

Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals
in den Bereichen Softwareentwicklung,
Internettechnologie und IT-Management aktuell
und kompetent relevantes Fachwissen über
Technologien und Produkte zur Entwicklung
und Anwendung moderner Informationstechnologien.

Gisela Engeln-Müllges · Klaus Niederdrenk ·
Reinhard Wodicka

Numerik-Algorithmen

Verfahren, Beispiele, Anwendungen

Zehnte, überarbeitete und erweiterte Auflage

Prof. Dr. Gisela Engeln-Müllges
FH Aachen
FB Maschinenbau
Inst. Num. Mathematik + DV
Kesselstr. 88
52076 Aachen
Deutschland
engeln-muellges@fh-aachen.de

Dr. Reinhard Wodicka
Am Kupferofen 34
52066 Aachen
Deutschland

Prof. Dr. Klaus Niederdrenk
FH Münster
FB Chemieingenieurwesen
Stegerwaldstr. 39
48565 Steinfurt
Deutschland

Additional material to this book can be downloaded from <http://extra.springer.com>. Password: 978-3-642-13472-2

ISSN 1439-5428
ISBN 978-3-642-13472-2 e-ISBN 978-3-642-13473-9
DOI 10.1007/978-3-642-13473-9
Springer Heidelberg Dordrecht London New York

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

© Springer-Verlag Berlin Heidelberg 2011

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Einbandentwurf: KuenkelLopka

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.com)

*Im Gedenken an unseren akademischen Lehrer
Prof. Dr. rer. techn. Fritz Reutter (1911 – 1990)
Rheinisch-Westfälische Technische Hochschule Aachen*

Vorwort zur 10. korrigierten und erweiterten Auflage

Auf vielfach geäußerten Wunsch der Leser des Buches haben wir für diese 10. Auflage den Inhalt der 9. Auflage um das Kapitel 16 „Anfangswertprobleme bei gewöhnlichen Differentialgleichungen“ erweitert. Der angekündigte weitere Band wird dann Randwertprobleme gewöhnlicher Differentialgleichungen, eine kurze Einführung in die Numerik partieller Differentialgleichungen sowie, ausführlich und praxisnah, wichtige stochastische Methoden, insbesondere statistische Schätz- und Prüfverfahren, enthalten.

Neben der Erweiterung um das Kapitel 16 wurden für die vorliegende 10. Auflage in den Kapiteln 1 bis 15 entdeckte Fehler korrigiert sowie Ergänzungen vorgenommen. Zum Beispiel wurde das Romberg-Verfahren zur numerischen Differentiation durch eine geeignete Wahl der Anfangsschrittweite optimiert und durch zusätzliche Beispiele erläutert. Bei der numerischen Quadratur wurden die Beispiele zu den Newton-Cotes-Formeln besser dargestellt und erklärt.

In diesem Buch werden zu fast allen Verfahren praxisnah formulierte Algorithmen angegeben. Viele durchgerechnete Beispiele dienen dem Verständnis der numerischen Verfahren und der Erläuterung der Algorithmen. Der Leser findet insgesamt 99 Algorithmen zu den behandelten Verfahren, 233 durchgerechnete Beispiele, viele erläuternde Skizzen und zahlreiche Anwendungsbeispiele aus der Praxis.

In diesem Buch werden viele numerische Verfahren, die für den praktischen Einsatz nützlich sind, zusätzlich zu den gängigen Verfahren behandelt. So werden zum Beispiel in Kapitel 2 auch Einschlussverfahren höherer Konvergenzordnung und Verfahren für mehrfache Nullstellen dargestellt. In Kapitel 3 findet man ausführlich Verfahren zur Bestimmung aller reellen und komplexen Lösungen einer algebraischen Gleichung.

Für Kapitel 4 wurden 32 Algorithmen zu Verfahren für spezielle Matrizen entwickelt, wie sie beispielsweise bei den unterschiedlichen Spline-Verfahren auftreten. Dazu gehören 49 Beispiele.

In Kapitel 10 werden Polynom-Splines (15 Algorithmen, 16 Beispiele) mit verschiedenen Randbedingungen sowie Hermite-Splines und Ausgleichssplines behandelt. Eine Besonderheit bei den parametrischen, kubischen Splines ist eine Verallgemeinerung der chordalen Parametrisierung, die zusätzliche Gestaltungsmöglichkeiten bietet.

In Kapitel 11 werden die praktisch sehr nützlichen nichtparametrischen und parametrischen Subsplines behandelt, die in anderen Büchern in dieser Ausführlichkeit nicht zu finden sind. Mit ihnen können die Länge einer Kurve und der Flächeninhalt einer einfach geschlossenen Kurve ermittelt werden.

Schließlich findet man in Kapitel 14 viele spezielle Quadraturformeln einschließlich der Gaußschen Formeln.

Ganz herzlich danken wir den Autoren der C-Programme Jürgen Dietel, Uli Eggermann und anderen, die diese Programme bereits für die 9. Auflage und frühere Auflagen entwickelt haben. Inzwischen wurde auch die Entwicklung einer C++-Version begonnen, die im DV-Labor des Fachbereiches Maschinenbau und Mechatronik der FH Aachen unter der Leitung von Prof. Dr. Wilhelm Hanrath, dem Nachfolger von Prof. Dr. Gisela Engeln-Müllges, fortgesetzt und dort gepflegt wird; sie kann auch von dort bezogen werden. Herzlichen Dank an Wilhelm Hanrath.

Doris Eggermann danken wir sehr herzlich dafür, dass sie das reproduktionsreife Manuskript mit allen für diese Auflage erforderlichen Ergänzungen und Korrekturen mit dem Satzprogramm LaTeX in altbewährter Zuverlässigkeit, mit größter Präzision und hervorragender Qualität fertig gestellt hat; Uli Eggermann hat sie dabei stark unterstützt. Er hat auch die Zeichnungen für die Darstellung der Richtungsfelder zu den Beispielen in Kapitel 16 angefertigt, herzlichen Dank auch an ihn für die äußerst wertvolle Arbeit.

Gisela Engeln-Müllges
Klaus Niederdrenk
Reinhard Wodicka

Informationen zu Quelltexten für die beschriebenen Algorithmen

Inhalt der aktuellen Distribution (Version 9.1) der Numerik-Bibliothek in C

CNum/src: Ansi-C-Quelltexte von Unterprogrammen zu den meisten der im Buch angegebenen Algorithmen (nach Kapiteln der Bibliothek in weitere Unterverzeichnisse aufgeteilt).

CNum/build: Makefile-Dateien zur Generierung einer Programmbibliothek aus allen Unterprogrammen für unterschiedliche Betriebssysteme:

- CNum/build/Linux/static:
Makefile-Datei zur Erstellung einer statischen Version der Programmbibliothek für das Betriebssystem Linux mit Linux-Standard-Werkzeugen. Mit dieser Makefile-Datei wird eine Headerdatei „CNum.h“ und die (statische) Programmbibliothek „libCNum.a“ erzeugt.
- CNum/build/Linux/dynamic:
Makefile-Datei zur Erstellung einer dynamischen Version der Programmbibliothek für das Betriebssystem Linux mit Linux-Standard-Werkzeugen. Mit dieser Makefile-Datei wird eine Headerdatei „CNum.h“ und die (dynamische) Programmbibliothek „libCNum.so.9.1“ erzeugt.
- CNum/build/Windows/static:
Makefile-Datei zur Erstellung einer statischen Version der Programmbibliothek für das Betriebssystem Windows mit Hilfe des frei verfügbaren Gnu-C-Compilers aus dem Paket MinGW und der Linux-ähnlichen Programmierungsumgebung Msys.

Mit dieser Makefile-Datei wird eine Headerdatei „CNum.h“ und die (statische) Programmbibliothek „libCNum.a“ erzeugt.

