Numerische Mathematik I

C. Cryer

Wintersemester 1993/94

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Definition der numerischen Mathematik

Numerische Mathematik wird oft auch als praktische Mathematik oder numerische Analysis bezeichnet. Die vier Wörter *praktisch*, *Mathematik*, *Analysis* und *numerisch* verweisen auf die verschiedenen Bestandteile des Gebietes (siehe Abb. 1.1).

Praktisch bedeutet, daß die Grundprobleme der praktischen Mathematik praxisbezogen sind und daß praktische Mathematik ein Teil der angewandten Mathematik ist. In den Ingenieur- und Naturwissenschaften ist es selbstverständlich, daß mathematische Methoden benutzt werden. Es gibt aber viele andere Wissenschaften, z.B. Erdkunde, Biologie und Betriebswirtschaft, bei denen mathematische Modelle zunehmend benutzt werden.

Mathematisch bedeutet, daß praktische Mathematik sich nur mit mathematischen Problemen beschäftigt.

Analysis bezieht sich auf zwei Kennzeichen von praktischer Mathematik.

1. Aus historischen Gründen befaßt sich praktische Mathematik nur mit mathematischen Problemen der Analysis und nicht mit Problemen der Statistik, der Optimierung oder der Kombinatorik, obwohl diese für die Modellierung auch sehr wichtig sind.

2. Es wird viel Wert darauf gelegt, die Methoden der praktischen Mathematik genau zu analysieren. Es muß aber betont werden, daß mehrere Methoden unter Bedingungen angewandt werden, wofür die vollständige Analysis noch fehlt.

Numerisch bedeutet, daß die Lösung eines Problems letztendlich Berechnungen erfordert. Es besteht deshalb ein enger Zusammenhang zwischen den gebräuchlichen Methoden und den vorhandenen Rechenfazilitäten, da die Methoden praktisch anwendbar sein müssen. Die ersten Computer wurden zwecks Lösung von numerischen Aufgaben gebaut, und es bestehen noch viele Verbindungen zwischen der Numerik und der Informatik.

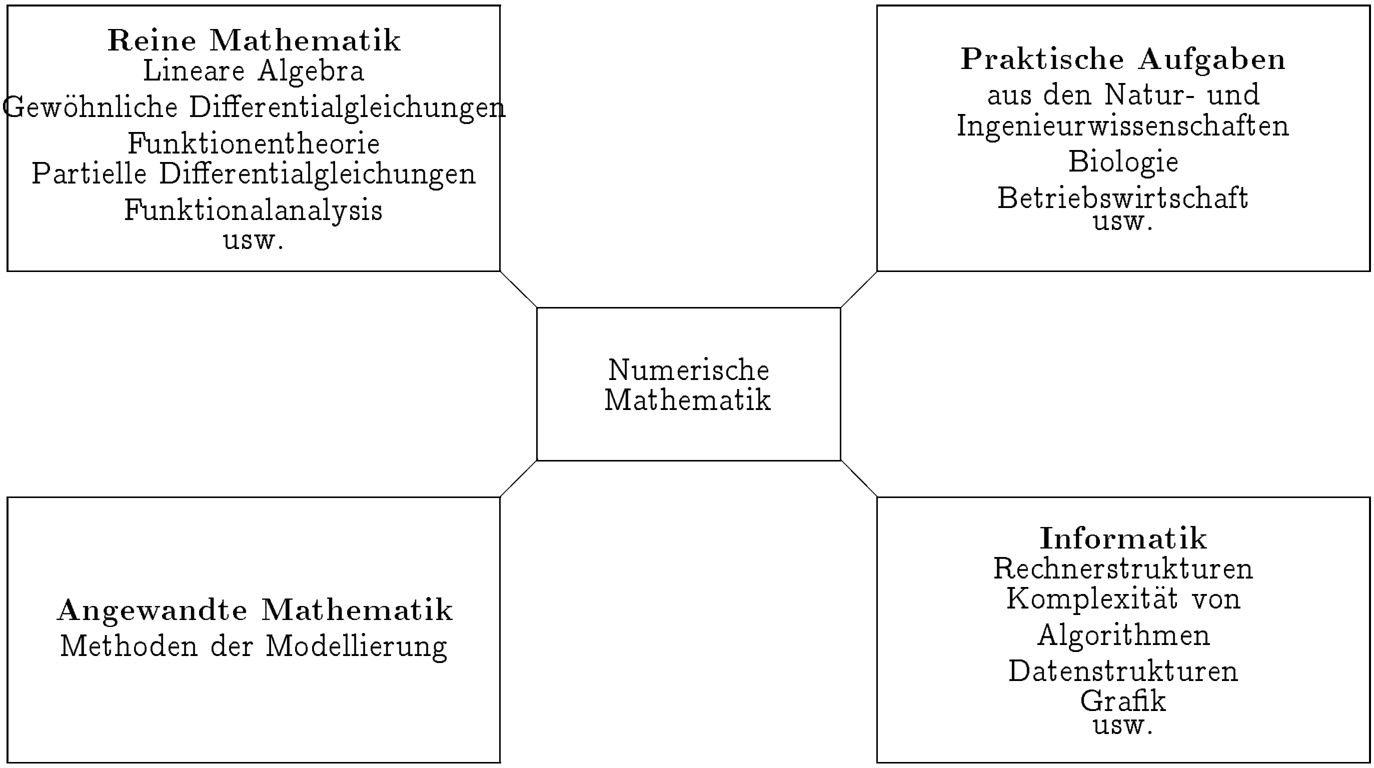


Abbildung 1.1: Die angrenzenden Gebiete der praktischen Mathematik

1.2 Grundaufgaben der praktischen Mathematik

Die Grundaufgaben der praktischen Mathematik werden traditionell folgendermaßen aufgegliedert:

1. Mathematische Hilfsmittel.
2. Eigenschaften von Rechenanlagen und Rechnerarithmetik. Einfache Fehleranalyse.
3. Die Lösung von Systemen nicht-linearer algebraischer Gleichungen.
4. Die Approximation von Funktionen durch einfachere Funktionen, z.B. Polynomen und Splines.
5. Numerische Integration und Differentiation.
6. Die numerische Lösung von Integral- und Integral-Differential- gleichungen.
7. Probleme der linearen Algebra, insbesondere die Lösung von Gleichungssystemen und die Berechnung von Eigenwerten.

8. Die numerische Lösung von gewöhnlichen Differentialgleichungen.

9. Die numerische Lösung von partiellen Differentialgleichungen.

Die verschiedenen Grundaufgaben können oft getrennt behandelt werden. In den Anwendungen kommen aber oft mehrere Grundaufgaben zu gleicher Zeit vor.

Für jede der Grundaufgaben mußte folgendes gemacht werden:

1. Verschiedene Methoden oder Algorithmen zur Lösung der Grundaufgabe müssen gegeben werden.

2. Für jede der Algorithmen muß mathematisch bewiesen werden, daß der Algorithmus die Grundaufgabe löst. Wenn das nur unter bestimmten Voraussetzungen gilt, müssen diese Voraussetzungen bestimmt werden.

   Es kann sein, daß eine Grundaufgabe nicht genau gelöst werden kann, weil dazu unendlich viel Rechenzeit benötigt würde. Zum Beispiel ist es nicht möglich, das Integral einer allgemeinen Funktion genau zu berechnen, weil ein Integral durch einen unendlichen Limesprozeß definiert ist. Für solche Grundaufgaben kann nur gefragt werden, ob ein Algorithmus die Aufgaben mit einem kleinen Fehler lösen kann. Es muß dann bewiesen werden, daß dieser Fehler mit steigendem Rechenaufwand nach Null strebt, und das Verhältnis des Fehlers zum Rechenaufwand muß angegeben werden.

3. Es gibt Algorithmen, die kleine Fehler in den Dateneingaben oder kleine Fehler in den Berechnungen sehr stark vergrößern. Solche Algorithmen heißen *schlecht konditioniert* und müssen gekennzeichnet werden, damit sie vermieden werden können.

4. Wenn möglich sollte die Gesamtmenge von Algorithmen erfaßt und verglichen werden, damit die besten Algorithmen (bezüglich Rechenaufwand, Programmieraufwand usw.) ausgesucht werden können.

1.3 Ein Beispiel

Um zu verdeutlichen, wie die verschiedenen Grundaufgaben in der Praxis auftreten können, betrachten wir ein Modellproblem aus der Strömungsmechanik.

Ein rechteckiger Behälter Ω wird mit Flüssigkeit gefüllt, und die obere Seite (der Deckel) wird in Bewegung gesetzt. Unter bestimmten physikalischen Voraussetzungen kann gezeigt werden, daß der Druck p und die Komponenten u, v des Geschwindigkeitsvektors in

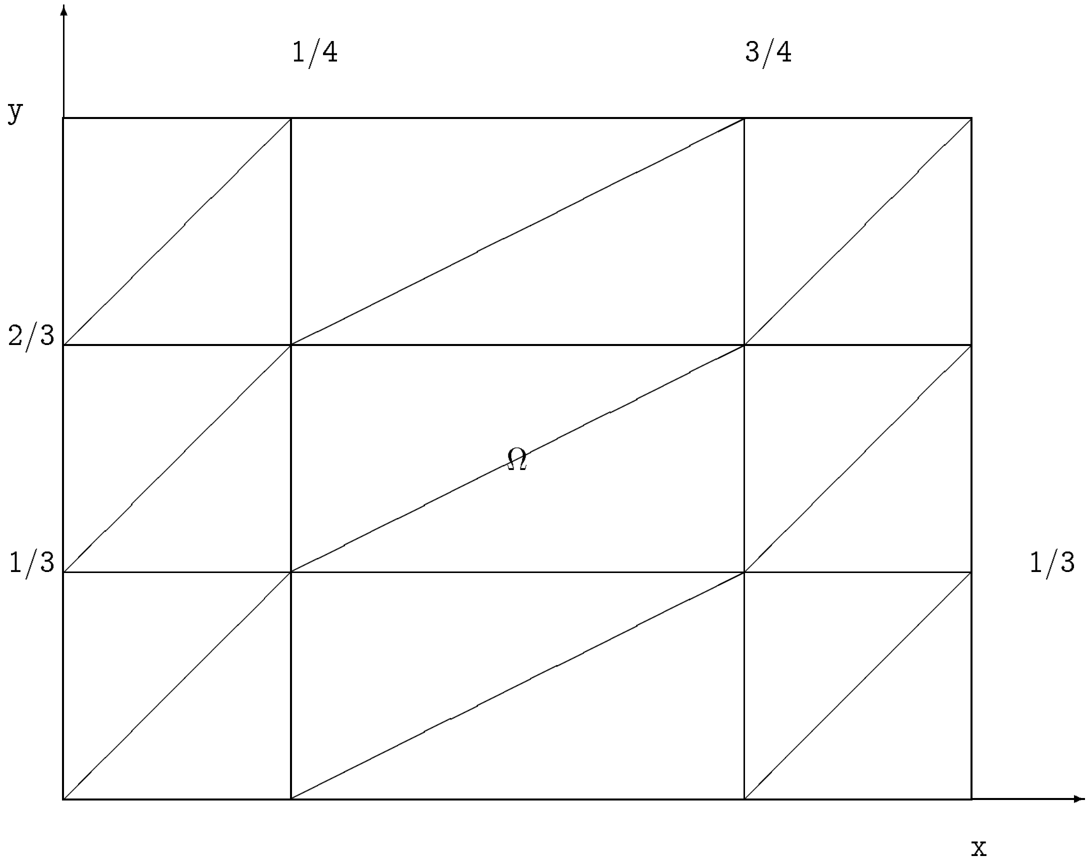


Abbildung 1.2: Die Zerlegung des Gebietes Ω

der x- bzw. y-Richtung die Navier-Stokes-Gleichungen erfüllen:

$$\begin{aligned} uu_x + vu_y &= -p_x + \nu(u_{xx} + u_{yy}) \\ uv_x + vv_y &= -p_y + \nu(v_{xx} + v_{yy}) \\ u_x + v_y &= 0 \end{aligned} \tag{1.1}$$

mit $u_x = \frac{\partial u}{\partial x}$ usw.

Die Funktionen p, u, v müssen die Gleichungen (1.1) für alle $(x, y) \in \Omega$ erfüllen. Damit die Lösung eindeutig ist, müssen p, u, v zusätzliche Randbedingungen auf dem Rande $\partial\Omega$ erfüllen, die wir nicht angeben.

Um dieses Problem mit Hilfe *finiter Elemente* zu lösen, sind folgende Schritte erforderlich:

1. Schritt:
Das Gebiet Ω wird in kleine Gebiete E_i zerlegt. In Abb. 1.2 wird Ω in Dreiecke E_i zerlegt, $1 \le i \le 18$. (In der Praxis werden mehrere hundert Dreiecke benutzt.)

2. Schritt:

In jedem der Elemente wird für die gesuchten Funktionen u, v, p eine einfache Approximation mit noch unbekannten Parametern gewählt, z.B.:

$$p(x,y) \approx p^{(i)} := c_1^{(i)} + c_2^{(i)} x + c_3^{(i)} y = \sum_{j=1}^{3} c_j^{(i)} \varphi_j(x,y) \quad \text{in} \quad E_i$$

Die Wahl einer geeigneten Approximation ist eine Aufgabe der *Approximationstheorie*.

Wird Ω wie in Abb. 1.2 zerlegt, ergeben sich insgesamt n = 162 unbekannte Koeffizienten, die als Vektor $\mathbf{c} \in \mathbb{R}^n$ zusammengefaßt werden können.

3. Schritt:
Um den Vektor $\mathbf{c}$ zu bestimmen, wird verlangt, daß n Gleichungen erfüllt werden müssen. Jede dieser Gleichungen entspricht einer Integralbedingung, wie z.B.:

$$\int_\Omega (UU_x + VU_y) W \, dx \, dy = \int_\Omega [-P_x + \nu(U_{xx} + U_{yy})] W \, dx \, dy$$
$$U = U(\mathbf{c}, x, y), \quad V = V(\mathbf{c}, x, y), \quad P = P(\mathbf{c}, x, y), \quad W = W(x, y).$$

Zusammengefaßt: es gilt:

$$F(\mathbf{c}) = 0$$

mit

$$F : \mathbb{R}^n \to \mathbb{R}^n .$$

Die Angabe der Abbildung F erfordert *symbolische Integration* oder *numerische Integration*.

4. Schritt:
Zur Lösung der *nichtlinearen Gleichung* $F(\mathbf{c}) = 0$ müssen iterative Verfahren benutzt werden, die grundsätzlich als Zwischenschritt die *Lösung von linearen Gleichungen* erfordern:

$$A^{(k)} \mathbf{c}^{(k+1)} = b^{(k)} ,$$

mit

$$A^{(k)} \in M(n \times n)$$
$$k = 0, 1, 2, \cdots \text{ (Iterationszahl)} .$$

1.4 Geschichte

Der Anfang der numerischen Analyse läßt sich mit der Erfindung von Logarithmen (Napier 1614) verbinden. Wichtige Verfahren sind früh von Newton (1643-1727) entwickelt worden.

Unter den deutschen Mathematikern, die bedeutende frühe Beiträge zur numerischen Mathematik geleistet haben, sind: **Gauß** (1777-1855), **Runge** (1856-1927).

Seit der Einführung von elektronischen Digitalrechnern (ENIAC - Februar 1946) hat sich die numerische Analyse weitgehend geändert. Früher wurden praktische Methoden betont (siehe z.B. Willers [1957]). Digitalrechner haben es erforderlich gemacht, Algorithmen genauer zu untersuchen.

Kapitel 2

Informationshilfsmittel

Es kommt in der Praxis und Forschung oft vor, daß Informationen über mathematische Probleme und Software eingeholt werden müssen. Es stehen mehrere Informationshilfsmittel zur Verfügung (Stand 14. Oktober 1990):

2.1 Klassische Bibliotheksdienste

Klassische Bibliotheken bieten Stichwortkataloge usw. an. Falls Bücher oder Zeitschriften nicht in der Universitätsbibliothek oder Fachbibliotheken vorhanden sind, können sie durch die Fernleihe ausgeliehen werden:

1. Die UB Münster bietet Fernleihefazilitäten an.

2. Die Universität Bielefeld ist Standort für das Zeitschriften- Schwerpunktprogramm (Mathematik) der Hochschulbibliotheken des Landes NRW. Zeitschriftenartikel können aus Bielefeld direkt angefordert werden.

Es gibt mehrere Referatenorgane (die alle in Münster vorhanden sind):

1. Zentralblatt für Mathematik und ihre Grenzgebiete (seit 1931). Jährlich 54000 Referate (1988). Mehr als 2000 Zeitschriften und Serien ausgewertet, rund 2000 Monographien und Sammelbände erfaßt.

2. Mathematical Reviews (seit 1940).

3. Current Mathematical Publications (Herausgeber: Amer. Math. Soc.).

4. Computing Reviews.

5. Science Citation Index (in der UB: ZC 1956, 1957, 1958, Kat Saal). Diese Zeitschrift hat sowohl einen ‘Citation Index’ als auch einen ‘Permuterm Subject Index’.

6. Dissertation Abstracts International (in der UB: Z Qu 3721, Kat Saal).

2.2 Tabellenwerke

Die Berechnung von mathematischen Konstanten und die Erstellung von Tabellen waren früher wichtige Aufgaben. Die Benutzung von Tabellen wird später im Zusammenhang mit der Interpolation angesprochen. Allgemeine Tabellenwerke sind:

1. Abramowitz, M. und Stegun, I.A.:
   Handbook of Mathematical Functions. National Bureau of Standards, Washington, 1964.
   Dies ist heute das am meisten benutzte Tabellenwerk. Es ist in billigen Ausgaben vorhanden. Eine gekürzte Ausgabe ist auch verfügbar.

2. Jahnke-Emde-Lösch:
   Tafeln Höherer Funktionen. Teubner, 1960.
   Die sechste Ausgabe eines berühmten Werkes.

3. A. Fletcher, J.C.P. Miller, L. Rosenhead:
   An Index of Mathematical Tables, 2 vols. Addison-Wesley, 1962. Gibt an, wo Tabellen in Büchern und Zeitschriften vorhanden sind.

4. Lebedev, A.V. und Fedorova, R.M.:
   A Guide to Mathematical Tables. Pergamon, 1960. Supplement 1 (Burunova, N.M.). Gibt an, wo Tabellen in Büchern und Zeitschriften vorhanden sind.

5. Gradshteyn, I.S. und Ryzhik, I.W.:
   Table of Integrals, Series and Products. Academic Press, 1965.
   Eine bekannte Formelsammlung.

6. Die Zeitschrift „Mathematics of Computation”, die Fortsetzung der Zeitschrift „Mathematical Tables and Aids to Computation”, enthält Informationen über neue Tabellenwerke.

Alle diese Tabellenwerke sind in der Mathematischen Bibliothek oder der Bibliothek des Instituts für Numerische und instrumentelle Mathematik vorhanden.

2.3 Mathematische Systeme

Es gibt mehrere mathematische Systeme, die integrierte symbolische, numerische und graphische Fazilitäten anbieten:

- Maple (in C)
- Mathematica
- Reduce (in Lisp)
- Derive
- SMP

Diese Systeme können exakte Arithmetik durchführen, mehrere Integrale analytisch exakt berechnen usw. Diese Systeme werden die Benutzung von Tabellenwerken wahrscheinlich überflüssig machen.

2.4 Software

Es gibt mehrere Quellen von mathematischer Software:

1. **Vollständige Software-Bibliotheken:**

   (a) NAG Bibliothek (Numerical Algorithms Group). Mark 14 (1990) umfaßt 889 Routinen.

   (b) IMSL Bibliothek

   Beide Bibliotheken sind z.B. auf dem Großrechner verfügbar.

2. **„Public Domain“ Software-Pakete:**

   (a) EISPACK (Eigenwertaufgaben)

   (b) LINPACK (Lineare Algebra)

   (c) ELLPACK (Elliptische partielle Differentialgleichungen)

   Weitere Beispiele werden unter NETLIB (siehe unten) zitiert.

3. **Wissensbasierte Systeme:** Um die Benutzung von Software-Bibliotheken zu erleichtern, werden z.Zt. wissensbasierte Systeme wie NAXPERT, KASTLE oder GAMS eingesetzt.

4. **Kleinere Software Bibliotheken:** Es gibt in der letzten Zeit mehrere kleinere Sammlungen von numerischen Algorithmen, die zu niedrigen Preisen erhältlich sind. Zum Beispiel:

   (a) **Gerald**, Curtis F., **Wheatley**, Patrick O.: Applied Numerical Analysis. Fourth Edition. Addison Wesley, 1989.

   (b) **Kahaner**, D., **Moler**, C., **Nash**, St.: Numerical Methods and Software. Prentice Hall, 1989.

   (c) **Köckler**, N.: Numerische Algorithmen in Softwaresystemen. Teubner, 1990.

   (d) **Press**, W.H., **Flannery**, B.P., **Teukolsky**, S.A., **Vetterling**, W.T.: Numerical Recipes. Cambridge University Press, 1987.

5. **Literatur:**

   (a) Die Zeitschrift ACM TOMS (Association for Computing Machinery, Transactions on Mathematical Software) ist mathematischen Algorithmen gewidmet. Insbesondere wurden bisher über 680 FORTRAN Algorithmen (die sogenannten ACM Algorithmen) veröffentlicht und dokumentiert.
   Diese Zeitschrift ist in der Bibliothek des Instituts für Numerische und instrumentelle Mathematik vorhanden.

   (b) Die ACM (Association for Computing Machinery) SIG (Special Interest Group) SIGNUM (Special Interest Group on Numerical Mathematics) ist eine internationale Zeitschrift für Numeriker.

   (c) Es gibt auch mehrere Informationsblätter, z.B.:

      - Multigrid Newsletter
        Klaus Stüben
        GMD/F1
        Postfach 1240
        5205 St. Augustin 1
      - Rundbrief Numerische Software. Herausgegeben von der Fachgruppe 2.2.2 Numerische Software der GI. Anschrift: W. Mackens, Institut für Angewandte Mathematik der Universität Hamburg, Bundesstraße 55, 2000 Hamburg 13.
      - Computer-Algebra Rundbrief. Herausgegeben von der Fachgruppe 2.2.1 Computer-Algebra der GI. Anschrift: Dr. F. Schwarz, GMD, Institut F1, Postfach 1240, 5205 St. Augustin.

2.5 Elektronische Informationsdienste

Es gibt mehrere elektronische Informationsdienste:

1. **NANET** ist eine weltweite elektronische Vernetzung von Numerikern. Es gibt ein wöchentliches Rundschreiben, und es ist möglich, Fragen an alle Teilnehmer zu richten. Zum Beispiel e-mail an na.cryer@na-net. stanford. edu wird an mich weitergeleitet.

2. **NETLIB** ist ein Informationsdienst mit folgender elektronischer Adresse:

   USA: NETLIB@ORNL.GOV
   Europa: NETLIB@NAC.NO

   mit dessen Hilfe eine große Anzahl von mathematischen Programmen als auch Information über neue Computer usw. kostenlos zu erhalten ist.

   Die folgenden Informationen wurden mit dem Befehl

   MAIL
   NETLIB@NAC.NO
   Send index

   eingeholt.

```
===== How to use netlib =====

This file is the reply you'll get to:
    mail netlib@nac.no
    send index
Here are examples of the various kinds of requests.
*  get the full index for a library
    send index from eispack
*  get a particular routine and all it depends on
    send dgeco from linpack
*  get just the one routine, not subsidiaries
    send only dgeco from linpack
*  get dependency tree, but excluding a subtree
    send dgeco but not dgefa from linpack
*  just tell how large a reply would be, don't actually send the file
    send list of dgeco from linpack
*  search for somebody in the SIAM membership list:
    who is gene golub
*  keyword search for netlib software  (somewhat out of date)
    find cubic spline
*  bibliographic search
    find schumaker from approximation
    find aasen from linalg
*  notes on the Fortran-to-C converter
    send index for f2c

The Internet address "netlib@nac.no" refers to a machine
in Oslo, Norway.  This address
```

```
should be understood on all the major networks.
        EARN/BITNET:    netlib%nac.no@norunix.bitnet
        X.400:          s=netlib; o=nac; c=no;
        EUNET/uucp:     nac!netlib
A similar collection of statistical software is available from
    statlib@temper.stat.cmu.edu.

An excellent guide to the mysteries of networks and address syntax is:
Donnalyn Frey and Rick Adams (1989) "!%@:: A Directory of Electronic
Mail Addressing and Networks", O'Reilly & Associates, Inc, 632 Petaluma
Ave, Sebastopol CA 95472.  Background about netlib is in Jack J.
Dongarra and Eric Grosse, Distribution of Mathematical Software Via
Electronic Mail, Comm. ACM (1987) 30,403-407.

Bugs reports, comments, and annual lists of recipients will be
forwarded to the code authors when possible.  Many of these codes are
designed for use by professional numerical analysts who are capable of
checking for themselves whether an algorithm is suitable for their
needs.  One routine can be superb and the next awful.  So be careful!

An inventory list is given below and in the indices for the individual
libraries.  If you know exactly what you're looking for, these guides
may be enough.  An interactive system called "walk" provides a more
systematic list (not limited to netlib) but at present covers only
approximation.  Volunteers from other fields are needed.  The reference
is Eric Grosse, "A Catalog ...", in Algorithms for Approximation",
Mason and Cox (eds.), Chapman and Hall, 1989.  Dialup (at 1200 baud)
201-582-1238  or telnet to research.att.com and login as walk; no
password is required.

-------quick summary of contents---------
a - approximation algorithms
alliant - set of programs collected from Alliant users
amos - special functions by D. Amos.  = toms/644
apollo - set of programs collected from Apollo users
benchmark - various benchmark programs and a summary of timings
bib - bibliographies
bihar - Bjorstad's biharmonic solver
blas3 - matrix * matrix BLAS
bmp - Brent's multiple precision package
c - another "misc" library, for software written in C
cascade - analysis and design of linear control systems
cheney-kincaid - programs from the text Numerical Mathematics and Computing.
conformal - Schwarz-Christoffel codes by Trefethen; Bjorstad+Grosse
core - machine constants, vector and matrix * vector BLAS
c++ - code in the C++ language
dierckx - spline fitting
domino - communication and scheduling of multiple tasks; Univ. Maryland
eispack - matrix eigenvalues and vectors
elefunt - Cody and Waite's tests for elementary functions
```

```
errata - corrections to numerical books
f2c - Fortran to C converter
fishpack - separable elliptic PDEs; Swarztrauber and Sweet
fitpack - Cline's splines under tension
fftpack - Swarztrauber's Fourier transforms
fmm - software from the book by Forsythe, Malcolm, and Moler
fn - Fullerton's special functions
uncon/data - optimization test problems
gcv - Generalized Cross Validation
go - "golden oldies"  gaussq, zeroin, lowess, ...
graphics - auto color, ray-tracing benchmark
harwell - MA28 sparse linear system
hompack - nonlinear equations by homotopy method
itpack - iterative linear system solution by Young and Kincaid
jakef - automatic differentiation of Fortran subroutines
lanczos - Cullum and Willoughby's Lanczos programs
laso - Scott's Lanczos program for eigenvalues of sparse matrices
linpack - gaussian elimination, QR, SVD  by Dongarra, Bunch, Moler, Stewart
lp - linear programming
machines - short descriptions of various computers
matlab - software from the MATLAB user's group
microscope - Alfeld and Harris' system for discontinuity checking
minpack - nonlinear equations and least squares by More, Garbow, Hillstrom
misc - everything else
na-digest - archive of mailings to NA distribution list
napack - numerical algebra programs
ode - ordinary differential equations
odepack - ordinary differential equations from Hindmarsh
paranoia - Kahan's floating point test
parmacs  -  parallel programmming macros
pchip - hermite cubics  Fritsch+Carlson
picl - portable instrumented communication library for multiprocessors
pltmg - Bank's multigrid code;  too large for ordinary mail
polyhedra - Hume's database of geometric solids
port - the public subset of PORT library
pppack - subroutines from de Boor's Practical Guide to Splines
quadpack - univariate quadrature by Piessens, de Donker, Kahaner
sched - environment for portable parallel algorithms in a Fortran setting.
slap - Seager + Greenbaum, iterative methods for symmetric and unsymmetric
slatec - machine constants and error handling package from the Slatec library
sparse - Kundert + Sangiovanni-Vincentelli, C sparse linear algebra
sparse-blas - BLAS by indirection
sparspak - George + Liu, sparse linear algebra core
specfun - transportable special functions
toeplitz - linear systems in Toeplitz or circulant form by Garbow
toms - Collected Algorithms of the ACM
typesetting - typesetting macros and preprocessors
vanhuffel - total least squares, partial SVD  by Van Hufell
vfftpk - vectorized FFT; variant of fftpack
voronoi - Voronoi diagrams and Delaunay triangulations
```

```
y12m - sparse linear system  (Aarhus)

--------a bit more detail--------

The first few libraries here are widely regarded as being of high quality.
The likelihood of your encountering a bug is relatively small; if you do,
we certainly want to hear about it!   mail ehg@research.att.com

CORE    Machine constants (i1mach,r1mach,d1mach), blas (level 1 and 2)

EISPACK  A collection of Fortran subroutines that compute the eigenvalues
    and eigenvectors of nine classes of matrices.  The package can
    determine the eigensystems of complex general, complex Hermitian,
    real general, real symmetric, real symmetric band, real symmetric
    tridiagonal, special real tridiagonal, generalized real, and
    generalized real symmetric matrices. In addition, there are two
    routines which use the singular value decomposition to solve
    certain least squares problems.
    Developed by the NATS Project at Argonne National Laboratory.
    (d.p. refer to eispack, s.p. refer to seispack)

FFTPACK A package of Fortran subprograms for the Fast Fourier
    Transform of periodic and other symmetric sequences
    This package consists of programs which perform Fast Fourier
    Transforms for both complex and real periodic sequences and
    certian other symmetric sequences.
    Developed by Paul Swarztrauber, at NCAR.
VFFTPK   a vectorized version of FFTPACK for multiple sequences.
    By Sweet, Lindgren, and Boisvert.

FISHPACK A package of Fortran subprograms providing finite difference
    approximations for elliptic boundary value problems.
    Developed by Paul Swarztrauber and Roland Sweet.

FNLIB   Wayne Fullerton's special function library. (single and double)

GO      Golden Oldies:  routines that have been widely used,
    but aren't available through the standard libraries.
    Nominations welcome!

HARWELL  Sparse matrix routine MA28 from the Harwell library.  from Iain Duff

LINPACK  A collection of Fortran subroutines that analyze and solve linear
    equations and linear least squares problems.  The package solves
    linear systems whose matrices are general, banded, symmetric
    indefinite, symmetric positive definite, triangular, and tridiagonal
    square.  In addition, the package computes the QR and singular value
    decompositions of rectangular matrices and applies them to least
    squares problems.
    Developed by Jack Dongarra, Jim Bunch, Cleve Moler and Pete Stewart.
```

```
    (all precisions contained here)

PPPACK  Subroutines from: Carl de Boor, A Practical Guide to Splines,
    Springer Verlag. This is an old version, from around the time the book
    was published.  We will install a newer version as soon as we can.

TOMS    Collected algorithms of the ACM. When requesting a specific
    item, please refer to the Algorithm number.

----------------

In contrast to the above libraries, the following are collections of codes
from a variety of sources.  Most are excellent, but you should exercise
caution.  We include research codes that we haven't tested and codes
that may not be state-of-the-art but useful for comparisons.
The following list is chronological, not by merit:

MISC    Contains various pieces of software collected over time and:
    the source code for the netlib processor itself;
    the paper describing netlib and its implementation;
    the abstracts list maintained by Richard Bartels.

FMM      Routines from the book Computer Methods for Mathematical
    Computations, by Forsythe, Malcolm, and Moler.
    Developed by George Forsythe, Mike Malcolm, and Cleve Moler.
    (d.p. refer to fmm, s.p. refer to sfmm)

QUADPACK A package for numerical computation of definite univariate integrals.
    Developed by Piessens, Robert(Appl. Math. and Progr. Div.- K.U.Leuven)
    de Donker, Elise(Appl. Math. and Progr. Div.- K.U.Leuven
    Kahaner, David(National Bureau of Standards)      (slatec version)

TOEPLITZ A package of Fortran subprograms for the solution of systems
    of linear equations with coefficient matrices of Toeplitz or
    circulant form, and for orthogonal factorization of column-
    circulant matrices.
     Developed by Burt Garbow at Argonne National Laboratory,
    as a culmination of Soviet-American collaborative effort.
    (d.p. refer to toeplitz, s.p. refer to stoeplitz)

ITPACK   Iterative Linear System Solver based on a number of methods:
    Jacobi method, SOR, SSOR with conjugate gradient acceleration
    or with Chebyshev (semi-iteration - SI) acceleration.
    Developed by Young and Kincaid and the group at U of Texas.

BIHAR   Biharmonic solver in rectangular geometry and polar coordinates.
    These routines were obtained from Petter Bjorstad,
    Veritas Research, Oslo Norway in July 1984.

LANCZOS procedures computing a few eigenvalues/eigenvectors of a large (sparse)
```

```
    symmetric matrix.  Jane Cullum and Ralph Willoughby, IBM Yorktown.

LASO    A competing Lanczos package.  David Scott.

CONFORMAL   contains routines to solve the "parameter problem" associated
    with the Schwarz-Christoffel mapping.  Includes:
    SCPACK (polygons with straight sides) from Nick Trefethen.
    CAP (circular arc polygons) from Petter Bjorstad and Eric Grosse.

FITPACK  A package for splines under tension.  (an early version)
    For a current copy and for other routines, contact:
    Pleasant Valley Software, 8603 Altus Cove, Austin TX 78759, USA

BENCHMARK contains benchmark programs and the table of Linpack timings.

MACHINES contains information on high performance computers that
    are or soon to be made available

MINPACK  A package of Fortran programs for the solution of systems of
    nonlinear equations and nonlinear least squares problems.
    Five algorithmic paths each include a core subroutine and an
    easy-to-use driver.  The algorithms proceed either from an analytic
    specification of the Jacobian matrix or directly from the problem
    functions.  The paths include facilities for systems of equations
    with a banded Jacobian matrix, for least squares problems with a
    large amount of data, and for checking the consistency of the
    Jacobian matrix with the functions.
    Developed by Jorge More', Burt Garbow, and Ken Hillstrom at
    Argonne National Laboratory.
    (d.p. refer to minpack, s.p. refer to sminpack)

PORT    The public subset of the PORT library.  Includes the latest version
    of Gay's NL2SOL nonlinear least squares. The rest of the PORT3
    library is available by license from AT&T.

Y12M  calculation of the solution of systems of linear systems of
    linear algebra equations whose matrices are large and sparse.
    authors: Zahari Zlatev, Jerzy Wasniewski and Kjeld Schaumburg

PCHIP is a fortran package for piecewise cubic hermite inter-
    polation of data.  It features software to produce a monotone and
    "visually pleasing" interpolant to monotone data.
    Fred N. Fritsch, Lawrence Livermore National Laboratory

LP      Linear Programming - At present, this consists of one subdirectory,
    data: a set of test problems in MPS format, maintained by David Gay.
    For more information, try a request of the form
    send index for lp/data

ODE various initial and boundary value ordinary differential equation
```

```
    solvers: colsys, dverk, rkf45, ode
    A subset of these in single precision is in the library sode.

ODEPACK The ODE package from Hindmarch and others.
    This is the double precision verison; to get sp refer to sodepack.
    Alan Hindmarch, Lawrence Livermore National Laboratory

ELEFUNT is a collection of transportable Fortran programs for testing
    the elementary function programs provided with Fortran compilers.  The
    programs are described in detail in the book "Software Manual for the
    Elementary Functions" by W. J. Cody and W. Waite, Prentice Hall, 1980.

SPECFUN is an incomplete, but growing, collection of transportable
    Fortran programs for special functions, and of accompanying test
    programs similar in concept to those in ELEFUNT.
    W.J. Cody, Argonne National Laboratory

PARANOIA is a rather large program, devised by Prof. Kahan of Berkeley,
    to explore the floating point system on your computer.

SLATEC library    DoE policy apparently prohibits us from distributing this.
    Contact the National Energy Software Center or your congressman.

HOMPACK is a suite of FORTRAN 77 subroutines for solving nonlinear systems
    of equations by homotopy methods.  There are subroutines for fixed
    point, zero finding, and general homotopy curve tracking problems,
    utilizing both dense and sparse Jacobian matrices, and implementing
    three different algorithms: ODE-based, normal flow, and augmented
    Jacobian.

DOMINO is a set of C-language routines with a short assembly language
       interface that allows multiple tasks to communicate and schedules
       local tasks for execution.  These tasks may be on a single processor
       or spread among multiple processors connected by a message-passing
       network.  (O'Leary, Stewart, Van de Geijn, University of Maryland)

GCV     software for Generalized Cross Validation from:  O'Sullivan,
        Woltring (univariate spline smoothing), Bates, Lindstrom,
        Wahba and Yandell (multivariate thin plate spline smoothing
        and ridge regression), Gu (multiple smoothing parameters),
    Fessler (vector smoothing spline)

Cheney-Kincaid  programs from: Ward Cheney & David Kincaid, Numerical
       Mathematics and Computing.

POLYHEDRA    a database of angles, vertex locations, and so on for over a
       hundred geometric solids, compiled by Andrew Hume.

GRAPHICS    presently just contains some C routines for testing ray-tracing
```

```
A      approximation algorithms  (almost empty, but soon to grow)
    lowess:   multivariate smoothing of scattered data; Cleveland+Devlin+Grosse

Apollo  A set of programs collected from Apollo users.

Alliant A set of programs collected from Alliant users.

parmacs  -  parallel programmming macros for monitors and send/receive
      Rusty Lusk, Argonne National Laboratory, June 5, 1987 (lusk@anl-mcs.arpa)

sched - The Schedule Package is an environment for the transportable
        implementation of parallel algorithms in a Fortran setting.
        Jack Dongarra and Dan Sorensen, Argonne National Laboratory,
        June 5, 1987 (dongarra@anl-mcs.arpa sorensen@anl-mcs.arpa)

NAPACK  A collection of Fortran subroutines to solve linear systems,
        to estimate the condition number or the norm of a matrix,
        to compute determinants, to multiply a matrix by a vector,
        to invert a matrix, to solve least squares problems, to perform
        unconstrained minimization, to compute eigenvalues, eigenvectors,
        the singular value decomposition, or the QR decomposition.
        The package has special routines for general, band, symmetric,
        indefinite, tridiagonal, upper Hessenberg, and circulant matrices.
        Code author: Bill Hager, Mathematics Department, University of
        Florida, Gainesville, FL 32611, e-mail: hager@math.ufl.edu.
        Related book: Applied Numerical Linear Algebra, Prentice-Hall,
        Englewood Cliffs, New Jersey, 1988.

SPARSPAK  Subroutines from the book "Computer Solution of Large Sparse
        Positive Definite Systems" by George and Liu, Prentice Hall 1981.

VANHUFFEL
   The TLS problem assumes an overdetermined set of linear equations
   AX = B, where both the data matrix A as well as the observation
   matrix B are inaccurate.
   The subroutine PTLS solves the Total Least Squares (TLS) problem by
   using a Partial Singular Value Decomposition (PSVD), hereby improving
   considerably the computational efficiency with respect to the classi-
   cal TLS algorithm.
      Sabine VAN HUFFEL
      ESAT Laboratory, KU Leuven.
      Kardinaal Mercierlaan 94, 3030 Heverlee, Belgium

DIERCKX
   A package of spline fitting routines for various kinds of data and
   geometries.  Written by: Professor Paul Dierckx, Dept. Computer Science,
   K. U. Leuven, Celestijnenlaan 200A, B-3030 Heverlee, Belgium.

VORONOI
   Algorithms for Voronoi regions and Delaunay triangulations.  Currently
```

```
   contains Fortune's 2d sweepline method.

SPARSE  A library of subroutines written in C that solve large sparse
        systems of linear equations using LU factorization.  The
        package is able to handle arbitrary real and complex square
        matrix equations.  Besides being able to solve linear systems,
        it is solves transposed systems, find determinants, multiplies
        a vector by a matrix, and estimate errors due to
        ill-conditioning in the system of equations and instability in
        the computations.  Sparse does not require or assume symmetry
        and is able to perform numerical pivoting (either diagonal or
        complete) to avoid unnecessary error in the solution.  Sparse
        also has an optional interface that allow it to be called from
        FORTRAN programs.
    Ken Kundert, Alberto Sangiovanni-Vincentelli. (sparse@ic.berkeley.edu)

SLAP This is  the official release version  2.0  of the  Sparse Linear
     Algebra Package:  a  SLAP for the   Masses!  It  contains "core"
     routines for the  iterative solution symmetric  and non-symmetric
     positive  definite  and positive  semi-definite  linear  systems.
     Included in  this package  are  core routines  to   do  Iterative
     Refinement    iteration,   Preconditioned    Conjugate    Gradient
     iteration,  Preconditioned Conjugate Gradient   iteration on  the
     Normal Equations,  Preconditioned BiConjugate Gradient iteration,
     Preconditioned BiConjugate Gradient  Squared iteration,  Orthomin
     iteration  and  Generalized  Minimum  Residual  iteration.   Core
     routines  require the   user  to supply  "MATVEC"  (Matrix Vector
     Multiply) and "MSOLVE" (Preconditiong) routines.  This allows the
     core routines to be written in a way  that makes them independent
     of the matrix data  structure.  For each  core routine  there are
     several  drivers and support  routines  that allow   the user  to
     utilize  Diagonal Scaling and  Incomplete Cholesky/Incomplete  LU
     factorization as preconditioners with no  coding.   The price for
     this  convience is that one  must use the  a specific matrix data
     structure: SLAP Column or SLAP Triad format.
        Written by Mark K. Seager & Anne Greenbaum

sequent software from the Sequent Users Group.
      Jack Dongarra 9/88

UNCON/DATA test problems: unconstrained optimization, nonlinear least squares.
    Problems from More, Garbow, and Hillstrom;  Fraley, matrix square
    root;  Hanson, Salane;  McKeown;  De Villiers and Glasser;
    Dennis, Gay, and Vu.   Collected by Chris Fraley.

JAKEF is a precompiler that analyses a given Fortran77 source code for
    the evaluation of a scalar or vector function and then generates an
    expanded Fortran subroutine that simultaneously evaluates the gradient
    or Jacobian respectively. For scalar functions the ratio between the
    run-time of the resulting gradient routine and that of the original
```

```
    evaluation routine is never greater than a fixed bound of about five.
    The storage requirement may be considerable as it is also proportional
    to the run-time of the original routine. Since no differencing is done
    the partial derivative values obtained are exact up to round-off errors.   A
 Griewank, Argonne National Laboratory, griewank@mcs.anl.gov, 12/1/88.

sparse-blas an extension to the set of Basic Linear Algebra Subprograms.
     The extension is targeted at sparse vector operations, with the goal of
     providing efficient, but portable, implementations of algorithms for high
     performance computers.
     convex!dodson@anl-mcs.ARPA Mon Aug 31 19:53:21 1987 (Dave Dodson)

matlab - software from the MATLAB Users Group.
     Christian Bischof bischof@mcs.anl.gov 12/89

picl - is a subroutine library that implements a generic message-passing
       interface for a variety of multiprocessors. It also provides
       timestamped trace data, if requested.
       authors: Geist, Heath, Peyton, and Worley, Oak Ridge National Lab.
       worley@msr.epm.ornl.gov 4/17/90.

MADPACK is a a compact package for solving systems of linear equations using
multigrid or aggregation-disaggregation methods.  Imbedded in the algorithms
are implementations for sparse Gaussian elimination and symmetric Gauss-Seidel
(unaccelerated or accelerated by conjugate gradients or Orthomin(1)).  This
package is particularly useful for solving problems which arise from
discretizing partial differential equations, regardless of whether finite
differences, finite elements, or finite volumes are used.
It was written by Craig Douglas.    installed 2/90

FORTRAN    contains tools specific to Fortran.
    At present, it contains a single-double precision converter.

ParaGraph - a graphical display system for visualizing the behavior of
            parallel algorithms on message-passing multiprocessor architectures.
            Authors: Jennifer Etheridge and Michael Heath, Oak Ridge National
 Lab.
            mth@ornl.gov, 4/19/90.

GMAT -  Mark Seager (LLNL Oct 8, 1987)
    multi-processing Time Line and State Graph tools.

TYPESETTING  troff and LaTeX macros, mostly written at Bell Labs.  Also,
    AMS-TeX macros by Arnold, Lucier, and SIAM.

C++   miscellaneous codes in the C++ language.   At present this includes
    Hansen's C++ Answer Book.


OPT miscellaneous optimization software.  Contains Brent's praxis.
```

```
BIB bibliographies:  Golub and Van Loan, 2nd ed.
```

