Numerical Methods for Ordinary Differential Equations

Vuik, Cornelis; Vermolen, F.J.; van Gijzen, M.B.; Vuik, Thea

DOI
10.5074/t.2023.001

Publication date
2023

Document Version
Final published version

Citation (APA)
Vuik, C., Vermolen, F. J., van Gijzen, M. B., & Vuik, T. (2023). *Numerical Methods for Ordinary Differential Equations*. TU Delft OPEN Publishing. https://doi.org/10.5074/t.2023.001

Numerical Methods for Ordinary Differential Equations

C.Vuik, F.J. Vermolen, M.B. van Gijzen & M.J. Vuik

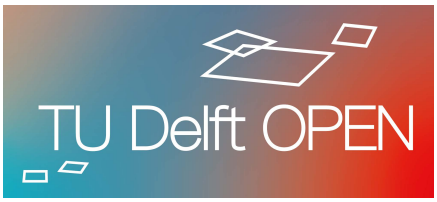
TU Delft OPEN

Numerical Methods for Ordinary Differential Equations

C. Vuik
F.J. Vermolen
M.B. van Gijzen
M.J. Vuik

First edition 2007
Second edition 2015

ISBN 978-94-6366-665-7 (Ebook)
ISBN 978-94-6366-664-0 (Paperback)
Doi: https://doi.org/10.5074/t.2023.001



Keywords: numerical analysis, ordinary differential equations

Preface

In this book we discuss several numerical methods for solving ordinary differential equations. We emphasize the aspects that play an important role in practical problems. We confine ourselves to ordinary differential equations with the exception of the last chapter in which we discuss the heat equation, a parabolic partial differential equation. The techniques discussed in the introductory chapters, for instance interpolation, numerical quadrature and the solution to nonlinear equations, may also be used outside the context of differential equations. They have been included to make the book self-contained as far as the numerical aspects are concerned. Chapters, sections and exercises marked with a * are not part of the Delft Institutional Package.

The numerical examples in this book were implemented in Matlab, but also Python or any other programming language could be used. A list of references to background knowledge and related literature can be found at the end of this book. Extra information about this course can be found at `http://NMODE.ewi.tudelft.nl`, among which old exams, answers to the exercises, and a link to an online education platform. We thank Matthias Möller for his thorough reading of the draft of this book and his helpful suggestions.

Delft, June 2016
C. Vuik

The figure at the cover shows the Erasmus bridge in Rotterdam. Shortly after the bridge became operational, severe instabilities occurred due to wind and rain effects. In this book we study, among other things, numerical instabilities and we will mention bridges in the corresponding examples. Furthermore, numerical analysis can be seen as a bridge between differential equations and simulations on a computer.

Contents

Chapter 1

Introduction

1.1 Some historical remarks

Modern applied mathematics started in the 17th and 18th century with scholars like Stevin, Descartes, Newton and Euler. Numerical aspects found a natural place in the analysis but the expression "numerical mathematics" did not exist at that time. However, numerical methods invented by Newton, Euler and at a later stage by Gauss still play an important role even today.

In the 17th and the 18th century fundamental laws were formulated for various subdomains of physics, like mechanics and hydrodynamics. These laws took the form of simple looking mathematical equations. To the disappointment of many scientists, these equations could be solved analytically in a few special cases only. For that reason technological development has only been loosely connected with mathematics. The introduction and availability of digital computers has changed this. Using a computer it is possible to gain quantitative information with detailed and realistic mathematical models and numerical methods for a multitude of phenomena and processes in physics and technology. Application of computers and numerical methods has become ubiquitous. Statistical analysis shows that non-trivial mathematical models and methods are used in 70% of the papers appearing in the professional journals of engineering sciences.

Computations are often cheaper than experiments; experiments can be expensive, dangerous or downright impossible. Real life experiments can often be performed on a small scale only, which makes their results less reliable.

1.2 What is numerical mathematics?

Numerical mathematics is a collection of methods to approximate solutions to mathematical equations numerically by means of *finite* computational processes.

In large parts of mathematics the most important concepts are mappings and sets. In numerical mathematics the concept of computability should be added. Computability means that the result can be obtained in a finite number of operations (so the computation time will be finite) on a finite subset of the rational numbers (because a computer has only finite memory).

In general the result will be an approximation of the solution to the mathematical problem, since most mathematical equations contain operators based on infinite processes, like integrals and derivatives. Moreover, solutions are functions whose domain and image may (and usually do) contain irrational numbers.

Because, in general, numerical methods can only obtain approximate solutions, it makes sense to apply them only to problems that are insensitive to small perturbations, in other words to problems that are *stable*. The concept of stability belongs to both numerical and classical mathematics. An important instrument in studying stability is functional analysis. This discipline

also plays an important role in error analysis (investigating the difference between the numerical approximation and the solution).

Calculating with only a finite subset of the rational numbers has many consequences. For example: a computer cannot distinguish between two polynomials of sufficiently high degree. Consequently, methods based on the main theorem of algebra (i.e. that an nth degree polynomial has exactly n complex zeros) cannot be trusted. Errors that follow from the use of finitely many digits are called *rounding errors* (Section 1.4).

An important aspect of numerical mathematics is the emphasis on efficiency. Contrary to ordinary mathematics, numerical mathematics considers an increase in efficiency, i.e. a decrease of the number of operations and/or amount of storage required, as an essential improvement. Progress in this aspect is of great practical importance and the end of this development has not been reached yet. Here, the creative mind will meet many challenges. On top of that, revolutions in computer architecture will overturn much conventional wisdom.

1.3 Why numerical mathematics?

A big advantage of numerical mathematics is that it can provide answers to problems that do not admit closed-form solutions. Consider for example the integral

$$\int_0^\pi \sqrt{1+\cos^2 x}dx.$$

This is an expression for the arc length of one arc of the curve $y(x) = \sin x$, which does not have a solution in closed form. A numerical method, however, can approximate this integral in a very simple way (Chapter 5). An additional advantage is that a numerical method only uses standard function evaluations and the operations addition, subtraction, multiplication and division. Because these are exactly the operations a computer can perform, numerical mathematics and computers form a perfect combination.

An advantage of analytical methods is that the solution is given by a mathematical formula. From this, insight in the behavior and the properties of the solution can be gained. For numerical approximations, however, this is not the case. In that case, visualization tools may be used to gain insight in the behavior of the solution. Using a numerical method to draw a graph of a function is usually a more useful tool than evaluating the solution at a large number of points.

1.4 Rounding errors

A computer uses a finite representation of the all numbers in $\mathbb{R}$. These are stored in a computer in the form

$$\pm 0.d_1 d_2 \dots d_n \cdot \beta^e, \tag{1.1}$$

in which, by definition, $d_1 > 0$ and $0 \leq d_i < \beta$. The normalization is needed in order to prevent a waste of digits and to make the representation unambiguous. We call the value in equation (1.1) a *floating point number* (representation) in which $0.d_1 d_2 \dots d_n$ is called the *mantissa*, β the *base* and e (integer) the *exponent*, where $L < e < U$. Characteristic values for $|L|$ and U are in the range $[100, 1000]$, often, $\beta = 2$ (binary representation) and $n = 24$ (*single* precision) or $n = 53$ (*double* precision). Most computers and software packages (Matlab) satisfy the IEEE-754 standard, and hence provide single-[1] and double-precision[2] computations.

Let for $x \in \mathbb{R}$

$$0.d_1 \dots d_n \cdot \beta^e \leq x < 0.d_1 d_2 \dots (d_n + 1) \cdot \beta^e,$$

[1] http://en.wikipedia.org/wiki/Single-precision_floating-point_format
[2] http://en.wikipedia.org/wiki/Double-precision_floating-point_format

where for simplicity x is taken positive, and we assume that $d_n < \beta - 1$. *Rounding* x means that x will be replaced with the floating point number closest to x, which is called $fl(x)$. The error caused by this process is called *rounding error*, and $fl(x)$ is written as

$$fl(x) = x(1 + \varepsilon). \tag{1.2}$$

The value $|x - fl(x)| = |\varepsilon x|$ is called the *absolute error* and $|x - fl(x)|/|x| = |\varepsilon|$ the *relative error*. The difference between the floating point numbers enclosing x is β^{e-n}. Rounding yields $|x - fl(x)| \leq \beta^{e-n}/2$, hence the absolute error is bounded by

$$|\varepsilon x| \leq \frac{1}{2}\beta^{e-n}.$$

Because $|x| \geq \beta^{e-1}$ (since $d_1 > 0$), the relative error is bounded by

$$|\varepsilon| \leq eps \tag{1.3}$$

with the computer's relative precision eps defined as

$$eps = \frac{1}{2}\beta^{1-n}. \tag{1.4}$$

From $\beta = 2$ and $n = 53$ it follows that $eps \approx 10^{-16}$, thus in double precision approximately 16 decimal digits are used.

Figure 1.1 shows the distribution of the floating point numbers $0.1d_2d_3 \cdot \beta^e \ ; \ e = -1, 0, 1, 2$ in base 2 (binary numbers). These floating point numbers are not uniformly distributed and there is a gap around 0. A computational result lying within this neighborhood is called *underflow*. Most machines provide a warning, replace the result with 0 and continue. A computational result with absolute value larger than the largest floating point number that can be represented is called *overflow*. The machine warns and may continue with Inf's (infinity).

A final remark is that rounding errors are deterministic. If one repeats the algorithm, the same results are obtained.

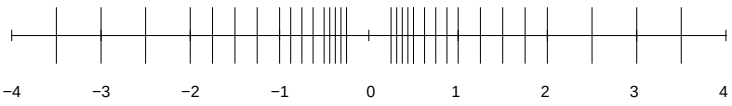


Figure 1.1: Distribution of $\pm 0.1d_2d_3 \cdot \beta^e, \ \beta = 2, e = -1, 0, 1, 2$.

Next, let us investigate how computers execute arithmetic operations in floating point arithmetic. Processors are very complex and usually the following model is used to simulate reality. Let $\circ$ denote an arithmetic operation ($+, -, \times$ or $/$) and let x and y be floating point numbers (hence, $x = fl(x)$, $y = fl(y)$). Then the machine result of the operation $x \circ y$ will be

$$z = fl(x \circ y). \tag{1.5}$$

The exact result of $x \circ y$ will not be a floating point number in general, hence an error results. From formula (1.2) it follows that

$$z = \{x \circ y\}(1 + \varepsilon), \tag{1.6}$$

for some ε satisfying (1.3) and $z \neq 0$. Expression (1.6) describes the error due to converting the result of an exact calculation to floating point form.

Next, we suppose that x and y are no floating point numbers: the corresponding floating point numbers $fl(x)$ and $fl(y)$ satisfy $fl(x) = x(1+\varepsilon_1)$, $fl(y) = y(1+\varepsilon_2)$. Again, $x \circ y$ should be calculated. Using the floating point numbers in this calculation, the absolute error in the calculated result $fl(fl(x) \circ fl(y))$ satisfies:

$$|x \circ y - fl(fl(x) \circ fl(y))| \leq |x \circ y - fl(x) \circ fl(y)| + |fl(x) \circ fl(y) - fl(fl(x) \circ fl(y))|. \quad (1.7)$$

From this expression it follows that the error consists of two terms. The first term is caused by an error in the *data* and the second one by converting the result of an exact calculation to floating point form (as in equation (1.6)).

A few examples are provided to show how rounding errors may affect the result of a calculation. Furthermore, general computational rules will be presented regarding the propagation of rounding errors.

Example 1.4.1 (Absolute and relative errors)

In this example, $x = 5/7$ and $y = 1/3$ are taken, and the calculations are computed on a system that uses $\beta = 10$ and a precision of 5 digits. In Table 1.1 the results of various calculations applied to $fl(x) = 0.71429 \times 10^0$ and $fl(y) = 0.33333 \times 10^0$ can be inspected.

Table 1.1: Absolute and relative error for various calculations.

operation	result	exact value	absolute error	relative error
$x + y$	0.10476×10^1	$22/21$	0.190×10^{-4}	0.182×10^{-4}
$x - y$	0.38096×10^0	$8/21$	0.762×10^{-5}	0.200×10^{-4}
$x \times y$	0.23809×10^0	$5/21$	0.524×10^{-5}	0.220×10^{-4}
$x \div y$	0.21429×10^1	$15/7$	0.429×10^{-4}	0.200×10^{-4}

In order to show how the values in de table are computed, the addition is expressed as

$$fl(x) + fl(y) = (0.71429 + 0.33333) \times 10^0 = 0.1047620000\ldots \times 10^1.$$

This result has to be rounded to 5 digits:

$$fl(fl(x) + fl(y)) = 0.10476 \times 10^1.$$

The exact value is $x + y = 22/21 = 1.0476190476\ldots$. Therefore, the absolute error is

$$1.0476190476\ldots - 0.10476 \times 10^1 \approx 0.190 \times 10^{-4},$$

and the relative error is

$$\frac{1.0476190476\ldots - 0.10476 \times 10^1}{1.0476190476\ldots} \approx 0.182 \times 10^{-4}.$$

The error analysis of the other three operations follows the same strategy.

Example 1.4.2 (Absolute and relative errors)

In this example, the same numbers x and y and the same precision as in the previous example are used. Furthermore, $u = 0.714251$, $v = 98765.1$ and $w = 0.111111 \times 10^{-4}$, hence $fl(u) = 0.71425$, $fl(v) = 0.98765 \times 10^5$ and $w = 0.11111 \times 10^{-4}$. These numbers are chosen in such a way that rounding error problems can be clearly illustrated. In Table 1.2, $x - u$ has a small absolute error but a large relative error. By dividing $x - u$ by a small number w or multiplying it with a large number v, the absolute error increases, whereas the relative error is not affected. On the other

Table 1.2: Absolute and relative error for various calculations.

operation	result	exact value	absolute error	relative error
$x-u$	0.40000×10^{-4}	0.34714×10^{-4}	0.529×10^{-5}	0.152
$(x-u)/w$	0.36000×10^{1}	0.31243×10^{1}	0.476	0.152
$(x-u)\times v$	0.39506×10^{1}	0.34286×10^{1}	0.522	0.152
$u+v$	0.98766×10^{5}	0.98766×10^{5}	0.186	0.188×10^{-5}

hand, adding a larger number u to a small number v results in a large absolute error but only a small relative error.

The first row in Table 1.2 is created using the exact results $u = 0.714251$ and

$$x - u = 5/7 - 0.714251 = 0.3471428571\ldots\times10^{-4}.$$

The computed results, however, are $fl(u) = 0.71425\times10^{0}$ and

$$fl(x) - fl(u) = 0.71429 - 0.71425 = 0.0000400000\times10^{0}.$$

Normalization yields $fl(fl(x) - fl(u)) = 0.40000\times10^{-4}$ leading to the absolute error

$$(x-u) - fl(fl(x) - fl(u)) = (0.3471428571\ldots - 0.40000)\times10^{-4} \approx 0.528\times10^{-5}.$$

The relative error is

$$\frac{0.52857\ldots\times10^{-5}}{0.34714\ldots\times10^{-4}} \approx 0.152.$$

It is interesting to note that the large relative error has nothing to do with the limitations of the floating point system (the subtraction of $fl(x)$ and $fl(u)$ is without error in this case) but is only due to the fact that the data are represented in no more than 5 decimal digits. The zeros that remain after normalization in the single precision result $fl(fl(x) - fl(u)) = 0.40000$ have no significance, only the digit 4 is significant; the zeros that are substituted are a mere formality and represent no information. This phenomenon is called *loss of significant digits*. The loss of significant digits has a large impact on the relative error, because of division by the small result.

A large relative error will sooner or later have some unpleasant consequences in later stages of the process, also for the absolute error. For example, if $x - u$ is multiplied by a large number, then a large absolute error is generated, together with the already existing large relative error. This can be seen in the third row of the table, where the exact result is $(x-u)\times v = 3.42855990000460\ldots$. Calculating $fl(fl(x) - fl(u))\times fl(v)$ yields

$$fl(fl(x) - fl(u))\times fl(v) = 0.4\times10^{-4}\times0.98765\times10^{5} = 0.3950600000\times10^{1}.$$

After rounding, the computed result is $fl(fl(fl(x) - fl(u))\times fl(v)) = 0.39506\times10^{1}$. This yields the absolute error

$$3.42855990000460\ldots - 0.39506\times10^{1} \approx 0.522,$$

and the relative error is

$$\frac{0.522\ldots}{3.42855990\ldots} \approx 0.152.$$

Suppose that y^2 is added to the result $(x-u)\times v$. Because $y = 1/3$ and therefore $y^2 = 1/9$, the result of this operation is indistinguishable, due to the large absolute error. In other words, for the reliability of the result it does not make a difference if the last operation had been omitted (and the numerical process was altered). Therefore, something is fundamentally wrong in this case.

Almost all numerical processes exhibit loss of significant digits for a certain set of input data, one might call such a set *ill conditioned*. There are also numerical processes that exhibit these phenomena for all possible input data. Such processes are called *unstable*. One of the objectives of numerical analysis is to identify unstable processes and classify them as useless, or improve them in such a way that they become stable.

Remark
The standard mathematical laws (such as the commutative laws, associative laws, and the distributive law) do not necessarily hold for floating point arithmetic.

Computational rules for error propagation
In the analysis of a complete numerical process, the accumulated error of all previous steps should be interpreted as a perturbation of the original data. Moreover, in the result of the subsequent step, the propagation of these perturbations should be taken into account together with the floating point error. In general, this error source will be more important than the floating point error after a considerable number of steps (in the previous example of $(x - u) \times v$ even after two steps!). In that stage the error in a numerical process will be largely determined by the 'propagation' of the accumulated errors. The computational rules to calculate numerical error propagation are the same as those to calculate propagation of error in measurements in physical experiments. There are two rules: one for addition and subtraction and one for multiplication and division.

The approximations of x and y will be denoted by $\tilde{x}$ and $\tilde{y}$ and the (absolute) perturbations by $\delta x = x - \tilde{x}$ and $\delta y = y - \tilde{y}$.

a) For addition, the error is $(x + y) - (\tilde{x} + \tilde{y}) = (x - \tilde{x}) + (y - \tilde{y}) = \delta x + \delta y$. In other words, the absolute error in the sum of two perturbed terms is equal to the sum of the absolute perturbations. A similar rule holds for differences: $(x - y) - (\tilde{x} - \tilde{y}) = \delta x - \delta y$. The rule is often presented in the form of an inequality: $|(x \pm y) - (\tilde{x} \pm \tilde{y})| \leq |\delta x| + |\delta y|$. We call this inequality an *error estimate*.

b) This rule does not hold for multiplication and division. Efforts to derive a rule for the *absolute* error will lead nowhere, but one may derive a similar rule for the *relative* error. The *relative* perturbations ε_x and ε_y are defined as $\tilde{x} = x(1 + \varepsilon_x)$, and similarly for y. For the relative error in a product xy it holds that

$$\frac{xy - \tilde{x}\tilde{y}}{xy} = \frac{xy - x(1+\varepsilon_x)y(1+\varepsilon_y)}{xy} = -(\varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y) \approx -(\varepsilon_x + \varepsilon_y),$$

assuming ε_x and ε_y are negligible compared to 1. In words: the relative error in a product of two perturbed factors is approximately equal to the sum of the two relative perturbations. A similar rule can be derived for division, where

$$\frac{|xy - \tilde{x}\tilde{y}|}{|xy|} \leq |\varepsilon_x| + |\varepsilon_y|.$$

Identification of $\tilde{x}$ with $fl(x)$ and $\tilde{y}$ with $fl(y)$ makes it possible to explain various phenomena in floating point computations, using these two simple rules.

1.5 Landau's $\mathcal{O}$-symbol

In the analysis of numerical methods estimating the behavior of the error is of primary importance. However, it is often more important to know the growth rate of the error rather than its explicit value which is impossible to compute in many practical problems. Landau's $\mathcal{O}$-symbol is a handy tool to compare the growth rate of two functions in the neighborhood of a point a. For all applications of the Landau $\mathcal{O}$-symbol in this book, $a = 0$, which leads to the following (simplified) definition.

Definition 1.5.1 ($\mathcal{O}$-symbol) *Let f and g be given functions. Then $f(x) = \mathcal{O}(g(x))$ for $x \to 0$, if there exist positive r and finite M such that*

$$|f(x)| \leq M|g(x)| \quad \textit{for all} \quad x \in [-r, r].$$

To estimate errors, the following computational rules are used.

Computational rules
If $f(x) = \mathcal{O}(x^p)$ and $g(x) = \mathcal{O}(x^q)$ for $x \to 0$, with $p \geq 0$ and $q \geq 0$, then

a) $f(x) = \mathcal{O}(x^s)$ for all s with $0 \leq s \leq p$.

b) $\alpha f(x) + \beta g(x) = \mathcal{O}(x^{\min\{p,q\}})$ for all $\alpha, \beta \in \mathbb{R}$.

c) $f(x)g(x) = \mathcal{O}(x^{p+q})$.

d) $f(x)/|x|^s = \mathcal{O}(x^{p-s})$ if $0 \leq s \leq p$.

Example 1.5.1 ($\mathcal{O}$-symbol)

Let $f(x) = x^2$ and $g(x) = x$. In this case:

a) $f(x) = \mathcal{O}(x^2)$ for $x \to 0$, but also $f(x) = \mathcal{O}(x)$ for $x \to 0$.

b) Function $h(x) = x^2 + 3x$ is $\mathcal{O}(x)$ for $x \to 0$ but *not* $\mathcal{O}(x^2)$ for $x \to 0$.

c) $f(x) \cdot g(x)$ is $\mathcal{O}(x^3)$ for $x \to 0$.

d) $f(x)/g(x) = x$ is $\mathcal{O}(x)$ for $x \to 0$.

1.6 Some important concepts and theorems from analysis

In this section some important concepts and theorems from analysis are reviewed that are often used in numerical analysis. Note that $C[a, b]$ is the set of all functions being continuous on the interval $[a, b]$ and $C^p[a, b]$ is the set of all functions of which all derivatives up to the pth exist and are continuous on the interval (a, b).

Definition 1.6.1 (Well posedness) *Mathematical models of physical phenomena are called* well posed *if*

- *A solution exists;*
- *The solution is unique;*
- *The solution's behavior changes continuously with the data.*

Models that do not satisfy these conditions are called ill posed.

Definition 1.6.2 (Well conditioned) *Mathematical models are called* well conditioned *if small errors in the initial data lead to small errors in the results. Models that do not satisfy this condition are called* ill conditioned.

Definition 1.6.3 (Modulus of a complex number) *The modulus of a complex number $a + ib$ is equal to $|a + ib| = \sqrt{a^2 + b^2}$. Note that also $|a - ib| = \sqrt{a^2 + b^2}$.*

Definition 1.6.4 (Kronecker delta) *The Kronecker delta is defined as the following function of two integer variables:*

$$\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

Definition 1.6.5 (Eigenvalues) *Let* $\mathbf{A}$ *be an* $m \times m$ *matrix, and* $\mathbf{I}$ *be the* $m \times m$ *identity matrix. The eigenvalues* $\lambda_1, \ldots, \lambda_m$ *of* $\mathbf{A}$ *satisfy* $det(\mathbf{A} - \lambda\mathbf{I}) = 0$.

Theorem 1.6.1 (Triangle inequality with absolute value) *For any* $x, y \in \mathbb{R}$, $|x + y| \leq |x| + |y|$. *Furthermore, for any function* $f : (a, b) \to \mathbb{R}$, $|\int_a^b f(x)dx| \leq \int_a^b |f(x)|dx$.

Theorem 1.6.2 (Intermediate-value theorem) *Assume* $f \in C[a, b]$. *Let* $f(a) \neq f(b)$ *and let* F *be a number between* $f(a)$ *and* $f(b)$. *Then there exists a number* $c \in (a, b)$ *such that* $f(c) = F$.

Theorem 1.6.3 (Rolle's theorem) *Assume* $f \in C[a, b]$ *and* f *differentiable on* (a, b). *If* $f(a) = f(b)$, *then there exists a number* $c \in (a, b)$ *such that* $f'(c) = 0$.

Theorem 1.6.4 (Mean-value theorem) *Assume* $f \in C[a, b]$ *and* f *differentiable on* (a, b), *then there exists a number* $c \in (a, b)$ *such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Theorem 1.6.5 (Mean-value theorem for integration) *Assume* $G \in C[a, b]$ *and* φ *is an integrable function that does not change sign on* $[a, b]$. *Then there exists a number* $x \in (a, b)$ *such that*

$$\int_a^b G(t)\varphi(t)dt = G(x)\int_a^b \varphi(t)dt.$$

Theorem 1.6.6 (Weierstrass approximation theorem) *Suppose* $f \in C[a, b]$ *is given. For every* $\varepsilon > 0$, *there exists a polynomial function* p *such that for all* x *in* $[a, b]$, $|f(x) - p(x)| < \varepsilon$.

Theorem 1.6.7 (Taylor polynomial) *Assume* $f \in C^{n+1}(a, b)$ *is given. Then for all* $c, x \in (a, b)$ *there exists a number* ξ *between* c *and* x *such that*

$$f(x) = P_n(x) + R_n(x),$$

in which the Taylor polynomial $P_n(x)$ *is given by*

$$P_n(x) = f(c) + (x - c)f'(c) + \frac{(x - c)^2}{2!}f''(c) + \ldots + \frac{(x - c)^n}{n!}f^{(n)}(c),$$

and the remainder term $R_n(x)$ *is:*

$$R_n(x) = \frac{(x - c)^{n+1}}{(n + 1)!}f^{(n+1)}(\xi).$$

Proof
Take $c, x \in (a, b)$ with $c \neq x$ and let K be defined as

$$f(x) = f(c) + (x - c)f'(c) + \frac{(x - c)^2}{2!}f''(c) + \ldots + \frac{(x - c)^n}{n!}f^{(n)}(c) + K(x - c)^{n+1}. \tag{1.8}$$

Consider the function

$$F(t) = f(t) - f(x) + (x - t)f'(t) + \frac{(x - t)^2}{2!}f''(t) + \ldots + \frac{(x - t)^n}{n!}f^{(n)}(t) + K(x - t)^{n+1}.$$

By (1.8) it follows that $F(c) = 0$ and $F(x) = 0$. Hence, by Rolle's theorem there exists a number ξ between c and x such that $F'(\xi) = 0$. Further elaboration yields

$$\begin{aligned} F'(\xi) &= f'(\xi) + \{f''(\xi)(x - \xi) - f'(\xi)\} + \left\{\frac{f'''(\xi)}{2!}(x - \xi)^2 - f''(\xi)(x - \xi)\right\} + \ldots \\ &+ \left\{\frac{f^{(n+1)}(\xi)}{n!}(x - \xi)^n - \frac{f^{(n)}(\xi)}{(n - 1)!}(x - \xi)^{(n-1)}\right\} - K(n + 1)(x - \xi)^n \\ &= \frac{f^{(n+1)}(\xi)}{n!}(x - \xi)^n - K(n + 1)(x - \xi)^n = 0. \end{aligned}$$

Hence,

$$K = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

which proves the theorem. ⊠

Theorem 1.6.8 (Taylor polynomial of two variables) *Let $f : D \subset \mathbb{R}^2 \mapsto \mathbb{R}$ be continuous with continuous partial derivatives up to and including order $n+1$ in a ball $B \subset D$ with center $\mathbf{c} = (c_1, c_2)$ and radius ρ: $B = \{(x_1, x_2) \in \mathbb{R}^2 : \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2} < \rho\}$. Then for each $\mathbf{x} = (x_1, x_2) \in B$ there exists a $\theta \in (0, 1)$, such that*

$$f(\mathbf{x}) = P_n(\mathbf{x}) + R_n(\mathbf{x}),$$

in which the Taylor polynomial $P_n(\mathbf{x})$ is given by

$$P_n(\mathbf{x}) = f(\mathbf{c}) + (x_1 - c_1)\frac{\partial f}{\partial x_1}(\mathbf{c}) + (x_2 - c_2)\frac{\partial f}{\partial x_2}(\mathbf{c}) + \frac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{2}(x_i - c_i)(x_j - c_j)\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{c}) + \ldots$$

$$+\frac{1}{n!}\sum_{i_1=1}^{2}\sum_{i_2=1}^{2}\ldots\sum_{i_n=1}^{2}(x_{i_1} - c_{i_1})(x_{i_2} - c_{i_2})\ldots(x_{i_n} - c_{i_n})\frac{\partial^n f}{\partial x_{i_1}\partial x_{i_2}\ldots\partial x_{i_n}}(\mathbf{c}),$$

and the remainder term is

$$R_n(\mathbf{x}) = \frac{1}{(n+1)!}\sum_{i_1=1}^{2}\ldots\sum_{i_{n+1}=1}^{2}(x_{i_1} - c_{i_1})\ldots(x_{i_{n+1}} - c_{i_{n+1}})\frac{\partial^{n+1} f}{\partial x_{i_1}\ldots\partial x_{i_{n+1}}}(\mathbf{c} + \theta(\mathbf{x} - \mathbf{c})).$$

Proof
Let $\mathbf{x} \in B$ be fixed, and $\mathbf{h} = \mathbf{x} - \mathbf{c}$, hence $\|\mathbf{h}\| < \rho$. Define the function $F : (-1, 1) \mapsto \mathbb{R}$ by:

$$F(s) = f(\mathbf{c} + s\mathbf{h}).$$

Because of the differentiability conditions satisfied by f in the ball B, $F \in C^{n+1}(-1, 1)$ and $F^k(s)$ is given by (verify)

$$F^k(s) = \sum_{i_1=1}^{2}\sum_{i_2=1}^{2}\ldots\sum_{i_k=1}^{2}\frac{\partial^k f(\mathbf{c} + s\mathbf{h})}{\partial x_{i_1}\partial x_{i_2}\ldots\partial x_{i_k}}h_{i_1}h_{i_2}\ldots h_{i_k}.$$

Expand F into a Taylor polynomial about 0. This yields

$$F(s) = F(0) + sF'(0) + \ldots + \frac{s^n}{n!}F^n(0) + \frac{s^{n+1}}{(n+1)!}F^{n+1}(\theta s),$$

for some $\theta \in (0, 1)$. By substituting $s = 1$ into this expression and into the expressions for the derivatives of F, the result follows. ⊠

Example 1.6.1 (Taylor polynomial of two variables)

For $n = 1$, the Taylor polynomial equals

$$P_1(\mathbf{x}) = f(c_1, c_2) + (x_1 - c_1)\frac{\partial f}{\partial x_1}(c_1, c_2) + (x_2 - c_2)\frac{\partial f}{\partial x_2}(c_1, c_2),$$

and for the remainder term: $R_1(\mathbf{x})$ is $\mathcal{O}(\|\mathbf{x} - \mathbf{c}\|^2)$.

Theorem 1.6.9 (Power series of $1/(1-x)$) *Let $x \in \mathbb{R}$ with $|x| < 1$. Then:*

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots = \sum_{k=0}^{\infty} x^k.$$

Theorem 1.6.10 (Power series of e^x) *Let $x \in \mathbb{R}$. Then:*

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \ldots = \sum_{k=0}^{\infty} \frac{x^k}{k!}.$$

1.7 Summary

In this chapter, the following subjects have been discussed:

- Numerical mathematics;
- Rounding errors;
- Landau's $\mathcal{O}$-symbol;
- Some important concepts and theorems from analysis.

1.8 Exercises

1. Let $f(x) = x^3$. Determine the second-order Taylor polynomial of f about the point $x = 1$. Compute the value of this polynomial in $x = 0.5$. Give an upper bound for the error and compare this with the actual error.

2. Let $f(x) = e^x$. Give the nth order Taylor polynomial about the point $x = 0$, and the remainder term. How large should n be chosen in order to make the error less than 10^{-6} in the interval $[0,\ 0.5]$?

3. We use the polynomial $P_2(x) = 1 - x^2/2$ to approximate $f(x) = \cos x$ in the interval $[-1/2,\ 1/2]$. Give an upper bound for the error in this approximation.

4. Let $x = 1/3,\ y = 5/7$. We calculate with a precision of three (decimal) digits. Express x and y as floating point numbers. Compute $fl(fl(x) \circ fl(y)),\ x \circ y$ and the rounding error taking $\circ = +,\ -,\ *,\ /$ respectively.

5. Write a program to determine the machine precision of your system.
   Hint: start with $\mu = 1$. Divide μ repeatedly by a factor 2 until the test $1 + \mu = 1$ holds. The resulting value is close to the machine precision of your system.

Chapter 2

Interpolation

2.1 Introduction

In practical applications it is frequently the case that only a limited amount of measurement data is available, from which intermediate values should be determined (interpolation) or values outside the range of measurements should be predicted (extrapolation). An example is the number of cars in The Netherlands, which is tabulated in Table 2.1 (per 1000 citizens) from 1990 every fifth year up to 2010. How can these numbers be used to estimate the number of cars in intermediate years, e.g. in 2008, or to predict the number in the year 2020?

Table 2.1: Number of cars (per 1000 citizens) in The Netherlands (source: Centraal Bureau voor de Statistiek).

year	1990	1995	2000	2005	2010
number	344	362	400	429	460

In this chapter several interpolation and extrapolation methods will be considered.

A second example of the use of interpolation techniques is image visualization. By only storing a limited number of pixels, much memory can be saved. In that case, an interpolation curve should be constructed in order to render a more or less realistic image on the screen.

A final application is the computation of trigonometric function values on a computer, which is time consuming. However, if a number of precalculated function values is stored in memory, the values at intermediate points can be determined from these in a cheap way.

2.2 Linear interpolation

The simplest way to interpolate is zeroth degree (constant) interpolation. Suppose the function value at a certain point is known. Then, an approximation in the neighborhood of this point is set equal to this known value. A well known example is the prediction that tomorrow's weather will be the same as today's. This prediction appears to be correct in 80% of all cases (and in the Sahara this percentage is even higher).

A better way of interpolation is a straight line between two points (see Figure 2.1). Suppose that the function values $f(x_0)$ and $f(x_1)$ at the points x_0 and x_1 are known ($x_0 < x_1$). In that case, it seems plausible to take the linear function (a straight line) through $(x_0, f(x_0))$, $(x_1, f(x_1))$, in

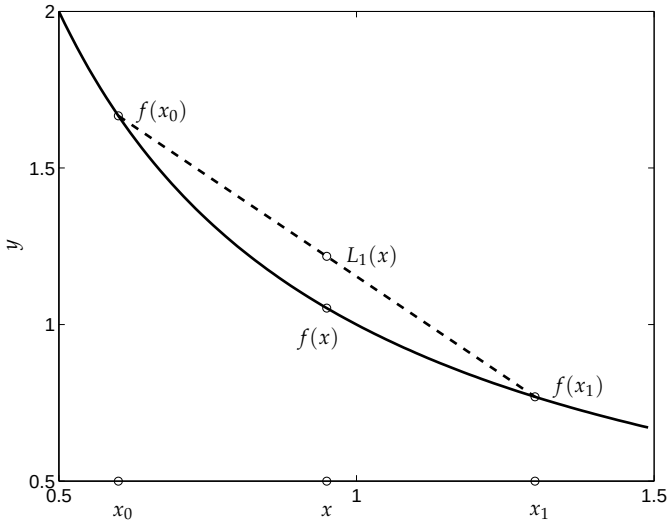


Figure 2.1: Linear interpolation polynomial, L_1, of the function $f(x) = 1/x$, $x_0 = 0.6$, $x = 0.95$, $x_1 = 1.3$.

order to approximate[1] $f(x), x \in [x_0, x_1]$. Such a function is called a *linear interpolation polynomial*, and can be expressed as

$$L_1(x) = f(x_0) + \frac{x - x_0}{x_1 - x_0}(f(x_1) - f(x_0)). \tag{2.1}$$

The mathematical derivation of this formula uses the following properties:

1. L_1 is linear, hence it can be written as $L_1(x) = c_0 + c_1 x$;
2. L_1 interpolates the given data, hence it holds that $L_1(x_0) = f(x_0)$ and $L_1(x_1) = f(x_1)$. This is called the *interpolation property*.

These properties lead to the following linear system of equations:

$$\begin{pmatrix} 1 & x_0 \\ 1 & x_1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \end{pmatrix}, \tag{2.2}$$

from which the unknown coefficients c_0 and c_1 can be determined.

The need to solve (2.2) can be overcome by making use of the so-called *Lagrange basis polynomials* $L_{i1}(x), i = 0, 1$. $L_{01}(x)$ and $L_{11}(x)$ are defined as linear polynomials such that

$$L_{01}(x_0) = 1,\ L_{01}(x_1) = 0,\ L_{11}(x_0) = 0,\ L_{11}(x_1) = 1.$$

From these properties it follows that

$$L_{01}(x) = \frac{x - x_1}{x_0 - x_1}, \quad \text{and} \quad L_{11}(x) = \frac{x - x_0}{x_1 - x_0}\,.$$

The linear interpolation polynomial $L_1(x)$ can be written as

$$L_1(x) = L_{01}(x) f(x_0) + L_{11}(x) f(x_1),$$

[1] Whenever the word 'approximation' occurs in this chapter, we mean that the unknown function value is approximated using an interpolation polynomial.

which is called the Lagrange form of the interpolation polynomial, or the *Lagrange interpolation* polynomial for short. The interpolation points x_0 and x_1 are often called *nodes*.

Theorem 2.2.1 provides a statement for the interpolation error if the interpolated function is at least twice continuously differentiable.

Theorem 2.2.1 *Let x_0 and x_1 be distinct nodes in $[a, b]$, $x_0 < x_1$, $f \in C^2[a, b]$ and let L_1 be the linear interpolation polynomial that equals f at the nodes x_0, x_1. Then for each $x \in [a, b]$ there exists a $\xi \in (a, b)$ such that*

$$f(x) - L_1(x) = \frac{1}{2}(x - x_0)(x - x_1)f''(\xi). \tag{2.3}$$

Proof
If $x = x_0$ or $x = x_1$, then $f(x) - L_1(x) = 0$ and ξ can be chosen arbitrarily. Assume $x \neq x_0$ and $x \neq x_1$. For each x there exists a constant q such that

$$f(x) - L_1(x) = q(x - x_0)(x - x_1).$$

To find an expression for q consider the function

$$\varphi(t) = f(t) - L_1(t) - q(t - x_0)(t - x_1).$$

φ satisfies $\varphi(x_0) = \varphi(x_1) = \varphi(x) = 0$. By Rolle's theorem, there exist at least two points y and z in (a, b) such that $\varphi'(y) = \varphi'(z) = 0$. Again by Rolle's theorem there is a ξ between y and z and hence $\xi \in (a, b)$ such that $\varphi''(\xi) = 0$. Because $\varphi''(t) = f''(t) - 2q$ this means that $q = \frac{1}{2}f''(\xi)$. ☒

If $x \notin [x_0, x_1]$, then the linear polynomial (2.1) can be used to extrapolate. Relation (2.3) is still the correct expression for the error.

From Theorem 2.2.1, an upper bound for the linear interpolation error follows:

$$|f(x) - L_1(x)| \leq \frac{1}{8}(x_1 - x_0)^2 \max_{\xi \in (a,b)} |f''(\xi)|. \tag{2.4}$$

In many practical applications the values $f(x_0)$ and $f(x_1)$ are a result of measurements (perturbed) or calculations (round-off errors). Assume that the measurement error is bounded by ε, that is, $|f(x_0) - \hat{f}(x_0)| \leq \varepsilon$ and $|f(x_1) - \hat{f}(x_1)| \leq \varepsilon$, where the $\hat{}$ (hat) refers to the measured, hence available data. The difference between the exact linear interpolation polynomial L_1 and the perturbed polynomial $\hat{L}_1$ is bounded by

$$|L_1(x) - \hat{L}_1(x)| \leq \frac{|x_1 - x| + |x - x_0|}{x_1 - x_0}\varepsilon. \tag{2.5}$$

For $x \in [x_0, x_1]$ (interpolation), the error resulting from uncertainties in the measurements is bounded by ε.

However, if $x \notin [x_0, x_1]$ (extrapolation), the error can grow to arbitrarily large values when moving away from the interval $[x_0, x_1]$. Suppose that $x \geq x_1$, then the additional inaccuracy is bounded by

$$|L_1(x) - \hat{L}_1(x)| \leq \left(1 + 2\frac{x - x_1}{x_1 - x_0}\right)\varepsilon. \tag{2.6}$$

In Figure 2.2, the impact of the rounding or measurement errors on linear interpolation and extrapolation is visualized. It graphically illustrates the danger of extrapolation, where the region of uncertainty, and hence, the errors due to uncertainties in the data, make the extrapolation error large. Note that the truncation error in equation (2.3) also becomes arbitrarily large outside the interval $[x_0, x_1]$.

The total error is the sum of the interpolation/extrapolation error and the measurement error.

Example 2.2.1 (Linear interpolation)

In Table 2.2, the value of the sine function is presented for 36° and 38°. The approximation for 37° using the linear interpolation polynomial provides a result of 0.601723. The difference with the exact value is only 0.9×10^{-4}.

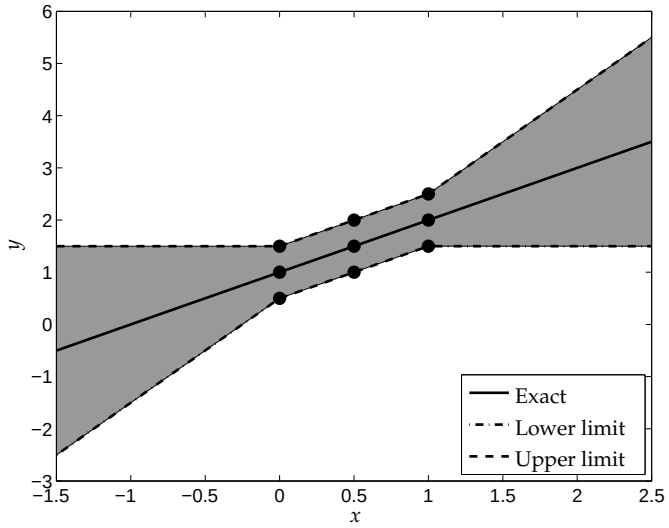


Figure 2.2: Interpolation and extrapolation errors resulting from measurement errors, using, as an example, $x_0 = 0, x_1 = 1, f(x_0) = 1, f(x_1) = 2$, and $\varepsilon = 1/2$. The region of uncertainty is filled.

Table 2.2: The value of $\sin \alpha$.

α	$\sin \alpha$
36°	0.58778525
37°	0.60181502
38°	0.61566148

2.3 Higher-order Lagrange interpolation

Suppose that $n + 1$ function values are provided. Instead of using linear interpolation, $f(x)$ can be approximated by a polynomial $L_n(x)$ of degree at most n, such that the values of f and L_n at the $n + 1$ different points $x_0, \ldots, x_n$ coincide. L_n is called an *nth degree Lagrange interpolation polynomial.*

The polynomial $L_n(x)$ that satisfies these constraints has the following properties:

1. $L_n(x)$ is a polynomial of degree n, hence it can be written as

$$L_n(x) = \sum_{k=0}^{n} c_k x^k.$$

2. The values at the interpolation points coincide: $L_n(x_j) = f(x_j),\ j = 0, \ldots, n$.

These properties lead to the following linear system:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

from which the unknown coeffients $c_i, i = 0, \ldots, n$, can be determined. This matrix, in which each row consists of the terms of a geometric series, is called a *Vandermonde matrix*.

It is again possible to avoid the linear system of equations by using Lagrange basis polynomials. To this end, the nth degree interpolation polynomial is written as

$$L_n(x) = \sum_{k=0}^{n} f(x_k) L_{kn}(x), \tag{2.7}$$

in which the nth degree Lagrange basis polynomials satisfy $L_{kn}(x_j) = \delta_{kj}$ in the nodes x_j (see definition 1.6.4). The *Lagrange basis polynomials* $L_{kn}(x)$ are explicitly given by

$$L_{kn}(x) = \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}, \tag{2.8}$$

and can also be written as:

$$L_{kn}(x) = \frac{\omega(x)}{(x - x_k)\omega'(x_k)} \quad \text{with} \quad \omega(x) = \prod_{i=0}^{n} (x - x_i).$$

Theorem 2.2.1 can be generalized:

Theorem 2.3.1 *Let $x_0, \ldots, x_n$ be distinct nodes in $[a, b]$, $x_0 < x_1 < \cdots < x_n$. Let $f \in C^{n+1}[a, b]$ and let L_n be the Lagrange polynomial that equals f at these nodes. Then for each $x \in [a, b]$ there exists a $\xi \in (a, b)$ such that*

$$f(x) - L_n(x) = (x - x_0) \cdots (x - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Proof:
The proof is completely analogous to that of Theorem 2.2.1. ⊠

If Lagrange interpolation is used on tabulated values, then the best results are obtained by choosing the nodes in such a way that x is in the (or an) innermost interval.

Suppose that measurement errors are made, and $|f(x) - \hat{f}(x)| \leq \varepsilon$. Then the error in the perturbed interpolation polynomial is at most

$$|L_n(x) - \hat{L}_n(x)| \leq \sum_{k=0}^{n} |L_{kn}(x)| \varepsilon.$$

If the nodes are equidistant, $x_k = x_0 + kh$, with $h = (x_n - x_0)/n$, then the value of $\sum_{k=0}^{n} |L_{kn}(x)|$ increases slowly with n. A number of upper bounds is given in Table 2.3.

Table 2.3: Upper bounds for $\sum_{k=0}^{n} |L_{kn}(x)|$.