Die statische Programmbibliothek kann auch in anderen Entwicklungsumgebungen (etwa in Microsoft-Visual-Studio) verwendet werden.
- CNum/build/Windows/dynamic:
Makefile-Datei zur Erstellung einer dynamischen Version der Programmbibliothek für das Betriebssystem Windows mit Hilfe des frei verfügbaren Gnu-C-Compilers aus dem Paket MinGW und der Linux-ähnlichen Programmierungsumgebung Msys.

Mit dieser Makefile-Datei wird eine Headerdatei „CNum.h“ und die (dynamische) Programmbibliothek „CNum.dll“ erzeugt.

Zur Verwendung der dynamischen Bibliothek CNum.dll unter Microsoft-Visual-Studio wird zusätzlich die Linkbibliothek „CNum.lib“ benötigt, die ebenfalls mit dieser Makefile-Datei unter Verwendung der zur Microsoft-Visual-Studio-Umgebung gehörenden Programme „Lib.exe“ und „Link.exe“ erzeugt werden kann.

CNum/built: fertig erzeugte Programmbibliotheken für unterschiedliche Betriebssysteme:

- CNum/built/Linux/static:
Dateien „CNum.h“ und „libCNum.a“ der statischen Version der Programmbibliothek für das Betriebssystem Linux.
- CNum/built/Linux/dynamic:
Dateien „CNum.h“ und „libCNum.so.9.1“ der dynamischen Version der Programmbibliothek für das Betriebssystem Linux.
- CNum/built/Windows/static:
Dateien „CNum.h“ und „libCNum.a“ der statischen Version der Programmbibliothek für das Betriebssystem Windows.
- CNum/built/Windows/dynamic:
Dateien „CNum.h“ und „CNum.dll“ sowie die Linkbibliothek „CNum.lib“ der dynamischen Version der Programmbibliothek für das Betriebssystem Windows.

Examples/src: Quelltexte zu Demonstrationsbeispielen zu den meisten Algorithmen aus der Programmbibliothek (nach Kapiteln der Bibliothek in weitere Unterverzeichnisse aufgeteilt).

Examples/build: Makefile-Dateien zur Erzeugung von Beispielen zur Demonstration der Verwendung der Programmbibliothek:

- Examples/build/Linux/static/general:
Makefile-Datei zur Erzeugung des Demonstrationsbeispiels „pegasus1“ zur Verwendung der statischen Version der Bibliothek unter Linux.
- Examples/build/Linux/static/docexamples:
Makefile-Datei zur Erzeugung aller Demonstrationsbeispiele aus der Dokumentation und deren Ausgaben unter Verwendung der statischen Version der Bibliothek unter Linux.
- Examples/build/Linux/dynamic/general:
Makefile-Datei zur Erzeugung des Demonstrationsbeispiels „pegasus1“ zur Verwendung der dynamischen Version der Bibliothek unter Linux.
- Examples/build/Windows/MinGW/static/general:
Makefile-Datei zur Erzeugung des Demonstrationsbeispiels „pegasus1“ zur Verwendung der statischen Version der Bibliothek unter Windows/MinGW.
- Examples/build/Windows/MinGW/dynamic/general:
Makefile-Datei zur Erzeugung des Demonstrationsbeispiels „pegasus1“ zur Verwendung der dynamischen Version der Bibliothek unter Windows/MinGW.

- Examples/build/Windows/VC98/static/general:
Projektdatei zur Erzeugung des Demonstrationsbeispiels „pegasus1“ zur Verwendung der statischen Version der Bibliothek unter Windows/Microsoft-Visual-Studio-6.0.
- Examples/build/Windows/VC98/dynamic/general:
Projektdatei zur Erzeugung des Demonstrationsbeispiels „pegasus1“ zur Verwendung der dynamischen Version der Bibliothek unter Windows/Microsoft-Visual-Studio-6.0.

Doc: Dokumentation zur Verwendung der Programmbibliothek und Schnittstellenbeschreibung der einzelnen Unterprogramme aus der Programmbibliothek (inklusive der LaTeX-Quelltexte).

- Doc/CNum.pdf:
Dokumentation als PDF-Datei.
- Doc/CNum.html:
Startdatei zur HTML-Dokumentation (weitere Dateien zur HTML-Dokumentation im Unterverzeichnis „Doc/CNum“).

Bemerkungen zur vorliegenden C-Version

Die Version 9.1 der Numerik-Bibliothek in Ansi-C beruht auf der Vorgängerversion 9.0, wobei inhaltlich nur einige kleinere Änderungen und Korrekturen vorgenommen wurden (Einzelheiten hierzu in der zur Distribution gehörenden Datei „README.txt“), im Wesentlichen aber der Generierungsprozess überarbeitet wurde.

Hauptziel der Überarbeitung des Generierungsprozesses war es, für Linux und Windows jeweils auch eine dynamische Programmbibliothek zu erzeugen (und das möglichst unter Verwendung von Standard-Werkzeugen) und somit für unerfahrene Programmierer (hoffentlich) die Nutzerfreundlichkeit der Bibliothek zu erhöhen.

Ein erfahrener Programmentwickler mag Einzelheiten der Generierung der unterschiedlichen Versionen der Bibliothek aus den Makefile-Dateien entnehmen - hier findet er ggf. auch Anhaltspunkte darüber, wie einzelne Teile der Bibliothek separat übersetzt werden können.

Die Überarbeitung des Generierungsprozesses erfolgte im Datenverarbeitungslabor des Fachbereichs für Maschinenbau und Mechatronik der Fachhochschule Aachen unter der Leitung von Prof. Dr. Wilhelm Hanrath.

Weitere Software im Umfeld der Numerik-Bibliothek

Excel-Interface zur C-Numerik-Bibliothek

Die als DLL vorliegende Windows-Version „CNum.dll“ der C-Numerik-Bibliothek kann jetzt auch über Visual-Basic (VBA) von Microsoft-Excel aus genutzt werden.

Durch entsprechende (ebenfalls im Datenverarbeitungslabor des Fachbereiches 8 der Fachhochschule Aachen entwickelte) C-seitige und VBA-seitige Interfaces sind hierbei über VBA-Makros die Algorithmen aus „CNum.dll“ von Excel aus aufrufbar und die innerhalb der DLL berechneten Ergebnisse können dann anschließend in Excel weiterverwendet werden.

Weitere Informationen zum Excel-Interface zur C-Bibliothek (insbesondere zu Lizenzierung, Bezugsmöglichkeiten und Kosten) finden Sie unter

„www.fh-aachen.de/AcNumBib.html“

und dort unter dem Unterpunkt

„Excel-Interface zur C-Version“.

C++-Version der Numerik-Bibliothek

Im Datenverarbeitungslabor des Fachbereichs 8 der Fachhochschule Aachen wird zur Zeit eine C++-Version der Numerik-Bibliothek entwickelt.

In dieser C++-Version werden die prozeduralen Algorithmen aus den Vorgänger-Versionen der Numerik-Bibliothek unter Verwendung von objektorientierten Techniken nach C++ portiert.

Die Entwicklung der C++-Version ist nahezu abgeschlossen.

Aktuelle Informationen zur C++-Version der Numerik-Bibliothek (insbesondere zu Lizenzierung, Bezugsmöglichkeiten und Kosten) finden Sie unter

„www.fh-aachen.de/AcNumBib.html“

und dort unter dem Unterpunkt

„C++-Version“.