3. **eLib:** Seit Mai 1990 wird diese neue Softwareblibliothek am ZIB (Konrad-Zuse-Zentrum für Informationstechnik Berlin) angeboten.
   eLib kann zur Zeit über die Datex-P Nummer

   45050 331 033

   erreicht werden. In Zukunft wird eLib über das Internet angesprochen werden können und zwar mit

   telnet eLib.ZIB-Berlin.DE oder
   telnet 130.73.108.11

   und gibt bei Aufforderung zum „login" die Zeichenfolge „elib" an - Achtung: Das „l" muß in diesem Zusammenhang klein geschrieben werden. Danach sollte sich die eLib mit ihrem Logo melden.

   Die folgenden Informationen über eLib wurden mit dem Befehl

   MAIL elib@sc.ZIB-Berlin.dbp.de
   Allgemeines Prospekt; mailit

   eingeholt.

```
Subject: eLib mails '2.1 Prospekt : eine Uebersicht ueber die eLib' - 23766
 Bytes


        DIE ELEKTRONISCHEN SOFTWAREBIBLIOTHEK eLib
        ==========================================
        Prospekt - VORLAEUFIG!  Stand:  03.03.1990


        UEBERSICHT

Seit Anfang des Jahres 1989 arbeitet am  "Konrad-Zuse-Zentrum fuer Infor=
mationstechnik Berlin - ZIB" die Gruppe "Software Information" an der
Entwicklung eines Systems  zum Aufbau und zur Pflege einer "elektronischen
Software Library - eLib" fuer mathematische Algorithmen aus allen Bereichen
des "Scientific Computing". In erster Linie sollen gesammelt und elektronisch
weitergegeben werden:

        Algorithmen und Anwenderprogramme in FORTRAN, C etc. zusammen mit
        ihren Testprogrammen, Testdaten und ihrer Dokumentation. Vorwiegend
        soll es sich dabei um neuere Forschungscodes handeln, die dem
```

"State of the Art" entsprechen.

Nicht nur numerische Codes, sondern z.B. auch Verfahren aus der Computeralgebra oder parallele Algorithmen. Am ZIB gehoeren zum "Scientific Computing" die Arbeitsbereiche Numerik, Numerische Softwareentwicklung, Symbolik, Software Information, Grafik und Systolik.

Abstracts und Referenzen zu den Codes sowie Hinweise auf verwandte Themenbereiche.

Benutzer der eLib sollen ausserdem Gelegenheit dazu erhalten, Ihre eigenen Anwendererfahrungen, Tests und Validierungen als Kommentare zu den Verfahren ebenfalls mit einzubringen.

Testergebnisse zu den bereitgestellten Verfahren, sind auch in grafischer Form (z.B. als Postscript File) willkommen.

Vertriebsbedingungen und Gebuehreninformationen sowie Institutsprofile und Kontaktadressen von den Herstellern und Entwicklern der Software.

Das ZIB kooperiert mit der gemeinsamen Fachgruppe "Numerische Software" der GAMM, der GI und des DMV bei der Errichtung der eLib, insbesondere bei der Sammlung von Codes aus dem deutschen Raum. Im Zusammenhang mit dem eLib-Projekt kooperiert das ZIB mit der bekannten amerikanischen "netlib", insbesondere zur Erschliessung von Software aus dem englischsprachigen Raum.

FUNKTIONSWEISE

Die eLib laesst sich in ihrer Funktion am besten durch die Analogie mit einer "Bibliothek" erklaeren: Neben einem "Katalog" mit mathematischem Register und einem "Lesesaal" fuer Dokumente aus dem Magazin gibt es eine "Leih- bzw. Kopier-Stelle" (technisch per e-Mail und ASCII-Transfer) und - spaeter - "Arbeitsraeume fuer oeffentlichen Erfahrungsaustausch" (conferencing). Ueber einer "Fernleihe" kann man auf Dokumente anderer "Bibliotheken" zugreifen. Jedoch ist im Gegensatz zu einer klassischen Bibliothek der Dienst vollstaendig auf elektronische Medien abgestellt, die einfach und schnell und zu jeder Tages- und Nachtzeit verfuegbar sind. Dokumente sind "selbst" vollstaendig verfuegbar - nicht nur ihre Abstracts.

Programme, Referenzen und Vertriebsunterlagen werden auf dem Wege ueber Methoden der technischen Kommunikation verteilt: Damit der Zugang nicht auf bestimmte Hersteller und Netze eingeschraenkt ist, geschieht dieses in erster Linie mit Hilfe von Kommunikationskomponenten, die mit den international genormten Protokollen der ISO arbeiten und die z.B. auch vom "Deutschen Forschungsnetz - DFN" verwandt werden. Auf diese Weise kann die eLib nicht nur aus dem gesamten DFN, sondern - auf dem Wege ueber Gateways - aus praktisch allen wichtigen Wissenschaftsnetzen erreicht werden.

Innerhalb der eLib werden in einer ersten Ausbaustufe z.Zt. die folgenden technischen Bestandteile der "elektronischen Bibliothek" angeboten:

* ein on-Line Dienst mit nach dem GAMS-Index klassifizierter Software /Boisvert/,
* ein off-Line Dienst fuer eLib-Anfragen per e-Mail,
* e-Mail Ausgabe im Rahmen des on-Line Dienstes und
* ein Fern-Kopier Dienst fuer Algorithmen der netlib /Dongarra/.

I. ON-LINE DIENST

Am schnellsten erreicht man die eLib im Datex-P Dialog. Man waehlt

```
national:             45 050 331 033
international:  + 264 45 050 331 033
```

und ist direkt - on line - mit der eLib verbunden ("Katalog und Lesesaal"). Die eLib verwendet eine einfache Menue-Steuerung im Dialog, mit deren Hilfe man sich in der angebotenen Software und den verfuegbaren Dokumenten unmit= telbar orientieren kann. Menues und Dokumente koennen ueber eine Nummer oder das erste Wort des jeweiligen Textes ausgewaehlt werden. Dabei steigt man immer tiefer in den Menue- und Dokumenten-Baum ein. Das Kommando

```
main
```

stellt die eLib immer auf das Haupt-Menue zurueck. Man kann in Menues und Dokumenten seitenweise blaettern (englisch: "browse"). Am besten gibt man zunaechst nur <return> ein, um weiterzulesen. Wenn man am Ende eines Doku= mentes angelangt ist, schaltet der Befehl "next" in das naechste verfuegbare Menue oder Dokument. Wie man gezielt blaettert, erfaehrt man mit dem Kommando

```
help
```

- das auch sonst nuetzliche Informationen ueber den eLib-Dialog gibt. Unter dem Menuepunkt "General" bzw. "Allgemein" findet man wichtige Information ueber die eLib selbst - z.B. eine allgemeinere Zugangsbeschreibung oder ein Dokument, in dem genauer beschrieben ist, wie die eLib per elektronischer Post angesprochen werden kann.

Zur Unterstuetzung bei der Suche nach mathematischer Software sind die Algo= rithmen nach dem GAMS-Index /Boisvert/ des National Bureau of Standards der USA gegliedert ("Katalog"). Dessen oberste Gliederungspunkte sind:

```
Arithmetic, Error Analysis                         - a
```

```
Number Theory                                          - b
Elementary and special Functions                       - c
Linear Algebra                                         - d
Interpolation                                          - e
Solution of Nonlinear Equations                        - f
Optimization                                           - g
Differentiation and Integration                        - h
Differential and Integral Equations                    - i
Integral Transforms                                    - j
Approximation                                          - k
Statistics and Probability                             - l
Simulation and Stochastic Modelling                    - m
Data Handling                                          - n
Symbolic Computation                                   - o
Computational Geometry                                 - p
Graphics                                               - q
Service Routines                                       - r
Software Development Tools                             - s
```

Im Dialog kann man auch Botschaften an die eLib-Administration versenden. Wir sind auch fuer Anregungen und Verbesserungsvorschlaege dankbar. Zunaechst wird man diesen Dienst aber benutzen, um sich ein Passwort einrichten zu lassen. Manche Dokumente - dazu gehoeren z.B. Lizenzprogramme - kann man nur dann vollstaendig lesen, wenn man ein Passwort dazu hat.

Technisch ist am Anfang ein sogenannter "linemode" fuer das anrufende Terminal eingestellt, den man aber, z.B. mit dem Befehl

```
term vt100     - bei DEC VT100 Terminals -,
```

unmittelbar an die Gegebenheiten seines Terminals anpassen kann. Wer seinen Terminaltyp nicht kennt, kann ihn im Submenue "Set-Up : Adapt Your Terminal to Your Local Needs" des Hauptmenues finden - z.B. mit dem Befehl "find IBM". Das Kommando

```
quit
```

beendet den Dialog mit der eLib. Gleichzeitig wird die Dialog-Verbindung abgebaut.

II. OFF-LINE DIENST

Der off-Line Dienst ist ein X.400 orientierter Mail-Dienst, mit dem man sich alle Leistungen der eLib auch dann erschliessen kann, wenn man z.B. zum direkteren on-Line Dienst keinen Zugang hat. Man addressiert die eLib mit:

```
X.400:          S=eLib;OU=sc;P=ZIB-Berlin;A=dbp;C=de
RFC822:         eLib@sc.ZIB-Berlin.dbp.de
```

```
EARN/BITNET:    eLib@sc.ZIB-Berlin.dbp.de
Internet:       elib@eLib.ZIB-Berlin.de (130.73.108.11)
UUCP:           unido!sc.ZIB-Berlin.dbp.de!eLib
```

und schickt einen elektronischen Brief mit seinen Anforderungen an die eLib.

```
help
```

liefert ein Dokument aus, in dem genauer beschrieben ist, wie man die eLib im Mail-Betrieb anspricht. Bis auf "help", das im off-Line Betrieb als Hilfe-Kommando fuer Mail-Anfragen interpretiert wird (mailhilfe), geht es im Prin= zip genau so, wie im Dialog: d.h. man positioniert durch Auswahl von Sub-Menues oder Dokumenten auf die eigentlich interessante Information und - weil man diese im off-Line Betrieb ja nicht direkt "lesen" kann - muss man nur dafuer sorgen, dass das "angewaehlte" Dokument auch zugesandt wird:

```
mailit
```

verschickt immer das "aktuelle" Menue oder Dokument an den Absender. Das letzte "mailit" kann man sich - der Einfachheit halber - sparen. Auch ein "quit" am Ende einer Anfrage kann man sich sparen. Mit - nur - der Anfrage

```
top             bzw.            main
```

erhaelt man das oberste Menue und damit den Einstieg in die eLib-Information:

```
TOP   ZIB Electronic Library                          L:1 - 18 (100%)  B:864
.............................................................................
    1  Gen-eral         : Introduction into the eLib System
    2  All-gemein       : Einfuehrung  in das eLib System

    3  G-AMS Index      : Access to Software  by Problem Oriented Structure
    4  Lib-rary Index   : Access to Software  Sorted by Libraries
    5  Mod-ule Index    : Access to Software  Sorted by Module Names

    6  Net-lib          : A "hot link" to the famous netlib

    7  Ser-ies Index    : Access to Abstracts and Documents  Sorted by Series
    8  Doc-ument Index  : Access to Abstracts and Documents  Sorted by Authors
    9  Inst-itutes      : Institute Descriptions and Distribution Conditions

   10  Me-ssage         : Message to System Administration
   11  E-ditmenu        : Edit Text in Scratch File

   12  Check-In        : To Get Further Access Rights
   13  Set-Up          : Adapt Your Terminal to Your Local Needs
.............................................................................
```

Im off-Line Betrieb verwendet man am besten nur Texte statt Nummern zur Aus=
wahl: Nummern koennen sich im Rahmen einer Maintenance schneller aendern.
Die Anfrage

```
Allgemein Prospekt;       mailit
all       Zugang;         m
all       'Dialog Hilfe'; m
all       'Mail Hilfe';   m
GAMS;                     m
Library;                   m
Document
protocol
```

erzeugt zum Beispiel 8 Briefe an den Absender, in denen neben dem GAMS-, dem Modul- und dem Dokument-Index die wichtigsten einfuehrenden Dokumente der eLib zurueckgeschickt werden. Der achte Brief besteht aus einem Ablaufprotokoll der ganzen Anfrage, das mit der mail-Anweisung "protocol" angefordert wird.

Das Beispiel zeigt auch, wie man Menue-Titel anwaehlt, in denen etwa ein Blank enthalten ist. Ferner: Gross- und Kleinschreibung spielen fuer die Anwahl keine Rolle. "all" ist die Abkuerzung fuer "Allgemein" und "m" steht fuer "mailit".

Der wichtigste - problemorientierte - Index der vorhandenen Software ist der GAMS-Index: der "Guide of Available Mathematical Software" /Boisvert/. In der Regel reichen einige Briefe, um sich den aktuellen GAMS-Index zu erschliessen. Wer den GAMS-Index bereits kennt, kann die Suche nach Software, die fuer sein Problem geeignet ist, stark abkuerzen. Die Anfrage

```
gams i
```

liefert z.B. eine Uebersicht ueber das Kapitel "Differential and Integral Equations - i" aus dem GAMS-Index. Abschnitte, zu denen es in der eLib Software gibt, sind im Menu durch eine laufende Nummer (zur Menue Auswahl) gekennzeichnet. Eine Mail mit der - einzigen - Anfrage

```
gams i1a2b;  LIMEX
```

positioniert zunaechst auf den entsprechenden Abschnitt "i1a2b" des GAMS-Index, selektiert dann das Programm LIMEX und schickt den zugaenglichen Teil an den Sender des Briefes zurueck.

III. E-MAIL AUSGABE IM ON-LINE DIENST

Eingeschriebene Benutzer sollen die im Dialog gefundene Software unmittelbar per e-mail (X.400) an ihr oertliches System senden koennen. ("Leih- bzw. Kopierstelle"). Um sicherzustellen, dass dieser Dienst nur von berechtigten Benutzern (und nicht von jedem "anonymen" Benutzer) in Anspruch genommen werden kann, sind zwei einfache Schritte erforderlich:

1. Lassen Sie sich von der eLib-Administration einen "Benutzer Namen" und ein entsprechendes Passwort eintragen. Dafuer sollte uns zumindest Ihre Post-Adresse bekannt sein bzw. die Anschrift Ihres Instituts.

2. Registrieren Sie anschliessend Ihr Mail-System bei der eLib, indem Sie in eine Ihrer e-Mail Anfragen die Kommandofolge

   password <user-name> <actual-password>
   register

   einfuegen. Die eLib wird Ihnen dann kurz bestaetigen, dass Sie von nun an (per "mailit") im Dialog auch Mail an sich versenden duerfen.

Wenn Sie Ihre Mailbox einmal gewechselt haben, brauchen Sie nur die o.g. register-Befehlsfolge erneut an die eLib abzusenden.

IV. FERN-KOPIER DIENST

Mit dem Fern-Kopier Dienst ("Fernleihe") soll man sich Software anderer Netz-Bibliotheken kostenguenstig beschaffen koennen. Konkret hat das ZIB hier eine Zusammenarbeit mit der "netlib" der AT&T Bell Laboratorien in New Jersey vereinbart und einen europaeischen netlib-Knoten am ZIB realisiert. Man addressiert die netlib am ZIB mit

X.400:	S=netlib;OU=sc;P=ZIB-Berlin;A=dbp;C=de
RFC822:	netlib@sc.ZIB-Berlin.dbp.de
EARN/BITNET:	netlib@sc.ZIB-Berlin.dbp.de
Internet:	netlib@netlib.ZIB-Berlin.de
UUCP:	unido!sc.ZIB-Berlin.dbp.de!netlib

und schickt eine e-Mail mit seinen Anforderungen. Mittels der Anfrage

    send index

erhaelt man von der netlib am ZIB eine einfache Gebrauchsanweisung und die wichtigsten Uebersichten bzw. Indices der netlib. Hier sind Beispiele der verschiedenen Arten von Anforderungen:

* schicke den vollstaendigen Index einer Software Bibliothek:
  send index from eispack
* schicke eine spezielle Routine - und alles, wovon sie abhaengt:
  send dgeco from linpack
* schicke nur genau eine Routine - und keine davon abhaengigen Moduln:
  send only dgeco from linpack
* schicke Routine mit Abhaengigkeits-Baum - aber schliesse Unter-Baum aus:
  send dgeco but not dgefa from linpack
* teile nur mit, wie umfangreich eine Antwort ist - schicke noch keine Datei:
  send list of dgeco from linpack

```
*  suche nach jemandem im SIAM Mitglieder-Verzeichnis:
        who is gene golub
*  Schluesselwort-Suche nach netlib-Software:
        find cubic spline
*  Schluesselwort-Suche im Approximations-Katalog:
        find schumaker from approximation
*  Bemerkungen zum Fortran-nach-C Konverter (verfuegbar bei AT&T)
        send index for f2c
```

Die netlib ist am ZIB technisch als "Cache" implementiert: Routinen, die nicht bereits vorliegen, werden aus den Staaten geholt. Anfragen nach Routinen, die bereits hier vorliegen, koennen dem entsprechend sofort - und damit recht schnell - beantwortet werden. Alle anderen Routinen muessen erst von der originalen netlib bei AT&T geholt ("entliehen") werden.

Der Transfer solcher Software aus den Staaten nach Deutschland braucht auf diese Weise nur einmal fuer jede neue Version durchgefuehrt zu werden.

V. SPEZIELLER ZUGANG: LOKAL UND TELEFON-MODEM

Oertliche Benutzer der Berliner Universitaeten und Forschungsanstalten koennen kostenguenstig auch auf dem Wege ueber lokale Netze (Ethernet) und Telefon-Leitungen (Modems) auf die eLib zugreifen. Den off-Line Dienst spricht man lokal wie folgt an:

```
        Mail-Adresse:    eLib                  und Angabe von eLib-Befehlen
                                               im Subject- oder im Textteil.
```

Den on-Line Dienst kann man im "lokalen" Dialog per

```
        telnet eLib                            und Eingabe von
        elib                                   nach der Aufforderung zum login
```

erreichen. Zusaetzlich fuehren drei Modem-Anschluesse des ZIB in das lokale Netz am ZIB:

```
        (030) 89604 270                        ein 1200 Baud E-7-1 Anschluss,
        (030) 89604 271                        ein 1200 Baud E-7-1 Anschluss,
        (030) 89140 72                         ein  300 Baud E-7-1 Anschluss.
```

Der Protokollkonverter meldet sich anschliessend mit einem Menue, mit dem der Typ des anwaehlenden Terminals zu bestimmen ist. Hier waehlt man etwa

```
        5                                      fuer ein DEC VT100 Terminal.
```

Anschliessend fragt der Protokollkonverter nach, mit welchem Sub-Netz im ZIB man verbunden werden will. Hier fuehrt die Angabe von

```
        2                                      zu einem PAD und
```

c 333 stellt dort die

gewuenschte Verbindung zum eLib-System her. Groessere Datenmengen wird man sinnvollerweise nur per elektronischer Post uebertragen. Man braucht dazu einen e-Mail Zugang zur eLib.

Kleinere Datenmengen lassen sich - auch mit einem Telfon Modem - im Rahmen eines einfachen ASCII Transfers (Kommando "list") uebermitteln. Hierbei handelt es sich z.Zt. nur um einen ungesicherten Transfer.

NUTZUNGSGEBUEHREN

- Die eLib ist noch NICHT allgemein FREIGEGEBEN!

Zur Zeit sind eLib und netlib nur einer eingeschraenkten Benutzergemeinde zur Erprobung freigeben. In erster Linie ist hier an Kooperationspartner des ZIB gedacht und an potentiell Beitragende, wie z.B. die Mitglieder der Fachgruppe "Numerische Software" von GAMM, GI und DMV. Die allgemeine Freigabe ist erst zum April 1990 geplant. Ab diesem Zeitpunkt sollen netlib und eLib zu folgenden Konditionen bereitgestellt werden.

1. Die netlib wird am ZIB zu denselben Bedingungen betrieben, wie die originale netlib bei AT&T, naemlich kostenfrei. Ueber die netlib ist grundsaetzlich nur "Public Domain Software" verfuegbar, fuer die keinerlei Garantie uebernommen werden kann.

2. "Public Domain Software" wird auch in der eLib - ohne irgendwelche Garantien - kostenfrei und ohne zusaetzliche Uebertragungsgebuehren weitergegeben. Die Inanspruchnahme der eLib fuer den Versand von allgemeinen Informationen ueber die eLib selbst und die hier anbietenden Institutionen ist ebenfalls kostenfrei.

3. Innerhalb der eLib gibt es geschuetzte Software oder Lizenz= produkte, die nur mit einem an den jeweiligen Benutzer gebundenen Passwort erreicht werden koennen.

Fuer Software dieser Art (z.B. die CodeLib des ZIB) sind dann auch die ent= sprechenden Dokumente verfuegbar (z.B. "Lizenzbedingungen" und "Preisliste"), die die Bezugsmodalitaeten beschreiben. Nutzer dieser Software muessen in der Regel mit dem Hersteller bzw. jeweiligen Entwickler einen entsprechenden Lizenz- oder Nutzungsvertrag abgeschlossen haben.

WEITERER AUSBAU DER eLIB

Mit dem weiteren Ausbau der eLib sollen die folgenden Ziele verfolgt werden: Es sollen Ordnungs-Schemata fuer die Bereitstellung von Testlaeufen, Testdaten, Testergebnissen und Kommentaren bzw. Validierungen dazu entwickelt und erprobt werden. Mit Test-Files sind nicht nur die normalerweise recht kleinen

Testprogramme gemeint, sondern auch umfangreichere Ausgaben, wie etwa "Bilder" oder andere graphische Informationen.

Informationen dieser Art lassen sich in der Regel nicht mehr in einem ein=fachen hierarchischen Schema - wie z.B. dem GAMS-Index - fassen. Bei Test-Daten ist z.B. an eine "zeitliche Entwicklung" zu denken. Neuere Ergebnisse muessen mit aelteren vergleichbar sein. Ferner sollen Test-Verfahren auch "von aussen" erprobt und etwa mit eigenen Test-Funktionen beschickt werden koennen.

Zum heutigen Zeitpunkt wird deshalb an den Ausbau der eLib in folgenden Schritten gedacht.

VI. SAMMLUNG VON SOFTWARE - ab Januar 1990

Zunaechst soll die "Public Domain Software" der netlib in die eLib integriert werden. In Ergaenzung zur netlib gibt es zu dieser Software dann:

* eine Klassifizierung nach dem GAMS-Index, die auch interaktiv zum Einsatz gebracht werden kann;

* einen on-Line Zugriff direkt auf die Software, die damit unmittel=bar erschlossen wird;

* weitere off-line Verbindungen aus Wissenschaftsnetzen, insbesondere im europaeischen Raum (X.400-Mail and DFN-Gateways).

Parallel dazu soll ueber die Fachgruppe "Numerische Software" von DMV, GAMM und GI innerhalb Deutschlands Software gesammelt und erschlossen werden. Im Kontrast und in Ergaenzung zur netlib sollen dabei auch:

* lizensierte oder anderweitig geschuetzte Programme oder

* Algorithmen aus der "Computer Algebra" und aus anderen Bereichen des Scientific Computing.

in die eLib aufgenommen werden.

VII. KONFERENZ-DIENST - geplant fuer 3. Quartal 1990

Eingeschriebene Anwender und Autoren sollen ihre Kommentare zu den Algorithmen - dazu gehoeren auch Rechenergebnisse und Bewertungen - im Rahmen von oeffent=lich zugaenglichen "elektronischen Konferenzen" hinterlegen koennen ("Diskus=sionsraeume"). Konferenzen sind in diesem Zusammenhang nicht als on-Line-Treffen (im Sinne eines sogenannten "chat-Systems") zu verstehen, sondern als problemorientiert strukturierte Bulletin-Boards in der Art des PortaCOM Systems der Universitaet Stockholm /QZ/ oder des CoSy Systems der Universitaet von Guelph /GUELPH/.

VIII. FILETRANSFER

In einer weiteren Ausbaustufe soll die eLib mit einem Filetransfer-Dienst (FTAM) fuer die zuverlaessige Uebertragung auch groesserer Programmsysteme und Dateien (z.B. fuer Bildinformation) ausgestattet werden. Dabei soll die Software-Suche durch einen problemorientierten Dialog bzw. Fileserver unter= stuetzt werden.

IX. PRUEFLABOR

Ferner ist an ein "Prueflabor" fuer ausgewaehlte Verfahren gedacht: Zur Realisierungs-Technik koennen zu diesem Zeitpunkt noch keine genaueren Angaben gemacht werden. Mathematische Algorithmen, die nicht mit ihrer ganzen Quelle oeffentlich zur Verfuegung stehen (z.B. Lizenzprodukte), sollen dennoch allgemein getestet und geprueft werden koennen. Die eLib soll dazu mit Datensaetzen oder Prueffunktionen "von aussen" beschickt werden koennen.

KONTAKTADRESSE

Interessenten an einem ersten eLib-Test und am eLib-Konzept wenden sich bitte an die Abteilung "Software Information" im ZIB:

Joachim Luegger, Wolfgang Dalitz

X.400: S=Luegger;OU=sc;P=ZIB-Berlin;A=dbp;C=de
RFC822: Luegger@sc.ZIB-Berlin.dbp.de

Konrad-Zuse-Zentrum fuer Informationstechnik Berlin - ZIB
Heilbronner Str.10, D-1000 Berlin 31, Germany

LITERATUR

/Boisvert/ R.F. Boisvert, S.E. Howe, D.K. Kahaner
GAMS: A Framework for the Management of Scientific Software
ACM TOMS, Vol. 11, No. 4, Dec. 1985, pp 313-355

/Dongarra/ J.J. Dongarra, E. Grosse
Distribution of Mathematical Software via Electronic Mail
Comm. ACM Vol 30, No 5, pp 403-407

/DFN/ Am Deutschen Forschungsnetz angeschlossene Institutionen
DFN Mitteilungen, jeweils im neuesten Heft

/GUELPH/ CoSy Computer Conferencing System Reference Manual
University of Guelph, Ontario, Dez. 1987

```
/QZ/            PortaCOM Computer Conferencing System
                Stockholms University Computing Center, QZ 1987
```

4. **Datenbanken:** Es ist möglich, z.B. bei der Universitätsbibliothek, für etwa 20 DM über Datenbanken mathematische Literatur anzufragen. Durch die Eingabe von Schlüsselwörtern, Verfassernamen usw. werden mehrere hunderttausende von Artikeln gezielt durchsucht, um zutreffende Artikel oder Bücher zu finden.

   Mehrere Referatenorgane sind jetzt automatisiert worden:

   (a) Mathematical Reviews und Current Mathematical Publications sind als eine Datenbank **Math Sci** verfügbar. Math Sci ist auch auf CD-ROM verfügbar.

   (b) Das Zentralblatt der Mathematik ist als eine Datenbank **MATH** verfügbar, und zwar vom Fachinformationszentrum Karlsruhe.

Kapitel 3

Das Gaußsche Eliminationsverfahren

3.1 Einleitung

Zu betrachten ist das lineare Gleichungssystem $Ax = b$, wobei $A = (a_{ij})$ eine reelle reguläre $n \times n$ Matrix, $x = (x_i)$ ein reeller n-Vektor und $b = (b_i)$ ein reeller n-Vektor ist. Nachdem A regulär ist, ist die Lösung x eindeutig, und es gilt

$$x = A^{-1}b \; .$$

Obwohl das Problem vom mathematischen Standpunkt aus sehr einfach ist, ist es vom numerischen Standpunkt aus gesehen höchst wichtig, da sehr viele Probleme die Lösung eines linearen Gleichungssystems erfordern. Einige solcher Probleme sind:

1. Spline-Interpolation
2. Das Newton-Verfahren für nichtlineare Gleichungssysteme im $\mathbb{R}^n$
3. Ausgleichsrechnung

Bei der numerischen Lösung von gewöhnlichen und partiellen Differentialgleichungen sowie bei der numerischen Lösung von anderen Gleichungen, wie z.B. Integralgleichungen, entstehen auch sehr große lineare Gleichungssysteme, die die Rechenkapazität jetziger Rechner überfordern.