	$x \in [x_0, x_1]$	$x \in [x_1, x_2]$	$x \in [x_2, x_3]$	$x \in [x_3, x_4]$	$x \in [x_4, x_5]$
$n = 1$	1				
$n = 2$	1.25	1.25			
$n = 3$	1.63	1.25	1.63		
$n = 4$	2.3	1.4	1.4	2.3	
$n = 5$	3.1	1.6	1.4	1.6	3.1

Assume that the interpolation points are equidistantly distributed on an interval $[a, b]$. One might wonder whether it is possible to approximate a provided (sufficiently smooth) function arbitrarily close using a Lagrange interpolation polynomial, by increasing the degree n of the interpolation polynomial. Surprisingly enough this is not the case. A well known example of this was suggested by Runge.

Example 2.3.1 (Runge's phenomenon) *

Consider the function $1/(1+x^2)$ on the interval $[-5,5]$, with nodes $x_k = -5 + 10k/n$, where $k = 0,\ldots,n$. In Figure 2.3(a) the function is visualized together with the 6th and 14th degree interpolation polynomials. Note that the 14th degree polynomial approximation is better than that of degree 6 on the interval [-3, 3]. In the neighborhood of the end points, however, the 14th degree polynomial exhibits large oscillations.

The oscillations of the interpolation polynomial near the boundary of the interval are known as Runge's phenomenon. The above example leads to the question whether Runge's phenomenon can be avoided. In order to answer this question, the expression for the interpolation error in Theorem 2.3.1 can be considered. This expression contains two factors that can become large: the unknown factor $f^{(n+1)}(\xi)$ and the polynomial factor $(x-x_0)\cdots(x-x_n)$. Clearly, $f^{(n+1)}(\xi)$ cannot be influenced, but the polynomial $(x-x_0)\cdots(x-x_n)$ can be made small in some sense by choosing better interpolation points $x_i, i = 0,\ldots,n$. More specifically, the interpolation points should be chosen such that

$$\max_{x\in[a,b]} |(x-x_0)\cdots(x-x_n)|$$

is minimized over all possible choices for the interpolation points $x_0,\ldots,x_n$. This problem was solved in the 19th century by the Russian mathematician Chebyshev, who found that the optimal interpolation points are given by

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{2k+1}{2(n+1)}\pi\right) \quad k = 0,\ldots,n,$$

which are called *Chebyshev nodes*. In Figure 2.3(b), it can be seen that the interpolation polynomials using Chebyshev nodes provide much better approximations compared to the equidistantly distributed ones in Figure 2.3(a). It can be shown that by using Chebyshev nodes, a continuously differentiable function can be approximated with arbitrarily small error by increasing the degree n of the interpolation polynomial.

Example 2.3.2 (Extrapolation)

Large errors may also occur in extrapolation. Consider the function $f(x) = 1/x$. The nth degree interpolation polynomial is determined with nodes $x_k = 0.5 + k/n,\ k = 0,\ldots,n$. In Figure 2.4, $f(x)$ and the interpolation polynomials of degree 6 and 10 are shown on the interval $[0.5, 1.8]$. The polynomials and the function itself are indistinguishable on the interval [0.5, 1.5]. With extrapolation $(x \geq 1.5)$, however, large errors occur. Again the largest error occurs in the highest-degree polynomial.

2.4 Interpolation with function values and derivatives *

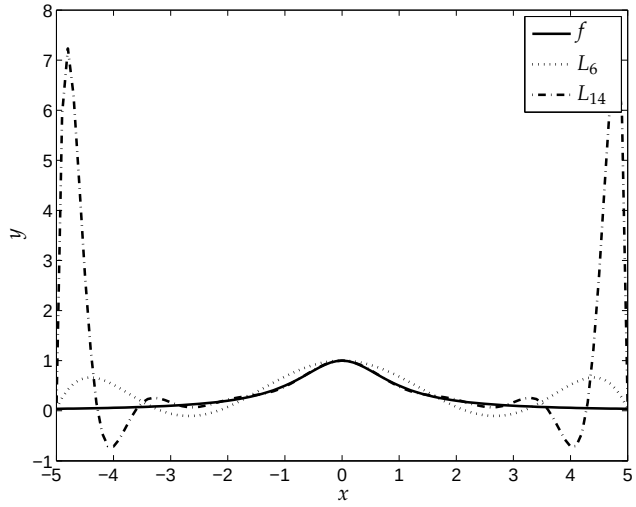
2.4.1 Taylor polynomial

A well known method to approximate a function by a polynomial is Taylor's method. As an approximation method it is not used very often in numerical mathematics, but instead it is used quite often for the analysis of numerical processes.

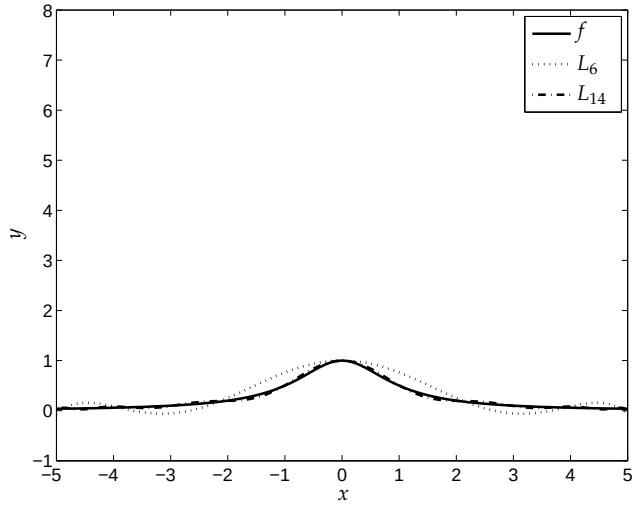
In many cases the approximation of the Taylor polynomial becomes better with increasing polynomial degree. However, this is not always true. This is shown in the following example:

Example 2.4.1 (Taylor polynomial)

The function $f(x) = 1/x$ is approximated in $x = 3$ by a Taylor polynomial of degree n about $x = 1$ (note that $x = 3$ is outside the region of convergence of the Taylor series). The derivatives



(a) Equidistant nodes



(b) Chebyshev nodes

Figure 2.3: Interpolation of $f(x) = 1/(1+x^2)$ with Lagrange polynomials of degree 6 and 14.

are given by: $f^{(k)}(x) = (-1)^k k! x^{-(k+1)}$, such that the nth degree Taylor polynomial about $x = 1$ equals

$$p_n(x) = \sum_{k=0}^{n} f^{(k)}(1)(x-1)^k/k! = \sum_{k=0}^{n} (-1)^k (x-1)^k.$$

The values of $p_n(3)$ that should approximate $f(3) = 1/3$ are tabulated in Table 2.4. The approximation becomes less accurate with increasing polynomial degree.

Table 2.4: Approximation of $f(x) = 1/x$ in $x = 3$, using an nth degree Taylor polynomial about $x = 1$.

n	0	1	2	3	4	5	6	7
$p_n(3)$	1	-1	3	-5	11	-21	43	-85

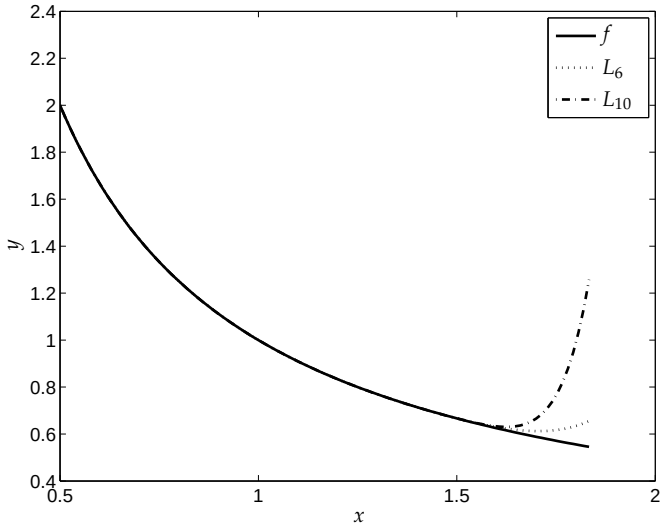


Figure 2.4: Extrapolation of $f(x) = 1/x$ with Lagrange polynomials of degree 6 and 10.

2.4.2 Interpolation in general

In general, given a function f, a polynomial p is constructed such that at a number of pairwise distinct nodes $x_0, \ldots, x_n$ not only the values of f and p coincide, but also the value of their derivatives up to and including order m_k at node x_k, $k = 0, \ldots, n$.

Suppose that $f \in C^m[a,b]$ with $m = \max\limits_{0 \le k \le n} m_k$. Then p is the polynomial of lowest degree such that

$$\frac{d^j p}{dx^j}(x_k) = \frac{d^j f}{dx^j}(x_k) \quad \text{for each} \quad k = 0, \ldots, n \quad \text{and} \quad j = 0, \ldots, m_k.$$

Remarks

- This polynomial is at most of degree $M = \sum\limits_{k=0}^{n} m_k + n$.
- If $n = 0$, then p is the Taylor polynomial of degree m_0 about x_0.
- If $m_0 = \ldots = m_n = 0$, then p is the nth degree Lagrange polynomial of f in the nodes $x_0, \ldots, x_n$.

As an example, Section 2.4.3 deals with the case $m_0 = \ldots = m_n = 1$. The resulting polynomials are called *Hermite interpolation polynomials*.

2.4.3 Hermite interpolation

If both function values and its first derivatives are known at certain points, these data can be used to construct an approximation polynomial of a higher degree.

For instance, assume that the function values and the value of the first derivative at two different points are known, that is,

$$\begin{array}{ll} (x_0, f(x_0)), & (x_1, f(x_1)), \\ (x_0, f'(x_0)), & (x_1, f'(x_1)) \end{array}$$

are provided. Using these four independent values, a third degree polynomial, H_3, can be constructed, which satisfies

$$\begin{array}{ll} H_3(x_j) = f(x_j), & j = 0, 1, \\ H_3'(x_j) = f'(x_j), & j = 0, 1. \end{array}$$

Writing $H_3(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$ leads to the following system of linear equations:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 0 & 1 & 2x_0 & 3x_0^2 \\ 0 & 1 & 2x_1 & 3x_1^2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ f'(x_0) \\ f'(x_1) \end{pmatrix}. \tag{2.9}$$

In the same spirit as for Lagrange interpolation, this polynomial can be written as

$$H_3(x) = \sum_{k=0}^{1} \left(f(x_k) H_{k1}(x) + f'(x_k) \mathcal{H}_{k1}(x) \right),$$

with

$$\begin{aligned} H_{k1}(x_j) &= \delta_{jk}, & \mathcal{H}_{k1}(x_j) &= 0, \\ H'_{k1}(x_j) &= 0, & \mathcal{H}'_{k1}(x_j) &= \delta_{jk}. \end{aligned}$$

This avoids having to solve system (2.9). Polynomials based both on given function values and given derivatives are called *Hermite interpolation polynomials*.

The general expression for Hermite interpolation polynomials containing function values and first derivatives is

$$H_{2n+1}(x) = \sum_{k=0}^{n} f(x_k) H_{kn}(x) + \sum_{k=0}^{n} f'(x_k) \mathcal{H}_{kn}(x),$$

in which

$$H_{kn}(x) = \left(1 - 2(x - x_k) L'_{kn}(x_k)\right) L_{kn}^2(x),$$

and

$$\mathcal{H}_{kn}(x) = (x - x_k) L_{kn}^2(x).$$

The functions $L_{kn}(x)$ are the Lagrange basis polynomials, as defined in equation (2.8).

The correctness of the Hermite interpolation polynomial is shown as follows: H_{kn} and $\mathcal{H}_{kn}$ are polynomials of degree $2n+1$. Note that $H_{kn}(x_j) = 0$ if $k \neq j$, because in that case, $L_{kn}(x_j) = 0$. If $k = j$, then, because $L_{kn}(x_k) = 1$,

$$H_{kn}(x_j) = H_{kn}(x_k) = \left((1 - 2(x_k - x_k) L'_{kn}(x_k)\right) L_{kn}^2(x_k) = 1.$$

Because $\mathcal{H}_{kn}(x_j) = 0$ for all $j = 0, \ldots, n$ it follows that $H_{2n+1}(x_j) = f(x_j)$.

Next, we have to show that the derivatives coincide at the nodes. Therefore, H'_{kn} is calculated:

$$H'_{kn}(x) = -2L'_{kn}(x_k) L_{kn}^2(x) + \left(1 - 2(x - x_k) L'_{kn}(x_k)\right) 2L_{kn}(x) L'_{kn}(x).$$

For all $j = 0, \ldots, n$, $H'_{kn}(x_j) = 0$. The derivative of $\mathcal{H}_{kn}$ is given by

$$\mathcal{H}'_{kn}(x) = L_{kn}^2(x) + 2(x - x_k) L_{kn}(x) L'_{kn}(x).$$

Hence $\mathcal{H}'_{kn}(x_j) = \delta_{jk}$ and therefore $H'_{2n+1}(x_j) = f'(x_j)$.

Indeed, the Hermite interpolation polynomial satisfies the conditions $H_{2n+1}(x_j) = f(x_j)$ and $H'_{2n+1}(x_j) = f'(x_j)$, $j = 0, \ldots, n$.

Theorem 2.4.1 *Let $x_0, \ldots, x_n$ be distinct nodes in $[a, b]$, $x_0 < x_1 < \cdots < x_n$. Let $f \in C^{2n+2}[a, b]$ and let H_{2n+1} be the Hermite interpolation polynomial that equals f and f' at these nodes. Then for each $x \in [a, b]$ there exists a $\xi \in (a, b)$ such that*

$$f(x) - H_{2n+1}(x) = \frac{(x - x_0)^2 \cdots (x - x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi).$$

Proof:
The proof of this theorem is analogous to that of Theorem 2.2.1. The auxiliary function is chosen as follows:

$$\varphi(t) = f(t) - H_{2n+1}(t) - \frac{(t-x_0)^2 \cdots (t-x_n)^2}{(x-x_0)^2 \cdots (x-x_n)^2}(f(x) - H_{2n+1}(x)).$$

It can be proved that $\varphi'(t)$ has $2n+2$ different zeros in (a,b). ⊠

The choice for Hermite interpolation instead of Lagrange interpolation is clarified in the following two examples.

Example 2.4.2 (Seismic waves)

Seismic waves, the same type of waves used to study earthquakes, are also used to explore deep underground for reservoirs of oil and natural gas. A simple model for wave propagation is described by the following set of ordinary differential equations:

$$\begin{cases} \frac{dx}{dt} &= c\sin\theta, \\ \frac{dz}{dt} &= -c\cos\theta, \\ \frac{d\theta}{dt} &= -\frac{dc}{dz}\sin\theta. \end{cases}$$

The position is denoted by (x,z) while θ is the angle between wave front and x-axis. Suppose that the propagation speed c depends on the vertical position only and that it is known at a finite number of positions. However, in order to solve this system an approximation of $c(z)$ in all intermediate points is required. If piecewise linear interpolation is used, the derivative $c'(z)$ does not exist in the nodes, which may lead to large errors in the solution. If both c and $c'(z)$ are known in the nodes, then the third degree Hermite interpolation polynomial can be used in each interval and the first derivative exists also at the nodes.

Example 2.4.3 (Visualization)

Suppose a finite number of points of a figure is known and a smooth curve should be drawn through them. Because a piecewise linear interpolation polynomial is not smooth in the nodes, this often leads to an unrealistic representation. A better result is obtained using Hermite interpolation.

Suppose the graph of the function $f(x) = 1/(1+x^3),\ x \in [0,4]$, is used to visualize half of a symmetric hill. In Figure 2.5 the graph is approximated with several interpolation polynomials. The resulting curve using piecewise linear polynomials does not resemble a hill even remotely. Hermite interpolation on the same nodes provides a much better result. However, this is not an honest comparison because third degree Hermite interpolation uses twice as many input values. Therefore, the third degree Lagrange interpolation on the intervals [0,2] and [2,4] is considered. Note that on the interval [2,4] the function and the polynomial almost coincide. However, due to the large jump in the derivative at $x = 2$, this result is useless.

2.5 Interpolation with splines

In the previous sections, it has been demonstrated that the approximation of a function using an interpolation polynomial can cause some problems. It is often better to divide the interpolation interval into subintervals and to construct an interpolation polynomial on each subinterval. One problem with this approach is the lack of smoothness of the interpolation polynomial at the interface of two subintervals. Hermite interpolation is a remedy for this problem, but for this to work one has to know the derivative at the interface. If the data are gathered from measurements, then

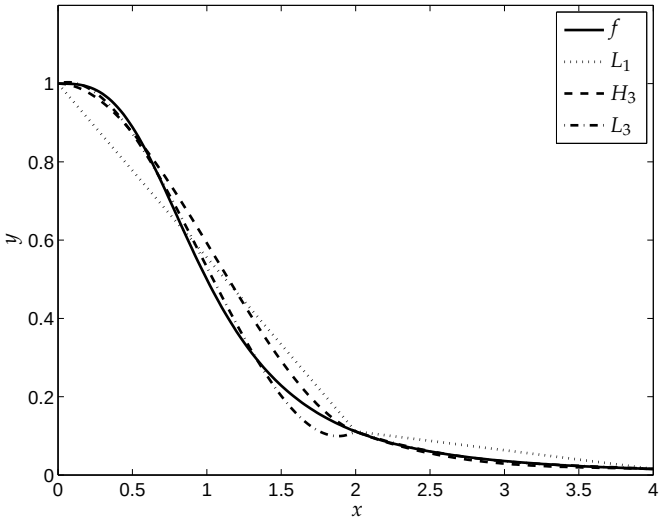


Figure 2.5: Interpolation of $f(x) = 1/(1+x^3)$ using two intervals: linear, Hermite (degree 3) and Lagrange (degree 3) interpolation.

the derivative at the nodes are often unknown. One way to ensure smoothness without knowing the derivatives is to use so-called splines.

A *spline* is a piecewise polynomial that is connected smoothly in the nodes. We define the nodes $a = x_0 < x_1 < \cdots < x_n = b$, which partition the interval $[a, b]$. A spline of degree p is defined as the piecewise function

$$s(x) = \begin{cases} s_0(x), & x \in [x_0, x_1], \\ s_1(x), & x \in [x_1, x_2], \\ \vdots & \\ s_{n-1}(x), & x \in [x_{n-1}, x_n], \end{cases}$$

where $s_k(x)|_{[x_k, x_{k+1}]}$ is a polynomial of degree p and a smooth connection of s_k and its derivatives, $s_k', \ldots, s_k^{(p-1)}$, is required in the nodes.

In this section, only linear ($p = 1$) and cubic ($p = 3$) interpolation splines are considered.

Definition 2.5.1 (Spline of degree 1) *Let $x_0, \ldots, x_n$ be distinct nodes in $[a, b]$, $x_0 < x_1 < \cdots < x_n$. For $f \in C[a, b]$, the interpolation spline s of degree 1 is a function $s \in C[a, b]$ such that*

a. On each subinterval $[x_k, x_{k+1}]$, s is linear, $k = 0, \ldots, n-1$,

b. The values at the nodes satisfy $s(x_k) = f(x_k)$, for $k = 0, \ldots, n$.

Note that an interpolation spline of degree 1 consists of piecewise linear interpolation polynomials.
A spline of degree 3, also called a *cubic spline*, consists of polynomials of degree 3. In the nodes the value is equal to the function value, and further, the first and second derivatives are continuous. Splines were originally developed for shipbuilding in the days before computer modeling. Naval architects needed a way to draw a smooth curve through a set of points. The solution was to place metal weights (called *knots* or *ducks*) at the control points and bend a thin metal or wooden beam (called a *spline*) through the weights. The physics of the bending spline meant that the influence of each weight was largest at the point of contact and decreased smoothly further along the spline. To get more control over a certain region of the spline the draftsman simply added more weights.

Definition 2.5.2 (Natural spline of degree 3) *Let the nodes $x_0, \ldots, x_n$ be distinct in $[a, b]$ and satisfy $x_0 < x_1 < \cdots < x_n$. For a function $f \in C[a, b]$, the interpolating spline s of degree 3 has the following properties:*

a. On each subinterval $[x_k, x_{k+1}]$, s is a third degree polynomial s_k , $k = 0, \ldots, n-1$,

b. The values at the nodes satisfy $s(x_k) = f(x_k)$, for $k = 0, \ldots, n$,

c. $s_k(x_{k+1}) = s_{k+1}(x_{k+1})$, $k = 0, \ldots, n-2$,
$s'_k(x_{k+1}) = s'_{k+1}(x_{k+1})$, $k = 0, \ldots, n-2$,
$s''_k(x_{k+1}) = s''_{k+1}(x_{k+1})$, $k = 0, \ldots, n-2$.

In order to uniquely determine the unknowns, two extra conditions are required. A viable choice is to prescribe

d. $s''_0(x_0) = s''_{n-1}(x_n) = 0$,

which implies that the curvature vanishes at the boundary.

Note that $s \in C^2[a, b]$. The conditions under d could be replaced with other conditions, but these so-called *natural boundary conditions* belong to the shipbuilding application.

In order to demonstrate how to determine such an interpolating spline, s_k is expressed as

$$s_k(x) = a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k,\ k = 0, \ldots, n-1. \tag{2.10}$$

Define $h_k = x_{k+1} - x_k$, $k = 0, \ldots, n-1$, and $f_k = f(x_k)$, $k = 0, \ldots, n$. From property b it follows that

$$s_k(x_k) = d_k = f_k,\ k = 0, \ldots, n-1. \tag{2.11}$$

Next, the various conditions from c are used.

1. Second derivatives: $s''_k(x_{k+1}) = s''_{k+1}(x_{k+1})$

   From (2.10) it follows that $s''_k(x) = 6a_k(x - x_k) + 2b_k$, such that

   $$6a_k h_k + 2b_k = s''_k(x_{k+1}) = s''_{k+1}(x_{k+1}) = 2b_{k+1},\ k = 0, \ldots, n-2.$$

   By defining the value b_n as $b_n = s''_{n-1}(x_n)/2$, the relation $6a_k h_k + 2b_k = 2b_{k+1}$ holds for $k = n-1$ as well. This means that a_k can be expressed in b_k:

   $$a_k = \frac{1}{3h_k}(b_{k+1} - b_k),\ k = 0, \ldots, n-1. \tag{2.12}$$

2. Function values: $s_k(x_{k+1}) = s_{k+1}(x_{k+1})$, together with $s(x_n) = f(x_n)$

   By defining the value d_n as $d_n = f_n$, this reduces to $a_k h_k^3 + b_k h_k^2 + c_k h_k + d_k = d_{k+1}$, $k = 0, \ldots, n-1$. By substituting equations (2.11) and (2.12) for d_k and a_k, respectively, the following expression for c_k is derived:

   $$c_k = \frac{f_{k+1} - f_k}{h_k} - h_k \frac{2b_k + b_{k+1}}{3},\ k = 0, \ldots, n-1. \tag{2.13}$$

3. First derivatives: $s'_k(x_{k+1}) = s'_{k+1}(x_{k+1})$

   This relation yields $3a_k h_k^2 + 2b_k h_k + c_k = c_{k+1}$, $k = 0, \ldots, n-2$. Substituting a_k, c_k (equation 2.13) and d_k leads to

   $$h_k(b_{k+1} - b_k) + 2h_k b_k + \frac{f_{k+1} - f_k}{h_k} - h_k \frac{2b_k + b_{k+1}}{3} = \frac{f_{k+2} - f_{k+1}}{h_{k+1}} - h_{k+1} \frac{2b_{k+1} + b_{k+2}}{3},$$

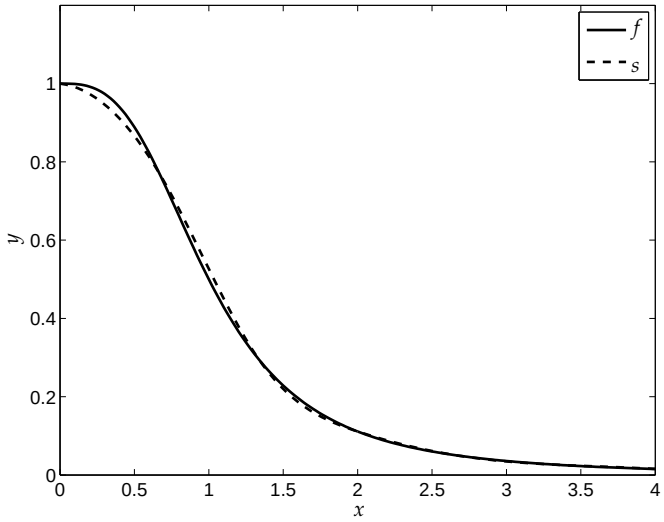


Figure 2.6: Interpolation of $f(x) = 1/(1+x^3)$ using a cubic spline on six subintervals.

which can be simplified to

$$h_k b_k + 2(h_k + h_{k+1})b_{k+1} + h_{k+1}b_{k+2} = 3\left(\frac{f_{k+2} - f_{k+1}}{h_{k+1}} - \frac{f_{k+1} - f_k}{h_k}\right),\ k = 0, \ldots, n-2.$$

Property d implies that $s_0''(x_0) = 2b_0 = 0$, such that $b_0 = 0$. Using the definition $b_n = s_{n-1}''(x_n)/2$ (step 1), we also find $b_n = 0$. Therefore, the resulting system consists of $n-1$ equations for $n-1$ unknowns $b_1, \ldots, b_{n-1}$:

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & \\ h_1 & 2(h_1+h_2) & h_2 & \\ \ddots & \ddots & \ddots & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} 3\left(\frac{f_2-f_1}{h_1} - \frac{f_1-f_0}{h_0}\right) \\ \vdots \\ \vdots \\ 3\left(\frac{f_n-f_{n-1}}{h_{n-1}} - \frac{f_{n-1}-f_{n-2}}{h_{n-2}}\right) \end{pmatrix}.$$

From this system the values b_k can be calculated, from which a_k and c_k can be determined using equations (2.12) and (2.13) and $b_0 = b_n = 0$.

Example 2.5.1 (Visualization)

In this example, a cubic spline on six subintervals is used to approximate a symmetric hill (example 2.4.3). The result in Figure 2.6 is of higher quality than the Hermite and Lagrange interpolation approximation in Figure 2.5. Note that for Hermite interpolation, the derivatives at the nodes should be known, whereas this is not required in the cubic spline approach.

Remark
Splines are used in CAD and experience a revival in modern numerical methods for partial differential equations.

2.6 Summary

In this chapter, the following subjects have been discussed:

- Linear interpolation;
- Higher-order Lagrange interpolation;
- Interpolation with derivatives:
  - Taylor polynomial;
  - Interpolation in general;
  - Hermite interpolation;
- Spline interpolation.

2.7 Exercises

1. (a) Prove upper bound (2.4) for linear interpolation:

$$|f(x) - L_1(x)| \leq \frac{1}{8}(x_1 - x_0)^2 \max_{\xi \in (a,b)} |f''(\xi)|.$$

   (b) Show that the difference between the exact and the perturbed linear interpolation polynomial is bounded by

$$|L_1(x) - \hat{L}_1(x)| \leq \frac{|x_1 - x| + |x - x_0|}{x_1 - x_0}\varepsilon,$$

   and that, for $x \in [x_0, x_1]$,

$$|L_1(x) - \hat{L}_1(x)| \leq \varepsilon,$$

   and if $x \geq x_1$,

$$|L_1(x) - \hat{L}_1(x)| \leq \left(1 + 2\frac{x - x_1}{x_1 - x_0}\right)\varepsilon.$$

2. Determine the second degree Lagrange polynomial of $f(x) = 1/x$ using nodes $x_0 = 2$, $x_1 = 2.5$ and $x_2 = 4$. Approximate $f(3)$ with this polynomial.

3. Determine $s(1/2)$, in which s is the cubic spline with nodes 0, 1 and 2, both for the function $f(x) = x$ and for $f(x) = x^2$.

4. The distance that a car has driven is recorded every minute. The measured data are tabulated below:

Time (in min):	0	1	2	3
Distance (in km):	0	0.5	1	2

   (a) Compute the Lagrange interpolation polynomial of degree 3. Use this polynomial to estimate the maximum speed of the car.

   (b) Compute the cubic spline that interpolates the above data. Use this spline to estimate the maximum speed of the car.

Chapter 3

Numerical differentiation

3.1 Introduction

Everyone who possesses a car and/or a driver's licence is familiar with speeding tickets. In The Netherlands, speeding tickets are usually processed in a fully automated fashion, and the perpetrator will receive the tickets within a couple of weeks after the offence. The Dutch police optimized the procedures of speed control such that this effort has become very profitable to the Dutch government. Various strategies for speed control are carried out by police forces, which are all based on the position of the vehicle at consecutive times. The actual velocity follows from the first-order derivative of the position of the vehicle with respect to time. Since no explicit formula for this position is available, the velocity can only be estimated using an approximation of the velocity based on several discrete vehicle positions at discrete times. This motivates the use of approximate derivatives, also called *numerical derivatives*. If the police want to know whether the offender drove faster before speed detection (in other words, whether the perpetrator hit the brakes after having seen the police patrol), or whether the driver was already accelerating, then they are also interested in the acceleration of the 'bad guy'. This acceleration can be estimated using numerical approximations of the second-order derivative of the car position with respect to time.

Since the time-interval of recording is nonzero, the velocity is not determined exactly in general. In this chapter, the resulting error, referred to as the *truncation error*, is estimated using Taylor series. In most cases, the truncation error increases with an increasing size of the recording interval (Sections 3.2 and 3.4). Next to the truncation error, the measurement of the position of the vehicle is also prone to measurement errors. Issues that influence the results are, for example, parallax, the measurement equipment, and in some cases even the performance of the police officer (in car-videoing and laser control). These measurement errors provide an additional deterioration of the approximation of the speed and acceleration. The impact of measurement errors on approximations of derivatives is treated in Section 3.3.

3.2 Simple difference formulae for the first derivative

Suppose f is a continuously differentiable function. The *forward difference* is defined as

$$Q_f(h) = \frac{f(x+h) - f(x)}{h}, \quad h > 0,$$

in which h is called the *step size*. By definition,

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

and hence, the forward difference converges to the derivative. The *truncation error* is defined as

$$R_f(h) = f'(x) - \frac{f(x+h) - f(x)}{h}. \tag{3.1}$$

Theorem 3.2.1 *If $f \in C^2[x, x+h]$, then there exists a $\xi \in (x, x+h)$ such that*

$$R_f(h) = -\frac{h}{2} f''(\xi). \tag{3.2}$$

Proof
Taylor expansion of $f(x+h)$ about x yields

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\xi), \text{ with } \xi \in (x, x+h).$$

Substitution into (3.1) yields

$$R_f(h) = f'(x) - \frac{f(x) + hf'(x) + \frac{h^2}{2} f''(\xi) - f(x)}{h} = -\frac{h}{2} f''(\xi),$$

which completes the proof. ⊠

Example 3.2.1 (Forward difference)

In this example, the derivative of the function $f(x) = -x^3 + x^2 + 2x$ is approximated in $x = 1$ using the step size $h = 0.5$. In Figure 3.1, it can be seen that the approximation of the derivative is not yet very accurate.

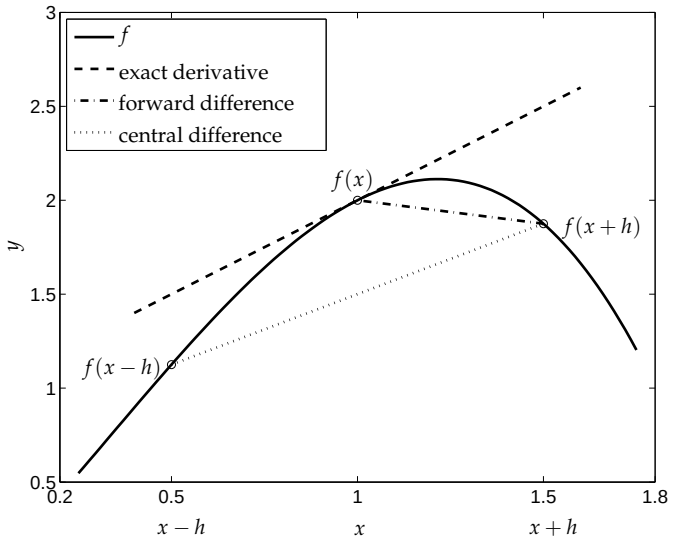


Figure 3.1: Approximation of the derivative of $f(x) = -x^3 + x^2 + 2x$ in $x = 1$ by a forward difference or central difference.

Similar to the forward difference, a *backward difference* can be defined as

$$Q_b = \frac{f(x) - f(x-h)}{h}, \quad h > 0.$$

The truncation error for this formula is

$$R_b(h) = \frac{h}{2} f''(\eta), \text{ for some } \eta \in (x-h, x),$$

which means that the accuracy is comparable to that of the forward difference.

Note that the errors have opposite sign. Moreover, ξ and η will differ, but for small h this difference is very small. It may therefore be interesting to average these two formulae such that these errors almost cancel. This average is known as the *central difference*:

$$Q_c(h) = \frac{f(x+h) - f(x-h)}{2h}.$$

For this particular example, this results in a better approximation (Figure 3.1). If the third derivative of f exists and is continuous, then the truncation error in the central difference becomes

$$R_c(h) = f'(x) - \frac{f(x+h) - f(x-h)}{2h} = -\frac{h^2}{6} f'''(\xi),$$

with ξ between $x-h$ and $x+h$. To prove this, f is expanded in a Taylor polynomial about x with remainder term, and the intermediate-value theorem is used. Note that the error in the central-difference approximation converges much faster towards zero as a function of h, than the error of the forward or backward formula.

Remark
In the limit $h \to 0$, a central difference approximation approximates the derivative more accurate. However, for a given h the error can still be larger than when forward or backward differences are used.

3.3 Rounding errors

Apart from truncation errors, also rounding errors play an important role in numerical differentiation. In this section, the rounding-error behavior of the central-difference formula is investigated.

Suppose that the function values contain a rounding error of at most ε, such that

$$|f(x-h) - \hat{f}(x-h)| \leq \varepsilon, \text{ and } |f(x+h) - \hat{f}(x+h)| \leq \varepsilon.$$

In that case, the rounding error of the central-difference approximation of $f'(x)$, $S_c(h)$, is bounded by

$$\begin{aligned} S_c(h) = |Q_c(h) - \hat{Q}_c(h)| &= \left| \frac{f(x+h) - f(x-h)}{2h} - \frac{\hat{f}(x+h) - \hat{f}(x-h)}{2h} \right| \\ &\leq \frac{|f(x-h) - \hat{f}(x-h)| + |f(x+h) - \hat{f}(x+h)|}{2h} \leq \frac{\varepsilon}{h}. \end{aligned}$$

We note that the upper bound of $S_c(h)$ increases if h decreases.

Furthermore, the truncation error of the approximation is given by

$$R_c(h) = f'(x) - Q_c(h) = -\frac{h^2}{6} f'''(\xi), \;\; \xi \in (x-h, x+h),$$

which will reduce if h becomes smaller.

The total error of the approximation is equal to

$$\begin{aligned} |E_c(h)| &= |f'(x) - \hat{Q}_c(h)| = |f'(x) - Q_c(h) + Q_c(h) - \hat{Q}_c(h)| \\ &\leq |f'(x) - Q_c(h)| + |Q_c(h) - \hat{Q}_c(h)| = |R_c(h)| + S_c(h) \leq \frac{h^2}{6} m + \frac{\varepsilon}{h} = \varphi(h), \end{aligned}$$

in which m is an upper bound for $|f'''(x)|$ in a sufficiently large neighborhood of x. The minimum value of this upper bound is $\varphi(h_{\text{opt}}) = \sqrt[3]{9\varepsilon^2 m/8}$, for $h_{\text{opt}} = \sqrt[3]{3\varepsilon/m}$.

Example 3.3.1 (Rounding errors)

In this example, the derivative of $f(x) = e^x$ is approximated in $x = 1$, using the central-difference formula and six digits. In that case, the used function values, $\hat{f}(x)$, contain rounding errors (Table 3.1), whose absolute value, $|f(x) - \hat{f}(x)|$, is at most $5 \cdot 10^{-6}$. The corresponding approximations, $\hat{Q}_c(h)$, are presented in Table 3.2 together with the errors $f'(1) - \hat{Q}_c(h)$.

Table 3.1: Value of e^x, six digits.

x	$\hat{f}(x)$
0.8000	2.22554
0.9000	2.45960
0.9900	2.69123
0.9990	2.71556
0.9999	2.71801
1.0000	2.71828
1.0001	2.71855
1.0010	2.72100
1.0100	2.74560
1.1000	3.00417
1.2000	3.32012

Table 3.2: Results of the central-difference formula.

h	$\hat{Q}_c(h)$	$E_c(h) = f'(1) - \hat{Q}_c(h)$
0.2000	2.73645	-0.01817
0.1000	2.72285	-0.00457
0.0100	2.71850	-0.00022
0.0010	2.72000	-0.00172
0.0001	2.70000	0.01828

In Table 3.2, it can be seen that for decreasing h, the approximation becomes better at first but it deteriorates rapidly for very small h, which is caused by the rounding errors in the function values.

For $h \leq 0.1$, we know that $m = \max\{f'''(\xi), \xi \in (1-h, 1+h)\} = e^{1.1}$. The function $\varphi(h)$ and the upper bounds of $|R_c(h)|$ (equal to $h^2 m/6$) and $S_c(h)$ (equal to ε/h) that belong to this example are drawn in Figure 3.2. Indeed, it can be seen that, for decreasing h, the upper bound of the truncation error decreases, whereas the rounding-error bound increases. It is easy to verify that $\varphi(h)$ is minimal for $h_{\text{opt}} = 0.0171$, with $\varphi(h_{\text{opt}}) = 0.00044$. Note that $\varphi(h)$ is almost constant in the neighborhood of h_{opt} (in particular for $h > h_{\text{opt}}$), hence a step size close to h_{opt} will lead to a comparable accuracy.

Remarks

- If the approximation of the derivative is insufficiently accurate, then there are two possible approaches to improve this:
  - Decrease the rounding and/or measurement error in the function values;
  - Use a higher-order difference formula. In that case the truncation error, $R(h)$, is often smaller compared to a lower-order formula. Note that h should not be too small, in order to avoid that the loss in significant digits due to the rounding error will annihilate the improvement.
- Difference formulae are often used to solve differential equations (Chapter 7). In that case the solution to the differential equation has to be determined accurately rather than the derivatives. The behavior with respect to rounding errors is often much better in that case.
- The effect of rounding errors is large when the error in the function values is large and if h is small. This often happens with inherently imprecise measurements. In a computer generated table, the rounding error is often much smaller than in a table generated by measurements, but still noticeable for a small h.
- For $h < h_{\text{opt}}$, the measurement or rounding error may even explode when $h \to 0$.

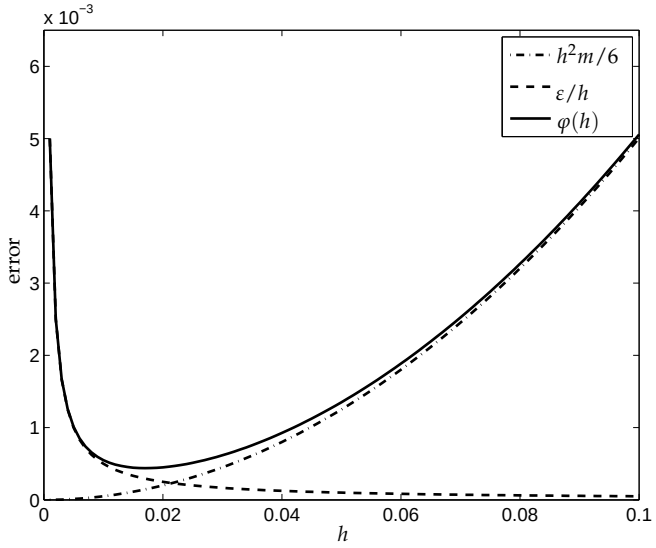


Figure 3.2: Upper bounds of absolute truncation error, rounding error, and total error, using central differences in approximating the derivative of $f(x) = e^x$ in $x = 1$.

3.4 General difference formulae for the first derivative

In the previous sections, some simple difference formulae to approximate the first derivative have been introduced. In many practical applications, it is necessary to use difference formulae of higher accuracy (for example to counteract rounding errors) or formulae with non-equidistant nodes. In this section, a general method to derive difference formulae will be presented. This is done in a way, such that the order of the truncation error, given by

$$|f'(x) - Q(h)| = \mathcal{O}(h^p),$$

is as high as possible.

Assume that $f'(x)$ is approximated by a difference formula based on nodes $x_i = x + ih$, $h > 0$. The general form of $Q(h)$ is given by

$$Q(h) = \frac{1}{h} \sum_i \alpha_i f_i, \text{ in which } f_i = f(x_i). \tag{3.3}$$

Since the derivative of $f(x)$ is defined as

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

it is natural to include the division by h in equation (3.3). The second-order derivative $f''(x)$ equals

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

and hence one more division by h follows. Repeating this argument for higher-order derivatives motivates the following rule of thumb:

Rule of thumb 1
In a difference formula for the kth derivative, the coefficients have to be divided by h^k.

In order to determine coefficients α_i, a Taylor expansion of f_i is computed about x in the variable h. The coefficients are determined such that the order of the error is maximized. Two examples of this idea will be provided using central or one-sided difference formulae.

Example 3.4.1 (Central difference)

In this example, $x_{-1} = x - h$, $x_0 = x$ and $x_1 = x + h$ are used, such that

$$Q(h) = \frac{\alpha_{-1} f(x-h) + \alpha_0 f(x) + \alpha_1 f(x+h)}{h}. \tag{3.4}$$

Taylor expansion yields

$$\begin{aligned} f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{3!} f'''(x) + \mathcal{O}(h^4), \\ f(x) &= f(x), \\ f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{3!} f'''(x) + \mathcal{O}(h^4). \end{aligned}$$

Substituting these expansions in equation (3.4) yields

$$\begin{aligned} Q(h) &= \frac{\alpha_{-1}\left(f(x) - hf'(x) + h^2/2 f''(x) - h^3/3! f'''(x) + \mathcal{O}(h^4)\right)}{h} \\ &+ \frac{\alpha_0 f(x) + \alpha_1\left(f(x) + hf'(x) + h^2/2 f''(x) + h^3/3! f'''(x) + \mathcal{O}(h^4)\right)}{h} \\ &= f(x)(\alpha_{-1}/h + \alpha_0/h + \alpha_1/h) + f'(x)(-\alpha_{-1} + \alpha_1) \\ &+ f''(x)(h\alpha_{-1}/2 + h\alpha_1/2) + f'''(x)(-h^2\alpha_{-1}/3! + h^2\alpha_1/3!) + \mathcal{O}(h^3). \end{aligned}$$

Since $Q(h)$ should approximate the first derivative of f, coefficients α_{-1}, α_0, and α_1 are determined by the conditions

$$\begin{array}{rccccccc} f(x): & \frac{\alpha_{-1}}{h} & + & \frac{\alpha_0}{h} & + & \frac{\alpha_1}{h} & = & 0, \\ f'(x): & -\alpha_{-1} & & & + & \alpha_1 & = & 1, \\ f''(x): & \frac{1}{2}h\alpha_{-1} & & & + & \frac{1}{2}h\alpha_1 & = & 0. \end{array}$$

Solving this system leads to $\alpha_{-1} = -1/2, \alpha_0 = 0$ and $\alpha_1 = 1/2$, which yields the central-difference formula

$$Q(h) = \frac{f(x+h) - f(x-h)}{2h}, \tag{3.5}$$

with truncation error $\mathcal{O}(h^2)$.

Example 3.4.2 (One-sided difference formula of order $\mathcal{O}(h^2)$)

Suppose a derivative has to be determined at the left boundary of an interval, with a truncation error of order $\mathcal{O}(h^2)$. Choosing $x_0 = x, x_1 = x + h$, $x_2 = x + 2h$ and

$$Q(h) = \frac{\alpha_0 f(x) + \alpha_1 f(x+h) + \alpha_2 f(x+2h)}{h},$$

the Taylor expansions are given by

$$\begin{aligned} f(x) &= f(x), \\ f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \mathcal{O}(h^3), \\ f(x+2h) &= f(x) + 2hf'(x) + 2h^2 f''(x) + \mathcal{O}(h^3). \end{aligned}$$

The following conditions are obtained:

$$\begin{array}{rccccccc} f(x): & \frac{\alpha_0}{h} & + & \frac{\alpha_1}{h} & + & \frac{\alpha_2}{h} & = & 0, \\ f'(x): & & & \alpha_1 & + & 2\alpha_2 & = & 1, \\ f''(x): & & & \frac{h}{2}\alpha_1 & + & 2h\alpha_2 & = & 0. \end{array}$$

Solving this system yields $\alpha_0 = -3/2, \alpha_1 = 2$ and $\alpha_2 = -1/2$, such that the one-sided difference formula equals

$$Q(h) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}, \tag{3.6}$$

with truncation error $\mathcal{O}(h^2)$.

3.5 Relation between difference formulae and interpolation *

In Section 3.4, a general way to derive difference formulae has been discussed. A clear disadvantage, however, is that a set of equations has to be solved, where the number of unknowns increases with the order of accuracy.

In this section, a different approach is considered using Lagrange interpolation polynomials to approximate functions (Section 2.3). It seems rather obvious that the derivative of the interpolation polynomial may be taken as an approximation of the derivative of the function. This idea is pursued to derive difference formulae in a different way.

