 50–67 (1947). DOI 10.1017/S0305004100023197

[5] Dorn, W.S., McCracken, D.D.: Numerical Methods with Fortran IV Case Studies. Wiley, New York (1972)

[6] Feng, K., Qin, M.: Symplectic Runge-Kutta Methods. In: K. Feng, M. Qin (eds.) Symplectic Geometric Algorithms for Hamiltonian Systems, pp. 277 – 364. Springer, Berlin, Heidelberg (2010)

[7] Fetter, A.L., Walecka, J.D.: Theoretical Mechanics of Particles and Continua. Dover Publications, New York (2004)

[8] Fließbach, T.: Mechanik, 7th edn. Lehrbuch zur Theoretischen Physik I. Springer, Berlin, Heidelberg (2015)

[9] Goldstein, H., Poole, C., Safko, J.: Classical Mechanics, 3rd edn. Addison-Wesley, Menlo Park (2013)

[10] Guillemin, V., Sternberg, S.: Symplectic Techniques in Physics. Cambridge University Press, Cambridge, UK (1990)

[11] Hairer, E.: Geometrical integration - symplectic integrators. TU München, Germany (2010). Lecture Notes

[12] Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I, 2nd edn. Springer Series in Computational Mathematics, Vol. 8. Springer, Berlin, Heidelberg (1993)

[13] Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Springer Series in Computational Mathematics, Vol. 14. Springer, Berlin, Heidelberg (1991)

[14] Levi, D., Oliver, P., Thomova, Z., Winteritz, P. (eds.): Symmetries and integrability of Difference Equations. London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, UK (2011)

[15] Ó'Mathúna, D.: Integrable Systems in Celestial Mechanics. Progress in Mathematical Physics, Vol. 51. Birkhäuser Basel, Basel (2008)

[16] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[17] Scheck, F.: Mechanics, 5th edn. Springer, Berlin, Heidelberg (2010)

[18] Süli, E., Mayers, D.: An Introduction to Numerical Analysis. Cambridge University Press, Cambridge, UK (2003)

[19] van Winckel, G.: Numerical methods for differential equations. Karl-Franzens Universität Graz, Austria (2012). Lecture Notes

Chapter 11

Ordinary Differential Equations - Boundary Value Problems

11.1 Introduction

It is the aim of this chapter to introduce some of the basics methods developed to solve boundary value problems. Since a treatment of all available concepts is far too extensive, we will concentrate on two approaches, namely the finite difference approach and shooting methods [4, 7, 12, 1, 9]. Furthermore, we will strictly focus on *linear boundary value problems* defined on a finite interval $[a, b] \subset \mathbb{R}$. A boundary value problem is referred to as linear if both the differential equation and the boundary conditions are linear. Such a problem of order n is of the form

$$\begin{cases} L[y] = f(x), & x \in [a, b] \; , \\ U_\nu[y] = \lambda_\nu, & \nu = 1, \dots, n \; . \end{cases} \tag{11.1}$$

Here, $L[y]$ is a linear operator

$$L[y] = \sum_{k=0}^{n} a_k(x) y^{(k)}(x) \; , \tag{11.2}$$

where $y^{(k)}(x)$ denotes the k-th spatial derivative of $y(x)$, i.e. $y^{(k)} \equiv \mathrm{d}^k y(x)/\mathrm{d}x^k$ and $f(x)$ as well as the $a_k(x)$ are given functions which we assume to be continuous. Accordingly, linear boundary conditions $U_\nu[y]$ can be formulated as

$$U_\nu[y] = \sum_{k=0}^{n-1} \left[\alpha_{\nu k} y^{(k)}(a) + \beta_{\nu k} y^{(k)}(b) \right] = \lambda_\nu \; , \tag{11.3}$$

where the $\alpha_{\nu k}$, $\beta_{\nu k}$ and λ_ν are given constants. The question in which cases a solution to the boundary value problem (11.1) exists and whether or not this solution will be unique [8], will not be discussed here.

Let us introduce some further conventions: The differential equation in the first line of Eq. (11.1) is referred to as *homogeneous* if the function $f(x) = 0$ for all $x \in [a, b]$. In analogy, the boundary conditions are referred to as *homogeneous* if the constants $\lambda_\nu = 0$ for all $\nu = 1, \dots, n$. Finally, the boundary value problem (11.1) is referred to as *homogeneous* if the differential equation is homogeneous and the boundary conditions are homogeneous as well. In all other cases it is referred to as *inhomogeneous*. Moreover, the boundary conditions are said to be *decoupled* if the function values at the two different boundaries do not mix.

One of the most important types of boundary value problems in physics are linear second order boundary value problems with decoupled boundary conditions. They are of the form:

$$\begin{aligned} a_2(x)y''(x) + a_1(x)y'(x) + a_0(x)y(x) &= f(x)\,, \quad x \in [a,b]\,, && (11.4a)\\ \alpha_0 y(a) + \alpha_1 y'(a) &= \lambda_1, \quad |\alpha_0| + |\alpha_1| \neq 0\,, && (11.4b)\\ \beta_0 y(b) + \beta_1 y'(b) &= \lambda_2, \quad |\beta_0| + |\beta_1| \neq 0\,. && (11.4c) \end{aligned}$$

This chapter focuses mainly on problems of this kind.

In particular, for second order differential equations, boundary conditions of the form

$$y(a) = \alpha\,, \qquad y(b) = \beta\,, \tag{11.5}$$

are referred to as *boundary conditions of the first kind* or DIRICHLET *boundary conditions*. On the other hand, boundary conditions of the form

$$y'(a) = \alpha\,, \qquad y'(b) = \beta\,, \tag{11.6}$$

are referred to as *boundary conditions of the second kind* or NEUMANN *boundary conditions* and boundary conditions of the form (11.4) are referred to as *boundary conditions of the third kind* or STURM *boundary conditions*.

We note, that the particular case of decoupled boundary conditions does not include problems like

$$y(a) = y(b) \neq 0\,. \tag{11.7}$$

In the following section the method of finite differences will be applied to solve boundary value problems of the form (11.4). On the other hand, shooting methods, in particular the method developed by NUMEROV (see, for instance, E. HAIRER *et al.* [6] and references therein), will be the topic of the third section.

A common alternative in the case of constant coefficients is to solve the differential equation with the help of FOURIER transform techniques. A brief introduction to the numerical implementation of the FOURIER transform is given in Sec. 3.3 .

11.2 Finite Difference Approach

For illustrative purposes, we regard a boundary value problem of the form (11.4). The extension to more complex problems might be tedious but follows the same line of arguments. We discretize the interval $[a, b]$ according to the recipe introduced in Chap. 7: the positions x_k are given by $x_k = a + (k-1)h$, where the grid-spacing h is determined via the maximum number of grid-points N as $h = (b-a)/(N-1)$. Hence, we have $x_1 = a$ and $x_N = b$. Furthermore, we use the notation $y_k \equiv y(x_k)$ for all $k = 1, \ldots, N$. It will be used for all functions which appear in Eqs. (11.4).

Let us now employ the central difference derivative (7.10c) in order to approximate

$$y''_k \equiv y''(x_k) \approx \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}\,, \tag{11.8}$$

for $k = 2, \dots, N-1$ and

$$y'_k \equiv y'(x_k) \approx \frac{y_{k+1} - y_{k-1}}{2h} \; . \tag{11.9}$$

The boundary points x_1 and x_N will be treated in a separate step. In order to abbreviate the notation we will rewrite the differential equation (11.4) without the indices as

$$a(x)y''(x) + b(x)y'(x) + c(x)y(x) = f(x) \; . \tag{11.10}$$

Eqs. (11.8) and (11.9) are then applied and we arrive at the difference equation

$$a_k \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} + b_k \frac{y_{k+1} - y_{k-1}}{2h} + c_k y_k = f_k \; , \tag{11.11}$$

where $k = 2, \dots, N-1$. Sorting the y_k yields

$$\left(\frac{a_k}{h^2} - \frac{b_k}{2h}\right) y_{k-1} + \left(c_k - \frac{2a_k}{h^2}\right) y_k + \left(\frac{a_k}{h^2} + \frac{b_k}{2h}\right) y_{k+1} = f_k \; , \tag{11.12}$$

and this equation is only valid for $k = 2, \dots, N-1$ because we defined N grid-points within the interval $[a, b]$.

A final step is necessary in which the boundary conditions are incorporated. This will then enable us to reduce the whole problem to a system of linear equations. Decoupled boundary conditions of a second order differential equation for the left-hand boundary (11.4b) are of the general form:

$$\alpha_0 y(a) + \alpha_1 y'(a) = \lambda_1, \quad |\alpha_0|, |\alpha_1| \neq 0 \; . \tag{11.13}$$

In analogy, we find for the right-hand boundary (11.4c):

$$\beta_0 y(b) + \beta_1 y'(b) = \lambda_2, \quad |\beta_0|, |\beta_1| \neq 0 \; . \tag{11.14}$$

We discretize $y'(a)$ as

$$y'_1 \equiv y'(a) \approx \frac{y_2 - y_0}{2h} \; , \tag{11.15}$$

and set $y_1 = y(a)$. Note that the function value y_0 in Eq. (11.15) is unknown since the virtual point $x_0 = a - h$ is not within our interval $[a, b]$. Nevertheless, we use Eq. (11.15) in Eq. (11.13) and obtain:

$$\alpha_0 y_1 + \alpha_1 \frac{y_2 - y_0}{2h} = \lambda_1 \; . \tag{11.16}$$

We solve now Eq. (11.16) for y_0 under the premise that $\alpha_1 \neq 0$,

$$y_0 = y_2 - \frac{2h}{\alpha_1}(\lambda_1 - \alpha_0 y_1) \; , \tag{11.17}$$

rewrite Eq. (11.12) for $k = 1$,

$$\left(\frac{a_1}{h^2} - \frac{b_1}{2h}\right) y_0 + \left(c_1 - \frac{2a_1}{h^2}\right) y_1 + \left(\frac{a_1}{h^2} + \frac{b_1}{2h}\right) y_2 = f_1 \; , \tag{11.18}$$

and insert (11.17) into (11.18):

$$\left[c_1 - \frac{2a_1}{h^2} + \frac{\alpha_0}{\alpha_1}\left(\frac{2a_1}{h} - b_1\right)\right] y_1 + \frac{2a_1}{h^2} y_2 = f_1 - \frac{\lambda_1}{\alpha_1}\left(b_1 - \frac{2a_1}{h}\right) \; . \tag{11.19}$$

On the other hand, in the specific case of $\alpha_1 = 0$ we immediately obtain from Eq. (11.16):

$$y_1 = \frac{\lambda_1}{\alpha_0} \; . \tag{11.20}$$

The same strategy can be applied to incorporate the right-hand side boundary condition, Eq. (11.14): We discretize Eq. (11.14) by introducing the function value y_{N+1} at the virtual grid-point $x_{N+1} = N + h$ outside the interval $[a, b]$ via:

$$\beta_0 y_N + \beta_1 \frac{y_{N+1} - y_{N-1}}{2h} = \lambda_2 \; . \tag{11.21}$$

This equation is solved for y_{N+1} under the premise that $\beta_1 \neq 0$

$$y_{N+1} = y_{N-1} + \frac{2h}{\beta_1} \left(\lambda_2 - \beta_0 y_N\right) \; , \tag{11.22}$$

and insert this into Eq. (11.12) for $k = N$. This results in:

$$\frac{2a_N}{h^2} y_{N-1} + \left[c_N - \frac{2a_N}{h^2} - \frac{\beta_0}{\beta_1}\left(b_N + \frac{2a_N}{h}\right)\right] y_N = f_N - \frac{\lambda_2}{\beta_1}\left(b_N + \frac{2a_N}{h}\right) \; . \tag{11.23}$$

In the specific case $\beta_1 = 0$, the value y_N is fixed at the boundary and one obtains from Eq. (11.14):

$$y_N = \frac{\lambda_2}{\beta_0} \; . \tag{11.24}$$

All these manipulations reduced the boundary value problem to a system of inhomogeneous linear equations, namely Eqs. (11.12), (11.19), and (11.23). It can be written as

$$A\mathbf{y} = \mathbf{F} \; , \tag{11.25}$$

where we introduced the vector $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$, the vector $\mathbf{F}$

$$\mathbf{F} = \begin{pmatrix} f_1 - \frac{\lambda_1}{\alpha_1}\left(b_1 - \frac{2a_1}{h}\right) \\ f_2 \\ f_3 \\ \vdots \\ f_{N-1} \\ f_N - \frac{\lambda_2}{\beta_1}\left(b_N + \frac{2a_N}{h}\right) \end{pmatrix} \; , \tag{11.26}$$

and the tridiagonal matrix A:

$$A = \begin{pmatrix} B_1 & C_1 & 0 & & \cdots & 0 \\ A_2 & B_2 & C_2 & & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ & & & A_{N-1} & B_{N-1} & C_{N-1} \\ 0 & \cdots & & 0 & A_N & B_N \end{pmatrix} \; . \tag{11.27}$$