Zur Zeit (1990) werden in der Praxis Probleme mit $n = 40.000$ (vollbesetzte) Matrizen und $n = 10^8$ (dünnbesetzte Matrizen) gelöst.

Zur rechnerischen Bestimmung des Lösungsvektors x gibt es grundsätzlich zwei Typen von Verfahren: *direkte* und *iterative* Verfahren.

1. Ein *direktes Verfahren* liefert die exakte Lösung nach endlich vielen exakten arithmetischen Operationen.

2. Ein *iteratives Verfahren* erzeugt eine Folge $\{x^k\}$ von Näherungslösungen.

Bei nichtlinearen Gleichungssystemen sind iterative Verfahren erforderlich, da i.a. direkte Verfahren nicht möglich sind. Bei linearen Gleichungssystemen dagegen sind gute direkte Verfahren bekannt, und es darf gefragt werden, ob es überhaupt einen Sinn hat, iterative Verfahren zu betrachten. Die Antwort auf diese Frage ist, daß bei linearen Gleichungssystemen mit speziellen Strukturen, die z.B. oft bei der Lösung von Differentialgleichungen auftreten, iterative Verfahren manchmal eine Näherungslösung mit der gewünschten Genauigkeit mit wesentlich geringerem Rechenaufwand liefern können als der Rechenaufwand, der mit einem direkten Verfahren erforderlich wäre.

In diesem Abschnitt wird das bekannteste direkte Verfahren, das Gaußsche Eliminationsverfahren, ausführlich dargestellt. Spätere Abschnitte werden anderen direkten und iterativen Verfahren gewidmet.

3.2 Gauß-Elimination ohne Pivotsuche

Das Verfahren besteht aus zwei Teilen:

Teil 1: Vorwärtseinsetzen (Elimination)

Im ersten Schritt subtrahiert man ein geeignetes Vielfaches der ersten Gleichung von den übrigen Gleichungen derart, daß die Koeffizienten von x_1 in diesen Gleichungen verschwinden. Durch Wiederholung dieses Schritts werden die Gleichungen zur gestaffelten Form gebracht:

$$Rx = y$$

Beispiel 1

$$\begin{array}{c|rcrcrcr}
\text{Eliminationsfaktoren} & & & & & & & \\
 & 3x_1 & + & x_2 & + & 6x_3 & = & 23 \\
(2/3) & 2x_1 & + & x_2 & + & 3x_3 & = & 13 \\
(1/3) & x_1 & + & x_2 & + & x_3 & = & 6
\end{array}$$

1. Schritt
Wird 2/3-mal die erste Gleichung von der zweiten Gleichung und 1/3-mal die erste Gleichung von der dritten Gleichung subtrahiert, erhält man:

$$\begin{array}{c|rcrcrcr}
\text{Eliminationsfaktoren} & & & & & & & \\
 & 3\cdot x_1 & + & 1\cdot x_2 & + & 6\cdot x_3 & = & 23 \\
 & 0\cdot x_1 & + & 1/3\cdot x_2 & - & 1\cdot x_3 & = & -2\,\frac{1}{3} \\
(2) & 0\cdot x_1 & + & 2/3\cdot x_2 & - & 1\cdot x_3 & = & -1\,\frac{2}{3}
\end{array}$$

2. Schritt
Wird nun zweimal die zweite Gleichung von der dritten Gleichung subtrahiert, erhält

man das gestaffelte System:

$$\begin{array}{rcrcrcr} 3x_1 & + & 1 \cdot x_2 & + & 6 \cdot x_3 & = & 23 \\ & & \frac{1}{3} \cdot x_2 & - & 1 \cdot x_3 & = & -2\,\frac{1}{3} \\ & & & & 1 \cdot x_3 & = & 3 \end{array}$$

d.h.

$$Rx = y$$

mit

$$R = \begin{pmatrix} 3 & 1 & 6 \\ 0 & 1/3 & -1 \\ 0 & 0 & 1 \end{pmatrix}, \; y = \begin{pmatrix} 23 \\ -2\frac{1}{3} \\ 3 \end{pmatrix}.$$

Teil 2: Rückwärtseinsetzen

Die letzte Gleichung des gestaffelten Systems $Rx = y$ enthält nur eine Unbekannte x_n und wird für x_n gelöst. Nachdem x_n bekannt ist, kann x_{n-1} aus der vorletzten Gleichung berechnet werden. Die Unbekannten x_i können deshalb in umgekehrter Reihenfolge bestimmt werden.

Beispiel 1: Fortsetzung

$$\begin{array}{lll} \text{1. Schritt}: & 1 \cdot x_3 & = 3 \\ & \Rightarrow x_3 & = 3/1 = 3 \\ \text{2. Schritt}: & \frac{1}{3}\,x_2 - 1 \cdot x_3 & = -2\,\frac{1}{3} \\ & \Rightarrow x_2 & = (-2\,\frac{1}{3} + 1 \cdot x_3)/(1/3) \\ & & = (-2\,\frac{1}{3} + 3)/(1/3) \\ & & = 2 \\ \text{3. Schritt}: & 3x_1 + x_2 + 6x_3 & = 23 \\ & \Rightarrow x_1 & = (23 - x_2 - 6x_3)/3 \\ & & = (23 - 2 - 18)/3 \\ & & = 1 \end{array}$$

Der Algorithmus wird jetzt als ein Pascal Programm implementiert und auf Beispiel 1 angewandt.

```
type gauss1.pas
program gauss(input,output);
var a:array[1..20,1..21]of real;
var b:array[1..20]of real;
var w,x:array[1..21]of real;
var r:real;
var i,j,k,n:integer;
procedure daten;
begin
```

```
n:=3;
a[1,1]:=3;a[1,2]:=1;a[1,3]:=6;b[1]:=23;
a[2,1]:=2;a[2,2]:=1;a[2,3].=3;b[2]:=13;
a[3,1]:=1;a[3,2]:=1;a[3,3]:=1;b[3]:=6;
for i:=1 to n do a[i,n+1]:=b[i];
end;
procedure elim;
(*Dieses Programm fuehrt Gauss-elimination aus ohne
Pivotsuche *)
begin
for k:=1 to n-1 do
   begin
   for j:=k+1 to n+1 do w[j]:=a[k,j];
   for i:=k+1 to n do
      begin
       r:=a[i,k]/a[k,k];
       a[i,k]:=r;
       for j:=k+1 to n+1 do
          a[i,j]:=a[i,j]-r*w[j];
      end;
   end;
end;
procedure rueckwaerts;
begin
for i:=n down to 1 do
begin
  x[i]:=a[i,n+1];
  for j:=i+1 to n do
    begin
      x[i]:=x[i]-a[i,j]*x[j];
    end;
  x[i]:=x[i]/a[i,i];
end;
end;
procedure druck;
begin
for i:=1 to n do
begin
for j:=1 to n+1 do write(a[i,j]);
writeln;
end;
end;
procedure druckx;
begin
writeln('Die Loesung x');
```

```
for i:=1 to n do
write(x[i]);
writeln;
end;
(*hauptprogramm *)
begin
writeln('Dieses Programm fuehrt Gauss-elimination aus');
writeln('ohne Pivotsuche');
daten;
druck;
elim;
druck;
rueckwaerts;
druckx;
end.
```

Dieses Programm führt Gauß-Elimination aus ohne Pivotsuche.

3.0000000000E+00	1.0000000000E+00	6.0000000000E+00	2.3000000000E+01
2.0000000000E+00	1.0000000000E+00	3.0000000000E+00	1.3000000000E+01
1.0000000000E+00	1.0000000000E+00	1.0000000000E+00	6.0000000000E+00
3.0000000000E+00	1.0000000000E+00	6.0000000000E+00	2.3000000000E+01
6.6666666667E-01	3.3333333333E-01	-1.0000000000E+00	-2.3333333333E+00
3.3333333333E-01	2.0000000000E+00	9.9999999999E-01	3.0000000000E+00
Die Lösung x			
1.0000000000E+00	2.0000000000E+00	3.0000000000E+00	

Tabelle 3.1: Die Lösung von Beispiel 1 mit dem Programm Gauss1 (auf einem Rainbow Rechner)

Das Gaußsche Verfahren ist in China schon ca. 250 v. Chr. für $n = 3$ bekannt gewesen und wurde auch von Lagrange für homogene Gleichungen im Jahre 1778 vorgeschlagen.

Das Gaußsche Verfahren ohne Pivotsuche kann öfter angewandt werden. Für viele Matrizen ist das Verfahren allerdings bösartig und muß geändert werden. Eine stabile Version, das Gaußsche Eliminations-Verfahren mit Pivotsuche, wird später ausführlich diskutiert.

3.3 Blockmatrizen

Oft ist es zweckmäßig, eine $m \times n$-Matrix in Blöcke zu zerlegen:

$$A = \begin{pmatrix} A_{11} & \dots & A_{1q} \\ \vdots & & \\ A_{p1} & \dots & A_{pq} \end{pmatrix}$$

wobei A_{ij} eine $(r_i \times s_j)$-Matrix ist und

$$\sum_{i=1}^{p} r_i = m \quad , \quad \sum_{j=1}^{q} s_j = n \ .$$

Eine Matrix, die so dargestellt wird, heißt eine *partitionierte Matrix.* Zum Beispiel

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} = \left(\begin{array}{c|ccc} 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{array}\right)$$

wobei $p = q = 2$, $r_1 = 1$, $r_2 = 2$, $s_1 = 1$, $s_2 = 3$

$$A_{11} = (1), \quad A_{12} = (2\ 3\ 4) A_{21} = \begin{pmatrix} 5 \\ 9 \end{pmatrix}, \quad A_{22} = \begin{pmatrix} 6 & 7 & 8 \\ 10 & 11 & 12 \end{pmatrix}$$

Hilfssatz 3.1 *Seien A und B $m \times n$-Matrizen,*

$$\sum_{i=1}^{p} r_i = m \quad , \quad \sum_{j=1}^{q} s_j = n \ ,$$

und

$$A = (A_{ij}) \quad , \quad B = (B_{ij}) \ ,$$

wobei A_{ij} und B_{ij} $(r_i \times s_j)$-Matrizen sind. Dann gilt

$$A + B = (C_{ij}) := (A_{ij} + B_{ij})$$

Beweis: Direkte Überprüfung. □

Hilfssatz 3.2 *Sei A eine $(m \times n)$-Matrix und B eine $(n \times o)$-Matrix. Sei $r_i, s_j, t_k \in \mathbb{N}$,*

$$\sum_{i=1}^{u} r_i = m \quad , \quad \sum_{j=1}^{v} s_j = n \quad , \quad \sum_{k=1}^{w} t_k = o \ ,$$

$$\begin{array}{lllll} A = (A_{ij}) & , & 1 \le i \le u & , & 1 \le j \le v & , & A_{ij} \in \mathit{Mat}(r_i \times s_j) \\ B = (B_{jk}) & , & 1 \le j \le v & , & 1 \le k \le w & , & B_{jk} \in \mathit{Mat}(s_j \times t_k) \end{array}$$

Dann gilt:

$$C := AB = (C_{ik}) \quad , \quad 1 \le i \le u \quad , \quad 1 \le k \le w \ ,$$

mit

$$C_{ik} := \sum_{j=1}^{v} A_{ij} B_{jk} \ ,$$

so daß C_{ik} eine $(r_i \times t_k)$-Matrix ist.

Beweis: Direkte Überprüfung. □

Partitionierte Matrizen A sind oft sehr nützlich:

1. Bei der Darstellung und Anwendung von Algorithmen.
2. Um die Struktur einer Matrix darzustellen.

Eine partitionierte $(n \times n)$-Matrix A mit der Gestalt

$$A = \begin{pmatrix} A_{11} & & O \\ & \ddots & \\ O & & A_{pp} \end{pmatrix}$$

heißt *blockdiagonal*, eine partitionierte $(n \times n)$- Matrix A mit der Gestalt

$$A = \begin{pmatrix} A_{11} & & O \\ \vdots & \ddots & \\ A_{1p} & \dots & A_{pp} \end{pmatrix}$$

heißt *linke Blockmatrix.*

Matrizen mit solchen Strukturen treten in verschiedenen Anwendungsgebieten auf:

1. Bei der numerischen Lösung gewöhnlicher und partieller Differentialgleichungen.
2. Bei der Modellierung der Wirtschaft, der Elektrizitätsversorgung, usw.

3.4 Dreiecksmatrizen

Definition 3.1 *Sei $A = (a_{ij})$ eine $n \times n$-Matrix. A heißt* linke *oder* untere Dreiecksmatrix*, falls $a_{ij} = 0$ für $j > i$. A heißt* rechte *oder* obere Dreiecksmatrix*, falls $a_{ij} = 0$ für $i < j$.*

Satz 3.1
1. *Das Produkt zweier linker Dreiecksmatrizen ist wieder eine linke Dreiecksmatrix.*
2. *Sei L eine reguläre linke Dreiecksmatrix. Dann ist L^{-1}, die Inverse von L, eine linke Dreiecksmatrix.*

Entsprechende Eigenschaften gelten für rechte Dreiecksmatrizen.

Beweis:

1. $L = (\ell_{ij})$ und $M = (m_{ij})$ seien untere $n \times n$- Dreiecksmatrizen, d.h.

$$\ell_{ij} = m_{ij} = 0 \quad \text{für} \quad j > i \ .$$

Sei $P := LM = (p_{ij})$. Dann gilt:

$$p_{ij} = \sum_{k=1}^{n} \ell_{ik} m_{kj} \ .$$

Ist $j > i$ und $1 \leq k \leq n$, dann ist entweder

$$j > k \Rightarrow m_{kj} = 0$$

oder

$$j \leq k \Rightarrow i < j \leq k \Rightarrow k > i \Rightarrow \ell_{ik} = 0 \ .$$

Es folgt, daß $p_{ij} = 0$ für $j > i$, d.h. P ist eine untere Dreiecksmatrix.

2. Der Beweis benutzt die vollständige Induktion für die Aussage $A(n)$: Der Satz, Teil 2, gilt für $m \times m$-Dreiecksmatrizen, $m \leq n$.
$A(1)$ ist offenbar richtig. Es sei $A(n)$ richtig für $n \in \mathbb{N}$. L sei eine reguläre $(n+1) \times (n+1)$ linke Dreiecksmatrix. Ohne Einschränkung gilt:

$$L = \left(\begin{array}{c|c} M & 0 \\ \hline a^T & \alpha \end{array} \right)$$

wobei $a \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^1$, $\alpha \neq 0$, und M eine reguläre $n \times n$ linke Dreiecksmatrix ist.
Wegen der Richtigkeit von $A(n)$ folgt, daß M^{-1} eine linke Dreiecksmatrix ist.
Es läßt sich durch Berechnung bestätigen, daß

$$L^{-1} = \begin{pmatrix} M^{-1} & 0 \\ -\frac{1}{\alpha} a^T M^{-1} & 1/\alpha \end{pmatrix}$$

Also ist auch $A(n+1)$ richtig.

$\square$

Eine linke oder rechte Dreiecksmatrix A heißt *normiert*, falls $a_{ii} = 1$ für $1 \leq i \leq n$, so daß nur Einsen in der Diagonale stehen.

Hilfssatz 3.3 *Seien L_1, L_2 und L_3 linke Dreiecksmatrizen mit Diagonalen D_1, D_2 und D_3. Sei $L_1 L_2 = L_3$. Dann gilt:*

$$D_1 D_2 = D_3 \ .$$

Satz 3.2 *Die Zerlegung einer nicht-singulären $n \times n$-Matrix A in das Produkt*

$$A = LDR$$

einer normierten linken Dreiecksmatrix L, einer Diagonalmatrix D und einer normierten rechten Dreiecksmatrix R ist eindeutig, wenn sie existiert.

Beweis: A sei eine nicht-singuläre $n \times n$-Matrix. Es sei

$$A = L_1 D_1 R_1 = L_2 D_2 R_2. \tag{3.1}$$

Da A regulär ist, sind L_1, D_1, R_1, L_2, D_2 und R_2 regulär. Multipliziere (3.1) mit L_2^{-1} (links) und R_1^{-1} (rechts). Es folgt

$$\hat{L} = \hat{R} \,,$$

wobei

$$\hat{L} := L_2^{-1} L_1 D_1$$

eine linke Dreiecksmatrix und

$$\hat{R} := D_2 R_2 R_1^{-1}$$

eine rechte Dreiecksmatrix ist. Es gilt also

$$\hat{L} = \hat{R} = \hat{D} \,,$$

wobei $\hat{D}$ eine Diagonalmatrix ist. Es folgt aus

$$\hat{L} = L_2^{-1} L_1 D_1 = \hat{D} \,,$$

daß

$$L_1 = L_2 \hat{D} D_1^{-1} = L_2 \tilde{D} \,,$$

wobei $\tilde{D} := \hat{D} D_1^{-1}$ eine Diagonalmatrix ist. Aus der Bedingung, daß L_1 und L_2 normiert sind, ergibt sich aus Hilfssatz 3.3, daß

$$D_1 = D_2 \tilde{D} \Rightarrow \tilde{D} = E \quad \text{und deshalb} \quad L_1 = L_2 \,.$$

Die Beziehungen $R_2 = R_1$, $D_1 = D_2$ folgen sofort. □

3.5 Elementare Matrizen

Definition 3.2

$$e_k := (\overbrace{0, 0, \ldots, 0}^{k-1}, 1 \,,\, \overbrace{0, 0, \ldots, 0}^{n-k})^T$$

heißt der k-te Achsenvektor.

Definition 3.3 *Sei $A = (a_{ij})$ eine $n \times n$-Matrix. A heißt* elementar, *falls A eine der folgenden Gestalten hat:*

1. $A = S_j := I - s e_j^T \ , \ s^T e_j = 0 \ , \ s \in \mathbb{R}^n$, *d.h.*

$$A = S_j = \begin{array}{c} j\text{-te Spalte} \\ \downarrow \\ \begin{pmatrix} 1 & & -s_1 & & \\ & 1 & \vdots & & O \\ & & 1 & & \\ & O & \vdots & 1 & \\ & & -s_n & & 1 \end{pmatrix} \end{array}$$

2. $A = Z_i := I - e_i z^T \ , \ z^T e_i = 0 \ , \ z \in \mathbb{R}^n$, *d.h.*

$$A = Z_i = \begin{array}{cc} \begin{pmatrix} & 1 & & & \\ & & 1 & & \\ -z_1 & \ldots & 1 & \ldots & -z_n \\ & & & 1 & \\ & & & & 1 \end{pmatrix} & \begin{array}{c} \\ \\ \leftarrow \quad i\text{-te Zeile} \\ \\ \\ \end{array} \end{array}$$

3. $A = L_j = I - \ell e_j^T \ , \ \ell^T e_k = 0$ *für* $k \leq j \ , \ \ell \in \mathbb{R}^n$,

$$L_j = \begin{array}{c} j\text{-te Spalte} \\ \downarrow \\ \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & -\ell_{j+1} & 1 & \\ & & -\ell_n & & 1 \end{pmatrix} \end{array}$$

L_j *heißt* elementare linke Dreiecksmatrix.

4. $A = R_i := I - e_i r^T \ , \ r^T e_k = 0$ *für* $k \leq i \ , \ r \in \mathbb{R}^n$,

$$R_i = \begin{array}{cc} \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & -r_{i+1} \ldots & -r_n \\ & & & 1 & \\ & & & & 1 \end{pmatrix} & \begin{array}{c} \\ \\ \leftarrow i - te Zeile \\ \\ \\ \end{array} \end{array}$$

R_i *heißt* elementare rechte Dreiecksmatrix.

5. $A = P_\sigma := (p_{ij})$, *wobei* $p_{i\sigma_i} = 1$ *für* $1 \leq i \leq n$ *und* $p_{ij} = 0$ *sonst und* $(\sigma_1, \ldots, \sigma_n)$ *eine Permutation von* $(1, \ldots, n)$ *ist.* σ *ist also eine eineindeutige (bijektive) Abbildung von der Menge* $M := \{1, 2, \ldots, n\}$ *auf* M *:*

(a) $\sigma : i \to \sigma_i$,

(b) $\sigma_i \neq \sigma_j$, $i \neq j$

(c) Für $k \in M$ *existiert* $i : \sigma_i = k$.

Die Menge aller Permutationen von $\{1, \ldots, n\}$ *bildet eine Gruppe der Ordnung* $n!$, *die symmetrische Gruppe* S_n.

Es gilt:

$$P_\sigma = \begin{pmatrix} e_{\sigma_1}^T \\ \vdots \\ e_{\sigma_n}^T \end{pmatrix}$$

P_σ *heißt* Permutationsmatrix.

Eine Permutation σ *wird oft tabellarisch angegeben:*

$$\sigma = \begin{pmatrix} 1 & 2 \ldots n \\ \sigma_1 & \sigma_2 \ldots \sigma_n \end{pmatrix}$$

Beispiel: $n = 3$

$$\sigma := \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

$$P_\sigma = \begin{pmatrix} e_3^T \\ e_1^T \\ e_2^T \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

6. $A = P_{rs}$, *wobei*

$$a_{ij} = \begin{cases} 1 & , \textit{ falls } \quad i = j \quad \textit{und} \quad i \neq r \quad \textit{und} \quad i \neq s \\ 1 & , \textit{ falls } \quad i = r \quad \textit{und} \quad j = s \\ 1 & , \textit{ falls } \quad i = s \quad \textit{und} \quad j = r \\ 0 & , \textit{ sonst.} \end{cases}$$

P_{rs} *ist eine Permutationsmatrix* P_σ *mit*

$$\sigma = \begin{pmatrix} 1 & 2 \ldots r \ldots s \ldots n \\ 1 & 2 \ldots s \ldots r \ldots n \end{pmatrix} ,$$

d.h.

$$\begin{aligned} \sigma_i &= i \quad \textit{für} \quad i \neq r, s \, , \\ \sigma_r &= s \, , \\ \sigma_s &= r \, . \end{aligned}$$

(σ *heißt* Transposition.)

Rechenregeln für elementare linke Dreiecksmatrizen:

Sei

$$A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \quad \text{mit} \quad a_i = (a_{i1}, \dots, a_{in}) \in \mathbb{R}^n ,$$

$$L_j := I - \ell_j e_j^T .$$

Dann gilt:

1.

$$L_j A = \begin{pmatrix} a_1 & & & \\ \vdots & & & \\ a_j & & & \\ a_{j+1} & - & \ell_{j+1,j} & a_j \\ \vdots & & & \\ a_n & - & \ell_{n,j} & a_j \end{pmatrix} = \begin{pmatrix} a_1 & & & \\ \vdots & & & \\ a_j & & & \\ a_{j+1} & - & \ell_{j+1} & a_j \\ \vdots & & & \\ a_n & - & \ell_n & a_j \end{pmatrix}$$

Die ersten j Zeilen von A bleiben daher unverändert, von den übrigen Zeilen werden Vielfache der j-ten Zeile subtrahiert.

2.

$$L_j^{-1} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & O & \\ & & 1 & & & \\ & & \ell_{j+1,j} & & & \\ & O & & 1 & & \\ & & \vdots & & \ddots & \\ & & \ell_{nj} & & & 1 \end{pmatrix} = I + \ell e_j^T ,$$

da $(I - \ell e_j^T)(I + \ell e_j^T) = I - \ell e_j^T \ell e_j^T = I$.

3. Sei $j < k$. Dann gilt:

$$L_j \cdot L_k = \begin{array}{cc}
\begin{array}{ccccccc} & & j\text{-te Spalte} & & k\text{-te Spalte} & & \\ & & \downarrow & & \downarrow & & \end{array} & \\
\begin{pmatrix}
1 & & & & & & \\
 & \ddots & & & & & \\
 & & 1 & & & O & \\
 & & -\ell_{j+1,j} & \ddots & & & \\
 & & & & 1 & & \\
 & & \vdots & O & & \ddots & \\
 & & & & -\ell_{k+1,k} & & \\
 & & & & \vdots & & \\
 & & -\ell_{n,j} & & -\ell_{n,k} & & 1
\end{pmatrix} &
\begin{array}{l} \\ \\ \leftarrow \quad j\text{-te Zeile} \\ \\ \leftarrow \quad k\text{-te Zeile} \\ \\ \\ \\ \\ \end{array}
\end{array}$$

da

$$\begin{aligned}
(I - \ell_j e_j^T)(I - \ell_k e_k^T) &= I - \ell_j e_j^T - \ell_k e_k^T + \ell_j e_j^T \ell_k e_k^T \\
&= I - \ell_j e_j^T - \ell_k e_k^T
\end{aligned}$$

4. Sei $\prod_{j=m}^{n} L_j := L_m \cdot L_{m+1} \cdot \ldots \cdot L_n$, wobei auf die Reihenfolge der Multiplikationen zu achten ist. Dann gilt:

$$\prod_{j=m}^{n} L_j = \begin{pmatrix}
1 & & & & & & & \\
 & \ddots & & & & & & \\
 & & 1 & & & & & \\
 & & & 1 & & O & & \\
 & & & -\ell_{m+1,m} & & & & \\
 & O & & & 1 & & & \\
 & & & & -\ell_{m+2,m+1} & & & \\
 & & & \vdots & \vdots & \vdots & 1 & \\
 & & & -\ell_{n,m} & -\ell_{n,m+1} & & -\ell_{n,n-1} & 1
\end{pmatrix}$$

Rechenregeln für elementare rechte Dreiecksmatrizen:

Die Rechenregeln für elementare rechte Dreiecksmatrizen lassen sich aus den entsprechenden Regeln für linke Dreiecksmatrizen herleiten.

Rechenregeln für Permutationsmatrizen:

Sei

$$A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}, \ a_j \in \mathbb{R}^n, \ 1 \le j \le n,$$

$$P_\sigma = \begin{pmatrix} e_{\sigma_1}^T \\ \vdots \\ e_{\sigma_n}^T \end{pmatrix} ,$$

dann gilt:

1.
$$P_\sigma A = \begin{pmatrix} e_{\sigma_1}^T \\ \vdots \\ e_{\sigma_n}^T \end{pmatrix} A = \begin{pmatrix} a_{\sigma_1} \\ \vdots \\ a_{\sigma_n} \end{pmatrix} \qquad \text{(Zeilen vertauscht)}$$

2. $P_\sigma^T P_\sigma = I$, da
$$P_\sigma^T P_\sigma = (e_{\sigma_1}, \ldots, e_{\sigma_n}) \cdot \begin{pmatrix} e_{\sigma_1}^T \\ \vdots \\ e_{\sigma_n}^T \end{pmatrix}$$

3. Sei $A = (\hat{a}_1, \ldots, \hat{a}_n)$, dann gilt:
$$AP^T = (\hat{a}_{\sigma_1}, \ldots, \hat{a}_{\sigma_n}) \qquad \text{(Spalten vertauscht)} ,$$
da
$$AP^T = (PA^T)^T .$$

Sei A eine $n \times n$-Matrix und Z eine elementare Matrix. Die folgenden Bemerkungen gelten:

1. Das linke Produkt ZA entsteht durch Zeilenoperationen auf A.
2. Das rechte Produkt AZ entsteht durch Spaltenoperationen auf A.
3. Um die Auswirkung von Z zu bestimmen, betrachte man $Z = ZI$ bzw. $Z = IZ$.

3.6 Das Gaußsche Eliminationsverfahren als Matrixprozeß

In diesem Abschnitt wird das Gaußsche Eliminationsverfahren als Folge von Matrixoperationen dargestellt. Zuerst wird ein Hilfssatz bewiesen:

Hilfssatz 3.4 *Sei $u \in \mathbb{R}^m$, $u = (u_1, \ldots, u_m)^T$, $u_1 \neq 0$. Dann gibt es genau eine normierte linke Dreiecksmatrix M, $M = I - \ell e_1^T$,*

$$\begin{aligned} \ell &:= (0, \ell_2, \ldots, \ell_m)^T \\ e_1 &:= (1, 0, \ldots, 0)^T \end{aligned}$$

und genau ein $\alpha \in \mathbb{R}$ *wofür*

$$Mu = \alpha e_1 \ .$$

Es gilt:

$$\ell = (0, u_2/u_1, \ldots, u_n/u_1)^T$$

Beweis: Die Gleichung $Mu = \alpha e_1$ läßt sich umformieren:

$$(I - \ell e_1^T)u = \alpha e_1 \ ,$$

oder

$$u - \ell u_1 = \alpha e_1 \ ,$$

so daß

$$\ell = (u - \alpha e_1)/u_1 \ .$$

Aus der Bedingung $e_1^T \ell = 0$ folgt nun $e_1^T u - \alpha = 0$ oder $\alpha = u_1$, d.h.

$$\ell = (u - u_1 e_1)/u_1 = \begin{pmatrix} 0 \\ u_2/u_1 \\ \vdots \\ u_n/u_1 \end{pmatrix} .$$

$\square$

Wir betrachten nun das folgende Verfahren, das wir schon jetzt als das Gaußsche Verfahren bezeichnen, da es sich später damit als rechnerisch äquivalent zeigt.

Das Gaußsche Eliminationsverfahren als Matrixprozeß:
Problem: Bestimme $x \in \mathbb{R}^n$ mit $Ax = b$.

Schritt1: Berechne eine LR-Zerlegung von A, $A = LR$, wobei L eine normierte linke Dreiecksmatrix und R eine rechte Dreiecksmatrix ist.

Schritt2: Bestimme $y \in \mathbb{R}^n : Ly = b$.

Schritt3: Bestimme $x \in \mathbb{R}^n : Rx = y$.

Diese drei Schritte werden jetzt ausführlich besprochen:

3.6.1 Schritt 1: LR-Zerlegung von A

Die LR-Zerlegung von A erfolgt durch die Berechnung von mehreren Hilfsmatrizen: $A^{(1)}, \ldots, A^{(n)}$, $L_1, \ldots, L_{n-1}$, $M_1, \ldots, M_{n-1}$:

1.

$$
\begin{aligned}
A^{(k)} &:= (a_{ij}^{(k)}) \\
&= \begin{pmatrix}
a_{11}^{(k)} & a_{12}^{(k)} & a_{1,k-1}^{(k)} & a_{1,k}^{(k)} & \dots & a_{1n}^{(k)} \\
O & \ddots & \vdots & & & \\
O & O & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \dots & a_{k-1,n}^{(k)} \\
O & O & O & a_{kk}^{(k)} & \dots & a_{k,n}^{(k)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
O & O & O & a_{nk}^{(k)} & \dots & a_{nn}^{(k)}
\end{pmatrix} \\
&= \begin{pmatrix}
k-1 \text{ Spalten} & n-k+1 \text{ Spalten} \\
A_{11}^{(k)} & A_{12}^{(k)} \\
O & A_{22}^{(k)}
\end{pmatrix}
\begin{array}{l} \\ \} \quad k-1 \text{ Zeilen} \\ \} \quad n-k+1 \text{ Zeilen} \end{array} , \qquad (3.2)
\end{aligned}
$$

so daß $A_{11}^{(k)}$ eine $(k-1) \times (k-1)$ rechte Dreiecksmatrix und $A_{22}^{(k)}$ eine $(n-k+1) \times (n-k+1)$ Matrix ist.

2. $L_k = I - \ell_k e_k^T$, $1 \le k \le n-1$, mit

$$\ell_k := (0, \dots, 0 \ , \ \ell_{k+1,k}, \dots, \ell_{n,k})^T \ ,$$

d.h.

$$L_k = \begin{pmatrix} I_{k-1} & O \\ O & M_k \end{pmatrix} \text{mit} \quad M_k := \begin{pmatrix} 1 & & O & \\ -\ell_{k+1,k} & \ddots & & \\ \vdots & & & \\ -\ell_{n,k} & O & & 1 \end{pmatrix} .$$

3.

$$
\begin{aligned}
A^{(1)} &:= A \ , \\
A^{(k+1)} &:= L_k A^{(k)} \ , \ 1 \le k \le n-1 \ .
\end{aligned}
$$

4. Sei $u^{(k)} \in \mathbb{R}^{n-k+1}$ die erste Spalte von $A_{22}^{(k)}$,

$$u^{(k)} := (a_{kk}^{(k)}, \dots, a_{nk}^{(k)})^T \ .$$

Vorausgesetzt, daß $u_1^{(k)} = a_{kk}^{(k)} \neq 0$, gibt es wegen Hilfssatz 3.4 genau eine normierte linke Dreiecksmatrix M_k, so daß

$$M_k u^{(k)} = (u_1^{(k)}, 0, \dots, 0)^T \ .$$

Es gilt:

$$M_k = I - \ell e_1^{(n-k+1)T} \ ,$$

wobei $e_1^{(n-k+1)T}$ der erste Achsenvektor im $\mathbb{R}^{n-k+1}$ ist und

$$\ell := (0, u_2^{(k)}/u_1^{(k)}, \dots, u_n^{(k)}/u_1^{(k)}) \ ,= (0, a_{k+1,k}^{(k)}/a_{kk}^{(k)}, \dots, a_{nk}^{(k)}/a_{kk}^{(k)}).$$

Es gilt auch

$$L_k := \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & O \\ & & 1 & & & \\ & & \frac{-a_{k+1,k}^{(k)}}{a_{kk}^{(k)}} & 1 & & \\ & & \vdots & & \ddots & \\ & & \frac{a_{n,k}^{(k)}}{a_{kk}^{(k)}} & & & 1 \end{pmatrix}$$

5. Liegt $A^{(k)}$ vor, kann L_k wie oben berechnet werden. Anschließend wird $A^{(k+1)} := L_k A^{(k)}$ berechnet. Es bleibt zu überprüfen, ob $L_k A^{(k)}$ die Gestalt (3.2) hat.

   Sei

$$\begin{aligned} A_{12}^{(k)} &= (v^{(k)}, \tilde{A}_{12}^{(k)}) \ , \\ A_{22}^{(k)} &= (u^{(k)}, \tilde{A}_{22}^{(k)}) \end{aligned}$$

   mit

$$u^{(k)} \in \mathbb{R}^{n-k+1} \ , \ v^{(k)} \in \mathbb{R}^{k-1} \ .$$

   Dann folgt:

$$\begin{aligned} L_k A^{(k)} &= \begin{pmatrix} I_{k-1} & O \\ O & M_k \end{pmatrix} \cdot \begin{pmatrix} A_{11}^{(k)} & v^{(k)} & \tilde{A}_{12}^{(k)} \\ O & u^{(k)} & \tilde{A}_{22}^{(k)} \end{pmatrix} \\ &= \begin{pmatrix} A_{11}^{(k)} & v^{(k)} & \tilde{A}_{12}^{(k)} \\ O & M_k u^{(k)} & M_k \tilde{A}_{22}^{(k)} \end{pmatrix} \\ &= \begin{pmatrix} A_{11}^{(k)} & v^{(k)} & \tilde{A}_{12}^{(k)} \\ 0 & a_{kk}^{(k)} & \\ O & O & M_k \tilde{A}_{22}^{(k)} \end{pmatrix} \\ &= \begin{pmatrix} A_{11}^{(k+1)} & A_{12}^{(k+1)} \\ O & A_{22}^{(k+1)} \end{pmatrix} \end{aligned}$$

   wobei

$$A_{11}^{(k+1)} := \begin{pmatrix} A_{11}^{(k)} & v^{(k)} \\ O & a_{kk}^{(k)} \end{pmatrix}$$

   eine $(k \times k)$ rechte Dreiecksmatrix wie in (3.2) ist.

6. Durch die Wiederholung der Schritte 1 - 5 oben für $k = 1, \dots, n-1$ unter der Voraussetzung, daß $a_{kk}^{(k)} \neq 0$ für $1 \leq k \leq n-1$, können Matrizen $L_1, \dots, L_{n-1}$ und $A^{(1}, \dots, A^{(n)}$ berechnet werden, so daß

$$L_{n-1} \cdot L_{n-2} \cdot \ldots \cdot L_1 \cdot A = R ,$$

wo

$$R := A^{(n)}$$

eine rechte Dreiecksmatrix ist.