Bezeichnungen

$\Rightarrow$	wenn – dann bzw. hat zur Folge
$\Leftrightarrow$	dann und nur dann
$a := b$	a wird definiert durch b
$\stackrel{!}{=}$	geforderte Gleichheit
$< \leq$	kleiner, kleiner oder gleich
$> \geq$	größer, größer oder gleich
$a \ll b$	a ist wesentlich kleiner als b
$\approx$	ungefähr gleich
$\equiv$	identisch
$\sim$	proportional bzw. gleichmäßig zu
$\{a_1, a_2, \dots\}$	Menge aus den Elementen $a_1, a_2, \dots$
$\{x \dots\}$	Menge aller x für die gilt $\dots$
$\in$	Element von
$\notin$	nicht Element von
$\subseteq$	enthalten in oder Teilmenge von
$\subset$	echt enthalten in oder echte Teilmenge von
$\not\subset$	nicht Teilmenge von
$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{N}_0$	Menge der natürlichen Zahlen mit Null
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{Q}$	Menge der rationalen Zahlen
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{C}$	Menge der komplexen Zahlen
$\mathbb{R}^+, \mathbb{R}^-$	Menge der positiven bzw. negativen reellen Zahlen
(a, b)	offenes Intervall von a bis b , $a < b$
$[a, b]$	abgeschlossenes Intervall von a bis b , $a < b$
$[a, b)$	halboffenes Intervall von a bis b (rechts offen), $a < b$
$(a, b]$	halboffenes Intervall von a bis b (links offen), $a < b$
$n!$	n Fakultät mit $n! = 1 \cdot 2 \cdot 3 \cdots n$, $n \in \mathbb{N}$, $0! := 1$
$\binom{n}{k}$	n über k mit $\binom{n}{k} := \frac{n!}{k!(n-k)!}$, $k \in \mathbb{N}_0$, $k \leq n$, $n \in \mathbb{N}$
$\prod_{i=1}^n a_i$	$a_1 \cdot a_2 \cdot a_3 \dots a_n$
$\sum_{i=1}^n a_i$	$a_1 + a_2 + \dots + a_n$

$\int_a^b$	Integral in den Grenzen a und b
i	imaginäre Einheit $i := \sqrt{-1}$
e	Eulersche Zahl = 2.718 281 828 459 ...
$ a $	Betrag von a mit $ a := \begin{cases} a & \text{für } a \geq 0 \\ -a & \text{für } a < 0 \end{cases}$
$\ \cdot\ $	Norm von $\cdot$
$\{a_k\}$	Folge von a_k
$\lim_{k \rightarrow \infty} a_k$	Limes von a_k für $k \rightarrow \infty$
$\max \{f(x) x \in [a, b]\}$	Maximum aller Funktionswerte $f(x)$ für $x \in [a, b]$
$\min \{ M_i i = 1, 2, \dots, m\}$	Minimum aller $ M_i $ für $i = 1, 2, \dots, m$
(x, y)	geordnetes Paar
$(x_1, x_2, \dots, x_n)$	geordnetes n -Tupel
$f: I \rightarrow \mathbb{R}$	Abbildung f von I nach $\mathbb{R}$
$x \mapsto f(x), x \in D$	x wird $f(x)$ zugeordnet für $x \in D$
$f', f'', f''', f^{(4)}, \dots, f^{(n)}$	erste, zweite, dritte, vierte, ..., n -te Ableitung von f
$C[a, b]$	die Menge der auf $[a, b]$ stetigen Funktionen
$C^n[a, b]$	die Menge der auf $[a, b]$ n -mal stetig differenzierbaren Funktionen
$\mathbb{R}^n$	n -dimensionaler euklidischer Raum
$A = O(h^q)$	$ A/h^q \leq C$ für $h \rightarrow 0$, $C = \text{const.}$
$i = m(1)n$	$m, n \in \mathbb{Z}, \quad m \leq n, \quad i = m, m+1, \dots, n$
$\text{sign}(a), \text{sgn}(a)$	$\begin{cases} 1 & \text{für } a > 0 \\ 0 & \text{für } a = 0 \\ -1 & \text{für } a < 0 \end{cases}$
$\mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$	Vektoren
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \dots$	Matrizen
$\mathbf{0}$	Nullmatrix bzw. Nullvektor
$\mathbf{x} \times \mathbf{y}$	Vektorprodukt bzw. Kreuzprodukt
$\mathbf{E}$	Einheitsmatrix
$\mathbf{A}^\top$	transponierte Matrix von $\mathbf{A}$
$\mathbf{x}^\top = (x_1, x_2, \dots, x_n)$	transponierter Vektor zu $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$
$\mathbf{A}^{-1}$	inverse Matrix von $\mathbf{A}$
$ \mathbf{A} , \det(\mathbf{A})$	Determinante von $\mathbf{A}$
[Verweis]	s. im Literaturverzeichnis unter [Verweis]
o. B. d. A.	ohne Beschränkung der Allgemeinheit
$\square$	Ende eines Beispiels oder eines Beweises

Inhaltsverzeichnis

Vorwort zur 10. Auflage	VII
Informationen zur Programmbibliothek	IX
Bezeichnungen	XIII
1 Darstellung von Zahlen und Fehleranalyse	1
1.1 Definition von Fehlergrößen	1
1.2 Zahlensysteme	3
1.2.1 Darstellung ganzer Zahlen	3
1.2.2 Darstellung reeller Zahlen	6
1.3 Rechnung mit endlicher Stellenzahl	11
1.4 Fehlerquellen	17
1.4.1 Eingabefehler	17
1.4.2 Verfahrensfehler	18
1.4.3 Fehlerfortpflanzung und die Kondition eines Problems	19
1.4.4 Rechnungsfehler und numerische Stabilität	24
2 Lösung nichtlinearer Gleichungen	27
2.1 Aufgabenstellung und Motivation	27
2.2 Definitionen und Sätze über Nullstellen	29
2.3 Allgemeines Iterationsverfahren	31
2.3.1 Konstruktionsmethode und Definition	31
2.3.2 Existenz einer Lösung und Eindeutigkeit der Lösung	34
2.3.3 Konvergenz eines Iterationsverfahrens	37
2.3.3.1 Heuristische Betrachtungen	37
2.3.3.2 Analytische Betrachtung	39
2.3.4 Fehlerabschätzungen und Rechnungsfehler	40
2.3.5 Praktische Durchführung	46
2.4 Konvergenzordnung eines Iterationsverfahrens	49
2.5 Newtonsche Verfahren	51
2.5.1 Das Newtonsche Verfahren für einfache Nullstellen	51
2.5.2 Gedämpftes Newton-Verfahren	57
2.5.3 Das Newtonsche Verfahren für mehrfache Nullstellen. Das modifi- zierte Newtonsche Verfahren	57
2.6 Das Sekantenverfahren	63