Here we defined

$$A_k = \begin{cases} \left(\frac{a_k}{h^2} - \frac{b_k}{2h}\right) & k = 2, \dots, N-1 \; , \\ \frac{2a_N}{h^2} & k = N \; , \end{cases} \tag{11.28}$$

$$B_k = \begin{cases} \left[c_1 - \frac{2a_1}{h^2} + \frac{\alpha_0}{\alpha_1}\left(\frac{2a_1}{h} - b_1\right)\right] & k = 1 \, , \\ \left(c_k - \frac{2a_k}{h^2}\right) & k = 2, \dots N-1 \, , \\ \left[c_N - \frac{2a_N}{h^2} - \frac{\beta_0}{\beta_1}\left(b_N + \frac{2a_N}{h}\right)\right] & k = N \, , \end{cases} \tag{11.29}$$

and, finally,

$$C_k = \begin{cases} \frac{2a_1}{h^2} & k = 1 \, , \\ \left(\frac{a_k}{h^2} + \frac{b_k}{2h}\right) & k = 2, \dots, N-1 \, . \end{cases} \tag{11.30}$$

The remaining task is now to solve this linear system of equations (11.25). (A brief introduction to the numerical treatment of linear systems of equations can be found in Chap. 2.) Very effective methods exist for cases where the matrix A is tridiagonal [10] as it is the case here. Although we discussed the method of finite differences for the particular case of a second order differential equation with decoupled boundary conditions, the same strategy can be employed to derive similar methods for higher order boundary value problems. However, these methods will, in general, be more complex. Furthermore, we note that in cases where $\alpha_1 = \beta_1 = 0$ the function values at the boundaries y_1 and y_N are fixed and the corresponding system of linear equations reduces to $(N-2)$ equations.

Let us briefly investigate the differential equation which corresponds to the problem (11.4) together with periodic boundary conditions of the form (11.7). In this case we have to consider that

$$y_1 = y_N \, , \tag{11.31}$$

and, for a solution to exist, we have necessarily

$$a_1 = a_N, \quad b_1 = b_N, \quad \text{and} \quad c_1 = c_N \, . \tag{11.32}$$

The finite difference approximations (11.8) and (11.9) are again applied to derive the equations (11.12) for $k = 2, \dots, N-1$. For $k = 2$ Eq. (11.12) becomes

$$\left(\frac{a_2}{h^2} - \frac{b_2}{2h}\right) y_1 + \left(c_2 - \frac{2a_2}{h^2}\right) y_2 + \left(\frac{a_2}{h^2} + \frac{b_2}{2h}\right) y_3 = f_2 \, , \tag{11.33}$$

and we have for $k = N-1$

$$\left(\frac{a_{N-1}}{h^2} - \frac{b_{N-1}}{2h}\right) y_{N-2} + \left(c_{N-1} - \frac{2a_{N-1}}{h^2}\right) y_{N-1} + \left(\frac{a_{N-1}}{h^2} + \frac{b_{N-1}}{2h}\right) y_N = f_{N-1} \, . \tag{11.34}$$

Since $y_1 = y_N$ this can be rewritten as

$$\left(\frac{a_{N-1}}{h^2} - \frac{b_{N-1}}{2h}\right) y_{N-2} + \left(c_{N-1} - \frac{2a_{N-1}}{h^2}\right) y_{N-1} + \left(\frac{a_{N-1}}{h^2} + \frac{b_{N-1}}{2h}\right) y_1 = f_{N-1} \, . \tag{11.35}$$

Finally, Eq. (11.12) results for $k = 1$ in

$$\left(\frac{a_1}{h^2} - \frac{b_1}{2h}\right) y_{N-1} + \left(c_1 - \frac{2a_1}{h^2}\right) y_1 + \left(\frac{a_1}{h^2} + \frac{b_1}{2h}\right) y_2 = f_1 \, , \tag{11.36}$$

where we identified $y_0 = y(x_1 - h) \equiv y(x_N - h) = y_{N-1}$. All this results in a closed system of $N - 1$ equations, which is of the form (11.25)

$$A\mathbf{y} = \mathbf{F} \,, \tag{11.37}$$

where $\mathbf{y} = (y_1, y_2, \dots, y_{N-1})^T$,

$$\mathbf{F} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{pmatrix} , \tag{11.38}$$

and the $(N - 1) \times (N - 1)$ matrix A is given by

$$A = \begin{pmatrix} B_1 & C_1 & 0 & \cdots & 0 & A_1 \\ A_2 & B_2 & C_2 & & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & & & A_{N-2} & B_{N-2} & C_{N-2} \\ C_{N-1} & 0 & \cdots & 0 & A_{N-1} & B_{N-1} \end{pmatrix} . \tag{11.39}$$

Here, we defined

$$A_k = \left(\frac{a_k}{h^2} - \frac{b_k}{2h} \right) , \qquad k = 1, \dots, N-1 \,, \tag{11.40}$$

$$B_k = \left(c_k - \frac{2a_k}{h^2} \right) , \qquad k = 1, \dots, N-1 \,, \tag{11.41}$$

and

$$C_k = \left(\frac{a_k}{h^2} + \frac{b_k}{2h} \right) , \qquad k = 1, \dots, N-1 \,. \tag{11.42}$$

In contrast to the matrix (11.27) the matrix (11.39) is not tridiagonal since the matrix elements $(A)_{1,N-1}$ and $(A)_{N-1,1}$ are non-zero. Nevertheless, it was possible to reduce the boundary value problem to a system of linear equations which can be solved iteratively.

11.3 Shooting Methods

For illustrative purposes, we restrict here the discussion to a second order boundary value problem with decoupled boundary conditions of the form (11.4). The essential idea of shooting methods is to treat the boundary value problem as an initial value problem. The resulting equations can then be solved with the help of methods discussed in Chap. 10. Of course, such an approach is ill-defined because no initial conditions but only boundary conditions are given. The trick is, that one modifies the initial conditions iteratively in such a way that in the end the boundary conditions are fulfilled. Let us put this train of thoughts into a mathematical form: We rewrite the second order differential equation (11.4a) as

$$y'' = f(y, y', x) \,, \tag{11.43}$$

which can be reduced to a set of first order differential equations as was demonstrated in Chap. 10. We note that Eq. (11.43) is not yet well posed since the initial conditions have not been defined. The boundary condition on the left-hand side reads:

$$\alpha_0 y(a) + \alpha_1 y'(a) = \lambda_1 \ . \tag{11.44}$$

We now *assume* that $y'(a) = z$, where z is some number. This gives the well posed initial value problem

$$\begin{cases} y'' = f(y, y', x) \ , \\ y(a) = \dfrac{\lambda_1}{\alpha_0} - \dfrac{\alpha_1}{\alpha_0} z \ , \\ y'(a) = z \ , \end{cases} \tag{11.45}$$

under the assumption that $\alpha_0 \neq 0$. The solution of this problem will be written as $y(x; z)$ in order to indicate its dependence on the particular choice $y'(a) = z$. We remember, that the boundary condition at the right-hand boundary is defined as:

$$\beta_0 y(b) + \beta_1 y'(b) = \lambda_2 \ . \tag{11.46}$$

Let us introduce the function:

$$F(z) = \beta_0 y(b; z) + \beta_1 y'(b; z) - \lambda_2 \ . \tag{11.47}$$

We observe that the solution of the equation

$$F(z) = 0 \ , \tag{11.48}$$

gives the desired solution to the boundary value problem (11.4), because in this case the second boundary condition (11.46) is fulfilled. In practice, one tries several values of z until relation (11.48) is fulfilled. However, from a numerical point of view this method is very inefficient since usually several initial value problems have to be solved until the correct value of z is found. Nevertheless, in some cases shooting methods proved to be very useful [6].

For instance, shooting methods are particularly effective if a solution to an eigenvalue problem of the form

$$a(x)y''(x) + b(x)y'(x) + c(x)y(x) = \lambda y(x) \ , \tag{11.49a}$$

in combination with homogeneous boundary conditions,

$$\alpha_0 y(a) + \alpha_1 y'(a) = 0 \ , \tag{11.49b}$$

and

$$\beta_0 y(b) + \beta_1 y'(b) = 0 \ , \tag{11.49c}$$

is to be found. We note that Eq. (11.49a) has the trivial solution $y(x) \equiv 0$ for all values of λ. However a non-trivial solution will only exist for particular values of λ. These particular values will be indexed by λ_n and are referred to as *eigenvalues* of Eq. (11.49a) [5]. The corresponding functions $y_n(x)$ are referred to as *eigenfunctions*. We note that the differential equation (11.49a) in combination with the boundary conditions (11.49b) and (11.49c) define a homogeneous boundary value problem. Such a problem is commonly referred to as an *eigenvalue problem* [5]. Furthermore, we note the following property of homogeneous boundary value problems: Suppose that $y(x)$ is a solution of the boundary value problem (11.49), then $\tilde{y}(x) = \gamma y(x)$, with $\gamma = \text{const}$ will also be a solution of (11.49). Hence, the solution of a homogeneous boundary value problem is not unique but invariant under multiplication by a constant γ. Typically, the multiplicative factor γ is fixed by some additional condition, such as a normalization condition of the form

$$\int_a^b \mathrm{d}x |y(x)|^2 = 1 \; . \tag{11.50}$$

We now employ this property and *choose* $y(a) = 1$. Inserting this choice into (11.49b) yields

$$y'(a) = -\frac{\alpha_0}{\alpha_1} \; . \tag{11.51}$$

Note that for $\alpha_0 = 0$ or $\alpha_1 = 0$, we are restricted to the choices $y'(a) = 0$ and $y(a)$ is arbitrary or $y(a) = 0$ and $y'(a)$ is arbitrary, respectively. If we assume that $a(x) \neq 0$ for all $x \in [a, b]$, we can solve the initial value problem

$$\begin{cases} y''(x) = -\dfrac{b(x)}{a(x)} y'(x) - \dfrac{c(x) - \lambda}{a(x)} y(x) \; , \\ y(a) = 1 \; , \\ y'(a) = -\dfrac{\alpha_0}{\alpha_1} \; . \end{cases} \tag{11.52}$$

The solutions are denoted by $y(x; \lambda)$ in order to emphasize that they will highly depend on the choice of the parameter λ. The strategy is to solve the initial value problem (11.52) for several values of λ and whenever one finds that

$$F(\lambda_n) = \beta_0 y(b; \lambda) + \beta_1 y'(b; \lambda) - \lambda_n = 0 \; , \tag{11.53}$$

is satisfied, an eigenvalue λ_n with the corresponding eigenfunction $y_n(x) = y(x; \lambda_n)$ of the eigenvalue problem (11.49) has been found.

