Springer Series in Computational Mathematics 45

Wolfgang Hackbusch

The Concept of Stability in Numerical Mathematics

Springer

Springer Series in Computational Mathematics

45

For further volumes:
http://www.springer.com/series/797

Wolfgang Hackbusch

The Concept of Stability in Numerical Mathematics

Springer

Wolfgang Hackbusch
MPI für Mathematik in den
Naturwissenschaften
Leipzig, Germany

ISSN 0179-3632
ISBN 978-3-642-39385-3 ISBN 978-3-642-39386-0 (eBook)
DOI 10.1007/978-3-642-39386-0
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014931977

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Dedicated to Jana and Jörn

Preface

Usually, the subject of a book on numerical mathematics pertains to a certain field of application or a certain numerical method. In this book we proceed in a different way. Stability is a concept that appears in various fields of numerical mathematics (as well as in other parts of mathematics). Although in each subfield stability is defined differently, there is a common meaning for this term, roughly described by the fact that perturbations are not amplifying the result in a dangerous way. In examining different fields of numerical mathematics concerning stability, we have the opportunity to recall some facts from numerical analysis. However, numerical mathematics cannot control stability exclusively for its own purpose. It turns out that stability is an ambiguous term, which also has strong connections to analysis and functional analysis.

Although stability is an essential requirement for numerical methods, the particular stability conditions are often not as obvious as, e.g., consistency conditions. The book may lead the reader to a better understanding of this term.

This book is an extension of a lecture held in the summer semester of 2003 at the University of Kiel (Christian-Albrechts-Universität zu Kiel). The exposition is self-contained, and the necessary facts from numerical analysis and analysis are provided. Hence, the book is well suited, e.g., as material for seminars in numerical mathematics.

The author wishes to express his gratitude to the publisher Springer for its friendly cooperation. In particular, he thanks Ann Kostant, editorial consultant for Springer, for polishing the English.

Leipzig, October 2013 *Wolfgang Hackbusch*

Contents

List of Symbols

Symbols

$\|\cdot\|_2$	spectral norm of a matrix; cf. §2.4.2.2
$\|\cdot\|_2$	norm of functions in $L^2(\Omega)$
$\|\cdot\|_\infty$	maximum or supremum norm of vectors or functions; cf. §2.4.2.2
$\|\cdot\|_\infty$	row-sum norm of matrices; cf. §2.4.2.2, §5.5.4.1
$\|\cdot\|_X$	norm of the Banach space X
$\|\cdot\|_X^*$	dual norm; cf. §3.3.2
$\|\cdot\|_{Y\leftarrow X}$	operator norm; cf. (3.23)
$\langle\cdot,\cdot\rangle_{X^*\times X}$, $\langle\cdot,\cdot\rangle_{X\times X^*}$	dual pairing; cf. §3.3.2

Greek Letters

$\delta a, \delta A, \ldots$	perturbations of quantities $a, A, \ldots$
$\Delta t, \Delta x$	step size in time and space; cf. §6.3.1
ε	often, symbol for an error
$\varepsilon_{\mathrm{abs}}$	absolute error; cf. footnote 1 on page 6
$\varepsilon_{\mathrm{rel}}$	relative error; cf. footnote 1 on page 6
$\eta(x,h), \eta_i$	discrete solution of ordinary differential equation at x or x_i; cf. §5.1.2
λ	often, an eigenvalue
λ	either $\Delta t/\Delta x$ or $\Delta t/\Delta x^2$ in §6; cf. (6.11)
λ	parameter of the Fredholm integral operator of the second kind; cf. (8.1)
Π_n	projection in §8.1.2.2
$\rho(\cdot)$	spectral radius; cf. (6.25)
$\sigma(M)$	spectrum of matrix M; cf. §5.5.4.1
$\tau(x,y(x);h)$	consistency error of a one-step method; cf. (5.18)
$\Phi_{i,n}(\cdot)$	Lagrange function; cf. (4.2)

$\phi(x_i, \eta_i, [\eta_{i+1},]h; f)$ function defining an explicit [implicit] one-step method; cf. §5.1.2, §5.4.1
$\phi(x_j, \eta_{j+r-1}, \ldots, \eta_j, h; f)$ function defining a multistep method; cf. (5.20a)
$\psi(\zeta)$ characteristic polynomial of a multistep method; cf. (5.21a)
Ω_n grid for difference method; cf. §7.2