7. Es folgt:

$$\begin{array}{rcl} A &=& LR \qquad \text{mit} \\ R &:=& A^{(n)} \\ L &=& L_1^{-1} \cdot L_2^{-1} \ldots L_{n-1}^{-1} \ . \end{array}$$

Wegen der Eigenschaften von elementaren normierten linken Dreiecksmatrizen gilt:

$$L = \begin{pmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & & & \\ \vdots & \vdots & \ddots & & \\ & & & 1 & \\ \ell_{n1} & \ell_{n2} & & \ell_{n-1,n} & 1 \end{pmatrix},$$

d.h.

$$L = \begin{pmatrix} 1 & & & & \\ a_{21}^{(1)}/a_{11}^{(1)} & 1 & & & \\ a_{31}^{(1)}/a_{11}^{(1)} & a_{32}^{(2)}/a_{22}^{(2)} & & & \\ \vdots & \vdots & \ddots & & \\ & & & 1 & \\ a_{n1}^{(1)}/a_{11}^{(1)} & a_{n2}^{(2)}/a_{22}^{(2)} & \ldots & a_{n,n-1}^{(n-1)}/a_{n-1,n-1}^{(n-1)} & 1 \end{pmatrix}$$

Zusammenfassend:
Sei

$$A = A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & \ldots & a_{1n}^{(1)} \\ \vdots & & \vdots \\ a_{n1}^{(1)} & \ldots & a_{nn}^{(1)} \end{pmatrix}$$

1. Teilschritt:

$$L_1 A = \begin{pmatrix} a_{11}^{(1)} & . & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix}$$

mit

$$\begin{aligned} \ell_{i1} &= \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} && i = 2, \dots, n \\ a_{11}^{(1)} &\neq 0 \\ a_{i,k}^{(2)} &= a_{i,k}^{(1)} - \ell_{i1} \cdot a_{i,k}^{(1)} && i, k = 2, \dots, n \end{aligned}$$

2. Teilschritt:

$$L_2 L_1 A = \begin{pmatrix} a_{11}^{(1)} & . & . & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & \dots & a_{2n}^{(2)} \\ & 0 & a_{33}^{(3)} & \dots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \dots & a_{nn}^{(3)} \end{pmatrix}$$

mit

$$\begin{aligned} \ell_{i2} &= a_{i2}^{(2)} / a_{22}^{(2)} && i = 3, \dots, n && a_{22}^{(2)} \neq 0 \\ a_{i,k}^{(3)} &= a_{i,k}^{(2)} - \ell_{i2} a_{2k}^{(2)} && i, k = 3, \dots, n \end{aligned}$$

Vor dem j-ten Teilschritt:

$$L_{j-1} \dots L_1 A = \begin{pmatrix} a_{11}^{(1)} & & \dots & & a_{1n}^{(1)} \\ & \ddots & & & \vdots \\ & & a_{jj}^{(j)} & \dots & a_{jn}^{(j)} \\ O & & \vdots & & \vdots \\ & & a_{nj}^{(j)} & \dots & a_{nn}^{(j)} \end{pmatrix}$$

j-ter Teilschritt:

$$L_j L_{j-1} \dots L_1 A = \begin{pmatrix} a_{11}^{(1)} & & . & . & \dots & a_{1n}^{(1)} \\ & \ddots & & & & \vdots \\ & & a_{jj}^{(j)} & \dots & \dots & a_{jn}^{(j)} \\ & & 0 & a_{j+1,j+1}^{(j+1)} & \dots & a_{j+1,n}^{(j+1)} \\ & & \vdots & \vdots & & \vdots \\ & & 0 & a_{n,j+1}^{(j+1)} & \dots & a_{n,n}^{(j+1)} \end{pmatrix}$$

mit

$$\begin{aligned} \ell_{ij} &= a_{ij}^{(j)} / a_{jj}^{(j)} && i = j+1, \dots, n && a_{jj}^{(j)} \neq 0 \\ a_{i,k}^{(j+1)} &= a_{i,k}^{(j)} - \ell_{ij} a_{jk}^{(j)} && i, k = j+1, \dots, n \end{aligned}$$

Nach $n-1$ Schritten:

$$L_{n-1}\dots L_1 A = \begin{pmatrix} a_{11}^{(1)} & \dots & a_{1n}^{(1)} \\ & \ddots & \vdots \\ O & & \\ & & a_{nn}^{(n)} \end{pmatrix}$$

Nun invertieren wir die L_j nacheinander und erhalten:

$$A = L_1^{-1}\dots L_{n-1}^{-1}\cdot R = L\ R\ ,$$

mit

$$L = \begin{pmatrix} 1 & & & \\ & & O & \\ \ell_{2,1} & & \ddots & \\ \vdots & \ddots & & \\ \ell_{n,1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix}$$

Beispiel:

$$A = A^{(1)} = \begin{pmatrix} 3 & 1 & 6 \\ 2 & 1 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2/3 & 1 & 0 \\ -1/3 & 0 & 1 \end{pmatrix}\ ,\ A^{(2)} = L_1 A^{(1)} = \begin{pmatrix} 3 & 1 & 6 \\ 0 & 1/3 & -1 \\ 0 & 2/3 & -1 \end{pmatrix}\ ,$$

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}\ ,\ A^{(3)} = L_2 A^{(2)} = \begin{pmatrix} 3 & 1 & 6 \\ 0 & 1/3 & -1 \\ 0 & 0 & 1 \end{pmatrix}\ .$$

$$L = L_1^{-1}L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 1/3 & 2 & 1 \end{pmatrix} \qquad \text{(eine normierte linke Dreiecksmatrix)}$$

$$R = \begin{pmatrix} 3 & 1 & 6 \\ 0 & 1/3 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$D = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1 \end{pmatrix}\ ,\ D^{-1}R = \begin{pmatrix} 1 & 1/3 & 2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}$$

$A = LR = LD(D^{-1}R)$.

3.6.2 Schritt 2: Bestimme $y \in \mathbb{R}^n$ mit $Ly = b$

$$y = L^{-1}b = L_{n-1} \dots L_1 b.$$

Der Vektor y wird also erhalten, wenn der Vektor b den gleichen Operationen wie die Spalten von A bei der Berechnung ihrer LR-Zerlegung unterworfen wird.

3.6.3 Schritt 3: Bestimme $x \in \mathbb{R}^n$ mit $Rx = y$

Die Berechnung von x erfolgt wie schon beschrieben.

3.6.4 Berechnung der Inversen A^{-1}

Die Identität

$$AA^{-1} = I$$

hat zur Folge, daß die k-te Spalte von A^{-1} , $\overline{a}_k$ die Gleichung

$$A\overline{a}_k = e_k$$

erfüllt. Ist die LR-Zerlegung von A bekannt, kann A^{-1} mit $\frac{2}{3}\, n^3$ Operationen berechnet werden.

Die Inverse einer Matrix wird in der Praxis nur selten gewünscht. Eine zweite Methode - das Gauß-Jordan-Verfahren - bietet organisatorische Vorteile.

3.7 Alternative Implementierungen des Gaußschen Eliminationsverfahrens

3.7.1 Das Crout-Doolittle Verfahren

Die Grundidee ist, daß die LR-Zerlegung direkt ohne Zwischenberechnungen berechenbar ist.

$$\begin{aligned} LR &= A \\ \Rightarrow a_{ij} &= \sum_{k=1}^{\min(i,j)} \ell_{ik} r_{kj} \ . \end{aligned}$$

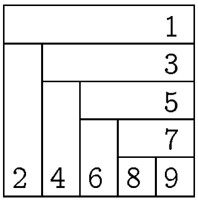
Die Koeffizienten ℓ_{ik} , r_{kj} lassen sich der Reihe nach berechnen.

Man erhält:

$$
\begin{array}{rcl}
\text{for} \quad i & := & 1 \quad \text{to n do begin} \\
\ell_{ii} & := & 1 \\
\text{for} \quad j & := & i \quad \text{to n do begin} \\
r_{ij} & := & a_{ij} - \sum\limits_{k=1}^{i-1} \ell_{ik} r_{kj}; \\
\ell_{ji} & = & \left(a_{ji} - \sum\limits_{k=1}^{i-1} \ell_{jk} r_{ki} \right) / r_{ii}; \\
\text{end;} & & \\
\text{end;} & &
\end{array}
\tag{3.3}
$$

(3.4)

Man verwendet also die folgende Parkettierung von A :



Gaußelimination und direkte Crout-Doolittle Dreieckszerlegung unterscheiden sich nur in der Reihenfolge der Operationen. Beide Algorithmen sind theoretisch und numerisch völlig äquivalent. Bei der Gaußelimination werden allerdings die Skalarprodukte in (3.3) nur schrittweise, unter zeitweiliger Abspeicherung der Zwischenresultate, bei der direkten Dreieckszerlegung dagegen in einem Zuge gebildet. Deshalb verdient die direkte Crout-Doolittle Dreieckszerlegung den Vorzug, wenn

a) zur Reduktion der Rundungsfehler die Skalarprodukte in doppelter Genauigkeit akkumuliert werden sollen (keine Zwischenspeicherung von doppelt genauen Zahlen!),

b) aus organisatorischen Gründen die Zwischenspeicherung von Ergebnissen (z.B. auf Magnetband oder Magnetplatte) zeitaufwendig oder (z.B. bei Handrechnung) fehleranfällig ist.

3.7.2 Das Cholesky Verfahren

Wenn A symmetrisch und positiv definit ist, ist es möglich, A als $A = LL^T$ zu zerlegen. Die Berechnungen erfordern nur die Hälfte des Speicherplatzes und nur die Hälfte des Rechenaufwandes.

Definition 3.4 *Eine Matrix* $A \in Mat(n \times n)$ *heißt* positiv definit *genau dann, wenn*

$$x^T A x > 0 \,, \quad \textit{für alle } x \in \mathbb{R}^n \,, \ x \neq 0 \,.$$

Es ist möglich, die Eigenschaften einer positiv-definiten Matrix mit unterschiedlichen Methoden herzuleiten. Hier benutzen wir Ideen aus der Gaußschen Elimination.

Hilfssatz 3.5 *$A^{(1)}$ sei eine symmetrische und positiv definite $n \times n$- Matrix. Dann gilt:*

1. *$a_{11}^{(1)} > 0$, so daß der erste LR-Teilschritt (Multiplikation mit L_1) möglich ist.*
2. *$A_{22}^{(2)}$ sei die $(n-1)\times(n-1)$-Matrix, die nach einem Teilschritt der LR-Zerlegung (Multiplikation mit L_1) entsteht. Dann ist $A_{22}^{(2)}$ auch symmetrisch und positiv definit.*

Beweis:

1. $a_{11}^{(1)} = e_1^T A^{(1)} e_1 > 0$
2. $a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i1}^{(1)} a_{1j}^{(1)} / a_{11}^{(1)}$, $i,j \geq 2$, woraus sofort folgt, daß $A_{22}^{(2)}$ symmetrisch ist.
   Sei nun $y \in \mathbb{R}^{n-1}$, $y \neq 0$.
   Sei $\tilde{a} := (a_{12}^{(1)}, \dots, a_{1n}^{(1)})^T \in \mathbb{R}^{n-1}$, und $\alpha := \tilde{a}^T y / a_{11}^{(1)}$,

$$x := \begin{pmatrix} -\alpha \\ y \end{pmatrix} \in \mathbb{R}^n .$$

Dann folgt:

$$A = \begin{pmatrix} a_{11} & \tilde{a}^T \\ \tilde{a} & A_{22}^{(1)} \end{pmatrix} , \quad L_1 = \left(\begin{array}{c|c} 1 & 0 \\ \hline \frac{-\tilde{a}}{a_{11}^{(1)}} & I \end{array}\right) , \quad L_1 A = L_1 \begin{pmatrix} a_{11} & \tilde{a}^T \\ \tilde{a} & A_{22}^{(1)} \end{pmatrix} ,$$

so daß

$$A_{22}^{(2)} = A_{22}^{(1)} - \frac{1}{a_{11}^{(1)}} \tilde{a}\tilde{a}^T$$

und

$$\begin{aligned} y^T A_{22}^{(2)} y &= y^T A_{22}^{(1)} y - \frac{1}{a_{11}^{(1)}} y^T \tilde{a}\tilde{a}^T y \\ &= y^T A_{22}^{(1)} y - a_{11}^{(1)} \alpha^2 \\ &= (-\alpha, y^T) \begin{pmatrix} a_{11}^{(1)} & \tilde{a}^T \\ \tilde{a} & A_{22}^{(1)} \end{pmatrix} \begin{pmatrix} -\alpha \\ y \end{pmatrix} \\ &= x^T A^{(1)} x > 0 . \end{aligned}$$

Da $y \in \mathbb{R}^{n-1}$ beliebig war, ist $A_{22}^{(2)}$ positiv definit.

□

Hilfssatz 3.6 *Sei $A \in Mat(n \times n)$ eine symmetrische positiv definite Matrix. Dann gibt es eine LR-Zerlegung von A mit $r_{ii} = diag(R)_{ii} > 0 \;\; 1 \leq i \leq n$.*

Beweis: Wiederholte Anwendung von Hilfssatz 3.5. □

Satz 3.3 *Zu jeder reellen positiv definiten und symmetrischen $n \times n$- Matrix A gibt es genau eine reelle untere $n \times n$- Dreiecksmatrix $L = (\ell_{ik})$ mit $\ell_{ik} = 0$ für $k > i$ und $\ell_{ii} > 0$, $i = 1, 2, \ldots, n$, so daß $A = LL^T$.*

Beweis: Wenn A symmetrisch und positiv definit ist, gibt es nach Hilfssatz 3.6 eine LR-Zerlegung

$$A = \tilde{L}\tilde{R}$$

mit $\tilde{D} = \mathrm{diag}(\tilde{R})$ und $\tilde{d}_{ii} > 0$. Es folgt

$$A = \tilde{L}\tilde{D}\tilde{U}$$

mit

$$\tilde{U} := \tilde{D}^{-1}\tilde{R} .$$

Da A symmetrisch ist, gilt auch

$$A = A^T = \tilde{U}^T\tilde{D}^T\tilde{L}^T = \tilde{U}^T\tilde{S}$$

mit $\tilde{U}^T$ eine normierte linke Dreiecksmatrix und $\tilde{S}$ eine rechte Dreiecksmatrix. Die LR-Zerlegung einer Matrix ist eindeutig, woraus folgt:

$$\tilde{U}^T = \tilde{L} ,$$

also

$$A = \tilde{L}\tilde{D}\tilde{L}^T = LL^T$$

mit

$$L = \tilde{L}\tilde{D}^{1/2} .$$

Sei nun A eine Matrix mit

$$A = L_1L_1^T = L_2L_2^T ,$$

wobei L_1 und L_2 linke Dreiecksmatrizen sind mit positiven Diagonalelementen. Sei

$$\begin{aligned} D_1 &= \mathrm{diag}(L_1), \quad D_2 = \mathrm{diag}(L_2), \\ \tilde{L}_1 &= L_1 \cdot D_1^{-1}, \quad \tilde{L}_2 = L_2 \cdot D_2^{-1}, \\ \tilde{R}_1 &= D_1L_1^T, \quad \tilde{R}_2 = D_2L_2^T. \end{aligned}$$

Dann gilt:

$$A = L_1L_1^T = \tilde{L}_1\tilde{R}_1 = L_2L_2^T = \tilde{L}_2\tilde{R}_2.$$

Es folgt, daß $\tilde{L}_1 = \tilde{L}_2$ und $\tilde{R}_1 = \tilde{R}_2$. Weiter gilt:

$$\mathrm{diag}\;\; \tilde{R}_2 = D_2\;\mathrm{diag}\;(L_2^T) = D_2^2 = \mathrm{diag}\;\tilde{R}_1 = D_1^2.$$

Da die Diagonalelemente von D_1 und D_2 positiv sind, folgt $D_1 = D_2$. □

Satz 3.4 *Sei A eine symmetrische positiv definite Matrix. Dann gilt:*

1. *A^{-1} existiert und ist symmetrisch und positiv definit.*
2. *Alle Hauptuntermatrizen von A sind positiv definit.*
3. *Alle Hauptminoren von A sind positiv.*

Beweis:

1. Es existiert eine linke Dreiecksmatrix L mit positiven Diagonalelementen, so daß
$$A = LL^T \ .$$
Es folgt:
$$A^{-1} = (L^T)^{-1}L^{-1} = (L^{-1})^T L^{-1}.$$
Sei nun $x \in \mathbb{R}^n$, $x \neq 0$. Dann ist $L^{-1}x \neq 0$. Es gilt:
$$x^T A^{-1} x = x^T (L^{-1})^T L^{-1} x = (L^{-1}x)^T (L^{-1}x) = y^T y > 0,$$
so daß A^{-1} positiv definit ist.

2. Sei B eine $m \times m$-Hauptuntermatrix von A,
$$B = A \begin{bmatrix} i_1 \dots i_m \\ i_1 \dots i_m \end{bmatrix} ,$$
d.h. B wird aus den i_1-ten, $\dots, i_m$-ten Zeilen und i_1-ten, $\dots, i_m$-ten Spalten von A gebildet. Sei $y \in \mathbb{R}^n$, $y \neq 0$. Sei $S = \{i_1, \dots, i_m\}$. Sei $x \in \mathbb{R}^n$, $x = (x_i)$ mit
$$\begin{aligned} x_i &= 0 \quad \text{für } i \notin S \ , \\ x_i &= y_j \quad \text{für } i = i_j \in S \ . \end{aligned}$$
Dann gilt:
$$y^T B y = x^T A x > 0 \ .$$

3. Sei M ein $m \times m$-Hauptminor von A. Dann gilt:
$$M = \det B \ , \ B = A \begin{bmatrix} i_1 \dots i_m \\ i_1 \dots i_m \end{bmatrix} \ .$$
Da alle Hauptuntermatrizen von A positiv definit sind, ist B positiv definit.

Es folgt aus Satz 3.3, daß es $L \in \text{Mat}(m \times m)$ gibt mit L ist eine linke Dreiecksmatrix mit positivem Diagonal D, so daß $B = LL^T$. Es gilt:

$$M = \det B = (\det\ L) \cdot (\det\ L^T) = (\det D)^2 > 0.$$

□

Hilfssatz 3.7 *Sei $A^{(1)} \in Mat(n \times n)$. Seien $L_1, \ldots, L_{k-1}$ linke normierte Dreiecksmatrizen (wie bei Gaußscher Elimination) und*

$$A^{(k)} = L_{k-1} L_{k-2} \ldots L_1 A^{(1)} .$$

Dann gilt:

$$A^{(k)} \begin{pmatrix} 1, \ldots, k \\ 1, \ldots, k \end{pmatrix} = A^{(1)} \begin{pmatrix} 1, \ldots, k \\ 1, \ldots, k \end{pmatrix} = \prod_{j=1}^{k} a_{jj}^{(j)} ,$$

wobei

$$A \begin{pmatrix} i_1, \ldots, i_m \\ j_1, \ldots, j_m \end{pmatrix}$$

die entsprechende Unterdeterminante von A bezeichnet.

Beweis: Sei

$$A^{(1)} = \begin{pmatrix} B & C \\ D & E \end{pmatrix}$$

mit

$$B = A^{(1)} \begin{bmatrix} 1 \ldots k \\ 1 \ldots k \end{bmatrix} \in \text{Mat}(k \times k)$$

und

$$L_j = \begin{pmatrix} B_j & O \\ D_j & E_j \end{pmatrix}$$

mit

$$B_j = L_j \begin{bmatrix} 1, \ldots, k \\ 1, \ldots, k \end{bmatrix} \in \text{Mat}(k \times k)$$

und

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{pmatrix}$$

mit $A_{11}^{(k)} \in \text{Mat}(k \times k)$. Dann gilt:

a) B_j ist eine normierte linke Dreiecksmatrix.

b) $A_{11}^{(k)} = B_{k-1} \ldots B_1 B$

Es folgt:

$$A^{(k)}\begin{pmatrix} 1\ldots k \\ 1\ldots k \end{pmatrix} = \det(A_{11}^{(k)}) = \prod_{j=1}^{k} a_{jj}^{(i)} = \left[\prod_{j=1}^{k-1} \det(B_j)\right] \det(B) = A^{(1)}\begin{pmatrix} 1\ldots k \\ 1\ldots k \end{pmatrix}$$

□

Satz 3.5 *Sei $A \in Mat(n \times n)$ symmetrisch. Die folgenden Aussagen sind paarweise äquivalent:*

a) A ist positiv definit.

b) Alle Eigenwerte von A sind positiv.

c) Alle führenden Hauptunterdeterminanten von A sind positiv.

d) A besitzt eine eindeutige Zerlegung $A = LL^T$ mit einer unteren Dreiecksmatrix L mit positiven Diagonalelementen.

Beweis:

1. $a \Longleftrightarrow d$ ist schon bewiesen worden.

2. $a \Longrightarrow c$ ist schon bewiesen worden.

3. Sei c erfüllt. Es folgt aus Hilfssatz 3.7, daß A eine LR-Zerlegung besitzt mit einer normierten linken Dreiecksmatrix L und einer rechten Dreiecksmatrix R, so daß

$$r_{jj} = a_{jj}^{(j)} > 0\ ,\quad 1 \le j \le n\ .$$

   Wie in dem Beweis von Satz 3.3 folgt nun, daß es eine linke Dreiecksmatrix $\tilde{L}$ mit positiven Diagonalelementen gibt, so daß

$$A = \tilde{L}\tilde{L}^T\ ,$$

   d.h. $c \Longrightarrow d$ und, da $d \Longrightarrow a$, folgt $c \Longrightarrow a$.

4. Der Fall b wird erst später untersucht.

□

Sei nun A eine symmetrische positiv definite $n \times n$- Matrix. Die LL^T-Zerlegung von A kann mit Hilfe des Cholesky-Verfahrens berechnet werden:

Der Cholesky-Algorithmus

```
for i := 1 to n do begin
for j := i to n do begin
    s := a[i,j];
    for k := i-1 down to 1 do
        s := s - a[j,k]*a[i,k];
    if j = i then p[i] = 1.0/sqrt(s);
    a[j,i] = s*p[i];
end;
end;
```

Es wird nur die Information aus dem oberen Dreieck von A benutzt. Die untere Dreiecksmatrix L wird auf dem unteren Dreieck von A gespeichert mit Ausnahme der Diagonalelemente von L, deren reziproke Werte auf p gespeichert sind.

Beispiel:

$$A = \begin{pmatrix} 4.0 & 5.0 & 2.0 \\ 5.0 & 6.5 & 1.5 \\ 2.0 & 1.5 & 6.0 \end{pmatrix}. \quad L = \begin{pmatrix} 2.0 & 0.0 & 0.0 \\ 2.5 & 0.5 & 0.0 \\ 1.0 & -2.0 & 1.0 \end{pmatrix}$$

3.8 Das Gaußsche Eliminationsverfahren mit Permutationen

Die LR-Zerlegung mit Permutation einer Matrix A erfolgt wie unten:

$$\begin{aligned} A^{(1)} &:= A \\ A^{(2)} &:= L_1 P_1 A^{(1)} \pi_1 \\ A^{(k+1)} &:= L_k P_k A^{(k)} \pi_k \\ &\quad \ldots \\ R &:= A^{(n)} = L_{n-1} P_{n-1} \ldots L_1 P_1 A^{(1)} \pi_1 \ldots \pi_{n-1} \,, \end{aligned} \tag{3.5}$$

wobei

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ O & A_{22}^{(k)} \end{pmatrix} \,,$$

$A_{11}^{(k)}$ eine $k \times k$ obere Dreiecksmatrix,

$$L_k = \begin{pmatrix} I_{k-1} & O \\ O & M_k \end{pmatrix}$$

eine normierte untere Dreiecksmatrix ist und

1. P_k und π_k Permutationsmatrizen sind, die die ersten k-1 Zeilen bzw. Spalten unverändert lassen. Es gilt also

$$P_k = \begin{pmatrix} I_{k-1} & O \\ O & P_k' \end{pmatrix} \pi_k = \begin{pmatrix} I_{k-1} & O \\ O & \pi_k' \end{pmatrix} .$$

2. P_k und π_k werden so gewählt, daß das (k,k)-Element von $P_k A^{(k)} \pi_k$ nicht null ist, so daß L_k berechenbar ist.

3. Ist A singulär, dann ist $A_{22}^{(s)}$ null für ein $s < n$, d.h. $a_{ij}^{(s)} = 0$ für $s \le i$, $j \le n$ und der Prozeß endet.

Hilfssatz 3.8 *Für $j < k$ gilt $P_k L_j = L_j' P_k$, mit $L_j' = I - \ell_j' e_j^T$, $\ell_j' = P_k \ell_j$, so daß L_j' eine linke elementare Dreiecksmatrix ist.*

Beweis: Sei $\ell = (\ell_{j+1,j}, \dots, \ell_{n,j})^T$. Dann gilt:

$$\begin{aligned} P_k L_j P_k^T &= \begin{pmatrix} I_j & \vdots & O \\ \cdots & \cdots & \cdots \\ O & \vdots & P \end{pmatrix} \cdot \begin{pmatrix} I_j & \vdots & O \\ \cdots & \cdots & \cdots \\ O \vdots \ell & \vdots & I_{n-j} \end{pmatrix} \cdot \begin{pmatrix} I_j & \vdots & O \\ \cdots & \cdots & \cdots \\ O & \vdots & P^T \end{pmatrix} \\ &= \begin{pmatrix} I_j & \vdots & O \\ \cdots & \cdots & \cdots \\ O \vdots P\ell & \vdots & P \end{pmatrix} \cdot \begin{pmatrix} I_j & \vdots & O \\ \cdots & \cdots & \cdots \\ O & \vdots & P^T \end{pmatrix} = \begin{pmatrix} I_j & \vdots & O \\ \cdots & \cdots & \cdots \\ O \vdots P\ell & \vdots & I_{n-j} \end{pmatrix} \end{aligned}$$

□

Satz 3.6 *A sei eine nicht-singuläre $n \times n$-Matrix. $A^{(k)}$, π_k , L_k , P_k , R seien wie in (3.5).*

$$\begin{aligned} P &:= P_{n-1} \dots P_1 \\ \pi &:= \pi_1 \dots \pi_{n-1} \\ L &:= P(L_{n-1} P_{n-1} \dots L_1 P_1)^{-1} . \end{aligned}$$

Dann gilt:

$$PA\pi = LR .$$

Die Matrizen L und R sind linke bzw. rechte Dreiecksmatrizen.

Beweis: Da $A^{(1)} = A$ regulär ist, gibt es Permutationsmatrizen P_1 und π_1, so daß das (1,1) Element von $P_1 A^{(1)} \pi_1$ nicht null ist. Es existiert L_1 und $A^{(2)}$. Durch Induktion folgt, daß die Matrizen $A^{(k)}$, $L^{(k)}$, P_k und π_k für $1 \le k \le n$ existieren. Folglich tritt Fall 3) nicht auf und der Prozeß bricht nicht ab.

Mit Hilfssatz 3.8 folgt:

$$L^{-1} = (L_{n-1}P_{n-1}\dots L_1P_1)P^{-1} = \tilde{L}_{n-1}\dots\tilde{L}_1 \ ,$$

wobei

$$\tilde{L}_k = P_{n-1}\dots P_{k+1}L_kP_{k+1}^T\dots P_{n-1}^T \ .$$

Es folgt:

$$L^{-1}PA\pi = R \ ,$$

oder

$$PA\pi = LR \ .$$

$\square$

Beispiel:

$$A = A^{(1)} = \begin{pmatrix} 2 & 1 & 3 \\ 3 & 1 & 6 \\ 1 & 1 & 1 \end{pmatrix} \quad , \quad b = \begin{pmatrix} 13 \\ 23 \\ 6 \end{pmatrix}$$

Wir setzen

$$P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad \pi_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$P_1A^{(1)}\pi_1 = \begin{pmatrix} 6 & 3 & 1 \\ 3 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ -1/6 & 0 & 1 \end{pmatrix} \quad , \quad A^{(2)} = L_1P_1A^{(1)}\pi_1 = \begin{pmatrix} 6 & 3 & 1 \\ 0 & 1/2 & 1/2 \\ 0 & 1/2 & 5/6 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad , \quad \pi_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$P_2A^{(2)}\pi_2 = \begin{pmatrix} 6 & 1 & 3 \\ 0 & 5/6 & 1/2 \\ 0 & 1/2 & 1/2 \end{pmatrix}$$

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3/5 & 1 \end{pmatrix}$$

$$R = A^{(3)} = L_2P_2A^{(2)}\pi_2 = \begin{pmatrix} 6 & 1 & 3 \\ 0 & 5/6 & 1/2 \\ 0 & 0 & 1/5 \end{pmatrix}$$

$$P = P_2P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\pi = \pi_1\pi_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$PA\pi = \begin{pmatrix} 6 & 1 & 3 \\ 1 & 1 & 1 \\ 3 & 1 & 2 \end{pmatrix}$$

$$\begin{aligned} L &= PP_1^{-1}L_1^{-1}P_2^{-1}L_2^{-1} \\ &= P_2L_1^{-1}P_2^TL_2^{-1} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/6 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3/5 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 1/6 & 1 & 0 \\ 1/2 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3/5 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 1/6 & 1 & 0 \\ 1/2 & 3/5 & 1 \end{pmatrix} \end{aligned}$$

$$LR = \begin{pmatrix} 1 & 0 & 0 \\ 1/6 & 1 & 0 \\ 1/2 & 3/5 & 1 \end{pmatrix} \cdot \begin{pmatrix} 6 & 1 & 3 \\ 0 & 5/6 & 1/2 \\ 0 & 0 & 1/5 \end{pmatrix} = \begin{pmatrix} 6 & 1 & 3 \\ 1 & 1 & 1 \\ 3 & 1 & 2 \end{pmatrix}$$

$$PA\pi =: \overline{A}$$

Gesucht ist die Lösung des Gleichungssystems $Ax = b$.

$$\begin{aligned} \Rightarrow \quad (PA\pi)\pi^Tx &= Pb \\ \Rightarrow \quad LRz &= Pb \\ x &= \pi z \\ \Rightarrow \quad Ly &= Pb \\ Rz &= y \\ x &= \pi z \end{aligned}$$

$$\begin{array}{lclclcr} 1y_1 & & & & & = & 23 \\ \frac{1}{6}\,y_1 & + & y_2 & & & = & 6 \\ \frac{1}{2}\,y_1 & + & \frac{3}{5}\,y_2 & + & y_3 & = & 13 \end{array}$$

$$\begin{aligned} \Rightarrow \quad y_1 &= 23 \\ y_2 &= 13/6 \\ y_3 &= 1/5 \end{aligned}$$

$$\begin{array}{rcrcrcl} \Rightarrow \; 6z_1 & + & z_2 & + & 3z_3 & = & 23 \\ & & \frac{5}{6}\, z_2 & + & \frac{1}{2}\, z_3 & = & 13/6 \\ & & & & \frac{1}{5}\, z_3 & = & 1/5 \end{array}$$

$$\begin{aligned} \Rightarrow \quad z_3 &= 1 \\ z_2 &= 2 \\ z_1 &= 3 \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad x_1 &= z_3 = 1 \\ x_2 &= z_2 = 2 \\ x_3 &= z_1 = 3 \end{aligned}$$

Die Inverse von $\overline{A}$, C, erfüllt die Bedingungen:

$$\overline{A}C = I \,.$$

Wir setzen $C = (c_1, c_2, c_3)$. Dann gilt:

$$LRc_1 = e_1 \quad , \quad LRc_2 = e_2 \quad , \quad LRc_3 = e_3$$

$$\Rightarrow C = \begin{pmatrix} 1 & 1 & -2 \\ 1 & 3 & -3 \\ -2 & -3 & 5 \end{pmatrix}$$

$$\Rightarrow A^{-1} = \pi C P = \begin{pmatrix} 5 & -2 & -3 \\ -3 & 1 & 3 \\ -2 & 1 & 1 \end{pmatrix}$$

Weitere Bemerkungen

Bei der Implementierung der Gaußschen Elimination mit Permutationen ist zu beachten:

1. Die Permutationen P_k und π_k müssen nicht notwendig physikalisch durchgeführt werden. Damit wird Rechenzeit gespart, aber das Programm wird aufwendiger.

2. Wenn die Matrizen L_k im gleichen Feld wie die Matrizen $A^{(k)}$ gespeichert werden, was ja möglich ist, weil die entsprechenden Elemente von $A^{(k)}$ null sind, dann können $A^{(k)}$ und L_k gleichzeitig permutiert werden. Die Matrizen $\tilde{L}_k$ entstehen dann automatisch.

3. Es ist nicht nötig, die Permutationsmatrizen zu speichern, sondern nur die gegenwärtigen Vektoren:
$$P_k \dots P_1 u \quad \text{und} \quad v\pi_1 \dots \pi_k \,,$$
wobei
$$u = v^T = (1, 2, \dots, n)^T \in \mathbb{R}^n \,.$$

3.9 Effizienz: einfache Abschätzungen

Der Rechenaufwand, der für die Lösung von n linearen Gleichungen erforderlich ist, ist einer der wichtigsten Merkmale eines Algorithmus zur Lösung von Gleichungssystemen.

In diesem Abschnitt werden einige einfache Abschätzungen unter folgenden Voraussetzungen hergeleitet:

a) Es gibt einen einheitlichen Speicher, der groß genug ist, um das Programm und alle Daten zu speichern. Sehr schnelle Speicher (Cache-Speicher) und langsamere Speicher (Platten usw.) werden nicht berücksichtigt.

b) Nur die arithmetischen Berechnungen werden in Betracht gezogen. Speicherzuweisungen, Schleifenberechnungen usw. werden nicht berücksichtigt.

c) Nur ein einfacher von-Neumann-Rechner wird vorausgesetzt. Parallelrechner, Vektorrechner usw. werden nicht berücksichtigt.

d) Es wird vorausgesetzt, daß n, die Anzahl der Gleichungen, so groß ist, daß

$$n^r = O(n^s) \quad , \quad \text{für} \quad s > r > 0 \, .$$

e) Da Additionen oder Subtraktionen sehr oft in Verbindung mit Multiplikationen auftreten, wird die Recheneinheit:

$$1 \text{ Operation} := 1 \text{ Multiplikation} + 1 \text{ Addition/Subtraktion}$$

benutzt.

f) Schnelle Verfahren, sei es das Verfahren von Strassen oder iterative Verfahren für spezielle Matrizen, werden später diskutiert.

Man benutzt die Formeln:

$$\sum_{k=1}^{n} 1 = n \quad , \quad \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^{n} k^r = \frac{B_{r+1}(n+1) - B_{r+1}}{r+1}$$

mit

$$\frac{te^{xt}}{e^t - 1} = \sum_{n=0}^{\infty} B_n(x) \frac{t^n}{n!}$$

Satz 3.7 *Die LR-Zerlegung einer $n \times n$-Matrix mit Hilfe des Gaußschen Verfahrens erfordert*

$$\begin{aligned} \sum_{k=1}^{n-1} k &= \frac{n(n-1)}{2} \quad \textit{Divisionen}\,, \\ \sum_{k=1}^{n-1} (k-1)^2 &= \frac{n(n-1)(2n-1)}{6} \quad \textit{Operationen}\,. \end{aligned}$$

Für großes n ergibt sich aus Satz 3.7, daß die LR-Zerlegung einer Matrix mit Hilfe des Gaußschen Verfahrens $0(n^3/3)$ Operationen erfordert.

Satz 3.8 *Die Lösung eines $n \times n$ gestaffelten Gleichungssystems erfordert:*

$$\begin{aligned} \sum_{k=1}^{n} 1 &= n \quad \textit{Divisionen} \\ \sum_{k=1}^{n-1} k &= \frac{n(n-1)}{2} \quad \textit{Operationen.} \end{aligned}$$

Für großes n erfordert die Lösung eines $n \times n$ gestaffelten Gleichungssystems $0(n^2/2)$ Operationen.