2.6.1	Das Sekantenverfahren für einfache Nullstellen	63
2.6.2	Das modifizierte Sekantenverfahren für mehrfache Nullstellen . . .	66
2.7	Einschlussverfahren	66
2.7.1	Das Prinzip der Einschlussverfahren	67
2.7.2	Das Bisektionsverfahren	69
2.7.3	Die Regula falsi	71
2.7.4	Das Pegasus-Verfahren	74
2.7.5	Das Verfahren von Anderson-Björck	77
2.7.6	Die Verfahren von King und Anderson-Björck-King. Das Illinois- Verfahren	80
2.7.7	Ein kombiniertes Einschlussverfahren	81
2.7.8	Das Zeroin-Verfahren	83
2.8	Anwendungsbeispiele	85
2.9	Effizienz der Verfahren und Entscheidungshilfen	89
3	Verfahren zur Lösung algebraischer Gleichungen	91
3.1	Vorbemerkungen	91
3.2	Das Horner-Schema	92
3.2.1	Das einfache Horner-Schema für reelle Argumentwerte	93
3.2.2	Das einfache Horner-Schema für komplexe Argumentwerte	95
3.2.3	Das vollständige Horner-Schema für reelle Argumentwerte	97
3.2.4	Anwendungen	100
3.3	Bestimmung von Lösungen algebraischer Gleichungen	101
3.3.1	Vorbemerkungen und Überblick	101
3.3.2	Das Verfahren von Muller	102
3.3.3	Das Verfahren von Bauhuber	109
3.3.4	Das Verfahren von Jenkins und Traub	111
3.4	Anwendungsbeispiel	112
3.5	Entscheidungshilfen	113
4	Lösung linearer Gleichungssysteme	115
4.1	Aufgabenstellung und Motivation	115
4.2	Definitionen und Sätze	120
4.3	Lösbarkeitsbedingungen für ein lineares Gleichungssystem	132
4.4	Prinzip der direkten Methoden zur Lösung linearer Gleichungssysteme . .	133
4.5	Der Gauß-Algorithmus	136
4.5.1	Gauß-Algorithmus mit Spaltenpivotsuche als Rechenschema	136
4.5.2	Spaltenpivotsuche	141
4.5.3	Gauß-Algorithmus als Dreieckszerlegung	145
4.5.4	Gauß-Algorithmus für Systeme mit mehreren rechten Seiten	149
4.6	Matrizeninversion mit dem Gauß-Algorithmus	151
4.7	Verfahren für Systeme mit symmetrischen Matrizen	153
4.7.1	Systeme mit symmetrischer, streng regulärer Matrix	154
4.7.2	Systeme mit symmetrischer, positiv definiter Matrix. Cholesky- Verfahren	155

4.7.3	Systeme mit symmetrischer, positiv definiter Matrix. Verfahren der konjugierten Gradienten (CG-Verfahren)	160
4.8	Das Gauß-Jordan-Verfahren	164
4.9	Gleichungssysteme mit tridiagonaler Matrix	165
4.9.1	Systeme mit tridiagonaler Matrix	165
4.9.2	Systeme mit symmetrischer, tridiagonaler, positiv definiter Matrix	169
4.10	Gleichungssysteme mit zyklisch tridiagonaler Matrix	172
4.10.1	Systeme mit zyklisch tridiagonaler Matrix	172
4.10.2	Systeme mit symmetrischer, zyklisch tridiagonaler Matrix	175
4.11	Gleichungssysteme mit fünfdiagonaler Matrix	177
4.11.1	Systeme mit fünfdiagonaler Matrix	177
4.11.2	Systeme mit symmetrischer, fünfdiagonaler, positiv definiter Matrix	180
4.12	Gleichungssysteme mit Bandmatrix	183
4.13	Householdertransformation	194
4.14	Fehler, Kondition und Nachiteration	199
4.14.1	Fehler und Kondition	199
4.14.2	Konditionsschätzung	203
4.14.3	Möglichkeiten zur Konditionsverbesserung	208
4.14.4	Nachiteration	208
4.15	Gleichungssysteme mit Blockmatrix	210
4.15.1	Vorbemerkungen	210
4.15.2	Gauß-Algorithmus für Blocksysteme	211
4.15.3	Gauß-Algorithmus für tridiagonale Blocksysteme	213
4.15.4	Weitere Block-Verfahren	214
4.16	Algorithmus von Cuthill-McKee	215
4.17	Entscheidungshilfen	219
5	Iterationsverfahren zur Lösung linearer Gleichungssysteme	223
5.1	Vorbemerkungen	223
5.2	Vektor- und Matrizennormen	223
5.3	Das Iterationsverfahren in Gesamtschritten	225
5.4	Das Gauß-Seidelsche Iterationsverfahren	234
5.5	Relaxation beim Gesamtschrittverfahren	236
5.6	Relaxation beim Einzelschrittverfahren. SOR-Verfahren	236
5.6.1	Schätzung des Relaxationskoeffizienten. Adaptives SOR-Verfahren	237
6	Systeme nichtlinearer Gleichungen	241
6.1	Aufgabenstellung und Motivation	241
6.2	Allgemeines Iterationsverfahren für Systeme	244
6.3	Spezielle Iterationsverfahren	250
6.3.1	Newtonsche Verfahren für nichtlineare Systeme	250
6.3.1.1	Das quadratisch konvergente Newton-Verfahren	250
6.3.1.2	Gedämpftes Newton-Verfahren für Systeme	253
6.3.2	Sekantenverfahren für nichtlineare Systeme	254

6.3.3	Das Verfahren des stärksten Abstiegs (Gradientenverfahren) für nichtlineare Systeme	255
6.3.4	Das Verfahren von Brown für Systeme	257
6.4	Entscheidungshilfen	258
7	Eigenwerte und Eigenvektoren von Matrizen	259
7.1	Definitionen und Aufgabenstellungen	259
7.2	Diagonalähnliche Matrizen	260
7.3	Das Iterationsverfahren nach v. Mises	262
7.3.1	Bestimmung des betragsgrößten Eigenwertes und des zugehörigen Eigenvektors	262
7.3.2	Bestimmung des betragskleinsten Eigenwertes	269
7.3.3	Bestimmung weiterer Eigenwerte und Eigenvektoren	269
7.4	Konvergenzverbesserung	271
7.5	Das Verfahren von Krylov	272
7.5.1	Bestimmung der Eigenwerte	272
7.5.2	Bestimmung der Eigenvektoren	274
7.6	QD-Algorithmus	275
7.7	Transformationen auf Hessenbergform	276
7.7.1	Transformation einer Matrix auf obere Hessenbergform	276
7.7.2	LR - Verfahren	280
7.7.3	QR - Verfahren	282
7.8	Verfahren von Martin, Parlett, Peters, Reinsch und Wilkinson	283
7.9	Entscheidungshilfen	284
7.10	Anwendungsbeispiel	285
8	Lineare und nichtlineare Approximation	291
8.1	Aufgabenstellung und Motivation	291
8.2	Lineare Approximation	294
8.2.1	Approximationsaufgabe und beste Approximation	294
8.2.2	Kontinuierliche lineare Approximation im quadratischen Mittel	296
8.2.3	Diskrete lineare Approximation im quadratischen Mittel	302
8.2.3.1	Normalgleichungen für den diskreten linearen Ausgleich	302
8.2.3.2	Diskreter Ausgleich durch algebraische Polynome unter Verwendung orthogonaler Polynome	308
8.2.3.3	Lineare Regression. Ausgleich durch lineare algebraische Polynome	310
8.2.3.4	Householder-Transformation zur Lösung des linearen Ausgleichsproblems	313
8.2.4	Approximation von Polynomen durch Tschebyscheff-Polynome	316
8.2.4.1	Beste gleichmäßige Approximation, Definition	316
8.2.4.2	Approximation durch Tschebyscheff-Polynome	317
8.2.5	Approximation periodischer Funktionen	323
8.2.5.1	Kontinuierliche Approximation periodischer Funktionen im quadratischen Mittel	324