Latin Letters

$a_{i,n}$ quadrature weights in §3
B Banach space in §6
$\mathrm{cond}(\cdot)$ condition of a matrix; cf. (8.10)
$\mathbb{C}$ set of complex numbers
$C(\lambda, \Delta t)$ difference operator in §6.3.3
$C(D)$ space of continuous functions defined on D
$C_0(D)$ space of continuous functions with zero boundary value on ∂D
$C^k(D)$ $k \in \mathbb{N}_0$, space of functions with continuous derivatives of order $\leq k$
$C_0^\infty(\mathbb{R})$ infinitely differentiable functions with compact support
$C^\lambda(D)$ $\lambda > 0$, $\lambda \notin \mathbb{N}$, Hölder space; cf. §4.9
C_n stability constant; cf. (3.13b), §4.3, §4.5
C_{stab} stability constant; cf. (3.14), (4.5), §8.2.2, (7.11)
D integration domain for the integral operator; cf. (8.2)
$\mathrm{degree}(\cdot)$ degree of a polynomial
E_j shift operator; cf. (6.22)
eps machine precision; cf. §2.3
$\mathcal{F}$ Fourier map; cf. §6.5.2
$\mathbf{f}_n$ right-hand side in finite difference equation (7.4)
$\mathrm{fl}(\cdot)$ operation in the argument performed in floating point arithmetic; cf. (2.13)
$G(\xi)$ characteristic function; cf. (6.34)
$G(x, \xi)$ characteristic function frozen at x; cf. (6.48)
h step size for ordinary or partial differential equations; cf. §5.1.2
$H^k(\Omega), H^t(\Omega), H_0^k(\Omega), H_0^t(\Omega)$ Sobolev spaces; cf. pages 148, 149
I_n interpolation; cf. (4.4)
J_n Bessel function; cf. (2.2)
$k(x, y)$ kernel of the integral operator; cf. (8.2)
$k_n(x, y)$ approximation of $k(x, y)$; cf. (8.5)
K integral operator; cf. (8.2)
K_n discrete integral operator; cf. (8.4a)
ℓ^2, ℓ^∞ Banach spaces of sequences in $\mathbb{Z}$; cf. §6.3.1
L elliptic differential operator in §7
$\mathcal{L}(X, Y)$ set of linear and continuous mappings from X to Y
$L_{i,n}(\cdot)$ Lagrange polynomial; cf. (4.3)
$\mathbf{L}_n$ finite difference matrix corresponding to L; cf. (7.4)

$L_n(\cdot)$	Legendre polynomial; cf. §3.1.4
$\mathcal{M}$	set of machine numbers; cf. §2.3
$\mathbb{N}$	natural numbers $\{1, 2, \ldots\}$
$\mathbb{N}_0$	$\mathbb{N} \cup \{0\} = \{0, 1, 2, \ldots\}$
$\mathbb{N}', \mathbb{N}''$	suitable infinite subsets of $\mathbb{N}$; cf. §7.2
p	prolongation in §6.3.2
P	often, polynomial
P_Y^n	prolongation in §7.3.1
$Q_n(\cdot)$	quadrature rule; cf. §3
r	restriction in §6.3.2
$r(\cdot)$	numerical radius; cf. §6.5.5
$\mathrm{rd}(\cdot)$	rounding to next machine number; cf. §2.3
$\mathbb{R}$	set of real numbers
R_X^n, R_Y^n	restrictions in §7.3.1
$T(\cdot, h)$	trapezoidal rule of step size h; cf. §3.4.4
$T(t)$	solution operator; cf. §6.2
$\mathbf{u}_n$	solution of finite difference equation (7.4)
U_ν^μ	discretisation of solution $u(t, x)$ in §6 at $t = \mu\Delta t$, $x = \nu\Delta x$
$x_{i,n}$	quadrature points in §3
X, Y	often, Banach spaces
X, X_n, Y, Y_n	Banach spaces in §7.3.1
X^*	dual space $\mathcal{L}(X, \mathbb{R})$; cf. §3.3.2
$\hat{X}, \hat{X}_n$	Banach spaces in §7.3.2
$\mathbb{Z}$	entire numbers