Die Tatsache, daß die Lösung eines gestaffelten Gleichungssystems relativ schnell ist, erklärt, warum Iterative Verbesserung ein praktisches Verfahren ist.

Satz 3.9 *A sei eine $n \times n$-Matrix mit Bandbreite m, also*

$$a_{ij} = 0 \quad , \quad \textit{falls} \quad |i-j| > m\,.$$

Die LR-Zerlegung von A sei mit dem Gaußschen Verfahren ohne Pivotsuche berechnet. Die erforderlichen Operationen sind:

$$\begin{aligned} &\frac{m(2n-m-1)}{2} \quad \textit{Divisionen} \\ &m^2(n-m) + \frac{m(m-1)(2m-1)}{6} \quad \textit{Operationen}\,. \end{aligned}$$

Insbesondere für $m = 1$:

$$\begin{aligned} n-1 &\quad \textit{Divisionen} \\ n-1 &\quad \textit{Operationen.} \end{aligned}$$

Satz 3.10 *Die Lösung eines $n \times n$ gestaffelten m-Band Gleichungssystems erfordert:*

$$\begin{aligned} 2n &\quad \textit{Divisionen} \\ (m)(2n-m-1) &\quad \textit{Operationen}\,. \end{aligned}$$

Insbesondere für $m = 1$

$$\begin{array}{rl} 2n & \textit{Divisionen} \\ 2(n-1) & \textit{Operationen}\,. \end{array}$$

Es folgt aus den Sätzen 3.9 und 3.10, daß das Gaußsche Verfahren ohne Pivotsuche für ein Tridiagonales Gleichungssystem nur

$$\begin{array}{rl} 3n-1 & \text{Divisionen} \\ 3n-3 & \text{Operationen} \end{array}$$

erfordert.

Satz 3.11 *Die LR-Zerlegung einer* $n \times n$ *symmetrischen positiv definiten Matrix mit Hilfe des Choleski-Verfahrens erfordert*

$$\begin{array}{rcll} & & n & \textit{Quadratwurzeln} \\ \displaystyle\sum_{k=1}^{n-1} k & = & \dfrac{n(n-1)}{2} & \textit{Divisionen} \\ \displaystyle\sum_{k=1}^{n-1} \frac{(k-1)^2}{2} & = & \dfrac{n(n-1)(2n-1)}{6} & \textit{Operationen} \end{array}$$

Kapitel 4

Rechenhilfsmittel

4.1 Einführung

In jeder Epoche hat der Mensch Rechenhilfsmittel benutzt, und seien es auch nur seine Finger gewesen. Grundsätzlich gibt es zwei Arten von Rechenhilfsmittel:

1. *Digitale Rechenhilfsmittel* stellen alle Zahlen durch eine Anzahl von Ziffern dar. Ein Taschenrechner ist ein typisches digitales Hilfsmittel.

2. *Analoge Rechenhilfsmittel* stellen alle Zahlen durch physikalische Größen dar. Ein Rechenstab ist ein typisches analoges Hilfsmittel.

Beispiele digitaler und analoger Rechenhilfsmittel werden in den Tabellen 4.1 und 4.2 angegeben, wobei die Symbole die folgenden Bedeutungen haben:

v	:	veraltet
Ib	:	wird in Numerik I benutzt
IIb	:	wird in Numerik II benutzt
IId	:	wird diskutiert

Zur Zeit haben Computer (digitale Rechenanlagen) wegen ihrer Leistungsfähigkeit und Vielseitigkeit eine herausragende Stellung eingenommen, und wir werden uns im folgenden nur mit diesen Rechenhilfsmitteln beschäftigen.

Die heutigen Rechenanlagen sind sehr leistungsfähige Rechenhilfsmittel, die numerische Rechnungen mit hoher Genauigkeit und Geschwindigkeit durchführen können. Mit Hilfe von Sprachen, wie FORTRAN, PASCAL, C usw. ist es möglich, numerische Rechnungen auf einer Rechenanlage durchzuführen, ohne genaue Kenntnisse über diese Rechenanlage zu besitzen. Es ist trotzdem wichtig, sich mit den Eigenschaften der vorhandenen Rechenanlagen vertraut zu machen, und zwar aus folgenden Gründen:

Lochkartengeräte usw.	v
Mechanische Tischrechenmaschinen	v
Elektrische Tischrechenmaschinen	v
Taschenrechner	Ib
Microcomputer	Ib
Großrechner	Ib
Spezialrechner	IId
Superrechner (Vektorrechner oder Parallelrechner)	IId

Tabelle 4.1: Digitale Rechenhilfsmittel

Rechenschieber usw.	v
Nomogramme	v
Mechanischer Analogrechner	v
Elektronischer Analogrechner	IId
Spezialgeräte (Mechanisch): Planimeter	v
Graphisches Papier	v
Graphischer Bildschirm	Ib
Graphischer Plotter	Ib

Tabelle 4.2: Analoge Rechenhilfsmittel

1. Es besteht ein enger Zusammenhang zwischen den vorhandenen Rechenhilfsmitteln und den numerischen Verfahren, die dafür geeignet sind. Es gibt z.B. Verfahren, die seit langem bekannt sind, aber nicht benutzt wurden, da sie für die vorhandenen Rechenanlagen ungeeignet waren. Sie werden aber jetzt auf dem neuesten Superrechner angewandt.

2. Trotz der Leistungsfähigkeit heutiger Computer gibt es viele Probleme, deren Anforderungen die Fähigkeiten dieser Computer überragen, so daß es notwendig ist, die Computer bestmöglichst zu benutzen:

   (a) Die Voraussetzungen für die Berechnung der Strömung durch einen Verdichter innerhalb einer Stunde werden in Abbildung 4.1 dargestellt (G. Johnson, *Computational Aspects of Internal Flow Problems*, SIAM News 17 No.3: 12-13(1984)).

   (b) i. Eine $n \times n$-Matrix erfordert Speicherplatz für n^2 Zahlen.

   ii. Die Lösung des Gleichungssystems $Ax = b$ erfordert $O(\frac{n^3}{3})$ Multiplikationen und $O(\frac{n^3}{3})$ Additionen.

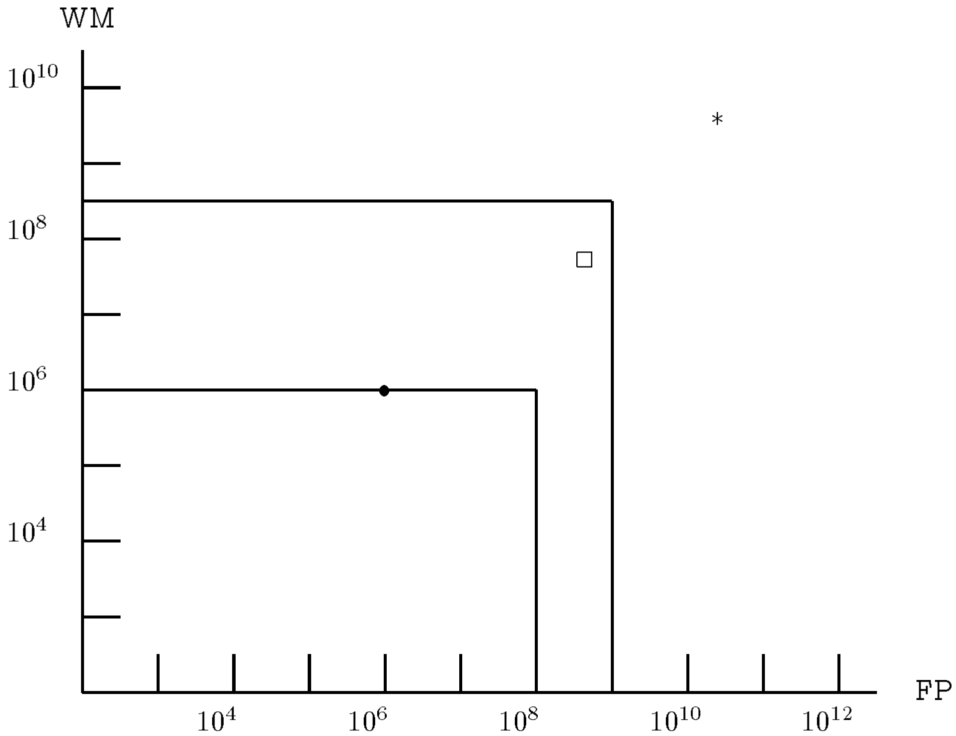


Abbildung 4.1: Voraussetzung für die Berechnung der Strömung durch einen Verdichter

WM = Words of Memory
FP = Floating-Point Operations per Second
• = Reibungslose Euler-Gleichungen
□ = Navier-Stokes-Gleichungen mit Turbulenz
$*$ = Navier-Stokes-Gleichungen mit Wirbeln

3. Obwohl die heutigen Rechenanlagen mit hoher Genauigkeit rechnen können, entstehen trotzdem kleine Rundungsfehler, die mit katastrophalen Auswirkungen fortgepflanzt werden können, wenn ungeeignete Verfahren benutzt werden.

4. Selbst in FORTRAN oder PASCAL ist es möglich, Programme zu schreiben, die wegen mangelhafter Berechnungen falsche Ergebnisse ohne Warnung liefern. Einige Beispiele werden später gegeben.

4.2 Rechnereigenschaften

Es gibt eine Vielfalt von Rechnern mit sehr unterschiedlichen Eigenschaften. Vom Standpunkt der Numerik aus sind folgende Merkmale besonders wichtig:

```
      program timing
c     ermittelt durchschnittsrechenzeiten für die arithmetischen Operationen
      parameter (n=1000,m=1000)
      dimension a(0:n),z(m)
      real timearray(2)
      integer time1,time2,time
      do 100 i=0,n
      z(i)=cos(float(i))
100   a(i)=sin(float(i))
      timediff=dtime(timearray)
      time1=time()
      do 300 j=1,m
      do 200 i=1,n
200   z(i)=a(i)*a(i-1)+z(j)
      a(j)= z(j)*a(i-1)+z(j)
300   continue
      time2=time()
      print *,(time2-time1),
     *       ' Zeit für 1 Operation (Mikrosek) '
      timediff=dtime(timearray)
      print *,timediff,'timediff'
      stop
      end AUSGABE
18    Zeit für 1 Operation (Mikrosek)
      18.18000 timediff
```

Tabelle 4.3: Gleitkomma-Leistung der SUN 3/60

4.2.1 Rechnerische Leistung

Die rechnerische Leistung oder Geschwindigkeit läßt sich nicht durch eine einzige Zahl kennzeichnen, da die Leistung von der Aufgabe abhängt. Mehrere Programme sind entworfen worden, um die Leistungen in bestimmten Bereichen zu messen. Zwei solch bekannte *Benchmarks*, die die Leistung im wissenschaftlichen Bereich messen, sind LINPACK und WHETSTONE.

Es ist auch möglich, eine erste grobe Abschätzung der Leistung eines Rechners mit Hilfe eines einfachen FORTRAN Programms zu gewinnen. In Tabelle 4.3 wird so ein Programm und seine Ergebnisse für die SUN 3/60 gezeigt.

Aus Tabelle 4.3 läßt sich schließen, daß der SUN 3/60 Arbeitsplatzrechner (mit Benutzung des MC68881-Gleitkomma- Coprozessors und optimiertes FORTRAN-Code) eine

	Speicher	**FLOPS** (Floating point operations per second)
Taschenrechner	50 Zahlen	10
Microcomputer	256K bytes	100
Großrechner	8M bytes	1 Million (1 MFLOP)
Superrechner	4M bytes	800 Millionen (800 MFLOP)
(Cyber 205)		(4 Pipes)

Tabelle 4.4: Typische Rechnerkapazitäten (1985)

Gleitkomma-Addition und eine Gleitkomma-Multiplikation in etwa 18μs ausführen kann, was in etwa 50.000 Operationen pro Sekunde gleich 50K FLOPS gleich 0,05 MFLOPS entspricht.

4.2.2 Speicherkapazität

Die Speicherkapazität eines Rechners besteht aus einem schnellen Hauptspeicher und einem oder mehreren langsameren Massenspeichern (Festplatten-Laufwerke, Disketten-Laufwerke, Magnetbandsystem, usw.). Die Speicherkapazität wird in KB (Kilobytes), MB (Megabytes) oder GB (Gigabytes) ausgedruckt, wobei zu beachten ist, daß eine Gleitkommazahl üblicherweise entweder 4 Bytes oder 8 Bytes (Doppelgenauigkeit) erfordert.

4.2.3 Die Merkmale einiger Rechner

In den Tabellen 4.4 und 4.5 werden einige typische Merkmale von Rechnern im Jahre 1985, 1988 und 1990 gezeigt, woraus die rasche Entwicklung deutlich zu erkennen ist.

4.3 Die Darstellung reeller Zahlen

4.3.1 Einführung

In der Analysis werden mehrere Methoden benutzt, um reelle Zahlen darzustellen, die auch auf dem Rechner anwendbar sind:

1. **Symbolisch** (z.B. π oder $\sqrt{2}$)

2. **Menge der natürlichen Zahlen** $\mathbb{N}$ (z.B. $7 \qquad 49 \qquad 13 \cdot 2^5$)

3. **Menge der ganzen Zahlen** $\mathbb{Z}$ (z.B. $\quad -7$)

	Hauptspeicher	**FLOPS**
Taschenrechner	32KB	?
Personal Computer (Compaq 386)	4MB	?
Arbeitsplatzrechner (Sun 3/60)	4MB-23MB	0,05 MFLOPS
Superrechner (Cray 2, 4 Prozessoren)	$256 \times$ 8MB	2 GFLOPS (4 Prozessoren)
Connection CM-2 (65536 Prozessoren)	512MB	3,5 GFLOPS
Superrechner (ETA 10, 8cpus)	32M words	10 GFLOPS

Tabelle 4.5: Typische Rechnerkapazitäten (1988)

Typ	**Name**	**Addition** $[\mu s]$	**Multiplikation** $[\mu s]$
sparc station	fafnir	2.165	2.368
sparc station	elvis	2.310	2.588
4/100	polya	4.305	5.671
3/80	switek	10.296	10.970
3/60	sunny	13.080	13.934
3/260	trudy	13.600	14.776

Tabelle 4.6: Typische Rechenleistungen im Institut 1990 (doppelte Genauigkeit, compeliert mit gcc -g)

4. **Körper der rationalen Zahlen Q** (z.B. $-\frac{3}{7}$)
5. **Dezimalbrüche** (z.B. $\frac{1}{7} = 0,\overline{142857}$, $\pi = 3,14159\ldots$)

Mit Ausnahme der symbolischen Darstellung, die in Numerik II im Zusammenhang mit symbolischen Systemen wie REDUCE, MAPLE, usw. kurz diskutiert wird, benutzen alle Darstellungsmethoden Positionssysteme, die wir jetzt eingehend besprechen.

4.3.2 Positionssysteme

Definition 4.1 *Seien $d \in \mathbb{Z}$, $|d| > 1$ und $x \in \mathbb{R}$. Die* d-näre Darstellung *von x ist*

$$x = \pm \sum_{k=\ell}^{\infty} a_k d^{-k}\,,\ 0 \le a_k < |d|\,,\ \ell \in \mathbb{Z}\,,\ d \in \mathbb{Z}$$

Beispiele

1. $\mathbf{d = 10}$ (Dezimale Darstellung)

   (a) $1 = 1.000\ldots\,,$

   $$= \sum_{k=\ell}^{\infty} a_k 10^{-k}\,;\ \ell = 0\,,\ a_k = \begin{cases} 1 & , \text{ für } k = 0 \\ 0 & , \quad sonst \end{cases}$$

   (b) $\frac{\pi}{10} = 0,31415\ldots\,,$

   $$= \sum_{k=\ell}^{\infty} a_k 10^{-k}\,;\ \ell = 1\,,\ a_1 = 3\,,\ a_2 = 1,\ldots$$

2. $\mathbf{d = 2}$ (Lineare oder duale Darstellung)
   Hierbei werden oft die Ziffern 0 und 1 durch O und L bezeichnet, um Verwechslungen mit der Dezimaldarstellung zu vermeiden.

   $$\begin{aligned} 5 &= LOL \\ 3,2 &= LL\,, \overline{OOLL} \end{aligned}$$

3. $\mathbf{d = 16}$ (Hexadezimale oder Sedezimale Darstellung)
   Die „Ziffern" $10, 11, 12, 13, 14, 15$ werden mit A, B, C, D, E, F bezeichnet.

   $$29 = 1D\,.$$

4. $\mathbf{d = 60}$ (Sexagesimale Darstellung)
   Wenn Verwirrung möglich ist, wird die Basiszahl als untergeordnete Zahl in Buchstaben (B, D, O, H) oder in der Dezimaldarstellung hinzugefügt, z.B.:

   $$29_{10} = 29_{zehn} = 35_8 = 1D_H = 29_D = 35_O\,.$$

   Die d-näre Darstellung (1) wird oft folgendermaßen geschrieben:

   $$x = \pm\, a_\ell a_{\ell-1}\ldots a_o, a_{-1}a_{-2}\ldots\,,$$

   wobei das Trennzeichen Komma (angelsächsisch: Punkt) die Lage von a_o bestimmt.

   Eine verwandte Schreibweise (angelsächsisch: Scientific) ist die Schreibweise mit abgetrennten Zehnerpotenzen:

   $$x = \pm\, a_\ell, a_{\ell-1}a_{\ell-2}\ldots x\ d^\ell\,,$$

   wobei $1 \leq a_\ell < 10$ vorausgesetzt wird.
   Diese Schreibweise ist bei sehr großen oder kleinen Zahlen fast notwendig, z.B.

   $$\begin{aligned} \text{Avogadro Konstante} \quad N &= 6,02250 \times 10^{23} \\ \text{Planck Konstante} \quad \hbar &= 1,0545 \times 10^{-27} \end{aligned}$$

Wie das Beispiel $3,999\cdots = 3,\overline{9} = 4$ zeigt, ist die d-näre Darstellung einer reellen Zahl x nicht immer eindeutig. Es liegt nahe, im Zweifelsfall die endliche Darstellung zu nehmen.

In der Analysis wird die d-näre Darstellung oft die d-adische Darstellung genannt. Jeder b-adische Bruch konvergiert (Forster I, S. 29) und jedes $x \in \mathbb{R}$ läßt sich in einen b-adischen Bruch entwickeln (Forster I, S. 30).

Es gibt auch andere Möglichkeiten:

5. **Gemische d-näre Darstellungen**

   (a) Das alte englische Währungssystem:

   1 Pound (\$) = 20 Shillings
   1 Shilling (s) = 12 Pennies
   1 Penny (d) = 4 Farthings,

   so daß \$ $1/3/6\frac{1}{2} = \$\left(1 + \frac{3}{20} + \frac{6\frac{1}{2}}{240}\right)$.

   (b) Die Zeitrechnung: Jahrzehnt, Jahr, Tag, Stunde, Minute, Sekunde.

   (c) Die römische Darstellung:

   1984 = MCMLXXXIV.

   (d) Darstellung durch Kettenbrüche.

Weitere Information finden Sie bei: D. Knuth, Seminumerical Algorithms, Kapitel 4.

Da die Mathematik und Technik heute ausschließlich die Dezimaldarstellung benutzt, werden wir es selbstverständlich auch tun.

4.4 Zahldarstellung auf einem Rechner

Für die Zahldarstellung auf einem Rechner gibt es zwei wesentliche Entscheidungen:

1. Eine interne Basiszahl d muß gewählt werden.

2. Vereinbarungen über die Menge A der maschinendarstellbaren Zahlen, die sog. *Maschinenzahlen*, müssen getroffen werden.

Diese beiden Entscheidungen werden jetzt weiter besprochen.

4.4.1 Wahl der internen Basiszahl d

Bei Taschenrechnern ist die interne Basiszahl d meistens 10. Bei Großrechnern dagegen ist die interne Basiszahl meistens $d = 2$ oder $d = 16$.

Es wäre für den Benutzer leichter, wenn die interne Basiszahl $d = 10$ wäre. Dagegen spricht nur die Tatsache, daß alle Hardware binäre elektronische Schaltungen benutzt, so daß Arithmetik und Speicher für $d = 2^k$ billiger sind.

Da man bei der Eingabe und Ausgabe eines Rechners meistens die Dezimaldarstellung benutzt, ist die interne Darstellung für den Benutzer größtenteils unsichtbar. Es ist aber immer darauf zu achten, daß Dezimalzahlen nicht immer exakt intern gespeichert werden können, wenn die interne Basiszahl nicht 10 ist. Daß dadurch unerwartete Ergebnisse vorkommen können, wird später demonstriert (Absatz Rechnerarithmetik).

Andere Basiszahlen wurden öfter vorgeschlagen (Kahan [1973]), aber zur Zeit werden nur $d = 2$, $d = 16$ und $d = 10$ ernsthaft betrachtet (Cody and Waite [1980, S. 6]).

4.4.2 Maschinenzahlen

In einer Rechenanlage gibt es nur beschränkten Speicherplatz. Es ist deshalb nicht möglich, beliebige Zahlen exakt zu speichern, weil dies im allgemeinen unbeschränkten Speicherplatz erfordert.

Es gibt mehrere Möglichkeiten, dieses Problem zu lösen:

1. Zahlen können durch 'Intervallzahlen' approximiert werden, wie beim ACRITH-System. Diese Idee wird in Numerik II behandelt.

2. Die Berechnungen können ohne Approximation durchgeführt werden, wobei die Anzahl der Ziffern schnell wachsen kann, so daß die Berechnungen wegen Mangel an Speicherplatz abgebrochen werden müssen - wie bei symbolischen Systemen wie REDUCE, die ebenfalls in Numerik II diskutiert werden.

3. Es kann vereinbart werden, daß jeder Zahl ein beschränkter Speicherplatz zugeordnet wird, so daß nicht alle Zahlen exakt darstellbar sind.

Definition 4.2 *Eine Maschinenzahl ist eine Zahl, die unter Berücksichtigung der vorhandenen Vereinbarungen in einer Maschine exakt darstellbar ist.*

In herkömmlichen Rechenanlagen werden Maschinenzahlen als d-näre Zahlen mit Exponent dargestellt:

$$x = \pm \sum_{k=1}^{t} a_k d^{-k} \times d^e \quad , \quad 0 \le a_k < d \, ,$$

wobei a_k und e ganz sind. e heißt der Exponent. Es gibt mehrere spezielle Datentypen, die vorkommen:

1. **Ganze Zahlen (Integers)**

$$x = \pm \sum_{k=1}^{t} a_k d^{-k} \times d^t = \pm \sum_{k=1}^{t} a_k d^{t-k} \quad .$$

2. **Festkommazahlen (Fixed Point)**

$$x = \pm \sum_{k=1}^{t} a_k d^{-k} \times d^q \,, q \quad \text{fest} \,.$$

3. **(Normierte) Gleitkommazahlen ((Normalized) Floating Point)**

$$x = \pm \sum_{k=1}^{t} a_k d^{-k} \times d^e = m \times d^e \quad , a_1 \neq 0$$

   wobei

$$m = \pm \sum_{k=1}^{t} a_k d^{-k}$$

   die *Mantisse* heißt.

4. **(Normierte) Gleitkommazahlen mit Doppelgenauigkeit**

5. **(Normierte) Gleitkommazahlen mit erweiterter Genauigkeit**

6. **Gleitkommazahlen** $+0, -0$
   Diese Zahlen werden oft folgendermaßen dargestellt:

$$0 = \sum_{k=1}^{t} 0 \, d^{-k} \,, \text{d.h.} \quad a_k = 0 \quad \text{für} \quad 1 \leq k \leq t \,.$$

7. **nichtnormierte Gleitkommazahlen**
   Diese Zahlen werden wie normierte Gleitkommazahlen dargestellt, aber die Bedingung $a_1 \neq 0$ wird nicht erforderlich.

8. **Sonderzahlen**
   $+\infty \,, -\infty \,, \quad NaN$ (Not a Number) laut, NaN ruhig.

Es bleibt noch festzustellen, wie die Maschinenzahlen gespeichert werden sollen:

1. **Größe des Speicherbereichs**
   Obwohl Rechner, wie z.B. der Univac 1108, eine Wortlänge von 36 Bits hatten, ist es heute üblich, die Speicher in Bytes von je 8 Bits aufzuteilen. Die Speichergröße

	BASIC	FORTRAN	PL1	PASCAL	C	MODULA
Ganze Zahlen	-	Integer*n	Fixed Decimal(t)	Integer		
Festkomma	-	-	-Fixed Decimal(t,t-q)	- -		
Gleitkomma	Implizit	Real*4	Float Decimal(t)	Real		
Gleitkomma (Doppelgenauigkeit)	-	Double Precision	Float Decimal(t)	-		
Gleitkomma (erw. Genauigkeit)	-	Real*16	Float Decimal(t)	-		

Tabelle 4.7: Datentypen für Maschinenzahlen

für verschiedene Maschinenzahlen in verschiedenen Sprachen wird in Tabelle 4.7 dargestellt.

Diese speziellen Datentypen werden in PASCAL, FORTRAN, PL1, C, MODULA und BASIC definiert.

2. **Darstellung von negativen Zahlen**
   Drei Methoden werden verwendet, um das Vorzeichen V einer Festkommazahl darzustellen:

   (a) Das Vorzeichen V wird als ein Bit gespeichert.

   (b) Im Fall $d = 2$ kann eine negative Mantisse m als die komplementäre Zahl $\overline{m} = K - |m|$ dargestellt werden, wobei

   $$\begin{aligned} K &= 2 - 2^{-t} \qquad &\text{(Einer-Komplement)} \\ K &= 2 \qquad &\text{(Zweier-Komplement)} \end{aligned}$$

   und nur eine Stelle links vom Komma behalten wird.

Die zwei Methoden unterscheiden sich, wie in 4.8 angegeben.

Mit leichten Änderungen werden diese Methoden auch benutzt, um negative ganze Zahlen darzustellen.

Bemerkung: Im Fall $d = 10$ wurde Komplement-Arithmetik auf Tischrechnern benutzt, was sehr anschaulich war.

Bemerkung: Bei Gleitkommazahlen ist es heute üblich, das Vorzeichen getrennt zu speichern, aber auf dem Univac 1108 wurden Komplemente gebildet.

	Einer-Komplement	**Zweier-Komplement**
Darstellung von 0	$0,000\ldots0$	$0,00\ldots0$
	$1,11\ldots1$	
Größte Zahl	$1-2^{-t}$	$1-2^{-t}$
Kleinste Zahl	$-(1-2^{-t})$	-1
z.B. für $d=2$ und $t=3$:		
Darstellung von	**Einer-Komplement**	**Zweier-Komplement**
	$(K=1,111_B)$	$(K=10,000_B)$
0	$0,000$ oder $1,111$	$0,000$
Größte Zahl	$+0,111 \quad (=7/8)$	$+0,111 \quad (=7/8)$
Kleinste Zahl	$1,000 \quad (=-0,111_B)$	$1,000 \quad (=-1)$
$-0,101_B=-\frac{5}{8}$	$1,010$	$1,011$

Tabelle 4.8: Darstellung von Festkommazahlen in Einer- und Zweier-Komplement Arithmetik

3. **Darstellung des Exponenten** Für den Exponenten e und das Vorzeichen V in der Darstellung
$$x=\pm\sum_{k=1}^{t}a_kd^{-k}\times d^e$$
ist es üblich, eine feste Anzahl von d-Stellen, die wir mit E bezeichnen, zu benutzen. Man definiert die Charakteristik C
$$C:=C_1+e\,,\quad \text{wo} \quad C_1 \quad \text{eine Konstante ist}$$
und setzt voraus, daß C eine nicht negative ganze Zahl ist.

Eine typische interne Hardware Darstellung von
$$x=\pm\sum_{k=1}^{t}a_kd^{-k}\times d^e=m\times d^e$$
ist deshalb:

	E-Stellen	**t-Stellen**
V	Charakteristik	Mantisse

Bemerkung: Auf einigen Rechnern, z.B. den Burroughs, wurde das Komma hinter a_t statt vor a_1 plaziert.

4.4.3 Ein Beispiel: Der IEEE P754 Standard für Gleitkommazahlen

Dieser Standard wird jetzt häufiger benutzt. Es werden normierte binäre „scientific" Gleitkommazahlen verwendet.

Einfache Genauigkeit

s	e	f
0	8	31

Die Felder eines 4 Byte Wortes sind:

1. Das Vorzeichen s (1 Bit)
2. Die Charakteristik e (8 Bits)
3. Die Mantisse f (23 Bits)

Der Wert von v ist:

1. Wenn $e = 255$ und $f \neq 0$, dann ist $v = NaN$
2. Wenn $e = 255$ und $f = 0$, dann ist $v = (-1)^s \infty$
3. Wenn $0 < e < 255$, dann ist $v := (-1)^s\, 2^{e-127}(1, f)$
4. Wenn $e = 0$ und $f \neq 0$, dann ist $v := (-1)^s\, 2^{-126}(0, f)$
5. Wenn $e = 0$ und $f = 0$, dann ist $v = (-1)^s 0$.

Doppelgenauigkeit

s	e	f
0	11	63

Die Felder eines 8-Byte Wortes sind:

1. Das Vorzeichen s (1 Bit)
2. Die Charakteristik e (11 Bits)
3. Die Mantisse f (52 Bits)

Der Wert v ist:

1. $e = 2047$ und $f \neq 0 \Longrightarrow v = NaN$
2. $e = 2047$ und $f = 0 \Longrightarrow v = (-1)^s \infty$
3. $0 < e < 2047 \Longrightarrow v = (-1)^s\, 2^{e-1023}(1, f)$
4. $e = 0$ und $f \neq 0 \Longrightarrow v = (-1)^s\, 2^{-1022}(0, f)$
5. $e = 0$ und $f = 0 \Longrightarrow v = (-1)^s 0$.

4.5 Rechnerarithmetik

Jede Rechenanlage besitzt ein Rechenwerk oder mehrere Rechenwerke, in denen die vier arithmetischen Operationen *,/,+,- durchgeführt werden.

Ein Rechenwerk muß mindestens ein 1-stelliges Addierwerk enthalten (wie bei Menschen). Es ist möglich, viel schnellere Rechenwerke zu bauen (siehe z.B. Spaniol [1976]). Die genauen Einzelheiten der Rechenwerke sind für die numerische Mathematik uninteressant. Die wichtigen Merkmale sind:

1. die Rechenzeit für die vier arithmetischen Operationen,
2. die Behandlung von Überlauf und Unterlauf,
3. Rundung und Schneiden,
4. die arithmetischen Operationen.

Nur die Ergebnisse sind von Interesse. Ob sie mit Hardware oder in Zusammensetzung von Hardware und Software erreicht werden, ist unwichtig.

4.5.1 Die Rechenzeit

Die Rechenzeit für die vier arithmetischen Operationen ist von den Operanden abhängig, z.B. Multiplikation mit 0 ist oft sehr schnell. Durchschnittswerte und Maximalwerte sind manchmal vom Hersteller der Rechenanlage gegeben. Sie lassen sich auch oft leicht feststellen, z.B auf dem FX-720P Taschenrechner sind die Laufzeiten der folgenden zwei Programme gemessen worden:

10 REM TIMING	10 REM TIMING
20 N = 1000	20 N = 1000
30 S = 0	30 S = 0
40 FOR K = 1 TO N	40 FOR K = 1 TO N
50 S = S* · 9845362162	50 S = 0
60 NEXT K	60 NEXT K
70 PRINT “ZEIT”	70 PRINT “ZEIT”
80 STOP	80 STOP
Laufzeit: 37 Sekunden	**Laufzeit:** 22 Sekunden

woraus sich feststellen läßt, daß eine Gleitkomma-Multiplikation in BASIC ungefähr

$$15\,ms = 1.5 \times 10^{-2} \text{ Sekunden}$$

Addition:	10ns
Multiplikation:	10ns
gekoppelte Addition und Multiplikation:	10ns
Division:	$20/0{,}32 \approx 62{,}5$ns

Tabelle 4.9: Rechenzeiten auf einem 2-Pipe Cyber 205 für 64-bit-Zahlen (1ns = 1 nanosekunde = 1×10^{-9} Sekunden)

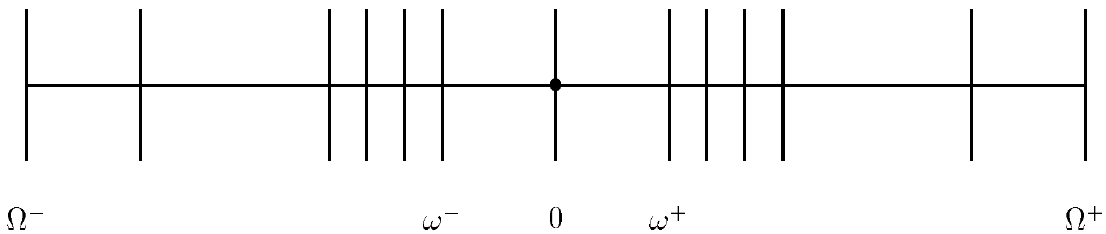


Abbildung 4.2: Die Menge der Maschinenzahlen

benötigt.

Die Rechenzeit ist ein sehr wichtiger Parameter, da Probleme mit $0(10^9)$ Operation heute keine Seltenheit sind. Typische Rechenzeiten sind in Tabelle 4.9 gegeben.

Bei einfachen Mikrocomputern ist die Zeit für eine Multiplikation wesentlich größer als die Zeit für eine Addition, und die Zeit für eine Division ist noch größer. Deshalb werden Algorithmen mit weniger Divisionen und Multiplikationen bevorzugt. Auf Großrechnern ist der Unterschied oft geringer, da hochentwickelte Rechenwerke benutzt werden. Auf dem Superrechner Cyber 205 dagegen ist Division um einen Faktor 6 langsamer als Addition und Multiplikation. (Siehe Tabelle 4.9.)

4.5.2 Überlauf und Unterlauf

Die Menge A von Maschinenzahlen läßt sich graphisch darstellen.

Die Maschinenzahlen bilden eine endliche Teilmenge A einer begrenzten Menge $\tilde{A}$ der reellen Zahlen

$$A \subset \tilde{A} := [\Omega^-, \omega^-] \cup \{0\} \cup [\omega^+, \Omega^+] \subset \mathbb{R} \quad ,$$

wobei Ω^- bzw. Ω^+ die kleinsten negativen und größten positiven Maschinenzahlen und

ω^- und ω^+ die größten negativen und kleinsten positiven Maschinenzahlen sind. z.B. für den IBM 4381 gilt:

Integer*4 Maschinenzahlen:

$$\Omega^- = -2^{31}\,,\ \omega^- = -1\,,\ \omega^+ = 2^{31} - 1\,.$$

Real*4 Maschinenzahlen:

$$\begin{aligned}\Omega^+ &= -\Omega^- = 16^{63} \\ \omega^+ &= -\omega^- = 16^{-65}\end{aligned}$$

Während einer Berechnung ist es durchaus möglich, daß Zwischenwerte außerhalb $\tilde{A}$ liegen, d.h. $x, y \in \tilde{A}$, aber $z := x \Box y$ und $z \notin \tilde{A}$. Die folgenden Möglichkeiten treten auf:

$$\begin{aligned}\text{Unterlauf} &: \ \omega^- < z < 0 \quad \text{oder} \quad 0 < z < \omega^+ \\ \text{Überlauf} &: \ z < \Omega^- \quad \text{oder} \quad z > \Omega^+\,.\end{aligned}$$

Bei ganzen Zahlen kommt ein Unterlauf nicht vor. Bei Gleitkommazahlen kommen sowohl Unterlauf als auch Überlauf vor und werden als Exponentenunterlauf und Exponentenüberlauf bezeichnet.

Überlauf von ganzen Zahlen
Bei den ganzen Zahlen wird der Überlauf durch den Rechner oder Compiler nicht kontrolliert. Die Ausgabe des Programms Arith1 läßt sich durch Überlauf erklären.

