Numerical Methods for Ordinary Differential Equations

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

DAP Delft Academic Press

Related titles published by VSSD:
Numerical methods in scientific computing, J. van Kan, A. Segal and F. Vermolen, xii + 279 pp., hardback, 2014 edtion ISBN 97890-6562-3638
www.delftacademicpress.nl/a002.php

In Dutch:
Numerieke methoden voor differentiaalvergelijkingen, J. van Kan, P. van Beek, F. Vermolen, K. Vuik, x + 122 pp. (Dutch version of previous edition of this book)
ISBN 9 7890-7130-1742
www.delftacademicpress.nl/a018.php

Second edition, second revision 2017

Published by Delft Academic Press / VSSD
Leeghwaterstraat 42, 2628 CA Delft, The Netherlands
tel. +31 15 27 82124
dap@vssd.nl
www.delftacademicpress.nl

about this book: **www.delftacademicpress.nl /a026.php**

ISBN- 97890-6562-3744 (ebook edition)
NUR 919

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

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

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 $0.d_1 d_2 \ldots d_n$ 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 $x \in \mathbb{R}$

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

[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.

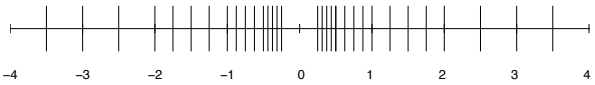

Figure 1.1: Distribution of $\pm 0.1d_2d_3 \cdot \beta^e, \ \beta = 2, e = -1, 0, 1, 2$.

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.