If $x_0, x_1, \ldots, x_n$ do not coincide and $f \in C^{n+1}[a,b]$, then the Lagrange interpolation formulation is given by

$$f(x) = \sum_{k=0}^{n} f(x_k) L_{kn}(x) + (x - x_0) \cdots (x - x_n) \frac{f^{(n+1)}(\xi(x))}{(n+1)!},$$

in which $\xi(x) \in (a,b)$. Differentiation of this expression yields

$$\begin{aligned} f'(x) = \sum_{k=0}^{n} f(x_k) L'_{kn}(x) + \frac{d}{dx}\left((x - x_0) \cdots (x - x_n)\right) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \\ + (x - x_0) \cdots (x - x_n) \frac{d}{dx} \frac{f^{(n+1)}(\xi(x))}{(n+1)!}. \end{aligned} \tag{3.7}$$

The last term generally cannot be evaluated. However, if $x = x_j$, then equation (3.7) becomes

$$f'(x_j) = \sum_{k=0}^{n} f(x_k) L'_{kn}(x_j) + \prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k) \frac{f^{(n+1)}(\xi(x_j))}{(n+1)!}.$$

Example 3.5.1 (Forward difference)

For $n = 1, x = x_0$ and $x_1 = x_0 + h$, the Lagrange basis polynomials satisfy

$$\begin{aligned} L_{01}(x) &= \tfrac{x - x_1}{x_0 - x_1} \quad , \quad L'_{01}(x) &= -\tfrac{1}{h}, \\ L_{11}(x) &= \tfrac{x - x_0}{x_1 - x_0} \quad , \quad L'_{11}(x) &= \tfrac{1}{h}. \end{aligned}$$

This means that the derivative is given by

$$f'(x_0) = -\frac{1}{h} f(x_0) + \frac{1}{h} f(x_0 + h) + \frac{h}{2} f''(\xi),$$

which is exactly the forward-difference formula.

3.6 Difference formulae of higher-order derivatives

In the introduction it has been explained that second-order derivatives are required in order to compute the acceleration from measured position values.

Higher-order derivatives may be approximated using a similar approach as for the first derivative. Suppose that $f''(x)$ should be approximated with a central-difference formula $Q(h)$. Choosing $x_{-1} = x - h, x_0 = x$ and $x_1 = x + h$, the approximation is computed as

$$Q(h) = \frac{\alpha_{-1} f(x-h) + \alpha_0 f(x) + \alpha_1 f(x+h)}{h^2},$$

where the division by h^2 has been motivated in Rule of thumb 1. Taylor expansion about x yields

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{3!} f'''(x) + \mathcal{O}(h^4), \\ f(x) &= f(x), \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{3!} f'''(x) + \mathcal{O}(h^4). \end{aligned}$$

The conditions for α_{-1}, α_0 and α_1 are

$$\begin{array}{lccccccc} f(x): & \frac{\alpha_{-1}}{h^2} & + & \frac{\alpha_0}{h^2} & + & \frac{\alpha_1}{h^2} & = & 0, \\ f'(x): & -\frac{\alpha_{-1}}{h} & & & + & \frac{\alpha_1}{h} & = & 0, \\ f''(x): & \frac{1}{2}\alpha_{-1} & & & + & \frac{1}{2}\alpha_1 & = & 1. \end{array}$$

Solving this system yields $\alpha_{-1} = 1, \alpha_0 = -2$ and $\alpha_1 = 1$, such that

$$Q(h) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{3.8}$$

Note that the error in this formula is $\mathcal{O}(h^2)$.

Rule of thumb 2
To derive a numerical method for the kth derivative with truncation error $\mathcal{O}(h^p)$, the Taylor expansions should have a remainder term of order $\mathcal{O}(h^{p+k})$.

Another way to determine the second derivative is the repeated application of the formula of the first derivative. To approximate the second derivative numerically, at least three points are required. Suppose that the second derivative is approximated using three equidistant function values, $f(x-h)$, $f(x)$, and $f(x+h)$.

The obvious way to approximate the second derivative is to take the difference of the first derivative at the points $x + h$ and $x - h$. Using central differences, this means that the function values in $x + 2h$ and $x - 2h$ are required. This can be avoided by introducing auxiliary points $x \pm h/2$ (Figure 3.3) and taking a central difference of the derivatives at these points:

$$f''(x) \approx \frac{f'(x + \frac{1}{2}h) - f'(x - \frac{1}{2}h)}{h}.$$

Applying a central-difference scheme again to the first-order derivatives yields

$$f''(x) \approx \frac{1}{h}\left(\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h}\right),$$

which can be simplified to

$$Q(h) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

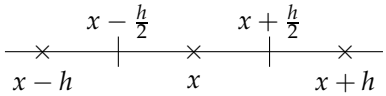


Figure 3.3: Nodes to approximate $f''(x)$ using a repeated central-difference formula.

The truncation error is obtained using Taylor polynomials, and equals

$$f''(x) - Q(h) = -\frac{h^2}{12} f^{(4)}(\xi). \tag{3.9}$$

Note that the effect of rounding errors is even more serious here. An upper bound for the error caused by rounding errors is:

$$S(h) \leq \frac{4\varepsilon}{h^2}.$$

To solve a certain type of differential equation it may be necessary to approximate $(vf')'$, in which v is a (given) function. Analogous to the above, the following approximation of $(vf')'$ can be used:

$$\frac{(vf')(x + \frac{1}{2}h) - (vf')(x - \frac{1}{2}h)}{h}.$$

Applying central differences once more yields

$$\frac{v(x + \frac{1}{2}h)(f(x+h) - f(x)) - v(x - \frac{1}{2}h)(f(x) - f(x-h))}{h^2}.$$

3.7 Richardson's extrapolation

3.7.1 Introduction

In the previous sections, some truncation error estimates for various difference formulae were presented. These expressions often contain a higher-order derivative of the function. However, if the first derivative is unknown, it is impossible to determine a higher-order derivative. In short, these estimates are often of theoretical importance, but useless in practice. In Section 3.7.2, it will be explained how Richardson's extrapolation can be used as a practical method to make a numerical estimate for the error. Furthermore, Section 3.7.3 explains the use of Richardson's extrapolation to obtain a higher accuracy with a low-order method in case the order of the error, p, is known.

In Richardson's extrapolation, a numerical method $Q(h)$ is used to approximate an unknown value M, where we assume that the error in this approximation has the form

$$M - Q(h) = c_p h^p + \mathcal{O}(h^{p+1}), \tag{3.10}$$

in which $c_p \neq 0$, $p \in \mathbb{N}$. Note that the error has this form if it can be expressed as a Taylor series and that p is the order of the error.

In this section, Richardson's extrapolation will be applied to difference formulae, but the method can also be used for other types of approximations (interpolation, numerical integration etc.), as long as the error has the form (3.10).

Table 3.3: Approximation of $f'(1)$ using $f(x) = e^x$, and forward difference.

Grid size	Approximation	Differences
$4h = 0.1$	$Q(4h) = 2.8588\ldots$	-
$2h = 0.05$	$Q(2h) = 2.7873\ldots$	$Q(2h) - Q(4h) = -0.0714\ldots$
$h = 0.025$	$Q(h) = 2.7525\ldots$	$Q(h) - Q(2h) = -0.0348\ldots$

3.7.2 Practical error estimate

In this section, Richardson's extrapolation will be used to make a numerical estimate of the error $M - Q(h)$.

For h sufficiently small, equation (3.10) can be approximated by

$$M - Q(h) = c_p h^p. \tag{3.11}$$

For a given h, $Q(h)$ can be computed, but M, c_p and p are still unknown. However, these values can be approximated by computing for example $Q(h)$, $Q(2h)$ and $Q(4h)$. In that way, the following set of equations is obtained:

$$M - Q(4h) = c_p(4h)^p, \tag{3.12a}$$
$$M - Q(2h) = c_p(2h)^p, \tag{3.12b}$$
$$M - Q(h) = c_p(h)^p. \tag{3.12c}$$

By subtracting equation (3.12b) from (3.12a) and equation (3.12c) from (3.12b), M can be eliminated and only two unknowns are left:

$$Q(2h) - Q(4h) = c_p(2h)^p(2^p - 1), \tag{3.13a}$$
$$Q(h) - Q(2h) = c_p(h)^p(2^p - 1). \tag{3.13b}$$

Next, c_p and h can be eliminated by dividing these two expressions, which yields

$$\frac{Q(2h) - Q(4h)}{Q(h) - Q(2h)} = 2^p, \tag{3.14}$$

from which p may be determined. Substitution of p into (3.13b) provides an approximation for $c_p h^p$, which is used in equation (3.12c) to find the error estimate $M - Q(h)$.

Example 3.7.1 (Practical error estimate)

Once again, the numerical approximation of the derivative of $f(x) = e^x$ is approximated in $x = 1$, here using a forward-difference formula. The exact value of the derivative is known: $M = f'(1) = e = 2.71828\ldots$. Using $h = 0.025$, the values $Q(h)$, $Q(2h)$, and $Q(4h)$ are tabulated in Table 3.3.

Substituting these results into (3.14) yields

$$\frac{Q(2h) - Q(4h)}{Q(h) - Q(2h)} = \frac{-0.0714\ldots}{-0.0348\ldots} = 2.0509\ldots = 2^p.$$

As expected the value of p is almost equal to 1. Because p (the order of the error) should be an integer, we take it equal to 1. From equation (3.13b), the following error estimate is obtained:

$$M - Q(h) = c_p h^p = \frac{Q(h) - Q(2h)}{2^p - 1} = -0.0348\ldots.$$

Note that the exact error equals

$$M - Q(h) = e - 2.7525\ldots = -0.0342\ldots.$$

In this example the error estimate is very reliable.

To receive a better approximation the error estimate can be added to the approximation:

$$Q(h) + c_p h^p = 2.7525\ldots - 0.0348\ldots = 2.7177\ldots.$$

In the above example, the value of p was computed using Richardson's extrapolation. However, using Theorem 3.2.1, it is clear that $p = 1$, and this value could have been used immediately in equation (3.13b) in order to determine $c_p h^p$. In practice, more complex situations are found, and the following complications may occur:

- It is not known whether higher-order derivatives exist and/or are bounded.
- The final result is a combination of various approximation methods. The influence of these approximations on p is not always clear.
- During implementation of the algorithm in a computer program, errors may be made.

To reveal any of these complications it is good practice to verify whether the calculated p is close to the p that follows from theory.

3.7.3 Formulae of higher accuracy from Richardson's extrapolation *

In several applications the value of p in (3.10) is known. In that case Richardson's extrapolation can be used to determine formulae of higher accuracy.

This is done by making use of the fact that the error estimates for $Q(h)$ and $Q(2h)$ equal

$$M - Q(h) = c_p h^p + \mathcal{O}(h^{p+1}), \tag{3.15a}$$

$$M - Q(2h) = c_p (2h)^p + \mathcal{O}(h^{p+1}). \tag{3.15b}$$

Multiplying equation (3.15a) by 2^p and subtracting equation (3.15b) from this yields

$$2^p (M - Q(h)) - (M - Q(2h)) = 2^p (c_p h^p) - c_p (2h)^p + \mathcal{O}(h^{p+1}),$$

such that

$$(2^p - 1)M - 2^p Q(h) + Q(2h) = \mathcal{O}(h^{p+1}).$$

This means that

$$M = \frac{2^p Q(h) - Q(2h)}{2^p - 1} + \mathcal{O}(h^{p+1}). \tag{3.16}$$

The value $(2^p Q(h) - Q(2h))/(2^p - 1)$ is a new approximation formula for M with an accuracy that is one order higher than the order of $Q(h)$.

Example 3.7.2 (Forward difference of higher accuracy)

As an example, the forward-difference method is considered. The error in the forward-difference formula may be written as

$$f'(x) - Q_f(h) = c_1 h + \mathcal{O}(h^2), \tag{3.17}$$

and the difference for $2h$ equals

$$f'(x) - Q_f(2h) = c_1 2h + \mathcal{O}(h^2). \tag{3.18}$$

Multiplying equation (3.17) by 2 and subtracting (3.18) from this result yields

$$f'(x) - (2Q_f(h) - Q_f(2h)) = \mathcal{O}(h^2).$$

Hence the difference formula

$$2Q_f(h) - Q_f(2h) = \frac{2f(x+h) - 2f(x)}{h} - \frac{f(x+2h) - f(x)}{2h} = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}$$

has a truncation error of order $\mathcal{O}(h^2)$. This formula is equal to (3.6).

3.8 Summary

In this chapter, the following subjects have been discussed:

- Difference methods and truncation errors for the first derivative;
- The effect of measurement errors;
- Difference methods and truncation errors for higher-order derivatives;
- Richardson's extrapolation:
  - Practical error estimate;
  - Increasing accuracy.

3.9 Exercises

1. Prove for $f \in C^3[x-h, x+h]$, that the truncation error in the approximation of $f'(x)$ with central differences is of order $\mathcal{O}(h^2)$.

2. Assume that the position of a ship can be determined with a measurement error of at most 10 meters, and that the real position of the ship during start up is given by the function $S(t) = 0.5at^2$, in which S is expressed in meters and t in seconds. The velocity is approximated by a backward difference with step size h. Give the truncation error and the measurement error in this formula. Take $a = 0.004$. Determine h such that the error in the calculated velocity is minimal. How large is the error?

3. In this exercise, the central-difference formula

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{h^2}{12} f^{(4)}(\xi),\ \xi \in (x-h, x+h)$$

is used to approximate the second derivative of $f(x) = \sin x$ in $x = 1$. The computer, and hence Matlab, uses finite precision arithmetic, and for this reason it computes perturbed function values $\hat{f}(x)$. The rounding error in the function value is bounded by

$$\frac{|f(x) - \hat{f}(x)|}{|f(x)|} \leq eps,$$

in which eps is the relative machine precision (assume that $eps = 10^{-16}$).

   (a) Give an upper bound for the total error (truncation error + rounding error) and minimize this upper bound to find an optimal value for h.

   (b) Take $h = 1$. Compute the true error in the approximation for $f''(1)$. Repeat this eight times and divide h by 10 after every computation. Tabulate the results. Does the h for which the error is minimal agree with the value found in question (a)? Remark: the computations for this question have to be carried out using Matlab.

4. Let the function $f(x) = \sin x, x \in [0, \pi]$ be given. Compute $f'(1)$ by a central difference with $h = 0.1$. Estimate the error by Richardson's extrapolation.

5. Given are $f(x),\ f(x+h)$ and $f(x+2h)$. Derive a formula of maximal order to approximate $f'(x)$.

Chapter 4

Nonlinear equations

4.1 Introduction

The pressure drop in a fluid in motion is examined. For a flow in a pipe with a circular cross section of diameter D (meter), the Reynolds number, Re, is given by

$$Re = \frac{Dv}{\nu},$$

in which v (m/s) is the average flow velocity and ν (m^2/s) is the viscosity of the fluid. The flow is called *laminar* if $Re < 2100$ (low flow velocity) and *turbulent* if $Re > 3000$. For $2100 \leq Re \leq 3000$, the flow is neither laminar nor turbulent.

For turbulent flows, the pressure drop between inflow and outflow is given by

$$P_{\text{out}} - P_{\text{in}} = \frac{\rho w L v^2}{2gD},$$

in which w is a friction coefficient, ρ (kg/m^3) is the fluid density, L (m) is the length and g (m/s^2) is the acceleration of gravity. If the fluid contains particles (sand, paper fibers), then the friction coefficient w satisfies the equation

$$\frac{1}{\sqrt{w}} = \frac{\ln(Re\sqrt{w}) + 14 - \frac{5.6}{k}}{k},$$

in which k is a parameter known from experiments.

In this chapter, numerical methods will be discussed that can be used to determine w if the values of Re and k are known.

4.2 Definitions

In this chapter, various iterative methods will be considered to solve nonlinear equations of the form $f(p) = 0$. The point p is called a *zero* of the function f, or a *root* of the equation $f(x) = 0$. First, some useful definitions and concepts are introduced.

Convergence
Each numerical method generates a sequence $\{p_n\} = p_0, p_1, p_2, \ldots$ which should converge to p: $\lim_{n\to\infty} p_n = p$. Assume that the sequence indeed converges, with $p_n \neq p$ for all n. If there exist positive constants λ and α satisfying

$$\lim_{n\to\infty} \frac{|p - p_{n+1}|}{|p - p_n|^{\alpha}} = \lambda, \tag{4.1}$$

then $\{p_n\}$ converges to p with order α and asymptotic constant λ.

In general a higher-order method converges faster than a lower-order method. The value of the asymptotic constant, λ, is less important. There are two important cases:

- $\alpha = 1$: the process is linearly convergent. In this case λ is called the *asymptotic convergence factor*.
- $\alpha = 2$: the process is quadratically convergent.

Theorem 4.2.1 *Suppose that a sequence* $\{p_n\} = p_0, p_1, p_2, \ldots$ *satisfies* $|p - p_n| \leq k|p - p_{n-1}|$, *for* $n = 1, 2, \ldots$, *where* $0 \leq k < 1$. *Then* $\lim_{n\to\infty} p_n = p$: *the sequence is convergent.*

Proof:
By induction,

$$\lim_{n\to\infty} |p - p_n| \leq \lim_{n\to\infty} k^n |p - p_0| = 0,$$

because $k < 1$. Hence p_n converges to p. ⊠

Stopping criteria
Because an iterative method will be used to approximate p, it is necessary to specify when the method should stop. Some examples of *stopping criteria* are listed below.

1. If the solution, p, is known, then $|p - p_n| < \varepsilon$ is the most useful stopping criterion. Since p is not known in general, this criterion cannot be used in practice.
2. Two successive approximations can be used in the stopping criterion $|p_n - p_{n-1}| < \varepsilon$. However, for some methods, it is possible that $|p_n - p_{n-1}| < \varepsilon$, whereas $|p - p_n| \gg \varepsilon$. In that case the method stops at a wrong approximation of the solution.
3. Since p may be very large or very small, it is often more meaningful to consider the relative error
$$\frac{|p_n - p_{n-1}|}{|p_n|} < \varepsilon, \text{ if } p \neq 0.$$
4. Finally, the accuracy of p_n can be determined by the criterion $|f(p_n)| < \varepsilon$. If this criterion is satisfied, and the first derivative of f exists and is continuous in the neighborhood of p, we can bound the error of the approximation. Using the intermediate-value theorem, we know that $|f(p) - f(p_n)| = |f'(\xi)| \cdot |p - p_n|$ for some ξ between p and p_n. Since $f(p) = 0$ and $|f(p_n)| < \varepsilon$, we find that $|f'(\xi)| \cdot |p - p_n| < \varepsilon$. By defining $m = \min\{|f'(\xi)|, \xi \text{ between } p \text{ and } p_n\}$ it follows that
$$|p - p_n| < \frac{\varepsilon}{m}, \quad \text{if} \quad m \neq 0. \tag{4.2}$$

Uncertainty interval
In numerical simulations, the exact function values, $f(x)$, are often unknown. Instead, an approximation of the function values is used, which is denoted by $\hat{f}(x)$, in which $|f(x) - \hat{f}(x)| \leq \bar{\varepsilon}$. Using the perturbed function, $\hat{f}$, the zero of f cannot be determined exactly, as each point of the set $I = \{x \in [a, b] \mid |f(x)| < \bar{\varepsilon}\}$ can be a solution.

The width of this *uncertainty interval* can be computed using that $f(p) = 0$ (p is the zero of f), and assuming that x^+ is the zero of the maximally perturbed function $\hat{f}(x) = f(x) + \bar{\varepsilon}$, such that $\hat{f}(x^+) = 0$.

Linearization of $f(x^+)$ about p yields

$$0 = \hat{f}(x^+) = f(x^+) + \bar{\varepsilon} \approx f(p) + (x^+ - p)f'(p) + \bar{\varepsilon} = (x^+ - p)f'(p) + \bar{\varepsilon}.$$

This means that $\hat{f}(x^+) = 0$ for

$$x^+ \approx p - \frac{\bar{\varepsilon}}{f'(p)}.$$

Similarly, the zero of the function $\hat{f}(x) = f(x) - \bar{\varepsilon}$ is given by

$$x^- \approx p + \frac{\bar{\varepsilon}}{f'(p)}.$$

The values x^- and x^+ form the bounds of interval I. Because $f'(p)$ can be either positive or negative, the general notation for the uncertainty interval is

$$I \approx \left[p - \frac{\bar{\varepsilon}}{|f'(p)|},\ p + \frac{\bar{\varepsilon}}{|f'(p)|} \right]$$

if $f'(p) \neq 0$. If $|f'(p)|$ is close to 0, then finding the root of the equation $f(p) = 0$ is an ill-posed problem.

If $\varepsilon < \bar{\varepsilon}$, then the stopping criterion $|f(p_n)| < \varepsilon$ is useless, because it is possible that the algorithm continues, although $p_n \in I$.

4.3 A simple root finder: the Bisection method

The first method, the Bisection method, is based on the intermediate-value theorem. Suppose f is a continuous function defined on an interval $[a, b]$, where $f(a)$ and $f(b)$ have opposite sign (which means that $f(a) \cdot f(b) < 0$). Then, according to the intermediate-value theorem, there is a number p in (a, b) with $f(p) = 0$. The Bisection method divides $[a, b]$ into two equal parts and continues with the interval that contains p.

To begin with, let $a_0 = a$, $b_0 = b$. In general, a new approximation for the zero is computed by

$$p_n = \frac{a_n + b_n}{2}.$$

If the chosen stopping criterion is satisfied, then the zero of f is found. If not, then a new interval $[a_{n+1}, b_{n+1}]$ is constructed using

$$a_{n+1} = a_n \text{ and } b_{n+1} = p_n \text{ if } f(a_n) \cdot f(p_n) < 0,$$

and

$$a_{n+1} = p_n \text{ and } b_{n+1} = b_n \text{ otherwise.}$$

Convergence
Note that this method always converges to a solution: $\lim_{n\to\infty} p_n = p$. In general, it converges slowly, and it may happen that $|p - p_{n-1}| \ll |p - p_n|$. The unconditional convergence makes the Bisection method a useful method for computing a starting estimate for the more efficient methods that will be discussed later in this chapter.

Theorem 4.3.1 *Assume $f \in C[a, b]$ and $f(a) \cdot f(b) < 0$. Then the Bisection method generates a sequence $\{p_n\}$ converging to a zero p of f in which*

$$|p - p_n| \leq \frac{b - a}{2^{n+1}}, \quad n \geq 0.$$

Proof:
For each $n \geq 0$, $b_n - a_n = (b - a)/2^n$ and $p \in (a_n, b_n)$. Since $p_n = (a_n + b_n)/2$ it follows that

$$|p - p_n| \leq \frac{b_n - a_n}{2} = \frac{b - a}{2^{n+1}}.$$

⊠

4.4 Fixed-point iteration (Picard iteration)

A different way to compute the zero of a function, is the use of a fixed-point method. A *fixed point* of a function g is a number p such that $g(p) = p$. In this section, a numerical method is presented to approximate p.

The relation between zeros and fixed points can be explained using a function f that has a zero p. It is always possible to define a function g, for example, $g(x) = x - f(x)$, such that the fixed point of g is the zero of f. Note that g is not unique, for example, the fixed point of the function $g(x) = x - cf(x), c \neq 0$ is also a zero of f. Conversely, if $g(p) = p$, then $f(x) = x - g(x)$ vanishes in p.

In the next theorem, sufficient conditions are provided for the existence and uniqueness of a fixed point.

Theorem 4.4.1 *Let $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$.*

1. *Then, g has a fixed point in $[a,b]$ (Brouwer's fixed-point theorem).*
2. *If, in addition, $g'(x)$ exists for $x \in [a,b]$ and if there exists a positive constant $k < 1$ such that*

$$|g'(x)| \leq k \quad \text{for all} \quad x \in [a,b],$$

then the fixed point in $[a,b]$ is unique (Banach's fixed-point theorem).

Proof:

1. If $g(a) = a$ or $g(b) = b$, then g has a fixed point at one of the end points of the interval. If this is not the case, then $g(a) > a$ and $g(b) < b$ since $g(x) \in [a,b]$. Let the function h be defined as $h(x) = g(x) - x$. This function is continuous on $[a,b]$, and furthermore, $h(a) > 0$ and $h(b) < 0$. From the intermediate-value theorem it follows that there is a p such that $h(p) = 0$, hence p is a fixed point of g.
2. Suppose $|g'(x)| \leq k < 1$ and assume that there are two fixed points p and q with $p < q$. By the mean-value theorem there exists a $\xi \in [p,q]$ such that

$$\frac{g(p) - g(q)}{p - q} = g'(\xi).$$

From this it follows that

$$|p - q| = |g(p) - g(q)| = |g'(\xi)||p - q| < |p - q|,$$

which is a contradiction. Hence, $p = q$ and the fixed point is unique. ⊠

To approximate a fixed point of a function g, a starting value p_0 has to be chosen. For $n \geq 1$, p_n is determined by $p_n = g(p_{n-1})$. If the sequence converges to p and g is continuous, then

$$p = \lim_{n\to\infty} p_n = \lim_{n\to\infty} g(p_{n-1}) = g(\lim_{n\to\infty} p_{n-1}) = g(p),$$

hence p is a fixed point of g. This method is called *fixed-point* (or *Picard*) *iteration*.

Remarks

- It is easily seen that each converging fixed-point method is at least linearly convergent.
- Close to the solution p, the error decreases as $|p - p_{n+1}| \approx |g'(p)||p - p_n|$. If $|g'(p)|$ is small, convergence will be fast, but if $|g'(p)|$ is only marginally smaller than 1, convergence will be slow. If $|g'(p)| \geq 1$ there is no convergence.

  Note: if $g'(p) \neq 0$, then the process is linearly convergent, with asymptotic convergence factor $|g'(p)|$. If $g'(p) = 0$, then the process is higher-order convergent.

- Note that k can be chosen as $k = \max_{x\in[a,b]} |g'(x)|$.
- If the fixed-point iteration $p_n = g(p_{n-1})$ satisfies the conditions of Banach's fixed-point theorem (Theorem 4.4.1) and p_0 and p are located in $[a, b]$, then another stopping criterion can be formulated. Note that for $m > n \geq 0$,

$$\begin{aligned} |p_m - p_n| &= |p_m - p_{m-1} + p_{m-1} - p_{m+2} + \ldots + p_{n+1} - p_n| \\ &\leq |p_m - p_{m-1}| + |p_{m-1} - p_{m-2}| + \ldots + |p_{n+1} - p_n|. \end{aligned}$$

Using the mean-value theorem and the fact that $|g'(x)| \leq k$ for all $x \in [a, b]$, we know that

$$|p_{n+1} - p_n| = |g(p_n) - g(p_{n-1})| \leq k|p_n - p_{n-1}|.$$

Applying this recursively, we find that

$$\begin{aligned} |p_m - p_n| &\leq k^{m-n}|p_n - p_{n-1}| + \ldots + k|p_n - p_{n-1}| \\ &= (k + \ldots + k^{m-n})|p_n - p_{n-1}|, \end{aligned}$$

and, since $\lim_{m\to\infty} p_m = p$ it follows that

$$|p - p_n| = \lim_{m\to\infty} |p_m - p_n| \leq k \sum_{i=0}^{\infty} k^i |p_n - p_{n-1}| = \frac{k}{1-k}|p_n - p_{n-1}|. \tag{4.3}$$

This means that stopping criterion

$$|p_n - p_{n-1}| \leq \frac{1-k}{k}\varepsilon$$

implies that $|p - p_n| \leq \varepsilon$.

Example 4.4.1 (Picard iteration)

To determine the real zero of the function $f(x) = x^3 + 3x - 4$ (which is $p = 1$), the function

$$g(x) = \frac{4}{x^2 + 3}$$

can be used in the iterative process. This function is determined by the following steps:

$$f(p) = 0 \leftrightarrow p^3 + 3p - 4 = 0 \leftrightarrow p^3 + 3p = 4 \leftrightarrow p(p^2 + 3) = 4 \leftrightarrow p = \frac{4}{p^2 + 3} = g(p).$$

Using $p_n = g(p_{n-1})$ and starting with $p_0 = 0$, we find the values as in the second column of Table 4.1 (rounded to four decimals). Note that $g'(x) = -8x/(x^2 + 3)^2$, and hence it can be verified that $|g'(x)| \leq 1/2 = k$. Using the third and fourth column of Table 4.4.1, the validity of equation (4.3) is shown: $|p - p_n| \leq |p_n - p_{n-1}|$. Finally, the fifth column shows that indeed, $|p - p_{n+1}|/|p - p_n| \approx |g'(p)| = 1/2$ (remark after Theorem 4.4.1).

Figure 4.1 illustrates the iterative process. The line $y = x$ is used for the transformation of $y = g(p_{n-1})$ to $x = p_n$.

Theorem 4.4.2 *Suppose $g \in C[a, b]$, $g(x) \in [a, b]$, for $x \in [a, b]$, and $|g'(x)| \leq k < 1$ for $x \in [a, b]$. Then the fixed-point iteration converges to p for each value $p_0 \in [a, b]$.*

Proof:
Under the provided conditions, g has a unique fixed point p. From the mean-value theorem it follows that

$$|p - p_n| = |g(p) - g(p_{n-1})| = |g'(\xi)||p - p_{n-1}| \leq k|p - p_{n-1}|, \text{ where } \xi \text{ is between } p \text{ and } p_{n-1}.$$

From Theorem 4.2.1 it follows that p_n converges to p. ⊠

Table 4.1: Picard iteration $p_n = g(p_{n-1})$ corresponding to example 4.4.1.

n	p_n	$\lvert p - p_n \rvert$	$\lvert p_n - p_{n-1} \rvert$	$\lvert p - p_n \rvert / \lvert p - p_{n-1} \rvert$
0	0	1	-	-
1	1.3333	0.3333	1.3333	0.3333
2	0.8372	0.1628	0.4961	0.4884
3	1.0808	0.0808	0.2436	0.4964

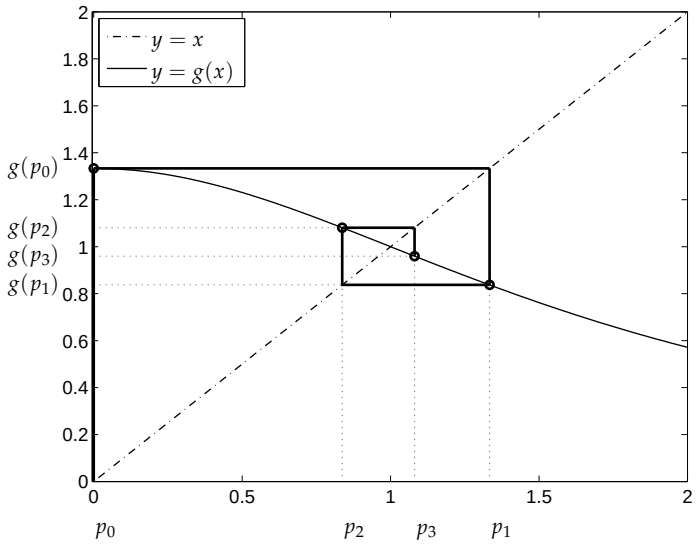


Figure 4.1: Fixed-point method, used to solve $p = 4/(p^2 + 3) = g(p)$. The iteration process is marked by a bold line.

4.5 The Newton-Raphson method

The Newton-Raphson method is one of the most powerful and best-known numerical methods to solve a nonlinear equation $f(x) = 0$. The method will be explained using a Taylor polynomial.

Suppose $f \in C^2[a, b]$. Let $\bar{x} \in [a, b]$ be an approximation of the root p such that $f'(\bar{x}) \neq 0$, and suppose that $|p - \bar{x}|$ is small. Consider the first-degree Taylor polynomial about $\bar{x}$:

$$f(x) = f(\bar{x}) + (x - \bar{x})f'(\bar{x}) + \frac{(x - \bar{x})^2}{2} f''(\xi(x)), \tag{4.4}$$

in which $\xi(x)$ between x and $\bar{x}$. Using that $f(p) = 0$, equation (4.4) yields

$$0 = f(\bar{x}) + (p - \bar{x})f'(\bar{x}) + \frac{(p - \bar{x})^2}{2} f''(\xi(x)).$$

Because $|p - \bar{x}|$ is small, $(p - \bar{x})^2$ can be neglected, such that

$$0 \approx f(\bar{x}) + (p - \bar{x})f'(\bar{x}).$$

Note that the right-hand side is the formula for the tangent in $(\bar{x}, f(\bar{x}))$. Solving for p yields

$$p \approx \bar{x} - \frac{f(\bar{x})}{f'(\bar{x})}.$$

This motivates the Newton-Raphson method, that starts with an approximation p_0 and generates a sequence $\{p_n\}$ by

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad \text{for} \quad n \geq 1.$$

The graphical illustration is that p_n is the zero of the tangent in $(p_{n-1}, f(p_{n-1}))$, which is depicted in Figure 4.2.

Example 4.5.1 (Newton-Raphson method)

In this example, the positive zero of the function $f(x) = x^2 - 2$ will be determined using the Newton-Raphson method. To six decimal places the exact result is $p = \sqrt{2} = 1.41421$. Note that $f'(x) = 2x$, such that the method generates approximations using

$$p_n = p_{n-1} - \frac{(p_{n-1})^2 - 2}{2p_{n-1}}.$$

Starting with $p_0 = 1.00000\ldots$, the values $p_1 = 1.50000\ldots$, $p_2 = 1.41666\ldots$ and $p_3 = 1.41421\ldots$ are calculated. This means that after only three steps, the solution is found to six decimal places. The procedure for the first iteration is explained graphically in Figure 4.2.

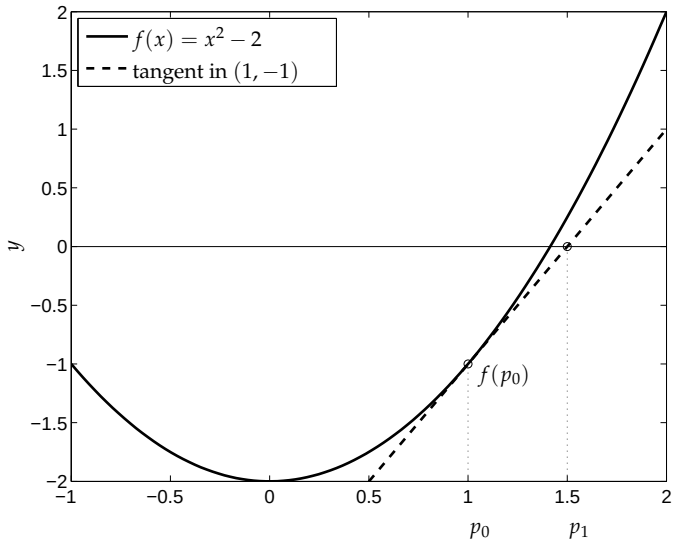


Figure 4.2: The Newton-Raphson method used to solve the positive zero of $f(x) = x^2 - 2$. The zero of the tangent in $(p_0, f(p_0))$ is p_1.

In the next theorem the Newton-Raphson iteration is considered as a fixed-point method.

Theorem 4.5.1 *Let $f \in C^2[a, b]$. If $p \in [a, b]$ such that $f(p) = 0$ and $f'(p) \neq 0$, then there exists a $\delta > 0$, such that the Newton-Raphson method generates a sequence $\{p_n\}$ converging to p for each $p_0 \in [p - \delta,\ p + \delta]$.*

Proof:
In the proof, the Newton-Raphson method is considered as a fixed-point method $p_n = g(p_{n-1})$ with

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

In order to use Theorem 4.4.2, it is necessary to show that there exists a $\delta > 0$, such that g satisfies the conditions of the theorem for $x \in [p - \delta, p + \delta]$.

- Continuity of g

  Since $f'(p) \neq 0$ and f' is continuous, there exists a $\delta_1 > 0$ such that $f'(x) \neq 0$ for all $x \in [p - \delta_1,\ p + \delta_1]$. Therefore, g is well defined and continuous for all x in $[p - \delta_1,\ p + \delta_1]$.

- Derivative of g

  The derivative of g is given by

  $$g'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}.$$

  Since $f \in C^2[a, b]$ it follows that $g'(x)$ is continuous: $g(x) \in C^1[p - \delta_1,\ p + \delta_1]$. Note that $g'(p) = 0$ because $f(p) = 0$. Since g' is continuous, there exists a $\delta < \delta_1$ such that

  $$|g'(x)| \leq k < 1 \quad \text{for all} \quad x \in [p - \delta,\ p + \delta].$$

- Domain and range of g

  Finally, we show that $g(x) \in [p - \delta, p + \delta]$ if $x \in [p - \delta, p + \delta]$. Using the mean-value theorem,

  $$|g(p) - g(x)| = |g'(\xi)||p - x| \quad \text{for some } \xi \text{ between } x \text{ and } p.$$

  Since $x \in [p - \delta,\ p + \delta]$ it follows that $|p - x| < \delta$ and $|g'(\xi)| < 1$, and hence,

  $$|g(p) - g(x)| < |p - x| < \delta.$$

Using Theorem 4.4.2, the sequence $\{p_n\}$ converges to p for each $p_0 \in [p - \delta, p + \delta]$. This proves the theorem. ⊠

Quadratic convergence
The convergence behavior of the Newton-Raphson method can be computed by making use of the following observation:

$$0 = f(p) = f(p_n) + (p - p_n)f'(p_n) + \frac{(p - p_n)^2}{2}f''(\xi_n) \text{ for } \xi_n \text{ between } p_n \text{ and } p. \tag{4.5a}$$

The Newton-Raphson method is defined such that

$$0 = f(p_n) + (p_{n+1} - p_n)f'(p_n). \tag{4.5b}$$

Subtracting expression (4.5b) from (4.5a) yields

$$(p - p_{n+1})f'(p_n) + \frac{(p - p_n)^2}{2}f''(\xi_n) = 0,$$

such that

$$\frac{|p - p_{n+1}|}{|p - p_n|^2} = \left|\frac{f''(\xi_n)}{2f'(p_n)}\right|.$$

From equation (4.1) it follows that the Newton-Raphson method converges quadratically, with $\alpha = 2$ and

$$\lambda = \lim_{n\to\infty}\left|\frac{f''(\xi_n)}{2f'(p_n)}\right| = \left|\frac{f''(p)}{2f'(p)}\right|.$$

4.5.1 Variants of the Newton-Raphson method

In the Newton-Raphson method, the derivative $f'(p_{n-1})$ should be computed. In order to circumvent this, several variations on the standard Newton-Raphson method will be defined. They have the common form

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{K}, \tag{4.6}$$

where K is an approximation of $f'(p_{n-1})$.

Quasi-Newton method
The Quasi-Newton method takes

$$K = \frac{f(p_{n-1}+h) - f(p_{n-1})}{h},$$

where $h > 0$.

Secant method
The Secant method replaces $f'(p_{n-1})$ by

$$K = \frac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}.$$

This method is derived in exercise 3.

Regula-Falsi method
The Regula-Falsi method combines the Newton-Raphson method with the Bisection algorithm. Suppose that the interval $[a_{n-1}, b_{n-1}]$ contains a zero of f, such that $f(a_{n-1}) \cdot f(b_{n-1}) < 0$. Then, approximation p_n is computed as follows:

$$p_n = a_{n-1} - f(a_{n-1}) \cdot \frac{b_{n-1} - a_{n-1}}{f(b_{n-1}) - f(a_{n-1})}.$$

If the chosen stopping criterion is satisfied, then the zero of f is found. If not, then, similar to the Bisection approach, the new interval $[a_n, b_n]$ is constructed by defining

$$a_n = a_{n-1} \text{ and } b_n = p_n \text{ if } f(a_{n-1}) \cdot f(p_n) < 0,$$

and

$$a_n = p_n \text{ and } b_n = b_{n-1} \text{ otherwise.}$$

4.6 Systems of nonlinear equations

In this section, the theory will be extended to systems of nonlinear equations, which can be written either in fixed-point form

$$\begin{cases} g_1(p_1, \ldots, p_m) = p_1, \\ g_2(p_1, \ldots, p_m) = p_2, \\ \qquad\qquad \vdots \\ g_m(p_1, \ldots, p_m) = p_m, \end{cases} \tag{4.7a}$$

or in the general form

$$\begin{cases} f_1(p_1, \ldots, p_m) = 0, \\ f_2(p_1, \ldots, p_m) = 0, \\ \qquad\qquad \vdots \\ f_m(p_1, \ldots, p_m) = 0. \end{cases} \tag{4.7b}$$

In matrix-vector notation, these systems can be written as $\mathbf{g}(\mathbf{p}) = \mathbf{p}$ and $\mathbf{f}(\mathbf{p}) = \mathbf{0}$, respectively, where

$$\begin{aligned}\mathbf{p} &= (p_1, \ldots, p_m)^\top,\\ \mathbf{g}(\mathbf{p}) &= (g_1(p_1, \ldots, p_m), \ldots, g_m(p_1, \ldots, p_m))^\top,\\ \mathbf{f}(\mathbf{p}) &= (f_1(p_1, \ldots, p_m), \ldots, f_m(p_1, \ldots, p_m))^\top.\end{aligned}$$

As in the scalar case, an initial estimate for the solution is used to construct a sequence of successive approximations, $\{\mathbf{p}^{(n)}\}$, of the solution, until a desired tolerance is reached. A possible stopping criterion is $||\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)}|| < \varepsilon$, where $||\mathbf{x}|| = \sqrt{x_1^2 + \ldots + x_m^2}$ denotes the Euclidean norm.

4.6.1 Fixed-point iteration (Picard iteration)

Similar to the scalar case, the fixed-point iteration approximates the solution to system (4.7a), using

$$\mathbf{p}^{(n)} = \mathbf{g}(\mathbf{p}^{(n-1)}). \tag{4.8}$$

To illustrate the use of the fixed-point method, the following example is given.

Example 4.6.1 (Picard iteration)

In this example, the following system of equations is solved:

$$\begin{cases} 2p_1 - \ p_2 + p_1^2 &= \frac{1}{9},\\ -p_1 + 2p_2 + p_2^2 &= \frac{13}{9}. \end{cases} \tag{4.9}$$

It is easy to see that the solution equals $(p_1, p_2)^\top = (1/3, 2/3)^\top$. The Picard iteration uses

$$\begin{aligned} 2p_1^{(n)} - \ p_2^{(n)} + p_1^{(n-1)}p_1^{(n)} &= \tfrac{1}{9},\\ -p_1^{(n)} + 2p_2^{(n)} + p_2^{(n-1)}p_2^{(n)} &= \tfrac{13}{9}, \end{aligned}$$

which can be written as

$$\mathbf{A}(\mathbf{p}^{(n-1)})\mathbf{p}^{(n)} = \mathbf{b} \Leftrightarrow \mathbf{p}^{(n)} = \mathbf{A}^{-1}(\mathbf{p}^{(n-1)})\mathbf{b}, \tag{4.10}$$

where

$$\mathbf{A}(\mathbf{p}^{(n-1)}) = \begin{pmatrix} 2 + p_1^{(n-1)} & -1 \\ -1 & 2 + p_2^{(n-1)} \end{pmatrix},\ \mathbf{b} = \begin{pmatrix} \frac{1}{9} \\ \frac{13}{9} \end{pmatrix}.$$

If the initial estimate equals $\mathbf{p}^{(0)} = (0, 0)^\top$, then the new approximation equals $\mathbf{p}^{(1)} = (5/9, 1)^\top$. Note that system (4.9) can be written as $\mathbf{g}(\mathbf{p}) = \mathbf{p}$, with $\mathbf{g}(\mathbf{p}) = \mathbf{A}(\mathbf{p})^{-1}\mathbf{b}$.

4.6.2 The Newton-Raphson method

A faster iterative method to approximate the solution to system (4.7b) is the Newton-Raphson method. Similar to the scalar case of the Newton-Raphson method, the successive approximation is found by linearizing function f about iterate $\mathbf{p}^{(n-1)}$:

$$\begin{aligned} f_1(\mathbf{p}) &\approx f_1(\mathbf{p}^{(n-1)}) + \frac{\partial f_1}{\partial p_1}(\mathbf{p}^{(n-1)})(p_1 - p_1^{(n-1)}) + \ldots + \frac{\partial f_1}{\partial p_m}(\mathbf{p}^{(n-1)})(p_m - p_m^{(n-1)}),\\ &\vdots\\ f_m(\mathbf{p}) &\approx f_m(\mathbf{p}^{(n-1)}) + \frac{\partial f_m}{\partial p_1}(\mathbf{p}^{(n-1)})(p_1 - p_1^{(n-1)}) + \ldots + \frac{\partial f_m}{\partial p_m}(\mathbf{p}^{(n-1)})(p_m - p_m^{(n-1)}). \end{aligned}$$

Defining the Jacobian matrix of $\mathbf{f}(\mathbf{x})$ by

$$\mathbf{J}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_m}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_m}{\partial x_m}(\mathbf{x}) \end{pmatrix},$$

the linearization can be written in the more compact form

$$\mathbf{f}(\mathbf{p}) \approx \mathbf{f}(\mathbf{p}^{(n-1)}) + \mathbf{J}(\mathbf{p}^{(n-1)})(\mathbf{p} - \mathbf{p}^{(n-1)}).$$

The next iterate, $\mathbf{p}^{(n)}$, is obtained by setting the linearization equal to zero:

$$\mathbf{f}(\mathbf{p}^{(n-1)}) + \mathbf{J}(\mathbf{p}^{(n-1)})(\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)}) = 0, \tag{4.11}$$

which can be rewritten as

$$\mathbf{J}(\mathbf{p}^{(n-1)})\mathbf{s}^{(n)} = -\mathbf{f}(\mathbf{p}^{(n-1)}), \tag{4.12}$$

where $\mathbf{s}^{(n)} = \mathbf{p}^{(n)} - \mathbf{p}^{(n-1)}$. The new approximation equals $\mathbf{p}^{(n)} = \mathbf{p}^{(n-1)} + \mathbf{s}^{(n)}$.