However, this strategy is also very time consuming. The most common application of the shooting method is its combination with a very fast and accurate solution of initial value problems. This method is known as the NUMEROV method [6]. It is applicable whenever one is confronted with a differential equation of the form

$$y''(x) + k(x) y(x) = 0 \; , \tag{11.54}$$

in combination with homogeneous boundary conditions. Here $k(x)$ is some function. If we are particularly interested in eigenvalue problems then $k(x)$ has the form $k(x) = q(x) - \lambda$, where $q(x)$ is some function and λ is the eigenvalue [see the discussion after equation (11.49)]. For instance, consider the one-dimensional stationary SCHRÖDINGER equation [2, 3, 11],

$$\psi''(x) + \frac{2m}{\hbar^2} \left[E - V(x) \right] \psi(x) = 0 \; , \tag{11.55}$$

where $\psi(x)$ is the wave-function, m is the mass, $\hbar$ denotes the reduced PLANCK constant, E is the energy, and $V(x)$ is some potential. In this case we identify

$$k(x) = \frac{2m}{\hbar^2} \left[E - V(x) \right] \; . \tag{11.56}$$

We note that Eq. (11.55) together with its boundary conditions defines an eigenvalue problem with eigenvalues E_n, the possible energies of the system. We remember from Chap. 7, Eq. (7.34), that

$$y_j'' = \frac{y_{j+1} - 2y_j + y_{j-1}}{h^2} - \frac{h^2}{12} y_j^{(4)} - \ldots = -k_j y_j \ . \tag{11.57}$$

In the last step we made use of Eq. (11.54) and introduced $k_j \equiv k(x_j)$. Furthermore, we write the fourth derivative of $y(x)$ at point $x = x_j$ as

$$y_j^{(4)} \approx \frac{y_{j+1}'' - 2y_j'' + y_{j-1}''}{h^2} = \frac{-k_{j+1}y_{j+1} + 2k_j y_j - k_{j-1}y_{j-1}}{h^2} \ , \tag{11.58}$$

where we employed equation (11.54). Truncating (11.57) after the fourth order derivative $y_j^{(4)}$, inserting relation (11.58), and solving for y_{j+1} yields

$$y_{j+1} = \frac{2\left(1 - \frac{5h^2}{12}k_j\right) y_j - \left(1 + \frac{h^2}{12}k_{j-1}\right) y_{j-1}}{1 + \frac{h^2}{12}k_{j+1}} \ . \tag{11.59}$$

This gives a very fast algorithm to solve the differential equation (11.54) with some initial values of the form (11.52). The remaining strategy is the same as discussed above, i.e. one *screens* the parameter λ in order to find the eigenvalues λ_n and eigenfunctions $y_n(x)$. In case of the SCHRÖDINGER equation, one can screen the energy E in order to obtain the energy eigenvalues E_n which satisfy a condition of the form (11.53).

11.3.1 A Simple Example: The Particle in a Box

We concentrate on the one-dimensional SCHRÖDINGER equation (11.55) and discuss a simple problem which will then be solved numerically using NUMEROV's shooting method. This model describes a particle whithin a one-dimensional box of length L. We rewrite this equation, with $x \in \mathbb{R}$, as

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\psi_n(x) + V(x)\psi_n(x) = E_n \psi_n(x) \ , \tag{11.60}$$

and specify

$$V(x) = \begin{cases} 0 & 0 \le x \le L, \\ \infty & \text{elsewhere,} \end{cases} \tag{11.61}$$

together with the boundary conditions

$$\psi_n(0) = \psi_n(L) = 0 \ , \tag{11.62}$$

and the normalization condition

$$\int \mathrm{d}x |\psi_n(x)|^2 = \int_0^L \mathrm{d}x |\psi_n(x)|^2 = 1 \ . \tag{11.63}$$

Here E_n denotes the n^{th} eigenvalue of Eq. (11.60) and the function $\psi_n(x)$ is the corresponding eigenfunction. We also note that the boundary conditions are dictated by the particular form of the potential (11.61) which requires that $\psi_n(x) = 0$ for

$x \notin [0, L]$ because the particle is not allowed to exist outside the box. This problem is commonly referred to as the *particle in a one-dimensional box* problem.

Let us introduce dimensionless variables in order to simplify the numerics of Eq. (11.60). We define new variables

$$s = \frac{x}{L}, \qquad \varepsilon_n = \frac{E_n}{\overline{E}} , \tag{11.64}$$

where L is the length scale and $\overline{E}$ is the energy scale. The energy scale $\overline{E}$ is fully determined by the relation

$$\overline{E} = \frac{\hbar^2}{mL^2} . \tag{11.65}$$

We note that $s \in [0, 1]$, hence the rescaled wave-function is given by

$$\varphi_n(s) = \sqrt{L}\psi_n(x) , \tag{11.66}$$

which satisfies the normalization condition

$$\int_0^L \mathrm{d}x|\psi_n(x)|^2 = \int_0^1 \mathrm{d}s|\varphi_n(s)|^2 = 1 . \tag{11.67}$$

The rescaled Schrödinger equation can be obtained by multiplying Eq. (11.60) with $1/\overline{E}$:

$$\begin{aligned} -\frac{\hbar^2}{2m\overline{E}}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\psi_n(x) + \frac{V(x)}{\overline{E}}\psi_n(x) &= -\frac{L^2}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\psi_n(x) + v(s)\psi_n(x) \\ &= -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}s^2}\psi_n(x) + v(s)\psi_n(x) \\ &= \frac{E_n}{\overline{E}}\psi_n(x) \\ &= \varepsilon_n\psi_n(x) . \end{aligned} \tag{11.68}$$

Here we introduced the rescaled potential $v(s)$

$$v(s) = \begin{cases} 0 & 0 \le s \le 1, \\ \infty & \text{elsewhere.} \end{cases} \tag{11.69}$$

Hence, the rescaled wave-function (11.66) is a solution of the differential equation:

$$-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}s^2}\varphi_n(s) + v(s)\varphi_n(s) = \varepsilon_n\varphi_n(s) . \tag{11.70}$$

The form (11.69) of the potential implies that $\varphi_n(s) = 0$ for all $s \notin [0, 1]$ and the complete boundary value problem is defined as:

$$\begin{cases} -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}s^2}\varphi_n(s) = \varepsilon_n\varphi_n(s) , \quad s \in [0, 1] , \\ \varphi_n(0) = 0 , \\ \varphi_n(1) = 0 . \end{cases} \tag{11.71}$$

It is an easy task to solve this boundary value problem analytically. For $s \in [0, 1]$ we choose the ansatz

$$\varphi_n(s) = A_n \sin(k_n s) + B_n \cos(k_n s) , \tag{11.72}$$

where A_n and B_n are some constants and k_n is given by

$$k_n = \sqrt{2\varepsilon_n} \; . \tag{11.73}$$

From the boundary conditions we obtain

$$\varphi_n(0) = B_n = 0 \; , \tag{11.74}$$

and

$$\varphi_n(1) = A_n \sin(k_n) = 0 \; . \tag{11.75}$$

Thus,

$$k_n = n\pi \; , \tag{11.76}$$

and the eigenenergies ε_n are quantized:

$$\varepsilon_n = \frac{n^2\pi^2}{2} \; . \tag{11.77}$$

The corresponding eigenfunctions $\varphi_n(s)$ are then given by:

$$\varphi_n(s) = \begin{cases} A_n \sin(n\pi s) & s \in [0,1] \; , \\ 0 & \text{elsewhere.} \end{cases} \tag{11.78}$$

The constants A_n are determined from the normalization condition (11.67)[1]

$$\begin{aligned} \int_0^1 \mathrm{d}s |\varphi_n(s)|^2 &= A_n^2 \int_0^1 \mathrm{d}s \sin^2(n\pi s) \\ &= \frac{A_n^2}{2} \\ &\overset{!}{=} 1 \; , \end{aligned} \tag{11.81}$$

and:

$$A_n = \sqrt{2} \; . \tag{11.82}$$

Finally, we apply the relations (11.64), (11.65), and (11.66) and obtain the eigenfuctions

$$\psi_n(x) = \frac{1}{\sqrt{L}} \varphi_n\left(\frac{x}{L}\right) = \begin{cases} \sqrt{\frac{2}{L}} \sin\left(\frac{n\pi x}{L}\right) & x \in [0, L] \; , \\ 0 & \text{elsewhere,} \end{cases} \tag{11.83}$$

and corresponding eigenvalues:

$$E_n = \varepsilon_n \overline{E} = \frac{\hbar^2\pi^2 n^2}{2mL^2} \; . \tag{11.84}$$

[1]Here we make use of

$$\begin{aligned} \int \mathrm{d}u \sin^2(u) &= -\cos(u)\sin(u) + \int \mathrm{d}u \cos^2(u) \\ &= -\cos(u)\sin(u) + \int \mathrm{d}u \left[1 - \sin^2(u)\right] \; , \end{aligned} \tag{11.79}$$

and, therefore

$$\int \mathrm{d}u \sin^2(u) = \frac{1}{2}\left[u - \cos(u)\sin(u)\right] \; . \tag{11.80}$$

11.3.2 Numerical Solution

The following discussion is based on the scaled SCHRÖDINGER equation (11.70) with the potential $v(s)$ defined in Eq. (11.69). This results in the boundary value problem:

$$\begin{cases} -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}s^2}\varphi_n(s) + v(s)\varphi_n(s) = \varepsilon_n\varphi_n(s), \quad s \in [0,1], \quad n \in \mathbb{N}\,, \\ \varphi_n(0) = 0\,, \\ \varphi_n(1) = 0\,. \end{cases} \tag{11.85}$$

As our numerical treatment will be based on shooting methods, the second order differential equation (11.85) will be transformed into a form which corresponds to Eq. (11.54), namely:

$$\varphi_n''(s) + 2\left[\varepsilon_n - v(s)\right]\varphi_n(s) = 0\,. \tag{11.86}$$

The interval $[0,1]$ is discretized using $N+1$ grid-points $s_\ell = \ell/N$, $\ell = 0,1,2,\ldots,N$ ($h = 1/N$) and we denote with $\varphi_n(s_\ell)$ and $v(s_\ell) \equiv v_\ell$ the values of $\varphi_n(s)$ and $v(s)$ at those grid-points. Thus, Eq. (11.59) can immediately be applied and we get

$$\varphi_n(s_{\ell+1}) = \frac{2\left[1 - \frac{5}{6N^2}(\varepsilon_n - v_\ell)\right]\varphi_n(s_\ell) - \left[1 + \frac{1}{6N^2}(\varepsilon_n - v_{\ell-1})\right]\varphi_n(s_{\ell-1})}{1 + \frac{1}{6N^2}(\varepsilon_n - v_{\ell+1})}\,, \tag{11.87}$$

and denote the solutions of Eq. (11.86) by $\varphi_n(s_\ell;\varepsilon)$ to emphasize their dependence on the eigenvalue ε.

Furthermore, we make use of the initial conditions $\varphi_n(s_0) = 0$ and $\varphi_n'(s_0) = 1$ which is always possible, since (11.85) is a homogeneous boundary value problem. This gives

$$\varphi_n'(s_0) \approx \frac{\varphi_n(s_1) - \varphi_n(s_{-1})}{2h} = 1 \quad \Rightarrow \quad \varphi_n(s_1) = \frac{2}{N}\,. \tag{11.88}$$

The normalization of the wave-function (11.67) is then approximated with the help of the forward rectangular rule (8.9):

$$\int_0^1 \mathrm{d}s|\varphi_n(s)|^2 \approx h\sum_{\ell=0}^{N}\left[\varphi_n(s_\ell)\right]^2 \overset{!}{=} 1\,. \tag{11.89}$$

The NUMEROV shooting algorithm is then defined by the following steps:

1. Choose two trial energies ε_a and ε_b and define the required accuracy η.
2. Calculate $\varphi(s_N;\varepsilon_a) \equiv \varphi_a$ and $\varphi(s_N;\varepsilon_b) \equiv \varphi_b$ using Eq. (11.87).
3. If $\varphi_a\varphi_b > 0$, choose new values for ε_a or ε_b and go to step 1.
4. If $\varphi_a\varphi_b < 0$, calculate $\varepsilon_c = (\varepsilon_a + \varepsilon_b)/2$ and determine $\varphi(s_N;\varepsilon_c) \equiv \varphi_c$ using Eq. (11.87).
5. If $\varphi_a\varphi_c < 0$, set $\varepsilon_b = \varepsilon_c$ and go to step 4.
6. If $\varphi_c\varphi_b < 0$, set $\varepsilon_a = \varepsilon_c$ and go to step 4.
7. Terminate the loop when $|\varepsilon_a - \varepsilon_b| < \eta$.

Table 11.1: Comparison between analytic and numerical eigenvalues ε_n for the particle in a box with $N = 100$.

n	ε_n-analytic	ε_n-numeric
1	4.934802	4.934802
2	19.739209	19.739208
3	44.413219	44.413205
4	78.956835	78.956753
5	123.370055	123.369742
6	177.652879	177.651943
7	241.805308	241.802947
8	315.827341	315.822077
9	399.718978	399.708300
10	493.480220	493.460113

These steps have been carried out for 100 grid-points, a potential $v(s)$ defined in Eq. (11.69), and a required accuracy of $\eta = 10^{-10}$. The first ten eigenenergies are given in Tab. 11.1 and are compared with analytic results of Eq. (11.77).

In addition, Fig. 11.1 presents the first five eigenvalues ε_n (right hand scale) as horizontal straight lines. Aligned with these eigenvalues we find on the right hand side of this figure the corresponding normalized eigenfunctions calculated using $N = 100$ grid-points. The agreement with the analytic result of Eq. (11.78) is excellent.