Chapter 1
Introduction

In numerical mathematics we have to distinguish between two types of methods. There are finite algorithms, which solve a given task with a finite number of arithmetical operations. An example of such finite algorithms is the Gauss elimination for solving a system of linear equations. On the other hand, there are problems $\mathcal{P}$ that cannot be solved in finite time. Instead, there are (finite) approximation algorithms that involve solving substituting problems $\mathcal{P}_n$ producing results x_n, which, hopefully, tend to the true solution x of the original problem. The increasing closeness of $\mathcal{P}_n$ to $\mathcal{P}$ for $n \to \infty$ is the subject of the *consistency* condition. What really matters is the *convergence* $x_n \to x$. Whether consistency implies convergence depends on *stability*. It will turn out that under the assumption of consistency and possibly some technical conditions, convergence and stability are equivalent.

The original German manuscript has been used for Diplom students in mathematics. The material in this book is intended for master and Ph.D. students. Besides the discussion of the role of stability, a second goal is to review basic parts of numerical analysis.

We start in Chapter 2 with finite algorithms. We recall the condition of a problem and the stability of an algorithm. The amplification of input and intermediate floating point errors measures the quality of condition and stability. In this respect, the terms remain vague, since the amplification factors are some positive real numbers which may vary between 'small' (stable) and 'large' (unstable) without a clear separation.

Chapter 3 is devoted to quadrature methods, more precisely, to families of quadratures Q_n, where, with increasing n, the quality should improve ('consistency'). 'Stability' is again connected with the amplification of input errors. In contrast to Chapter 2, it is uniquely defined as to whether stability holds or not, since the terms 'small' and 'large' are replaced by finiteness or infiniteness of $\sup C_n$, the supremum of condition numbers C_n.

Although the stability definition is inspired by numerical phenomena, it is also suited to purely analytical purposes. Stability is almost equivalent to convergence of the quadrature result $Q_n(f)$ to the exact integral $\int f \mathrm{d}x$. Correspondingly,

W. Hackbusch, *The Concept of Stability in Numerical Mathematics*,
Springer Series in Computational Mathematics 45, DOI 10.1007/978-3-642-39386-0_1,

analytical tools from functional analysis are involved, namely Weierstrass' approximation theorem and the uniform boundedness theorem.

Interpolation treated in Chapter 4 follows the same pattern as in Chapter 3. In both chapters one can pose the question of how important the stability statement $\sup C_n < \infty$ is, if one wants to perform only one quadrature or interpolation for a *fixed* n. In fact, polynomial interpolation is unstable, but when applied to functions of certain classes it behaves quite well.

This is different in Chapter 5, where one-step and multistep methods for the solution of ordinary initial-value problems are treated. Computing approximations requires an increasing number of steps, when the step size approaches zero. Often an instability leads to exponential growth of an error, eventually causing a termination due to overflow.

For ordinary differential equations instability occurs only for proper multistep methods, whereas one-step methods are always stable. This is different for partial differential equations, which are investigated in Chapter 6. Here, difference methods for hyperbolic and parabolic differential equations are treated. Stability describes the uniform boundedness of powers of the difference operators.

Also in the case of elliptic differential equations discussed in Chapter 7, stability is needed to prove convergence. In this context, stability describes the boundedness of the *inverse* of the difference operator or the finite element matrix independently of the step size.

The final chapter is devoted to Fredholm integral equations. Modern projection methods lead to a very easy proof of stability, consistency, and convergence. However, the Nyström method—the first discretisation method based on quadrature—requires a more involved analysis. We conclude the chapter with the analysis of the corresponding eigenvalue problem.

Despite the general concept of stability, there are different aspects to consider in the subfields. One aspect is the practical importance of stability (cf. §4.6), another concerns a possible conflict between a higher order of consistency and stability (cf. §3.5.2, Remark 4.15, Theorem 5.47, §6.6, §7.5.9).