8.2.5.2	Diskrete Approximation periodischer Funktionen im quadratischen Mittel	326
8.2.5.3	Fourier-Transformation und FFT	329
8.2.6	Fehlerabschätzungen für lineare Approximationen	336
8.2.6.1	Gleichmäßige Approximation durch algebraische Polynome	337
8.2.6.2	Gleichmäßige Approximation durch trigonometrische Polynome	340
8.3	Diskrete nichtlineare Approximation	342
8.3.1	Transformationsmethode beim nichtlinearen Ausgleich	342
8.3.2	Nichtlinearer Ausgleich im quadratischen Mittel	348
8.4	Entscheidungshilfen	348
9	Polynomiale Interpolation sowie Shepard-Interpolation	351
9.1	Aufgabenstellung	351
9.2	Interpolationsformeln von Lagrange	353
9.2.1	Lagrangesche Formel für beliebige Stützstellen	353
9.2.2	Lagrangesche Formel für äquidistante Stützstellen	355
9.3	Aitken-Interpolationsschema für beliebige Stützstellen	356
9.4	Inverse Interpolation nach Aitken	360
9.5	Interpolationsformeln von Newton	362
9.5.1	Newtonsche Formel für beliebige Stützstellen	362
9.5.2	Newtonsche Formel für äquidistante Stützstellen	365
9.6	Abschätzung und Schätzung des Interpolationsfehlers	368
9.7	Zweidimensionale Interpolation	373
9.7.1	Zweidimensionale Interpolationsformel von Lagrange	374
9.7.2	Shepard-Interpolation	376
9.8	Entscheidungshilfen	385
10	Interpolierende Polynom-Splines zur Konstruktion glatter Kurven	387
10.1	Polynom-Splines dritten Grades	387
10.1.1	Aufgabenstellung	390
10.1.2	Woher kommen Splines? Mathematische Analyse	395
10.1.3	Anwendungsbeispiele	397
10.1.4	Definition verschiedener Arten nichtparametrischer kubischer Splinefunktionen	402
10.1.5	Berechnung der nichtparametrischen kubischen Splines	408
10.1.6	Berechnung der parametrischen kubischen Splines	425
10.1.7	Kombinierte interpolierende Polynom-Splines	433
10.1.8	Näherungsweise Ermittlung von Randableitungen durch Interpolation	438
10.1.9	Konvergenz und Fehlerabschätzungen interpolierender kubischer Splines	440
10.2	Hermite-Splines fünften Grades	442
10.2.1	Definition der nichtparametrischen und parametrischen Hermite-Splines	442

10.2.2	Berechnung der nichtparametrischen Hermite-Splines	443
10.2.3	Berechnung der parametrischen Hermite-Splines	447
10.3	Polynomiale kubische Ausgleichssplines	452
10.3.1	Aufgabenstellung und Motivation	452
10.3.2	Konstruktion der nichtparametrischen Ausgleichssplines	456
10.3.3	Parametrische kubische Ausgleichssplines	464
10.4	Entscheidungshilfen für die Auswahl einer geeigneten Splinemethode . . .	465
11	Akima- und Renner-Subsplines	469
11.1	Akima-Subsplines	469
11.2	Renner-Subsplines	476
11.3	Abrundung von Ecken bei Akima- und Renner-Kurven	486
11.4	Berechnung der Länge einer Kurve	490
11.5	Flächeninhalt einer geschlossenen ebenen Kurve	493
11.6	Entscheidungshilfen	496
12	Spezielle Splines	497
12.1	Interpolierende zweidimensionale Polynom-Splines	497
12.2	Zweidimensionale interpolierende Oberflächensplines	511
12.3	Bézier-Splines	514
12.3.1	Bézier-Spline-Kurven	515
12.3.2	Bézier-Spline-Flächen	519
12.3.3	Modifizierte (interpolierende) kubische Bézier-Splines	527
12.4	B-Splines	528
12.4.1	B-Spline-Kurven	528
12.4.2	B-Spline-Flächen	534
12.5	Anwendungsbeispiel	539
12.6	Entscheidungshilfen	544
13	Numerische Differentiation	547
13.1	Aufgabenstellung und Motivation	547
13.2	Differentiation mit Hilfe eines Interpolationspolynoms	548
13.3	Differentiation mit Hilfe interpolierender kubischer Polynom-Splines . . .	551
13.4	Differentiation mit dem Romberg-Verfahren	553
13.5	Entscheidungshilfen	559
14	Numerische Quadratur	561
14.1	Vorbemerkungen	561
14.2	Konstruktion von Interpolationsquadraturformeln	564
14.3	Newton-Cotes-Formeln	567
14.3.1	Die Sehnentrapezformel	569
14.3.2	Die Simpsonsche Formel	575
14.3.3	Die 3/8-Formel	579
14.3.4	Weitere Newton-Cotes-Formeln	583
14.3.5	Zusammenfassung zur Fehlerordnung von Newton-Cotes-Formeln .	587

14.4	Quadraturformeln von Maclaurin	588
14.4.1	Die Tangententrapezformel	588
14.4.2	Weitere Maclaurin-Formeln	591
14.5	Die Euler-Maclaurin-Formeln	592
14.6	Tschebyscheffsche Quadraturformeln	595
14.7	Quadraturformeln von Gauß	597
14.8	Verallgemeinerte Gauß-Quadraturformeln	601
14.9	Quadraturformeln von Clenshaw-Curtis	604
14.10	Das Verfahren von Romberg	605
14.11	Fehlerschätzung und Rechnungsfehler	612
14.12	Adaptive Quadraturverfahren	614
14.13	Konvergenz der Quadraturformeln	615
14.14	Anwendungsbeispiel	617
14.15	Entscheidungshilfen	618
15	Numerische Kubatur	619
15.1	Problemstellung	619
15.2	Konstruktion von Interpolationskubaturformeln	621
15.3	Newton-Cotes-Kubaturformeln für Rechteckbereiche	624
15.4	Das Romberg-Kubaturverfahren	632
15.5	Gauß-Kubaturformeln für Rechteckbereiche	635
15.6	Riemannsche Flächenintegrale	638
15.7	Vergleich der Verfahren anhand von Beispielen	638
15.8	Kubaturformeln für Dreieckbereiche	643
15.8.1	Kubaturformeln für Dreieckbereiche mit achsenparallelen Katheten	643
15.8.1.1	Newton-Cotes-Kubaturformeln für Dreieckbereiche	643
15.8.1.2	Gauß-Kubaturformeln für Dreieckbereiche	646
15.8.2	Kubaturformeln für Dreieckbereiche allgemeiner Lage	650
15.8.2.1	Newton-Cotes-Kubaturformeln für Dreieckbereiche allge- meiner Lage	651
15.8.2.2	Gauß-Kubaturformeln für Dreieckbereiche allgemeiner Lage	654
15.9	Entscheidungshilfen	657
16	Anfangswertprobleme bei gewöhnlichen Differentialgleichungen	659
16.1	Problemstellung	659
16.2	Prinzip der numerischen Verfahren	660
16.3	Einschrittverfahren	661
16.3.1	Das Polygonzugverfahren von Euler-Cauchy	661
16.3.2	Das verbesserte Euler-Cauchy-Verfahren	664
16.3.3	Praediktor-Korrektor-Verfahren von Heun	669
16.3.4	Explizite Runge-Kutta-Verfahren	673
16.3.4.1	Konstruktion von Runge-Kutta-Verfahren	673
16.3.4.2	Klassisches Runge-Kutta-Verfahren	673
16.3.4.3	Zusammenstellung expliziter Runge-Kutta-Formeln	679

16.3.4.4	Einbettungsformeln	684
16.3.5	Implizite Runge-Kutta-Verfahren vom Gauß-Typ	696
16.3.6	Gemeinsame Darstellung aller Einschrittverfahren. Verfahrensfunktion eines Einschrittverfahrens. Konsistenz	698
16.3.7	Fehlerschätzung und automatische Schrittweitensteuerung	700
16.3.7.1	Fehlerschätzung	700
16.3.7.2	Methoden zur automatischen Schrittweitensteuerung. Adaptive Anfangswertproblemlöser	701
16.4	Mehrschrittverfahren	704
16.4.1	Prinzip der Mehrschrittverfahren	704
16.4.2	Das explizite Verfahren von Adams-Bashforth	705
16.4.3	Das Praediktor-Korrektor-Verfahren von Adams-Moulton	707
16.4.4	Verfahren von Adams-Störmer	713
16.4.5	Fehlerschätzungsformeln für Mehrschrittverfahren	714
16.5	Extrapolationsverfahren von Bulirsch-Stoer-Gragg	715
16.6	Stabilität	717
16.6.1	Vorbemerkungen	717
16.6.2	Stabilität der Differentialgleichung	718
16.6.3	Stabilität des numerischen Verfahrens	718
16.7	Steife Differentialgleichungssysteme	723
16.7.1	Problemstellung	723
16.7.2	Kriterien für Steifheit eines Systems	723
16.7.3	Das Verfahren von Gear zur Integration steifer Systeme	724
16.8	Entscheidungshilfen	728

Literaturverzeichnis**733****Sachwortverzeichnis****745**

Kapitel 1

Darstellung von Zahlen und Fehleranalyse, Kondition und Stabilität

1.1 Definition von Fehlergrößen

Ein numerisches Verfahren liefert im Allgemeinen anstelle einer gesuchten Zahl a nur einen Näherungswert A für diese Zahl a . Zur Beschreibung dieser Abweichung werden Fehlergrößen eingeführt.