11.3.3 Final Remarks

We conclude this section with two important remarks on the Numerov method. We note from Eq. (11.59) that in order to compute y_3 one already needs the function values y_1 and y_2. Usually, one obtains these values from the boundary conditions in combination with some additional condition for the problem at hand. Such an additional condition might be, for instance, the normalization of the function $y(x)$, like Eq. (11.50). We also emphasized that one has to run the Numerov algorithm several times for different *trial values* of the parameter λ. In order to reduce the computational cost of the method it is in many cases advantageous to store the function values q_i, where $k_i = q_i - \lambda$, in an array which is then regarded as an input argument of the Numerov algorithm.

Bibliography

[1] Asher, U.M., Mattheij, R.M.M., Russell, R.D.: Numerical Solution of Boundary Value Problems for Ordinary Differential Eqautions. Classics in Applied Mathematics, No. 13. Cambridge University Press, Cambridge, UK (1995)

[2] Baym, G.: Lectures on Quantum Mechanics. Lecture Notes and Supplements in Physics. The Benjamin/Cummings Publ. Comp., Inc., London, Amsterdam (1969)

[3] Cohen-Tannoudji, C., Diu, B., Laloë, F.: Quantum Mechanics, vol. I. Wiley, New York (1977)

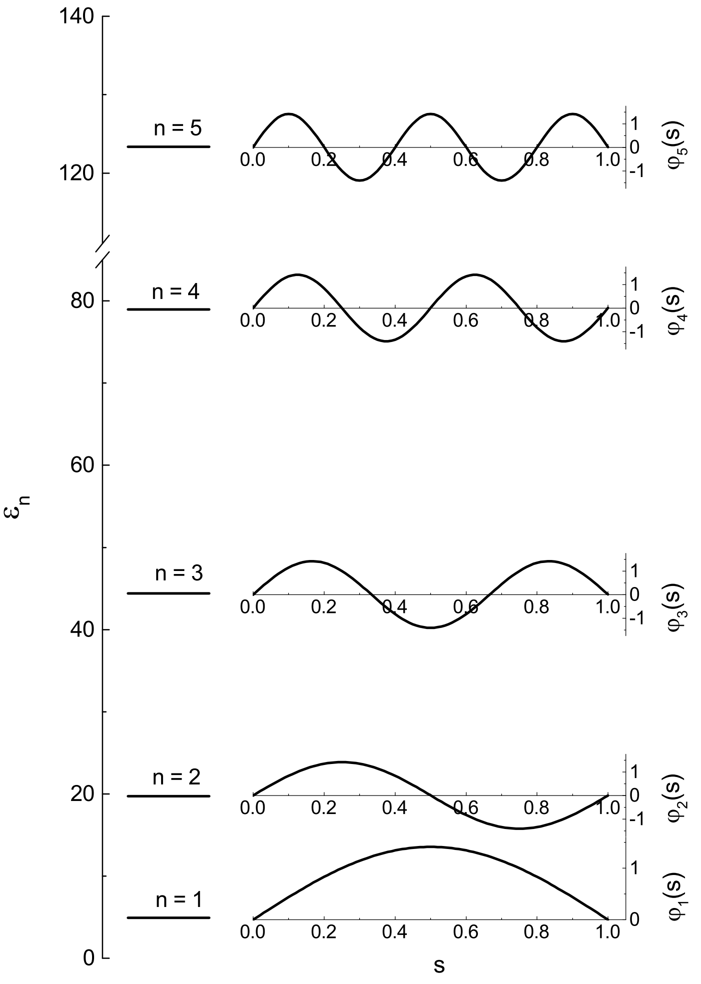

Figure 11.1: The first five numerically determined eigenvalues ε_n of Tab. 11.1 are presented as *horizontal lines (left hand scale)*. Aligned with these eigenvalues are the corresponding eigenfunctions $\varphi_n(s)$ vs s for $N = 100$ (*right hand scales*).

[4] Collatz, L.: The Numerical Treatment of Differential Equations. Springer, Berlin, Heidelberg (1960)

[5] Courant, R., Hilbert, D.: Methods of Mathematical Physics, vol. 1. Wiley, New York (1989)

[6] Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I, 2nd edn. Springer Series in Computational Mathematics, Vol. 8. Springer, Berlin, Heidelberg (1993)

[7] Lapidus, L., Pinder, G.F.: Numerical Solution of Partial Differential Equations. Wiley, New York (1982)

[8] Polyanin, A.D., Zaitsev, V.F.: Boundary Value Problems, 2nd edn. Chapman & Hall/CRC, Boca Raton, USA (2003)

[9] Powers, D.L.: Boundary Value Problems, 6th edn. Academic Press, San Diego (2009)

[10] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[11] Sakurai, J.J.: Modern Quantum Mechanics. Addison-Wesley, Menlo Park (1985)

[12] Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, 2nd edn. Springer, Berlin, Heidelberg (1993)

Chapter 12

The One-Dimensional Stationary Heat Equation

12.1 Introduction

This is the first of two chapters which illustrate the applicability of the methods introduced in Chap. 11. Within this chapter the finite difference approach is employed to solve the stationary heat equation. Let us motivate briefly this particular problem. We consider a rod of length L which is supposed to be kept at constant temperatures T_0 and T_N at its ends as illustrated in Fig. 12.1. The homogeneous

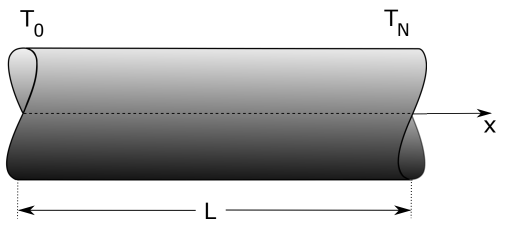

Figure 12.1: We consider a rod of length L. Its ends are kept at constant temperatures T_0 and T_N, respectively.

heat equation is a linear partial differential equation of the form

$$\frac{\partial}{\partial t} T(x,t) = \kappa \Delta T(x,t) \ . \tag{12.1}$$

Here $T(x,t)$ is the temperature as a function of space $x \in \mathbb{R}^3$ and time $t \in \mathbb{R}$, $\Delta = \nabla^2 = \partial_x^2 + \partial_y^2 + \partial_z^2$ is the LAPLACE operator, and $\kappa = \text{const}$ is the thermal diffusivity.

We remark, that Eq. (12.1) is a partial differential equation together with initial and boundary conditions. Moreover, we note in passing that the heat equation is equivalent to the diffusion equation [2]

$$\frac{\partial}{\partial t} \rho(x,t) = D \Delta \rho(x,t) \ , \tag{12.2}$$

with particle density $\rho(x,t)$ and the diffusion coefficient $D = \text{const}$. Here we restrict ourselves to a simplified situation in order to test the validity of the finite difference approach discussed in Sec. 11.2. (The problem of the one-dimensional heat equation was studied in all conceivable detail by J. R. CANNON [1].)

If we assume the cylindrical surface of the rod to be perfectly isolated, we can restrict the problem to one dimension. Furthermore, we assume that the steady-state has been reached, i.e. $\frac{\partial}{\partial t}T(x,t) = 0$. Hence, the remaining boundary value problem is of the form

$$\begin{cases} \frac{\mathrm{d}^2}{\mathrm{d}x^2}T(x) = 0, \qquad x \in [0, L] \ , \\ T(0) = T_0 \ , \\ T(L) = T_N \ . \end{cases} \tag{12.3}$$

The solution can easily be found analytically and one obtains

$$T(x) = T_0 + (T_N - T_0)\,\frac{x}{L} \ . \tag{12.4}$$

In the following section we will apply the approach of finite differences to the boundary value problem (12.3) as discussed in Sec. 11.2.

12.2 Finite Differences

We discretize the interval $[0, L]$ according Chap. 7 by the introduction of N grid-points $x_n = nh$, with $h = L/N$, $x_0 = 0$, and $x_N = L$. Furthermore, $T_n \equiv T(x_n)$ and, in particular, we refer to the boundary conditions (12.3) as T_0 and T_N, respectively.

On the basis of this discretization, we approximate Eq. (12.3) by

$$\frac{T_{n+1} - 2T_n + T_{n-1}}{h^2} = 0 \ , \tag{12.5}$$

or equivalently

$$T_{n+1} - 2T_n + T_{n-1} = 0 \ . \tag{12.6}$$

We can rewrite this as a matrix equation,

$$AT = F \ , \tag{12.7}$$

where the boundary conditions have already been included. In Eq. (12.7) the vector $T = (T_1, T_2, \ldots, T_{N-1})^T$. Furthermore, the tridiagonal matrix A is given by

$$A = \begin{pmatrix} -2 & 1 & 0 & & \ldots & 0 \\ 1 & -2 & 1 & 0 & \ldots & 0 \\ 0 & 1 & -2 & 1 & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \ldots & & & 1 & -2 \end{pmatrix} , \tag{12.8}$$

and the vector F by

$$F = \begin{pmatrix} -T_0 \\ 0 \\ \vdots \\ 0 \\ -T_N \end{pmatrix} . \tag{12.9}$$

It is an easy task to solve Eq. (12.7) analytically. It follows from Eq. (12.6) that

$$T_{n+1} = 2T_n - T_{n-1}, \qquad n = 1, \dots, N-1 \; . \tag{12.10}$$

We insert $n = 1, 2, 3$ in order to obtain

$$\begin{aligned}
T_2 &= 2T_1 - T_0 \; , && (12.11)\\
T_3 &= 2T_2 - T_1 \; , \\
&= 3T_1 - 2T_0 \; , && (12.12)\\
T_4 &= 2T_3 - T_2 \; , \\
&= 4T_1 - 3T_0 \; . && (12.13)
\end{aligned}$$

We recognize the pattern and conclude that T_n has the general form

$$T_n = nT_1 - (n-1)T_0 \; , \tag{12.14}$$

which we prove by complete induction:

$$\begin{aligned}
T_{n+1} &= 2T_n - T_{n-1} \\
&= 2(nT_1 - (n-1)T_0) - [(n-1)T_1 - (n-2)T_0] \\
&= (n+1)T_1 - nT_0 \; . && (12.15)
\end{aligned}$$

Hence, expression (12.14) is valid for all $n = 1, \dots, N$. However, since T_N is kept constant according to the boundary condition, we can determine T_1 from

$$T_N = NT_1 - NT_0 + T_0 \; , \tag{12.16}$$

which yields

$$T_1 = \frac{T_N - T_0}{N} + T_0 \; . \tag{12.17}$$

Inserting (12.17) into (12.14) gives

$$\begin{aligned}
T_n &= T_0 + (T_N - T_0)\frac{n}{N} \\
&= T_0 + (T_N - T_0)\frac{nh}{L} \; , && (12.18)
\end{aligned}$$

which is exactly the discretized version of Eq. (12.4). Hence the finite difference approach to the boundary value problem (12.3) is exact and independent of the grid-spacing h. This is not surprising since we proved already in Chap. 7 that finite difference derivatives are exact for linear functions.

12.3 A Second Scenario

We consider the inhomogeneous heat equation

$$\frac{\partial}{\partial t} T(x,t) = \kappa \Delta T(x,t) - \Gamma(x,t) \; . \tag{12.19}$$

Here $\Gamma(x,t) \equiv \Gamma(x)$ is some heat source or heat drain, which is assumed to be independent of time t. Again, we consider the one dimensional, stationary case, i.e.

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} T(x) = \frac{1}{\kappa}\Gamma(x) \; , \tag{12.20}$$

with the same boundary conditions as in Eq. (12.4). Furthermore, we assume $\Gamma(x)$ to be of the form

$$\Gamma(x) = \frac{\Theta}{\ell} \exp\left[-\frac{\left(x - \frac{L}{2}\right)^2}{\ell^2}\right] , \tag{12.21}$$

i.e. $\Gamma(x)$ has the form of a GAUSS peak which is centered at $x = L/2$ and has a width determined by the parameter ℓ and a maximum height given by the constant Θ. Such a situation might occur, for instance, when the rod is heated with some kind of a heat gun or cooled by a cold spot. (In cases where the diffusion equation (12.2) is used to describe the random motion of electrons in some device, one can imagine, that the density of electrons ρ is constant at the contacts at $x = 0$ and $x = L$. The source/drain term $\Gamma(x)$ then accounts for a constant generation or recombination rate of electrons, for instance, through incoming light or intrinsic traps, respectively [3].)