Chapter 2
Stability of Finite Algorithms

2.1 About Algorithms

An algorithm is used to solve a (numerical) problem. For the mathematical formulation of a *problem* (or *task*) we use a mapping $\Phi : X \to Y$, which is to be evaluated numerically (cf. [9, Chap. 1]).

An algorithm is *executable* if the mapping Φ is composed of units that are realisable in a computer program. We call these units *elementary operations*. In the standard case, these elementary operations are the basic arithmetical operations $+, -, *, /$ in the set of real or whole numbers. In addition, the programming languages offer the use of some special functions like $\sin$, $\cos$, $\exp$, $\sqrt{\cdot}, \ldots$

An *algorithm* is the composition of elementary operations. A *finite algorithm* is characterised as involving only a finite number of elementary operations. The algorithm is a realisation of the mapping

$$\Phi : (x_1, \ldots, x_n) \in X = \mathrm{domain}(\Phi) \mapsto (y_1, \ldots, y_m) \in Y = \mathrm{range}(\Phi). \quad (2.1)$$

If Φ is realisable by at least one algorithm, then there are even infinitely many algorithms of this kind. Therefore, there is no one-to-one correspondence between a task and an algorithm.

A finite algorithm can be described by a sequence of vectors

$$\mathbf{x}^{(0)} = (x_1, \ldots, x_n)\,,\ \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(j)} = (x_1^{(j)}, \ldots, x_{n_j}^{(j)}), \ldots, \mathbf{x}^{(p)} = (y_1, \ldots, y_m)\,,$$

where the values $x_i^{(j)}$ from level j can be computed by elementary operations from the components of $\mathbf{x}^{(j-1)}$.

Example 2.1. *The scalar product* $y = \left\langle \binom{x_1}{x_2}, \binom{x_3}{x_4} \right\rangle$ *has the input vector* $(x_1, \ldots, x_4)$. *The algorithm uses the intermediate values* $\mathbf{x}^{(1)} = (z_1, z_2)$ *with* $z_1 := x_1 x_2$, $z_2 := x_3 x_4$. *Then the output value is obtained from* $y := z_1 + z_2$.

The opposite of a finite algorithm is an infinite one, which, e.g., computes a sequence whose limit is the desired result of the problem. Since one has to terminate

W. Hackbusch, *The Concept of Stability in Numerical Mathematics*,
Springer Series in Computational Mathematics 45, DOI 10.1007/978-3-642-39386-0_2,

such an infinite process, one finally obtains a finite algorithm producing an approximate result.

Here we only want to motivate the concept that algorithms should be constructed carefully regarding stability. It remains to analyse various concrete numerical methods (see, e.g., the monograph of Higham [5]).

2.2 A Paradoxical Example

2.2.1 First Algorithm

The following problem involves the family of *Bessel functions*. In such a case, one is well advised to look into a handbook of special functions. One learns that the n-th *Bessel function* (also called *cylinder function*) can be represented by a power series or as an integral:

$$J_n(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \, (x/2)^{n+2k}}{k! \, (n+k)!} \qquad \text{for } n \in \mathbb{N}_0 \tag{2.2}$$

$$= \frac{(-1)^n}{\pi} \int_0^{\pi} \mathrm{e}^{\mathrm{i}x\cos(\varphi)} \cos(n\varphi) \mathrm{d}\varphi \qquad \text{for } n \in \mathbb{Z}. \tag{2.3}$$

The chosen task is the computation of $J_5(0.6)$.

Assume that tabulated values of the first two Bessel functions J_0 and J_1 are offered as, e.g., in [13, page 99],

$$J_0(0.6) = 0.9120, \quad J_1(0.6) = 0.2867, \tag{2.4}$$

but not the value of $J_5(0.6)$. Furthermore, assume that the book contains the recurrence relation

$$J_{n+1}(x) + J_{n-1}(x) = \frac{2n}{x} J_n(x) \qquad \text{for } n \in \mathbb{Z} \tag{2.5}$$

as well as the property

$$\sum_{n=-\infty}^{\infty} J_n(x) = 1 \qquad \text{for all } x \in \mathbb{R}. \tag{2.6}$$

Exercise 2.2. *Prove convergence of the series (2.2) for all $x \in \mathbb{C}$ (i.e., J_n is an entire function).*