Definition 1.1. (*Wahrer und absoluter Fehler*)

Ist A ein Näherungswert für die Zahl a , so heißt die Differenz

$$\Delta_a = a - A$$

der *wahren Fehler* von A und deren Betrag

$$|\Delta_a| = |a - A|$$

der *absolute Fehler* von A .

Sehr oft wird in der mathematischen Literatur Δ_a bereits als absoluter Fehler und $|\Delta_a|$ als Absolutbetrag des Fehlers bezeichnet. In ingenieurwissenschaftlichen Anwendungen ist allerdings die Schreibweise in Definition 1.1 häufiger anzutreffen.

In den meisten Fällen ist die Zahl a nicht bekannt, so dass weder der wahre noch der absolute Fehler eines Näherungswertes A angegeben werden können. Daher versucht man, für den absoluten Fehler $|\Delta_a|$ von A eine möglichst kleine obere Schranke $\varepsilon_a > 0$ anzugeben, so dass $|\Delta_a| \leq \varepsilon_a$ gilt.

Definition 1.2. (*Fehlerschranke für den absoluten Fehler, absoluter Höchstfehler*)

Ist $|\Delta_a|$ der absolute Fehler eines Näherungswertes A und ist $\varepsilon_a > 0$ eine obere Schranke für $|\Delta_a|$, so dass

$$|\Delta_a| \leq \varepsilon_a$$

gilt, dann heißt ε_a eine *Fehlerschranke für den absoluten Fehler* von A oder *absoluter Höchstfehler* von A .

Bei bekanntem ε_a ist wegen $|\Delta_a| = |a - A| \leq \varepsilon_a$

$$A - \varepsilon_a \leq a \leq A + \varepsilon_a, \quad \text{also} \quad a \in [A - \varepsilon_a, A + \varepsilon_a]. \quad (1.1)$$

Um einen Näherungswert A unabhängig von der Größenordnung von a beurteilen zu können, wird der relative Fehler eingeführt.

Definition 1.3. (*Relativer Fehler*)

Ist $|\Delta_a|$ der absolute Fehler eines Näherungswertes A für die Zahl a , so heißt der Quotient

$$|\delta_a| = \frac{|\Delta_a|}{|a|} \quad \text{für } a \neq 0$$

der *relative Fehler* von A .

Streng genommen müsste man $\delta_a = \Delta_a/a$ als relativen Fehler von A bezeichnen. Das Vorzeichen von δ_a gibt dann eine zusätzliche Information über die Richtung des Fehlers, d. h. für eine positive Zahl a hat $\delta_a = (a - A)/a < 0$ demnach $a < A$, also einen zu großen Näherungswert A für a zur Folge.

Da in der Regel a unbekannt ist, wird häufig auch

$$|\delta_a| = \frac{|\Delta_a|}{|A|} \quad \text{für } A \neq 0$$

relativer Fehler von A genannt. Da dann auch Δ_a nicht exakt angebbbar ist, wird man sich wieder mit der Angabe einer möglichst guten oberen Schranke für den relativen Fehler behelfen müssen.

Definition 1.4. (*Fehlerschranke für den relativen Fehler, relativer Höchstfehler*)

Ist $|\delta_a|$ der relative Fehler eines Näherungswertes A und gilt mit einem $\varrho_a > 0$

$$|\delta_a| \leq \varrho_a,$$

dann heißt ϱ_a eine *Fehlerschranke für den relativen Fehler* $|\delta_a|$ oder *relativer Höchstfehler* von $|\delta_a|$.

Ist ε_a ein absoluter Höchstfehler von A , so ist

$$\varrho_a = \varepsilon_a/|a| \quad \text{bzw.} \quad \varrho_a = \varepsilon_a/|A|$$

ein relativer Höchstfehler von A .

Definition 1.5. (*Prozentualer Fehler, Fehlerschranke für den prozentualen Fehler*)

Ist $|\delta_a|$ der relative Fehler des Näherungswertes A , so heißt

$$|\delta_a| \cdot 100$$

der *prozentuale Fehler* von A (relativer Fehler in Prozent), und σ_a mit

$$|\delta_a| \cdot 100 \leq 100 \cdot \varrho_a = \sigma_a$$

heißt eine *Fehlerschranke für den prozentualen Fehler*.

Beispiel 1.6.

Für die Zahl $x = \pi$ sind $X = 3.14$ ein Näherungswert und $\varepsilon_x = 0.0016$ eine Schranke für den absoluten Fehler $|\Delta_x|$. Also gilt nach (1.1)

$$3.1384 = 3.14 - 0.0016 \leq \pi \leq 3.14 + 0.0016 = 3.1416,$$

und der relative Höchstfehler ergibt sich zu

$$|\delta_x| \leq \varrho_x = \frac{0.0016}{3.14} \approx 0.00051.$$

Für den prozentualen Fehler folgt nach Definition 1.5

$$100 \cdot |\delta_x| = 0.051\%. \quad \square$$

1.2 Zahlensysteme

1.2.1 Darstellung ganzer Zahlen

Für jede ganze Zahl a gibt es genau eine Potenzentwicklung zur Basis 10 (*Zehnerpotenzen*) der Gestalt

$$a = v \cdot (a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1 10^1 + a_0 10^0) = v \cdot \sum_{k=0}^n a_k 10^k$$

mit dem Vorzeichen $v \in \{-1, 1\}$, den Koeffizienten $a_k \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ und einer nicht negativen ganzen Zahl n ($n \in \mathbb{N}_0$). Diese *Dezimaldarstellung* von a erhält man, indem man die Ziffern, die zur Bezeichnung der Zahlen a_k dienen, in absteigender Reihenfolge aufschreibt:

$$a = v \cdot a_n a_{n-1} \dots a_1 a_0 \tag{1.2}$$

Die Ziffern a_k in der Dezimaldarstellung (1.2) werden auch *Stellen* von a genannt; die Stellung einer Ziffer in der Zahl gibt ihren Wert (Einer – Zehner – Hunderter usw.) wieder.

Algorithmus 1.8. (*Umwandlung einer β -Zahl in eine Dezimalzahl*)

Berechnet man für eine im Stellenwertsystem zur Basis β gegebene ganze Zahl $a = v \cdot (a_n a_{n-1} \dots a_1 a_0)_\beta$, $v \in \{+1, -1\}$, ausgehend von $s_n = a_n$, nacheinander die Größen

$$s_k = \beta \cdot s_{k+1} + a_k, \quad k = n-1, n-2, \dots, 1, 0,$$

dann ist $a = v \cdot s_0$ im Dezimalsystem.

Umgekehrt folgt aus dem letzten Zwischenergebnis

$$a = v \cdot s_0 = v \cdot (s_1 \cdot \beta + a_0),$$

dass a_0 als der Rest der Division der ganzen Zahl s_0 durch β angesehen werden kann:

$$\frac{s_0}{\beta} = s_1 + \frac{a_0}{\beta} \quad \text{oder} \quad \frac{s_0}{\beta} = s_1 \quad \text{Rest } a_0$$

Geht man schrittweise weiter zurück, so folgen die nächsten β -Ziffern a_1, a_2 und so weiter.