Furthermore, we note that in the limiting case $\ell \to 0$ we have

$$\lim_{\ell \to 0} \Gamma(x) = \Theta\delta\left(x - \frac{L}{2}\right) , \tag{12.22}$$

where $\delta(\cdot)$ is the DIRAC δ-distribution; in this case the spatial extension of the source/drain term $\Gamma(x)$ is infinitesimal.

We now employ the results of Sec. 11.2 and rewrite the system of equations in the familiar form[1]

$$AT = F , \tag{12.23}$$

where A has already been defined in Eq. (12.8), $T = (T_1, T_2, \dots, T_{N-1})^T$, and F is given by

$$F = \frac{h^2}{\kappa} \begin{pmatrix} \Gamma_1 - \frac{\kappa}{h^2} T_0 \\ \Gamma_2 \\ \vdots \\ \Gamma_{N-2} \\ \Gamma_{N-1} - \frac{\kappa}{h^2} T_N \end{pmatrix} . \tag{12.24}$$

Here we used the notation $\Gamma_n \equiv \Gamma(x_n)$.

The system is solved numerically quite easily using methods discussed by PRESS *et al.* [4] for the solution of sets of algebraic equations of the kind (12.24) with tridiagonal matrix A. We chose $L = 10$, $\kappa = 1$, $\Theta = -0.4$, $\ell = 1$, $T_0 = 0$ and $T_N = 2$. The resulting temperature profiles $T(x)$ (solid line) for different values of N can be found in Figs. 12.2, 12.3, and 12.4 as well as the respective form of the function $\Gamma(x)$ (dashed line). With increasing number of steps we see, as it was to be expected, a refinement of the temperature profile. Its maximum does not quite agree with the minimum of $\Gamma(x)$, it is shifted slightly towards the end of the rod because of the boundary conditions, i.e. $T_0 < T_N$.

Bibliography

[1] Cannon, J.R.: The One-Dimensional Heat Equation. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, UK (1985)

[1]We note that Eq. (12.20) can also be solved with the help of FOURIER transforms.

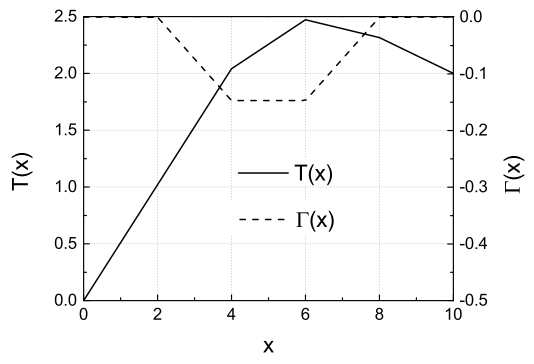

Figure 12.2: Temperature profile $T(x)$ (solid line, left hand scale) and the source function $\Gamma(x)$ (dashed line, right hand scale) for $N = 5$.

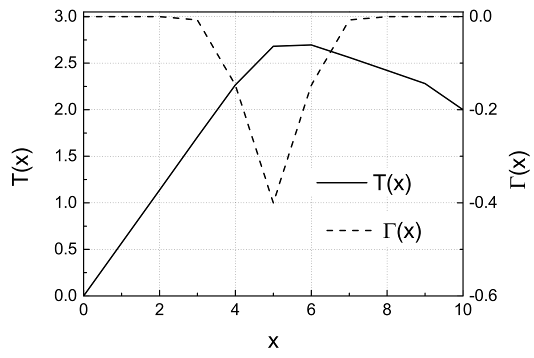

Figure 12.3: Temperature profile $T(x)$ (solid line, left hand scale) and the source function $\Gamma(x)$ (dashed line, right hand scale) for $N = 10$.

[2] Cussler, E.L.: Diffusion. Cambridge University Press, Cambridge, UK (2009)

[3] Glicksman, M.E.: Diffusion in Solids. Wiley, New York (1999)

[4] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

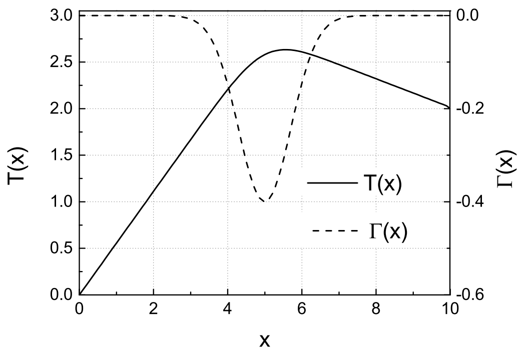

Figure 12.4: Temperature profile $T(x)$ (solid line, left hand scale) and the source function $\Gamma(x)$ (dashed line, right hand scale) for $N = 100$.

Chapter 13

Pseudo-Random Number Generators

13.1 Introduction

Stochastic methods in Computational Physics are based on the availability of random numbers and on the concepts of probability theory. (We refer readers not familiar with the basic concepts of probability to the literature [23, 20, 2, 17, 12].) The required random numbers are provided by numerical random number generators and, thus, we have to speak, more precisely, of pseudo-random numbers. Let us now motivate the problem and discuss some preliminary items.

A popular example of randomness in physical systems is certainly the outcome of a dice-throw or the drawing of lotto numbers. Even though the outcome of a dice-throw is completely determined by the initial conditions, it is effectively unpredictable because the initial conditions cannot be determined accurately enough. A probabilistic description, which assigns the *random variables* $1, 2, 3, 4, 5$, and 6 a probability of $1/6$, respectively, is much more convenient and promising. It has to be kept in mind, of course, that all predictions obtained on the basis of such an approach are also clearly probabilistic in their nature.

Another example is Brownian motion or diffusion which describes the random motion of dust particles on a fluid surface. It is in this case particularly obvious that a description with the help of a *stochastic differential equation*, such as the LANGEVIN equation [5], is completely sufficient and more to the point than a description based on the dynamics of a large number of interacting particles.

Stochastic methods are not confined to physics: They are applied very successfully in many other fields of expertise, like biology [7], economics [24, 16], medicine [15], etc. Finally, an interesting and purely mathematical application can be found in the evaluation of integrals as an alternative to the methods discussed in Chap. 8. This method is referred to as Monte-Carlo integration.

From the numerical point of view, there is one common denominator to all these applications: Random numbers are an essential tool and, consequently, so are random number generators. Thus, a closer inspection of randomness in general and the generation of random numbers or sequences in particular is required. We have to explain, what we understand by randomness and how it can be measured. Moreover, based on this discussion we have to formulate requirements to be imposed on the random number generators to deliver useful random numbers.

Although, we might have an intuitive picture of randomness it is hard to formulate without relying on mathematics. For instance, consider the sequence s_1 which consists of N elements:

$$s_1 = 0, 0, 0, 0, 0, \dots\ , \qquad N \text{ elements.} \tag{13.1}$$

Is it random? The question cannot be answered without further information. Suppose, the numbers in sequence s_1 were drawn from some set $\mathcal{S}$. If this set is of the form

$$\mathcal{S}_1 = \{0\}\ , \tag{13.2}$$

then the above sequence s_1 is certainly not random since there is only one possible outcome. However, suppose the numbers of the sequence s_1 were drawn from the set $\mathcal{S}_2$

$$\mathcal{S}_2 = \{0, 1\}\ , \tag{13.3}$$

with the *events* 0 and 1 together with assigned probabilities $P(0)$ and $P(1)$. These probabilities describe the probability that the outcome of a measurement on the set $\mathcal{S}_2$ yields either the event 0 or 1, respectively. For instance, in the case of tossing a coin the event 0 may correspond to *heads* while 1 stands for *tails*. (To register this result is, within this context, a *measurement*.) In this case the probabilities are given by $P(0) = P(1) = 1/2$ under the premise that the coin is perfectly ideal and has not been manipulated. Even, if we know that the coin has not been manipulated, sequence (13.1) is still a possible outcome, although it is rather improbable for a large number N of measurements (repeated tosses of the coin).

A literal definition of randomness within the context of a random sequence was given by G. J. Chaitin [3]:

> *[...] a series of numbers is random if the smallest algorithm capable of specifying it to a computer has about the same number of bits of information as the series itself.*

This definition seems to include the most important features of randomness which we are used to from our experience. Since, no universal trend is observable, reproducing the sequence requires the knowledge of every single constituent. Hence, one may employ the sloppy definition: *Randomness is the lack of a universal trend.*

But how can we test whether or not a given sequence really follows a certain distribution? Of course, one can simply exploit the statistical definition of probability: The probability of a certain outcome is *measured* by counting the particular results. This procedure seems to be quite promising, however, it has to be kept in mind that the statistical definition of probability is only valid in the limit $m \to \infty$, where m is the number of measurements. Hence, it is fundamentally impossible to determine whether or not a sequence is random because an infinite number of elements would have to be evaluated and analyzed. More promising appears to be the calculation of moments or correlations from the sequence and to compare such a result with known values for real random numbers. These statistical tests will be discussed in Sec. 13.3. If we consider the sequence (13.1) drawn from the set $\mathcal{S}_2$, [Eq. (13.3), uniform distribution assumed] we can deduce that it is a very improbable result for large N, although it is certainly a possible outcome. Methods based on this train of thoughts are known as *hypothesis testings* and we discuss the χ^2 test as a simple representative of such tests in Sec. 13.3.

We are now in a position to clarify what we understand by a random number: We regard a random sequence drawn from the set

$$\mathcal{S}_3 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\ . \tag{13.4}$$

If the random number is to be uniformly distributed, we assign probabilities $P(k) = 1/10,\ k = 0, 1, \dots, 9$ and if we would like to obtain a random number out of the interval

$$\Omega_1 = [0, 1)\ , \tag{13.5}$$

we may simply draw the sequence $s_2 = a_1, a_2, a_3, \dots$ from $\mathcal{S}_3$ and compose the random number r as

$$r = 0.a_1a_2a_3\dots\ . \tag{13.6}$$

Section 13.2 is dedicated to the discussion of different methods of how to generate so called *pseudo-random numbers*. A pseudo-random number is a number generated with the help of a deterministic algorithm, however, it shows a behavior as if it were random. This implicates that its statistical properties are close to that of true random numbers. In contrast to pseudo-random numbers, *real random numbers* are truly random. A real random number can be obtained from experiments. One could, for instance, simply toss a coin and register the resulting sequence of zeros and ones. A more sophisticated method is to exploit the radioactive decay of a nucleus, which is believed to be purely stochastic. There are also more exotic ideas, such as using higher digits of π which are assumed to behave as if they were random. However, all these methods have in common that they are far too slow for computational purposes. Moreover, an *experimental* approach is obviously not *reproducible* in the sense, that a random sequence cannot be reproduced on demand, but the reproducibility of a random number sequence is essential for many applications.

This leads us to the formulation of several criteria a random number generator will have to comply with. It should

- produce pseudo-random numbers whose statistical properties are as close as possible to those of real random numbers.
- have a long period: It should generate a non-repeating sequence of random numbers which is sufficiently long for computational purposes.
- be reproducible in the sense defined above, as well as restartable from an arbitrary break-point.
- be fast and parallelizable: It should not be the limiting component in simulations.

We restrict, whithin this chapter, our discussion to random numbers that are uniformly distributed over a finite set. Thus, we assign to all possible outcomes of a measurement the same probability. The generation of non-uniformly distributed random numbers from uniformly distributed random numbers is not a difficult task [6, 21].

13.2 Different Methods

We discuss here different types of pseudo-random number generators [13, 8, 22] which generate a pseudo-random number r which is uniformly distributed within the interval $[0, 1)$. Hence, its *probability density function* (pdf) is given by

$$p(r) = \begin{cases} 1 & r \in [0, 1)\ , \\ 0 & \text{elsewhere,} \end{cases} \tag{13.7}$$

and from this follows the *cumulative distribution function*:

$$P(r) = \int_0^r \mathrm{d}r' p(r') = \begin{cases} 0 & r < 0 \, , \\ r & 0 \leq r < 1 \, , \\ 1 & r \geq 1 \, . \end{cases} \tag{13.8}$$

We introduce here only some of the most basic concepts for pseudo-random number generators. However, in huge simulations based on random numbers standard pseudo-random number generators provided by the various compilers may not be sufficient due to their rather short period and bad statistical properties. In this case it is, therefore, recommended to consult the literature [13, 21] and use more advanced techniques in order to obtain reliable results.