A fast way to compute $\mathbf{p}^{(n)}$ uses equation (4.11):

$$\mathbf{p}^{(n)} = \mathbf{p}^{(n-1)} - \mathbf{J}^{-1}(\mathbf{p}^{(n-1)})\mathbf{f}(\mathbf{p}^{(n-1)}). \tag{4.13}$$

If the computation of the partial derivatives in the Jacobian matrix is not possible, then the derivatives can be approximated using, e.g., forward divided differences:

$$\frac{\partial f_j}{\partial x_i}(\mathbf{x}) \approx \frac{f_j(\mathbf{x} + \mathbf{e}_i h) - f_j(\mathbf{x})}{h}, \tag{4.14}$$

where $\mathbf{e}_i$ is the ith unit vector. This is the *Quasi-Newton* method for systems. It is also possible to use central differences to approximate the derivatives.

Example 4.6.2 (Newton-Raphson method)

In this example, the Newton-Raphson scheme is applied to the following system for p_1 and p_2:

$$\begin{cases} 18p_1 - 9p_2 + p_1^2 = 0, \\ -9p_1 + 18p_2 + p_2^2 = 9, \end{cases} \tag{4.15}$$

with initial estimate $\mathbf{p}^{(0)} = (0, 0)^\top$.

System (4.15) can be written as $\mathbf{f}(\mathbf{p}) = \mathbf{0}$, where

$$\begin{aligned} f_1(p_1, p_2) &= 18p_1 - 9p_2 + p_1^2, \\ f_2(p_1, p_2) &= -9p_1 + 18p_2 + p_2^2 - 9. \end{aligned}$$

The Jacobian of $\mathbf{f}(\mathbf{x})$ is given by

$$\mathbf{J}(\mathbf{x}) = \begin{pmatrix} 18 + 2x_1 & -9 \\ -9 & 18 + 2x_2 \end{pmatrix}.$$

Defining $\mathbf{s}^{(1)} = \mathbf{p}^{(1)} - \mathbf{p}^{(0)}$, equation (4.12) equals

$$\mathbf{J}(\mathbf{p}^{(0)})\mathbf{s}^{(1)} = -\mathbf{f}(\mathbf{p}^{(0)}),$$

such that the following system has to be solved:

$$\begin{aligned} 18s_1^{(1)} - 9s_2^{(1)} &= 0, \\ -9s_1^{(1)} + 18s_2^{(1)} &= 9. \end{aligned}$$

The solution equals $\mathbf{s}^{(1)} = (1/3, 2/3)^\top$, such that $\mathbf{p}^{(1)} = \mathbf{p}^{(0)} + \mathbf{s}^{(1)} = (1/3, 2/3)^\top$.

4.7 Summary

In this chapter, the following subjects have been discussed:

- Stopping criteria;
- Rounding errors;
- Convergence;
- Bisection method;
- Fixed-point iteration (Picard iteration);
- Newton-Raphson method;
- Quasi-Newton method, Secant method, Regula-Falsi method;
- Systems of nonlinear equations: Picard iteration, Newton-Raphson method.

4.8 Exercises

1. Let $f(x) = 3(x+1)(x-1/2)(x-1)$. Use the Bisection method to determine p_2 on the intervals $[-2, 1.5]$ and $[-1.25, 2.5]$.

2. In this exercise, two different fixed-point methods are considered:

$$p_n = \frac{20p_{n-1} + 21/p_{n-1}^2}{21} \text{ and } p_n = p_{n-1} - \frac{p_{n-1}^3 - 21}{3p_{n-1}^2}.$$

   Show that for both methods, $(21)^{1/3}$ is the fixed point. Estimate the convergence speeds, and determine p_3 if $p_0 = 1$.

3. In this exercise, the Secant method is derived to approximate a zero of f.

   (a) Suppose that p_0 and p_1 are two given values. Determine the linear interpolation polynomial of f between p_0 and p_1.

   (b) Compute p_2 as the intersection point of this interpolation polynomial with the x-axis. Repeat this process with p_1 and p_2 to obtain p_3, and derive a general method for computing p_n from p_{n-2} and p_{n-1} (Secant method).

   (c) Perform two iterations with this method using the function $f(x) = x^2 - 2$ with $p_0 = 1$ and $p_1 = 2$.

4. Consider the function $f(x) = x - \cos x, x \in [0, \pi/2]$. Determine an approximation of the zero of f with an error less than 10^{-4} using the method of Newton-Raphson.

5. Perform two iterations with the Newton-Raphson method with starting vector $(1,1)^\top$ to solve the nonlinear system

$$\begin{cases} x_1^2 - x_2 - 3 = 0, \\ -x_1 + x_2^2 + 1 = 0. \end{cases}$$

   Compare the approximation with the solution $(2,1)^\top$.

Chapter 5

Numerical integration

5.1 Introduction

Determining the physical quantities of a system (for example volume, mass, or length) often involves the integral of a function. In most cases, an analytic evaluation of the integral is not possible. In such cases one has resort to numerical quadratures.

As an example, we investigate the production of a spoiler that is mounted onto the cabin of a truck (Figure 5.1). The shape of the spoiler is described by a sine function with a 2π meter period. The aluminium spoiler is made out of a flat plate by extrusion. The manufacturer wants to know the width of the plate such that the horizontal dimension of the spoiler will be 80 cm. The answer to this question is provided by the arc length of the curve

$$\begin{cases} x(t) = t, \\ y(t) = \sin t, \quad 0 \leq t \leq 0.8. \end{cases}$$

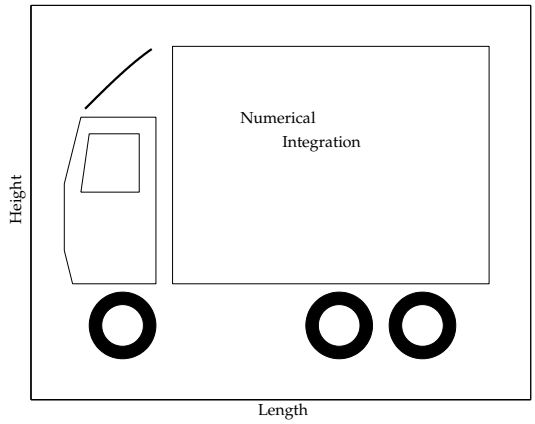


Figure 5.1: Truck with spoiler on the cabin.

To determine the arc length the formula

$$l = \int_0^{0.8} \sqrt{1 + \left(\frac{dy}{dt}\right)^2}\, dt = \int_0^{0.8} \sqrt{1 + (\cos t)^2}\, dt$$

is used. However, this integral cannot be evaluated in a simple way. In this chapter, numerical integration methods will be investigated in order to determine the arc length.

5.2 Riemann sums

To begin this chapter, the definition of an integral is recalled.

Definition 5.2.1 (Integral) *A* partition P_n *of* $[a,b]$ *is a finite number of distinct points* x_k *such that* $a = x_0 < x_1 < \cdots < x_n = b$. *Next,* T_n *is defined as a set of intermediate points* t_k *such that* $x_{k-1} \leq t_k \leq x_k$. *The length of an interval is denoted by* $h_k = x_k - x_{k-1}$ *and the mesh width by* $m(P_n) = \max_{1\leq k\leq n}\{h_k\}$. *The* Riemann sum *for a function* $f \in C[a,b]$ *is defined as*

$$R(f, P_n, T_n) = \sum_{k=1}^{n} h_k f(t_k).$$

Let a sequence of partitions, $P_1, P_2, \ldots$, *and corresponding* $T_1, T_2, \ldots$, *be given, such that* $\lim_{n\to\infty} m(P_n) = 0$. *We call* f Riemann integrable over $[a,b]$ *if*

$$R(f, P_n, T_n) \quad \textit{converges to a limit} \quad \mathcal{I} = \int_a^b f(x)dx.$$

Riemann sums are usually used to study integrability theoretically, but they are not very useful in practice. Numerical integration rules have a similar structure to Riemann sums:

$$I = \sum_{k=0}^{n} w_k f(t_k).$$

Here, t_k are called the *integration points*, and w_k are the corresponding *weights*. Numerical integration rules are also called *quadrature rules*.

5.3 Simple integration rules

In this section, the integration of a function f over a single interval $[x_L, x_R]$ will be considered. Several simple integration rules will be presented, together with their approximation errors. The effect of rounding errors will also be considered.

5.3.1 Rectangle rule

The most simple integration rule is called the *Rectangle rule*. Two different versions exist:

$$\int_{x_L}^{x_R} f(x)dx \approx (x_R - x_L)f(x_L), \text{ and } \int_{x_L}^{x_R} f(x)dx \approx (x_R - x_L)f(x_R),$$

which use the left and the right end point of the interval, respectively.

Theorem 5.3.1 *Let* $f \in C^1[x_L, x_R]$, *and* $m_1 = \max_{x\in[x_L,x_R]} |f'(x)|$. *Then*

$$\left|\int_{x_L}^{x_R} f(x)dx - (x_R - x_L)f(x_L)\right| \leq \frac{1}{2}m_1(x_R - x_L)^2,$$
$$\left|\int_{x_L}^{x_R} f(x)dx - (x_R - x_L)f(x_R)\right| \leq \frac{1}{2}m_1(x_R - x_L)^2.$$

Proof:
The proofs for the left and the right Rectangle rule follow the same structure. Therefore, only the left Rectangle rule will be considered. Using a Taylor expansion, we know that

$$f(x) = f(x_L) + (x - x_L)f'(\xi(x)) \quad \text{with } \xi(x) \in (x_L, x_R).$$

Integrating over $[x_L, x_R]$ yields

$$\int_{x_L}^{x_R} f(x)dx = \int_{x_L}^{x_R} \left\{ f(x_L) + (x - x_L)f'(\xi(x)) \right\} dx = (x_R - x_L)f(x_L) + \int_{x_L}^{x_R} (x - x_L)f'(\xi(x))dx.$$

Since $(x - x_L) \geq 0$, the mean-value theorem for integration can be used, which means that there exists an $\eta_L \in [x_L, x_R]$ such that

$$\int_{x_L}^{x_R} f(x)dx - (x_R - x_L)f(x_L) = f'(\eta) \int_{x_L}^{x_R} (x - x_L)dx = f'(\eta_L) \cdot \frac{1}{2}(x_R - x_L)^2.$$

Taking the absolute value on both sides and using m_1 completes the proof. ⊠

5.3.2 Midpoint rule

The *Midpoint rule* uses the integration point $x_M = (x_L + x_R)/2$, which leads to

$$\int_{x_L}^{x_R} f(x)dx \approx (x_R - x_L)f(x_M).$$

This rule is more accurate than expected at first sight.

Theorem 5.3.2 *Let $f \in C^2[x_L, x_R]$, and $m_2 = \max_{x \in [x_L, x_R]} |f''(x)|$. Then*

$$\left| \int_{x_L}^{x_R} f(x)dx - (x_R - x_L)f(x_M) \right| \leq \frac{1}{24} m_2 (x_R - x_L)^3.$$

Proof:
Using Taylor series,

$$f(x) = f(x_M) + (x - x_M)f'(x_M) + \frac{1}{2}(x - x_M)^2 f''(\xi(x)), \text{ with } \xi(x) \in (x_L, x_R).$$

Integrating over $[x_L, x_R]$ yields

$$\begin{aligned} \int_{x_L}^{x_R} f(x)dx &= \int_{x_L}^{x_R} \left(f(x_M) + (x - x_M)f'(x_M) + \frac{1}{2}(x - x_M)^2 f''(\xi(x)) \right) dx \\ &= (x_R - x_L)f(x_M) + \frac{1}{2} \int_{x_L}^{x_R} (x - x_M)^2 f''(\xi(x))dx. \end{aligned}$$

Since $(x - x_M)^2 \geq 0$, the mean-value theorem for integration can be used, which means that there exists an $\eta \in [x_L, x_R]$ such that

$$\int_{x_L}^{x_R} f(x)dx - (x_R - x_L)f(x_M) = \frac{1}{2} f''(\eta) \int_{x_L}^{x_R} (x - x_M)^2 dx.$$

Using the definition of x_M it follows that

$$\int_{x_L}^{x_R} f(x)dx - (x_R - x_L)f(x_M) = \frac{1}{24} f''(\eta)(x_R - x_L)^3.$$

Taking the absolute value on both sides and using m_2 completes the proof. ⊠

5.3.3 Trapezoidal rule

In the Trapezoidal rule, the linear interpolation polynomial on interpolation nodes x_L and x_R is used to approximate the integral of $f \in C[x_L, x_R]$ (see Section 2.2). This polynomial is given by

$$L_1(x) = \frac{x_R - x}{x_R - x_L} f(x_L) + \frac{x - x_L}{x_R - x_L} f(x_R).$$

The corresponding approximation of the integral is given by

$$\int_{x_L}^{x_R} f(x)dx \approx \int_{x_L}^{x_R} L_1(x)dx = \frac{x_R - x_L}{2}(f(x_L) + f(x_R)). \tag{5.1}$$

This approximation is called the *Trapezoidal rule*, since (5.1) equals the area of the trapezium with vertices $(x_L, 0)$, $(x_R, 0)$, $(x_R, f(x_R))$ and $(x_L, f(x_L))$.

Theorem 5.3.3 *Let $f \in C^2[x_L, x_R]$, and $m_2 = \max_{x \in [x_L, x_R]} |f''(x)|$. Then*

$$\left| \int_{x_L}^{x_R} f(x)dx - \frac{x_R - x_L}{2}(f(x_L) + f(x_R)) \right| \leq \frac{1}{12} m_2 (x_R - x_L)^3.$$

Proof:
From Theorem 2.2.1 it follows that linear interpolation through $(x_L, f(x_L))$ and $(x_R, f(x_R))$ has a truncation error

$$f(x) - L_1(x) = \frac{1}{2}(x - x_L)(x - x_R) f''(\xi(x)), \text{ with } \xi(x) \in (x_L, x_R). \tag{5.2}$$

Integrating both sides of equation (5.2) yields

$$\int_{x_L}^{x_R} f(x)dx - \frac{x_R - x_L}{2}(f(x_L) + f(x_R)) = \frac{1}{2} \int_{x_L}^{x_R} (x - x_L)(x - x_R) f''(\xi(x))dx. \tag{5.3}$$

Because $(x - x_L)(x - x_R) \leq 0$ for $x \in [x_L, x_R]$, the mean-value theorem for integration can be used, and there exists an $\eta \in [x_L, x_R]$ such that

$$\frac{1}{2} \int_{x_L}^{x_R} (x - x_L)(x - x_R) f''(\xi(x))dx = \frac{1}{2} f''(\eta) \int_{x_L}^{x_R} (x - x_L)(x - x_R)dx = -\frac{1}{12} f''(\eta)(x_R - x_L)^3. \tag{5.4}$$

Using equation (5.4) in equation (5.3), taking the absolute value and using m_2 completes the proof. ⊠

5.3.4 Simpson's rule

Using a quadratic interpolation polynomial with interpolation nodes x_L, $x_M = (x_R - x_L)/2$, and x_R, the *Simpson's rule* can be derived. The integral of f is approximated by

$$\int_{x_L}^{x_R} f(x)dx \approx \frac{x_R - x_L}{6}(f(x_L) + 4f(x_M) + f(x_R)).$$

The corresponding remainder term will be presented without proof.

Theorem 5.3.4 *Let $f \in C^4[x_L, x_R]$ and $m_4 = \max_{x \in [x_L, x_R]} |f^{(4)}(x)|$, then*

$$\left| \int_{x_L}^{x_R} f(x)dx - \frac{x_R - x_L}{6}(f(x_L) + 4f(x_M) + f(x_R)) \right| \leq \frac{1}{2880} m_4 (x_R - x_L)^5.$$

5.4 Composite rules

The integration rules that have been presented in Section 5.3 usually generate errors that are larger than the required accuracy allows. In this section, more accurate approximations will be obtained by applying a composite integration rule.

The integral of $f \in C[a,b]$ will be approximated using a subdivision $a = x_0 < x_1 < \ldots < x_n = b$, with $x_k = a + kh$, $k = 0, \ldots, n$, and $h = (b-a)/n$. Using that

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx,$$

the integration rules of Section 5.3 can be applied on each subinterval $[x_{k-1}, x_k]$. Here, $x_L = x_{k-1}$, $x_R = x_k$, and $x_M = (x_{k-1} + x_k)/2$. If I_k is an approximation of the integral $\int_{x_{k-1}}^{x_k} f(x)dx$, for $k = 1, \ldots, n$, then the integral over $[a,b]$ can be approximated by

$$\int_a^b f(x)dx = \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)dx \approx \sum_{k=1}^{n} I_k = I.$$

In the following theorem, an upper bound for the remainder term of a composite rule will be considered.

Theorem 5.4.1 *Let f be given on $[a,b]$, and let $a = x_0 < x_1 < \ldots < x_n = b$, with $x_k = a + kh$, and $h = (b-a)/n$. Let I_k be an approximation of the integral $\int_{x_{k-1}}^{x_k} f(x)dx$, $k = 1, \ldots, n$. Assume that the absolute value of the remainder term of I_k has upper bound $c_k \cdot h^{p+1}$, with $p \in \mathbb{N}$. Define $c = \max\{c_1, \ldots, c_n\}$, then*

$$\left| \int_a^b f(x)dx - I \right| \leq c(b-a)h^p.$$

Proof:
Using the triangle inequality,

$$\left| \int_a^b f(x)dx - I \right| = \left| \sum_{k=1}^{n} \left(\int_{x_{k-1}}^{x_k} f(x)dx - I_k \right) \right| \leq \sum_{k=1}^{n} \left| \int_{x_{k-1}}^{x_k} f(x)dx - I_k \right|.$$

Using the remainder term for each I_k yields

$$\left| \int_a^b f(x)dx - I \right| \leq \sum_{k=1}^{n} c_k h^{p+1} \leq \sum_{k=1}^{n} ch^{p+1} = c \cdot n \cdot h^{p+1}.$$

Finally, we use that $n \cdot h = b - a$ to arrive at the given upper bound. ⊠

In the remainder of this section, the integration rules of Section 5.3 will be extended to their composite rules in order to approximate $\int_a^b f(x)dx$.

Rectangle rule
The left and right composite Rectangle rules are given by

$$I_L = h \sum_{k=0}^{n-1} f(x_k) = h\left(f(a) + f(a+h) + \ldots + f(b-h)\right),$$

$$I_R = h \sum_{k=1}^{n} f(x_k) = h\left(f(a+h) + f(a+2h) + \ldots + f(b)\right).$$

From Theorem 5.3.1 it follows that for each interval $[x_{k-1}, x_k]$, $c_k = \max_{x\in[x_{k-1},x_k]} |f'(x)|$. Defining $M_1 = \max_{x\in[a,b]} |f'(x)| = \max\{c_1, \ldots, c_n\}$, the corresponding upper bounds for the remainder terms are (use Theorem 5.4.1)

$$\left| \int_a^b f(x)dx - I_L \right| \leq \frac{1}{2} M_1 (b-a)h,$$
$$\left| \int_a^b f(x)dx - I_R \right| \leq \frac{1}{2} M_1 (b-a)h.$$

Midpoint rule
The composite Midpoint rule is given by

$$I_M = h \sum_{k=1}^{n} f\left(\frac{x_{k-1}+x_k}{2}\right) = h\left(f(a+\frac{1}{2}h) + f(a+\frac{3}{2}h) + \ldots + f(b-\frac{1}{2}h)\right).$$

Defining $M_2 = \max_{x\in[a,b]} |f''(x)|$, the upper bound for the remainder term (use Theorems 5.3.2 and 5.4.1) is

$$\left| \int_a^b f(x)dx - I_M \right| \leq \frac{1}{24} M_2 (b-a)h^2.$$

Trapezoidal rule
For the Trapezoidal rule, the composite version is

$$I_T = \frac{h}{2} \sum_{k=1}^{n} (f(x_{k-1}) + f(x_k)) = h\left(\frac{1}{2}f(a) + f(a+h) + \ldots + f(b-h) + \frac{1}{2}f(b)\right),$$

with remainder term (use Theorems 5.3.3 and 5.4.1)

$$\left| \int_a^b f(x)dx - I_T \right| \leq \frac{1}{12} M_2 (b-a)h^2,$$

where, again, $M_2 = \max_{x\in[a,b]} |f''(x)|$.

Simpson's rule
The repeated Simpson's rule is given by

$$\begin{aligned} I_S &= \frac{h}{6} \sum_{k=1}^{n} \left(f(x_{k-1}) + 4f\left(\frac{x_{k-1}+x_k}{2}\right) + f(x_k) \right) \\ &= h\left(\frac{1}{6}f(a) + \frac{2}{3}f(a+\frac{1}{2}h) + \frac{1}{3}f(a+h) + \frac{2}{3}f(a+\frac{3}{2}h) + \ldots + \frac{2}{3}f(b-\frac{1}{2}h) + \frac{1}{6}f(b)\right), \end{aligned}$$

with remainder term

$$\left| \int_a^b f(x)dx - I_S \right| \leq \frac{1}{2880} M_4 (b-a)h^4,$$

with $M_4 = \max_{x\in[a,b]} |f^{(4)}(x)|$.

Remarks

- As a consequence of the error estimates, the Rectangle rules are exact if f is a constant, the Midpoint and Trapezoidal rule are exact for linear polynomials, and Simpson's rule is exact for polynomials of degree 3.

- The remainder terms of the composite Midpoint and Trapezoidal rule are of order $\mathcal{O}(h^2)$, whereas the Rectangle rules have remainder terms of order $\mathcal{O}(h)$. The composite Simpson's rule has a remainder term of order $\mathcal{O}(h^4)$. The Rectangle, Midpoint and Trapezoidal rules all require the same amount of work. Among these, the Midpoint and Trapezoidal rules are clearly preferred. The composite Simpson's rule needs twice as many function evaluations, but converges faster.
- Note that the upper bound for the error of the Trapezoidal rule is twice as large as the bound for the Midpoint rule.
- The composite Trapezoidal rule may also be interpreted as the integral of the linear spline approximation of f. Suppose s is the linear spline approximation of f in $a = x_0, \ldots, x_n = b$. Then the composite Trapezoidal rule computes $I_T = \int_a^b s(x)dx$.

Example 5.4.1 (Spoiler)

For the example mentioned in the introduction, the length of the flat plate should be approximated with an error of at most 1 cm. Defining $f(t) = \sqrt{1 + (\cos t)^2}$, this length equals $\int_0^{0.8} f(t)dt$. Applying the composite left Rectangle rule, the upper bound for the remainder term equals

$$\left| \int_0^{0.8} f(t)dt - I_L \right| \leq \frac{1}{2} M_1 \cdot 0.8h,$$

where $M_1 = \max_{t \in [0,0.8]} |f'(t)|$. The derivative of the integrand is given by

$$f'(t) = \frac{-\cos t \sin t}{\sqrt{1 + (\cos t)^2}} = \frac{-\frac{1}{2} \sin 2t}{\sqrt{1 + (\cos t)^2}}.$$

From this it follows that

$$|f'(t)| \leq \frac{1}{2} = M_1.$$

Requiring that the error is smaller than 1 cm, h should be chosen such that

$$\frac{1}{2} M_1 \cdot 0.8h \leq \frac{1}{4} \cdot 0.8h \leq 0.01.$$

Step size $h = 0.05$ meets this requirement.

The exact value of the integral equals 1.0759354360688... m. For $h = 0.05$, the composite left Rectangle rule computes 1.0807 m, hence the error is, in fact, less than 1 cm.

In Table 5.1 the errors are tabulated for different values of the step size h. Because the Rectangle rule is $\mathcal{O}(h)$, we expect that by halving h, the error of the Rectangle rule decreases by a factor 2. The results are in agreement with these expectations. Similarly, the Midpoint rule and Trapezoidal rule errors decrease by a factor 4, as expected because the methods are $\mathcal{O}(h^2)$. Finally, the Simpson's rule errors decrease by a factor 16, because the method is $\mathcal{O}(h^4)$.

Table 5.1: The error $\left| \int_0^{0.8} f(t)dt - I \right|$ for $f(t) = \sqrt{1 + (\cos t)^2}$, using different composite integration rules I.

h	Left Rectangle rule	Midpoint rule	Trapezoidal rule	Simpson's rule
0.8	$5.5435 \cdot 10^{-2}$	$1.1698 \cdot 10^{-2}$	$2.2742 \cdot 10^{-2}$	$2.1789 \cdot 10^{-4}$
0.4	$3.3567 \cdot 10^{-2}$	$2.7815 \cdot 10^{-3}$	$5.5221 \cdot 10^{-3}$	$1.3618 \cdot 10^{-5}$
0.2	$1.8174 \cdot 10^{-2}$	$6.8643 \cdot 10^{-4}$	$1.3703 \cdot 10^{-3}$	$8.4852 \cdot 10^{-7}$
0.1	$9.4302 \cdot 10^{-3}$	$1.7105 \cdot 10^{-4}$	$3.4194 \cdot 10^{-4}$	$5.2978 \cdot 10^{-8}$
0.05	$4.8006 \cdot 10^{-3}$	$4.2728 \cdot 10^{-5}$	$8.5445 \cdot 10^{-5}$	$3.3102 \cdot 10^{-9}$

5.5 Measurement and rounding errors

Measurement and rounding errors can play an important role in numerical quadrature rules. In this section, this error behavior will be investigated. Therefore, we assume that the function values are perturbed by an error $\varepsilon > 0$, such that

$$|f(x) - \hat{f}(x)| = \varepsilon(x).$$

Defining $\varepsilon_{\max} = \max_{x\in[a,b]} \varepsilon(x)$ it follows that

$$\left| \int_a^b \Big(f(x) - \hat{f}(x) \Big)\, dx \right| \leq \int_a^b |f(x) - \hat{f}(x)| dx \leq \varepsilon_{\max}(b-a).$$

Next, the integral is approximated with an integration rule using the perturbed values. In this derivation, the composite left Rectangle rule will be used. The difference between the exact integral, $\mathcal{I} = \int_a^b f(x)dx$, and the perturbed approximation, $\hat{I}_L$, can be bounded by

$$\begin{aligned} |\mathcal{I} - \hat{I}_L| &= |\mathcal{I} - I_L + I_L - \hat{I}_L| \leq |\mathcal{I} - I_L| + |I_L - \hat{I}_L| \\ &= |\mathcal{I} - I_L| + \left| h\sum_{k=0}^{n-1} f(x_k) - h\sum_{k=0}^{n-1} \hat{f}(x_k) \right| \\ &\leq |\mathcal{I} - I_L| + h\sum_{k=0}^{n-1} |f(x_k) - \hat{f}(x_k)| = |\mathcal{I} - I_L| + h\sum_{k=0}^{n-1} \varepsilon(x_k). \end{aligned}$$

Assuming $\varepsilon(x) \leq \varepsilon_{\max}$, the total error for the composite left Rectangle rule is

$$\begin{aligned} \left| \int_a^b f(x)dx - \hat{I}_L \right| &\leq \frac{1}{2} M_1 (b-a)h + h\sum_{k=0}^{n-1} \varepsilon_{\max} \\ &= \frac{1}{2} M_1 (b-a)h + h \cdot n\varepsilon_{\max} = \left(\frac{1}{2} M_1 h + \varepsilon_{\max} \right)(b-a), \end{aligned}$$

where $M_1 = \max_{x\in[a,b]} |f'(x)|$. Note that it is useless to take h any smaller than $2\varepsilon_{\max}/M_1$, because then the measurement error dominates the total error.

Note that this derivation is specific for the composite left Rectangle rule. Similar error estimates can be derived for the Midpoint, Trapezoidal and Simpson's rule.

Example 5.5.1 (Spoiler)

The computation of the length of the flat plate can be perturbed assuming a small manufacturing error. Consequently, the integrand contains a rounding error $\varepsilon(x) = 10^{-3}$. In Figure 5.2, the effect of rounding errors can be seen for different values of h (using the composite left Rectangle rule). It turns out that the total error remains larger than 0.8×10^{-3}. It serves no purpose to take the step size smaller than $4\varepsilon_{\max} = 4 \cdot 10^{-3}$.

Conditioning of the integral
The numerical approximation of an integral $\mathcal{I}$ is either a well-conditioned or an ill-conditioned problem. If $\epsilon_{\max}$ is an upper bound for the relative error (different from $\varepsilon_{\max}$ for the absolute error), then the absolute error in the function values is bounded by the inequality

$$|f(x) - \hat{f}(x)| \leq |f(x)|\epsilon_{\max}.$$

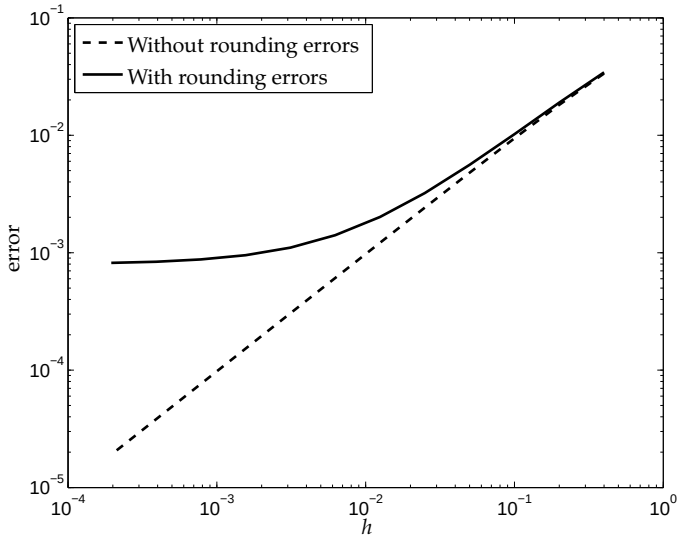


Figure 5.2: The errors $\left|\int_0^{0.8} f(t)dt - I_L\right|$ (without rounding errors) and $\left|\int_0^{0.8} f(t)dt - \hat{I}_L\right|$ (with rounding errors) for $f(t) = \sqrt{1+(\cos t)^2}$, using $\varepsilon(x) = 10^{-3}$, and the composite left Rectangle rule.

In that case, an upper bound for the relative error in the integral is given by

$$\frac{\left|\int_a^b f(x)dx - \int_a^b \hat{f}(x)dx\right|}{\left|\int_a^b f(x)dx\right|} \leq \frac{\int_a^b |f(x)|dx}{\left|\int_a^b f(x)dx\right|} \cdot \epsilon_{\max}.$$

The value

$$K_{\mathcal{I}} = \frac{\int_a^b |f(x)|dx}{\left|\int_a^b f(x)dx\right|}$$

is called the *condition number* of the integral $\mathcal{I}$. If $K_{\mathcal{I}} \cdot \epsilon_{\max} = \mathcal{O}(1)$, then the determination of $\mathcal{I}$ is *ill conditioned*: the relative error in the computed integral is of order $\mathcal{O}(1)$.

Example 5.5.2 (Profits)

The profits or losses per day of a car manufacturer depend on the season. The following profits formula (in billions of $) are assumed:

$$w_{\text{spring}}(t) = 0.01 + \sin\left(\pi t - \frac{\pi}{2}\right),\ t \in [0,1], \text{ and } w_{\text{fall}}(t) = 0.01 + \sin\left(\pi t + \frac{\pi}{2}\right),\ t \in [0,1].$$

The total profits in spring (W_{spring}) and fall (W_{fall}) equal

$$W_{\text{spring}} = \int_0^1 w_{\text{spring}}(t)dt = 0.01, \quad W_{\text{fall}} = \int_0^1 w_{\text{fall}}(t)dt = 0.01,$$

which equal 10 million $.

The composite left Rectangle rule using $h = 0.02$ approximates W_{spring} with -0.01 and W_{fall} with 0.03. The composite Midpoint and Trapezoidal rule obtain the exact results for $h = 0.02$.

Note that

$$W_{\text{spring}} = |W_{\text{spring}}| = 0.01 \text{ and } \int_0^1 |w_{\text{spring}}(t)|dt \simeq 0.64.$$

Therefore, the condition number of W_{spring} equals $K_{\mathcal{I}} = 64$. If the function values only have an accuracy of two digits, then $\epsilon_{\max} = 0.005$. This means that $K_{\mathcal{I}} \cdot \epsilon_{\max} = 0.32$, such that the determination of the integral is an ill-conditioned problem. In the worst case, no digit of the approximation is correct.

5.6 Interpolatory quadrature rules *

Subsequently, quadrature rules based on higher-order interpolation will be discussed. Newton-Cotes quadrature rules will be considered as a particular case. In this section, integration over a single interval $[x_L, x_R]$ will be considered.

General quadrature rules
Let $x_0, \ldots, x_N \in [x_L, x_R]$. Let L_N be the Lagrange interpolation polynomial of f at the nodes $x_0, \ldots, x_N$, as defined in Section 2.3, equations (2.7) and (2.8). Then

$$\int_{x_L}^{x_R} f(x)dx \approx \int_{x_L}^{x_R} L_N(x)dx = \int_{x_L}^{x_R} \sum_{\ell=0}^{N} f(x_\ell) L_{\ell N}(x)dx = \sum_{\ell=0}^{N} w_\ell f(x_\ell),$$

using the weights

$$w_\ell = \int_{x_L}^{x_R} L_{\ell N}(x)dx, \tag{5.5}$$

which do not depend on f.

Theorem 5.6.1 *Let $x_0, \ldots, x_N \in [x_L, x_R]$, and let f be a polynomial of degree at most N. Then every $N+1$-point quadrature rule based on interpolation is exact for f:*

$$\int_{x_L}^{x_R} f(x)dx = \sum_{\ell=0}^{N} w_\ell f(x_\ell).$$

Proof:
From Theorem 2.3.1 it follows that f coincides with its Lagrange interpolation polynomial on $[x_L, x_R]$, and hence,

$$\int_{x_L}^{x_R} f(x)dx = \int_{x_L}^{x_R} L_N(x)dx.$$

⊠

Conversely:

Theorem 5.6.2 *If $x_0, \ldots, x_N$ and $w_0, \ldots, w_N$ are given and if*

$$\int_{x_L}^{x_R} p(x)dx = \sum_{\ell=0}^{N} w_\ell p(x_\ell)$$

holds for all polynomials p of degree at most N, then $\sum_{\ell=0}^{N} w_\ell f(x_\ell)$ is the quadrature rule based on interpolation for $\int_{x_L}^{x_R} f(x)dx$.

Proof:
Let L_N be the interpolation polynomial of f on the nodes $x_0, \ldots, x_N$. Then, by definition,

$$\int_{x_L}^{x_R} L_N(x)dx = \sum_{\ell=0}^{N} w_\ell L_N(x_\ell) = \sum_{\ell=0}^{N} w_\ell f(x_\ell),$$

and equation (5.5) uniquely determines the weights $w_0, \ldots, w_N$.

Note that the verification for L_N is already enough to prove the theorem. ⊠

The following theorem can be used to compute remainder terms for interpolatory quadrature rules. The proof of this theorem will be omitted, and can be found in [9].

Theorem 5.6.3 *Let $\int_{x_L}^{x_R} f(x)dx$ be approximated by the integration of the Lagrange interpolation polynomial L_N. Then there exists a $\xi \in [x_L, x_R]$ such that the remainder term of the approximation satisfies:*

if N is even and $f \in C^{N+2}[x_L, x_R]$:

$$\left| \int_{x_L}^{x_R} f(x)dx - \int_{x_L}^{x_R} L_n(x)dx \right| = C_N \left(\frac{x_R - x_L}{N} \right)^{N+3} f^{(N+2)}(\xi) \quad \textit{with}$$

$$C_N = \frac{1}{(N+2)!} \int_0^N t^2(t-1)\cdots(t-N)dt,$$

if N is odd and $f \in C^{N+1}[x_R, x_L]$:

$$\left| \int_{x_L}^{x_R} f(x)dx - \int_{x_L}^{x_R} L_N(x)dx \right| = D_N \left(\frac{x_R - x_L}{N} \right)^{N+2} f^{(N+1)}(\xi) \quad \textit{with}$$

$$D_N = \frac{1}{(N+1)!} \int_0^N t(t-1)\cdots(t-N)dt.$$

Remarks

- As a result of the theorem, interpolatory quadrature rules integrate polynomials up to degree $N+1$ (if N is even) or N (if N is odd) exactly.
- Composite interpolatory quadrature rules are found by repeating the interpolation on subintervals, as has been explained in Section 5.4.

Definition 5.6.1 (Newton-Cotes quadrature rules) *If $x_L = x_0 < x_1 < \ldots < x_N = x_R$ and the nodes are equidistantly distributed, then the sum $\sum_{\ell=0}^{N} w_\ell f(x_\ell)$ is called the* Newton-Cotes quadrature rule *for $\int_{x_L}^{x_R} f(x)dx$.*

The Trapezoidal rule (Section 5.3.3) uses linear interpolation in the nodes x_L and x_R. This means that the Trapezoidal rule is a Newton-Cotes quadrature rule with $N = 1$. Similarly, the Simpson's rule (Section 5.3.4) uses quadratic interpolation in the nodes x_L, $(x_L + x_R)/2$ and x_R: this is a Newton-Cotes quadrature rule using $N = 2$. Indeed, the error terms of the Trapezoidal and the Simpson's rule correspond to Theorem 5.6.3.

Newton-Cotes quadrature rules of higher order are rarely used: negative weights will occur and that is less desirable.

5.7 Gauss quadrature rules *

In the previous section, quadrature rules have been derived by choosing the nodes $x_0, \ldots, x_N$ beforehand and subsequently determining the weights $w_0, \ldots, w_N$. These weights are computed by integrating the polynomial that interpolates f on the nodes x_ℓ, $\ell = 0, \ldots, N$. These rules integrate polynomials up to degree N (or $N+1$) exactly.

Alternatively, the weights and the nodes can be determined together, in such a way that polynomials of a degree as high as possible will be integrated exactly. The $2N+2$ unknown values of w_ℓ and x_ℓ can be used to generate a system of $2N+2$ nonlinear equations such that the polynomials $1, x, \ldots, x^{2N+1}$ are integrated exactly. It can be shown that this system has a unique solution. The resulting quadrature formulae are called *Gauss quadrature rules*. The following illustration explains the computation of a two-point Gauss quadrature rule. Subsequently, the general approach will be explained. In the derivation, the interval $[-1, 1]$ is used. The procedure for general intervals $[x_L, x_R]$ will be treated later on.

The two-point Gauss quadrature rule requires weights w_0, w_1, and points x_0, x_1 such that the quadrature formula

$$\int_{-1}^{1} f(x)dx \approx w_0 f(x_0) + w_1 f(x_1)$$

computes the exact result when f is a polynomial of degree at most 3. Using f equal to $1, x, x^2$ and x^3 consecutively, this means that w_0, w_1, x_0 and x_1 must satisfy

$$\begin{aligned}
f(x) = 1 \quad &\Rightarrow \quad w_0 \quad + w_1 \quad = \int_{-1}^{1} 1dx \; = 2, \\
f(x) = x \quad &\Rightarrow \quad w_0 x_0 \; + w_1 x_1 = \int_{-1}^{1} x dx \; = 0, \\
f(x) = x^2 \quad &\Rightarrow \quad w_0 x_0^2 \; + w_1 x_1^2 = \int_{-1}^{1} x^2 dx = \tfrac{2}{3}, \\
f(x) = x^3 \quad &\Rightarrow \quad w_0 x_0^{(3)} + w_1 x_1^3 = \int_{-1}^{1} x^3 dx = 0.
\end{aligned}$$

By symmetry of the equations, $w_0 = w_1$, and therefore $w_0 = w_1 = 1$. Then it follows that

$$x_0 = -\frac{\sqrt{3}}{3}, \;\; x_1 = \frac{\sqrt{3}}{3},$$

hence the quadrature formula becomes

$$\int_{-1}^{1} f(x)dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right).$$

For Gauss quadrature formulae, the integration limits are always taken equal to $x_L = -1$ and $x_R = 1$. Consequently, Gauss quadrature rules have the following general form

$$\int_{-1}^{1} f(x)dx \approx \sum_{\ell=0}^{N} w_\ell f(x_\ell) = I_G,$$

in which the $2N+2$ parameters x_ℓ and w_ℓ, $\ell = 0, \ldots, N$ are such that

$$\int_{-1}^{1} x^j dx = \sum_{\ell=0}^{N} w_\ell x_\ell^j, j = 0, \cdots, 2N+1$$

Table 5.2: Weights and integration points for Gauss quadrature rules.

N	x_ℓ	w_ℓ
0	0	2
1	$-\frac{1}{3}\sqrt{3}$	1
	$\frac{1}{3}\sqrt{3}$	1
2	$-\frac{1}{5}\sqrt{15}$	$\frac{5}{9}$
	0	$\frac{8}{9}$
	$\frac{1}{5}\sqrt{15}$	$\frac{5}{9}$
3	$-\frac{1}{35}\sqrt{525+70\sqrt{30}}$	$\frac{1}{36}(18-\sqrt{30})$
	$-\frac{1}{35}\sqrt{525-70\sqrt{30}}$	$\frac{1}{36}(18+\sqrt{30})$
	$\frac{1}{35}\sqrt{525-70\sqrt{30}}$	$\frac{1}{36}(18+\sqrt{30})$
	$\frac{1}{35}\sqrt{525+70\sqrt{30}}$	$\frac{1}{36}(18-\sqrt{30})$

holds. Tables with Gauss integration points and corresponding weights can be found in many text books on numerical analysis. In Table 5.2, the values are given for the first four Gauss quadrature rules. Note that the Gauss quadrature rule for $N = 0$ corresponds to the Midpoint rule.

In order to integrate a function numerically over an arbitrary interval $[x_L, x_R]$ using a Gauss quadrature rule, the interval of integration has to be transformed to $[-1, 1]$. This change of interval is achieved by linear transformation using

$$\int_{x_L}^{x_R} f(y)dy = \frac{x_R - x_L}{2} \int_{-1}^{1} f\left(\frac{x_R - x_L}{2}x + \frac{x_L + x_R}{2}\right) dx. \tag{5.6}$$

Replacing the latter integral by a Gauss formula yields

$$\int_{x_L}^{x_R} f(y)dy \approx \frac{x_R - x_L}{2} \sum_{\ell=0}^{N} w_\ell f\left(\frac{x_R - x_L}{2}x_\ell + \frac{x_L + x_R}{2}\right). \tag{5.7}$$

The integration points x_ℓ correspond to the Gauss integration points as tabulated in Table 5.2.

If $f \in C^{2N+2}[x_L, x_R]$, then the $N+1$-point Gauss quadrature rule has the remainder term

$$\int_{x_L}^{x_R} f(y)dy - \frac{x_R - x_L}{2} \sum_{\ell=0}^{N} w_\ell f\left(\frac{x_R - x_L}{2}x_\ell + \frac{x_L + x_R}{2}\right) = \frac{(x_R - x_L)^{2N+3}((N+1)!)^4}{(2N+3)((2N+2)!)^3} f^{(2N+2)}(\xi),$$

with $\xi \in (x_L, x_R)$. This formula shows that Gauss integration rules are indeed exact for polynomials up to degree $2N + 1$.

Remark
Composite Gauss quadrature rules are found by repeating the Gauss quadrature rule on subintervals, as has been explained in Section 5.4.

Example 5.7.1 (Polynomial)

In this example, the value of $\int_1^{1.5} y^7 dy$ will be approximated. The exact value of the integral equals 3.07861328125. Using a change of interval as in equations (5.6) and (5.7), the integral is approximated by

$$\int_1^{1.5} f(y)dy \approx 0.25 \sum_{\ell=0}^{N} w_\ell f(0.25x_\ell + 1.25),$$

where $f(y) = y^7$, and x_ℓ are the Gauss integration points.

In Table 5.3, the errors are shown for different values of N. As expected, the 4-point Gauss quadrature rule ($N = 3$) gives the exact result. Note that also for smaller numbers of integration points, the Gauss quadrature rule is very accurate.

Table 5.3: The error $\left|\int_1^{1.5} y^7 dy - I_G\right|$, using different $N+1$-point Gauss integration rules I_G.

N	$\lvert \int_1^{1.5} y^7 dy - I_G \rvert$
0	$6.9443 \cdot 10^{-1}$
1	$1.1981 \cdot 10^{-2}$
2	$2.4414 \cdot 10^{-5}$
3	0

5.8 Summary

In this chapter, the following subjects have been discussed:

- Riemann sums;
- Simple integration rules (left and right Rectangle rule, Midpoint rule, Trapezoidal rule, Simpson's rule);
- Composite rules;
- Measurement and rounding errors;
- Interpolatory quadrature rules;
- Newton-Cotes quadrature rules;
- Gauss quadrature rules.

5.9 Exercises

1. In this exercise, the integral
$$\int_{-1}^{1} ((10x)^3 + 0.001)dx$$
should be computed.
   (a) The relative rounding error in the function values is less than ε. Determine the relative error in the integral due to the rounding errors.
   (b) Use the composite Midpoint rule and $\varepsilon = 4 \times 10^{-8}$. Give a reasonable value for the step size h.

2. Determine $\int_{0.5}^{1} x^4 dx$ with the Trapezoidal rule. Estimate the error and compare it with the real error. Also compute the integral with the composite Trapezoidal rule using step size $h = 0.25$. Estimate the error using Richardson's extrapolation.

3. Calculate $\int_1^{1.5} x^7 dx$ with the composite Trapezoidal rule using two equal intervals and with the Simpson's rule. Compare your results with the results for Gauss quadrature rules as tabulated in Table 5.3.

Chapter 6

Numerical time integration of initial-value problems

6.1 Introduction

An initial-value problem usually is a mathematical description (differential equation) of a time-dependent problem. The conditions, necessary to determine a unique solution, are given at the starting time, t_0, and are called *initial conditions*.

As an example, we consider a water discharge from a storage reservoir through a pipe (Figure 6.1). The water is at rest until, at $t_0 = 0$, the latch is instantaneously opened. The inertia of the water causes a gradual development of the flow. If this flowing water is used to generate electricity, then it is important to know how long it will take before the turbines work at full power.