Beispiel: Programm ARITH1

```
Program arith1  (input, output);
var i,n,jp,jf : integer;
begin
  n := 20;
  i := 0; jp := 1; jf := 1;
  repeat
    i := i + 1; jp := jp + jp; jf := jf * i;
    writeln (i:2, '  ', jp:7, '  ', jf : 7)
  until i = n;
end.
```

Die Ausgabe auf dem PC ist:

i	jp	jf
1	2	1
2	4	2
3	8	6
4	16	24
5	32	120
6	64	720
7	128	5040
8	256	-25216
9	512	-30336
10	1024	24320
11	2048	5376
12	4096	-1024
13	8192	-13312
14	16384	10240
15	-32768	22528
16	0	-32768
17	0	-32768
18	0	0
19	0	0
20	0	0

Es ist unsere Aufgabe, in diesem Abschnitt solche unerwarteten Ergebnisse zu verstehen und zu vermeiden.

1. Für $i = 14$ ist $jp = 16384 = 2^{14}$. Die Zuweisung $jp := jp + jp$ bewirkt, daß die Addition $2^{14} + 2^{14}$ durchgeführt wird. Dies wird richtig gemacht, außer wenn das führende Bit 2^{15} nicht richtig gespeichert wird, weil das Register nur 16 Stellen hat. Stattdessen wird dieses Bit als Vorzeichen interpretiert.

2. Für $i = 7$ ist $jf = 5040$. Die Zuweisung $jf = i * jf$ bewirkt, daß die Multiplikation 5040×8 durchgeführt wird.

$$5040 \times 8 = 40320 = -25216 + 2^{16}.$$

   Die berechneten Werte von jf sind immer gleich den richtigen Werten modulo $2^{16} = 65536$, wobei zuerst der Betrag und danach das Vorzeichen berechnet wird.

Andere Fälle, wo der Überlauf bei ganzen Zahlen nicht kontrolliert wird: IBM 4381 bei FORTRAN und PASCAL.

Exponentenunterlauf

Sprache	**Ergebnis**	**Folgen**
Pascal (Rainbow)	0	Programm läuft weiter
Fortran und PL1 (4381)	0	Programm läuft weiter mit Fehlermeldung

Exponentenüberlauf

Sprache	**Ergebnis**	**Folgen**
Pascal (Rainbow)	-	Programm unterbrochen
Fortran und PL1 (4381)	$\pm\Omega$	Programm läuft weiter mit Fehlermeldung

Bemerkung:

1. Fortran und PL1 haben Prozeduren, mit denen der Benutzer Exponentenüberlauf und -unterlauf ermitteln und korrigieren kann.

2. Im IEEE Standard 754 werden spezielle Maschinenzahlen $+\infty$ und $-\infty$ definiert zur Benutzung beim Exponentenüberlauf.

4.5.3 Rundung und Schneiden

A sei die Menge der Gleitkommamaschinenzahlen mit Basiszahl d und Mantissenlänge t.

Während einer Berechnung ist es durchaus möglich, daß Zahlen auftreten, die zu $\tilde{A}$, aber nicht zu A gehören, und zwar aus zwei Gründen:

1. Problemparameter wie π, die nicht zu A gehören.

2. Wenn $x, y \in A$ und $\square$ eine arithmetische Operation $+ - /$ oder $\times$ bezeichnen, ist es möglich, daß $x \square y \notin A$. A ist also nicht abgeschlossen bezüglich der arithmetischen Operation. Zum Beispiel

$$\begin{aligned} d = 10 \quad &, \quad t = 2 \\ 1,0/3,0 \quad &= \quad \frac{1}{3} = 0,\overline{3} \notin A \\ 1,1 \times 1,1 \quad &= \quad 1,21 \notin A \\ 1,0 + 0,01 \quad &= \quad 1,01 \notin A\,. \end{aligned}$$

Es ist deshalb nötig, solche Parameter oder Zwischenergebnisse auf A abzubilden.

Definition 4.3 *Eine Abbildung $rd : \mathbb{R} \longrightarrow A$ heißt Rundung, falls*

$$|rd(x) - x| \leq \min_{a \in A} |a - x|\,. \qquad \square$$

Beispiel 1: $d = 10$

1. $x = 0\,,\ rd(x) := 0$
2. $x \neq 0\,,\ x = \pm 0\,,\ a_1 a_2 \ldots \times 10^e\,,\ a_1 \neq 0\,,$

$$rd(x) := \pm\tilde{a} \times 10^e$$

$$\text{mit} \quad \tilde{a} := \begin{cases} 0\,, \quad a_1 \ldots a_t\,, & \text{falls} \quad a_{t+1} \leq 4 \\ 0\,, \quad a_1 \ldots a_t + 10^{-t}\,, & \text{falls} \quad a_{t+1} \geq 5 \end{cases}$$

Für $d = 10\,,\ t = 4$:

$$\begin{aligned} \pi &= 3,14159\ldots\,, rd(\pi) = 0,3142 \times 10^1 \\ \sqrt{57} &= 7,5498\ldots\,, rd(\sqrt{57}) = 0,7550 \times 10^1 \\ \frac{1}{1,00001} &= 0,9999900001\ldots \times 10^o\,,\ rd\left(\frac{1}{1,00001}\right) = 0,1000 \times 10^1 \end{aligned}$$

Beispiel 2: $d = 2$

1. $x = 0\,,\ rd(x) := 0$
2. $x \neq 0\,,\ x = \pm 0\,,\ a_1 a_2 \ldots \times 2^e\,,\ a_1 = L$

$$rd(x) := \pm\tilde{m} \times 2^e$$

$$\text{mit} \quad \tilde{m} := \begin{cases} 0\,, \quad a_1 a_2 \ldots a_t\,, & \text{falls} \quad a_{t+1} = 0 \\ 0\,, \quad a_1 a_2 \ldots a_t + 2^{-t}\,, & \text{falls} \quad a_{t+1} = L \end{cases}$$

Für $d = 2\,,\ t = 3$:

$$\begin{aligned} x &= 0,1_D = O,LLOOLLOO\ldots \times 2^{-3} \\ rd(x) &= O,LLO \times 2^{-3} \end{aligned}$$

Bemerkung:

1. Die Abbildung rd ist nicht eindeutig definiert, da in einigen Fällen zwei Maschinenzahlen a und a' existieren mit $a < a'$ und $|x - a| = |x - a'|$. Z.B. für $d = 10\,,\ t = 2\,,\ x = 0,125$, ist $a = 0,12$ und $a' = 0,13$. Siehe Kahan (1973, Kapitel 2).

   Im IEEE Standard 754 wird in solchen Fällen so gerundet, daß a_t eine gerade Zahl ist.

2. In den Beispielen ist das Ergebnis nicht in normalisierter Gestalt, wenn $|\tilde{m}| = 1$.

Definition 4.4 *Die Abbildung* schn : $\mathbb{R} \longrightarrow A$ *heißt* Abschneiden *oder Abrundung (englisch: chopping, truncation):*

$$\begin{aligned} \mathrm{schn}(x) &= \sup\{a \in A : a \leq x\}\,, \textit{falls} \quad x \geq 0 \\ \mathrm{schn}(x) &= \inf\{a \in A : a \geq x\}\,, \textit{falls} \quad x < 0 \end{aligned}$$

Beispiel: $t = 4$, $d = 10$

$$\mathrm{schn}(\pi) = \mathrm{schn}(3,14159\ldots) = 3,141\,.$$

4.5.4 Maschinenoperationen - Ganzzahlige Arithmetik

Berechnungen mit ganzen Zahlen werden immer exakt durchgeführt, wobei die Division speziell definiert werden muß:

PASCAL:	123	div	4	⟶ 30
	123	/	4	⟶ 30,75
FORTRAN:	123	/	4	⟶ 30
	123.	/	4	⟶ 30,75
	123	/	4.	⟶ 30,75

Für PL1 ist die Division durch ganze Zahlen etwas kompliziert. Z.B. der Ausdruck $25+1/3$ ist nicht definiert. (PL1 Handbuch, Seite 43.)

Wie schon erwähnt, wird oft Überlauf und Unterlauf bei ganzen Zahlen nicht berücksichtigt.

Gleitkommaarithmetik
Wie schon erwähnt und durch Beispiele erläutert, sind die arithmetischen Operationen $+, -, /$ und $\times$ nicht in A abgeschlossen.

Definition 4.5 *Seien* $x, y \in A$, $\Box \in \{\times, /, +, -\}$ *und* $x \Box y \in \tilde{A}$. *Dann wird* $\Box^*$ *folgendermaßen definiert:*

$$x \Box^* y := rd(x \Box y)\,.$$

Diese Arithmetik ist Bestandteil von IEEE Standard 754. Daß sie bisher auf fast allen Rechnern nicht vorhanden war, liegt daran, daß sie drei zusätzliche Ziffern erfordert (siehe Abbildung 4.3) was die angegebenen Beispiele deutlich machen.

Beispiele: $t = 2$, $d = 10$, Rundung.

1.

$$\begin{array}{r|l} t & KR \\ 0,15 \times 0,19 = 0,02 & 85 \\ 0,15 \,\Box^*\, 0,19 = 0,29 \times 10^{-1} & \end{array}$$

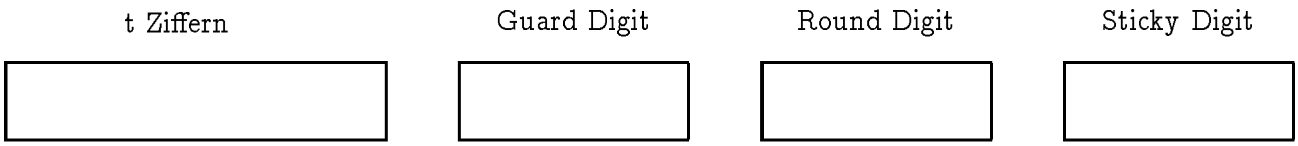


Abbildung 4.3: Die Kontroll-, Rundungs- und Klebrige-Ziffern

2. $0,10 - 0,011 = 0,089$
   Ohne die Kontrollziffer wird falsch abgeschnitten:

$$\begin{array}{rl|l} & 0{,}10 & \\ - & \mathbf{0{,}01} & 1 \\ & \mathbf{0{,}09} & \end{array}$$

3. $0,10 - 0,099 = 0,001$
   Ohne die Kontrollziffer erhalten wir $0,01$, was um einen Faktor 10 zu groß ist.

$$\begin{array}{rl|l} & 0{,}10 & 00 \\ - & \mathbf{0{,}09} & 90 \\ & \mathbf{0{,}01} & \end{array}$$

4. $0,10 - 0,1 \cdot 10^{-20}$
   Ohne eine Klebrige Ziffer wird falsch abgeschnitten.

Auf den vorhandenen Rechnern gilt:

1. Auf dem IBM 4381 werden Kontrollziffern benutzt.

2. Auf dem Rainbow 100 wird eine Kontrollziffer bei Multiplikation, aber nicht bei Addition benutzt. Dies führt dazu, daß große relative Fehler bei Addition entstehen können:
   $$x +^* y = x\rho_1 + y\rho_2 \,, \text{ wobei } \quad |\ell n\, \rho_1| \,, \, |\ell n\, \rho_2| \le eps \,;$$
   wie das folgende Programm zeigt:

```
program numer6(input,output);
var xg,xp,xn,xd,xpm1,xpmid:real;
var i:integer;
begin
   writeln(lst,'program numer6');
   xg:=2;i:=1;
   repeat
```

```
          xg:=2*xg; i:=i+1;
          until i=39;
      xp:=xg+xg; xn:=xg+(xg-1); xpm1:=xp-1;
      xd:=xp-xn; xpm1d:=xp-xpm1;
      writeln(lst,'xg=',xg:20);
      writeln(lst,'xp=',xp:20);
      writeln(lst,'xn=',xn:20);
      writeln(lst,'xd=',xd:20);
      writeln(lst,'xpm1',xpm1:20);
      writeln(lst,'xpm1d',xpm1d:20);
      end.
```

mit Ausgabe

```
xg=    5.4975581389E+11
xp=    1.0995116278E+12
xn=    1.0995116278E+12
xd=    2.0000000000E+00
xpm1     1.0995116278E+12
xpm1d     0.0000000000E+00
```

4.6 Die automatische Bestimmung der Merkmale eines Rechners

Es ist möglich, unter bestimmten Voraussetzungen ein Programm zu schreiben, das die Merkmale eines Rechners automatisch feststellen kann.

Das Programm MACHAR ist das am besten bekannte Programm dieser Art. Das Programm envron, ein kleiner Ausschnitt aus MACHAR, kann die Werte von t und d bestimmen und auch feststellen, ob Rundung oder Schneiden benutzt wird. Das Programm mit seiner Ausgabe auf dem Rainbow 100 ist:

```
program envron(output);
var one,zero,a,b,beta,betam1:real;
var ibeta,it,irnd:integer;
begin
   writeln(lst,'program envron');
   one:=1; zero:=0;   a:=one;
   repeat
      a:=a+a;
   until (((a+one)-a) - one)<> zero;
   b:=one;
   repeat
```

```
      b:=b+b;
   until ((a+b)-a)<>zero;
   ibeta:=trunc((a+b)-a);    beta:=ibeta;
   it:=0;    b:=one;
   repeat
      it:=it+1; b:=b*beta;
      until (((b+one)-b)-one) <> zero;
   irnd:=0;betam1:=beta-one;
   if ((a+betam1)-a) <> zero   then irnd:=1;
   writeln(lst,'a=',a,'betam1=',betam1);
   writeln(lst,'Basiszahl ibeta =',ibeta);
   writeln(lst,'Anzahl digits it=',it);
   if irnd=0 then writeln(lst,'Addieren mit Abrundung');
   if irnd=1 then writeln(lst,'Addieren mit Rundung');
end.

program envron
a= 1.0995116278E+12betam1= 1.0000000000E+00
Basiszahl ibeta =2
Anzahl digits it= 40
Addieren mit Abrundung
```

4.7 Die Merkmale von einigen Rechnern und Sprachen in Münster

4.7.1 IBM 4381 FORTRAN

Bemerkenswert bei IBM FORTRAN ist, daß das EQUIVALENCE Statement es möglich macht, einen Speicherbereich für mehrere Datentypen, z.B. Gleitkomma und Hexadezimal, gleichzeitig zu benutzen.

```
C     PROGRAMM NUMER3
      REAL*4 X
      INTEGER*4 I
      CHARACTER*1  AX(4), AI(4)
      EQUIVALENCE (AX,X), (AI,I)
      X = -29.
      I = -29
      DO 10 J = 1,4
      KX = ICHAR(AX(J))
      KI = ICHAR(AI(J))
```

```
      PRINT 6, KX, KI
  6   FORMAT (2I5)
 10   CONTINUE
      END
```

mit Ausgabe (auf dem IBM 4381):

```
66(42)     255(FF)
29(1D)     255(FF)
 0(00)     255(FF)
 0(00)     227(E3)
```

wobei die Zahlen in den Klammern die äquivalente Hexadezimale Zahl angeben (VS Fortran, Handbuch, GC 26-3986-3, 1983, Seiten 272-277). Es folgt:

$$
\begin{array}{lllllllll}
-29. & \sim & 1|1000010|0001 & 1101 & 0000 & 0000 & 0000 & 0000 & \\
 & & V \quad\; C & M & & & & & \\
-29 & \sim & 1111 \quad 1111 & 1111 & 1111 & 1111 & 1111 & 1110 & 0011
\end{array}
$$

Real*4 Zahlen werden also als hexadezimale Zahlen mit Vorzeichen gespeichert. Negative Integer*4 Zahlen werden als „Zweier"-komplimentäre Zahlen gespeichert.

Positive Real*4 Zahlen müssen deshalb zwischen

$$16^{-65} \approx 5.3 \times 10^{-79} \quad \text{und} \quad 16^{+63} \approx 7,23 \times 10^{+75}$$

liegen. Die Konstante C_1 ist die Anzahl der Zahlen, die mit den für C vorhandenen Bits (7) darstellbar sind, also

$$C_1 = 128_D = 2^7 .$$

$E = 2$ und $t = 6$.

Integer*4 Zahlen müssen zwischen

$$-2^{+31} = -2147483648 \quad \text{und} \quad 2^{31} - 1 = 2147483647$$

liegen. Das Fortran Handbuch erlaubt negative Zahlen nur bis $-2^{31} + 1$, aber der Compiler akzeptiert -2^{31}, wie das folgende Programm zeigt:

```
C     PROGRAMM NUMER4
10    FORMAT (I20)
      I = 2**30
      J = -I
```

```
J = J - I
PRINT 10,J
J = J/2
PRINT 10,J
```

mit Ausgabe:

```
-2147483648
-1073741824.
```

4.7.2 IBM 4381 PL1

Es folgt aus dem PL1 Handbuch, GC26-3977-1, 1984:

1. Fixed Decimal Zahlen werden mit Basiszahl $d = 10$ gespeichert, mit zwei Dezimalziffern pro Byte. Das Vorzeichen wird in einem Byte mitgespeichert (Handbuch Seite 13).
2. Float Decimal Zahlen werden als hexadezimale Zahlen gespeichert. Positive Zahlen müssen zwischen

   $$16^{-65} \approx 5,3 \times 10^{-79} \quad \text{und} \quad 16^{+63} \approx 7,23 \times 10^{+75}$$

   liegen.

4.7.3 FX-720P Basic

Alle Zahlen werden als Gleitkommazahlen behandelt. Die Basiszahl ist $d = 10$, und es kann nur bis zu 10 Ziffern gedruckt werden. Intern werden aber 12 Stellen (lt. Hersteller) und 11 Stellen (wie von uns festgestellt) benutzt. Es ist typisch für Taschenrechner, daß intern mehr Stellen zur Verfügung stehen als die Anzahl Stellen, die ausgegeben werden kann, und diese Tatsache muß immer berücksichtigt werden:

```
 5    REM NUMER5
10    x = 1. E 10
12    z = x + 1
14    A = z - x
16    PRINT "x ="; x
18    PRINT "z ="; z
20    PRINT "A ="; A
22    STOP
```

mit Ausgabe:

```
x = 1. E 10
z = 1. E 10
A = 1
```

Positive Zahlen müssen zwischen

$$1 \times 10^{-99} \quad \text{und} \quad 9.999999999 \times 10^{99}$$

liegen, wobei es bemerkenswert ist, daß diese Grenzen innerhalb des Rechners streng eingehalten werden, obwohl intern mindestens 11 Stellen zur Verfügung stehen.

Es folgt auch, daß der FX-72OP die „Scientific Notation" benutzt mit $E = 2$.

4.7.4 Rainbow 100 Pascal

Ganze Zahlen werden als binäre Zahlen mit $t = 15$ gespeichert. Negative Zahlen werden als „Zweier"-Komplement gespeichert. Die ganzen Maschinenzahlen i liegen deshalb in dem Bereich:

$$-2^{15} = -32768 \leq i \leq 32767 = 2^{15} - 1 \,.$$

Reelle Zahlen werden als Gleitkommazahlen mit

$$d = 2\,,\ t = 40\,,\ E = 8$$

gespeichert. Die größtmögliche Zahl ist deshalb

$$(1 - 2^{-40}) \times 2^{127} \,.$$

Das Vorzeichen wird als 1 Bit in der führenden Stelle der Mantisse gespeichert. $C_1 = 2^8$. Ist der Exponent $= -C_1/2$, wird die Zahl als Null betrachtet.

Literatur

Cody jr., W.J. und **Waite**, W.: Software Manual for the Elementary Functions, Appendix B. Prentice Hall, Englewood Cliffs, N.J., 1980.

IBM System/370. Principles of Operation. GA22-7000-9, 1983.

Kahan, W.: Implementation of Algorithms. Technical Report. University of California, Berkeley, 1973. National Technical Information Service AD-769 124.

Kulisch, U.W. und **Miranker**, W.L.: Computer Arithmetic in Theory and Practice. New York, Academic Press, 1981.

Malcolm, M.A.: Algorithms to Reveal Properties of Floating-Point Arithmetic. Communications ACM 15, 949-951 (1972).

Spaniol, O.: Arithmetik in Rechenanlagen, Teubner 1976.

Sterbenz, P.H.: Floating-point Computation. Prentice Hall, Englewood Cliffs, N.J., 1974.

Kapitel 5

Fehleranalyse und Konditionszahlen

5.1 Einführung

Unsere Aufgabe ist, Probleme mit vorgeschriebenen Eingabedaten d zu lösen. Es gibt mehrere Gründe, weshalb diese Aufgabe im allgemeinen nicht exakt durchzuführen ist:

1. Fehler in den gegebenen Eingabedaten, z.B. Meßfehler.

2. Rundungsfehler während der Berechnungen.

3. Abbruchfehler oder Diskretisierungsfehler.
   Wird z.B. die Funktion e^x mit Hilfe ihrer Taylorreihe

   $$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

   berechnet, ist es nur möglich, eine endliche Summe

   $$S_N := \sum_{k=0}^{N} \frac{x^k}{k!}$$

   zu berechnen. Es entsteht ein Abbruchfehler

   $$e^x - S_N = \sum_{k=N+1}^{\infty} \frac{x^k}{k!}.$$

   Bei der numerischen Berechnung von Integralen und der numerischen Lösung von Differentialgleichungen spricht man oft von Diskretisierungsfehlern anstatt von Abbruchfehlern.

4. Vereinfachungen im mathematischen Modell.

5. Menschliche und maschinelle Fehler.

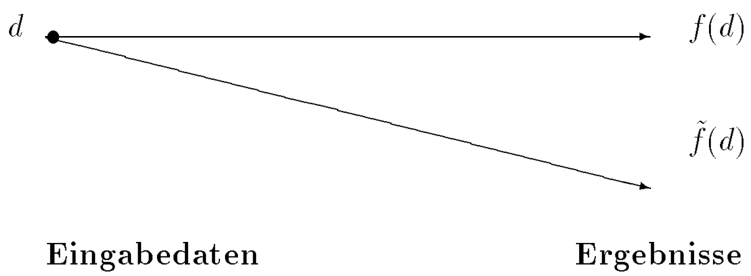


Abbildung 5.1: Eine allgemeine Darstellung der Fehleranalyse

Obwohl alle oben zitierten Fehlerquellen wichtig sind, sind die Rundungsfehler und Abbruchfehler von besonderer Bedeutung für Numeriker, weil diese Fehler im eigenen Bereich liegen.

Anstatt der exakten Lösung $x = f(d)$ des Problems P mit Eingabedaten d wird deshalb im allgemeinen eine Approximation $\tilde{x} = \tilde{f}(d)$ berechnet. Hierbei sind d und x nicht als einfache Zahlen zu betrachten, aber als Vektoren oder Elemente von Banachräumen. Ebenso sind f und $\tilde{f}$ allgemeine Abbildungen.

Diese allgemeine Darstellung läßt sich gut graphisch darstellen. (Abbildung 5.1). **Beispiel:** $d = (d_o, \ldots, d_n) \in \mathbb{R}^{n+1}$ sei die Menge der Koeffizienten eines Polynoms p_d, und $f(d) \in \mathbf{C}^n$ sei die Menge der Nullstellen von p_d

Wir gehen jetzt davon aus, daß das Problem P mit Daten d die exakte Lösung $x = f(d)$ hat. Ein Algorithmus A zur Lösung von Problem P erzeugt das Ergebnis $\tilde{x} = \tilde{f}(d)$. Es entstehen mehrere Fragen, die wir zuerst sehr allgemein und ungenau formulieren und dann durch konkrete Beispiele erläutern.

Definition 5.1 *Ein Problem P heißt gut (schlecht) konditioniert, wenn „kleine" Änderungen in den Daten „kleinen" („großen") Änderungen in der Lösung entsprechen. Schlecht konditionierte Probleme werden manchmal schlechtgestellte Probleme genannt. (Siehe Abbildung 5.2.)*

Definition 5.2 *Ein Algorithmus A für ein Problem P heißt gutartig (bösartig), wenn der Algorithmusfehler $x - \tilde{x}$ nicht wesentlich größer (wesentlich größer) ist als die Lösungsänderungen, die bei kleinen Datenänderungen entstehen. In einigen Teilbereichen der praktischen Mathematik werden gutartige bzw. bösartige Algorithmen als numerisch stabil bzw. instabil bezeichnet.*

1. **Erster Fragenkomplex (Vorwärts-) Fehleranalyse:**

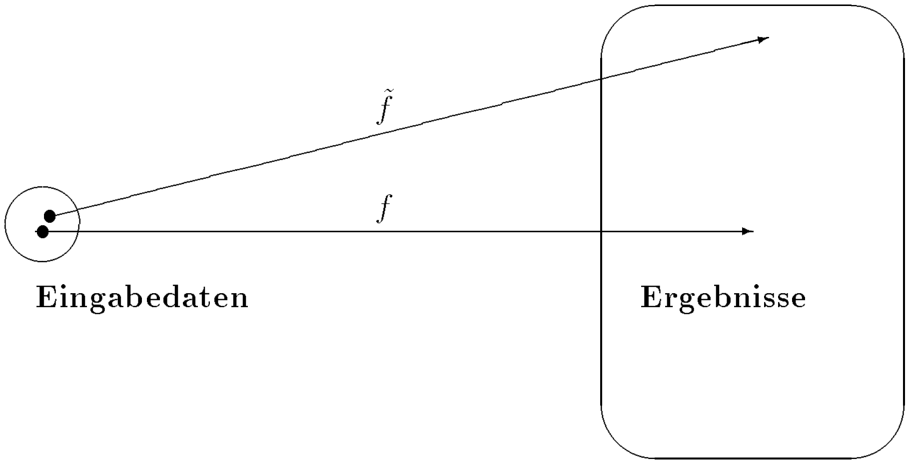


Abbildung 5.2: Ein schlecht konditioniertes Problem

(a) Wie groß ist der Fehler $x - \tilde{x}$?

(b) Gibt es Abschätzungen oder Approximationen für den Fehler $x - \tilde{x}$?

2. **Zweiter Fragenkomplex (Rückwärts-) Fehleranalyse:**
Gibt es Daten $\tilde{d}$ derart, daß $\tilde{x}$ die Lösung von Problem P mit Daten $\tilde{d}$ ist, d.h. $\tilde{x} = f(\tilde{d})$? (Siehe Abbildung 5.3.)

3. **Dritter Fragenkomplex: Konditionsfragen**

(a) Wenn die Daten d geringfügig geändert werden, wie groß ist die Änderung in der Lösung? Ist das Problem gut konditioniert oder schlecht konditioniert?

(b) Ist der Algorithmus gutartig oder bösartig?

(c) Falls das Problem schlecht konditioniert oder der Algorithmus bösartig ist, gibt es bessere Alternativen?

5.2 Einige Beispiele

5.2.1 Das Wilkinsonsche Polynom

(Wilkinson war ein Pionier in der Numerik, der viele Probleme der Rundungsfehler und der Fehleranalyse erläutert.)

Das Wilkinsonsche Polynom

$$p(z) = \sum_{k=0}^{20} d_k z^k := \prod_{k=1}^{20} (z-k)$$

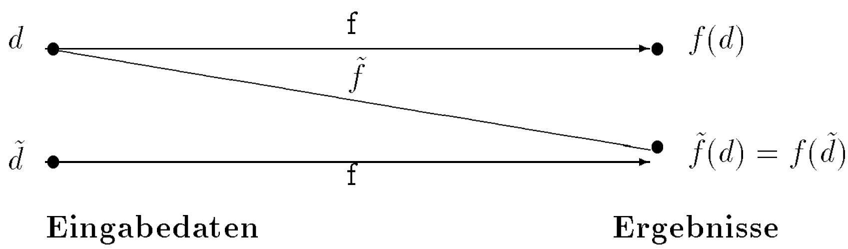


Abbildung 5.3: Rückwärtsfehleranalyse

hat die Nullstellen $x_k = k\,,\ 1 \leq k \leq 20$. Die Koeffizienten d_k können berechnet werden, z.B.:

$$d_{20} = 1\,,\ d_{19} = -\sum_{k=1}^{20} k = -\frac{20 \cdot 21}{2} = -210\,.$$

Die Nullstellen von

$$p(z) - 2^{-23} \cdot z^{19}$$

bis auf 5 Dezimale sind:

1,00000	10,o9526	± 0,64350i
2,00000	11,79363	± 1,65232i
3,00000	13,99235	± 2,51883i
4,00000	16,73073	± 2,81262i
5,00000	19,50243	± 1,94033i
6,00000		
6,99969		
8,00726		
8,91725		
20,84690		

Das heißt, eine relative Änderung von

$$2^{-23}/210 \doteq 5,67 \cdot 10^{-10}$$

in der Datenkoeffizienten d_{19} entspricht einer relativen Änderung von

$$\cdot 84690/20 \doteq \cdot 0423$$

in x_{20}. Die reellen Nullstellen x_{10} bis x_{19} verändern sich sogar in komplexe Nullstellen.

Die Wurzeln sind unterschiedlich empfindlich gegenüber Änderungen in d_{19}. Insbesondere sind die kleinen Wurzeln unempfindlich.

Ist $\epsilon = eps$ die Maschinengenauigkeit, so gilt für Gleitkommaarithmetik mit einfacher Genauigkeit $\epsilon > 5 \cdot 10^{-10}$. Dies bedeutet, daß *ein einziger Rundungsfehler* die oben angegebenen Änderungen auslöschen könnte.

Das Problem ist sehr schlecht konditioniert.

Bemerkung:

1. Wird für das Wilkinsonsche Polynom nur die Summe der Wurzel gewünscht, läßt sich dies sehr bequem und mit hoher Genauigkeit bestimmen:

$$\sum_{k=1}^{20} x_k = -d_{19}/d_{20}\,.$$

2. Die Eigenwerte λ einer Matrix A sind Nullstellen des charakteristischen Polynoms

$$p(\lambda) = \det(A - \lambda I)\,.$$

Die Bestimmung der Eigenwerte einer symmetrischen Matrix ist aber im allgemeinen ein gut konditioniertes Problem, z.B. die Eigenwerte der Diagonalmatrix

$$A = \begin{pmatrix} 1 & & & \bigcirc \\ & 2 & & \\ & & \ddots & \\ \bigcirc & & & 20 \end{pmatrix}$$

sind sofort zu berechnen, obwohl das zugehörige charakteristische Polynom das Wilkinsonsche Polynom ist.

Ist ein Problem schlecht konditioniert, so empfiehlt es sich zu fragen, ob die Lösung tatsächlich in dieser Gestalt gewünscht wird. Es kommt nämlich oft vor, daß das Problem anders gestellt werden kann.

5.2.2 Eine Rekursionsformel für Integrale

Die Integrale

$$y_n := \int_o^1 \frac{x^n}{5+x}\,dx$$

erfüllen die Rekursionsformel

$$y_n + 5y_{n-1} = \int_o^1 \frac{x^n + 5x^{n-1}}{x+5}\, dx = \frac{1}{n}\,,\ n \geq 1 \tag{5.1}$$

mit Anfangswert

$$y_o = \int_o^1 \frac{1}{5+x}\, dx = [\ell n(5+x)]_o^1 = \ell n(6/5)\,.$$

Auf dem FX-720P ergibt das Programm 'Instabilität'

```
10  REM INSTABILITAET
15  Y = LN(6/5)
20  FOR N=1 TO 20
30  Y = 1/N - 5*Y
40  PRINT "N="; N; "Y="
45  SET E5:PRINT Y : SET N
50  NEXT N
60  STOP
```

die Ergebnisse in Tabelle 5.1. Die Fehler sind zunehmend und oszillieren.

N	Y	N	Y
1	$8,8392 \cdot 10^{-2}$	11	$1,3776 \cdot 10^{-2}$
2	$5,8093 \cdot 10^{-2}$	12	$1,4453 \cdot 10^{-2}$
3	$4,3139 \cdot 10^{-2}$	13	$4,6571 \cdot 10^{-3}$
4	$3,4306 \cdot 10^{-2}$	14	$4,8143 \cdot 10^{-2}$
5	$2,8468 \cdot 10^{-2}$	15	$-1,7405 \cdot 10^{-1}$
6	$2,4325 \cdot 10^{-2}$	16	$9,3275 \cdot 10^{-1}$
7	$2,1232 \cdot 10^{-2}$	17	$-4,6049 \cdot 10^{-0}$
8	$1,8839 \cdot 10^{-2}$	18	$2,3080 \cdot 10^{1}$
9	$1,6915 \cdot 10^{-2}$	19	$-1,1535 \cdot 10^{2}$
10	$1,5427 \cdot 10^{-2}$	20	$+5,7679 \cdot 10^{2}$

Tabelle 5.1: Berechnete Lösung für y_n

Dieses Phänomen läßt sich dadurch erklären, daß die Lösung y_n der Differenzengleichung (5.1) gleich die Summe einer speziellen Lösung s_n und einer Vielfachheit c der Lösung $h_n := -5^n$ der homogenen Gleichung ist:

$$y_n = s_n + c\, h_n\,.$$

Selbst wenn c exakt Null ist, werden Rundungsfehler dafür sorgen, daß die berechneten Werte $\tilde{y}_n$ eine Komponente $c\,h_n$ haben. Da h_n sehr rasch wächst, dominiert die Komponente $c\,h_n$ für $n \to \infty$.

Diese Erklärung wird durch die numerischen Ergebnisse untermauert, da - wie aus der Tabelle folgt -

$$-\frac{Y_{20}}{Y_{19}} \doteq \frac{-5,7679\,10^2}{-1,1535\,10^2} = 5,00035\,.$$

5.2.3 Lösung eines tridiagonalen Gleichungssystems

Sei A eine $n \times n$ tridiagonale Matrix mit hauptdiagonalen Koeffizienten $a_{ii} = 5$ und nebendiagonalen Koeffizienten $a_{i,i-1} = a_{i,i+1} = -1$. Z.B. für $n = 5$ ist

$$A = \begin{pmatrix} 5 & -1 & 0 & 0 & 0 \\ -1 & 5 & -1 & 0 & 0 \\ 0 & -1 & 5 & -1 & 0 \\ 0 & 0 & -1 & 5 & -1 \\ 0 & 0 & 0 & -1 & 5 \end{pmatrix}$$

Gesucht ist ein Algorithmus zur Lösung des Gleichungssystems

$$Ax = b$$

mit beliebigen b. Zur Probe setzen wir $b = As$, wobei $s = (s_i)$ mit $s_i := i^2$. Z.B. für $n = 5$ ist $s = (1,4,9,16,25)^T$.