Linear Congruential Generators

Linear congruential generators are the simplest and most prominent random number generators. They produce a sequence of integers $\{x_n\}$, $n \in \mathbb{N}$ following the rule

$$x_{n+1} = (ax_n + c) \bmod m \, , \tag{13.9}$$

where a, c and m are positive integers which obey $0 \leq a, c < m$. Furthermore, the generator is initialized by its *seed* x_0, which is also a positive (in most cases odd) integer in the range $0 \leq x_0 < m$. The seed is commonly taken from, for instance, the system time in order to avoid repetition at a restart of the sequence. In many environments it is, therefore, necessary to fix the seed artificially whenever one wants to perform reproducible tests.

We note that the sequence resulting from Eq. (13.9) is bounded to the interval $x_n = [0, m-1]$ and, hence, its maximum period is m. However, the actual period of the sequence highly depends on the choices of the parameters a, c and m as well as on the seed x_0. In general, linear congruential generators are very fast and simple, however, they have rather short periods. Moreover, they are very susceptible to correlations since the value x_{n+1} is calculated from x_n only. (This is obviously a property which does not apply to real independent random numbers and it should therefore be eliminated!) In Sec. 13.3 we will discuss some simple methods which allow to identify such correlations.

One of the most prominent choices for the parameters in Eq. (13.9) are the Park - Miller parameters [21]:

$$a = 7^5, \quad c = 0, \quad m = 2^{31} - 1 \, . \tag{13.10}$$

Note that one has to be particularly careful when choosing $c = 0$. It follows from Eq. (13.9) that if $c = 0$ and if for any n, $x_n = 0$ one obtains $x_k = 0$ for all $k > n$.

The random numbers r_n described by the pdf (13.7) are obtained via

$$r_n = \frac{x_n}{m} \in [0, 1) \, . \tag{13.11}$$

Let us briefly discuss two famous improvements which concentrate on the reduction of correlations and an elongation of the period: The first idea which is referred to as *shuffling* [21] includes a second *pseudo-random* step. One calculates N numbers r_n from Eqs. (13.9) and (13.11) and stores these numbers in an array. If a random number is needed by the executing program, a second random integer $k \in [1, N]$ is drawn and the k-th element is taken from this array. In order to avoid that the same random number is used again, the k-th element of the array is replaced by a new random number which, again, is calculated from (13.9) and (13.11).

The second idea to improve the linear congruential generator is simply to include more previous elements of the sequence:

$$x_{n+1} = \left(\sum_{k=0}^{\ell} a_k x_{n-k} \right) \bmod m \ , \tag{13.12}$$

where $\ell > 0$ and $a_\ell \neq 0$. Again, the periodicity depends highly on the choice of the parameters and on the seed. A specific variation of random number generators using Eq. (13.12) are the FIBONACCI generators.

FIBONACCI Generators

The FIBONACCI sequence is given by

$$x_{n+1} = x_n + x_{n-1}, \qquad x_0 = 0, \quad x_1 = 1 \ , \tag{13.13}$$

which results for $n \geq 1$ in

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots \ . \tag{13.14}$$

Choosing in Eq. (13.12) $m = 10$, $\ell = 1$ and $a_0 = a_1 = 1$ simply leaves the last digits of the sequence (13.14):

$$1, 1, 2, 3, 5, 8, 3, 1, 4, 5, 9, \ldots \ . \tag{13.15}$$

This suggests the definition of a pseudo-random number generator based on the FIBONACCI sequence [14]. It is of the form

$$x_{n+1} = (x_n + x_{n-1}) \bmod m \ , \tag{13.16}$$

which, according to our previous discussion, allows a periodicity exceeding m. A straightforward generalization results in the so called *lagged* FIBONACCI *generators*:

$$x_{n+1} = (x_{n-p} \otimes x_{n-q}) \bmod m \ , \tag{13.17}$$

where $p, q \in \mathbb{N}$ and the operator $\otimes$ stands for any binary operation, such as addition, subtraction, multiplication or some logical operation. Two of the most popular lagged FIBONACCI generators are the *shift register generator* and the MARSAGLIA-ZAMAN *generator*.

The shift register generator is based on the *exclusive or* (XOR; $\oplus$) operation, which acts on each bit of the numbers x_{n-p} and x_{n-q}. In particular, the recurrence relation reads

$$x_n = x_{n-p} \oplus x_{n-q} \ . \tag{13.18}$$

The XOR operation $\oplus$ is shown in the following multiplication table:

a	b	$a \oplus b$
0	0	0
1	0	1
0	1	1
1	1	0

Hence, suppose that the binary representation of x_{n-p} is of the form 01001110... and for x_{n-q} we have 11110011... Then we get from Eq. (13.18):

x_{n-p}	0	1	0	0	1	1	1	0	...
x_{n-q}	1	1	1	1	0	0	1	1	...
x_n	1	0	1	1	1	1	0	1	...

A very prominent choice is given by $p = 250$ and $q = 103$ which yields a superior periodicity of order $\mathcal{O}(10^{75})$. The algorithm is initialized with the help of, for instance, a linear congruential generator.

In contrast, the Marsaglia-Zaman generator [18] uses the subtraction operation and may be written by introducing the so called *carry bit* Δ as

$$\Delta = x_{n-p} - x_{n-q} - c_{n-1} \, , \tag{13.19}$$

where $x_i \in [0, m]$ for all i. Then,

$$x_n = \begin{cases} \Delta & \Delta \geq 0 \, , \\ \Delta + m & \Delta < 0 \, . \end{cases} \tag{13.20}$$

and c_n is obtained via

$$c_n = \begin{cases} 0 & \Delta \geq 0 \, , \\ 1 & \Delta < 0 \, . \end{cases} \tag{13.21}$$

For the particular choice $p = 10$, $q = 24$ and $m = 2^{24}$ one finds an amazingly large periodicity of order $\mathcal{O}(10^{171})$. The random numbers x_n are integers in the interval $x_n \in [0, m]$, hence dividing the random numbers by m gives $r_n \in [0, 1]$.

13.3 Quality Tests

Here, we discuss some tests to check whether or not a given, finite sequence of numbers x_n consists of uniformly distributed random numbers out of the interval $x_n \in [0, 1]$.[1]

Statistical Tests

Statistical tests are generally the most simple methods to arrive at a first idea of the quality of a pseudo-random number generator. Statistical tests are typically based on the calculation of moments or correlations. Since we regard the simplified case of

[1]From now on we define quite generally the interval out of which random numbers x_n are drawn by $x_n \in [0, 1]$ keeping in mind that this interval depends on the actual method applied. This method determines whether zero or one is contained in the interval.

uniformly distributed, uncorrelated random numbers within the interval $[0,1]$, the moments can be calculated immediately from

$$\left\langle X^k \right\rangle = \int \mathrm{d}x\, x^k p(x) = \int_0^1 \mathrm{d}x\, x^k = \frac{1}{k+1}\,, \tag{13.22}$$

for $k \in \mathbb{N}$. These moments are approximated using the generated finite sequence of numbers $\{x_n\}_{n=1,\ldots,N}$ via

$$\left\langle X^k \right\rangle \approx \overline{x^k} = \frac{1}{N}\sum_{n=1}^{N} x_n^k\,. \tag{13.23}$$

The error of this approximation is of order $\mathcal{O}\left(1/\sqrt{N}\right)$ and

$$\left\langle X^k \right\rangle = \overline{x^k} + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)\,. \tag{13.24}$$

Another method studies correlations between the random numbers of the sequence and compare it with the analytical result. We obtain for uncorrelated random numbers:

$$\langle X_n X_{n+k}\rangle = \langle X_n\rangle^2 = \frac{1}{4}\,. \tag{13.25}$$

Another, quite evident test, is the analysis of the symmetry of the distribution. If $X_n \in [0,1]$ is uniformly distributed then it follows that $(1 - X_n) \in [0,1]$ should also be uniformly distributed.

Finally, we discuss a graphical test, known as the *spectral test* [13]. The spectral test consists of plotting subsequent random numbers x_n vs x_{n+1} and of visual inspection of the result. One expects the random numbers to uniformly fill the unit-square, however, if correlations exist, particular patterns might evolve. We illustrate this method in Fig. 13.1 where it is applied to a linear congruential generator (13.9) with two different sets of parameters.

Hypothesis Testing

Basically, one could employ different hypothesis tests, such as the KOLMOGOROV - SMIRNOV test, to test random numbers. These tests are rather basic and are discussed in numerous books on statistics. In what follows we shall briefly mention the χ^2-test; for more advanced techniques we refer the reader to the literature [9, 10].

The χ^2-test tests the pdf directly. One starts by sorting the N elements of the sequence into a histogram. Suppose we would like to have M bins and, hence, the width of every bin is given by $1/M$. We now count the number of elements which lie within bin k, i.e. within the interval $[(k-1)/M, k/M]$, and denote this number by n_k. The histogram array h is given by $h = c(n_1, n_2, \ldots, n_M)^T$ where the constant $c = M/N$ normalizes the histogram. In Fig. 13.2 we show three different histograms for $N = 10^5$, $N = 10^6$ and $N = 10^7$ uniformly distributed random numbers as obtained with the PARK - MILLER linear congruential generator. In Fig. 13.3 we present a histogram for $N = 10^7$ obtained with the bad linear congruential generator defined in Fig. 13.1(b). We recognize numerous empty bins which are a clear indication that the random numbers are not uniformly distributed.

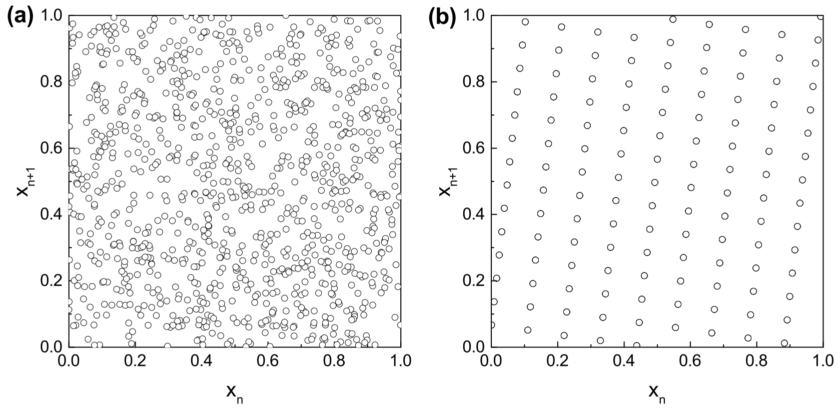

Figure 13.1: Spectral test for a linear congruential generator. We used the PARK - MILLER parameters, **(a)** $a = 7^5$, $c = 0$, and $m = 2^{31}-1$, **(b)** $a = 137$, $c = 0$, and $m = 2^{11}$, and plotted $N = 10^3$ subsequent pairs (x_n, x_{n+1}) of random numbers. In frame **(a)** the random numbers evolve nicely distributed within the unit square, showing no obvious correlations. On the other hand, in frame **(b)** subsequent random numbers lie on hyperplanes and, thus, develop correlations: They do not fill the unit square uniformly.

Let us briefly remember some points from probability theory [4, 11]. One can show, that if numbers Q_n are normally distributed random variables, their sum

$$x = \sum_{n=1}^{\nu} Q_n^2 \,, \tag{13.26}$$

follows a χ^2-distribution where ν is the number of degrees of freedom. The pdf of the χ^2-distribution is given by

$$p(x;\nu) = \frac{x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)} \,, \quad x \geq 0 \,, \tag{13.27}$$

where $\Gamma(\cdot)$ denotes the Γ-function. The probability of finding the variable x within the interval $[a, b] \subset \mathbb{R}^+$ can be calculated as

$$P(x \in [a,b];\nu) = \int_a^b \mathrm{d}x \, p(x;\nu) \,, \tag{13.28}$$

and in particular for $a = 0$ we obtain

$$P(x < b;\nu) = \int_0^b \mathrm{d}x \, p(x;\nu) = F(b;\nu) \,. \tag{13.29}$$

Here we introduced the cdf $F(b;\nu)$. Let us consider the inverse problem: the probability that $x \leq b$ is equal to α, i.e. $F(b;\nu) = \alpha$. We then calculate the upper bound b by inverting Eq. (13.29) and obtain:

$$b = F^{-1}(\alpha;\nu) \,. \tag{13.30}$$

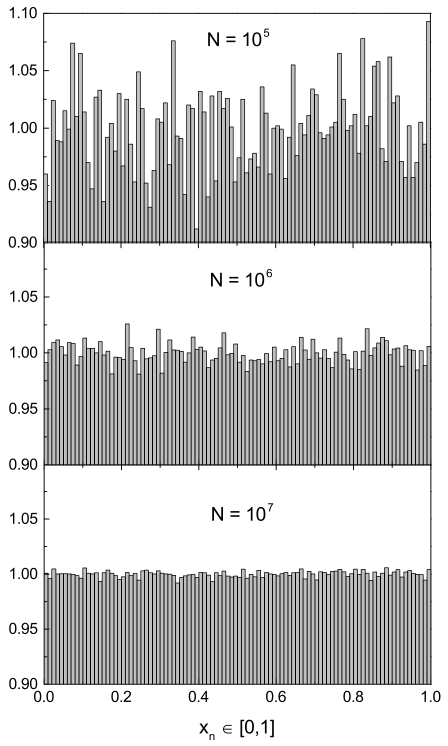

Figure 13.2: Histograms for $N = 10^5$, $N = 10^6$ and $N = 10^7$, $M = 100$ bins as obtained with the Park - Miller linear congruential generator, $a = 7^5$, $c = 0$ and $m = 2^{31} - 1$.