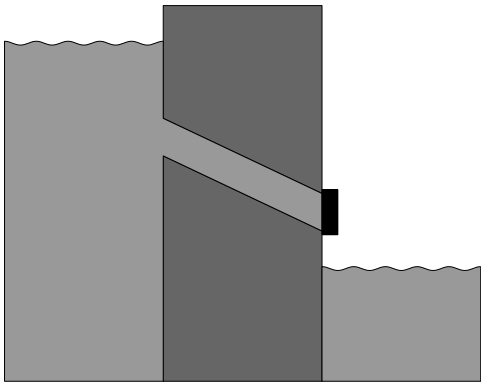


Figure 6.1: The storage reservoir, the drain pipe and the latch in closed position.

This process is described by the nonlinear initial-value problem

$$\begin{cases} \frac{dq}{dt} = p(t) - aq^2, \quad t > 0, \\ q(0) = 0. \end{cases} \tag{6.1}$$

In this problem, $q(m^3/s)$ is the mass flow rate, p is the driving force, which is equal to

$$\frac{\text{force/length}}{\text{density}} \quad \text{and measured in} \quad \frac{N/m}{kg/m^3} = m^3/s^2,$$

and aq^2 is the friction. The driving force depends –among other things– on the water level in the reservoir. Assuming that $p(t)$ is constant ($p(t) = p_0$), a solution in closed form is known:

$$q(t) = \sqrt{\frac{p_0}{a}} \tanh(t\sqrt{ap_0}).$$

For general functions p, an analytic solution cannot be obtained. In that case, a numerical method must be used to approximate the solution.

6.2 Theory of initial-value problems

In this section, a short summary of the theory of initial-value problems will be presented. Before applying a numerical method for solving an initial-value problem, it is important to make sure that:

- The initial-value problem has a solution;
- The solution to the initial-value problem is unique;
- The solution changes continuously when the parameters in the differential equation or the initial conditions are slightly perturbed.

A theorem about these issues uses the concepts of a *well-posed* initial-value problem (see also definition 1.6.1) and *Lipschitz continuity*.

Definition 6.2.1 (Well posedness) *The initial-value problem*

$$\begin{cases} \frac{dy}{dt} = f(t,y), \quad a \le t \le b, \\ y(a) = y_a \end{cases} \tag{6.2}$$

is called well posed *if the problem has a unique solution, that depends continuously on the data (a, y_a and f).*

Definition 6.2.2 (Lipschitz continuity) *A function $f(t,y)$ is called* Lipschitz continuous *in y on a set $D \subset \mathbb{R}^2$, if there exists a constant $L > 0$ such that*

$$|f(t,y_1) - f(t,y_2)| \le L|y_1 - y_2|, \text{ for all } (t,y_1), (t,y_2) \in D.$$

The constant L is called the Lipschitz constant.

The following theorem yields a sufficient condition for a function to be Lipschitz continuous. The proof uses the intermediate-value theorem.

Theorem 6.2.1 *If the function f is differentiable with respect to the variable y, then Lipschitz continuity is implied if*

$$\left|\frac{\partial f}{\partial y}(t,y)\right| \le L, \text{ for all } (t,y) \in D,$$

for some $L > 0$.

The following theorem presents a sufficient condition for an initial-value problem to be well posed. The proof can be found in many text books on differential equations, for example in Boyce and DiPrima, [4].

Theorem 6.2.2 *Suppose that $D = \{(t,y)|a \le t \le b, -\infty < y < \infty\}$ and that $f(t,y)$ is continuous on D. If f is Lipschitz continuous in the variable y on D, then the initial-value problem (6.2) is well posed.*

6.3 Elementary single-step methods

In contrast to the example in Section 6.1, for many initial-value problems it is impossible to determine a solution explicitly. Although there are generally applicable techniques to find qualitative properties of the solution, these properties usually provide insufficient quantitative information. For that purpose, numerical methods are designed to approximate the solution. Most numerical methods are designed for first-order initial-value problems. Numerical time-integration methods that can be applied directly to higher-order initial-value problems do exist but are not treated in this book.

In this section, several methods will be presented that can be used to approximate the solution to the initial-value problem based on the scalar first-order differential equation

$$\begin{cases} y' = f(t,y), & t > t_0, \\ y(t_0) = y_0. \end{cases}$$

Therefore, it is illuminating to consider the slope field generated by the differential equation. A differential equation is a rule that provides the slope of the tangent of a solution curve at each point in the (t,y)-plane. A specific solution curve is selected by specifying a point through which that solution curve passes. A formal expression for the solution curve through a point (t_0, y_0) is generated by integrating the differential equation over t:

$$y(t) = y(t_0) + \int_{t_0}^{t} f(\tau, y(\tau))d\tau.$$

Because the unknown function y appears under the integral sign, this equation is called an *integral equation*. Although the integral equation is as difficult to solve as the original initial-value problem, it provides a better starting point for approximating the solution.

In order to approximate the solution to the integral equation numerically, the time axis is divided into a set of discrete points $t_n = t_0 + n\Delta t$, $n = 0, 1, \ldots$. The solution at each time step is computed using the integral equation. Assume that the solution to the differential equation at time t_n is known, and denoted by $y_n = y(t_n)$. Then, the solution at t_{n+1} is equal to

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{6.3}$$

The numerical time-integration methods that are presented in this section can be characterized by the approximation of the integral term $\int_{t_n}^{t_{n+1}} f(t, y(t))dt$: different numerical quadrature rules (Chapter 5) can be used. Because integral equation (6.3) only considers one time interval $[t_n, t_{n+1}]$, these numerical methods are called *single-step methods*. Multi-step methods, which may also use $t_0, \ldots, t_{n-1}$, will be described in Section 6.11.

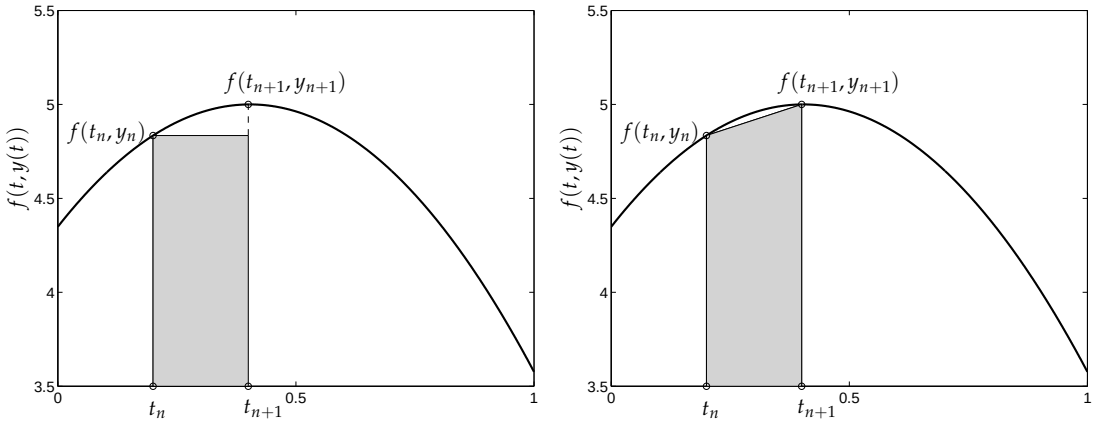
The numerical approximation of the solution at t_n is denoted by w_n, $n = 0, 1, \ldots$, where $w_0 = y_0$.

Forward Euler method
First, the *Forward Euler method* will be presented, which is the most simple, earliest, and best-known time-integration method. In this method, an approximation of the integral in equation (6.3) is obtained by using the left Rectangle rule (Figure 6.2(a)):

$$y_{n+1} \approx y_n + (t_{n+1} - t_n) f(t_n, y_n). \tag{6.4}$$

Note that $f(t_n, y_n)$ is the slope of the tangent in (t_n, y_n). The numerical approximation of the solution at t_{n+1}, denoted by w_{n+1}, is computed by

$$w_{n+1} = w_n + \Delta t f(t_n, w_n), \tag{6.5}$$



(a) Left Rectangle rule (b) Trapezoidal method

Figure 6.2: Two different approximations of the integral $\int_{t_n}^{t_{n+1}} f(t, y(t))dt$.

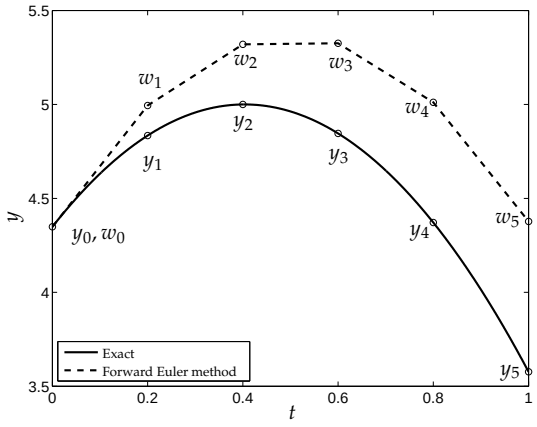


Figure 6.3: Approximation of solution using the Forward Euler method.

in which $f(t_n, w_n)$ is the slope of the tangent in (t_n, w_n). This procedure leads to a sequence $\{t_n, w_n\}$, $n = 0, 1, \ldots$ of approximation points. Geometrically this represents a piecewise-linear approximation of the solution curve, the *Euler polygon*, see Figure 6.3.

Because w_{n+1} can be computed directly from equation (6.5), the Forward Euler method is called an *explicit method*.

Backward Euler method

The *Backward Euler method* is obtained by using the right Rectangle rule to approximate the integrals. At time t_{n+1}, the solution is approximated by

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))dt \approx y_n + \Delta t f(t_{n+1}, y_{n+1}).$$

The numerical approximation at $t = t_{n+1}$ equals

$$w_{n+1} = w_n + \Delta t f(t_{n+1}, w_{n+1}). \tag{6.6}$$

Note that the unknown w_{n+1} also occurs in the right-hand side. For that reason, the Backward Euler method is called an *implicit method*. If f depends linearly on y, then the solution to (6.6) can be easily computed.

For nonlinear initial-value problems, the solution to (6.6) is non-trivial. In that case, a numerical nonlinear solver has to be applied (Chapter 4). This means that the Backward Euler method uses more computation time per time step than the Forward Euler method. However, the Backward Euler method also has some advantages, as will be seen later.

Trapezoidal method
The *Trapezoidal method* is constructed using the Trapezoidal rule of Chapter 5 to approximate the integral (see Figure 6.2(b)). This means that

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))dt \approx y_n + \frac{\Delta t}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right). \tag{6.7}$$

The approximation of the solution at $t = t_{n+1}$ equals

$$w_{n+1} = w_n + \frac{\Delta t}{2}\left(f(t_n, w_n) + f(t_{n+1}, w_{n+1})\right). \tag{6.8}$$

Note that the Trapezoidal method is also an implicit method.

Modified Euler method
To derive an explicit variant of (6.8), the unknown w_{n+1} in the right-hand side of (6.8) is predicted using the Forward Euler method. This results in the following *predictor-corrector* formula:

$$\text{predictor:} \quad \overline{w}_{n+1} = w_n + \Delta t f(t_n, w_n), \tag{6.9a}$$

$$\text{corrector:} \quad w_{n+1} = w_n + \frac{\Delta t}{2}\left(f(t_n, w_n) + f(t_{n+1}, \overline{w}_{n+1})\right). \tag{6.9b}$$

From w_n (calculated in the previous step), the predictor $\overline{w}_{n+1}$ can be computed. Using this value in the corrector, this results in an explicit formula: the *Modified Euler method*.

6.4 Analysis of numerical time-integration methods

The *global truncation error* e_{n+1} at time t_{n+1} is defined as the difference between the solution and the numerical approximation:

$$e_{n+1} = y_{n+1} - w_{n+1}. \tag{6.10}$$

In order to analyze the global truncation error, we need to introduce two important concepts: stability and consistency.

6.4.1 Stability

A physical phenomenon is called *stable* if a small perturbation of the parameters (including the initial condition) results in a small difference in the solution. Hence this concept is connected to the continuous dependence of the solution on the data. Some examples of unstable phenomena are buckling of an axially loaded bar or resonance of a bridge[1]. Throughout this chapter, we assume that the initial-value problem is stable. First, we will consider the conditions for an initial-value problem to be stable.

Analytical stability of an initial-value problem
Let an initial-value problem be given by

$$\begin{cases} y' = f(t, y), & t > t_0, \\ y(t_0) = y_0. \end{cases}$$

[1] http://ta.twi.tudelft.nl/nw/users/vuik/information/tacoma_eng.html

Suppose next that the initial condition contains an error ε_0. Then, the perturbed solution $\tilde{y}$ satisfies

$$\begin{cases} \tilde{y}' = f(t, \tilde{y}), & t > t_0, \\ \tilde{y}(t_0) = y_0 + \varepsilon_0. \end{cases}$$

The difference between the two functions is given by $\varepsilon(t) = \tilde{y}(t) - y(t)$. An initial-value problem is called *stable* if

$$|\varepsilon(t)| < \infty \text{ for all } t > t_0,$$

and *absolutely stable* if it is stable and

$$\lim_{t \to \infty} |\varepsilon(t)| = 0.$$

If $|\varepsilon(t)|$ is unbounded, then the initial-value problem is *unstable*.

It should be noticed that stability of the initial-value problem does not imply that the numerical approximation is stable. The stability of the approximation depends on the numerical method that is used. In the rest of this section, a numerical stability analysis is performed, both for linear and nonlinear initial-value problems.

The test equation
We first analyze how a perturbation in the initial condition propagates for

$$\begin{cases} y' = \lambda y + g(t), & t > t_0, \\ y(t_0) = y_0. \end{cases} \tag{6.11}$$

The perturbed problem is given by

$$\begin{cases} \tilde{y}' = \lambda \tilde{y} + g(t), & t > t_0, \\ \tilde{y}(t_0) = y_0 + \varepsilon_0. \end{cases} \tag{6.12}$$

Subtracting the equations in (6.11) from those in (6.12), we obtain an initial-value problem for the error ε in the approximation:

$$\begin{cases} \varepsilon' = \tilde{y}' - y' = \lambda(\tilde{y} - y) = \lambda \varepsilon, & t > t_0, \\ \varepsilon(t_0) = \varepsilon_0. \end{cases} \tag{6.13}$$

This initial-value problem plays an important role in stability analysis, and is often called the initial-value problem for the so-called *test equation*.

The solution to (6.13) is

$$\varepsilon(t) = \varepsilon_0 e^{\lambda(t - t_0)},$$

which shows that (6.13) is stable if $\lambda \leq 0$ and absolutely stable if $\lambda < 0$.

Numerical stability of initial-value problems based on the test equation
Next, we explain how perturbations propagate if initial-value problem (6.11) is approximated by a numerical method. For example, the Forward Euler method applied to (6.11) yields

$$w_{n+1} = w_n + \Delta t(\lambda w_n + g(t_n)).$$

For the perturbed initial-value problem, we obtain

$$\tilde{w}_{n+1} = \tilde{w}_n + \Delta t(\lambda \tilde{w}_n + g(t_n)).$$

Subtracting these equations yields the following propagation formula for the numerical approximation of the error ($\bar{\varepsilon}_n = w_n - \tilde{w}_n$):

$$\bar{\varepsilon}_{n+1} = (1 + \lambda \Delta t)\bar{\varepsilon}_n.$$

This means that the error propagation for the Forward Euler method is modeled by the test equation.

For each of the numerical methods from Section 6.3, the application to the test equation can be written as

$$\bar{\varepsilon}_{n+1} = Q(\lambda\Delta t)\bar{\varepsilon}_n. \tag{6.14}$$

The value $Q(\lambda\Delta t)$ determines the factor by which the existing perturbation is amplified, and is therefore called the *amplification factor*. By substitution it may be verified that

$$\text{Forward Euler method:} \qquad Q(\lambda\Delta t) = 1 + \lambda\Delta t, \tag{6.15a}$$

$$\text{Backward Euler method:} \qquad Q(\lambda\Delta t) = \frac{1}{1-\lambda\Delta t}, \tag{6.15b}$$

$$\text{Trapezoidal method:} \qquad Q(\lambda\Delta t) = \frac{1+\frac{1}{2}\lambda\Delta t}{1-\frac{1}{2}\lambda\Delta t}, \tag{6.15c}$$

$$\text{Modified Euler method:} \qquad Q(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2. \tag{6.15d}$$

The amplification factor plays an important role in the analysis of the various numerical methods.

Using equation (6.14) recursively it follows that $\bar{\varepsilon}_n = (Q(\lambda\Delta t))^n \bar{\varepsilon}_0$. This expression shows how the initial perturbation is amplified over time. For stability, we require that $|\bar{\varepsilon}_n| < \infty$ for all $n \in \mathbb{N}$. This means that the numerical scheme is stable if and only if

$$|Q(\lambda\Delta t)| \leq 1. \tag{6.16}$$

Because λ is known, we can compute the values of Δt such that inequality (6.16) is still satisfied. For absolute stability, we require $|Q(\lambda\Delta t)| < 1$. Note that the method is unstable if $|Q(\lambda\Delta t)| > 1$. This leads to the following characterization of stability:

$$\text{absolute stability} \iff |Q(\lambda\Delta t)| < 1, \tag{6.17a}$$

$$\text{stability} \iff |Q(\lambda\Delta t)| \leq 1, \tag{6.17b}$$

$$\text{no stability} \iff |Q(\lambda\Delta t)| > 1. \tag{6.17c}$$

As an example, we investigate the Forward Euler method, where $Q(\lambda\Delta t) = 1 + \lambda\Delta t$. Here, we assume that $\lambda < 0$, hence the initial-value problem is absolutely stable. To satisfy (6.16), Δt should be chosen such that

$$-1 \leq 1 + \lambda\Delta t \leq 1,$$

or

$$-2 \leq \lambda\Delta t \leq 0.$$

Since $\Delta t > 0$ and $\lambda \leq 0$, the right-hand side is automatically satisfied. From the left-hand side it follows that

$$\Delta t \leq -\frac{2}{\lambda}, \tag{6.18}$$

with strict inequality for absolute stability.

For the Backward Euler method, stability is found when

$$-1 \leq \frac{1}{1-\lambda\Delta t} \leq 1.$$

It is easily seen that this condition is satisfied for each $\Delta t \geq 0$, since $\lambda \leq 0$. Therefore, the Backward Euler method is unconditionally stable if $\lambda \leq 0$.

A similar analysis shows that, for $\lambda \leq 0$, the Modified Euler method is stable if

$$\Delta t \leq -\frac{2}{\lambda},$$

with strict inequality for absolute stability. The Trapezoidal method is again unconditionally stable.

Stability of a general initial-value problem
For a general (nonlinear) initial-value problem, the stability of the corresponding linearized problem will be analyzed. The initial-value problem

$$\begin{cases} y' = f(t,y), & t > t_0, \\ y(t_0) = y_0 \end{cases} \tag{6.19}$$

is approximated by a linear Taylor expansion (Theorem 1.6.8) about the point $(\hat{t},\hat{y})$:

$$f(t,y) \approx f(\hat{t},\hat{y}) + (y-\hat{y})\frac{\partial f}{\partial y}(\hat{t},\hat{y}) + (t-\hat{t})\frac{\partial f}{\partial t}(\hat{t},\hat{y}).$$

In the neighborhood of $(\hat{t},\hat{y})$, initial-value problem (6.19) can be approximated by

$$y' = f(\hat{t},\hat{y}) + (y-\hat{y})\frac{\partial f}{\partial y}(\hat{t},\hat{y}) + (t-\hat{t})\frac{\partial f}{\partial t}(\hat{t},\hat{y}) \ = \lambda y + g(t), \tag{6.20}$$

with

$$\lambda = \frac{\partial f}{\partial y}(\hat{t},\hat{y}), \tag{6.21a}$$

$$g(t) = f(\hat{t},\hat{y}) - \hat{y}\frac{\partial f}{\partial y}(\hat{t},\hat{y}) + (t-\hat{t})\frac{\partial f}{\partial t}(\hat{t},\hat{y}). \tag{6.21b}$$

The value of λ in equation (6.21a) can be used in the requirement for stability ($|Q(\lambda\Delta t)| \leq 1$). Note that the stability condition depends on $\hat{t}$ and $\hat{y}$.

Example 6.4.1 (Nonlinear problem)

A simple model of the water-discharge problem is given by

$$\begin{cases} y' = -10y^2 + 20, & t > 0, \\ y(0) = 0. \end{cases}$$

This model can be related to initial-value problem (6.1), using $a = 10$ and $p_0 = 20$, and the solution is given by

$$y(t) = \sqrt{2}\tanh(\sqrt{300}t).$$

This means that $y(t) \in (0, \sqrt{2})$ for all $t > 0$.

In this example, $f(t,y) = -10y^2 + 20$. The linearized approximation of the differential equation uses

$$\lambda = \frac{\partial f}{\partial y}(\hat{t},\hat{y}) = -20\hat{y}.$$

Because $y(t) > 0$ for all $t > 0$, this initial-value problem is absolutely stable: $\lambda < 0$.

Furthermore, it is important to investigate numerical stability. Note that if the Forward Euler method is stable, then the numerical approximation of the solution satisfies $w_n \in (0, \sqrt{2})$ for all $n \in \mathbb{N}$. The maximum $\sqrt{2}$ can be used as an upper bound to derive a criterion for the largest admissible time step that ensures stability:

$$\Delta t \leq -\frac{2}{\lambda} = \frac{1}{10\hat{y}} \leq \frac{1}{10\cdot\sqrt{2}}.$$

6.4.2 Local truncation error

After having considered the stability of numerical methods for initial-value problems, the accuracy of these methods will be analyzed. In this subsection, the new error in each time step (the local truncation error) will be considered. The cumulative effect of these errors in the final result (the global truncation error), will be considered in Section 6.4.3.

Definition 6.4.1 (Local truncation error) *The* local truncation error *at time step $n+1$, τ_{n+1}, is defined as*

$$\tau_{n+1} = \frac{y_{n+1} - z_{n+1}}{\Delta t},$$

in which y_{n+1} is the solution at time step $n+1$ and z_{n+1} is the numerical approximation obtained by applying the numerical method with starting point y_n, the solution at time step n.

The estimation of the local truncation error will be explained for the Forward Euler method and the Modified Euler method. We assume that all derivatives that we use are continuous.

Forward Euler method
For the Forward Euler method, we have

$$\tau_{n+1} = \frac{y_{n+1} - (y_n + \Delta t f(t_n, y_n))}{\Delta t}. \tag{6.22}$$

Taylor expansion of y_{n+1} about t_n yields

$$\begin{aligned} y_{n+1} = y(t_{n+1}) &= y(t_n) + (t_{n+1} - t_n)y'(t_n) + \frac{(t_{n+1} - t_n)^2}{2} y''(\xi) \\ &= y_n + \Delta t y'_n + \frac{\Delta t^2}{2} y''(\xi), \end{aligned} \tag{6.23}$$

for a $\xi \in (t_n, t_{n+1})$.

Note that y satisfies the differential equation, such that

$$y' = f(t, y). \tag{6.24}$$

Using equations (6.23) and (6.24) in (6.22) yields

$$\tau_{n+1} = \frac{y_n + \Delta t y'_n + \frac{\Delta t^2}{2} y''(\xi) - (y_n + \Delta t y'_n)}{\Delta t} = \frac{\Delta t}{2} y''(\xi),$$

where $\xi \in (t_n, t_{n+1})$. This means that the Forward Euler method is of order $\mathcal{O}(\Delta t)$.

Modified Euler method
For the Modified Euler method, the value of z_{n+1} is obtained by replacing the numerical approximation w_n in equation (6.9) by the exact value y_n:

$$\overline{z}_{n+1} = y_n + \Delta t f(t_n, y_n), \tag{6.25a}$$

$$z_{n+1} = y_n + \frac{\Delta t}{2}(f(t_n, y_n) + f(t_{n+1}, \overline{z}_{n+1})). \tag{6.25b}$$

Using a two-dimensional Taylor expansion about (t_n, y_n) (Theorem 1.6.8), the final term of (6.25b) can be written as

$$\begin{aligned} f(t_{n+1}, \overline{z}_{n+1}) &= f(t_n + \Delta t, y_n + \Delta t f(t_n, y_n)) \\ &= f(t_n, y_n) + (t_n + \Delta t - t_n)\left(\frac{\partial f}{\partial t}\right)_n + (y_n + \Delta t f(t_n, y_n) - y_n)\left(\frac{\partial f}{\partial y}\right)_n \\ &+ \frac{1}{2}(t_n + \Delta t - t_n)^2 \left(\frac{\partial^2 f}{\partial t^2}\right)_n + \frac{1}{2}(y_n + \Delta t f(t_n, y_n) - y_n)^2 \left(\frac{\partial^2 f}{\partial y^2}\right)_n \\ &+ (t_n + \Delta t - t_n)(y_n + \Delta t f(t_n, y_n) - y_n)\left(\frac{\partial^2 f}{\partial t \partial y}\right)_n + \dots. \end{aligned}$$

Writing $f_n = f(t_n, y_n)$, this means that

$$f(t_{n+1}, \overline{z}_{n+1}) = f_n + \Delta t \left(\frac{\partial f}{\partial t}\right)_n + \Delta t f_n \left(\frac{\partial f}{\partial y}\right)_n + \frac{1}{2}\Delta t^2 \left(\frac{\partial^2 f}{\partial t^2}\right)_n + \frac{1}{2}(\Delta t f_n)^2 \left(\frac{\partial^2 f}{\partial y^2}\right)_n + \Delta t^2 f_n \left(\frac{\partial^2 f}{\partial t \partial y}\right)_n + \ldots \tag{6.26}$$

Note that the last three specified terms of (6.26) are $\mathcal{O}(\Delta t^2)$ (and that the dots represent terms of order $\mathcal{O}(\Delta t^3)$ and higher). Since higher-order terms are not required, we arrive at

$$f(t_{n+1}, \overline{z}_{n+1}) = f_n + \Delta t \left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}\right)_n + \mathcal{O}(\Delta t^2).$$

Substitution of this expression into (6.25b) yields

$$z_{n+1} = y_n + \Delta t f_n + \frac{\Delta t^2}{2}\left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}\right)_n + \mathcal{O}(\Delta t^3). \tag{6.27}$$

Differentiating the differential equation with respect to t and using the chain rule, yields

$$y''(t) = \frac{dy'}{dt} = \frac{df(t, y(t))}{dt} = \frac{\partial f}{\partial t}\frac{dt}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = \left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}\right). \tag{6.28}$$

Using relations (6.24) and (6.28), equation (6.27) transforms into

$$z_{n+1} = y_n + \Delta t y'_n + \frac{\Delta t^2}{2} y''_n + \mathcal{O}(\Delta t^3). \tag{6.29}$$

Note that the first three terms of this expression are exactly the first three terms of the Taylor expansion

$$y_{n+1} = y_n + \Delta t y'_n + \frac{\Delta t^2}{2} y''_n + \mathcal{O}(\Delta t^3). \tag{6.30}$$

This means that

$$\tau_{n+1} = \frac{y_{n+1} - z_{n+1}}{\Delta t} = \mathcal{O}(\Delta t^2).$$

Local truncation error for initial-value problems based on the test equation
For implicit time-integration methods, the procedure to estimate the local truncation error has to be adapted. Therefore, we only consider the error estimation for the test equation $y' = \lambda y$. The treatment for implicit time-integration methods to the general equation $y' = f(t, y(t))$ is outside the scope of this book.

Remember that z_{n+1} is obtained by applying the numerical scheme to the solution y_n. Using the amplification factor, this can be computed easily: $z_{n+1} = Q(\lambda\Delta t)y_n$ (Section 6.4.1). Therefore, the local truncation error equals

$$\tau_{n+1} = \frac{y_{n+1} - Q(\lambda\Delta t)y_n}{\Delta t}. \tag{6.31}$$

Furthermore, because the solution to the test equation equals $y(t) = y_0 e^{\lambda t}$, it is clear that

$$y_{n+1} = y_0 e^{\lambda(t_n + \Delta t)} = y_n e^{\lambda \Delta t}. \tag{6.32}$$

Substitution of (6.32) into (6.31) yields

$$\tau_{n+1} = \frac{e^{\lambda\Delta t} - Q(\lambda\Delta t)}{\Delta t} y_n. \tag{6.33}$$

This means that the size of the local truncation error is determined by the difference between the amplification factor and the exponential function. To determine this difference, the Taylor expansion of $e^{\lambda\Delta t}$ can be used (see Theorem 1.6.10):

$$e^{\lambda\Delta t} = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{3!}(\lambda\Delta t)^3 + \ldots = \sum_{k=0}^{\infty} \frac{1}{k!}(\lambda\Delta t)^k. \tag{6.34}$$

Comparing (6.34) with $Q(\lambda\Delta t) = 1 + \lambda\Delta t$ we conclude that the Forward Euler method has a local truncation error of order $\mathcal{O}(\Delta t)$. For the Modified Euler method,

$$Q(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2,$$

such that the local truncation error is $\mathcal{O}(\Delta t^2)$ and this method is more accurate in the limit $\Delta t \to 0$.

For the Backward Euler method, a Taylor expansion of the amplification factor yields (see Theorem 1.6.9)

$$Q(\lambda\Delta t) = \frac{1}{1 - \lambda\Delta t} = 1 + \lambda\Delta t + (\lambda\Delta t)^2 + \ldots = \sum_{k=0}^{\infty} (\lambda\Delta t)^k,$$

hence the local truncation error of the Backward Euler method is $\mathcal{O}(\Delta t)$. The Trapezoidal method has a local truncation error of order $\mathcal{O}(\Delta t^2)$ (see exercise 3).

6.4.3 Global truncation error

In this section, we will relate stability, the local truncation error, and the global truncation error. Therefore, we need the definitions of consistency and convergence.

Definition 6.4.2 (Consistency) *A numerical time-integration method is called* consistent *if*

$$\lim_{\Delta t \to 0} \tau_{n+1}(\Delta t) = 0,$$

where $(n+1)\Delta t = T$, and T is a fixed time. We call a method consistent of order p *if $\tau_{n+1} = \mathcal{O}(\Delta t^p)$.*

Definition 6.4.3 (Convergence) *A scheme is called* convergent *if the global truncation error e_{n+1} satisfies*

$$\lim_{\Delta t \to 0} e_{n+1} = 0,$$

where $(n+1)\Delta t = T$, and T is a fixed time. A method is convergent of order p *if $e_{n+1} = \mathcal{O}(\Delta t^p)$.*

The theorem below is one of the most important theorems to prove convergence.

Theorem 6.4.1 (Lax's equivalence theorem, [6]) *If a numerical method is stable and consistent, then the numerical approximation converges to the solution for $\Delta t \to 0$. Moreover, the global truncation error and the local truncation error are of the same order.*

Proof
This theorem will only be proved for the test equation $y' = \lambda y$. In that case, we know that $w_{n+1} = Q(\lambda\Delta t)w_n$, such that the global truncation error, e_{n+1}, satisfies

$$e_{n+1} = y_{n+1} - w_{n+1} = y_{n+1} - Q(\lambda\Delta t)w_n.$$

By definition, $w_n = y_n - e_n$, such that

$$e_{n+1} = y_{n+1} - Q(\lambda\Delta t)y_n + Q(\lambda\Delta t)e_n.$$

Next, using equation (6.31), we arrive at

$$e_{n+1} = \Delta t \tau_{n+1} + Q(\lambda \Delta t) e_n.$$

Repeating this argument yields

$$e_{n+1} = \sum_{\ell=0}^{n} \left(Q(\lambda \Delta t)\right)^{\ell} \Delta t \tau_{n+1-\ell}. \tag{6.35}$$

Using stability ($|Q(\lambda \Delta t)| \leq 1$) and $(n+1)\Delta t = T$, the global truncation error can be bounded by

$$|e_{n+1}| \leq \sum_{\ell=0}^{n} |Q(\lambda \Delta t)|^{\ell} \Delta t |\tau_{n+1-\ell}| \leq \sum_{\ell=0}^{n} \Delta t |\tau_{n+1-\ell}| \leq T \cdot \max_{1 \leq \ell \leq n+1} |\tau_{\ell}|.$$

This implies that the order of convergence of the global truncation error equals that of the local truncation error. Furthermore, the global truncation error tends to zero as $\Delta t \to 0$, since the numerical method is consistent. ⊠

6.5 Higher-order methods

In this section, the importance of higher-order methods will be investigated. Furthermore, an example of a higher-order method will be treated.

First, we illustrate the importance of the order of accuracy. One advantage of higher-order methods is that a higher accuracy is achieved at least for sufficiently small time steps Δt. Since larger time steps are admissible to obtain a fixed accuracy, fewer time steps are required. However, the total amount of work also depends on the number of function evaluations per time step. For example, the Forward Euler method needs one function evaluation per time step, whereas the Modified Euler method needs two evaluations. In the next example, the total number of function evaluations is computed for a fixed accuracy.

Example 6.5.1 (Order of accuracy)

To illustrate the importance of a higher-order method, as a thought example we assume that the global truncation error for the Forward Euler method equals $|e_n| \leq \Delta t$, and $|e_n| \leq \Delta t^2$ for the Modified Euler method. If we integrate an initial-value problem from $t_0 = 0$ to $T = 1$, and require that $|e_n| \leq 10^{-1}$, then it follows that $\Delta t = 10^{-1}$ for the Forward Euler method, and $\Delta t = 1/3$ for the Modified Euler method. This means that 10 function evaluations are required using the Forward Euler method, whereas only $3 \cdot 2 = 6$ evaluations are required for the Modified Euler method.

This effect is even more visible when a higher accuracy is required, as can be seen in Table 6.1.

Table 6.1: Number of function evaluations (denoted by # fe) for the Forward Euler method and the Modified Euler method using various accuracies. Integration is done from $t_0 = 0$ to $T = 1$, and Δt is chosen such that the desired global truncation errors are found.

	Forward Euler method		Modified Euler method	
$\lvert y_n - w_n \rvert$	Δt	# fe	Δt	# fe
10^{-1}	10^{-1}	10	0.3	6
10^{-2}	10^{-2}	100	10^{-1}	20
10^{-4}	10^{-4}	10000	10^{-2}	200

Example 6.5.1 shows that a higher-order method is to be preferred, if a high accuracy is required. This observation is valid as long as the solution is sufficiently smooth.

The fourth-order method of Runge-Kutta (RK4 method)
Next, the fourth-order method of Runge-Kutta will be introduced. The use of this higher-order method not only results in large savings, the RK4 method also has attractive stability properties despite its explicit nature.

The RK4 method approximates the solution at t_{n+1} by

$$w_{n+1} = w_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{6.36}$$

in which the predictors k_1 to k_4 are given by

$$k_1 = \Delta t f(t_n, w_n), \tag{6.37a}$$
$$k_2 = \Delta t f(t_n + \frac{1}{2}\Delta t, w_n + \frac{1}{2}k_1), \tag{6.37b}$$
$$k_3 = \Delta t f(t_n + \frac{1}{2}\Delta t, w_n + \frac{1}{2}k_2), \tag{6.37c}$$
$$k_4 = \Delta t f(t_n + \Delta t, w_n + k_3). \tag{6.37d}$$

Note that the corrector formula is based on *Simpson's rule* for numerical integration:

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))dt \\ &\approx y(t_n) + \frac{\Delta t}{6}\left(f(t_n, y(t_n)) + 4f\left(t_n + \frac{1}{2}\Delta t, y\left(t_n + \frac{1}{2}\Delta t\right)\right) + f(t_n + \Delta t, y(t_n + \Delta t))\right). \end{aligned}$$

Here, $y(t_n)$ is approximated by w_n, and the values of $y(t_n + \Delta t/2)$ and $y(t_n + \Delta t)$ have to be predicted. From equations (6.37) it follows that both $w_n + k_1/2$ and $w_n + k_2/2$ approximate $y(t_n + \Delta t/2)$, and that $y(t_n + \Delta t)$ is predicted by $w_n + k_3$.

The amplification factor of the RK4 method can be computed using the test equation. Application of formulae (6.36) and (6.37) to $y' = \lambda y$ results in

$$w_{n+1} = \left(1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4\right) w_n.$$

This means that the amplification factor of the RK4 method is given by

$$Q(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4. \tag{6.38}$$

Next, the local truncation error is computed for the test equation. The quantity z_{n+1} is defined by replacing w_n in (6.36) and (6.37) by y_n. Substitution of $f(t, y) = \lambda y$ leads to $z_{n+1} = Q(\lambda\Delta t)y_n$. Using that $y_{n+1} = e^{\lambda\Delta t}y_n$ (equation (6.32)), we obtain

$$y_{n+1} - z_{n+1} = \left(e^{\lambda\Delta t} - Q(\lambda\Delta t)\right) y_n. \tag{6.39}$$

The right-hand side of (6.38) consists exactly of the first five terms of the Taylor series of $e^{\lambda\Delta t}$. Hence in the right-hand side of (6.39), only the 5th and the higher powers of Δt remain. Division by Δt shows that the local truncation error is $\mathcal{O}(\Delta t^4)$. Also for a general initial-value problem, it can be shown that the RK4 method is of fourth order.

Finally, it is important to investigate the stability properties. From (6.38) it follows that for each $\Delta t > 0$, $Q(\lambda\Delta t)$ is larger than one if λ is positive, hence exponential error growth will be inevitable.

For $\lambda < 0$, the RK4 method is stable when

$$-1 \leq 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 \leq 1, \text{ where } x = \lambda\Delta t. \tag{6.40}$$

The left inequality of (6.40) yields

$$2 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 \geq 0,$$

which is satisfied for all x. This is seen by considering the extremes of the polynomial

$$P(x) = 2 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4.$$

The extremes are attained at the zeros of P', i.e. for those values $\tilde{x}$ for which

$$P'(\tilde{x}) = 1 + \tilde{x} + \frac{1}{2}\tilde{x}^2 + \frac{1}{6}\tilde{x}^3 = 0.$$

An extreme value of P is found by substituting $\tilde{x}$ into P and using the relation $P'(\tilde{x}) = 0$:

$$P(\tilde{x}) = 2 + \tilde{x} + \frac{1}{2}\tilde{x}^2 + \frac{1}{6}\tilde{x}^3 + \frac{1}{24}\tilde{x}^4 = 1 + P'(\tilde{x}) + \frac{1}{24}\tilde{x}^4 = 1 + \frac{1}{24}\tilde{x}^4.$$

Note that $P(\tilde{x}) > 0$ for all $\tilde{x}$. This means that all extremes of P are positive, hence also the minimum of P. Consequently, P is positive for all x and that proves the proposition. The left inequality can therefore not be used to derive a stability bound for Δt.

Next, the right inequality in (6.40) should be considered. This inequality can be written as

$$x(1 + \frac{1}{2}x + \frac{1}{6}x^2 + \frac{1}{24}x^3) \leq 0. \tag{6.41}$$

Because $x = \lambda\Delta t < 0$, equation (6.41) is equivalent to

$$1 + \frac{1}{2}x + \frac{1}{6}x^2 + \frac{1}{24}x^3 \geq 0.$$

Analysis of this polynomial shows that it possesses only one zero, for $x \approx -2.8$. The function is negative for $x < -2.8$, and positive for $x > -2.8$. From this it follows that $x = \lambda\Delta t \geq -2.8$, or

$$\Delta t \leq -\frac{2.8}{\lambda}, \tag{6.42}$$

in order to satisfy $-1 \leq Q(\lambda\Delta t) \leq 1$.

Example 6.5.2 (Order of accuracy)

To extend example 6.5.1, we assume that the global truncation error for the RK4 method equals $|e_n| \leq \Delta t^4$. Note that the RK4 method requires four function evaluations per time step. In Table 6.2, the results can be seen. Note that for a certain accuracy, the RK4 method can use much larger time steps than the Forward Euler and the Modified Euler method. For higher accuracies, the number of function evaluations is therefore much smaller than for lower-order methods.

Table 6.3 presents an overview of the stability conditions and local truncation errors for all discussed methods.

6.6 Global truncation error and Richardson error estimates

In this section we analyze the behavior of the global truncation error as a function of the time step. For the given numerical time-integration methods, the global truncation error is of order $\mathcal{O}(\Delta t^p)$, with $p = 1$ for the Forward Euler and Backward Euler method, $p = 2$ for the Trapezoidal and the Modified Euler method, and $p = 4$ for the RK4 method. We assume that for sufficiently small values of Δt, the global truncation error can be approximated by

$$e(t, \Delta t) = c_p(t)\Delta t^p. \tag{6.43}$$

In general, the solution to the initial-value problem is unknown. In this section, we apply Richardson's extrapolation (Section 3.7) to numerically estimate the global truncation error.

Table 6.2: Number of function evaluations (denoted by # fe) for the Forward Euler, Modified Euler and RK4 method using various accuracies. Integration is done from $t_0 = 0$ to $T = 1$, and Δt is chosen such that the desired global truncation errors are found.

	Forward Euler		Modified Euler		RK4	
$\lvert y_n - w_n \rvert$	Δt	# fe	Δt	# fe	Δt	# fe
10^{-1}	10^{-1}	10	0.3	6	0.5	8
10^{-2}	10^{-2}	100	10^{-1}	20	0.3	12
10^{-4}	10^{-4}	10000	10^{-2}	200	10^{-1}	40

Table 6.3: Amplification factors, stability conditions ($\lambda \in \mathbb{R}$, $\lambda < 0$), and local truncation errors.

method	amplification factor	stab. condition	τ_{n+1}
Forward Euler	$1 + \lambda\Delta t$	$\Delta t \leq -\frac{2}{\lambda}$	$\mathcal{O}(\Delta t)$
Backward Euler	$\frac{1}{1-\lambda\Delta t}$	all Δt	$\mathcal{O}(\Delta t)$
Trapezoidal	$\frac{1+\frac{1}{2}\lambda\Delta t}{1-\frac{1}{2}\lambda\Delta t}$	all Δt	$\mathcal{O}(\Delta t^2)$
Modified Euler	$1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2$	$\Delta t \leq -\frac{2}{\lambda}$	$\mathcal{O}(\Delta t^2)$
RK4	$1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 + \frac{1}{6}(\lambda\Delta t)^3 + \frac{1}{24}(\lambda\Delta t)^4$	$\Delta t \leq -\frac{2.8}{\lambda}$	$\mathcal{O}(\Delta t^4)$

6.6.1 Error estimate, p is known

Similar to Section 3.7.2, two numerical approximations of the solution at time t are used to compute the global truncation error estimate. Let $w_{N/2}^{2\Delta t}$ and $w_N^{\Delta t}$ denote the approximations for $y(t) = y(N\Delta t)$ using $N/2$ time steps of length $2\Delta t$ and using N time steps of length Δt, respectively.

For Δt small enough (see Section 6.6.2), higher-order terms are disregarded, and hence we obtain

$$e(t, 2\Delta t) = y(t) - w_{N/2}^{2\Delta t} = c_p(t)(2\Delta t)^p, \tag{6.44a}$$

$$e(t, \Delta t) = y(t) - w_N^{\Delta t} \quad = c_p(t)\Delta t^p. \tag{6.44b}$$

By subtracting these two equations we obtain

$$w_N^{\Delta t} - w_{N/2}^{2\Delta t} = c_p(t)\Delta t^p(2^p - 1), \tag{6.45}$$

from which the value of $c_p(t)$ follows:

$$c_p(t) = \frac{w_N^{\Delta t} - w_{N/2}^{2\Delta t}}{\Delta t^p(2^p - 1)}. \tag{6.46}$$

By substituting this estimate of $c_p(t)$ into (6.44b), we obtain a global truncation-error estimate for the approximation with time step Δt:

$$y(t) - w_N^{\Delta t} = \frac{w_N^{\Delta t} - w_{N/2}^{2\Delta t}}{2^p - 1}. \tag{6.47}$$

The order of the method manifests itself in the factor $\frac{1}{2^p-1}$ of the global truncation-error estimate.

6.6.2 Error estimate, p is unknown

In the practical situation in which both the solution and the order of the global truncation error are not known, there is no immediate quality check of the global truncation-error estimate. For

a certain value Δt, it is unknown whether relation (6.43) is sufficiently well satisfied to obtain an accurate truncation-error estimate. Applying the same approach as in Section 3.7.2, and defining $w_{N/4}^{4\Delta t}$ as the approximation for $y(t)$ using $N/4$ time steps of length $4\Delta t$, equation (3.14) gives that

$$\frac{w_{N/2}^{2\Delta t} - w_{N/4}^{4\Delta t}}{w_N^{\Delta t} - w_{N/2}^{2\Delta t}} = 2^p. \tag{6.48}$$

If for a certain Δt the value of equation (6.48) is in the neighborhood of the expected 2^p, then Δt is small enough to obtain an accurate global truncation-error estimate.

Example 6.6.1 (Water discharge)

In this example, Richardson's extrapolation is applied to a slightly modified water-discharge problem (see initial-value problem (6.1)), given by

$$\begin{cases} y' = 50 - 2y^{2.1}, \quad t > 0, \\ y(0) = 0. \end{cases} \tag{6.49}$$

An explicit solution to this problem is not known. Table 6.4 gives the numerical approximation of the solution at $t = 0.2$ using the Forward Euler method for various time steps. The estimate for $y(t) - w_N^{\Delta t}$ (equation (6.47)) is also given, together with the estimated value of 2^p, using equation (6.48). Note that indeed, the global truncation-error estimate is doubled when $2\Delta t$ is taken. The results in the fourth column show that from $\Delta t = 0.00125$ onwards the global truncation error may be considered linear in Δt.

Table 6.4: Numerical approximation and global truncation-error estimates for the solution to problem (6.49) at $t = 0.2$, using the Forward Euler method and equations (6.47) and (6.48).