Algorithmus 1.9. (*Umwandlung einer Dezimalzahl in eine β -Zahl*)

Berechnet man für eine im Dezimalsystem gegebene ganze Zahl a , ausgehend von $s_0 = |a|$, aus der Gleichung

$$\frac{s_k}{\beta} = s_{k+1} \quad \text{Rest } a_k$$

nacheinander für $k = 0, 1, 2, \dots$ die Größen s_1 und a_0 , s_2 und a_1 usw., bis für ein $k = n$ der Wert $s_{n+1} = 0$ ist, dann ist mit den auf diese Weise gewonnenen Ziffern $a_0, a_1, \dots, a_n \in \{0, 1, \dots, \beta-1\}$ die β -Darstellung von a gegeben durch

$$a = v \cdot (a_n a_{n-1} \dots a_1 a_0)_\beta,$$

wobei $v \in \{-1, 1\}$ das Vorzeichen von a bezeichnet.

Die Zahl 2004 besitzt demnach wegen

$$\begin{array}{rclcl} \frac{2004}{2} & = & 1002 & \text{Rest } 0 & (= a_0) \\ \frac{1002}{2} & = & 501 & \text{Rest } 0 & (= a_1) \\ \frac{501}{2} & = & 250 & \text{Rest } 1 & (= a_2) \\ \frac{250}{2} & = & 125 & \text{Rest } 0 & (= a_3) \\ \frac{125}{2} & = & 62 & \text{Rest } 1 & (= a_4) \\ \frac{62}{2} & = & 31 & \text{Rest } 0 & (= a_5) \\ \frac{31}{2} & = & 15 & \text{Rest } 1 & (= a_6) \\ \frac{15}{2} & = & 7 & \text{Rest } 1 & (= a_7) \\ \frac{7}{2} & = & 3 & \text{Rest } 1 & (= a_8) \\ \frac{3}{2} & = & 1 & \text{Rest } 1 & (= a_9) \\ \frac{1}{2} & = & 0 & \text{Rest } 1 & (= a_{10}) \end{array} \quad \uparrow$$

die Dualdarstellung

$$2004 = (+1) \cdot (11111010100)_2.$$

Rechnerintern wird eine ganze Zahl $a = v \cdot |a|$ als Dualzahl durch eine Reihe von Bits ("binary digits") abgespeichert, gewöhnlich in zwei oder vier Bytes (1 Byte = 8 Bits). Bei einer 2-Byte-Hinterlegung bedeutet das:

1. Byte								2. Byte							
μ	α_{14}	α_{13}	α_{12}	α_{11}	α_{10}	α_9	α_8	α_7	α_6	α_5	α_4	α_3	α_2	α_1	α_0

Das Vorzeichen wird dann über

$$\mu = \begin{cases} 0, & \text{falls } v = +1 \\ 1, & \text{falls } v = -1 \end{cases}$$

gewählt, und im Fall einer positiven Zahl entsprechen α_{14} bis α_0 den Dualziffern von a :

$$a = (+1) \cdot (\alpha_{14}\alpha_{13} \dots \alpha_1\alpha_0)_2$$

Als größte darstellbare positive Zahl ergibt sich dann

$$a = (+1) \cdot (11111111111111)_2 = \sum_{k=0}^{14} 1 \cdot 2^k = \frac{2^{15} - 1}{2 - 1} = 32767.$$

Damit eine Subtraktion auf eine Addition über $a - b = a + (-b)$ zurückgeführt werden kann, werden negative Zahlen rechnerintern durch ein Komplement dargestellt, d. h. im Fall $\mu = 1$ bzw. $v = -1$ stimmen die Hinterlegungen α_k nicht mehr mit den Dualziffern a_k der Zahl $a = (-1) \cdot (a_{14}a_{13} \dots a_1a_0)_2$ überein.

Vorsicht ist geboten, wenn bei einer Addition oder Subtraktion das Ergebnis nicht mehr im darstellbaren Bereich liegt: Addiert man beispielsweise bei einer 1-Byte-Zahl [größte darstellbare positive Zahl: $2^7 - 1 = 127$] $97 = (+1) \cdot (1100001)_2$ und $43 = (+1) \cdot (0101011)_2$, so folgt

$$\begin{array}{r} 0 : 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \\ + \quad 0 : 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \\ \hline 1 : 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \end{array}$$

d. h. ein Übertrag beeinflusst letztendlich das Vorzeichenbit und führt auf ein völlig inplausibles negatives Resultat! Solche Effekte treten leider häufiger auf, ohne dass vom Rechnersystem eine Fehlermeldung oder zumindest eine Warnung ausgesprochen wird. Deshalb ist **bei Rechnerergebnissen stets eine Prüfung auf Plausibilität** erforderlich!

1.2.2 Darstellung reeller Zahlen

Jede nicht ganze, reelle Zahl a besitzt eine Entwicklung der Form

$$a = v \cdot \left(\sum_{k=0}^n a_k 10^k + \sum_{k=1}^{\infty} b_k 10^{-k} \right)$$

mit einem Vorzeichen $v \in \{-1, 1\}$ und Ziffern $a_k, b_k \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, wobei mindestens ein Term

$$b_k 10^{-k} \neq 0$$

auftritt. Die Dezimaldarstellung einer nicht ganzen Zahl heißt *Dezimalbruch*. Im Dezimalbruch einer Zahl a wird zwischen a_0 und b_1 ein Punkt, der sog. *Dezimalpunkt*, gesetzt:

$$a = v \cdot a_n a_{n-1} a_{n-2} \dots a_1 a_0 \cdot b_1 b_2 \dots b_t \dots \quad (1.3)$$

Die rechts vom Punkt notierten Stellen heißen *Dezimalstellen* oder *Dezimalen*. Gibt es in einem Dezimalbruch eine Dezimale $b_j \neq 0$, so dass alle folgenden Dezimalen $b_{j+1} = b_{j+2} = \dots = 0$ sind, dann heißt der Dezimalbruch *endlich*, andernfalls *unendlich*. Eine Darstellung (1.3) mit endlich vielen Dezimalstellen heißt *Festpunktdarstellung*.

Demnach vergegenwärtigt $2\frac{1}{8} = 2.125$ einen endlichen Dezimalbruch und $\frac{1}{3} = 0.3333\dots$ nicht. Dezimalbruchdarstellungen sind, abgesehen vom Sonderfall der Neunerperiode (z. B. ist $-1.29999\dots = -1.\overline{29} = -1.3$), eindeutig.

In einem Stellenwertsystem zur Basis β gilt analog die allgemeine Darstellung

$$\begin{aligned} a &= v \cdot \left(\sum_{k=0}^n a_k \beta^k + \sum_{k=1}^{\infty} b_k \beta^{-k} \right) \\ &= v \cdot (a_n a_{n-1} \dots a_1 a_0 \cdot b_1 b_2 \dots)_\beta \end{aligned}$$

mit $v \in \{+1, -1\}$ und $a_k, b_k \in \{0, 1, \dots, \beta-1\}$; der Index β verweist wieder auf das zugrunde liegende Stellenwertsystem und entfällt nur im vertrauten Fall $\beta = 10$. Der zwischen den Ziffern a_0 und b_1 gesetzte Punkt heißt nun β -*Punkt*, und man nennt $(a_n a_{n-1} \dots a_1 a_0)_\beta$ – manchmal auch unter Berücksichtigung des Vorzeichens v – den *ganzzahligen* Anteil und $(.b_1 b_2 b_3 \dots)_\beta$ den *gebrochenen* oder *fraktionierten* Anteil von a . Der Sonderfall $\beta = 2$ führt nun zur *Dualbruchdarstellung* einer reellen Zahl. Gibt es nur endliche viele „Nachkomma“-Stellen, so spricht man von einem *endlichen β -Bruch*, andernfalls von einem *unendlichen β -Bruch*.