These values are tabulated [1, 19].

We return to our particular example: the hypothesis is that the sequence $\{x_n\}$ generated by some pseudo-random number generator complies to a uniform distribution. It is a consequence of the central limit theorem that the deviations from the theoretically expected values n_k^{th} obey a normal distribution. We define the variable

$$x = \chi^2 = \sum_{k=1}^{M} \frac{(n_k - n_k^{th})^2}{n_k^{th}} \ . \tag{13.31}$$

If our hypothesis is true, χ^2 follows a χ^2-distribution with $\nu = M - 1$ because the requirement that the sum of all numbers n_k is equal to N reduces the degrees of freedom by one. We employ relation (13.30) for $\alpha = 0.85$ and $\nu = 99$ and obtain $b = 113$. Hence, values $\chi^2 < b$ are very likely if χ^2 really follows a χ^2 distribution, while values $\chi^2 > b$ are unlikely and therefore the hypothesis may require a review. However, it has to be emphasized that it is fundamentally impossible to *verify* a hypothesis. It can only be falsified or strengthened. We note that the resulting value of χ^2 will highly depend on the seed number of the generator as long as the maximum period has not been reached.

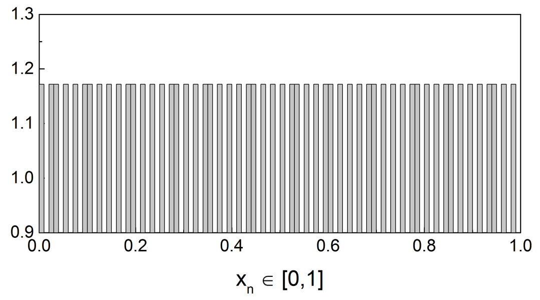

Figure 13.3: Histogram for $N = 10^7$ and $M = 100$ bins as obtained with a linear congruential generator with parameters $a = 137$, $c = 0$ and $m = 2^{11}$.

Bibliography

[1] Abramovitz, M., Stegun, I.A. (eds.): Handbook of Mathemathical Functions. Dover, New York (1965)

[2] Breuer, H.P., Petruccione, F.: Open Quantum Systems, chap. 1. Clarendon Press, Oxford, U. K. (2010)

[3] Chaitin, G.J.: Randomness and mathematical proof. Scientific American **232**, 47 (1975)

[4] Chow, Y.S., Teicher, H.: Probability Theory, 3rd edn. Springer Texts in Statistics. Springer, Berlin, Heidelberg (1997)

[5] Coffey, W.T., Kalmykov, Y.P.: The Langevin Equation, 3rd edn. World Scientific Series in Contemporary Chemical Physics: Volume 27. World Scientific, New Jersey (2012)

[6] Devroye, L.: Non-Uniform Random Variate Generation. Springer, Berlin, Heidelberg (1986)

[7] Dubitzky, W., Wolkenhauer, O., Cho, K.H., Yokota, H. (eds.): Encyclopedia of Systems Biology, p. 1596. Springer, Berlin, Heidelberg (2013)

[8] Gentle, J.E.: Random Number Generation and Monte Carlo Methods. Statistics and Computing. Springer, Berlin, Heidelberg (2003)

[9] Iversen, G.P., Gergen, I.: Statistics. Springer Undergraduate Textbooks in Statistics. Springer, Berlin, Heidelberg (1997)

[10] Keener, R.W.: Theoretical Statistics. Springer, Berlin, Heidelberg (2010)

[11] Kienke, A.: Probability Theory. Universitext. Springer, Berlin, Heidelberg (2008)

[12] Klenke, A.: Probability Theory. Universitext. Springer, Berlin, Heidelberg (2014)

[13] Knuth, D.: The Art of Computer Programming, vol. II, 3rd edn. Addison Wesley, Menlo Park (1998)

[14] Knuth, D.: The Art of Computer Programming, vol. IV. Addison Wesley, Menlo Park (2011)

[15] Laing, C., Lord, G.J. (eds.): Stochastic Methods in Neuroscience. Oxford University Press, Oxford, UK (2009)

[16] Lax, M., Cai, W., Xu, M.: Random Processes in Physics and Finance. Oxford Finance Series. Oxford University Press, Oxford, UK (2013)

[17] von der Linden, W., Dose, V., von Toussaint, U.: Bayesian Probability Theory. Cambridge University Press, Cambridge, UK (2014)

[18] Marsaglia, G., Zaman, A.: A new class of random number generators. Ann. Appl. Prob. **1**, 462 – 480 (1991)

[19] Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W.: NIST Handbook of Mathematical Functions. Cambridge University Press, Cambridge, UK (2010)

[20] Papoulis, A., Pillai, S.: Probability, Random Variables and Stochastic Processes. McGraw Hill, New York (2001)

[21] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[22] Ripley, B.D.: Stochastic Simulation. Wiley, New York (2006)

[23] Rotar, V.: Probability Theory. World Scientific, Singapore (1998)

[24] Tapiero, C.S.: Risk and Financial Management: Mathematical and Computational Methods. Wiley, New York (2004)

Appendix A

Vector and matrix norms

An inner product on a (complex) vector space $\mathbb{X}$ is any mapping s from $\mathbb{X} \times \mathbb{X}$ into $\mathbb{C}$,

$$\mathbf{x} \in \mathbb{X}, \mathbf{y} \in \mathbb{X} \quad \rightarrow \quad s(\mathbf{x}, \mathbf{y}) \in \mathbb{C},$$

which satisfies the following conditions:

1. $s(\mathbf{x}, \mathbf{y})$ is linear with respect to $\mathbf{x}$:

$$s(\lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2, \mathbf{y}) = \lambda_1 s(\mathbf{x}_1, \mathbf{y}) + \lambda_2 s(\mathbf{x}_2, \mathbf{y}), \quad \forall \mathbf{x}_1, \mathbf{x}_2, \mathbf{y} \in \mathbb{X}, \lambda_1, \lambda_2 \in \mathbb{C}.$$

2. $s(\mathbf{x}, \mathbf{y})$ is *Hermitian*:

$$s(\mathbf{y}, \mathbf{x}) = s(\mathbf{x}, \mathbf{y})^\dagger, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{X}.$$

3. $s(\mathbf{x}, \mathbf{x})$ is *positive definit*:

$$s(\mathbf{x}, \mathbf{x}) > 0, \quad \forall \mathbf{x} \neq 0.$$

A useful relation satisfied by any inner product is the so-called Cauchy - Schwartz inequality:

$$s(\mathbf{x}, \mathbf{y})^2 \leq s(\mathbf{x}, \mathbf{x}) s(\mathbf{y}, \mathbf{y}). \tag{A.1}$$

In the particular case of the vector space $\mathbb{X} = \mathbb{C}^n$, a "canonical" inner product is the *Euclidean inner product*. The Euclidean inner product of two vectors $\mathbf{x} = [x_i], i = 1, \ldots, n$ and $\mathbf{y} = [y_i], i = 1, \ldots, n$ of $\mathbb{C}^n$ is defined by

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i y_i^\star, \tag{A.2}$$

or written in a matrix notation

$$(x, y) = y^\dagger x. \tag{A.3}$$

A *vector norm* $||\mathbf{x}|| : \mathbb{X} \rightarrow \mathbb{R}^+$ satisfies the following three conditions:

1. $||\mathbf{x}|| \geq 0, \quad \forall \mathbf{x} \in \mathbb{X}$ and $||\mathbf{x}|| = 0$ if and only if $\mathbf{x} = 0$.

2. $||\alpha \mathbf{x}|| = |\alpha| \, ||\mathbf{x}||, \quad \alpha \in \mathbb{C}, \mathbf{x} \in \mathbb{X}.$

3. $||\mathbf{x}+\mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{C}.$

For the particular case when $\mathbb{X} = \mathbb{C}^n$, we can associate with the inner product (A.2) the Euclidean norm of a complex vector defined by

$$||\mathbf{x}||_2 = (\mathbf{x}, \mathbf{x})^{1/2}.$$

The most commonly used vector norms in numerical linear algebra are special cases of the Hölder norms:

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}. \tag{A.4}$$

Note that the limit of $||\mathbf{x}||_p$ for $p \to \infty$ exists and that it is equal to the maximum modulus of the x_is. This defines a norm denoted by $||\cdot||_\infty$. The cases $p = 1$, $p = 2$, and $p = \infty$ are the most important norms in practice:

$$\begin{aligned} ||\mathbf{x}||_1 &= |x_1| + |x_2| + \cdots + |x_n|, \\ ||\mathbf{x}||_2 &= \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2}, \\ ||\mathbf{x}||_\infty &= \max_{i=1,2,\ldots,n} |x_i|. \end{aligned}$$

The Cauchy - Schwartz inequality (A.1) becomes:

$$|(\mathbf{x}, \mathbf{y})| \leq ||\mathbf{x}||_2 ||\mathbf{y}||_2.$$

We define now for a general matrix A in $\mathbb{C}^{n \times m}$ the following special set of norms:

$$||A||_{pq} = \max_{\mathbf{x} \in \mathbb{C}^m, \mathbf{x} \neq 0} \frac{||A\mathbf{x}||_p}{||\mathbf{x}||_q}. \tag{A.5}$$

This is the maximum amplification length that a matrix A is able to induce on a vector $\mathbf{x}$. According to Eq. (A.5) the norm $||\cdot||_{pq}$ is induced by two vector norms, namely $||\cdot||_p$ and $||\cdot||_q$. These norms satisfy the usual properties of norms and therefore:

1. $||A|| \geq 0, \quad \forall A \in \mathbb{C}^{n \times m}$ and $||A|| = 0$ if and only if $A = 0$.

2. $||\alpha A|| = |\alpha|\, ||A||, \quad \forall A \in \mathbb{C}^{n \times m}, \forall \alpha \in \mathbb{C}.$

3. $||A + B|| \leq ||A|| + ||B||, \quad \forall A, B \in \mathbb{C}^{n \times m}.$

A matrix norm which satisfies the above three properties is nothing but a vector norm applied to the matrix A which is considered as a vector consisting of the m columns stacked into a vector of size $n \times m$.

The most important cases are again those associated with $p, q = 1, 2, \ldots, \infty$. The case $q = p$ is of particular interest and the associated norm $||\cdot||_{pq}$ is simply denoted by $||\cdot||_p$ and is called a "p-norm". A fundamental property of a p-norm is that

$$||AB||_p \leq ||A||_p \, ||B||_q,$$

which is an immediate consequence of Eq. (A.5).

The following equalities satisfied by the matrix norms defined above, lead to alternative definitions that are often easier to work with:

$$||A||_1 = \max_{j=1,\ldots,m} \sum_{i=1}^{n} |a_{ij}|, \tag{A.6}$$

$$||A||_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|, \tag{A.7}$$

$$||A||_2 = \sqrt{\rho(A^\dagger A)} = \sqrt{\rho(AA^\dagger)}, \tag{A.8}$$

with $\rho(A)$ the spectral radius of matrix A (see Subsec. 2.2.2).

The eigenvalues of $A^\dagger A$ are non-negative. Their square roots are called *singular values* of A and are denoted by $\sigma_i, i = 1, \ldots, m$. Thus, the relation (A.8) states that $||A||_2$ is equal to σ_1, the largest singular value of A.