Δt	$w_N^{\Delta t}$	$y(t) - w_N^{\Delta t}$	2^p
0.01000000000	4.559913710927	-	-
0.00500000000	4.543116291062	-0.01679742	-
0.00250000000	4.534384275072	-0.00873202	1.9236589
0.00125000000	4.529943322643	-0.00444095	1.9662485
0.00062500000	4.527705063356	-0.00223826	1.9841099
0.00031250000	4.526581601706	-0.00112346	1.9922881
0.00015625000	4.526018801777	-0.00056280	1.9962008
0.00007812500	4.525737136255	-0.00028167	1.9981144
0.00003906250	4.525596237317	-0.00014090	1.9990607
0.00001953125	4.525525771331	-0.00007047	1.9995312

In general, the global truncation error at a certain time t should satisfy $|e(t, \Delta t)| \leq \varepsilon$, for a certain bound ε. Richardson's extrapolation can be used to determine whether a certain value of Δt is accurate enough.

In practice, Δt is often based on knowledge of the general behavior of the solution or an idea about the number of points necessary to recognize the solution visually. For example, to visualize the sine function on the interval $(0, \pi)$, about 10 to 20 points are required. To estimate the global truncation error for this value of Δt, the numerical integration is performed three times (for Δt, $2\Delta t$ and $4\Delta t$), and the value of (6.48) is computed. Note that if the estimation of 2^p is not accurate enough, then Δt should apparently be taken smaller. In practice, the time step is halved, such that two earlier approximations can be reused. As soon as 2^p is approximated accurately enough, an estimate for the global truncation error according to (6.47) is computed, using the last two approximations. If the estimated global truncation error exceeds the threshold, we again take a

smaller time step, until the error estimate is small enough. The approximation of the solution will become more and more accurate because (6.48) is satisfied better and better as Δt decreases.

This characterizes the use of adaptive time-stepping in the numerical integration of initial-value problems. More details about adaptive time-stepping can be found in [8].

6.7 Numerical methods for systems of differential equations

This section focuses on numerical time-integration methods for systems of differential equations. Here, we assume that $y_1, \ldots, y_m$ are m unknown functions and that each function is characterized by an initial-value problem of the form

$$\begin{cases} y_j' = f_j(t, y_1, \ldots, y_m), & t > t_0, \\ y_j(t_0) = y_{j,0}, \end{cases}$$

$j = 1, \ldots, m$. Writing this in vector notation by using $\mathbf{y} = (y_1, \ldots, y_m)^\top$ and $\mathbf{f} = (f_1, \ldots, f_m)^\top$, yields the vector-valued initial-value problem

$$\begin{cases} \mathbf{y}' = \mathbf{f}(t, \mathbf{y}), & t > t_0, \\ \mathbf{y}(t_0) = \mathbf{y}_0. \end{cases}$$

For the Forward Euler method, it will be shown that the application to systems of equations is similar to the scalar case. Similarly, the vector-valued generalization of the scalar case will be presented for the Backward Euler, Trapezoidal, Modified Euler, and RK4 method.

For systems of differential equations, the Forward Euler method (equation (6.5)) should be applied to each differential equation separately. For the jth component, the solution at time t_{n+1} is approximated by

$$w_{j,n+1} = w_{j,n} + \Delta t f_j(t_n, w_{1,n}, \ldots, w_{m,n}), \tag{6.50}$$

$j = 1, \ldots, m$. The corresponding vector notation reads

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \Delta t \mathbf{f}(t_n, \mathbf{w}_n). \tag{6.51a}$$

Indeed, the Forward Euler method for systems of differential equations is a straightforward vector-valued generalization of the corresponding scalar method. Using the definitions of the Backward Euler (equation (6.6)), Trapezoidal (equation (6.8)), Modified Euler (equation (6.9)) and RK4 method (equation (6.36)) we obtain their vector-valued generalizations:

Backward Euler method:
$$\mathbf{w}_{n+1} = \mathbf{w}_n + \Delta t \mathbf{f}(t_{n+1}, \mathbf{w}_{n+1}), \tag{6.51b}$$

Trapezoidal method:
$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\Delta t}{2} \left(\mathbf{f}(t_n, \mathbf{w}_n) + \mathbf{f}(t_{n+1}, \mathbf{w}_{n+1}) \right), \tag{6.51c}$$

Modified Euler method:
$$\bar{\mathbf{w}}_{n+1} = \mathbf{w}_n + \Delta t \mathbf{f}(t_n, \mathbf{w}_n),$$
$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\Delta t}{2} \left(\mathbf{f}(t_n, \mathbf{w}_n) + \mathbf{f}(t_{n+1}, \bar{\mathbf{w}}_{n+1}) \right), \tag{6.51d}$$

RK4 method:
$$\mathbf{k}_1 = \Delta t \mathbf{f}(t_n, \mathbf{w}_n), \tag{6.51e}$$
$$\mathbf{k}_2 = \Delta t \mathbf{f}(t_n + \frac{1}{2}\Delta t, \mathbf{w}_n + \frac{1}{2}\mathbf{k}_1), \tag{6.51f}$$
$$\mathbf{k}_3 = \Delta t \mathbf{f}(t_n + \frac{1}{2}\Delta t, \mathbf{w}_n + \frac{1}{2}\mathbf{k}_2), \tag{6.51g}$$
$$\mathbf{k}_4 = \Delta t \mathbf{f}(t_n + \Delta t, \mathbf{w}_n + \mathbf{k}_3), \tag{6.51h}$$
$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \tag{6.51i}$$

For the implicit methods, an approximation of the solution to a nonlinear system should be determined at each time step. This can be achieved using a method from Section 4.6.

Higher-order initial-value problems

A higher-order initial-value problem relates the mth derivative of a function to its lower-order derivatives. Using the notation

$$x^{(j)} = \frac{d^j x}{dt^j},\ j = 0, \ldots, m,$$

we consider the following initial-value problem:

$$\begin{cases} x^{(m)} = f(t, x, x^{(1)}, \ldots, x^{(m-1)}), \quad t > t_0, \\ x(t_0) = x_0, \\ x^{(1)}(t_0) = x_0^{(1)}, \\ \vdots \\ x^{(m-1)}(t_0) = x_0^{(m-1)}, \end{cases} \tag{6.52}$$

which consists of a single mth order differential equation and a set of initial conditions for x and all of its derivatives up to the $m-1$th order.

One way to solve such a higher-order initial-value problem is to transform the single differential equation into a system of first-order differential equations, by defining

$$\begin{aligned} y_1 &= x, \\ y_2 &= x^{(1)}, \\ &\vdots \\ y_m &= x^{(m-1)}. \end{aligned}$$

In this way, problem (6.52) transforms into the following first-order system:

$$\begin{cases} y_1' = y_2, \\ \vdots \\ y_{m-1}' = y_m, \\ y_m' = f(t, y_1, \ldots, y_m), \quad t > t_0, \\ y_1(t_0) = x_0, \\ \vdots \\ y_m(t_0) = x_0^{(m-1)}. \end{cases}$$

Example 6.7.1 (Mathematical pendulum)

The angular displacement of a pendulum is described by the initial-value problem

$$\begin{cases} \psi'' + \sin\psi = 0, \quad t > 0, \\ \psi(0) = \psi_0, \\ \psi'(0) = 0. \end{cases}$$

By letting $y_1 = \psi$ and $y_2 = \psi'$, the corresponding first-order system reads

$$\begin{cases} y_1' = y_2, \\ y_2' = -\sin y_1, \quad t > 0, \\ y_1(0) = \psi_0, \\ y_2(0) = 0. \end{cases}$$

The Forward Euler method approximates the solution at t_{n+1} by

$$\begin{aligned} w_{1,n+1} &= w_{1,n} + \Delta t w_{2,n}, \\ w_{2,n+1} &= w_{2,n} - \Delta t \sin w_{1,n}, \end{aligned}$$

or, in vector notation:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \Delta t \begin{pmatrix} w_{2,n} \\ -\sin w_{1,n} \end{pmatrix}.$$

Example 6.7.2 (Second-order initial-value problem)

Consider the initial-value problem

$$\begin{cases} ax'' + bx' + cx = g(t), \quad t > t_0, \\ x(t_0) = x_0, \\ x'(t_0) = x_0^{(1)}. \end{cases}$$

The second-order differential equation can be transformed to a system of first-order differential equations by defining $y_1 = x$, $y_2 = x'$:

$$\begin{cases} y_1' = y_2, \\ y_2' = -\frac{b}{a} y_2 - \frac{c}{a} y_1 + \frac{1}{a} g(t), \quad t > t_0, \\ y_1(t_0) = x_0, \\ y_2(t_0) = x_0^{(1)}. \end{cases}$$

By defining

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -\frac{c}{a} & -\frac{b}{a} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \text{and} \quad \mathbf{g}(t) = \begin{pmatrix} 0 \\ \frac{1}{a} g(t) \end{pmatrix},$$

we arrive at the matrix-vector form $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{g}(t)$.

6.8 Analytical and numerical stability for systems

It becomes clear from Section 6.7 that the numerical methods for systems of differential equations have a similar structure as their scalar counterparts. In this section, we will analyze their stability behavior and relate it to that of the scalar case.

As in the scalar case, we first consider a simplified initial-value problem for a linear differential equation with constant coefficient matrix, which constitutes a direct generalization of the scalar case (6.11):

$$\begin{cases} \mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{g}(t), \quad t > t_0, \\ \mathbf{y}(t_0) = \mathbf{y}_0. \end{cases} \tag{6.53}$$

Here $\mathbf{y}$ and $\mathbf{g}(t)$ are vectors of length m and $\mathbf{A}$ is the $m \times m$ coefficient matrix. Assuming perturbations only in the initial values, the perturbed problem is given by

$$\begin{cases} \tilde{\mathbf{y}}' = \mathbf{A}\tilde{\mathbf{y}} + \mathbf{g}(t), \quad t > t_0, \\ \tilde{\mathbf{y}}(t_0) = \mathbf{y}_0 + \boldsymbol{\varepsilon}_0. \end{cases}$$

For the difference $\boldsymbol{\varepsilon} = \tilde{\mathbf{y}} - \mathbf{y}$, the test system holds:

$$\begin{cases} \boldsymbol{\varepsilon}' = \mathbf{A}\boldsymbol{\varepsilon}, \quad t > t_0, \\ \boldsymbol{\varepsilon}(t_0) = \boldsymbol{\varepsilon}_0. \end{cases} \tag{6.54}$$

If we assume that the matrix $\mathbf{A}$ can be diagonalized, then the stability properties of system (6.54) can be related to the stability properties of a scalar initial-value problem. It will be shown that the eigenvalues of the matrix $\mathbf{A}$ (see definition 1.6.5) play the same role as λ in the scalar test equation $y' = \lambda y$. The major difference to the scalar case is the possible occurrence of complex eigenvalues that has to be taken into account.

6.8.1 Analytical stability of the test system

The analytical stability of the test system is explained in depth in Section 6.8.2. The most important conclusion is that the system of differential equations is analytically stable depending on the eigenvalues λ_j of $\mathbf{A}$ (see definition 1.6.5). The following characterization holds [4]:

$$\begin{array}{rcll} \text{absolute stability} & \Longleftrightarrow & \text{for all } \lambda_j: & Re(\lambda_j) < 0, \\ \text{stability} & \Longleftrightarrow & \text{for all } \lambda_j: & Re(\lambda_j) \leq 0, \\ \text{no stability} & \Longleftrightarrow & \text{there exists a } \lambda_j: & Re(\lambda_j) > 0. \end{array}$$

The following example is used to show how the procedure works.

Example 6.8.1 (Second-order initial-value problem)

In this example we use the system of differential equations as given in Example 6.7.2. The corresponding test system is equal to $\mathbf{y}' = \mathbf{A}\mathbf{y}$. The eigenvalues of $\mathbf{A}$ are given by the solutions to the equation

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \begin{vmatrix} 0-\lambda & 1 \\ -\frac{c}{a} & -\frac{b}{a}-\lambda \end{vmatrix} = -\lambda\left(-\frac{b}{a}-\lambda\right) + \frac{c}{a} = \lambda^2 + \frac{b}{a}\lambda + \frac{c}{a} = 0.$$

Multiplying this equation by a leads to the equivalent equation $a\lambda^2 + b\lambda + c = 0$, with solutions

$$\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}:$$

note that the eigenvalues are complex if $b^2 - 4ac < 0$. Table 6.5 contains some possible choices for a, b, and c, the corresponding eigenvalues and the stability result.

Table 6.5: Analytical stability investigation for initial-value problem of Example 6.8.1.

a	b	c	λ_1	λ_2	stability result
1	2	0	0	-2	stable
1	2	1	-1	-1	absolutely stable
1	0	1	$0+i$	$0-i$	stable
1	2	2	$-1+i$	$-1-i$	absolutely stable
1	-2	0	2	0	unstable

6.8.2 Motivation of analytical stability *

If $\mathbf{A}$ is diagonalizable, then there exist a non-singular matrix $\mathbf{S}$ and a matrix $\mathbf{\Lambda}$ such that

$$\mathbf{A} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}. \tag{6.55}$$

Here, $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_m)$ contains the eigenvalues of $\mathbf{A}$ (which can be complex)[2], and $\mathbf{S}$ is the corresponding eigenvector matrix having the (right) eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ of $\mathbf{A}$ as columns. For the sake of completeness we recall the definition: $\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_j$, $j = 1, \ldots, m$.

Let $\boldsymbol{\eta} = \mathbf{S}^{-1}\boldsymbol{\varepsilon}$, such that $\boldsymbol{\varepsilon} = \mathbf{S}\boldsymbol{\eta}$. Substituting this transformation in (6.54) and using factorization (6.55) yields

$$\begin{cases} \mathbf{S}\boldsymbol{\eta}' = \mathbf{A}\mathbf{S}\boldsymbol{\eta} &= \mathbf{S}\mathbf{\Lambda}\boldsymbol{\eta}, \\ \boldsymbol{\eta}(t_0) = \mathbf{S}^{-1}\boldsymbol{\varepsilon}_0 = \boldsymbol{\eta}_0. \end{cases}$$

[2] We write $\text{diag}(\lambda_1, \ldots, \lambda_m)$ to denote a diagonal matrix with $\lambda_1, \ldots, \lambda_m$ on the main diagonal.

Since $\mathbf{S}$ is non-singular, this yields

$$\begin{cases} \boldsymbol{\eta}' = \boldsymbol{\Lambda}\boldsymbol{\eta}, \\ \boldsymbol{\eta}(t_0) = \boldsymbol{\eta}_0. \end{cases} \tag{6.56}$$

Note that the above system is fully decoupled.

For each component of the vector $\boldsymbol{\eta}$, the solution is given by

$$\eta_j = \eta_{j,0} e^{\lambda_j (t-t_0)},\ j = 1, \ldots, m.$$

Note that in general, the eigenvalues can be complex. Writing each eigenvalue as

$$\lambda_j = \mu_j + i\nu_j,$$

with $\mu_j, \nu_j \in \mathbb{R}$ it follows that

$$\eta_j = \eta_{j,0} e^{\mu_j (t-t_0)} e^{i\nu_j (t-t_0)}.$$

Because $|e^{i\nu_j (t-t_0)}| = 1$ (since $|e^{i\xi}| = |\cos\xi + i\sin\xi| = \sqrt{\cos^2\xi + \sin^2\xi} = 1$ for all $\xi \in \mathbb{R}$), it holds that $|\eta_j| = |\eta_{j,0}| e^{\mu_j (t-t_0)}$. Therefore,

$$\lim_{t\to\infty} |\eta_j| = \begin{cases} |\eta_{j,0}|, & \text{if } \mu_j = 0, \\ 0, & \text{if } \mu_j < 0, \\ \infty, & \text{if } \mu_j > 0. \end{cases}$$

From this observation it follows that the system is stable if $\mu_j = \mathrm{Re}(\lambda_j) \leq 0$ for all $j = 1, \ldots, m$. Absolute stability is obtained when $\mu_j = \mathrm{Re}(\lambda_j) < 0$ for all $j = 1, \ldots, m$. If there exists a j for which $\mu_j = \mathrm{Re}(\lambda_j) > 0$, then the system is unstable.

6.8.3 Amplification matrix

For each numerical time-integration method, the amplification matrix can be computed by applying the time-integration method to (6.54). We obtain

Forward Euler method: $$\mathbf{Q}(\Delta t\mathbf{A}) = \mathbf{I} + \Delta t\mathbf{A}, \tag{6.57a}$$

Backward Euler method: $$\mathbf{Q}(\Delta t\mathbf{A}) = (\mathbf{I} - \Delta t\mathbf{A})^{-1}, \tag{6.57b}$$

Trapezoidal method: $$\mathbf{Q}(\Delta t\mathbf{A}) = \left(\mathbf{I} - \frac{1}{2}\Delta t\mathbf{A}\right)^{-1} \cdot \left(\mathbf{I} + \frac{1}{2}\Delta t\mathbf{A}\right), \tag{6.57c}$$

Modified Euler method: $$\mathbf{Q}(\Delta t\mathbf{A}) = \mathbf{I} + \Delta t\mathbf{A} + \frac{\Delta t^2}{2}\mathbf{A}^2, \tag{6.57d}$$

RK4 method: $$\mathbf{Q}(\Delta t\mathbf{A}) = \mathbf{I} + \Delta t\mathbf{A} + \frac{\Delta t^2}{2}\mathbf{A}^2 + \frac{\Delta t^3}{6}\mathbf{A}^3 + \frac{\Delta t^4}{24}\mathbf{A}^4. \tag{6.57e}$$

6.8.4 Numerical stability of the test system

As has been mentioned in Section 6.8.1 and motivated in Section 6.8.2, the stable integration of system (6.53) is only possible if all eigenvalues of the matrix $\mathbf{A}$ have non-positive real parts. These are stable systems with non-increasing solutions.

In Section 6.8.5, the numerical stability of the time-integration methods for systems will be derived. The most important conclusion is that for numerical stability, the scalar amplification factor can be used. If

$$|Q(\lambda_j \Delta t)| \leq 1$$

for all eigenvalues λ_j of $\mathbf{A}$ ($j = 1, \ldots, m$), then the numerical method is stable. Absolute stability is obtained when $|Q(\lambda_j \Delta t)| < 1$ for all eigenvalues, and instability when there is at least one

eigenvalue for which $|Q(\lambda_j \Delta t)| > 1$. This leads to the following characterization of stability for systems (cf. equations (6.17)):

$$\begin{array}{rcll} \text{absolute stability} & \Longleftrightarrow & \text{for all } \lambda_j : & |Q(\lambda_j \Delta t)| < 1, \\ \text{stability} & \Longleftrightarrow & \text{for all } \lambda_j : & |Q(\lambda_j \Delta t)| \leq 1, \\ \text{no stability} & \Longleftrightarrow & \text{there exists a } \lambda_j : & |Q(\lambda_j \Delta t)| > 1. \end{array}$$

Note that in general, λ_j is complex, such that the modulus of $Q(\lambda_j \Delta t)$ needs to be computed (see definition 1.6.3).

In this section, we will investigate the stability conditions for several time-integration methods.

Example 6.8.2 (Stability condition for the Forward Euler method)

Absolute stability for the Forward Euler approximation of a system of differential equations is obtained when all eigenvalues λ_j of the matrix $\mathbf{A}$ satisfy the strict inequality

$$| \, 1 + \lambda_j \Delta t \, | < 1, \tag{6.58}$$

$j = 1, \ldots, m$. Writing $\lambda_j = \mu_j + i\nu_j$, with $\mu_j, \nu_j \in \mathbb{R}$ yields (apply definition 1.6.3)

$$| \, 1 + \lambda_j \Delta t \, | = \, | \, (1 + \mu_j \Delta t) + i\nu_j \Delta t \, | = \sqrt{(1 + \mu_j \Delta t)^2 + \nu_j^2 \Delta t^2}.$$

Taking squares on both sides of the stability condition (6.58) yields the following inequality:

$$(1 + \mu_j \Delta t)^2 + \nu_j^2 \Delta t^2 < 1. \tag{6.59}$$

This inequality has to be satisfied by all eigenvalues of matrix $\mathbf{A}$ to ensure absolutely stable numerical integration. Note that the above equation cannot be satisfied if eigenvalue λ_j has a positive real part ($\mu_j > 0$). Hence, if at least one eigenvalue of the matrix $\mathbf{A}$ has a positive real part, then the system is unstable.

The corresponding stability condition for Δt follows from the equation $(1 + \Delta t \mu_j)^2 + \nu_j^2 \Delta t^2 = 1$. The nonzero root of this quadratic polynomial in Δt is

$$\Delta t = -\frac{2\mu_j}{\mu_j^2 + \nu_j^2}.$$

For each eigenvalue λ_j, the stability condition is given by

$$\Delta t < -\frac{2\mu_j}{\mu_j^2 + \nu_j^2}, \; j = 1, \ldots, m.$$

If λ_j is real-valued, then this condition can be related to the scalar case (condition (6.18)). Ensuring that the above condition is satisfied for all eigenvalues, the integration of the system is absolutely stable if Δt satisfies

$$\Delta t < \min_{j=1,\ldots,m} \left\{ -\frac{2\mu_j}{\mu_j^2 + \nu_j^2} \right\}. \tag{6.60}$$

For an imaginary eigenvalue ($\mu_j = 0$), the Forward Euler method is always unstable.

Example 6.8.3 (Unconditional stability of the Backward Euler method)

Stable integration using the Backward Euler method is obtained when

$$|Q(\lambda_j \Delta t)| = \frac{1}{|1 - \lambda_j \Delta t|} \leq 1$$

for all eigenvalues $\lambda_j = \mu_j + i\nu_j$ of $\mathbf{A}$. Note that this is equivalent to $|1 - \lambda_j \Delta t| \geq 1$, which holds when

$$| 1 - \lambda_j \Delta t |^2 = | 1 - \mu_j \Delta t - i\nu_j \Delta t |^2 = (1 - \mu_j \Delta t)^2 + (\nu_j \Delta t)^2 \geq 1. \tag{6.61}$$

If all eigenvalues have non-positive real part μ_j (which is required to obtain an analytically stable system), then equation (6.61) is automatically satisfied, independent of the time step Δt, and therefore, $|Q(\lambda_j \Delta t)| \leq 1$. In that case, the Backward Euler method is unconditionally stable.

If all eigenvalues are imaginary ($\mu_j = 0$), then the Backward Euler method is absolutely stable, since $|Q(\lambda_j \Delta t)| < 1$ for all eigenvalues.

Example 6.8.4 (Unconditional stability Trapezoidal method)

For the Trapezoidal method, the amplification factor for eigenvalue $\lambda_j = \mu_j + i\nu_j$ is

$$\begin{aligned} | Q(\lambda_j \Delta t) | &= \left| \frac{1 + \frac{1}{2}\lambda_j \Delta t}{1 - \frac{1}{2}\lambda_j \Delta t} \right| = \frac{| (1 + \frac{1}{2}\mu_j \Delta t) + \frac{1}{2} i\nu_j \Delta t |}{| (1 - \frac{1}{2}\mu_j \Delta t) - \frac{1}{2} i\nu_j \Delta t |} \\ &= \frac{\sqrt{(1 + \frac{1}{2}\mu_j \Delta t)^2 + (\frac{1}{2}\nu_j \Delta t)^2}}{\sqrt{(1 - \frac{1}{2}\mu_j \Delta t)^2 + (\frac{1}{2}\nu_j \Delta t)^2}}. \end{aligned}$$

For eigenvalues with non-positive real part ($\mu_j \leq 0$) the numerator is always smaller than or equal to the denominator, and hence, $|Q(\lambda_j \Delta t)| \leq 1$. If this holds for all eigenvalues, then the Trapezoidal method is stable for each time step, hence this method is unconditionally stable.

If all eigenvalues are imaginary ($\mu_j = 0$), then the Trapezoidal Method is stable, since $|Q(\lambda_j \Delta t)| = 1$ for all eigenvalues.

6.8.5 Motivation of numerical stability *

Theorem 6.8.1 *Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be diagonalizable, and $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_m)$ be the matrix with the eigenvalues of $\mathbf{A}$, with corresponding eigenvector matrix $\mathbf{S}$ (having right eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$ of $\mathbf{A}$ as columns). Then for each numerical time-integration method, the amplification matrix, $\mathbf{Q}(\Delta t\mathbf{A})$, is diagonalizable, and its eigenvalues are given by $Q(\lambda_1 \Delta t), \ldots, Q(\lambda_m \Delta t)$: $\mathbf{Q}(\Delta t\mathbf{A}) = \mathbf{S}\mathbf{Q}(\mathbf{\Delta t \Lambda})\mathbf{S}^{-1}$.*

Proof
First, it should be noticed that

$$\mathbf{A}^k = (\mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1})^k = \mathbf{S}\mathbf{\Lambda}^k\mathbf{S}^{-1}, \text{ for } k \in \mathbb{Z}.$$

From this observation it follows that any polynomial

$$P(x) = p_0 + p_1 x + \ldots + p_{n_p} x^{n_p},\ n_p \in \mathbb{N},$$

evaluated in $\Delta t\mathbf{A}$ equals

$$P(\Delta t\mathbf{A}) = \sum_{k=0}^{n_p} p_k (\Delta t\mathbf{A})^k = \sum_{k=0}^{n_p} p_k \mathbf{S}(\Delta t\mathbf{\Lambda})^k \mathbf{S}^{-1} = \mathbf{S} \left(\sum_{k=0}^{n_p} p_k (\Delta t\mathbf{\Lambda})^k \right) \mathbf{S}^{-1} = \mathbf{S}P(\Delta t\mathbf{\Lambda})\mathbf{S}^{-1}, \tag{6.62}$$

in which $P(\Delta t\mathbf{\Lambda}) = \text{diag}(P(\lambda_1 \Delta t), \ldots, P(\lambda_m \Delta t))$.

Next, note that each amplification matrix in Section 6.8.3 can be written as

$$\mathbf{Q}(\Delta t\mathbf{A}) = R(\Delta t\mathbf{A})^{-1} P(\Delta t\mathbf{A}), \tag{6.63}$$

in which $P(\Delta t\mathbf{A})$ and $R(\Delta t\mathbf{A})$ are polynomials in $\Delta t\mathbf{A}$. For example, for the Trapezoidal method, we have

$$R(\Delta t\mathbf{A}) = \mathbf{I} - \frac{1}{2}\Delta t\mathbf{A}, \text{ and } P(\Delta t\mathbf{A}) = \mathbf{I} + \frac{1}{2}\Delta t\mathbf{A}.$$

Using equation (6.62) in equation (6.63) yields

$$\mathbf{Q}(\Delta t\mathbf{A}) = (\mathbf{S}R(\Delta t\mathbf{\Lambda})\mathbf{S}^{-1})^{-1}(\mathbf{S}P(\Delta t\mathbf{\Lambda})\mathbf{S}^{-1}) = \mathbf{S}R(\Delta t\mathbf{\Lambda})^{-1}P(\Delta t\mathbf{\Lambda})\mathbf{S}^{-1}. \tag{6.64}$$

Note that $R(\Delta t\mathbf{\Lambda})^{-1}$ and $P(\Delta t\mathbf{\Lambda})$ are both diagonal matrices, and therefore, their product is also a diagonal matrix:

$$\begin{aligned} R(\Delta t\mathbf{\Lambda})^{-1}P(\Delta t\mathbf{\Lambda}) &= \mathrm{diag}(P(\lambda_1\Delta t)/R(\lambda_1\Delta t), \ldots, P(\lambda_m\Delta t)/R(\lambda_m\Delta t)) \\ &= \mathrm{diag}(Q(\lambda_1\Delta t), \ldots, Q(\lambda_m\Delta t)) = \mathbf{Q}(\Delta t\mathbf{\Lambda}). \end{aligned} \tag{6.65}$$

The above derivation makes use of the fact that the amplification matrix is built using the scalar amplification factors.

Substituting the expression (6.65) in equation (6.64) proves that $\mathbf{Q}(\Delta t\mathbf{A}) = \mathbf{S}\mathbf{Q}(\mathbf{\Delta t\Lambda})\mathbf{S^{-1}}$. ⊠

In order to investigate numerical stability of time-integration methods, the test system (6.54) is used. For each time-integration method, the approximation at time t_{n+1} equals

$$\boldsymbol{\varepsilon}_{n+1} = \mathbf{Q}(\Delta t\mathbf{A})\boldsymbol{\varepsilon}_n. \tag{6.66}$$

Next, using that $\boldsymbol{\varepsilon} = \mathbf{S}\boldsymbol{\eta}$ (as in Section 6.8.2), equation (6.66) is transformed into (using Theorem 6.8.1)

$$\mathbf{S}\boldsymbol{\eta}_{n+1} = \mathbf{Q}(\Delta t\mathbf{A})\mathbf{S}\boldsymbol{\eta}_n = \mathbf{S}\mathbf{Q}(\Delta t\mathbf{\Lambda})\boldsymbol{\eta}_n.$$

Because $\mathbf{S}$ is non-singular, this means that

$$\boldsymbol{\eta}_{n+1} = \mathbf{Q}(\Delta t\mathbf{\Lambda})\boldsymbol{\eta}_n,$$

which can be written componentwise as

$$\begin{aligned} \eta_{1,n+1} &= Q(\lambda_1\Delta t)\eta_{1,n}, \\ \eta_{2,n+1} &= Q(\lambda_2\Delta t)\eta_{2,n}, \\ &\vdots \\ \eta_{m,n+1} &= Q(\lambda_m\Delta t)\eta_{m,n}. \end{aligned}$$

Stability of $\boldsymbol{\eta}_n$ is equivalent with stability of $\boldsymbol{\varepsilon}_n$, since $\mathbf{S}$ does not depend on the approximation. This means that a sufficient condition for numerical stability is

$$|Q(\lambda_j\Delta t)| \leq 1, \quad j = 1, \ldots, m.$$

If for one of the eigenvalues $|Q(\lambda_j\Delta t)| > 1$, then the system is unstable.

6.8.6 Stability regions

In this section, the stability regions of the different methods are investigated graphically.

First of all, the Forward Euler method will be considered. In Figure 6.4(a), the region where this method is stable is depicted:

$$S_{FE} = \{\lambda\Delta t \in \mathbb{C} : |1 + \lambda\Delta t| \leq 1\}.$$

Note that this set represents a circle with midpoint $(-1, 0)$ and radius 1. For points $\lambda\Delta t \in S_{FE}$ the modulus of the amplification factor is less than 1 and outside the circle it is larger than 1. The region inside the circle is therefore called the *stability region of the Forward Euler method*. The stability region of the Forward Euler method lies completely to the left of the imaginary axis and is tangent to that axis.

The stability region may be used to provide a graphical estimate of the stability condition for Δt. As a starting point, each eigenvalue λ_j should be marked in the complex plane (which corresponds to assuming $\Delta t = 1$). The position of the marked point shows whether the integration is stable for $\Delta t = 1$ or not. If any marked point lies outside the stability region, then Δt must be taken smaller, such that $\lambda_j \Delta t$ falls inside the stability region. Graphically, this corresponds to moving from the marked point to the origin along a straight line. In many cases even the unassisted eye can determine fairly accurately for which Δt the line intersects the circle. This process has to be performed for all eigenvalues; the smallest Δt that results determines the stability bound.

Obviously, this operation will only succeed if all eigenvalues have a strictly negative real part. Shrinking Δt will never bring $\lambda \Delta t$ inside the stability region when an eigenvalue has positive or zero real part, the latter because the stability region is tangent to the imaginary axis. Imaginary eigenvalues are often found when representing systems with oscillatory solutions without damping.

In Figure 6.4, all stability regions can be seen for the different numerical time-integration methods. Note that both implicit methods (Backward Euler and Trapezoidal method) are unconditionally stable when all eigenvalues have non-positive real part. If λ_j falls in the white region of Figure 6.4(b) (unstable case of the Backward Euler method), then Δt has to be increased to reach the stability region.

The stability region of the Modified Euler method is very similar to that of the Forward Euler method. As a consequence both methods have almost the same time step constraint. In particular, both methods share the property that systems containing imaginary eigenvalues cannot be integrated in a stable way, whichever the time step is: unbounded error growth will ensue.

The RK4 method is conditionally stable. It is important to note that this is the only discussed explicit method that may be stable for systems which have imaginary eigenvalues. In that case, RK4 is stable if $|\lambda \Delta t| \leq 2.8$.

Note that the observations in this section can be related to Section 6.4.1, in which numerical stability was derived.

6.8.7 Stability of general systems

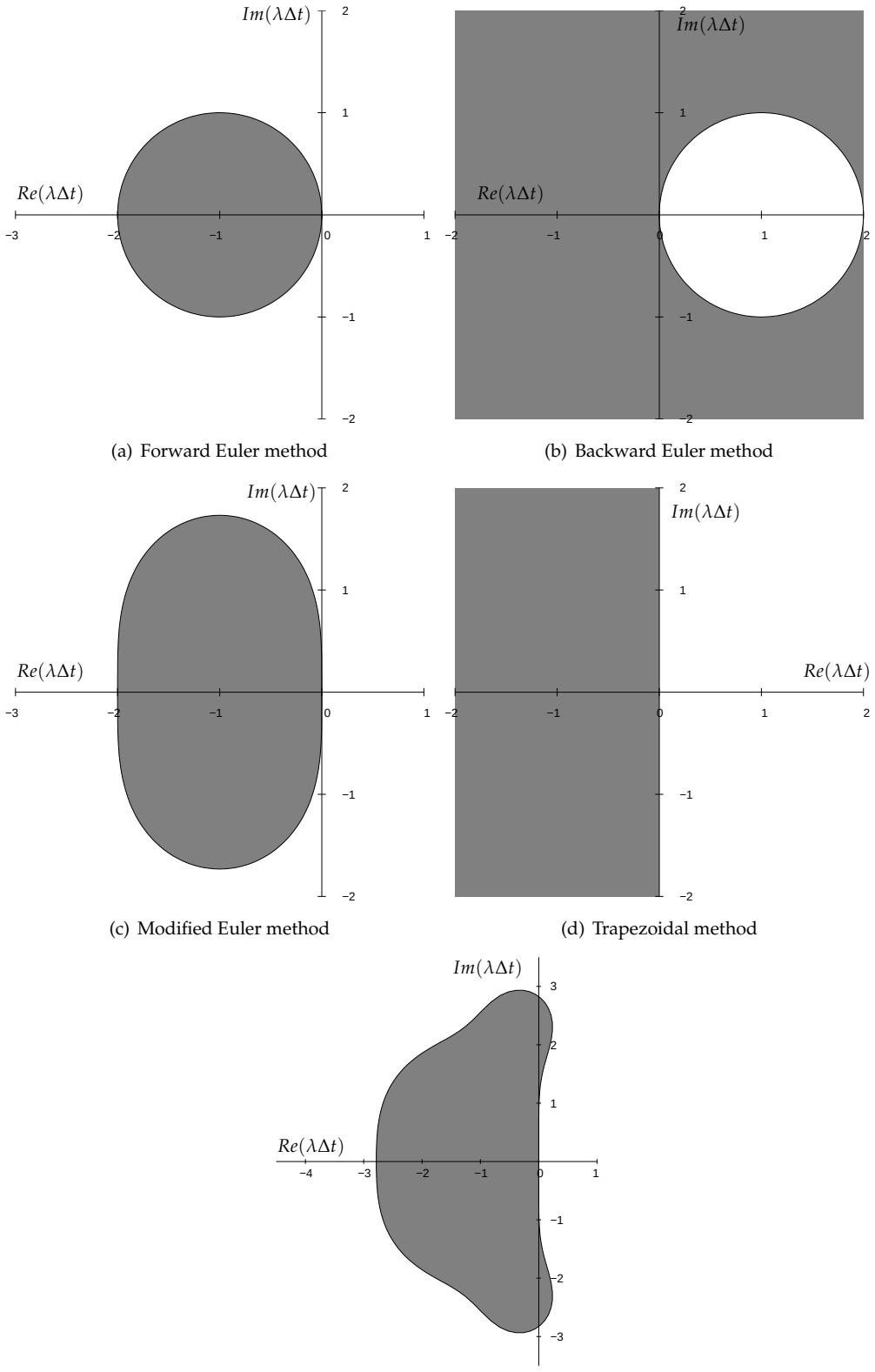
Differential equations for systems of the form

$$\begin{cases} \mathbf{y}' &= \mathbf{f}(t, \mathbf{y}), \\ \mathbf{y}(t_0) &= \mathbf{y}_0, \end{cases}$$

locally have the same properties as the linear system (6.53). The stability behavior can be determined by linearization of the initial-value problem about $(\hat{t}, \hat{\mathbf{y}})$ (note that a similar approach was adopted in Section 6.4.1 for general scalar equations). The role of $\mathbf{A}$ is taken over by the Jacobian matrix of $\mathbf{f}$, evaluated in $(\hat{t}, \hat{\mathbf{y}})$:

$$\mathbf{J}|_{(\hat{t},\hat{\mathbf{y}})} = \left(\begin{array}{ccc} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_m} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial y_1} & \cdots & \frac{\partial f_m}{\partial y_m} \end{array} \right) \Bigg|_{(\hat{t},\hat{\mathbf{y}})} . \tag{6.67}$$

Note that the eigenvalues of the Jacobian matrix, $\lambda_1(\hat{t}, \hat{\mathbf{y}}), \ldots, \lambda_m(\hat{t}, \hat{\mathbf{y}})$, depend on $\hat{t}$ and $\hat{\mathbf{y}}$. The stability properties of numerical time-integration methods therefore depend on time and approximation.



(a) Forward Euler method

(b) Backward Euler method

(c) Modified Euler method

(d) Trapezoidal method

(e) RK4 method

Figure 6.4: Stability regions of numerical time-integration methods.

Example 6.8.5 (Competing species)

In this example, we investigate the population densities of two species of bacteria competing with each other for the same supply of food. A model for this competition leads to the system of differential equations

$$\begin{cases} y_1' = y_1(1 - y_1 - y_2), \\ y_2' = y_2(0.5 - 0.75y_1 - 0.25y_2), \end{cases}$$

in which y_1 and y_2 are the population densities of the bacteria. The solution to this system of equations is unknown, and therefore numerical time-integration methods are applied.

Assume that the approximation at time t_n is given by $\mathbf{w}_n = (1.5, 0)^\top$. We choose $\hat{t} = t_n, \hat{\mathbf{y}} = \mathbf{w}_n$, and linearize the system about $(\hat{t}, \hat{\mathbf{y}})$. The Jacobian matrix of this model, evaluated at $(\hat{t}, \hat{\mathbf{y}})$, is given by

$$\mathbf{J}|_{(\hat{t},\hat{\mathbf{y}})} = \begin{pmatrix} 1 - 2\hat{y}_1 - \hat{y}_2 & -\hat{y}_1 \\ -0.75\hat{y}_2 & 0.5 - 0.75\hat{y}_1 - 0.5\hat{y}_2 \end{pmatrix} = \begin{pmatrix} -2 & -1.5 \\ 0 & -0.625 \end{pmatrix},$$

with eigenvalues $\lambda_1(\hat{t}, \hat{\mathbf{y}}) = -2$, $\lambda_2(\hat{t}, \hat{\mathbf{y}}) = -0.625$.

In order to compute the approximation at time t_{n+1}, the Forward Euler method is used. Since the eigenvalues are real, criterion (6.18) is applied to each eigenvalue to show that the method is stable in $(\hat{t}, \hat{\mathbf{y}})$ if

$$\Delta t \leq \min_{j=1,2} \left\{ -\frac{2}{\lambda_j(\hat{t}, \hat{\mathbf{y}})} \right\} = 1.$$

Note that the same procedure needs to be executed at each single time step.

Example 6.8.6 (Stability of mathematical pendulum)

In this example, the mathematical pendulum of example 6.7.1 is revisited. Note that the corresponding system can be written as

$$\begin{cases} y_1' &= \quad y_2 \quad &= f_1(t, \mathbf{y}), \\ y_2' &= -\sin y_1 &= f_2(t, \mathbf{y}), \end{cases}$$

with $\mathbf{y} = (y_1, y_2)^\top$. The Jacobian matrix, evaluated at $(\hat{t}, \hat{\mathbf{y}})$, equals

$$\mathbf{J}|_{(\hat{t},\hat{\mathbf{y}})} = \begin{bmatrix} 0 & 1 \\ -\cos \hat{y}_1 & 0 \end{bmatrix}.$$

If $-\pi/2 < \hat{y}_1 < \pi/2$, then the eigenvalues of this matrix are imaginary, which are given by

$$\lambda_{1,2}(\hat{t}, \hat{\mathbf{y}}) = \pm i\sqrt{\cos \hat{y}_1}.$$

In Figures 6.4(a) and 6.4(c), it can be seen that the Forward Euler method and the Modified Euler method are unstable for every Δt when the eigenvalues are imaginary (this has also been shown for the Forward Euler method in example 6.8.2).

However, the Backward Euler method and the Trapezoidal method are unconditionally stable (cf. Figures 6.4(b) and 6.4(d)). We remark that the amplification factor of the Backward Euler method satisfies $|Q(\lambda_{1,2}(\hat{t}, \hat{\mathbf{y}}))| < 1$, such that a damped oscillation is found when this method is applied (cf. example 6.8.3). However, since imaginary eigenvalues imply neither losses (dissipation or damping) nor growth of the solution, the Trapezoidal method is more useful in this case ($|Q(\lambda_{1,2}(\hat{t}, \hat{\mathbf{y}}))| = 1$, see example 6.8.4).

The RK4 method is stable if (cf. Figure 6.4(e))

$$\Delta t \leq \frac{2.8}{|\lambda_{1,2}(\hat{t}, \hat{\mathbf{y}})|} = \frac{2.8}{\sqrt{\cos \hat{y}_1}},$$

which means that a suitable time step for the RK4 method depends on $\hat{\mathbf{y}}$ and $\hat{t}$. If Δt is strictly smaller than the stability bound, then the RK4 method gives a damped oscillation.

6.9 Global truncation error for systems

The analogy of the scalar and vectorial case extends further to global truncation errors. As in the scalar case, $\mathbf{y}_{n+1}$ is the solution at time t_{n+1}, and $\mathbf{z}_{n+1}$ is the approximation at time t_{n+1} after applying one step of the numerical time-integration method to $\mathbf{y}_n$. Then, the local truncation-error vector is given by

$$\boldsymbol{\tau}_{n+1} = \frac{\mathbf{y}_{n+1} - \mathbf{z}_{n+1}}{\Delta t}, \tag{6.68}$$

and the global truncation-error vector is given by $\mathbf{e}_{n+1} = \mathbf{y}_{n+1} - \mathbf{w}_{n+1}$. It turns out that the orders of these errors are equal to the corresponding orders in the scalar case (Section 6.9.1). For example, the Forward Euler method for systems has a local truncation error of order $\mathcal{O}(\Delta t)$, and if the method is stable and consistent, then the global truncation error is of the same order.

For systems, Richardson's extrapolation should be performed componentwise to obtain global truncation-error estimates (Section 6.6).

6.9.1 Motivation of the global truncation error for systems *

Test system
First, the test system $\mathbf{y}' = \mathbf{A}\mathbf{y}$ will be investigated (cf. proof of Theorem 6.4.1). In that case, the numerical approximation at time t_{n+1} equals $\mathbf{w}_{n+1} = \mathbf{Q}(\Delta t\mathbf{A})\mathbf{w}_n$ (equation 6.66), such that $\mathbf{z}_{n+1} = \mathbf{Q}(\Delta t\mathbf{A})\mathbf{y}_n$. The local truncation-error vector is therefore given by (cf. equation (6.31))

$$\boldsymbol{\tau}_{n+1} = \frac{\mathbf{y}_{n+1} - \mathbf{Q}(\Delta t\mathbf{A})\mathbf{y}_n}{\Delta t},$$

and the global truncation-error vector by $\mathbf{e}_{n+1} = \mathbf{y}_{n+1} - \mathbf{Q}(\Delta t\mathbf{A})\mathbf{w}_n$. Because $\mathbf{w}_n = \mathbf{y}_n - \mathbf{e}_n$, the global truncation-error vector equals

$$\mathbf{e}_{n+1} = \mathbf{y}_{n+1} - \mathbf{Q}(\Delta t\mathbf{A})\mathbf{y}_n + \mathbf{Q}(\Delta t\mathbf{A})\mathbf{e}_n = \Delta t\boldsymbol{\tau}_{n+1} + \mathbf{Q}(\Delta t\mathbf{A})\mathbf{e}_n. \tag{6.69}$$

Next, equation (6.69) is decoupled using the transformation $\mathbf{e}_{n+1} = \mathbf{S}\boldsymbol{\eta}_{n+1}$, where $\mathbf{S}$ is the matrix having the eigenvectors, $\mathbf{v}_1, \ldots, \mathbf{v}_m$, of $\mathbf{A}$ as columns. This means that

$$\mathbf{S}\boldsymbol{\eta}_{n+1} = \Delta t\boldsymbol{\tau}_{n+1} + \mathbf{Q}(\Delta t\mathbf{A})\mathbf{S}\boldsymbol{\eta}_n,$$

such that (multiply by $\mathbf{S}^{-1}$ and apply Theorem 6.8.1)

$$\boldsymbol{\eta}_{n+1} = \Delta t\mathbf{S}^{-1}\boldsymbol{\tau}_{n+1} + \mathbf{S}^{-1}\mathbf{Q}(\Delta t\mathbf{A})\mathbf{S}\boldsymbol{\eta}_n = \Delta t\mathbf{S}^{-1}\boldsymbol{\tau}_{n+1} + \mathbf{Q}(\Delta t\boldsymbol{\Lambda})\boldsymbol{\eta}_n, \tag{6.70}$$

in which $\boldsymbol{\Lambda}$ is the matrix having the eigenvalues of $\mathbf{A}$, $\lambda_1, \ldots, \lambda_m$, on the diagonal.