Es gilt der

Hilfssatz 1.10.

Jede rationale Zahl p/q mit teilerfremden $p \in \mathbb{Z}$ und $q \in \mathbb{N}$ wird durch einen endlichen oder durch einen unendlichen periodischen Dezimal- oder Dualbruch dargestellt, jede irrationale Zahl durch einen unendlichen nicht periodischen Dezimal- oder Dualbruch.

Zur Umwandlung reeller Zahlen in ein β -System reicht es, den ganzzahligen Anteil – siehe Abschnitt 1.2.1 – und den gebrochenen Anteil unabhängig voneinander zu konvertieren. Beispielsweise folgt mit $\beta = 2$ aus

$$\begin{aligned} 0.625 &= (+1) \cdot (.b_1 b_2 b_3 \dots)_2 && \text{mit } b_k \in \{0, 1\} \\ &= b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \dots \end{aligned}$$

durch Multiplikation mit $\beta = 2$

$$\begin{aligned} 1.25 &= b_1 + b_2 2^{-1} + b_3 2^{-2} + \dots \\ &= (b_1 . b_2 b_3 \dots)_2 . \end{aligned}$$

Somit entspricht b_1 dem ganzzahligen Anteil 1 von 1.25 und weiter

$$0.25 = (.b_2b_3\dots)_2 = b_2 2^{-1} + b_3 2^{-2} + \dots$$

Führt man nun mit einer erneuten Multiplikation mit $\beta = 2$ fort

$$0.5 = (b_2.b_3\dots)_2 = b_2 + b_3 2^{-1} + \dots,$$

so führt der Vergleich der ganzzahligen und der gebrochenen Anteile in dieser Gleichung auf $b_2 = 0$ und $0.5 = (.b_3b_4\dots)_2$. Der nächste gleichartige Schritt ergibt über

$$1.0 = (b_3.b_4\dots)_2 = b_3 + b_4 2^{-1} + \dots$$

schließlich $b_3 = 1$ und $b_k = 0$ für $k \geq 4$. Mithin gilt

$$0.625 = (+1) \cdot (.101)_2,$$

was auch plausibel ist, denn 0.625 ist die Summe von $\frac{1}{2} = 2^{-1}$ und $\frac{1}{8} = 2^{-3}$.

Allgemein formuliert heißt das:

Algorithmus 1.11. (*Umwandlung einer echt gebrochenen Zahl in einen β -Bruch*)

Gegeben sei eine echt gebrochene Zahl a , $-1 < a < 1$.

Berechnet man, ausgehend von $c_0 = |a|$, für $k = 1, 2, 3, \dots$ nacheinander die Größen

$$\begin{aligned} b_k &= \text{int}(\beta \cdot c_{k-1}) && \text{„ganzzahliger Anteil von } \beta \cdot c_{k-1}\text{“,} \\ c_k &= \beta \cdot c_{k-1} - b_k && \text{„echt gebrochener Anteil von } \beta \cdot c_{k-1}\text{“,} \end{aligned}$$

dann ist

$$a = v \cdot (.b_1b_2b_3\dots)_\beta,$$

wobei $v \in \{-1, 1\}$ das Vorzeichen von a wiedergibt.

Die Berechnung wird abgebrochen, wenn hinreichend viele „Nachkomma“-Stellen ermittelt wurden.

Beispiel 1.12.

Algorithmus 1.11 führt mit $a = 0.1$ auf die folgenden Größen für das Dualsystem:

$$\begin{aligned} b_1 &= \text{int}(0.2) = 0, & c_1 &= 0.2 \\ b_2 &= \text{int}(0.4) = 0, & c_2 &= 0.4 \\ b_3 &= \text{int}(0.8) = 0, & c_3 &= 0.8 \\ b_4 &= \text{int}(1.6) = 1, & c_4 &= 0.6 \\ b_5 &= \text{int}(1.2) = 1, & c_5 &= 0.2 \\ b_6 &= \text{int}(0.4) = 0, & c_6 &= 0.4 \\ &\dots & &\dots \end{aligned}$$

Man erkennt, dass sich die Dualziffernfolge periodisch wiederholt:

$$\begin{aligned} 0.1 &= (+1) \cdot (.00011001100110011\dots)_2 \\ &= (+1) \cdot (.000\overline{11})_2. \end{aligned}$$

Mithin entspricht 0.1 einem unendlichen periodischen Dualbruch. $\square$

Definition 1.13. (*Tragende Ziffern*)

Alle Ziffern in der β -Darstellung einer Zahl $a = v \cdot (a_n a_{n-1} \dots a_0 \cdot b_1 b_2 \dots)_\beta$ mit $v \in \{+1, -1\}$ und $a_k, b_k \in \{0, 1, \dots, \beta-1\}$, beginnend mit der ersten von Null verschiedenen Ziffer ($a_n \neq 0$), heißen *tragende Ziffern*.

Beispiel 1.14.

$a = 0.0024060$	besitzt 7 Dezimalen	und 5 tragende Ziffern,
$a = 1573800$	besitzt keine Dezimalen	und 7 tragende Ziffern.
$a = 47.110$	besitzt 3 Dezimalen	und 5 tragende Ziffern.
$a = 0.1$	besitzt 1 Dezimale	und 1 tragende Ziffer.

Für die letzte Zahl ergibt sich im Dualsystem $a = (0.00011)_2$, eine Zahl mit unendlich vielen Dualstellen und somit unendlich vielen tragenden Ziffern. $\square$

Definition 1.15. (*Normalisierte Gleitpunktdarstellung*)

Jede reelle Zahl $a \neq 0$ kann als β -Zahl in der Form

$$a = v \cdot (.d_1 d_2 d_3 \dots d_s d_{s+1} \dots)_\beta \cdot \beta^k \quad (1.4)$$

für ein $k \in \mathbb{Z}$ dargestellt werden, wobei $v \in \{+1, -1\}$ das Vorzeichen von a angibt und $d_1 \neq 0$ gilt. (1.4) heißt *normalisierte β -Gleitpunktdarstellung* von a , $m = (.d_1 d_2 d_3 \dots)_\beta$ ihre β -Mantisse, und k ihr β -Exponent. Besitzt die Mantisse s tragende Ziffern, $s \in \mathbb{N}$, so heißt sie *s-stellig*.

So besitzen $a = 346.5201$ und $b = -0.005386$ im Dezimalsystem die normalisierten Gleitpunktdarstellungen

$$\begin{aligned} a &= 0.3465201 \cdot 10^3 && (7\text{-stellige Mantisse}), \\ b &= -0.5386 \cdot 10^{-2} && (4\text{-stellige Mantisse}). \end{aligned}$$

Einem Computer stehen für Berechnungen nur endlich viele in ihm darstellbare Zahlen, die *Maschinenzahlen*, zur Verfügung. Die Mantissen m dieser Maschinenzahlen haben gewöhnlich eine feste Anzahl von Ziffern. Ferner ist der Exponent $k \in \mathbb{Z}$ durch $-k_1 \leq k \leq k_2$ mit $k_1, k_2 \in \mathbb{N}$ begrenzt. Eine weltweit gültige Norm nach ANSI (American National Standards Institute) und IEEE (Institute of Electrical and Electronics Engineers) schreibt für reelle Zahlen, als Dualbruch mit 4 Bytes (= 32 Bits) im Rechner hinterlegt, das Format

$$\mu : e_1 \dots e_8 : d_2 \ d_3 \dots \dots \dots \dots \dots \dots d_{24}$$