We define one more matrix norm which is, in contrast to the p-norms, not derived from vector norms, the FROBENIUS norm:

$$||A||_{\mathrm{F}} = \sqrt{\sum_{j=1}^{m} \sum_{i=1}^{n} |a_{ij}|^2}.$$

It can be viewed as the 2-norm of the column (or row) vector in $\mathbb{C}^{n^2}$ consisting of all the columns (respectively rows) of A listed from 1 to m (respectively 1 to n.) It does not satisfy some of the properties of the p-norms. For example, the FROBENIUS norm of the identity matrix is not equal to one.

This appendix followed quite closely Chap. 1 of the book by Y. SAAD [5]. More information on vector and matrix norms can be found in the book by J. STOER and R. BULIRSCH [6], the book by J. H. KWAK and S. HONG, S [2], or in the book by G. STRANG [8].

Appendix B

GERSHGORIN discs

Diagonal dominance is related to an important result in Numerical Linear Algebra known as GERSHGORIN's theorem. This theorem allows rough locations for all the eigenvalues λ_i of the matrix A to be determined. In some situations, it is desirable to determine these locations in the complex plane by directly exploiting some knowledge of the entries of the matrix A. The simplest such result is the bound

$$|\lambda_i| \leq ||A||$$

for any matrix norm. GERSHGORIN's theorem provides a more precise localization result.

Theorem 1
(GERSHGORIN) Any eigenvalue λ of a matrix A is located in one of the closed discs of the complex plane centered at the element a_{ii} and having the radius

$$r_i = \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|.$$

In other words:

$$\forall\,\lambda \in \sigma(A), \quad \exists\, i \quad \text{such that} \quad |\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|. \tag{B.1}$$

Proof: Let λ be an eigenvalue of A. Choose a corresponding eigenvector $\mathbf{x} = [x_i]$ and scale $\mathbf{x}$ so that one component $|x_i| = 1$ and the others are $|x_j| \leq 1$ for $j \neq i$. Since $A\mathbf{x} = \lambda\mathbf{x}$ and in particular:

$$\sum_{j=1}^{n} a_{ij}x_j = \lambda x_i = \lambda.$$

Thus, splitting the sum and keeping in mind that $x_i = 1$, we get:

$$\sum_{i\neq j} a_{ij}x_j + a_{ii} = \lambda.$$

Therefore, applying the triangle inequality:

$$|\lambda - a_{ii}| = \left| \sum_{j \neq i} a_{ij}\, x_j \right| \leq \sum_{j \neq i} |a_{ij}||x_j| \leq \sum_{j \neq i} |a_{ij}| = r_i.$$

This completes the proof.

Since the result also holds for the transpose of A, a version of the theorem can also be formulated based on column sums instead of row sums.

Not surprising, the n discs defined in the theorem are called GERSHGORIN discs. The theorem states that the union of these n discs contains the spectrum of A. It can also be shown that if there are m GERSHGORIN discs whose union S is disjoint from all other discs, then S contains exactly m eigenvalues (counted with their multiplicities). For example, when one disc is disjoint from all the others, then it must contain exactly one eigenvalue.

More detailed information on GERSHGORIN discs can be found in the original article by GERSHGORIN [1], in the book by R. S. VARGA [9], and in the book by Y. SAAD [5] Subsec. 4.2.3.

Appendix C

Singular Value Decomposition of real matrices

C.1 Introduction

Mathematically, the *Singular Value Decomposition* (SVD) is a fundamental theorem of linear algebra. (One could argue that it is *the* fundamental theorem.) The singular value decomposition states that every $n \times p$, $n \geq p$ matrix A can be written as the product of three matrices: $A = U\Sigma V^T$ where

- U is an orthogonal $n \times n$ matrix.
- Σ is a diagonal $n \times p$ matrix. In practice, the diagonal elements are ordered so that $\sigma_{ii} \geq \sigma_{jj}$ for all $i < j$. These diagonal elements are called the *singular values*.
- V is an orthogonal $p \times p$ matrix and V^T represents its transpose.

The SVD represents the essential geometry of a linear transformation. It tells us that every linear transformation is a composition of three fundamental actions. Reading the equation $A = U\Sigma V^T$ from right to left:

- The matrix V represents a rotation or reflection of vectors in the p-dimensional domain.
- The matrix Σ represents a linear dilation or contraction along each of the p coordinate directions. If $n \neq p$, this step also canonically embeds (or projects) the p-dimensional domain into (or onto) the n-dimensional range.
- The matrix U represents a rotation or reflection of vectors in the n-dimensional range.

Thus the SVD specifies that every linear transformation is fundamentally a rotation or reflection, followed by a scaling, and followed by another rotation or reflection. The article by G. Strang [7] about this fundamental theorem of linear algebra included this geometric interpretation if the SVD.

Because of its usefulness, the singular value decomposition is a fundamental technique for multivariate data analysis. A common goal of multivariate data analysis is to reduce the dimension of the problem by choosing a small linear subspace that captures important properties of the data. The SVD is used in two important dimension-reducing operations:

- **Low-rank approximations:** Recall that the diagonal elements of the σ matrix in the SVD are computed in decreasing order. The SVD has a wonderful mathematical property: if you choose some integer $k \geq 1$ and let D be the diagonal matrix formed by replacing all singular values after the k^{th} by zero, then the matrix UDV^T is the best rank-k approximation to the original matrix A.

- **Principal components analysis:** The principal component analysis is usually presented in terms of eigenvectors of a correlation matrix, but you can show that the principal component analysis follows directly from the SVD. (In fact, you can derive the eigenvalue decomposition of a matrix, from the SVD.) The principal components of A^TA are the columns of the V matrix; the scores are the columns of U. Dimension reduction is achieved by truncating the number of columns in U, which results in the best rank-k approximation of the data.

These introductory remarks were found in a blog by R. WICKLIN [10].

C.2 Details

We have the basic formula of the singular value decomposition of some matrix A as

$$A = U\Sigma V^T. \tag{C.1}$$

The matrices U and V are each orthogonal in the sense that their columns are orthonormal:

$$\begin{aligned} \sum_{i=1}^{n} u_{ik}u_{in} &= \delta_{kn}, \quad k,n \in [1,p] && \text{(C.2)} \\ \sum_{j=1}^{p} v_{jk}v_{jn} &= \delta_{kn}, \quad k,n \in [1,p], && \text{(C.3)} \end{aligned}$$

with u_{ij} and v_{ij} the elements of the matrices U and V, respectively. Since the matrix V is a square matrix it is also row-orthnormal and $V \cdot V^T = 1$.

The decomposition (C.1) can always be done, no matter how singular the matrix A is, and it is almost “unique”. It is unique up to (i) making the same permutation of the columns of U, elements of Σ, and columns of V (or rows of V^T), or (ii) forming linear combinations of any columns of U and V whose corresponding elements of Σ happen to be exactly equal.

C.2.1 SVD of a square matrix

If A is a square matrix of size $n \times n$, the the matrices U, V, and Σ are all square matrices of the same size. The inverse are also trivial to compute: U and V are orthogonal, thus their inverse are equal to their transpose. Σ is diagonal and, thus, its inverse is the diagonal matrix whose elements are the reciprocals of the elements σ_i. It follows from Eq. (C.1) that the inverse of A is given by

$$A^{-1} = V \cdot \mathrm{diag}(1/\sigma_i) \cdot U^T. \tag{C.4}$$

There is only one thing which can go wrong with this construction: one of the singular values σ_i is zero or (numerically) so small that its value is dominated by roundoff error. If more than one of the σ_i have this problem, the matrix A is even "more" singular. Consequently, SVD provides you with a clear diagnosis of the situation you are dealing with.

For singular matrices the concepts of *nullspace* and *range* are important. We consider the familiar set of linear equations

$$A\mathbf{x} = \mathbf{b}, \tag{C.5}$$

where A is a square matrix, $\mathbf{x}$ and $\mathbf{b}$ are vectors. Eq. (C.5) defines A is a linear mapping from the vector space $\mathbf{x} \in \mathbb{X}$ to the vector space $\mathbf{b} \in \mathbb{B}$. If A is singular then there is some subspace of $\mathbb{X}$, called the nullspace, that is mapped to zero, $A\mathbf{x} = 0$. The dimension of the nullspace (the number of linearly independent vectors $\mathbf{x}$ that can be found in it) is called the *nullity* of A.

There is, of course, some subspace of $\mathbb{B}$ that can be reached by A in the sense that there exists some $\mathbf{x}$ which is mapped there. This subspace of $\mathbb{B}$ is called the *range* if A. The dimension of the range is called the *rank* of A. If A is non-singular then its range will be all of the vector space $\mathbb{B}$ and its rank will be n. If A is singular then its rank will be less than n and the relevant theorem is "rank plus nullity is n". SVD explicitly constructs orthonormal bases for the nullspace and the range of the matrix. In particular, the columns of U whose same-numbered elements σ_j are non-zero are an orthonormal set of basis vectors that span the range. On the other hand, the columns of V whose same-numbered elements σ_j are zero are an orthonormal base for the nullspace.

If $\mathbb{B}$ is not in the range of the singular matrix A then the set of equations (C.5) has no solution. Nevertheless, Eq. (C.4) can be applied in this case to construct a " solution" vector $\mathbf{x}$. This vector will not exactly solve $A\mathbf{x} = \mathbf{b}$ but, among all possible vectors $\mathbf{x}$, it will correspond to the closest possible solution in a least squares sense. In other words Eq. (C.4) finds

$$\mathbf{x} \quad \text{which minimizes} \quad r \equiv |A \cdot \mathbf{x} - \mathbf{b}|. \tag{C.6}$$

The number r is called the *residual* of the solution.

In this discussion we pretended that a matrix A is singular or else it isn't. This is certainly true analytically. However, numerically it is the far more common situation that some σ_j are very small but non-zero: the matrix A is ill-conditioned. In this situation the methods of *LU* decomposition or Gaussian elimination may actually give a formal solution to the set of equations; but the solution's vector may have wildly large components whose algebraic cancellation, when multiplied with the matrix A, may give a very poor approximation of the vector $\mathbf{b}$. In such cases, the solution vector $\mathbf{x}$ obtained by *zeroing* the small σ_j and then using Eq. (C.4) is very often better (in the sense of the residual $|A\mathbf{x} - \mathbf{b}|$ being smaller) than *both* the direct method solution *and* the SVD solution where small σ_j are left non-zero.

It may seem paradoxical that this can be so, since zeroing a singular value corresponds to throw away one linear combination of the set of equations we are trying to solve. The resolution of this paradox is that we are throwing away precisely a combination of equations that is so corrupted by roundoff error as to be at best useless; usually it is even worse than useless as it “pulls” the solution vector away off towards infinity along some direction that is almost a nullspace vector. In doing this it compounds the roundoff problem and makes the residual larger.

Consequently SVD cannot be applied blindly. One has to exercise some discretion in deciding at what threshold to zero the small σ_j are to be zeroed. One also has to have some idea what size of computed residuals $|A\mathbf{x} - \mathbf{b}|$ are acceptable.

If desired, more detailed information can be found, for instance, in Refs. [6, 4, 3].

Bibliography

[1] Gershgorin, S.: Über die Abgrenzung der Eigenwerte einer Matrix. Izv. Akad. Nauk. USSR Otd. Fiz.-Mat Nauk (in German) **6**, 749 (1931)

[2] Kwak, J.H., Hong, S.: Linear Algebra. Springer, Berlin, Heidelberg (2004)

[3] Martin, R.S., Peters, G., Wilkinson, J.H.: Symmetric decomposition of a positive definite matrix. In: J.H. Wilkinson, C. Reinsch (eds.) Handbook for Automatic Computation, Volume II: Linear Algebra, Die Grundlehren der mathematischen Wissenschaften, Vol. 186, pp. 9 – 30. Springer, Berlin, Heidelberg (1971)

[4] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C++, 2nd edn. Cambridge University Press, Cambridge, UK (2002)

[5] Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2003)

[6] Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, 2nd edn. Springer, Berlin, Heidelberg (1993)

[7] Strang, G.: The fundamental theorem of linear algebra. The American Mathematical Monthly **100**, 848–855 (1993). DOI 10.2307/2324660

[8] Strang, G.: Introduction to Linear Algebra, 4th edn. Cambridge University Press, Cambridge, UK (2009)

[9] Varga, R.S.: Geršgorin and His Circles. Springer Series in Computational Mathematics. Springer, Berlin, Heidelberg (2004)

[10] Wicklin, R.: The singular value decomposition: A fundamental technique in multivariate data analysis (2017)

Index