Wir betrachten zwei Algorithmen, die zusammen in einem Programm illconditioned (siehe Abbildung) aufgeführt werden:

```
      program illconditioned
c     zeigt zwei verschiedene Methoden zur Loesung von
c     tridiagonalen Gleichungssystemen
       parameter ( nmax=100,n=40)
       real d(n),b(n),e(n)
       real al(n),ad(n),ar(n),r(n),s(n)
       real ls(n),l1(n),l2(n)
       real f1,f2,f3,f4,c1,c2
       real soltri(n),soldif(n)
       real rs,r1,r2
       real sq,z1,z2

       do 100 i=1,n
         al(i)=-1
         ar(i)=-1
```

```
          ad(i)=+5
          s(i)=i*i
100     continue
          al(1)=0
          ar(n)=0
        do 110 i=1,n
110       r(i)=al(i)*s(i-1)+ad(i)*s(i)+ar(i)*s(i+1)

c       Differenzengleichungen
c       Spezielle Loesung
        ls(1)=0
        ls(2)=0
        l1(1)=1
        l1(2)=0
        l2(1)=0
        l2(2)=1

        do 200 i=3,n
          ls(i)=(r(i-1)-al(i-1)*ls(i-2)-ad(i-1)*ls(i-1))/ar(i-1)
          l1(i)=(-al(i-1)*l1(i-2)-ad(i-1)*l1(i-1))/ar(i-1)
          l2(i)=(-al(i-1)*l2(i-2)-ad(i-1)*l2(i-1))/ar(i-1)
200     continue

c       ad(1)*c1+ar(1)*c2=r(1)
c       al(n)*(ls(n-1)+c1*l1(n-1)+c2*l2(n-1))+
c       ad(n)*(ls(n)+c1*l1(n)+c2*l2(n))=r(n)
c       c1=f1*c2+f2
        f2=r(1)/ad(1)
        f1=-ar(1)/ad(1)
c       f3*c2=f4
        f4=r(n)-al(n)*(ls(n-1)+f2*l1(n-1))-ad(n)*(ls(n)+f2*l1(n))
        f3=al(n)*(f1*l1(n-1)+l2(n-1))+ad(n)*(f1*l1(n)+l2(n))
        c2=f4/f3
        c1=f1*c2+f2



c       Algorithmus von T
c       Golub und van Loan Seite 98
        do 450 k=1,n
           d(k)=ad(k)
           e(k)=ar(k)
           b(k)=r(k)
450     continue
        do 500 k=2,n
```

```
          t=e(k-1)
          e(k-1)=t/d(k-1)
          d(k)=d(k)-t*e(k-1)
500     continue
        do 550 k=2,n
550       b(k)=b(k)-e(k-1)*b(k-1)
        do 560 k=1,n
560       b(k)=b(k)/d(k)
        do 570 k=n-1,1,-1
570       b(k)=b(k)-e(k)*b(k+1)


        do 1000 i=1,n
        soltri(i)=b(i)
        soldif(i)=ls(i)+c1*l1(i)+c2*l2(i)
        print *,i,s(i),soltri(i),soldif(i)
1000    continue

        rs=ls(n)/ls(n-1)
        r1=l1(n)/l1(n-1)
        r2=l2(n)/l2(n-1)
        print *, rs,r1,r2

        sq=sqrt(ad(2)**2 -4*al(2)*ar(2))
        z1=(5+sq)/2
        z2=(5-sq)/2
        print *,z1,z2

        end
```

Algorithmus 1
Es gilt

$$-x_{i-1} + 5x_i - x_{i+1} = b_i \,,\; 2 \leq i \leq n-1 \,,$$

d.h., die Koeffizienten x_i erfüllen eine inhomogene Differenzengleichung zweiter Ordnung. Aus der Theorie von Differenzengleichungen folgt:

$$x_i = u_i + a_1 v_i^{(1)} + a_2 v_i^{(2)}$$

mit

$$
\begin{aligned}
-u_{i-1} &+ 5u_i - u_{i+1} = b_i\,,\ 2 \le i \le n-1 \\
u_1 &= 0\,,\ u_2 = 0 \\
-v^{(1)}_{i-1} &+ 5v^{(1)}_i - v^{(1)}_{i+1} = 0\,,\ 2 \le i \le n-1 \\
v^{(1)}_1 &= 1\,,\ v^{(1)}_2 = 0 \\
-v^{(2)}_{i-1} &+ 5v^{(2)}_i - v^{(2)}_{i+1} = 0\,,\ 2 \le i \le n-1 \\
v^{(2)}_1 &= 0\,,\ v^{(2)}_2 = 1
\end{aligned}
$$

u ist eine spezielle Lösung der inhomogenen Gleichung, und $v^{(1)}$ und $v^{(2)}$ sind linear unabhängige Lösungen der homogenen Gleichung.

u_i, $v^{(1)}_i$ und $v^{(2)}_i$ sind für $1 \le i \le n$ damit eindeutig bestimmt und leicht berechenbar.

Zur Bestimmung der Konstanten a_1 und a_2 wird verlangt, daß

$$
\begin{aligned}
-x_1 &+ 5x_2 = b_1 \\
-x_{n-1} &+ 5x_n = b_n
\end{aligned}
$$

Algorithmus 2

Dieser Algorithmus benutzt das Gaußsche Eliminationsverfahren, wobei die Tatsache, daß A eine trigiagonale Matrix ist, wesentliche Vereinfachungen erlaubt.

Sowohl Algorithmus 1 als auch Algorithmus 2 ist effizient, da nur $\bigcirc(n)$ Operationen erforderlich sind.

Die numerischen Ergebnisse (siehe Abbildung) zeigen, daß Algorithmus 1 bösartig ist, obwohl Algorithmus 2 gutartig ist.

1	1.000000	1.000000	1.000000
2	4.000000	4.000000	3.999999
3	9.000000	9.000001	8.999994
4	16.00000	16.00000	15.99997
5	25.00000	25.00000	24.99988
6	36.00000	36.00000	35.99927
7	49.00000	49.00000	48.99707
8	64.00000	64.00000	63.98438
9	81.00000	80.99999	80.92188
10	100.00000	100.00000	99.62500
11	121.0000	121.0000	119.5000
12	144.0000	144.0000	136.0000
13	169.0000	169.0000	128.0000
14	196.0000	196.0000	0.
15	225.0000	225.0000	-768.0000
16	256.0000	256.0000	-4096.000

```
  17    289.0000    289.0000  -20480.00
  18    324.0000    324.0000  -65536.00
  19    361.0000    361.0000  -262144.0
  20    400.0000    400.0000  -524288.0
  21    441.0000    441.0000   -4.19430e+06
  22    484.0000    484.0000   -3.35544e+07
  23    529.0000    529.0000   -2.01327e+08
  24    576.0000    576.0000   -5.36871e+08
  25    625.0000    625.0000   -2.14748e+09
  26    676.0000    676.0000   -8.58993e+09
  27    729.0000    729.0000   -6.87195e+10
  28    784.0000    784.0000   -2.74878e+11
  29    841.0000    841.0000   -2.19902e+12
  30    900.0000    900.0000   -4.39805e+12
  31    961.0000    961.0000   -1.75922e+13
  32    1024.000    1024.000   -7.03687e+13
  33    1089.000    1089.000  0.
  34    1156.000    1156.000   -2.25180e+15
  35    1225.000    1225.000   -1.80144e+16
  36    1296.000    1296.000   -7.20576e+16
  37    1369.000    1369.000   -5.76461e+17
  38    1444.000    1444.000   -1.15292e+18
  39    1521.000    1521.000   -4.61169e+18
  40    1600.000    1600.000   -1.84467e+19
    4.791288    4.791288    4.791288
    4.791288    0.208712
```

Dieses Ergebnis hat Konsequenzen bei der numerischen Lösung von elliptischen Differentialgleichungen. Das vorgestellte Beispiel ist eine Vereinfachung eines numerischen Problems, das in diesem Bereich auftritt, und Algorithmus 1 und 2 sind Vereinfachungen entsprechender Algorithmen A1 und A2.

Algorithmus A1 wäre eine sehr effiziente Methode zur Lösung elliptischer Differentialgleichungen, wenn er nicht bösartig wäre.

5.3 Die Modellierung von Rundungsfehlern

Obwohl es grundsätzlich möglich ist, jeden Rundungsfehler zu berechnen, ist dies glücklicherweise nur selten nötig. Was nötig ist, ist ein Modell, womit realistische Abschätzungen von Rundungsfehlern ohne einen zu großen Aufwand erreicht werden können. Das Modell, das in diesem Abschnitt vorgestellt wird, wird seit langem benutzt. Nach der Einführung der IEEE Standard 754 Arithmetik ist dieses Modell durchaus realistisch (was vorher nicht immer der Fall war).

5.3.1 Voraussetzungen

Es werde Gleitkommaarithmetik mit Basis d und Mantissenlänge t benutzt. A sei die Menge der Maschinenzahlen

$$\tilde{A} := [\omega_+ \,,\, \Omega_+] \cup \{0\} \cup [\Omega_- \,,\, \omega_-] \,.$$

Satz 5.1 *Sei eps* $:= \frac{1}{2} d^{-t+1}$, *eps* < 1, *rd der Rundungsoperator*, $x \in \tilde{A}\backslash\{0\}$. *Dann gilt:*

1. $\left|\frac{rd(x)-x}{x}\right| \leq$ *eps,*
2. $\left|\frac{rd(x)-x}{rd(x)}\right| \leq$ *eps,*
3. *Es gibt ein* $\rho = \rho(x) \in \mathbb{R}$ *mit* $rd(x) = \rho x$ *und* $|\ell n\, \rho| \leq$ *eps.*

Beweis:

1. $x = \pm\, 0, \quad a_1 \ldots \cdot d^e, \quad a_1 \neq 0$. Es folgt
$$|\mathrm{rd}(x) - x| = \min_{\alpha \in A} |\alpha - x| \leq \frac{d}{2}\, d^{-t-1} d^e$$
und
$$|x| \geq d^{-1} d^e \quad (\alpha \in A\,, \text{da} \quad x \quad \text{normalisiert ist})\,.$$
Es folgt sofort, daß
$$\left|\frac{\mathrm{rd}(x) - x}{x}\right| \leq \frac{1}{2}\, d^{-t} d^e / d^{-1} d^e = \frac{1}{2}\, d^{-t+1}\,.$$

2. Der Beweis ist wie bei 1.

3. Nach 1. und 2. existieren Zahlen ϵ_i, $i = 1, 2$, mit $|\epsilon_i| \leq$ eps, so daß gilt:
$$\frac{\mathrm{rd}(x) - x}{x} = \epsilon_1 \quad , \quad \frac{\mathrm{rd}(x) - x}{\mathrm{rd}(x)} = \epsilon_2\,,$$
d.h.
$$\mathrm{rd}(x) = (1 + \epsilon_1)x \quad , \quad \mathrm{rd}(x) = \frac{x}{1 - \epsilon_2}\,.$$
Es gilt: $1 + \epsilon \leq e^\epsilon \;\; \forall\, \epsilon \in \mathbb{R}$, so daß
$$\begin{aligned} (1 + \epsilon_1) &\leq e^{\epsilon_1} \leq e^{\mathrm{eps}} \\ (1 - \epsilon_2)^{-1} &\geq e^{+\epsilon_2} \geq e^{-\mathrm{eps}}\,. \end{aligned}$$
Durch Anwendung des Zwischenwertsatzes sieht man, daß $\rho \in \mathbb{R}$ existiert mit $\mathrm{rd}(x) = \rho x \quad$ und $\quad e^{-eps} \leq \rho \leq e^{eps}$.

□

Satz 5.2 *Sei* $eps := d^{-t+1}$, $eps < 1$, $x \in \tilde{A}\backslash\{0\}$. *Dann gilt:*

1. $\left|\frac{\mathrm{schn}(x)-x}{x}\right| \leq eps$

2. $\left|\frac{\mathrm{schn}(x)-x}{\mathrm{schn}(x)}\right| \leq eps$

3. *Es gibt ein* $\rho = \rho(x) \in \mathbb{R}$ *mit* $\mathrm{schn}(x) = \rho x$ *und* $|\ell n\, \rho| \leq eps$.

Beweis: Wie bei Satz 5.1. □

5.3.2 Gleitkommaoperationen

Seien $\Box \in \{+,-,.,/\}$, $x \in A$, $y \in A$, $x\Box y \in \tilde{A}\backslash\{0\}$. Dann ist

$$x\Box^* y := \mathtt{rd}(x\Box y)\,.$$

$\Box^*$ heißt die Gleitpunktoperation, die der arithmetischen Operation $\Box$ entspricht.

Bemerkung: Andere oft benutzte Schreibweisen sind:

$\odot$ statt $.*$,
$\oplus$ statt $+*$, usw.,
$f\ell(x\Box y) = g\ell(x\Box y) = x\Box^* y$.

Definition 5.3 *Sei* $x \in \mathbb{R}$. *Sei* $\tilde{x} \in \mathbb{R}$ *eine Approximation zu* x. *Dann heißt*

$\lvert x-\tilde{x}\rvert$	*der absolute Fehler von* $\tilde{x}$,
$\frac{\lvert x-\tilde{x}\rvert}{\lvert x\rvert}$	*der relative Fehler von* $\tilde{x}$, $x \neq 0$.

Satz 5.3 *Der relative Fehler der Gleitkommaoperationen ist klein, genauer gilt:*
Sei $x, y \in A$, $x\Box y \in \tilde{A}\backslash\{0\}$. *Dann existiert* $\rho \in \mathbb{R}$ *mit*

$$x\Box^* y = \rho(x\Box y) \quad \textit{und} \quad |\ell n\, \rho| \leq eps\,.$$

Beweis: Siehe Sätze 5.5 und 5.6. □

Bemerkung: Obwohl der relative Fehler bei einem Schritt klein ist, sind große relative Fehler bei der Nacheinanderausführung von mehreren Gleitkommaoperationen möglich.

Beispiel: $d = 10$, $t = 2$.

1. Arithmetische Operationen:

$$x := 0,75 + 0,055 - 0,80 = 0,005\,.$$

2. Gleitkommaoperationen:

$$\tilde{x} := \underbrace{(0,75 +^* 0,055)}_{\underbrace{\mathrm{rd}(0,805)}_{0{,}80}} -^* 0,80 = 0\,.$$

Der relative Fehler $|x - \tilde{x}|/|x|$ ist 1!

3. In anderer Reihenfolge der Gleitkommaoperationen erhält man

$$\overline{x} := 0,75 +^* \underbrace{(0,055 -^* 0,80)}_{\underbrace{\mathrm{rd}(-0,745)}_{-0{,}74}} = 0,01\,.$$

Der relative Fehler beträgt wieder 1.

In Absatz 2) haben wir eine gefährliche Situation kennengelernt, nämlich die, wo der Fehler mit einem großen Faktor in jedem Schritt multipliziert wird (*Instabilität* des Verfahrens). Das letzte Beispiel zeigt eine zweite gefährliche Situation, nämlich die, wo zwei fast gleiche Zahlen voneinander subtrahiert werden (*Auslöschung*).

Voraussetzung für Satz 3 ist, daß $x \Box y \in \tilde{A} \backslash \{0\}$. Diese Voraussetzung wird verletzt, wenn:

1. $x \Box y > \Omega_+$ oder $x \Box y < \Omega_-$, was *Exponentenüberlauf* heißt. Exponentenüberlauf wird auf allen Rechnern festgestellt. In einigen Fällen (z.B. Fortran auf dem IBM 4381) wird das Ergebnis gleich Ω^+ oder Ω^- gesetzt, d.h. Rundung oder Schneiden werden richtig durchgeführt. In anderen Fällen werden die Berechnungen unterbrochen.

2. $\omega_- < x \Box y < 0$ oder $0 < x \Box y < \omega_+$, was *Exponentenunterlauf* heißt.
   In vielen Fällen wird das Ergebnis ohne Warnung gleich 0 gesetzt.

5.4 Fehlerfortpflanzung

Wir betrachten jetzt die Fehlerfortpflanzung von relativen Fehlern. Dies ist oft besonders sinnvoll bei Experimenten, wo unterschiedliche Meßungen unterschiedlichen Fehlern unterworfen sind.

Satz 5.4 *Seien*

$$f : D \subset \mathbb{R}^p \longrightarrow \mathbb{R}^q \ , \ \ f \in C^1(D) \ , \ \ D \ \textit{konvex} \ , \ \ x = f(d) \, ,$$
$$x + \triangle x = f(d + \triangle d) \ , \ \ d \in D \ , \ \ d + \triangle d \in D \, .$$

Dann gilt:

$$|\triangle x_i| \leq \sum_{j=1}^{p} \sup_{u \in D} \left| \frac{\partial f_i}{\partial d_j}(u) \right| \cdot |\triangle d_j| \, .$$

Beweis: $\triangle x_i = f_i(d + \triangle d) - f_i(d)$. Wir betrachten nun die Hilfsfunktion g.

$$g(t) := f_i(d + t\triangle d) - f_i(d) \ , \ \ 0 \leq t \leq 1 \, ,$$

mit

$$g(0) = 0 \ , \ \ g(1) = \triangle x_i \, ,$$

wobei $d + t\triangle d \in D$ für $t \in [0,1]$, weil D konvex ist. Nach dem Mittelwertsatz (Forster I, S. 110):

$$\begin{aligned} |\triangle x_i| = |g(1) - g(0)| &\leq 1 \cdot \sup_{t \in [0,1]} |g'(t)| \, , \\ &= \sup_{t \in [0,1]} \left| \sum_{j=1}^{p} \frac{\partial f_i(d + t\triangle d)}{\partial d_j} \cdot \triangle d_j \right| \\ &\leq \sum_{j=1}^{p} \sup_{u \in D} \left| \frac{\partial f_i}{\partial d_j}(u) \right| \cdot |\triangle d_j| \end{aligned}$$

□

Definition 5.4 *Sei $f_i(d) \neq 0$ für $\forall_i$. Die Zahlen*

$$k_{ij}(d) := \frac{\partial f_i(d)}{\partial d_j} \cdot \frac{d_j}{f_i(d)} \ , \ \ 1 \leq i \leq q \ , \ \ 1 \leq j \leq p$$

heißen Verstärkungsfaktoren.

$$k(d) := \sum_{i=1}^{q} \sum_{j=1}^{p} |k_{ij}(d)|$$

heißt die Verstärkungszahl.

Satz 5.5 *$f, x, d, d + \triangle d$ und D erfüllen die Voraussetzungen von Satz 5.4. Sei $d_j \neq 0 \ \forall_j, f_i(d) \neq \forall_i$. Sei $f \in C^2(\overline{D})$. Dann gilt:*

$$\begin{aligned} \frac{\triangle x_i}{x_i} &= \sum_{j=1}^{p} k_{ij}(d) \cdot \frac{\triangle d_j}{d_j} + e_i \, , \\ e_i &= 0(\|\triangle d\|_\infty^2) = 0([\max_j |\triangle d_j|]^2) \, . \end{aligned}$$

Beweis: Sei g wie im Beweis von Satz 5.4 definiert. Es gilt:

$$\frac{\triangle x_i}{x_i} = (g(1) - g(0))/x_i = (g'(0) + \frac{1}{2}g''(\tau))/x_i\,,\ \tau \in [0,1]$$

wobei

$$\begin{aligned}
\frac{g'(0)}{x_i} &= \sum_{j=1}^{p} \frac{\partial f_i(d)}{\partial d_j}\frac{\triangle d_j}{x_i} = \sum_{j=1}^{p} k_{ij}(d)\cdot\frac{\triangle d_j}{d_j}\\
\frac{1}{2}\frac{g''(\tau)}{x_i} &= \frac{1}{2x_i}\sum_{m=1}^{p}\sum_{j=1}^{p}\frac{\partial^2 f_i}{\partial d_j \partial d_m}(d+\tau\triangle d)\cdot\triangle d_j\cdot\triangle d_m\,.
\end{aligned}$$

Es folgt, daß

$$\frac{\triangle x_i}{x_i} = \sum_{j=1}^{p} k_{ij}(d)\,\frac{\triangle d_j}{d_j} + e_i$$

mit

$$\begin{aligned}
|e_i| &\le \frac{1}{2|x_i|}\sum_{m=1}^{p}\sum_{j=1}^{p}\sup_{u\in D}\left|\frac{\partial^2 f_i}{\partial d_j\partial d_m}\right|\cdot|\triangle d_j|\cdot|\triangle d_m|\\
&\le \frac{p^2}{2|x_i|}\cdot\max_{u\in\overline{D}}\left|\frac{\partial^2 f_i}{\partial d_j\partial d_m}\right|\cdot(\|\triangle d\|_\infty)^2\,.
\end{aligned}$$

□

Die Zahl k_{ij} heißt *Verstärkungsfaktor*, weil - wie aus Satz 5.5 zu ersehen ist - der relative Fehler

$$\left|\frac{\triangle d_j}{d_j}\right|$$

mit dem Faktor k_{ij} „verstärkt" wird.

Satz 5.6 (Forster 2, S. 53) *Sei $U \subset \mathbb{R}^p$ offen und $f : U \longrightarrow \mathbb{R}^q$ eine stetig differenzierbare Abbildung. Sei U konvex, Df die Funktionalmatrix oder Jacobi-Matrix von f,*

$$Df := \left(\frac{\partial f_i}{\partial d_j}\right)_{,\ \substack{1\le i\le q\\ 1\le j\le p}}\ .$$

Sei $x, x+\xi \in U$ und

$$M := \sup_{0\le t\le 1}\ \|Df(x+t\xi)\|\ .$$

Dann gilt:

$$\|f(x+\xi) - f(x)\| \le M\|\xi\|\ .$$

5.4.1 Beispiel 1: Addition / Subtraktion

Wir berechnen nun die Verstärkungszahl für die arithmetischen Operationen:

$$x = x_1 = f_1(d) := d_1 + d_2 \;, \quad (p = 2, q = 1)\,.$$

Dann gilt:

$$\begin{aligned} k_{11} &= \frac{\partial f_1}{\partial d_1} \cdot \frac{d_1}{x_1} = \frac{d_1}{d_1 + d_2}\,, \\ k_{12} &= \frac{d_2}{d_1 + d_2}\,, \\ k &= \frac{|d_1| + |d_2|}{|d_1 + d_2|}\,. \end{aligned}$$

Das Problem ist schlecht konditioniert, falls $d_1/d_2 \doteq -1$.

Wichtig:
Subtraktion nahezu gleich großer Zahlen ist schlecht konditioniert. Dies nennt man *Auslöschung* (engl. catastrophic cancellation). Ein kleines Beispiel:

$$\begin{aligned} d &= (1,31\,,\; -1,25) \in \mathbb{R}^2 \\ x &= 0,06 \\ \triangle d &= (0,01\,,\; 0,01) \in \mathbb{R}^2 \\ \triangle x &= 0,02 \end{aligned}$$

$$\left|\frac{\triangle d_1}{d_1}\right|, \left|\frac{\triangle d_2}{d_2}\right| \leq 0,008$$

aber

$$\left|\frac{\triangle x}{x}\right| = 0,3\overline{3}\,,\; k = \frac{2,56}{0,06} = 42,\overline{6}\,.$$

Der relative Fehler im Ergebnis ist ca. 42mal größer als der relative Fehler in den Daten. Es muß erwähnt werden, daß das Auslöschen oft durch geeignete Transformationen vermieden werden kann, wie spätere Beispiele zeigen. Unnötiges Auslöschen ist häufig bei schlecht konditionierten Problemen vorhanden.

5.4.2 Beispiel 2: Multiplikation

$$\begin{aligned} x_1 = f_1(d) := d_1 \cdot d_2\,,\; (p = 2\,,\; q = 1) \\ k_{11} = k_{12} = 1 \end{aligned}$$

(Multiplikation ist gut konditioniert.)

5.4.3 Beispiel 3: Division

$$x_1 = f_1(d) := d_1/d_2\,,\; d_2 \neq 0 \;\; (p = 2\,,\; q = 1)$$
$$k_{11} = k_{12} = 1$$

(Division ist gut konditioniert.)

5.5 Implizite Funktionen

In den bisherigen Beispielen ist die Funktion f explizit angegeben worden. Es ist aber oft nützlich, den folgenden Satz zu benutzen:

Satz 5.7 (Implizite Funktionen) *(Maurer, Mathemecum, S. 97) Seien $U \subset \mathbb{R}^k$ und $V \subset \mathbb{R}^m$ offene Mengen und*

$$F : U \times V \longrightarrow \mathbb{R}^m\,,\; (x, y) \longmapsto F(x, y)$$

eine stetig differenzierbare Abbildung.

Dann gibt es zu jedem Punkt $(a, b) \in U \times V$, in dem die $m \times m$-Matrix

$$D_y F(a, b) = \left(\frac{\partial F_i}{\partial y_j}(a, b) \right)_{\substack{1 \leq i \leq m \\ 1 \leq j \leq m}}$$

invertierbar ist, eine Umgebung

$$U_1 \times V_1 \subset U \times V$$

und eine stetig differenzierbare Abbildung $\Phi : U_1 \longrightarrow V_1$ mit der Eigenschaft: Für alle $(x, y) \in U_1 \times V_1$ ist $F(x, y) = 0$ genau dann, wenn $y = \Phi(x)$ ist.

Die Funktionalmatrix (Jacobi-Matrix) von Φ ist gegeben durch die $m \times k$-Matrix

$$D\Phi(x) = -(D_y F(x, y))^{-1} \cdot D_x F(x, y)\,.$$

Ist F r-mal stetig differenzierbar, so auch Φ.

Der Satz über implizite Funktionen hat mehrere Erweiterungen, die es ermöglichen, Probleme wie Integralgleichungen anzupacken. **Beispiel:** Gleichungen der Gestalt $F(\mu, y) = 0$ mit $F : \mathbb{R} \times C(G) \longrightarrow C(G), G$ ein Gebiet aus $\mathbb{R}^n$,

$$F(\mu, y)(s) = y(s) - \mu \int\limits_G k(s, t) f(t, y(t)) dt$$

heißen *Hammerstein-Integralgleichungen.*

Literatur

Dolph, C.L. und **Minty**, G.J.: On nonlinear integral equations of Hammerstein type. In Nonlinear Integral Equations, Anselone, P.M. (Herausg.), University of Wisconsin Press, 1964.

Hammerstein, A.: Nichtlineare Integralgleichungen nebst Anwendungen. Acta Math. 54 (1930), 117.

5.5.1 Beispiel: Wurzeln von Polynomen 2. Grades

$$F(d,z) := p(d,z) := z^2 + d_1 z + d_o$$
$$p : \mathbb{R}^2 \times \mathbb{R}^1 \longrightarrow \mathbb{R}\,.$$

Sei z eine Wurzel von $p(d,z)$, d.h.

$$F(d,z) = p(d,z) = 0\,.$$

Ist

$$\left.\frac{\partial F}{\partial z}\right|_{(d,z)} \neq 0\,,$$

dann folgt aus dem Satz über implizite Funktionen, daß es genau eine Funktion $f(d)$ gibt mit:

1. $f : V \longrightarrow \mathbb{R}\,, V$ eine Umgebung von d,
2. $f(d) = z\,,$
3. $F(d, f(d)) = 0\,,\ s \in V\,.$

Die Ableitungen von f lassen sich direkt aus den Gleichungen

$$F(d,z) = z^2 + d_1 z + d_o = 0 \quad , \quad z := f(d)\,,$$

berechnen, z.B.

$$\begin{aligned} & 2z\,\tfrac{\partial z}{\partial d_1} \quad + \left(z + d_1\,\tfrac{\partial z}{\partial d_1}\right) + 0 = 0\,. \\ \Longrightarrow & \\ & \left.\tfrac{\partial z}{\partial d_1}\right|_d \quad = \tfrac{-z}{2z+d_1}\,. \end{aligned}$$

Es folgt:

$$|k_1| := \left|\frac{\partial z}{\partial d_1}\cdot\frac{d_1}{z}\right| = \left|\frac{z}{2z+d_1}\cdot\frac{d_1}{z}\right| = \left|\frac{d_1}{2z+d_1}\right| = \left|\frac{d_1}{z_o - z_1}\right| = \left|\frac{d_1}{s}\right| = \frac{|d_1|}{s}\,,$$

wobei $s := \sqrt{d_1^2 - 4d_o}$.

Seien x_1 und x_2 die zwei Nullstellen von $p(d, z)$. Dann gilt:

$$\begin{aligned}
|k_{11}| &= \left|\frac{\partial x_1}{\partial d_1} \cdot \frac{d_1}{x_1}\right| = \left|+\frac{1}{2}\left(-1 + \frac{d_1}{s}\right) \cdot \frac{d_1}{x_1}\right| = \frac{|d_1|}{s} \\
|k_{10}| &= \left|\frac{\partial x_1}{\partial d_o} \cdot \frac{d_o}{x_1}\right| = \left|-\frac{1}{s} \cdot \frac{d_o}{x_1}\right| = \frac{|d_o|}{s \cdot |x_1|} = \frac{x_2|}{s} \\
|k_{21}| &= \left|\frac{\partial x_2}{\partial d_1} \cdot \frac{d_1}{x_2}\right| = \frac{d_1|}{s} \\
|k_{20}| &= \left|\frac{\partial x_2}{\partial d_o} \cdot \frac{d_o}{x_2}\right| = \frac{|d_o|}{s \cdot |x_2|} = \frac{|x_1|}{s} \\
k &= \frac{2|d_1| + |x_1| + |x_2|}{s} .
\end{aligned}$$

Das Problem ist schlecht konditioniert, wenn

$$s \ll \min(|d_1|\,,\ |x_1|\,,\ |x_2|)\,.$$

Das Problem ist deshalb gut konditioniert, wenn die beiden Nullstellen z_o^o und z_o^1 getrennt sind.

Seien z_o^o und z_o^1 sehr nah oder sogar gleich, dann ist das Problem schlecht konditioniert, wie aus dem folgenden Beispiel ersichtlich:

$$\begin{aligned}
p(d) &= z^2 - 2z + 1\ ,\ d = (1, -2, 1) \\
x &= (x_1, x_2) = (1, 1) \\
p(\tilde{d}) &= z^2 - 2z + (1 - 10^{-12})\ ,\ \tilde{d} = (1, -2, 1 - 10^{-12}) \\
x &= (x_1, x_2) = (1 - 10^{-6}, 1 + 10^{-6})\ .
\end{aligned}$$

Eine Änderung von 10^{-12} in d_o hat also eine Änderung in x von 10^{-6} bewirkt.

Es gibt eine bekannte Formel für die Berechnung der Nullstellen von:

$$p(z) = az^2 + b2 + c := d_2z^2 + d_1z + d_o\ .$$

Algorithmus 1

$$\begin{aligned}
s &:= \sqrt{b^2 - 4ac} = \sqrt{d_1^2 - 4d_o \cdot d_2} \\
x_1 &:= (-b + s)/2a = \frac{-d_1 + s}{2d_2} \\
x_2 &:= (-b - s)/2a = \frac{-d_1 - s}{2d_2}\ .
\end{aligned}$$

Dieser Algorithmus ist leider wegen Auslöschung bösartig, wenn $(|d_1| - s)$ klein ist.
Es gibt einen zweiten Algorithmus, der mathematisch äquivalent und gutartig ist:

Algorithmus 2

$$\begin{aligned} x_{\max} &:= \frac{-(b + \operatorname{signum}(b) \cdot s)}{2a} \\ x_{\min} &:= \frac{c}{x_{\max} \cdot a}\,. \end{aligned}$$

Daß Algorithmus 1 bösartig und Algorithmus 2 gutartig ist, läßt sich aus den folgenden Programmergebnissen für die Polynome

$$z^2 - 10^k \cdot z + 1 = 0\,,\ k = 1, \ldots, 10$$

auf dem Rainbow ersehen:

```
type b:quadrati.pas
program quadratic(output);
var a,b,c,x1,x2,x1px2,xmin,xmax,s:real;
var k:integer;
begin
   b:=-1;
   for k:=1 to 10 do begin
   b:=b*10.; writeln('b=',b:5);
   a:=1.0;c:=1.0;
   s:=sqrt(b*b-4*a*c);
   x1:=(-b+s)/(2*a);
   x2:=(-b-s)/(2*a);
   x1px2:=x1*x2;
   if b>0 then xmax:=(-b-s)/(2*a)
          else xmax:=(-b+s)/(2*a);
   xmin:=c/(xmax*a);
   writeln('x1= ',x1,'  x2=  ',x2'  x1*x2=  ',x1px2:7);
   writeln('xmax= ',xmax,'  xmin=  ',xmin);
   end;
   end.
```

Für das Polynom

$$p(z) = z^2 - 2z + (1 - 10^{-12})$$

liefern die zwei Algorithmen auf dem Rainbow:

$$\tilde{x}_1 = \tilde{x}_{\max} = \tilde{x}_2 = \tilde{x}_{\min} = 1{,}0000000000E + 00\,,$$

was einem großen relativen Fehler entspricht. Allerdings sind $\tilde{x}_1$ und $\tilde{x}_2$ die exakten

Nullstellen des Polynoms

$$
\begin{aligned}
p(z) &= \tilde{d}_2 z^2 + \tilde{d}_1 z + \tilde{d}_o \, , \\
\tilde{d}_2 &= 1 - 10^{-12} \\
\tilde{d}_1 &= -(2 - 2 \cdot 10^{-12}) \\
\tilde{d}_o &= 1 - 10^{-12} \, , \qquad \text{so daß} \\
\|\tilde{d} - d\|_\infty &= 2 \cdot 10^{-12} \, .
\end{aligned}
$$

5.5.2 Das Wilkinsonsche Polynom

$$
p(d,z) = \sum_{k=0}^{20} d_k z^k = \prod_{r=1}^{20} (z - r) \, .
$$

Sei $z = z_r = r$ eine Nullstelle und werde der Koeffizient d_k variiert. Sei

$$
F(a,z) := p(d,z)
$$

mit

$$
d = (d_o, d_1, \cdots, d_{k-1} \, , \, a, d_{k+1}, \ldots d_{20})
$$

wobei

$$
d_o, \ldots, d_{k-1} \, , \, d_{k+1}, \ldots, d_{20}
$$

unverändert bleiben. Da

$$
\left. \frac{\partial F}{\partial z} \right|_{z=r} \neq 0 \, ,
$$

und $F(d_k, r) = 0$, folgt aus dem Satz über implizite Funktionen, daß es eine glatte Funktion $z(a)$ gibt mit $z(d_k) = r$ und $F(a, z(a)) = 0$.

Es gilt für $z = r$

$$
\begin{aligned}
\frac{\partial F}{\partial a} &= r^k \, , \\
\frac{\partial F}{\partial z} &= \left. \frac{d}{dz} \prod_{j=1}^{20} (z - j) \right|_{z=r} = \prod_{j=1}^{r-1} (r-j) \cdot \prod_{j=r+1}^{20} (r-j) = (r-1)! \, (-1)^{20-r} \, (20-r)! \, .
\end{aligned}
$$

Es folgt:

$$
\frac{\partial F}{\partial a} + \frac{\partial F}{\partial z} \frac{dz}{da} = 0,
$$

also

$$
\begin{aligned}
\frac{dz}{da} &= -\frac{\frac{\partial F}{\partial a}}{\frac{\partial F}{\partial z}} \\
\left. \frac{dz}{da} \right|_{z=r} &= \frac{r^k \, (-1)^{20-r}}{(r-1)!(20-r)!}
\end{aligned}
$$

$\left|\frac{dz_r}{da_k}\right|$ ist am größten für $k = 19$ und $r = 16$. In diesem Fall ist

$$\begin{aligned}
\left|\frac{dz_{16}}{da_{19}}\right| &= \frac{16^{19}}{4!\,16!} \doteq 0,24\ 10^{10} \\
|k_{16,19}| &= \left|\frac{\partial z_{16}}{\partial d_{19}}\right| \cdot \left|\frac{d_{19}}{z_{16}}\right| = 0,24 \times 10^{10} \cdot \frac{210}{16} = 3,15 \times 10^{10}
\end{aligned}$$

5.5.3 Polynomnullstellen

Das vorherige Beispiel läßt sich verallgemeinern.

Satz 5.8 (Fundamentalsatz der Algebra) *Jedes Polynom $P \in \mathbb{C}(t)$ mit $\deg P > 0$ hat in $\mathbb{C}$ mindestens eine Nullstelle.*

Satz 5.9 *Sei P ein Polynom, $P \in \mathbb{C}(t)$, mit $\deg\ P = n > 0$. Es gibt n Zahlen $z_i \in \mathbb{C}$ mit $P(z) = \alpha \prod_{i=1}^{n} (z - z_i)$, $\alpha \in \mathbb{C}$, $\alpha \neq 0$.*

Beweis: Induktion + Fundamentalsatz + Division mit Rest. □

5.5.4 Nichtlineare Fehleranalyse

Die bisherigen Beispiele haben immer nur das lineare Modell benutzt. Es ist manchmal nötig, die nichtlineare Fortpflanzung von Fehlern zu betrachten.

Satz 5.10 *Sei $\eta_i > 0$, $1 \leq i \leq n$ und $|\ell n\ \eta_i| \leq \epsilon$. Sei weiter $n\epsilon \leq 0,01$. Dann gilt:*

$$\left|\prod_{i=1}^{n} \eta_i - 1\right| \leq 1,02 n\epsilon\ .$$

Beispiel:

$$\begin{aligned}
x_1 &:= d_1^2 - d_2^2 \\
\tilde{x}_1 &= d_1 \cdot^* d_1 -^* d_2 \cdot^* d_2 \\
&= d_1 \cdot d_1 \cdot \eta_1 \ominus d_2 \cdot d_2 \cdot \eta_2 \\
&= (d_1^2\eta_1 - d_2^2\eta_2)\eta_3 \\
|\tilde{x}_1 - x_1| &\leq |d_1^2(\eta_1\eta_3 - 1)| + |d_2^2(\eta_2\eta_3 - 1)| \\
&\leq (d_1^2 + d_2^2) \cdot 1,02\epsilon
\end{aligned}$$

vorausgesetzt, daß $2\epsilon \leq 0,01$.