An obvious algorithm solving our problem uses the recursion (2.5) for $n = 1, 2, 3, 4$ together with the initial values (2.4):

$$\begin{aligned}
\mathrm{J}_2(0.6) &= \text{-}\mathrm{J}_0(0.6)+\tfrac{2}{0.6}\mathrm{J}_1(0.6) = \text{-}0.9120+\tfrac{2}{0.6}0.2867 &&= 4.36667_{10}\text{-}2,\\
\mathrm{J}_3(0.6) &= \text{-}\mathrm{J}_1(0.6)+\tfrac{4}{0.6}\mathrm{J}_2(0.6) = \text{-}0.2867+\tfrac{4}{0.6}4.36667_{10}\text{-}2 &&= 4.41111_{10}\text{-}3,\\
\mathrm{J}_4(0.6) &= \text{-}\mathrm{J}_2(0.6)+\tfrac{6}{0.6}\mathrm{J}_3(0.6) = \text{-}4.36667_{10}\text{-}2+\tfrac{6}{0.6}4.41111_{10}\text{-}3 &&= 4.44444_{10}\text{-}4,\\
\mathrm{J}_5(0.6) &= \text{-}\mathrm{J}_3(0.6)+\tfrac{8}{0.6}\mathrm{J}_4(0.6) = \text{-}4.41111_{10}\text{-}3+\tfrac{8}{0.6}4.44444_{10}\text{-}4 &&= 1.51481_{10}\text{-}3.
\end{aligned} \tag{2.7}$$

The result is obtained using only eight elementary operations. The underlying equations are exact. Nevertheless, the computed result for $J_5(0.6)$ is completely wrong, even the order of magnitude is incorrect! The exact result is $J_5(0.6) = 1.99482_{10}$-5.

Why does the computation fail? Are the tabulated values (2.4) misprinted? No, they are as correct as they can be. Is the (inexact) computer arithmetic, used in (2.5), responsible for the deviation? No, even exact arithmetic yields the same results. For those who are not acquainted with numerical effects, this might look like a paradox: exact computations using exact formulae yield completely wrong results.

2.2.2 Second Algorithm

Before we give an explanation, we show a second 'paradox': an algorithm based on inexact and even rather dubious formulae yields a perfect result. A numerical analyst asking for advice would recommend that we use the recurrence relation (2.5) in the opposite order; i.e., starting from $J_m(0.6)$ and $J_{m+1}(0.6)$ for some $m > 5$ and applying (2.5) in the order $n = m-1, m-2, \ldots$. The drawback is that neither $J_m(0.6)$ nor $J_{m+1}(0.6)$ are available. The expert's hint is to replace $J_{m+1}(0.6)$ by zero (vague reasoning: that value does not matter). The unknown value $J_m(0.6)$ will be obtained from the additional property (2.6).

We denote the candidates for $J_n(0.6)$ by j_n. The plan is to start from $j_{m+1} := 0$ and $j_m := J_m(0.6)$ and to apply (2.5): $j_{n-1} = \frac{2n}{0.6} j_n - j_{n+1}$. We observe that all results $j_{m-1}, j_{m-2}, \ldots$ depend linearly on j_m. Therefore, starting from

$$j'_{m+1} := 0, \qquad j'_m := 1 \tag{2.8}$$

and calculating

$$j'_{n-1} = \frac{2n}{0.6} j'_n - j'_{n+1} \qquad \text{for } n = m, m-1, \ldots, 0, \tag{2.9}$$

we get quantities j'_n with the property $j_n = j_m j'_n$. Now the unknown value j_m can be determined from (2.6). From (2.2) or even from (2.5) we can derive that $J_{-n}(x) = (-1)^n J_n(x)$. Hence, (2.6) is equivalent to

$$J_0(x) + 2J_2(x) + 2J_4(x) + \ldots = 1. \tag{2.10}$$

Replacing the infinite sum by a finite one: $j_0 + 2\sum_{\nu=1}^{\lfloor m/2 \rfloor} j_{2\nu} = 1$, we arrive at $j_m \cdot \left(j'_0 + 2\sum_{\nu=1}^{\lfloor m/2 \rfloor} j'_{2\nu} \right) = 1$ and, in particular,