Next, the local truncation-error vector $\boldsymbol{\tau}_{n+1}$ is decomposed along the eigenvectors of $\mathbf{A}$:

$$\boldsymbol{\tau}_{n+1} = \sum_{j=1}^{m} \alpha_{j,n+1}\mathbf{v}_j = \mathbf{S}\boldsymbol{\alpha}_{n+1}. \tag{6.71}$$

The values $\alpha_{j,n+1}$ are the components of the local truncation-error vector with respect to the basis of eigenvectors.

Transforming $\mathbf{S}^{-1}\boldsymbol{\tau}_{n+1}$ into $\mathbf{S}^{-1}\mathbf{S}\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_{n+1}$, the decoupled global truncation-error vector of equation (6.70) equals

$$\boldsymbol{\eta}_{n+1} = \Delta t\boldsymbol{\alpha}_{n+1} + \mathbf{Q}(\Delta t\boldsymbol{\Lambda})\boldsymbol{\eta}_n.$$

As in equation (6.35), this results in

$$\boldsymbol{\eta}_{n+1} = \sum_{\ell=0}^{n} \left(\mathbf{Q}(\Delta t\boldsymbol{\Lambda})\right)^{\ell} \Delta t\boldsymbol{\alpha}_{n+1-\ell}.$$

Because $\mathbf{Q}(\Delta t\boldsymbol{\Lambda}) = \mathrm{diag}(Q(\lambda_1\Delta t), \ldots, Q(\lambda_m\Delta t))$, the components of the global truncation-error vector are given by

$$\eta_{j,n+1} = \sum_{\ell=0}^{n} \left(Q(\lambda_j\Delta t)\right)^{\ell} \Delta t\boldsymbol{\alpha}_{j,n+1-\ell} \quad (j = 1, \ldots, m). \tag{6.72}$$

Note that this is a decoupled system: the scalar properties of local and global truncation errors can be applied to each component (as in Sections 6.4.2 and 6.4.3).

This automatically entails that each component of the decoupled local truncation-error vectors $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{n+1}$ has the same order of magnitude as the scalar local truncation error of the corresponding method, for example of order $\mathcal{O}(\Delta t)$ for the Forward Euler method. If a numerical method is stable and consistent, then the decoupled global truncation error is of the same order as the decoupled local truncation error (see Theorem 6.4.1). Because the matrix $\mathbf{S}$ does not depend on Δt, the original local and global truncation error are also of the same order.

6.10 Stiff differential equations

Stiff differential equations (stiff systems) describe problems that exhibit transients. Their solution is the sum of a rapidly decaying part, the transient, and a slowly-varying part. After a short period of time, the transient is invisible, and only the slowly-varying part of the solution remains: the quasi-stationary solution.

In this section, we explain the concept of stiffness using the following initial-value problem with a linear differential equation:

$$\begin{cases} y' = \lambda(y - F(t)) + F'(t), & t > t_0, \\ y(t_0) = y_0, \end{cases}$$

with solution $y(t) = (y_0 - F(t_0))e^{\lambda(t-t_0)} + F(t)$, as may be verified directly by substitution. This initial-value problem is stiff if λ is strongly negative and if variations of F occur on a large time scale (slowly varying). Then, the transient is $(y_0 - F(t_0))e^{\lambda(t-t_0)}$ (rapidly decaying) and $F(t)$ is the quasi-stationary solution.

We consider approximations using a numerical time-integration method. When we choose an explicit method, then stability is obtained if the time step is chosen small enough. Since λ is strongly negative, the condition $|Q(\lambda\Delta t)| \leq 1$ leads to a very small bound for Δt. Note that this restriction is only due to the transient part of the solution, in which λ occurs. With respect to the accuracy to approximate the quasi-stationary solution, larger time steps could be taken. This means that the stability condition restricts the time step more than the accuracy requirement. Therefore, implicit time-integration methods (which are unconditionally stable) are preferable for stiff problems.

Considering the error behavior, it is important to note that local truncation errors are relatively large in the beginning, since the transient decays very rapidly. Since the quasi-stationary solution is slowly varying, these errors are smaller at later times. The global truncation error, that adapts

to these local truncation errors, also decreases after passing the transient phase. This can be derived from definition (6.35):

$$\begin{aligned} e_{n+1} &= \sum_{\ell=0}^{n} \left(Q(\lambda\Delta t)\right)^{\ell} \Delta t \tau_{n+1-\ell} \\ &= \left(Q(\lambda\Delta t)\right)^{n} \Delta t \tau_1 + \left(Q(\lambda\Delta t)\right)^{n-1} \Delta t \tau_2 + \ldots + Q(\lambda\Delta t)\Delta t \tau_n + \Delta t \tau_{n+1}. \end{aligned} \tag{6.73}$$

Hence if the time-integration method is stable and the amplification factor is small enough (for example, when $|Q(\lambda\Delta t)| \leq 0.5$), then, due to exponential decay, initial local truncation errors have much less influence than local truncation errors in the last few steps, which have not decayed that much. The smaller $|Q(\lambda\Delta t)|$ is, the faster the effect of past local truncation errors decays. The global truncation error could even be unnecessarily small when the time step is chosen very small in relation to the timescale of variation of F. This is not efficient if the approximation after a long time is required.

Implicit methods

In the previous discussion, it has been explained that explicit time-integration methods are not very useful for stiff systems, as the stability condition requires a very small time step. Therefore, it is more efficient to apply an implicit time-integration method, such as the Backward Euler, or the Trapezoidal method. These methods are unconditionally stable, and therefore, the time step can be taken arbitrarily large, at least theoretically. In practice, it needs to be chosen such that sufficient accurate approximations of the quasi-stationary solution are obtained. Note that for each time step, a system of algebraic equations has to be solved.

Both implicit methods have their unconditional stability in common but exhibit, in applications, significant difference in behavior as shown by the following experiment.

Example 6.10.1 (Stiff differential equation)

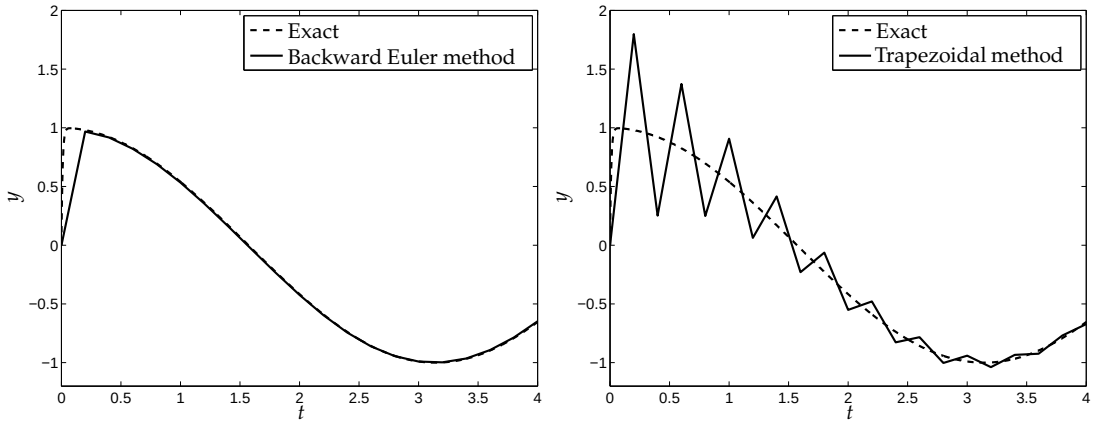
In this example, we investigate the scalar stiff problem

$$\begin{cases} y' = -100(y - \cos t) - \sin t, \quad t > 0, \\ y(0) = 0, \end{cases} \tag{6.74}$$

with solution $y(t) = -e^{-100t} + \cos t$ ($\lambda = -100$). The solution is approximated using the Backward Euler and the Trapezoidal method, in both cases with time step size 0.2. The size of the transient region is of order 0.01, which means that the first time step already exceeds this region. This means that the first local truncation error is large. The ease with which the global truncation error diminishes differs, as can be seen in Figure 6.5. Comparing the approximations to the solution makes it clearly visible how the neglected transient influences the global truncation error. The Backward Euler method is favorable: after four time steps the solution curve is almost reached. The Trapezoidal method needs more time steps to let the large initial local truncation error decay enough. This can be explained using the amplification factors of these methods (remember equations (6.15b) and (6.15c)):

$$\begin{aligned} \text{Backward Euler method:} \quad |Q(\lambda\Delta t)| &= \left|\tfrac{1}{1-\lambda\Delta t}\right| = \left|\tfrac{1}{1+100\cdot 0.2}\right| = \tfrac{1}{21} \approx 0.048, \\ \text{Trapezoidal method:} \quad |Q(\lambda\Delta t)| &= \left|\tfrac{1+\frac{1}{2}\lambda\Delta t}{1-\frac{1}{2}\lambda\Delta t}\right| = \left|\tfrac{1-\frac{1}{2}100\cdot 0.2}{1+\frac{1}{2}100\cdot 0.2}\right| = \tfrac{9}{11} \approx 0.82. \end{aligned}$$

Using these amplification factors in equation (6.73) shows that the initial local truncation error is damped out much faster when the Backward Euler method is used.



(a) Backward Euler method (b) Trapezoidal method

Figure 6.5: Approximation of the solution to problem (6.74) using implicit methods, $\Delta t = 0.2$.

Systems
In practice, stiffness is often encountered in systems. For systems of the form $\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{f}$, the solution is given by the sum of the homogeneous and the particular solution. The system is stiff if at least either of the following criteria is satisfied:

- Some of the real parts of the eigenvalues are strongly negative, whereas some of the other eigenvalues have a real part close to zero;
- The particular solution varies much more slowly than the homogeneous solution does.

In both cases, the essential feature is that the solution contains two timescales: a fast transient (which determines numerical stability of explicit time-integration methods) and a slowly-varying component.

Example 6.10.2 (Stiff systems)

Stiffness in systems occurs mostly in the following form. Suppose, for example, that $\mathbf{y}' = \mathbf{A}\mathbf{y}$, where $\mathbf{A}$ is a 2×2 matrix, having eigenvalues $\lambda_1 = -1$ and $\lambda_2 = -10000$. The solution is of the form $\mathbf{y}(t) = c_1\mathbf{v}_1 e^{-t} + c_2\mathbf{v}_2 e^{-10000t}$, in which $\mathbf{v}_1$ and $\mathbf{v}_2$ are the eigenvectors of $\mathbf{A}$ and the integration constants c_1 and c_2 are determined by the initial conditions of the two variables. The term proportional to $e^{-10000t}$ plays the role of the transient: this term vanishes much sooner than the term containing e^{-t}. The latter is slowly-varying compared to the transient and plays the role of the quasi-stationary solution. However, the transient still determines the stability condition (Forward Euler method: $\Delta t \leq 2/10000$) over the whole domain of integration. This inhibits adaptation of the time step to the relatively slowly-varying quasi-stationary part containing e^{-t}.

Definition 6.10.1 (Superstability) *A numerical method is called* superstable *if it is stable and*

$$\lim_{\lambda\Delta t \to -\infty} |Q(\lambda\Delta t)| < 1.$$

Figure 6.6 shows that the Backward Euler method is superstable, whereas for the Trapezoidal method,

$$\lim_{\lambda\Delta t \to -\infty} |Q(\lambda\Delta t)| = 1.$$

This means that initial perturbations in the fast components do not decay, or decay very slowly when using the Trapezoidal method.

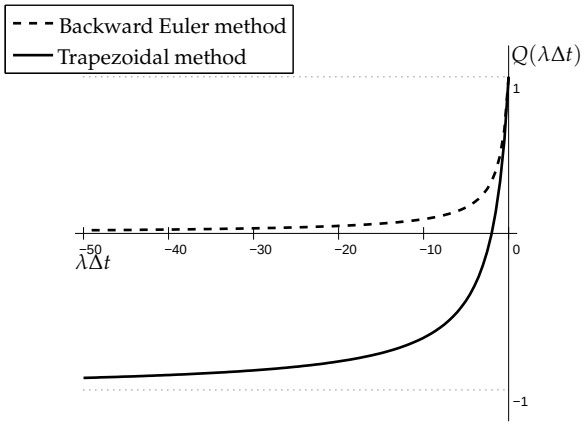


Figure 6.6: Amplification factors of the Backward Euler and Trapezoidal method.

This section only dealt with a simple system of differential equations, but stiffness may also occur in more complicated systems. Also in that case, implicit methods are to be recommended. However, at each time step, a nonlinear system of equations has to be solved, which increase the computational cost considerably. Therefore, the choice of a method is often a matter of careful consideration and explicit methods cannot be ruled out beforehand.

6.11 Multi-step methods *

All methods discussed so far in this chapter are single-step methods: the approximation at t_{n+1} depends solely on information of the previous point in time t_n. Although various methods use function evaluations at intermediate points, this information is only obtained and used inside the interval $[t_n, t_{n+1}]$.

Since approximations at times $t_0, t_1, \ldots, t_n$ are available, it seems reasonable to develop methods that use this information to design higher-order approximations. These methods are called *multi-step methods*. In general, ℓ-step methods require a single-step method (ideally of the same order) to create the required number of $w_0, w_1, \ldots, w_{l-1}$ values to start.

As a particular example of a two-step method, we consider the Adams-Bashforth method.

Adams-Bashforth method
For the derivation of the Adams-Bashforth method we start with the Trapezoidal method as constructed in (6.8):

$$w_{n+1} = w_n + \frac{\Delta t}{2}(f(t_n, w_n) + f(t_{n+1}, w_{n+1})). \tag{6.75}$$

The method can be made explicit by extrapolating to t_{n+1} using $f(t_{n-1}, w_{n-1})$ and $f(t_n, w_n)$. The linear interpolation polynomial based on t_{n-1} and t_n is given by

$$L_1(t) = f(t_{n-1}, w_{n-1}) + \frac{t - t_{n-1}}{\Delta t}(f(t_n, w_n) - f(t_{n-1}, w_{n-1}),$$

see equation (2.1). This means that $L_1(t_{n+1}) = 2f(t_n, w_n) - f(t_{n-1}, w_{n-1})$, and using this instead of $f(t_{n+1}, w_{n+1})$ in (6.75), the Adams-Bashforth method is obtained:

$$w_{n+1} = w_n + \frac{3}{2}\Delta t f(t_n, w_n) - \frac{1}{2}\Delta t f(t_{n-1}, w_{n-1}). \tag{6.76}$$

Note that only one function evaluation is required per time step, since $f(t_{n-1}, w_{n-1})$ has already been computed during the previous time step.

For $n = 1$, only one previous value is known, which is based on the initial condition. In order to compute a second value, a single-step method of the same order (in this case the Trapezoidal method, or the Modified Euler method) is applied to the initial condition. Setting w_0 equal to y_0 and w_1 to the single-step approximation for time t_1, the multi-step method can be applied to compute w_2. For $n \geq 2$, the multi-step algorithm is applied.

Stability
For the test equation $y' = \lambda y$, the time-integration method yields

$$w_{n+1} = \left(1 + \frac{3}{2}\lambda\Delta t\right) w_n - \frac{1}{2}\lambda\Delta t w_{n-1}. \tag{6.77}$$

In order to compute the amplification factor, we assume that

$$w_n = Q(\lambda\Delta t) w_{n-1} \quad \text{and} \quad w_{n+1} = Q(\lambda\Delta t)^2 w_{n-1}.$$

Using this in equation (6.77) yields

$$Q(\lambda\Delta t)^2 w_{n-1} = \left(1 + \frac{3}{2}\lambda\Delta t\right) Q(\lambda\Delta t) w_{n-1} - \frac{1}{2}\lambda\Delta t w_{n-1},$$

such that the amplification factor should satisfy

$$Q(\lambda\Delta t)^2 - \left(1 + \frac{3}{2}\lambda\Delta t\right) Q(\lambda\Delta t) + \frac{1}{2}\lambda\Delta t = 0.$$

The roots of this quadratic polynomial are

$$Q_1(\lambda\Delta t) = \frac{1 + \frac{3}{2}\lambda\Delta t + \sqrt{D}}{2},$$
$$Q_2(\lambda\Delta t) = \frac{1 + \frac{3}{2}\lambda\Delta t - \sqrt{D}}{2},$$

in which

$$D = 1 + \lambda\Delta t + \frac{9}{4}(\lambda\Delta t)^2.$$

As usual, the Adams-Bashforth method is stable if

$$|Q_{1,2}(\lambda\Delta t)| \leq 1. \tag{6.78}$$

The values of $Q_{1,2}(\lambda\Delta t)$ are visualized in Figure 6.7. Note that stability bound (6.78) is satisfied for all values of $Q_1(\lambda\Delta t)$, but $|Q_2(\lambda\Delta t)| \leq 1$ implies $\lambda\Delta t \geq -1$ (see lower bound in Figure 6.7). Therefore, the time step should be bounded by

$$\Delta t \leq -\frac{1}{\lambda}$$

in order to have a stable method.

Local truncation error
In order to derive the local truncation error, we use a Taylor expansion of the amplification factors. Here, we make use of the fact that for x sufficiently close to zero,

$$\sqrt{1+x} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \ldots.$$

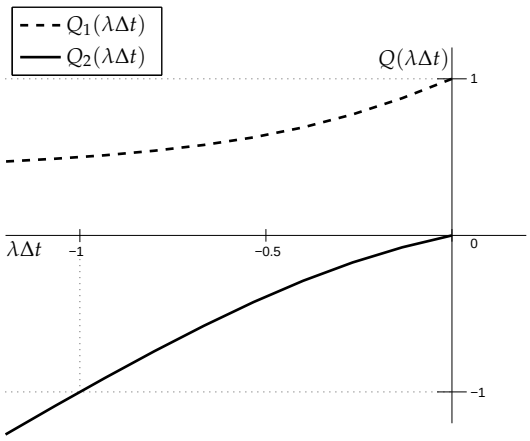


Figure 6.7: Amplification factors $Q_{1,2}(\lambda\Delta t)$ of the Adams-Bashforth method. The stability bounds are also visualized.

Applying this expansion to approximate $\sqrt{D}$ (taking $x = \lambda\Delta t + 9/4(\lambda\Delta t)^2$) the amplification factors equal

$$Q_1(\lambda\Delta t) = 1 + \lambda\Delta t + \frac{1}{2}(\lambda\Delta t)^2 - \frac{1}{4}(\lambda\Delta t)^3 + \mathcal{O}(\Delta t^4),$$
$$Q_2(\lambda\Delta t) = \frac{1}{2}\lambda\Delta t + \mathcal{O}(\Delta t^2).$$

The use of $Q_1(\lambda\Delta t)$ leads to a local truncation error for the test equation of (cf. equation (6.33))

$$\tau_{n+1} = \frac{e^{\lambda\Delta t} - Q_1(\lambda\Delta t)}{\Delta t} y_n = \frac{5}{12}\lambda^3\Delta t^2 y_n + \mathcal{O}(\Delta t^3) = \mathcal{O}(\Delta t^2), \tag{6.79}$$

whereas $Q_2(\lambda\Delta t)$ has a local truncation error of order $\mathcal{O}(1)$. This is the reason why $Q_1(\lambda\Delta t)$ is called the *principal root*, whereas $Q_2(\lambda\Delta t)$ is called the *spurious root*. This root does not belong to the differential equation, but it is a consequence of the chosen numerical method.

Comparison with the Modified Euler method
Regarding stability the time step in the Adams-Bashforth method has to be chosen twice as small as in the Modified Euler method. Since the Modified Euler method costs twice the amount of work in each time step the methods are comparable in this respect.

To obtain reasonable accuracy we choose the time step in such a way that the local truncation error, τ_{n+1}, is less than ε. For the Adams-Bashforth method it follows that (using equation (6.79) and neglecting higher-order terms)

$$\Delta t_{AB} = \sqrt{-\frac{12\varepsilon}{5\lambda^3}}.$$

For the Modified Euler method, the local truncation error for the test equation is (using the amplification factor as given by equation (6.15d))

$$\tau_{n+1} = \frac{1}{6}\lambda^3\Delta t^2 + \mathcal{O}(\Delta t^3),$$

such that the same approach yields the time step

$$\Delta t_{ME} = \sqrt{-\frac{6\varepsilon}{\lambda^3}}.$$

This means that the time step of the Adams-Bashforth method is $\sqrt{2/5} \approx 0.63$ times the time step of the Modified Euler method.

Because the Adams-Bashforth method reuses previous approximations, two steps of the Adams-Bashforth method require the same amount of work as one step of the Modified Euler method. This means that with the Adams-Bashforth method, the approximation at time interval $2\Delta t_{AB}$ can be computed using the same amount of work as with the Modified Euler method on interval Δt_{ME}, which is $2 \cdot \sqrt{2/5} \approx 1.26$ times as long as the Modified Euler method's time interval. Hence the Adams-Bashforth method requires less work than the Modified Euler method.

Discussion
Multi-step methods are less popular than Runge-Kutta methods. With multi-step methods starting problems occur, since approximations should be known at several time steps. Moreover, keeping track of the 'spurious roots' is a difficult matter. Finally, adaptive time-stepping is more difficult to apply.

6.12 Summary

In this chapter, the following subjects have been discussed:

- Theory of initial-value problems;
- Elementary single-step methods (Forward Euler, Backward Euler, Trapezoidal, and Modified Euler method);
- Analysis of numerical time-integration methods:
    - Test equation, amplification factor, stability;
    - Local truncation error;
    - Consistency, convergence, global truncation error;
- Higher-order time-integration methods (RK4 method);
- Global truncation error and Richardson error estimates;
- Numerical methods for systems:
    - Test system, amplification matrix, stability;
    - Local and global truncation error;
- Stiff systems, superstability;
- Multi-step method (Adams-Bashforth method).

6.13 Exercises

1. Use the Modified Euler method to approximate the solution to the initial-value problem

$$\begin{cases} y' = 1 + (t-y)^2, & 2 \leq t \leq 3, \\ y(2) = 1, \end{cases}$$

with $\Delta t = 0.5$.

The solution is given by

$$y(t) = t + \frac{1}{1-t}.$$

Determine the error in the numerical approximation.

2. The Midpoint method is given by

$$w_{n+1} = w_n + \Delta t f(t_n + \frac{\Delta t}{2}, w_n + \frac{\Delta t}{2} f(t_n, w_n)).$$

Show that the local truncation error of the Midpoint method is $\mathcal{O}(\Delta t^2)$.

3. Determine the amplification factor of the Trapezoidal method. Estimate with this amplification factor the local truncation error of the test equation. Show that the Trapezoidal method is stable for all $\Delta t > 0$, if $\lambda \leq 0$.

4. Consider the nonlinear initial-value problem: $y' = 1 + (t - y)^2$. Give the stability condition of the Modified Euler method at the point $t = 2$ and $y = 1$.

5. Consider the numerical time-integration method

$$\begin{aligned} \bar{w}_{n+1} &= w_n + \beta \Delta t f(t_n, w_n), \\ w_{n+1} &= \bar{w}_{n+1} + (1 - \beta)\Delta t f(t_n + \beta \Delta t, \bar{w}_{n+1}). \end{aligned}$$

   (a) Show that the local truncation error is $\mathcal{O}(\Delta t)$ for each value of β. Moreover, show that there is no β such that the truncation error is $\mathcal{O}(\Delta t^2)$.

   (b) Determine the amplification factor of this method.

   (c) Consider the nonlinear differential equation $y' = 2y - 4y^2$. Determine the maximal step size such that the method (with $\beta = 1/2$) is stable in the neighborhood of $y = 1/2$.

6. Perform a single step with the Forward Euler method with step size $\Delta t = 0.1$ for the system

$$\begin{cases} y_1' = -4y_1 - 2y_2 + e^t, \\ y_2' = 3y_1 + y_2, \qquad\qquad t > 0, \\ y_1(0) = \quad 0, \\ y_2(0) = -1. \end{cases}$$

7. Perform a single step with the Forward Euler method for the initial-value problem

$$\begin{cases} y'' - 2y' + y = te^t - t, \quad t > 0, \\ y(0) = 0, \\ y'(0) = 0, \end{cases}$$

   using step size $\Delta t = 0.1$. Determine the error with respect to the solution

$$y(t) = \frac{1}{6}t^3 e^t - te^t + 2e^t - t - 2.$$

8. Consider the equation of motion of the mathematical pendulum:

$$\begin{cases} \phi'' + \frac{g}{L}\phi = 0, \quad t > 0, \\ \phi(0) = 0, \\ \phi'(0) = 0. \end{cases}$$

   Write this as a first-order system. Is this system stable?

9. Consider the system

$$\begin{cases} y_1' = 1195y_1 - 1995y_2, \\ y_2' = 1197y_1 - 1997y_2, \quad t > 0, \\ y_1(0) = \quad 2, \\ y_2(0) = -2. \end{cases}$$

   The solution is given by

$$\begin{cases} y_1(t) = 10e^{-2t} - 8e^{-800t}, \\ y_2(t) = 6e^{-2t} - 8e^{-800t}. \end{cases}$$

(a) Perform a single step with the Forward and Backward Euler method with $\Delta t = 0.1$ and compare the results with the exact answer.

(b) Determine for which step sizes the Forward Euler method is stable.

(c) Perform a single step with the Forward and Backward Euler method with $\Delta t = 0.0001$ and compare the results with the exact answer.

What is your conclusion?

10. Consider the initial-value problem

$$\begin{cases} y' = y - t^2 + 1, & t > 0, \\ y(0) = \frac{1}{2}. \end{cases}$$

Approximate $y(0.1) = 0.6574145$ with the Forward Euler method, using $\Delta t = 0.025$, and the RK4 method, using $\Delta t = 0.1$. Which method is to be preferred?

11. The solution to an initial-value problem is approximated with two different methods. Both methods have used step sizes $\Delta t = 0.1, 0.05, 0.025$. The numerical approximations at the point $T = 1$ are tabulated below. Determine the order p of the global truncation error of both methods.

Table 6.6: Solution approximations of an initial-value problem at $T = 1$, computed by two different methods.

Δt	method 1	method 2
0.1	0.752790	0.710791
0.05	0.750686	0.730912
0.025	0.750180	0.740587

Chapter 7

The finite-difference method for boundary-value problems

7.1 Introduction

Many applications can be simulated by solving a boundary-value problem. A one-dimensional boundary-value problem consists of a differential equation on a line segment, where the function and/or its derivatives are given at *both* boundary points.

Stationary heat conduction in a bar
As an example we consider the temperature distribution in a bar with length L and cross-sectional area A (Figure 7.1).

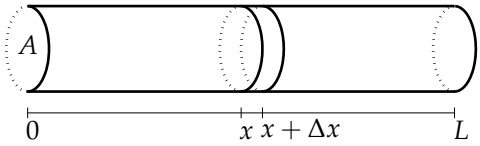


Figure 7.1: Bar with length L and cross-sectional area A.

We denote the temperature in the bar by $T(x)$ (measured in K), and assume that the temperature is known at both ends: $T(0) = T_l$ and $T(L) = T_r$. Further, heat is assumed to be generated inside the bar. We denote this heat production by $Q(x)$ $(J/(m^3 s))$. In general, the temperature profile evolves over time towards a steady state. In this example we are interested in the temperature after a long period of time, i.e. the steady-state temperature. For the derivation of the differential equation the energy conservation law is applied to the control volume between x and $x + \Delta x$ (see Figure 7.1).

There is heat transport by conduction through the faces in x and $x + \Delta x$. According to Fourier's law this heat transport per unit area and per unit time is called the *heat flow density*, and equals

$$q(x) = -\lambda \frac{dT}{dx}(x),$$

where λ $(J/(msK))$ is called the *heat-conduction coefficient*. For the control volume between x and $x + \Delta x$, the energy balance should be satisfied: the total heat outflow at $x + \Delta x$ minus the total heat inflow at x should equal the amount of heat produced in this segment. This can be expressed as

$$-\lambda A \frac{dT}{dx}(x + \Delta x) + \lambda A \frac{dT}{dx}(x) = AQ(x)\Delta x.$$

Division by $A\Delta x$ yields

$$-\lambda \frac{\frac{dT}{dx}(x+\Delta x) - \frac{dT}{dx}(x)}{\Delta x} = Q(x).$$

The corresponding boundary-value problem is obtained by letting $\Delta x \to 0$:

$$\begin{cases} -\lambda \frac{d^2T}{dx^2}(x) = Q(x), \quad 0 < x < L, \\ T(0) = T_l, \\ T(L) = T_r. \end{cases} \tag{7.1}$$

This example will often be used in this chapter to illustrate the finite-difference method.

In some applications the heat flux rather than the temperature is known at one of the end points of the bar. If this is the case at $x = L$, then Fourier's law leads to the boundary condition

$$-\lambda \frac{dT}{dx}(L) = q_L,$$

in which q_L $(J/(m^2 s))$ is known.

7.2 The finite-difference method

The general form of a linear boundary-value problem of the second order in one dimension is

$$-(p(x)y'(x))' + r(x)y'(x) + q(x)y(x) = f(x), \quad 0 < x < L,$$

with boundary conditions

$$a_0 y(0) + b_0 y'(0) = c_0 \quad \text{and} \quad a_L y(L) + b_L y'(L) = c_L.$$

We assume that $p(x) > 0$ and $q(x) \geq 0$ for all $x \in [0, L]$. Note that the boundary-value problem does not have a unique solution when $a_0 = a_L = 0$ and $q(x) = 0$, $0 < x < L$ (if solutions exist at all). The following terms will be used to describe the boundary conditions (we confine ourselves to $x = 0$):

$$\begin{aligned} \text{Dirichlet boundary condition:} &\quad a_0 y(0) = c_0, \quad \text{hence } b_0 = 0, \\ \text{Neumann boundary condition:} &\quad b_0 y'(0) = c_0, \quad \text{hence } a_0 = 0, \\ \text{Robin boundary condition:} &\quad a_0 \neq 0 \quad \text{and } b_0 \neq 0. \end{aligned}$$

In this chapter, we apply the finite-difference method to approximate the solution to a boundary-value problem. The key principle of this approach is to replace all derivatives in the differential equation by difference formulae (Chapter 3) and to neglect truncation errors. In this way, a discrete approximation w for the solution y is obtained.

Example 7.2.1 (Boundary-value problem with homogeneous Dirichlet boundary conditions)

In this example, we describe the finite-difference method for the boundary-value problem

$$\begin{cases} -y''(x) + q(x)y(x) = f(x), \quad 0 < x < 1, \\ y(0) = 0, \\ y(1) = 0. \end{cases} \tag{7.2}$$

First, the interval $[0, 1]$ is divided into $n+1$ equidistant subintervals with length $\Delta x = 1/(n+1)$. The nodes are given by $x_j = j\Delta x$ for $j = 0, \ldots, n+1$ (see Figure 7.2). The approximation of the solution to problem (7.2) will be represented on these nodes.

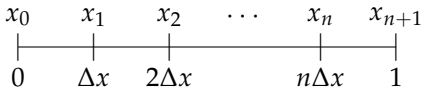


Figure 7.2: Discretization of the domain. The solution is approximated in the nodes $x_0, \ldots, x_{n+1}$.

The numerical approximation of $y_j = y(x_j)$ is denoted by w_j, which is computed using the differential equation in x_j:

$$-y_j'' + q_j y_j = f_j, \quad 1 \leq j \leq n. \tag{7.3}$$

Note that the solutions at $j = 0$ and $j = n + 1$ are known: $y_0 = y_{n+1} = 0$ (boundary conditions).

The finite-difference method approximates the second derivative in equation (7.3) by a central-difference formula (equation (3.8)) to obtain

$$-\frac{w_{j-1} - 2w_j + w_{j+1}}{\Delta x^2} + q_j w_j = f_j, \quad 1 \leq j \leq n. \tag{7.4}$$

The values w_0 and w_{n+1} follow from the boundary conditions: $w_0 = w_{n+1} = 0$.

Difference scheme (7.4) yields n equations for unknowns $w_1, \ldots, w_n$. This relation can be expressed in matrix-vector form as

$$\mathbf{A}\mathbf{w} = \mathbf{f}, \tag{7.5}$$

with $\mathbf{A} = \mathbf{K} + \mathbf{M}$, where $\mathbf{K}$ and $\mathbf{M}$ are given by

$$\mathbf{K} = \frac{1}{\Delta x^2} \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & \varnothing & \\ & & \ddots & & & \\ & & & \ddots & & \\ & \varnothing & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{M} = \begin{pmatrix} q_1 & & \varnothing \\ & \ddots & \\ \varnothing & & q_n \end{pmatrix}. \tag{7.6}$$

The vectors $\mathbf{w}$ and $\mathbf{f}$ are: $\mathbf{w} = (w_1, \ldots, w_n)^\top$ and $\mathbf{f} = (f_1, \ldots, f_n)^\top$.

Example 7.2.2 (Boundary-value problem, nonhomogeneous Dirichlet boundary conditions)

Similar to example 7.2.1, we investigate the finite-difference method for the boundary-value problem

$$\begin{cases} -y''(x) + q(x)y(x) = f(x), & 0 < x < 1, \\ y(0) = \alpha, \\ y(1) = \beta. \end{cases}$$

In this case, the finite-difference formula for $j = 1$ satisfies

$$-\frac{\alpha - 2w_1 + w_2}{\Delta x^2} + q_1 w_1 = f_1,$$

which is equivalent to

$$-\frac{-2w_1 + w_2}{\Delta x^2} + q_1 w_1 = f_1 + \frac{\alpha}{\Delta x^2}.$$

Similarly, the equation for $j = n$ is given by

$$-\frac{w_{n-1} - 2w_n}{\Delta x^2} + q_n w_n = f_n + \frac{\beta}{\Delta x^2}.$$

This leads to the following matrix-vector form:

$$\mathbf{A}\mathbf{w} = \mathbf{f} + \mathbf{r},$$

where $\mathbf{A}$, $\mathbf{w}$ and $\mathbf{f}$ are defined as in example 7.2.1, and $\mathbf{r} = 1/\Delta x^2 \cdot (\alpha, 0, \ldots, 0, \beta)^\top$.

7.3 Some concepts from Linear Algebra

In order to estimate the global accuracy of the numerical approximation, it is necessary to first recall several concepts from Linear Algebra. Herewith, the difference between the numerical approximation w_j and the solution y_j of the boundary-value problem will be analyzed.

Definition 7.3.1 (Scaled Euclidean norm of a vector) *The scaled Euclidean norm of a vector* $\mathbf{w} \in \mathbb{R}^n$ *is defined as*

$$\|\mathbf{w}\| = \sqrt{\frac{1}{n}\sum_{i=1}^{n} w_i^2}.$$

Definition 7.3.2 (Subordinate matrix norm) *The natural, or* subordinate, matrix norm, *related to the vector norm* $\|.\|$ *is defined as*

$$\|\mathbf{A}\| = \max_{\|\mathbf{w}\|=1} \|\mathbf{A}\mathbf{w}\|.$$

Let $\mathbf{x} = \mathbf{y}/\|\mathbf{y}\|$, where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{y} \neq \mathbf{0}$. Then, by definition, $\|\mathbf{x}\| = 1$, and

$$\|\mathbf{A}\| = \max_{\|\mathbf{w}\|=1} \|\mathbf{A}\mathbf{w}\| \geq \|\mathbf{A}\mathbf{x}\| = \|\mathbf{A}\frac{\mathbf{y}}{\|\mathbf{y}\|}\| = \frac{1}{\|\mathbf{y}\|}\|\mathbf{A}\mathbf{y}\|.$$

Hence, for each vector $\mathbf{y} \in \mathbb{R}^n$, we have

$$\|\mathbf{A}\mathbf{y}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{y}\|. \tag{7.7}$$

Condition number
Suppose we are interested in the vector $\mathbf{w}$, satisfying $\mathbf{A}\mathbf{w} = \mathbf{f}$. If the right-hand side $\mathbf{f}$ is perturbed with an error $\Delta\mathbf{f}$, then as a consequence the solution $\mathbf{w}$ will contain an error $\Delta\mathbf{w}$. This means that we solve

$$\mathbf{A}(\mathbf{w} + \Delta\mathbf{w}) = \mathbf{f} + \Delta\mathbf{f}.$$

Using that $\mathbf{A}\mathbf{w} = \mathbf{f}$ and $\mathbf{A}\Delta\mathbf{w} = \Delta\mathbf{f}$, inequality (7.7) yields that $\|\mathbf{f}\| = \|\mathbf{A}\mathbf{w}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{w}\|$, such that

$$\frac{1}{\|\mathbf{w}\|} \leq \frac{\|\mathbf{A}\|}{\|\mathbf{f}\|} \quad \text{and} \quad \|\Delta\mathbf{w}\| = \|\mathbf{A}^{-1}\Delta\mathbf{f}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\Delta\mathbf{f}\|,$$

and the relative error equals

$$\frac{\|\Delta\mathbf{w}\|}{\|\mathbf{w}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \frac{\|\Delta\mathbf{f}\|}{\|\mathbf{f}\|}. \tag{7.8}$$

The quantity $\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$ is called the *condition number* of the matrix $\mathbf{A}$. A large condition number implies that a small relative error in the right-hand side $\mathbf{f}$ may result in a large relative error in the approximation $\mathbf{w}$.

For a symmetric matrix $\mathbf{A}$, the eigenvalues, $\lambda_1, \ldots, \lambda_n$, are real valued, and

$$\|\mathbf{A}\| = |\lambda|_{\max} = \max_{1 \leq j \leq n} |\lambda_j| \quad \text{and} \quad \|\mathbf{A}^{-1}\| = \frac{1}{|\lambda|_{\min}} = \frac{1}{\min\limits_{1 \leq j \leq n} |\lambda_j|}. \tag{7.9}$$

Hence $\kappa(\mathbf{A}) = |\lambda|_{\max}/|\lambda|_{\min}$: it is sufficient to know the largest and smallest absolute eigenvalue (at least approximately) to compute the condition number of a symmetric matrix or an estimate to it.

A useful theorem to estimate these extremal eigenvalues is the Gershgorin circle theorem:

Theorem 7.3.1 (Gershgorin circle theorem) *The eigenvalues of a general $n \times n$ matrix* $\mathbf{A}$ *are located in the complex plane in the union of circles*

$$|z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad \textit{where} \quad z \in \mathbb{C}.$$

Proof
Suppose $\mathbf{Av} = \lambda \mathbf{v}$. By definition, component v_i satisfies

$$\sum_{j=1}^{n} a_{ij} v_j = \lambda v_i, \quad \text{such that} \quad a_{ii} v_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} v_j = \lambda v_i.$$

Therefore,

$$(\lambda - a_{ii}) v_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} v_j.$$

Next, we assume that v_i is the largest component in modulus of $\mathbf{v}$. Using this assumption and the triangle inequality (Theorem 1.6.1) it follows that

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \frac{|v_j|}{|v_i|} \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$

This relation holds for each eigenvalue and each possible eigenvector. Therefore, the complete set of eigenvalues is found in the union of these circles in the complex plane. ⊠

Example 7.3.1 (Gershgorin circle theorem)

In this example, the eigenvalues of the matrix

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

are estimated using the Gershgorin circle theorem. Following the theorem, the eigenvalues should satisfy $|\lambda - 2| \leq 1$. Note that the eigenvalues are $\lambda_1 = 1$, $\lambda_2 = 3$, which indeed satisfy this condition.

7.4 Consistency, stability and convergence

Next, we want to show that the difference between the numerical approximation and the solution tends to zero if the step size Δx tends to zero. To prove this, first some important concepts will be defined.

Definition 7.4.1 (Local truncation error) *The local truncation error* $\boldsymbol{\varepsilon}$ *of the scheme* $\mathbf{Aw} = \mathbf{f}$ *is defined as*

$$\boldsymbol{\varepsilon} = \mathbf{Ay} - \mathbf{Aw} = \mathbf{Ay} - \mathbf{f},$$

where the components $y_j = y(x_j)$ of the solution $\mathbf{y}$ *are the exact values in the nodes x_j: $y_j = y(x_j)$, and* $\mathbf{w}$ *is the corresponding numerical approximation.*

As an example, we investigate the local truncation error for the finite-difference discretization (7.4) in example 7.2.1. Using the error estimate for the central-difference discretization of the second derivative (Section 3.6), row j of $\mathbf{Ay}$ equals

$$\frac{-y_{j-1} + 2y_j - y_{j+1}}{\Delta x^2} + q_j y_j = -y_j'' + \mathcal{O}(\Delta x^2) + q_j y_j,$$

such that

$$\boldsymbol{\varepsilon} = \mathbf{A}\mathbf{y} - \mathbf{f} = -\mathbf{y}'' + \mathcal{O}(\Delta x^2) + \mathbf{q}\mathbf{y}^\top - \mathbf{f}, \tag{7.10}$$

and $\varepsilon_j = \mathcal{O}(\Delta x^2)$, $j = 1, \ldots, n$.

Definition 7.4.2 (Consistency) *A finite-difference scheme is called* consistent *if*

$$\lim_{\Delta x \to 0} \|\boldsymbol{\varepsilon}\| = 0.$$

For the boundary-value problem in example 7.2.1, $\|\boldsymbol{\varepsilon}\| = \mathcal{O}(\Delta x^2)$, hence system (7.5) is consistent.

Definition 7.4.3 (Stability) *A finite-difference scheme is called* stable *if* $\mathbf{A}^{-1}$ *exists and if there exists a constant* C*, independent of* Δx*, such that*

$$\|\mathbf{A}^{-1}\| \leq C, \quad \textit{for} \quad \Delta x \to 0.$$

Example 7.4.1 (Boundary-value problem)

In this example, example 7.2.1 is revisited. Note that the derived matrix $\mathbf{A}$ in this example is symmetric. This means that the eigenvalues are real and that (using equation (7.9))

$$\|\mathbf{A}^{-1}\| = \frac{1}{|\lambda|_{\min}}.$$

We investigate two different cases (recall that $q(x) \geq 0$ in this chapter):

- If $0 < q_{\min} \leq q(x) \leq q_{\max}$ for all $x \in [0, 1]$, then the Gershgorin circle theorem implies that the eigenvalues of $\mathbf{A}$ are located in the union of

  $$\left| z - \left(\frac{2}{\Delta x^2} + q_j \right) \right| \leq \frac{2}{\Delta x^2}, \; j = 1, \ldots, n,$$

  and because the eigenvalues are real, the following union of intervals is found:

  $$q_j \leq z \leq q_j + \frac{4}{\Delta x}, \; j = 1, \ldots, n.$$

  This means that

  $$q_{\min} \leq \lambda_j \leq q_{\max} + \frac{4}{\Delta x^2} \quad \text{for} \quad j = 1, \ldots, n.$$

  From this it follows that $\|\mathbf{A}^{-1}\| \leq 1/q_{\min}$, hence the scheme is stable.

- If $q(x) = 0$ for all $x \in [0, 1]$, then the Gershgorin circle theorem is of no use to estimate the smallest eigenvalue, since in that case the theorem gives no bound for the matrix norm $\|\mathbf{A}^{-1}\|$. Without proof we note that the eigenvalues of $\mathbf{A} = \mathbf{K}$ are

  $$\lambda_j = \frac{2 - 2\cos j\Delta x\pi}{\Delta x^2}, \quad j = 1, \ldots, n. \tag{7.11}$$

  This yields

  $$\lambda_{\min} = \frac{2 - 2\cos \Delta x\pi}{\Delta x^2} \approx \pi^2,$$

  which implies that also for $q \equiv 0$ the scheme is stable.

Definition 7.4.4 (Global truncation error) *The global truncation error of a finite-difference scheme is defined as* $\mathbf{e} = \mathbf{y} - \mathbf{w}$.

Definition 7.4.5 (Convergence) *A scheme is called* convergent *if the global truncation error satisfies*

$$\lim_{\Delta x \to 0} \|\mathbf{e}\| = 0.$$

The next theorem provides a relation between the concepts consistency, stability and convergence.

Theorem 7.4.1 *If a scheme is stable and consistent, then that scheme is convergent.*

Proof:
The global truncation error is related to the local truncation error ($\mathbf{Ae} = \mathbf{A}(\mathbf{y} - \mathbf{w}) = \boldsymbol{\varepsilon}$), and hence $\mathbf{e} = \mathbf{A}^{-1}\boldsymbol{\varepsilon}$. Taking norms and applying inequality (7.7) yields $\|\mathbf{e}\| \leq \|\mathbf{A}^{-1}\| \, \|\boldsymbol{\varepsilon}\|$. From stability and consistency it then follows that $\lim_{\Delta x \to 0} \|\mathbf{e}\| = 0$. ⊠

Note that consistency in itself is insufficient for convergence. Again it is true, that the order of the global truncation error equals the order of the local truncation error.

Example 7.4.2 (Convergence)

In this example, the heat transport in a bar (Section 7.1) is described by the boundary-value problem

$$\begin{cases} -y'' = 25e^{5x}, & 0 < x < 1, \\ y(0) = 0, \\ y(1) = 0. \end{cases} \tag{7.12}$$

Note that this problem is related to system (7.1), where y denotes the temperature in the bar, and the production term $Q(x)$ is given by $25e^{5x}$ (we expect positive temperatures inside the bar). The solution is given by

$$y(x) = -e^{5x} + (-1 + e^5)x + 1.$$

In Figure 7.3(a) the solution is plotted, together with the numerical approximations (computed using the finite-difference approach of Section 7.2) for different values of Δx. The numerical approximation converges rapidly to the solution. In a practical situation, the accuracy obtained with step size $\Delta x = 1/16$ will probably be sufficient.

Next, heat dissipation is included in the boundary-value problem. This leads to

$$\begin{cases} -y'' + 9y = 25e^{5x}, & 0 < x < 1, \\ y(0) = 0, \\ y(1) = 0, \end{cases} \tag{7.13}$$

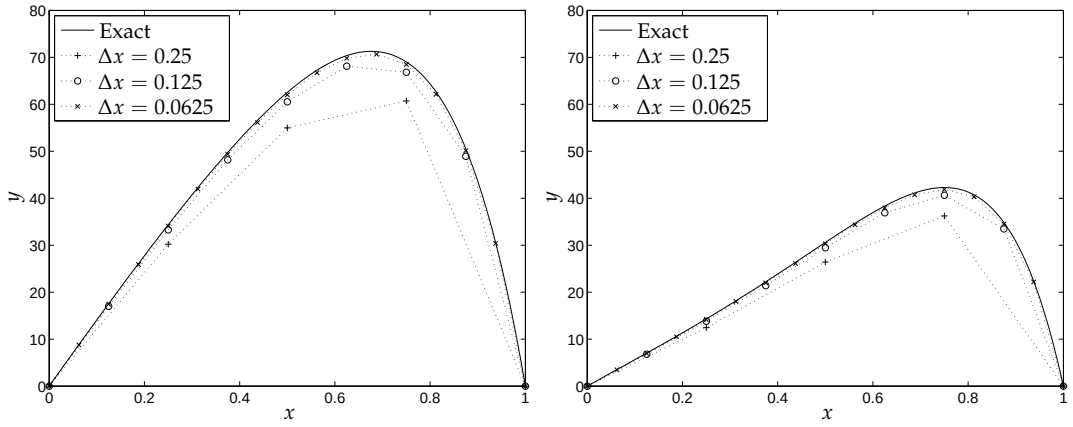
in which the term $9y$ describes heat dissipation to the environment, proportional to the temperature. The solution and the numerical approximation are plotted in Figure 7.3(b). Note that the maximal temperature is definitely lower than without dissipation (Figure 7.3(a)). For the rest, there is little difference between the convergence behavior of both problems.

7.5 Conditioning of the discretization matrix *

In this section, the boundary-value problem as stated in example 7.2.1 will be revisited to study the condition number of the discretization matrix. From example 7.4.1 it follows that the minimum and maximum eigenvalue when $q(x) = 0,\ 0 \leq x \leq 1$ are approximated by

$$\lambda_{\min} \approx \pi^2, \quad \lambda_{\max} \approx \frac{4}{\Delta x^2}, \quad \text{and} \quad \kappa(\mathbf{A}) \approx \frac{4}{(\pi \Delta x)^2}.$$

If Δx tends to zero, then the condition number of $\mathbf{A}$ is unbounded which is not desirable, since this would mean that perturbations in the right-hand side $\mathbf{f}$ could lead to arbitrarily large errors



(a) Without dissipation (b) With dissipation

Figure 7.3: Exact solution and numerical approximation of the temperature distribution in a bar, without (problem (7.12)) or with (problem (7.13)) dissipation.

in the approximation. It turns out that the derivation of the relative error in equation (7.8) was too pessimistic. A more realistic estimate is found by writing

$$\|\Delta \mathbf{w}\| = \|\mathbf{A}^{-1}\Delta \mathbf{f}\| \leq \frac{1}{|\lambda|_{\min}}\|\Delta \mathbf{f}\|.$$

Hence the relative error satisfies

$$\frac{\|\Delta \mathbf{w}\|}{\|\mathbf{w}\|} \leq \frac{1}{\lambda_{\min}}\frac{\|\mathbf{f}\|}{\|\mathbf{w}\|}\cdot\frac{\|\Delta \mathbf{f}\|}{\|\mathbf{f}\|} = \kappa_{\text{eff}}(\mathbf{A})\frac{\|\Delta \mathbf{f}\|}{\|\mathbf{f}\|}.$$

The role of $\kappa(\mathbf{A})$ in this expression is taken over by the *effective condition number*

$$\kappa_{\text{eff}}(\mathbf{A}) = \frac{1}{\lambda_{\min}}\frac{\|\mathbf{f}\|}{\|\mathbf{w}\|}.$$

We know that $\lambda_{\min} \approx \pi^2$ and in many applications $\|\mathbf{f}\|/\|\mathbf{w}\|$ is bounded as Δx tends to zero, hence the effective condition number is, in fact, bounded. Although this estimate is more accurate than the original condition number, it is more difficult to compute, since (an estimate for) $\mathbf{w}$ is required.

7.6 Neumann boundary condition

In the previous sections, boundary-value problems with two Dirichlet boundary conditions were considered. In this section, we provide a method to deal with Neumann boundary conditions.

Example 7.6.1 (Boundary-value problem with Neumann boundary condition)

Here, the boundary-value problem of example 7.2.1 is slightly modified to include a Neumann boundary condition on the right boundary:

$$\begin{cases} -y'' + q(x)y = f(x), & 0 < x < 1, \\ y(0) = 0, \\ y'(1) = 0. \end{cases} \tag{7.14}$$

In this case, w_{n+1} is no longer known, as it was for a Dirichlet boundary condition. Therefore, the discretization should also include an equation for $j = n+1$, such that a system of $n+1$ equations with $n+1$ unknowns, $w_1, \ldots, w_{n+1}$ has to be solved. For $j = n+1$ we have

$$\frac{-w_n + 2w_{n+1} - w_{n+2}}{\Delta x^2} + q_{n+1}w_{n+1} = f_{n+1}. \tag{7.15}$$

Note that w_{n+2} does not exist, since $x_{n+1} = 1$ is the right boundary of the domain. In order to approximate w_{n+1} properly, a virtual point $x_{n+2} = (n+2)\Delta x = 1 + \Delta x$ is introduced (see Figure 7.4).

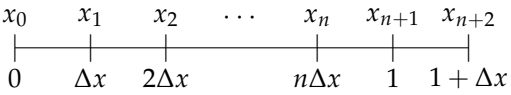


Figure 7.4: Discretization of the domain, with a virtual node x_{n+2}. The solution is approximated in the nodes $x_0, \ldots, x_{n+1}$.

The value of w_{n+2} follows from discretization of the Neumann boundary condition using central differences (equation (3.5)):

$$\frac{w_{n+2} - w_n}{2\Delta x} = 0.$$

This implies $w_{n+2} = w_n$. We substitute this into (7.15) and divide by two (to preserve symmetry of the matrix) to obtain

$$\frac{-w_n + w_{n+1}}{\Delta x^2} + \frac{1}{2}q_{n+1}w_{n+1} = \frac{1}{2}f_{n+1}. \tag{7.16}$$

This means that we should solve $\mathbf{Aw} = \mathbf{f}$, where the symmetric matrix $\mathbf{A}$ can be written as $\mathbf{A} = \mathbf{K} + \mathbf{M}$, with

$$\mathbf{K} = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \emptyset \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & \emptyset & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{M} = \begin{pmatrix} q_1 & & & \emptyset \\ & \ddots & & \\ & & \ddots & \\ & \emptyset & & q_n & \\ & & & & \frac{q_{n+1}}{2} \end{pmatrix},$$

$\mathbf{w} = (w_1, \ldots, w_{n+1})^\top$ and $\mathbf{f} = (f_1, \ldots, f_n, f_{n+1}/2)^\top$.

Next, the error behavior for this system of equations is considered. Recall from Section 7.4 that the discretization of the internal domain and Dirichlet boundary yields a local truncation error of order $\mathcal{O}(\Delta x^2)$ (equation (7.10)). For the Neumann-boundary discretization in equation (7.16), the local truncation error can be computed using the Taylor expansion of y_n and y_{n+2} about x_{n+1}:

$$y_n = y_{n+1} - \Delta x y'_{n+1} + \frac{\Delta x^2}{2}y''_{n+1} - \frac{\Delta x^3}{3!}y'''_{n+1} + \mathcal{O}(\Delta x^4),$$
$$y_{n+2} = y_{n+1} + \Delta x y'_{n+1} + \frac{\Delta x^2}{2}y''_{n+1} + \frac{\Delta x^3}{3!}y'''_{n+1} + \mathcal{O}(\Delta x^4).$$

Using that $y'_{n+1} = 0$, this means that $y_{n+2} = y_n + \mathcal{O}(\Delta x^3)$, such that the replacement of y_{n+2} by y_n in the discretization of the Neumann boundary yields a local truncation error of order $\mathcal{O}(\Delta x)$. This means that the local truncation error can be written as

$$\mathbf{Ay} - \mathbf{f} = \boldsymbol{\varepsilon} = \Delta x^2 \mathbf{u} + \Delta x \mathbf{v},$$

where $\Delta x^2\mathbf{u}$ results from the inner local truncation error (as was already found in equation (7.10)), and the component $\Delta x\mathbf{v}$ belongs to the local truncation error due to the Neumann boundary: $\mathbf{v} = (0,\ldots,0,v_{n+1})^\top$, with $v_{n+1} = \mathcal{O}(1)$. Note that the order of the local truncation error is different from the order of a boundary-value problem with pure Dirichlet boundary conditions.

It turns out that the global truncation error still remains of order $\mathcal{O}(\Delta x^2)$. This claim will be shown for $q(x) = 0,\ 0 \le x \le 1$, which results in a stable system. The global truncation error is split as follows:

$$\mathbf{e} = \mathbf{A}^{-1}\boldsymbol{\varepsilon} = \mathbf{A}^{-1}\left(\Delta x^2\mathbf{u} + \Delta x\mathbf{v}\right) = \mathbf{e}^{(1)} + \mathbf{e}^{(2)}.$$

By construction, $\|\mathbf{e}^{(1)}\| = \mathcal{O}(\Delta x^2)$. For $\mathbf{e}^{(2)}$, it holds that $e_j^{(2)} = \Delta x^3 v_{n+1} j$, $j = 1,\ldots,n+1$. This can be shown by verifying that $\mathbf{A}\mathbf{e}^{(2)} = \Delta x\mathbf{v}$. The jth component of $\mathbf{A}\mathbf{e}^{(2)}$ equals

$$(\mathbf{A}\mathbf{e}^{(2)})_j = \frac{-e_{j-1}^{(2)} + 2e_j^{(2)} - e_{j+1}^{(2)}}{\Delta x^2} = \frac{\Delta x^3 v_{n+1}(-(j-1) + 2j - (j+1))}{\Delta x^2} = 0,$$

if $j = 2,\ldots,n$, and similarly for $j = 1$, $(\mathbf{A}\mathbf{e}^{(2)})_1 = 0$. For $j = n+1$, we have

$$(\mathbf{A}\mathbf{e}^{(2)})_{n+1} = \frac{-e_n^{(2)} + e_{n+1}^{(2)}}{\Delta x^2} = \frac{\Delta x^3 v_{n+1}(-n + (n+1))}{\Delta x^2} = \Delta x v_{n+1},$$

hence indeed, $\mathbf{A}\mathbf{e}^{(2)} = \Delta x\mathbf{v}$.

By construction, $\Delta x j \le 1$ (size of domain), such that $e_j^{(2)} \le \Delta x^2 v_{n+1}$. Therefore, $\|\mathbf{e}^{(2)}\| = \mathcal{O}(\Delta x^2)$, and the global truncation error satisfies $\|\mathbf{e}\| = \mathcal{O}(\Delta x^2)$.

For $q \neq 0$, the same orders of the local and global truncation error are found, but the details are beyond the scope of this book.

7.7 The general problem *

In this section, a general boundary-value problem on $[0,1]$ is considered, given by

$$\begin{cases} -(p(x)y'(x))' + r(x)y'(x) + q(x)y(x) = f(x), & 0 < x < 1, \\ y(0) = \alpha, \\ p(1)y'(1) = \beta, \end{cases}$$

where we assume $p(x) > 0$ on $[0,1]$.

The discretization in an interior node x_j reads

$$\frac{-p_{j+\frac{1}{2}}(w_{j+1} - w_j) + p_{j-\frac{1}{2}}(w_j - w_{j-1})}{\Delta x^2} + r_j\frac{w_{j+1} - w_{j-1}}{2\Delta x} + q_j w_j = f_j, \quad j = 1,\ldots,n+1. \tag{7.17}$$

Note that if $r_j \neq 0$, then the discretization matrix is not symmetric.

The boundary condition in $x = 0$ yields $w_0 = \alpha$ which may be substituted directly into (7.17) for $j = 1$.

For $j = n+1$, the discretization is

$$\frac{-p_{n+\frac{3}{2}}(w_{n+2} - w_{n+1}) + p_{n+\frac{1}{2}}(w_{n+1} - w_n)}{\Delta x^2} + r_{n+1}\frac{w_{n+2} - w_n}{2\Delta x} + q_{n+1}w_{n+1} = f_{n+1}. \tag{7.18}$$

As in Section 7.6, a virtual grid point, x_{n+2}, is added to the domain. In order to remove the term $-p_{n+3/2}(w_{n+2} - w_{n+1})$ in equation (7.18), we discretize boundary condition $p(1)y'(1) = \beta$ by averaging the approximations for

$$p_{n+\frac{1}{2}}y'\left(1 - \frac{\Delta x}{2}\right) \quad \text{and} \quad p_{n+\frac{3}{2}}y'\left(1 + \frac{\Delta x}{2}\right).$$

This means that

$$\frac{1}{2}\left(p_{n+\frac{1}{2}}\frac{w_{n+1} - w_n}{\Delta x} + p_{n+\frac{3}{2}}\frac{w_{n+2} - w_{n+1}}{\Delta x}\right) = \beta,$$

such that

$$p_{n+\frac{3}{2}}(w_{n+2} - w_{n+1}) = -p_{n+\frac{1}{2}}(w_{n+1} - w_n) + 2\Delta x\beta.$$

Replacing this term in equation (7.18) and using that $(w_{n+2} - w_n)/(2\Delta x) = \beta/p(1)$ yields

$$\frac{2p_{n+\frac{1}{2}}(w_{n+1} - w_n)}{\Delta x^2} + \frac{r_{n+1}\beta}{p(1)} + q_{n+1}w_{n+1} = f_{n+1} + \frac{2\beta}{\Delta x}.$$

7.8 Convection-diffusion equation

In the previous section the discretization for a general boundary-value problem was derived by applying a central-difference scheme. However, if a second-order boundary-value problem also contains a first-order derivative, this may lead to (physically) unacceptable approximations. This will be illustrated both analytically and numerically for a specific type of boundary-value problems for the convection-diffusion equations.

Example 7.8.1 (Convection-diffusion equation)

An analytical investigation of boundary-value problems for convection-diffusion equations will be established using

$$\begin{cases} -y''(x) + vy'(x) = 0, & 0 < x < 1, \\ y(0) = 0, \\ y(1) = 1, \end{cases} \tag{7.19}$$

with $v \in \mathbb{R}$. This equation can be interpreted as a heat-transfer problem. Heat transport is caused by conduction (diffusion, the term $-y''$), and by a flowing medium (convection, the term vy').

It can be verified that the solution to this boundary-value problem is given by

$$y(x) = \frac{e^{vx} - 1}{e^v - 1}.$$

The solution increases strictly over the region $[0, 1]$ and therefore numerical approximations that contain any oscillations will be rejected. However, a careless discretization of boundary-value problem (7.19) can result in a numerical approximation that contains (spurious) oscillations.

Central differences

Using the same subintervals and nodes as in example 7.2.1, the central-difference discretization at interior nodes x_j is given by

$$-\frac{w_{j-1} - 2w_j + w_{j+1}}{\Delta x^2} + v\frac{w_{j+1} - w_{j-1}}{2\Delta x} = 0, \; j = 1, \ldots, n. \tag{7.20}$$

Together with the boundary conditions, this leads to a system of equations $\mathbf{Aw} = \mathbf{f}$, in which

$$\mathbf{A} = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 + \frac{v\Delta x}{2} & & & \\ -1 - \frac{v\Delta x}{2} & 2 & -1 + \frac{v\Delta x}{2} & & \emptyset \\ & & \ddots & & \\ & \emptyset & & -1 - \frac{v\Delta x}{2} & 2 & -1 + \frac{v\Delta x}{2} \\ & & & & -1 - \frac{v\Delta x}{2} & 2 \end{pmatrix},$$

$\mathbf{w} = (w_1, \ldots, w_n)^\top$, and $\mathbf{f} = 1/\Delta x^2 \cdot (0, \ldots, 0, 1 - v\Delta x/2)^\top$.

Equation (7.20) is a linear difference equation, and it is known that its solution is of the form $w_j = r^j$, for some unknown quantity r. Substitution of $w_j = r^j$ into equation (7.20) yields after multiplication by Δx^2:

$$-r^{j-1} + 2r^j - r^{j+1} + \frac{v\Delta x}{2}(r^{j+1} - r^{j-1}) = r^{j-1}\left(-(r^2 - 2r + 1) + \frac{v\Delta x}{2}(r^2 - 1)\right) = 0,$$

which is satisfied if

$$r = 0 \quad \text{or} \quad (r-1)^2 = \frac{v\Delta x}{2}(r+1)(r-1). \tag{7.21}$$

The solution $r = 0$ is the trivial solution. The roots of interest are given by

$$r_1 = 1, \qquad r_2 = \frac{1 + \frac{v\Delta x}{2}}{1 - \frac{v\Delta x}{2}}, \qquad v\Delta x \neq 2. \tag{7.22}$$

The solution to difference equation (7.20) can be written as $w_j = ar_1^j + br_2^j$. The constants a and b follow from the boundary conditions $w_0 = 0$ and $w_{n+1} = 1$, such that we arrive at

$$w_j = \frac{1 - r_2^j}{1 - r_2^{n+1}}. \tag{7.23}$$

Note that w_j becomes oscillatory if $r_2 < 0$, since r_2^j in that case is positive if j is even, and negative if j is odd. This happens if $|v|\Delta x > 2$. Hence, in order to obtain monotone solutions, we require $|v|\Delta x < 2$. In practice, the velocity is a given parameter, such that stability may require impractically small subintervals of size $\Delta x < 2/|v|$.

Upwind differences

A common remedy is to use *upwind discretizations*. In this discretization, vy_j' is approximated using one-sided differences:

$$vy_j' \approx \begin{cases} v\frac{w_j - w_{j-1}}{\Delta x}, & \text{if} \quad v \geq 0 \quad \text{(upwind difference)}, \\ v\frac{w_{j+1} - w_j}{\Delta x,} & \text{if} \quad v < 0 \quad \text{(downwind difference)}. \end{cases}$$

For $v \geq 0$, the corresponding discretization equals

$$-\frac{w_{j-1} - 2w_j + w_{j+1}}{\Delta x^2} + v\frac{w_j - w_{j-1}}{\Delta x} = 0. \tag{7.24}$$

Application of the same trial solution $w_j = r^j$, division by r^{j-1}, multiplication by Δx^2 and some rearrangement results in the non-trivial roots

$$v\Delta x(r-1) = (r-1)^2,$$

with solutions $r_1 = 1$, $r_2 = 1 + v\Delta x > 0$. These values reassure that the numerical approximation (7.23) is non-oscillatory, and therefore admissible for all possible choices of $\Delta x > 0$.

A similar conclusion can be drawn for $v < 0$.

An example of the results using either central or upwind differences is depicted in Figure 7.5. Here, we take $v = 30$ and $\Delta x = 0.1$, such that $|v|\Delta x > 2$. Note that indeed, the central-difference approximation is oscillatory, whereas the upwind-difference approximation is monotone.

It should be noticed that despite the desired physically correct behavior of the numerical approximation, the global truncation error of the upwind-difference discretization is of order $\mathcal{O}(\Delta x)$ only. Thus, the price to pay for non-oscillatory physically meaningful approximations with reasonable values for Δx is a lower order of accuracy. Without further details, we state that in practice more advanced methods are used to solve convection-diffusion equations.

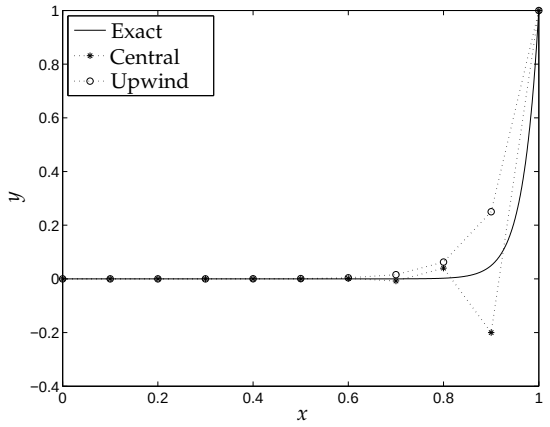


Figure 7.5: Approximation of the solution to convection-diffusion problem (7.19) for $v = 30$, $\Delta x = 0.1$.

Example 7.8.2 (Convection-diffusion equation)

In this example, the numerical results for the boundary-value problem

$$\begin{cases} -y'' + vy' = 1, \quad 0 < x < 1, \\ y(0) = 0, \\ y(1) = 0. \end{cases} \tag{7.25}$$

will be computed, using both a central and an upwind discretization. The solution to this convection-diffusion equation is given by

$$y(x) = \frac{1}{v}\left(x - \frac{1 - e^{vx}}{1 - e^{v}}\right).$$

Although the solution to this convection-diffusion equation is not monotone as was the case in example 7.8.1, a similar behavior is found.

Approximations of the solution are determined with stepsize $\Delta x = 0.1$ for various values of the velocity v (see Figure 7.6).

For $v = 10$, we have $v\Delta x < 2$. Therefore, the central-difference approximation does not introduce any spurious oscillations. Since the global truncation error is of order $\mathcal{O}(\Delta x^2)$, this results into a more accurate approximation than upwind discretization (which is of order $\mathcal{O}(\Delta x)$). If $v = 20$, then $v\Delta x = 2$, such that the discretization at node x_n is given by

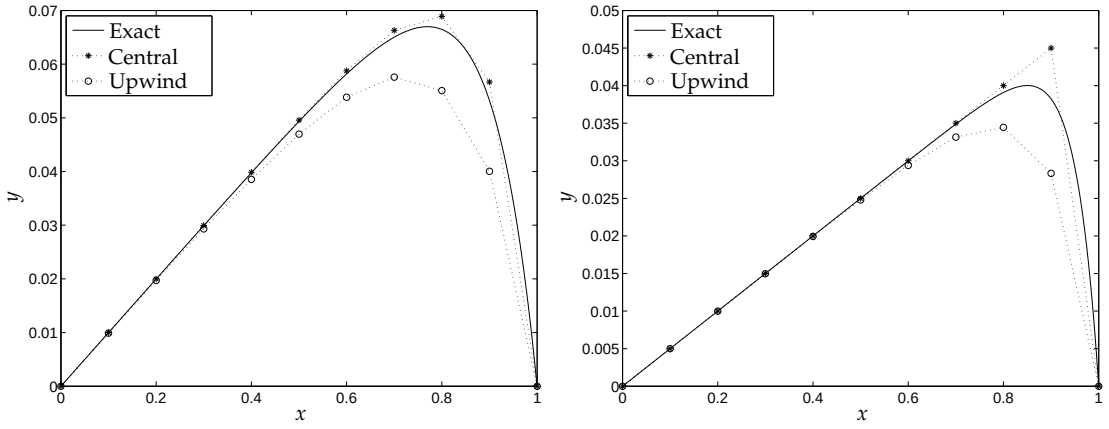
$$-\frac{w_{n-1} - 2w_n + w_{n+1}}{\Delta x^2} + v\frac{w_{n+1} - w_{n-1}}{2\Delta x} = -\frac{2}{\Delta x^2}w_{n-1} + \frac{2}{\Delta x^2}w_n = 1.$$

Since the term with w_{n+1} has been cancelled, the boundary condition in $x = 1$ does not have any effect on the rest of the central-difference approximation. However, for this example the result is still quite accurate. For $v = 100$ the central-discretization scheme produces oscillations in the approximation, and the errors are large. A more accurate approximation for $v = 100$ is found using the upwind-discretization scheme.

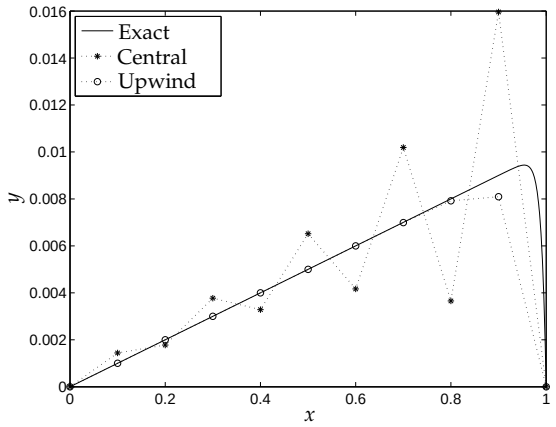
7.9 Nonlinear boundary-value problems

Consider a nonlinear boundary-value problem of the following form:

$$\begin{cases} -(py')' + g(y', y, x) = 0, \quad 0 < x < 1, \\ y(0) = 0, \\ y(1) = 0. \end{cases} \tag{7.26}$$



(a) $v = 10$ (b) $v = 20$



(c) $v = 100$

Figure 7.6: Approximation of the solution to convection-diffusion problem (7.25) for different values of v, $\Delta x = 0.1$.

The general approach to solve (7.26) uses an iterative process of Chapter 4 in which only linear equations have to be solved. The derived linear boundary-value problems are solved using the finite-difference schemes as explained in earlier sections. The iterative method generates a series of iterates $y^{(0)}, y^{(1)}, \ldots$ in such a way that $y^{(m)} \to y,\ m \to \infty$ where y is the solution to (7.26).

7.9.1 Picard iteration

The iterative method of Picard approximates the solution to boundary-value problem (7.26) by determining $y^{(m)}$ from

$$\begin{cases} -(py^{(m)\prime})' = -g(y^{(m-1)\prime}, y^{(m-1)}, x), \\ y^{(m)}(0) = 0, \\ y^{(m)}(1) = 0. \end{cases}$$

Here, the finite-difference approach is used to approximate the derivative of the $m-1$th and the mth iterate.

7.9.2 Newton-Raphson method

The Newton-Raphson method is based on linearization of g about $(y^{(m-1)\prime}, y^{(m-1)})$:

$$g(y', y, x) \approx g(y^{(m-1)\prime}, y^{(m-1)}, x) + (y' - y^{(m-1)\prime})r + (y - y^{(m-1)})q$$

in which

$$r = \frac{\partial g}{\partial y'}(y^{(m-1)\prime}, y^{(m-1)}, x) \quad \text{and} \quad q = \frac{\partial g}{\partial y}(y^{(m-1)\prime}, y^{(m-1)}, x).$$

The corresponding iterative method determines $y^{(m)}$ from

$$\begin{cases} -(py^{(m)\prime})' + ry^{(m)\prime} + qy^{(m)} = y^{(m-1)\prime}r + y^{(m-1)}q - g(y^{(m-1)\prime}, y^{(m-1)}, x), \\ y(0) = 0, \\ y(1) = 0. \end{cases}$$

7.10 Summary

In this chapter, the following subjects have been discussed:

- Boundary-value problems;
- Dirichlet, Neumann, Robin boundary condition;
- Finite-difference schemes;
- Norms, condition numbers, Gershgorin circle theorem;
- Local truncation error, consistency;
- Stability;
- Global truncation error, convergence;
- Implementation of Neumann boundary condition;
- Convection-diffusion problem;
- Upwind discretization;
- Nonlinear boundary-value problem.

7.11 Exercises

1. Consider matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ given by

$$\mathbf{A}_1 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \text{ and } \mathbf{A}_2 = \begin{pmatrix} 100 & 99 \\ 99 & 100 \end{pmatrix}.$$

   Determine the condition number of both matrices. Determine the solution to $\mathbf{A}_2\mathbf{w} = \mathbf{f}$ in which $\mathbf{f} = (199, 199)^\top$. Take $\Delta\mathbf{f} = (1, 0)^\top$. Estimate $\|\Delta\mathbf{w}\|$ using the condition number. Determine $\Delta\mathbf{w}$ and compare $\|\Delta\mathbf{w}\|$ with your estimate.

2. Consider the boundary-value problem

$$\begin{cases} y''(x) = 2(y(x) - x), \quad 0 < x < 1, \\ y(0) = 0, \\ y(1) = 0. \end{cases}$$

   (a) By discretization we obtain a system of linear equations $\mathbf{A}\mathbf{w} = \mathbf{f}$. Determine $\mathbf{A}$ and $\mathbf{f}$ for a general value of Δx.

   (b) Estimate the largest and smallest eigenvalue of $\mathbf{A}$.

   (c) Estimate $\|\Delta\mathbf{w}\|/\|\mathbf{w}\|$ if $\|\Delta\mathbf{f}\|/\|\mathbf{f}\| \leq 10^{-4}$.

3. Consider the boundary-value problem

$$\begin{cases} -y''(x) = \sin x, & 0 < x < \pi, \\ y(0) = 0, \\ y(\pi) = 0. \end{cases}$$

Determine the solution $\mathbf{y}$, and the numerical approximation $\mathbf{w}$, with $n = 2$. Compute $y_j - w_j$, $j = 1, 2$.

4. Consider the boundary-value problem

$$\begin{cases} -y'' + y = 0, & 0 < x < 1, \\ y(0) = 1, \\ y'(1) = 0. \end{cases}$$

Discretize this problem using the nodes $x_0 = 0$, $x_1 = 2/7$, $x_2 = 4/7$, $x_3 = 6/7$ and $x_4 = 8/7$. Determine the discretization matrix $\mathbf{A}$. Is $\mathbf{A}$ symmetric or nonsymmetric? Are the eigenvalues positive?

Chapter 8

The instationary heat equation *

8.1 Introduction

The time-dependent temperature distribution in a bar can be described by a parabolic partial differential equation. To solve that kind of initial- and boundary-value problems methods from Chapters 6 and 7 will be combined.

As an example, we consider the temperature distribution in a bar (Figure 8.1).

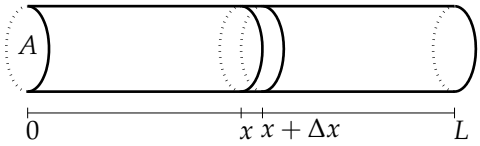


Figure 8.1: Bar with length L and cross-sectional area A.

We denote the temperature by $T(x,t)$ (measured in K). We assume that the temperature is known at both ends $T(0,t) = T_l$ and $T(L,t) = T_r$, and that the initial temperature at $t = 0$ is given by $T(x,t) = T_0(x)$. The heat production in the bar is denoted by $Q(x,t)$ $(J/(m^3 s))$. To derive the heat equation we use the energy conservation law applied to the control volume between x and $x + \Delta x$ and between the points in time t and $t + \Delta t$. According to Fourier's law the heat flow density is given by

$$q(x,t) = -\lambda \frac{\partial T}{\partial x}(x,t),$$

where $\lambda(J/(msK))$ is the heat-conduction coefficient.

The energy balance on the control volume is given by

$$\rho c T(x,t+\Delta t) A \Delta x = \rho c T(x,t) A \Delta x - \lambda \frac{\partial T}{\partial x}(x,t) A \Delta t + \lambda \frac{\partial T}{\partial x}(x+\Delta x,t) A \Delta t + Q(x,t) A \Delta x \Delta t,$$

in which $\rho(kg/m^3)$ is the mass density, and $c(J/(kgK))$ is the specific heat. After dividing by $A\Delta x \Delta t$ and rearranging the terms we obtain

$$\rho c \frac{T(x,t+\Delta t) - T(x,t)}{\Delta t} = \lambda \frac{\frac{\partial T}{\partial x}(x+\Delta x,t) - \frac{\partial T}{\partial x}(x,t)}{\Delta x} + Q.$$

Taking the limits $\Delta x \to 0$ and $\Delta t \to 0$ yields the following initial-boundary-value problem:

$$\begin{cases} \rho c \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2} + Q, & 0 < x < L, \quad 0 < t, \\ T(0,t) = T_l(t), \quad T(L,t) = T_r(t), & 0 \le t, \\ T(x,0) = T_0(x), & 0 \le x \le L. \end{cases}$$

8.2 Semi-discretization

In this chapter, the space and time discretizations will be exhibited for a simple heat problem. There is a clear difference between an explicit (Forward Euler) and an implicit (Backward Euler) method of time integration.

Example 8.2.1 (Heat-conduction problem)

We consider the following heat-conduction problem:

$$\begin{cases} \frac{\partial y}{\partial t} = \frac{\partial^2 y}{\partial x^2}, & 0 < x < 1, \quad 0 < t \leq T, \\ y(0,t) = y_L(t), \qquad y(1,t) = y_R(t), & 0 \leq t \leq T, \\ y(x,0) = y_0(x), & 0 \leq x \leq 1. \end{cases} \tag{8.1}$$

We discretize (8.1) in the x-direction by means of the finite-difference method (Chapter 7). The interval $[0,1]$ is divided into $n+1$ equal parts with length Δx, where the nodes are given by $x_i = i\Delta x$, $i = 0, \ldots, n+1$. We denote the numerical approximation of $y(x_i,t)$ by $u_i(t)$. The vector $\mathbf{u}(t)$ is given by $\mathbf{u}(t) = (u_1(t), \ldots, u_n(t))^\top$. The values at the boundary points are omitted, because they are known from the boundary conditions. Using the techniques from Chapter 7 we obtain the following system of first-order *ordinary* differential equations

$$\begin{cases} \frac{d\mathbf{u}}{dt} = \mathbf{K}\mathbf{u} + \mathbf{r}, & 0 < t \leq T, \\ \mathbf{u}(0) = \mathbf{y}_0. \end{cases} \tag{8.2}$$

The matrix $\mathbf{K}$ and vector $\mathbf{r}$ are given by

$$\mathbf{K} = \frac{1}{\Delta x^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \emptyset \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & \emptyset & & 1 & -2 \end{pmatrix} \quad \text{and} \quad \mathbf{r} = \frac{1}{\Delta x^2}(y_L(t), 0, \ldots, 0, y_R(t))^\top.$$

This is called *semi-discretization* (or method of lines), because discretization has been applied to the spatial x-direction but not to the temporal t-direction. Note that the eigenvalues λ_j of $\mathbf{K}$ satisfy (cf. the Gershgorin circle theorem) $\lambda_j \leq 0$ for all $j = 1, \ldots, n-1$. This implies stability of the semi-discrete system. In the boundary-value problems of Chapter 7, $-y''$ was considered. Therefore, the matrix $\mathbf{K}$ differs in sign from the matrix in equation (7.6).

8.3 Time integration

System (8.2) may be integrated in time by one of the methods from Chapter 6. The solution is approximated at times $t_j = j\Delta t$, using m time steps of length $\Delta t = T/m$. The numerical approximation of the temperature at position $i\Delta x$ and time $j\Delta t$ is denoted by w_i^j (which approximates $u_i(t_j)$, and hence $y(x_i,t_j)$). Due to initial and boundary conditions, it is known that $w_i^0 = y_0(x_i)$, $w_0^j = y_L(t_j)$ and $w_{n+1}^j = y_R(t_j)$.

8.3.1 Forward Euler method

First, the Forward Euler method will be considered. From Chapter 6 (equation (6.51a)) it follows that the approximation of the solution to problem (8.2) at t_{j+1} equals

$$\mathbf{w}^{j+1} = \mathbf{w}^j + \Delta t(\mathbf{K}\mathbf{w}^j + \mathbf{r}^j). \tag{8.3}$$

Note that w_i^{j+1} depends on the approximations in (x_{i-1}, t_j), (x_i, t_j) and (x_{i+1}, t_j) (as follows from the definition of $\mathbf{K}$). This idea is depicted in Figure 8.2(a).

The local truncation error of this method can be computed using definition (6.68) in Chapter 6:

$$\boldsymbol{\tau}^{j+1} = \frac{\mathbf{y}^{j+1} - \mathbf{z}^{j+1}}{\Delta t},$$

where $\mathbf{z}^{j+1}$ is computed by applying the Forward Euler method to the solution at time t_j (given by $\mathbf{y}^j$), hence $\mathbf{z}^{j+1} = \mathbf{y}^j + \Delta t(\mathbf{K}\mathbf{y}^j + \mathbf{r}^j)$. This yields (using the Dirichlet boundary conditions)

$$\boldsymbol{\tau}^{j+1} = \frac{\mathbf{y}^{j+1} - (\mathbf{y}^j + \Delta t(\mathbf{K}\mathbf{y}^j + \mathbf{r}^j))}{\Delta t} = \frac{\mathbf{y}^{j+1} - \mathbf{y}^j}{\Delta t} - (\mathbf{K}\mathbf{y}^j + \mathbf{r}^j) = \frac{\partial \mathbf{y}^j}{\partial t} + \boldsymbol{\varepsilon}^j - \frac{\partial^2 \mathbf{y}^j}{\partial x^2} + \boldsymbol{\mu}^j,$$

in which

$$\varepsilon_i^j = \frac{\Delta t}{2}\frac{\partial^2 y}{\partial t^2}(x_i, \zeta_j), \quad \zeta_j \in (t_j, t_{j+1}), \quad \text{and} \quad \mu_i^j = -\frac{\Delta x^2}{12}\frac{\partial^4 y}{\partial x^4}(\xi_i, t_j), \quad \xi_i \in (x_{i-1}, x_{i+1})$$

follow from the truncation errors of the numerical differentiation, equations (3.2) and (3.9).

Using the heat-conduction problem, we find that $\boldsymbol{\tau}^{j+1} = \boldsymbol{\varepsilon}^j + \boldsymbol{\mu}^j$, such that method (8.3) is of order $\mathcal{O}(\Delta t + \Delta x^2)$. It makes sense to choose Δt and Δx in such a way that both components in the truncation error have about the same size. This already suggests that the time step should be divided by four if the spatial step size is halved.

Stability
Since the Forward Euler method is only conditionally stable, it is important to determine the size of the time step in such a way that no instabilities will occur. The matrix $\mathbf{K}$ from (8.2) is symmetric, hence the eigenvalues are real. From the Gershgorin circle theorem it follows that

$$-\frac{4}{\Delta x^2} \leq \lambda_i \leq 0, \quad \text{for} \quad 1 \leq i \leq n.$$

The matrix has no positive eigenvalues, hence the system of differential equations (8.2) is analytically stable (Section 6.8.2). An explicit expression for the eigenvalues of $-\mathbf{K}$ was given in equation (7.11). For real eigenvalues, the Forward Euler method is stable if the time step Δt satisfies (equation (6.18))

$$\Delta t \leq -\frac{2}{\lambda_i}, \quad \forall i = 1, \ldots, n.$$

Using that $|\lambda|_{\max} = 4/\Delta x^2$, we obtain $\Delta t \leq \Delta x^2/2$. Note that if the spatial step size is halved, then the time step has to be taken four times as small to satisfy the stability condition.

Note that the condition number of $\mathbf{K}$ is given by (Section 7.3)

$$\kappa(\mathbf{K}) = \frac{\max_{i=1,\ldots,n} |\lambda_i|}{\min_{i=1,\ldots,n} |\lambda_i|} \approx \frac{4}{(\pi \Delta x)^2}.$$

For small values of Δx the initial-value problem (8.2) is a stiff system, since there is a large difference between $|\lambda|_{\max}$ and $|\lambda|_{\min}$. Because of the stability restrictions for explicit methods, it might be more convenient to use an implicit method in that case (Section 6.10).

8.3.2 Backward Euler method

Applying the Backward Euler method (equation (6.51b)) to (8.2) we obtain the formula

$$\mathbf{w}^{j+1} = \mathbf{w}^j + \Delta t(\mathbf{K}\mathbf{w}^{j+1} + \mathbf{r}^{j+1}), \tag{8.4}$$

such that $\mathbf{w}^{j+1}$ needs to be solved from the expression

$$(\mathbf{I} - \Delta t\mathbf{K})\mathbf{w}^{j+1} = \mathbf{w}^j + \Delta t\mathbf{r}^{j+1}. \tag{8.5}$$

The dependency between approximations in this case is depicted in Figure 8.2(b). Again the local truncation error is of order $\mathcal{O}(\Delta t + \Delta x^2)$. Note that the Backward Euler method is unconditionally stable, hence the time step may be chosen in accordance with the accuracy required (stiffness does not impair numerical stability).

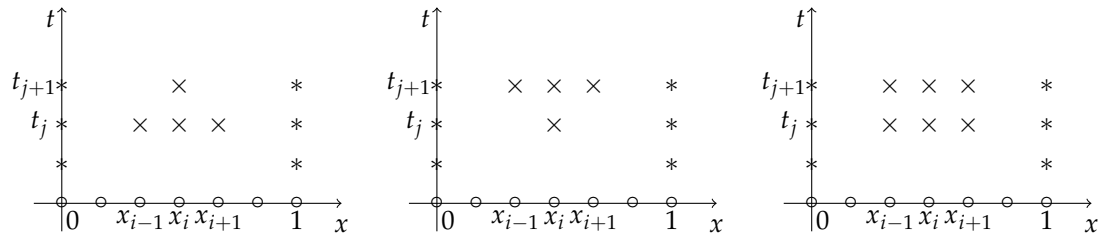
Note that the solution to (8.5) is computationally more costly than determining $\mathbf{w}^{j+1}$ using the Forward Euler method. However, since the matrix $\mathbf{K}$ contains many zeros, there are efficient methods to solve problem (8.5). Careful consideration is necessary: cheap approximation with many time steps (Forward Euler method) versus expensive approximation with few time steps (Backward Euler method).

8.3.3 Trapezoidal method

It is also possible to apply the Trapezoidal method for time integration. In the literature the Trapezoidal method applied to the heat equation is often called the *Crank-Nicolson method*. The corresponding formula that needs to be solved is

$$\mathbf{w}^{j+1} = \mathbf{w}^j + \frac{\Delta t}{2}\left(\mathbf{K}\mathbf{w}^j + \mathbf{r}^j + \mathbf{K}\mathbf{w}^{j+1} + \mathbf{r}^{j+1}\right),$$

cf. equation (6.51c). The corresponding dependency between the approximations is depicted in Figure 8.2(c).



(a) Forward Euler method (b) Backward Euler method (c) Trapezoidal method

Figure 8.2: Dependence of approximations using central-difference semi-discretization and time integration. $\times$: locations of approximations used in time-integration scheme. $\circ$: known by initial condition, $*$: known by boundary condition.

8.4 Summary

In this chapter, the following subjects have been discussed:

- Instationary heat equation;
- Semi-discretization;
- Time integration: Forward Euler, Backward Euler, and Crank-Nicolson method;
- Local truncation error;
- Stability.

References and further reading

Background knowledge:

[1] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, tenth edition, 1972. The Digital Library of Mathematical Functions can be found online: `http://dlmf.nist.gov/`

[2] R.A. Adams. *Calculus: a complete course*. Pearson Addison Wesley, eighth edition, 2013

[3] R.L. Borelli and C.S. Coleman. *Differential Equations: A Modelling Perspective*. John Wiley & Sons, second edition, 2004

[4] W.E. Boyce and R.C. DiPrima. *Elementary Differential Equations*. John Wiley & Sons, tenth edition, 2012

[5] D.C. Lay. *Linear algebra and its applications*. Pearson Addison Wesley, fourth edition, 2010

[6] R.D. Richtmyer and K.W. Morton. *Difference Methods for Initial-Value Problems*. Krieger Publishing Company, second edition, 1994

[7] J. Stewart. *Calculus: Early Transcendentals*. Brooks/Cole, Cengage Learning, seventh edition, 2011

Theoretical Numerical Mathematics:

[8] R.L. Burden and J.D. Faires. *Numerical Analysis*. Brooks/Cole, Cengage Learning, ninth edition, 2011

[9] E. Isaacson and H.B. Keller. *Analysis of Numerical Methods*. Dover Publications, 1994

[10] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer, Science+Business Media, third edition, 2002

[11] E. Süli and D. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, second edition, 2006

Numerical Recipes:

[12] S.C. Chapra and R.P. Canale. *Numerical Methods for Engineers*. McGraw-Hill Education, seventh edition, 2015

[13] G.R. Lindfield and J.E.T. Penny. *Numerical Methods using Matlab*. Academic Press, Elsevier, third edition, 2012

[14] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, third edition, 2007

Index

Numerical Methods for Ordinary Differential Equations

C. Vuik, F.J. Vermolen, M.B. van Gijzen & M.J. Vuik

Groupname
TU Delft | Applied Mathematics

Biography:

Prof.dr.ir. C. (Kees) Vuik is a Full Professor in Numerical Analysis at the Delft Institute of Applied Mathematics of the TU Delft in The Netherlands. He obtained his PhD degree from Utrecht University in 1988. Thereafter he joined the TU Delft as an assistant professor in the section Numerical Analysis. His research is related to the discretization of partial differential equations, moving boundary problems, High-Performance Computing, and iterative solvers for large linear systems originating from incompressible fluid flow; wave equations; and energy networks. He has teached the numerical analysis course for more than 30 years.

Prof.dr.ir. F.J. (Fred) Vermolen is a Full Professor in Computational Mathematics at the University of Hasselt in Belgium. He obtained his PhD degree from the TU Delft in 1998. Thereafter he worked at CWI and from 2000 he joined the TU Delft as an assistant professor in the section Numerical Analysis. His research is related to analysis, numerical methods and uncertainty quantification for partial differential equations. He has given courses in numerical analysis for more than 10 years.

Prof. dr. M.B. (Martin) van Gijzen is Full Professor in High-Performance Computing at the Delft Institute of Applied Mathematics of the TU Delft in The Netherlands. He obtained his PhD degree from the same university in 1994. Before returning to the TU Delft in 2004, he hold positions at the Utrecht University and at TNO, both in The Netherlands, and at CERFACS in France. His research topics include High Performance Computing, iterative solvers for large scale linear systems and eigenvalue problems, model order reduction, and inverse problems. He has worked on many applications, including topology optimisation, MR imaging, numerical modelling of ocean circulation, and acoustic and elastic wave propagation.

Thea Vuik got her PhD from the Numerical Analysis group at Delft University of Technology. In addition to her research on troubled-cell indication in DG methods, she assisted in the Numerical Analysis course for several years. Currently, she is working at VORtech as a scientific software engineer.

ISBN 978-94-6366-665-7
DOI: https://doi.org/10.5074/t.2023.001

textbooks.open.tudelft.nl