Kapitel 6

Das Gaußsche Eliminationsverfahren: Fortsetzung

6.1 Instabilität des Gaußschen Eliminationsverfahrens ohne Pivotsuche

Das Gaußsche Eliminationsverfahren ohne Pivotsuche kann numerisch instabil sein, wie die folgenden Beispiele zeigen: **Beispiel:** Das Gleichungssystem $Ax = b$ mit

$$A = \begin{pmatrix} 3 & 1 & 6 \\ 2 & 1 & 3 \\ 1 & 1 & 0 \end{pmatrix} \qquad b = \begin{pmatrix} 23 \\ 13 \\ 3 \end{pmatrix}$$

hat mehrere Lösungen, da

$$\text{rang}(A) = \text{rang}(A, b) = 2 \ ,$$

wobei rang(A) den Spaltenrang der Matrix A bezeichnet. Bekanntlich gilt:

$$\text{Spaltenrang } A = \text{Zeilenrang } A.$$

Benutzt man das Vorwärtssubstitutionsverfahren, so erhält man nach dem zweiten Schritt:

$$\begin{aligned} 3x_1 + x_2 + 6x_3 &= 23 \\ \frac{1}{3}\, x_2 - x_3 &= -2\frac{1}{3} \\ 0x_2 + 0x_3 &= 0 \ . \end{aligned}$$

Die Lösungsmenge ist:

$$L := \{x^0 + x^* : x^0 \quad \text{fest} \ , \quad Ax^0 = b \quad \text{und} \quad x^* \in \ker\ A \ , \quad \text{d.h.} \quad Ax^* = 0\} \ .$$

3.0000000000E+00	1.0000000000E+00	6.0000000000E+00	2.3000000000E+01
2.0000000000E+00	1.0000000000E+00	3.0000000000E+00	1.3000000000E+01
1.0000000000E+00	1.0000000000E+00	0.0000000000E+00	3.0000000000E+00
3.0000000000E+00	1.0000000000E+00	6.0000000000E+00	2.3000000000E+01
6.6666666667E-01	3.3333333333E-01	-1.0000000000E+00	-2.3333333333E+00
3.3333333333E-01	2.0000000000E+00	-5.4569682106E-12	-2.9103830457E-11
Die Lösung x			
-6.0000000000E+00	8.9999999999E+00	5.3333333333E+00	

Tabelle 6.1: Beispiel 1 gelöst mit dem Programm Gauß 1 auf dem Rainbow

Auf dem Rainbow erhalten wir mit Gauß-Elimination ohne Pivotsuche die Ausgabe in Tabelle 6.1. Die Erklärung hierfür ist folgende: Wegen der Rundungsfehler liefert der Rainbow nach dem zweiten Schritt des Vorwärtssubstitutionsverfahrens als dritte Gleichung:

$$-5,4569682106 \cdot 10^{-12} x_3 = -2,9103830457 \cdot 10^{-11}$$

oder

$$-3 \cdot 2^{-39} x_3 = -2^{-35} \ .$$

Hieraus ergibt sich $x_3 = 16/3$. Die Werte für x_2 und x_1 werden mit geringfügigen Fehlern während der Rückwärtssubstitution berechnet.

Das Programm berechnet deshalb die Lösung x aus der Lösungsmenge L, wofür $x_3 = 16/3$. Der Wert von x_3 ist nur durch die Rundungsfehler bestimmt, und jeder Wert von x_3 würde eine Lösung aus der Lösungsmenge L bestimmen.

Obwohl es interessant ist, daß das Programm eine Lösung liefert, ist es unbefriedigend, daß die berechnete Lösung ein fast beliebiges Element aus der Lösungsmenge ist und dem Nutzer keine Warnung gegeben wird.

Beispiel: Das Gleichungssystem

$$Ax = b$$

$$A = \begin{pmatrix} 3 & 3 & 6 \\ 2 & 2 & 3 \\ 1 & 2 & 1 \end{pmatrix} \quad , \qquad b = \begin{pmatrix} 27 \\ 15 \\ 8 \end{pmatrix}$$

hat die eindeutige Lösung $(1,2,3)^T$. Die Ergebnisse aus dem Programm Gauß 1 (Gauß-Elimination ohne Pivotsuche) sind falsch:

Nach der Vorwärtssubstitution sind die Koeffizienten:

$$\begin{array}{cccc} 3 & 3 & 6 & 27 \\ & 2^{-38} & -1 & -3 \\ & & 2^{+38} & 3 \cdot 2^{+38} \end{array}$$

3.0000000000E+00	3.0000000000E+00	6.0000000000E+00	2.7000000000E+01
2.0000000000E+00	2.0000000000E+00	3.0000000000E+00	1.5000000000E+01
1.0000000000E+00	2.0000000000E+00	1.0000000000E+00	8.0000000000E+00
3.0000000000E+00	3.0000000000E+00	6.0000000000E+00	2.7000000000E+01
6.6666666667E-01	3.6379788071E-12	-1.0000000000E+00	-3.0000000000E+00
3.3333333333E-01	2.7487790694E+11	2.7487790694E+11	8.2463372083E+11
Die Lösung x			
2.0000000000E+00	1.0000000000E+00	3.0000000000E+00	

Tabelle 6.2: Beispiel 2 gelöst mit dem Programm Gauß 1 auf dem Rainbow

Da der Koeffizient $a[2,2]$ als 2^{-38} und nicht als 0 berechnet wird, wird während der Vorwärtssubstitution ein Vielfaches der zweiten Gleichung subtrahiert, so daß die ursprünglichen Koeffizienten der dritten Gleichung keine Rolle spielen. Nach Berechnung von x_3 wird x_2 als

$$(-3 + x_3)/2^{-38}$$

berechnet, und das Ergebnis wird durch den Rundungsfehler bestimmt.

6.2 Abschätzung von Rundungsfehlern

In diesem Abschnitt werden die Rundungsfehler beim Gaußschen Eliminationsverfahren untersucht und einige Folgerungen daraus gezogen. Es wird angenommen, daß die arithmetischen Operationen die idealisierten Bedingungen erfüllen:

$$|g\ell(a\Box b) - (a\Box b)| \leq \epsilon |a\Box b| \ .$$

Satz 6.1 *Seien $\hat{L}$ und $\hat{R}$ die berechneten Dreiecksfaktoren der $n \times n$-Matrix A, die durch das Gaußsche Eliminationsverfahren mit nicht-null Pivotelementen $a_{kk}^{(k)}$ erhalten werden. Dann gilt:*

$$\begin{aligned} \hat{L}\hat{R} &= A + H \ , \\ |H| &\leq 3(n-1)\epsilon\{|A| + |\hat{L}| \cdot |\hat{R}|\} + 0(\epsilon^2) \ . \end{aligned}$$

Dabei sollen die Betragsstriche $(|.|)$ und die Kleinerrelation $(\leq)$ komponentenweise gelten, also z.B. $|A| := (|a_{ij}|)_{1\leq i,j\leq n}$ und für $n \times n$-Matrizen $A = (a_{ij})$ und $B = (b_{ij})$ ist $A \leq B$, falls $a_{ij} \leq b_{ij} \quad , \quad 1 \leq i \ , \ j \leq n$.

Beweis: Der Beweis benutzt die Induktion mit der Aussage:

$$\mathcal{A}(n) : \text{Der Satz gilt für alle } m \times m\text{-Matrizen, } m \leq n.$$

Die Richtigkeit der Aussage $\mathcal{A}(1)$ folgt sofort.
Sei die Aussage $\mathcal{A}(n)$ richtig, und sei $A^{(1)}$ eine $(n+1)\times(n+1)$-Matrix,

$$A^{(1)} = \begin{bmatrix} \alpha & w^T \\ v & B \end{bmatrix} \quad ,$$

wobei

$$\alpha \in \mathbb{R} \quad , \quad \alpha \neq 0 \quad , \quad v \in \mathbb{R}^n \quad , \quad w \in \mathbb{R}^n \, ,$$

und B eine $n \times n$-Matrix sei.
Der erste Schritt der Gaußschen Elimination liefert:

$$\hat{A}^{(2)} = g\ell(\hat{M}_1 A^{(1)}) \, ,$$

$$\hat{M}_1 := \begin{pmatrix} 1 & 0 \\ -\hat{z} & I \end{pmatrix} \, ,$$

$$\hat{z} = g\ell(v/\alpha) \, ,$$

$$\hat{A}^{(2)} := \begin{pmatrix} \alpha & w^T \\ 0 & \hat{C} \end{pmatrix} \, ,$$

$$\hat{C} := g\ell[B - \hat{z}w^T] \, .$$

Wegen $\mathcal{A}(n)$ führen die weiteren Schritte des Algorithmus zu einer approximativen Zerlegung von $\hat{C}$:

$$\begin{aligned} \hat{L}_1\hat{R}_1 &= \hat{C} + H_1 \, , \\ |H_1| &\leq 3(n-1)\epsilon\{|\hat{C}| + |\hat{L}_1|\cdot|\hat{R}_1|\} + 0(\epsilon^2). \end{aligned} \tag{6.1}$$

Die berechnete Zerlegung von A ist deshalb

$$A + H := \hat{L}\cdot\hat{R} = \begin{pmatrix} 1 & 0 \\ \hat{z} & \hat{L}_1 \end{pmatrix} \cdot \begin{pmatrix} \alpha & w^T \\ 0 & \hat{R}_1 \end{pmatrix} = \begin{pmatrix} \alpha & w^T \\ \alpha\hat{z} & \hat{z}w^T + \hat{C} + H_1 \end{pmatrix}$$

so daß

$$H = \begin{pmatrix} 0 & 0 \\ \alpha\hat{z} - v & \hat{z}w^T + \hat{C} + H_1 - B \end{pmatrix}. \tag{6.2}$$

Da die arithmetischen Operationen idealisiert sind, folgt:

$$\hat{z} = \frac{v}{\alpha} + f \, , \; |f| \leq \frac{|v|}{|\alpha|}\,\epsilon \tag{6.3}$$

$$\hat{C} = (B - \hat{z}w^T) + F \tag{6.4}$$

$$|F| \leq (|B| + 2|\hat{z}|\cdot|w|^T)\epsilon + 0(\epsilon^2). \tag{6.5}$$

Aus (6.3) bzw. (6.4) folgt

$$|\alpha|\;|f| \leq \epsilon|v| \tag{6.6}$$

und

$$|\hat{z}w^T + \hat{C} + H_1 - B| \leq |H_1| + |F|\;.$$

Aus (6.4) folgt:

$$|\hat{C}| \leq |B| + |\hat{z}| \cdot |w|^T + |F|$$

und deshalb aus (6.1) und (6.5):

$$|H_1| \leq 3(n-1)\epsilon\{|B| + |\hat{L}_1| \cdot |\hat{R}_1| + |\hat{z}| \cdot |w|^T\} + 0(\epsilon^2) \tag{6.7}$$

und aus (6.7) und (6.4):

$$|H_1| + |F| \leq 3n\epsilon\{|B| + |\hat{L}_1| \cdot |\hat{R}_1| + |\hat{z}||w|^T\} + 0(\epsilon^2). \tag{6.8}$$

Es folgt jetzt aus (6.2), (6.3), (6.6) und (6.8), daß

$$\begin{aligned}
|H| &\leq \begin{pmatrix} 0 & 0 \\ |\alpha||f| & |H_1| + |F| \end{pmatrix} \\
&\leq 3n\epsilon\left\{\begin{pmatrix} |\alpha| & |w|^T \\ |v| & |B| \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ |\hat{z}| & |\hat{L}_1| \end{pmatrix} \cdot \begin{pmatrix} |\alpha| & |w|^T \\ 0 & |\hat{R}_1| \end{pmatrix}\right\} + 0(\epsilon^2) \\
&= 3n\epsilon\{|A| + |\hat{L}| \cdot |\hat{R}|\} + 0(\epsilon^2)\;,
\end{aligned}$$

woraus die Richtigkeit von $\mathcal{A}(n+1)$ folgt. □

Auf ähnliche Weise erhält man:

Satz 6.2 *Sei L eine linke reguläre $n \times n$-Matrix, $b \in \mathbb{R}^n$, und x die Lösung von $Lx = b$. $\hat{x}$ sei durch Rückwärtssubstitution berechnet worden. Dann gilt:*

$$(L + F)\hat{x} = b$$

mit

$$|F| \leq n\epsilon|L| + 0(\epsilon^2)\;.$$

Es folgt aus den Sätzen 6.1 und 6.2:

Satz 6.3 *$\hat{L}$ und $\hat{R}$ seien die berechneten Dreiecksfaktoren der $n \times n$-Matrix A, die durch das Gaußsche Eliminationsverfahren mit nicht-null Pivotelementen erhalten werden. Sei $\hat{y}$ die berechnete Lösung von $\hat{L}y = b$ und $\hat{x}$ die berechnete Lösung von $\hat{R}x = \hat{y}$. Die arithmetischen Operationen erfüllen die idealisierten Bedingungen:*

$$|g\ell(a\Box b) - (a\Box b)| \leq \epsilon|(a\Box b)|\;.$$

Dann gilt:

$$(A+E)\hat{x} = b\ ,$$

wobei

$$|E| \leq n\epsilon[3|A| + 5|\hat{L}| \cdot |\hat{R}|] + 0(\epsilon^2)\ .$$

Beweis: Aus Satz 6.1 folgt:

$$\begin{aligned} \hat{L}\hat{R} &= A+H\ , \\ |H| &\leq 3n\epsilon(|A| + |\hat{L}| \cdot |\hat{R}|) + 0(\epsilon^2)\ . \end{aligned}$$

Aus Satz 6.2 folgt:

$$(\hat{L}+F)\hat{y} = b\ ,$$
$$|F| \leq n\epsilon|\hat{L}| + 0(\epsilon^2)$$

und

$$(\hat{R}+G)\hat{x} = \hat{y}$$
$$|G| \leq n\epsilon|\hat{R}| + 0(\epsilon^2)\ .$$

Zusammenfassend:

$$(\hat{L}+F)(\hat{R}+G)\hat{x} = b$$

oder

$$(A + H + F\hat{R} + \hat{L}G + FG)\hat{x} = b$$

also

$$(A+E)\hat{x} = b$$

mit

$$E := H + F\hat{R} + \hat{L}G + FG$$

und

$$\begin{aligned} |E| &\leq 3n\epsilon(|A| + |\hat{L}||\hat{R}|) + 0(\epsilon^2) + (n\epsilon|\hat{L}| + 0(\epsilon^2)) \cdot |\hat{R}| \\ &+ |\hat{L}| \cdot (n\epsilon|\hat{R}| + 0(\epsilon^2)) + (n\epsilon|\hat{L}| + 0(\epsilon^2))(n\epsilon|\hat{R}| + 0(\epsilon^2))\ . \end{aligned}$$

□

Bemerkung: Die ersten (sehr pessimistischen) Fehlerabschätzungen wurden von Goldstine und von Neumann [1951] hergeleitet. Viel bessere Abschätzungen wurden zuerst von Wilkinson [1961] bewiesen. Die o.a. Beweise stammen von de Boor und Pinkus [1977].

6.3 Vektor- und Matrixnormen

Um Fehler in linearen Räumen zu analysieren, sind Normen sehr nützlich:

Definition 6.1 *Sei V ein $\mathbf{K}$-Vektorraum. Eine Abbildung*

$$\|\cdot\| : V \longrightarrow \mathbb{R}\,, \quad v \longmapsto \|v\|$$

heißt Norm *auf V, falls für alle $v, w \in V$ und $\lambda \in \mathbf{K}$ gilt:*

1. $\|\lambda v\| = |\lambda| \cdot \|v\|$
2. $\|v + w\| \le \|v\| + \|w\|$ *(Dreiecksungleichung)*
3. $\|v\| = 0 \Longleftrightarrow v = 0.$

Die reelle Zahl $\|v\|$ heißt *Norm* des Vektors v.

Ein Paar $(V, \|\ \|)$, wobei V ein $\mathbf{K}$-Vektorraum und $\|\ \|$ eine Norm auf V ist, heißt *normierter Vektorraum.*

Die Norm $\|v\|$ eines Vektors v ist ein Maß für die „Größe" von v.

In $\mathbb{R}^n$ und $\mathbb{C}^n$ werden mehrere Normen benutzt:

$$\begin{aligned}
\|x\|_2 &:= \sqrt{x^H x} = \sqrt{\sum_{i=1}^{n} |x_i|^2} \qquad \textit{(euklidische Norm)} \\
\|x\|_\infty &:= \max_i |x_i| \qquad (\textit{Maximumnorm} \text{ oder } \textit{Sup-Norm}) \\
\|x\|_1 &:= \sum_{i=1}^{n} |x_i| \qquad (\ell_1\textit{–Norm}) \\
\|x\|_p &:= \left[\sum |x_i|^p\right]^{1/p}, \; p \neq \infty \qquad (p\textit{–Norm})
\end{aligned}$$

Selbstverständlich erfüllen all diese Normen die drei Bedingungen für eine Norm.

Satz 6.4 *Alle Normen auf $V = \mathbb{R}^n$ oder $V = \mathbb{C}^n$ sind äquivalent: Seien $\|\cdot\|_\alpha$ und $\|\cdot\|_\beta$ zwei Normen, dann gibt es Konstanten $m = m(\alpha, \beta) > 0$ und $M = M(\alpha, \beta) > 0$, so daß gilt:*

$$m\|x\|_\alpha \le \|x\|_\beta \le M\|x\|_\alpha\,.$$

Beweis: Sei $\|\cdot\|$ eine Norm auf V. Es gilt:

$$|\, \|x + h\| - \|x\|\, | \le \|h\| \quad , \quad \text{für alle} \quad x, h \in V\,,$$

so daß

$$|\, \|x + h\| - \|x\|\, | \le \sum_{i=1}^{n} |h_i|\, \|e_i\| \le n \cdot \|h\|_\infty \cdot \max_i \|e_i\|$$

wobei e_i den i-ten Achsenvektor bezeichnet. Also gilt für jedes $\epsilon > 0$ und alle h mit

$$\max_i \ |h_i| < \frac{\epsilon}{n \cdot \max_i \ \|e_i\|}$$

dßa

$$| \ \|x + h\| - \|x\| \ | < \epsilon \ .$$

Die Abbildung $\|\cdot\|$ ist also (gleichmäßig) stetig.

Nun sei

$$S := \{x \in V : \|x\|_1 = 1\} \ .$$

Die Menge S ist beschränkt und abgeschlossen, also kompakt (Satz von Heine-Borel, siehe Forster, Analysis 2, Seite 21). Die Abbildung $\|\cdot\| : S \to \mathbb{R}$ nimmt ihr Maximum und Minimum an (Forster, Analysis 2, Seite 22). Sei

$$\begin{aligned} m &= \inf_{x \in S} \ \|x\| = \|x^0\| \ , \\ M &= \max_{x \in S} \ \|x\| = \|x^1\| \ . \end{aligned}$$

Es gilt $m > 0$, da sonst $\|x^0\| = 0$, was $x^0 \in S$ widerspricht. Es gilt auch $M < \infty$.

Sei nun $x \in V$. Ist $x = 0$, dann gilt:

$$m \cdot \|x\| \le \|x\|_\infty \le M \cdot \|x\| \ .$$

Ist $x > 0$, dann ist $x/\|x\|_1 \in S$, so daß

$$\begin{aligned} \left\| \frac{x}{\|x\|_1} \right\| &= \frac{\|x\|}{\|x\|_1} \ge m \ , \qquad \text{und} \\ \left\| \frac{x}{\|x\|_1} \right\| &= \frac{\|x\|}{\|x\|_1} \le M \ . \end{aligned}$$

Es folgt

$$m \cdot \|x\|_1 \le \|x\| \le M \cdot \|x\|_1 \ ,$$

d.h. die Normen $\|\cdot\|$ und $\|\cdot\|_1$ sind äquivalent. □

Beispiel:

$$\begin{aligned} \|x\|_2 &\le \|x\|_1 \le \sqrt{n} \ \|x\|_2 \\ \|x\|_\infty &\le \|x\|_2 \le \sqrt{n} \ \|x\|_\infty \\ \|x\|_\infty &\le \|x\|_1 \le n \ \|x\|_\infty \end{aligned}$$

Die Matrizen $A \in M(m, n)$ bilden einen Vektorraum V, auf dem man Normen definieren kann. Es muß gelten:

1. $\|\alpha A\| = |\alpha| \cdot \|A\|$,

2. $\|A+B\| \leq \|A\| + \|B\|$,

3. $\|A\| = 0 \Longleftrightarrow A = 0$.

Die Matrixnorm $\|\cdot\|$ heißt mit den Vektornormen $\|\cdot\|_a$ auf dem $\mathbb{C}^n$ und $\|\cdot\|_b$ auf dem $\mathbb{C}^m$ *verträglich*, falls

$$\|Ax\|_b \leq \|A\| \cdot \|x\|_a \,, \quad \text{für alle} \quad x \in \mathbb{C}^n \;,\; A \in \mathrm{Mat}(m,n) \;.$$

Seien $\|\cdot\|_a$ und $\|\cdot\|_b$ Vektornormen auf dem $\mathbb{C}^n$ bzw. $\mathbb{C}^m$, dann ist

$$\|A\|_{a,b} := \sup_{\|x\|_a=1} \|Ax\|_b$$

verträglich mit den Vektornormen.

6.4 Pivotsuche

Es folgt aus Satz 6.3, daß die Rundungsfehler begrenzt sind, wenn die Matrizen $\hat{L}$ und $\hat{R}$ beschränkt sind. Um zu erreichen, daß $\hat{L}$ und $\hat{R}$ möglichst klein sind, werden Methoden zur Auswahl der Pivotelemente angewandt:

1. **Teilpivotsuche oder Spaltenpivotsuche:** Beim k-ten Schritt der Gaußschen Elimination mit Permutationen wird $s_k \geq k$ berechnet, so daß

   $$|a^{(k)}_{s_k k}| \geq |a^{(k)}_{ik}| \quad , \quad \text{für} \quad i \geq k \;.$$

   Die Permutationsmatrix P_k entspricht der Permutation

   $$\sigma := \begin{pmatrix} \dots & k & \dots & s_k & \dots \\ \dots & s_k & \dots & k & \dots \end{pmatrix}$$

   Bei Multiplikation mit P_k werden also die k-te Zeile und die s_k-te Zeile vertauscht.

2. **Totalpivotsuche:** Beim k-ten Schritt der Gaußschen Elimination mit Permutationen werden $s_k \geq k$ und $t_k \geq k$ berechnet:

   $$|a^{(k)}_{s_k,t_k}| \geq |a^{(k)}_{ij}| \quad , \quad \text{für} \quad k \leq i \;,\; j \leq n \;.$$

   Die Permutationsmatrizen P_k und π_k werden so gewählt, daß sie die Vertauschungen der k-ten und s_k-ten Zeile und der k-ten und t_k-ten Spalte bewirken.

   Es wird oft behauptet, daß die Totalpivotsuche im Vergleich zur Spaltenpivotsuche sehr zeitaufwendig ist. Diese Behauptung gilt nur für bestimmte Rechner. Sie gilt z.B. nicht für

- Menschen
- Feldrechner wie der DAP.

Bei Menschen ist die Totalpivotsuche ein schneller Prozeß. Beim DAP ist die Totalpivotsuche genau so schnell wie die Spaltenpivotsuche. Es ist natürlich richtig, daß bei den meisten von Neumann Rechnern die Totalpivotsuche viel mehr Zeit erfordert als die Spaltenpivotsuche.

3. **Dynamische Pivotsuche:** Wenn die Matrix A so groß ist, daß nur einige Zeilen zu einem Zeitpunkt im Hauptspeicher gespeichert werden können, ist eine dritte Möglichkeit von Interesse.

   Bevor x_k in der i-ten Gleichung eliminiert wird, werden die k-te und i-te Zeile vertauscht, falls
$$|a_{kk}^{(k)}| < |a_{ik}^{(k)}| \; .$$

Werden Totalpivotsuche, Teilpivotsuche oder Dynamische Pivotsuche benutzt, dann gilt

$$|(P_k \hat{A}^{(k)} \pi_k)_{kk}| \geq |(P_k \hat{A}^{(k)} \pi_k)_{ik}| \quad , \quad i \geq k \; ,$$

wobei $\hat{A}^{(k)} := g\ell(A^{(k)})$. Es folgt:
$$|\hat{\ell}_{ik}| \leq 1$$

und

$$\|\hat{L}\|_\infty \leq n \; .$$

Mit der Bezeichnung

$$\rho(n, A) := \max \left\{ 1, \max_{1 \leq i,j,k \leq n} \; |\hat{a}_{ij}^{(k)}| / \|A\|_\infty \right\}$$

erhalten wir aus Satz 6.3:

Satz 6.5 *Die Voraussetzungen von Satz 6.3 seien erfüllt. Total-, Spalten- oder Dynamische Pivotsuche werden benutzt. Dann gilt:*

$$(A + E)\hat{x} = b \; ,$$

wobei

$$\begin{aligned} \|E\|_\infty &\leq n\epsilon[3\|A\|_\infty + 5n\|\hat{R}\|_\infty] + 0(\epsilon^2) \\ &\leq n\epsilon[3\|A\|_\infty + 5n^2 \cdot \max_{i,j,k} \; |\hat{a}_{ij}^{(k)}|] + 0(\epsilon^2) \end{aligned}$$

und deshalb
$$\|E\|_\infty \leq 8n^3 \rho(n, A) \|A\|_\infty \epsilon + 0(\epsilon^2) \; .$$

Mit der Bezeichnung:

$$g(n,A) := \max |a_{ij}^{(k)}| / \max |a_{ij}^{(1)}| \quad , \quad 1 \le i,j,k \le n$$

($g(n,A)$ heißt der *Wachstumsfaktor*) ergibt sich:

Satz 6.6 *Die Voraussetzungen von Satz 6.3 seien erfüllt. Total-, Spalten- oder Dynamische Pivotsuche werden benutzt. Dann gilt:*

$$(A+E)\hat{x} = b ,$$

wobei

$$|E| \le 8n^2 \epsilon g(n,A) \max |a_{ij}^{(1)}| + 0(\epsilon^2) .$$

Weiter gilt:

$$\begin{aligned} \|E\|_\infty &\le 8n^3 \epsilon g(n,A) \max |a_{ij}^{(1)}| + 0(\epsilon^2) \\ &\le 8n^3 \epsilon g(n,A) \|A\|_\infty + 0(\epsilon^2) \end{aligned}$$

Es entsteht sofort die Frage, ob es möglich ist, $g(n,A)$ abzuschätzen. Folgendes ist bekannt:

1. Bei *Spaltenpivotsuche* kann gezeigt werden:

$$g(n,A) \le 2^{n-1}$$

und diese Schranke kann jedoch erreicht werden, z.B. für

$$A := \begin{pmatrix} 1 & 0 & & \dots & 0 & 1 \\ -1 & 1 & 0 & \dots & 0 & 1 \\ -1 & -1 & 1 & \dots & 0 & 1 \\ & & & & 1 & 1 \\ -1 & -1 & -1 & \dots & -1 & 1 \end{pmatrix}$$

Für spezielle Matrizen ist mehr bekannt:

(a) $g(n,A) \le n$ für obere Hessenberg Matrizen A. (A heißt eine obere Hessenberg-Matrix, falls $a_{ij} = 0 \quad , \quad i \ge j+2$.)

(b) $g(n,A) \le 2$ für Tridiagonalmatrizen A. (A heißt eine Tridiagonalmatrix, falls $a_{ij} = 0$ für $|i-j| > 1$.)

(c) $g(n,A) \le 2$, wenn A das starke Spaltensummenkriterium erfüllt:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \ne i}} |a_{ij}| \quad , \quad 1 \le i \le n .$$

(d) $g(n, A) \leq 1$, wenn A symmetrisch und positiv definit ist: $A^T = A$ und $x^T Ax \geq 0$ für $x \in \mathbb{R}^n$.

2. Bei Totalpivotsuche hat Wilkinson gezeigt:

$$g(n, A) \leq f(n) \;,$$

wobei

$$f(n) := \sqrt{n}\left[2^1 3^{1/2} 4^{1/3} \dots k^{\frac{1}{k-1}}\right]^{1/2} \;,$$

k	10	20	500	100
$f(k)$	19	67	530	3300

Es gilt auch (Wilkinson):

$$f(n) \leq 1,8\; n^{1/4\, \ell n\, n} \;.$$

Wilkinsonsche Vermutung

$$g(n, A) \leq n \quad , \quad n \in \mathbb{N} \;.$$

Diese Vermutung bewies Cryer 1968 für $n = 4$. Die Vermutung ist durch Tornheim für $n = 5$ bewiesen (nicht veröffentlicht), bleibt für $n > 5$ noch offen.

Higham und Higham [1988] geben für jedes n eine $n \times n$- Matrix C_n an, wofür (bei Totalpivotsuche) $g(n, C_n) \geq n/2$ gilt.

Trefethen und Schreiber [1988] haben Gaußsche Elimination auf eine große Population von Matrizen angewandt und stellen fest, daß

$$\begin{array}{lcll} E(g(n,A)) & \sim & n^{1/2} & \text{Totalpivotsuche} \\ E(g(n,A)) & \sim & n^{2/3} & \text{Spaltenpivotsuche} \end{array}$$

wobei $E(x)$ den Erwartungswert von x bezeichnet.

Es folgt hieraus, daß das Gaußsche Eliminationsverfahren mit Pivotsuche numerisch stabil ist.

3. **Gaußsche Elimination ohne Pivotsuche:** In einigen wichtigen Spezialfällen ist es nicht nötig, Pivotsuche zu benutzen.

(a) Matrizen, die das starke Spaltensummenkriterium erfüllen.

(b) Symmetrische positiv definite Matrizen. (Diese Behauptung folgt nicht aus den vorstehenden Ergebnissen. Weitere Überlegungen sind nötig.)

Die Matrix L ist begrenzt, da

$$\ell_{ij}^2 \leq \sum_{p=1}^{i} \ell_{ip}^2 = a_{ii} \; .$$

Wilkinson hat bewiesen, daß

$$(A+E)x = b \; ,$$

wobei

$$\|E\|_2 \leq c_n \epsilon \|A\|_2$$

und c_n eine kleine Konstante ist. Gilt weiter

$$q_n \epsilon \|A\|_2 \, \|A^{-1}\|_2 \leq 1 \; ,$$

wobei q_n eine kleine Konstante ist, dann ist das Cholesky-Verfahren wohldefiniert.

Die Fehlerabschätzung in den Sätzen 6.5 und 6.6 ist eine Abschätzung, und die numerischen Ergebnisse sind oft viel besser.

Heuristik I (Wilkinson)
Die numerischen Lösungen von gestaffelten Gleichungssystemen sind oft viel genauer als die Fehlerabschätzung in Satz 6.1.

Heuristik II (Wilkinson)
In der Praxis ist der Wachstumsfaktor $g(n, A)$ in den meisten Fällen nicht größer als 10, wenn Totalpivotsuche oder Spaltenpivotsuche benutzt wird.

Heuristik III
Das Residuum $\hat{r} := b - A\hat{x}$, wobei $\hat{x}$ durch das Gaußsche Verfahren berechnet sei, ist meistens sehr klein. Es gilt oft

$$\|\hat{r}\| \leq \|(A - rd(A)) \cdot b\| \; .$$

Das Residuum der berechneten Lösung ist also oft kleiner als das Residuum, das entsteht, wenn die Matrix A gerundet wird.

Literatur

Higham, N.J. and **Higham**, D.J.: Large growth factors in Gaussian elimination with pivoting. Numerical Analysis Report No. 152, University of Manchester, January 1988.

Trefethen, L.N. and **Schreiber**, R.S.: Average-case stability of Gaussian elimination. Numerical Analysis Report No. 88-3, Department of Mathematics, Massachusetts Institute of Technology.

6.5 Konditionszahlen und Fehlerabschätzung

In diesem Abschnitt wird vorausgesetzt, daß $\|A\|$ eine Matrixnorm bezeichnet, die der Vektornorm $\|\cdot\|$ zugeordnet ist. Wir haben in Kapitel 6 Konditionsfragen diskutiert (siehe Abbildung 6.1).

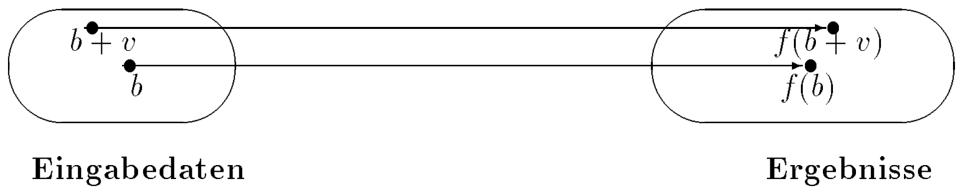


Abbildung 6.1: Berechnung der Konditionszahl

Für das Problem: Berechne die Lösung $x = f(b)$ des Gleichungssystems $Ax = b$, ist die Konditionszahl C_p leicht zu berechnen:

$$
\begin{aligned}
C_p(x) \quad &:= \quad \limsup_{\delta\to 0} \sup_{0<\|v\|<\delta} \frac{\|f(b+v)-f(b)\|/\|f(b)\|}{\|(b+v)-b\|/\|b\|} \\
&= \quad \limsup_{\delta\to 0} \sup_{0<\|v\|<\delta} \frac{\|A^{-1}(b+v)-A^{-1}(b)\|/\|A^{-1}(b)\|}{\|v\|/\|b\|} \\
&= \quad \limsup_{\delta\to 0} \sup_{0<\|v\|<\delta} \frac{\|A^{-1}v\|/\|A^{-1}b\|}{\|v\|/\|b\|} \\
&\leq \quad \|A^{-1}\| \cdot \frac{\|Ax\|}{\|x\|} \,.
\end{aligned}
$$

Es folgt:

$$\sup_x \; C_p(x) = \|A^{-1}\| \cdot \|A\|$$

und wir setzen

$$\kappa := \mathsf{cond}(A) := \|A^{-1}\| \cdot \|A\| \,.$$

κ heißt die *Konditionszahl* von A. Wird die p-Norm $\|\cdot\|_p$ benutzt, schreibt man $\kappa_p(A)$.

Satz 6.7 *Es sei A eine reguläre $n \times n$-Matrix mit Konditionszahl κ , $b \in \mathbb{R}^n$,*

$$
\begin{aligned}
Ax \quad &= \quad b \\
(A+\delta A)\hat{x} \quad &= \quad b+\delta b \,.
\end{aligned}
$$

Dann gilt:

$$\|x-\hat{x}\| \le \left[\frac{\|\delta b\|}{\|b\|}\cdot\|x\| + \frac{\|\delta A\|}{\|A\|}\cdot\|\hat{x}\|\right]\cdot cond(A)$$

Beweis: Da

$$\hat{x}-x = A^{-1}[\delta b - \delta A\cdot\hat{x}]$$

folgt

$$\begin{aligned}\|\hat{x}-x\| &\le \|A^{-1}\|\cdot\left[\frac{\|\delta b\|}{\|b\|}\cdot\|Ax\| + \frac{\|\delta A\|}{\|A\|}\cdot\|A\|\cdot\|\hat{x}\|\right]\\ &\le \kappa\left[\frac{\|\delta b\|}{\|b\|}\cdot\|x\| + \frac{\|\delta A\|}{\|A\|}\cdot\|\hat{x}\|\right].\end{aligned}$$

□

Die Konditionszahl ist deshalb ein Maßstab für die Empfindlichkeit der Gleichung $Ax=b$ gegen Störungen in A oder b.

Für jede reguläre Matrix A gilt:

$$\kappa(A) = \|A\|\cdot\|A^{-1}\| \ge \|A\cdot A^{-1}\| = \|I\| = 1.$$

$\kappa(A)$ ist „klein" für gutkonditionierte Probleme und „groß" für schlechtkonditionierte Probleme.

Es gibt eine zweite äquivalente Definition von κ :

Satz 6.8 (Gastinel)

$$[\kappa(A)]^{-1} = \inf_{A+E\,\text{sing.}}\frac{\|E\|}{\|A\|} \quad , \quad \mathit{mit} \quad \|M\| = \sup_x\frac{\|Mx\|}{\|x\|}\ .$$

Beweis:

1. Sei $A+E$ singulär. Dann gibt es $x\neq 0 \quad , \quad (A+E)x=0$. Es folgt

$$\|E\| \ge \|Ex\|/\|x\| = \|Ax\|/\|x\| = \|y\|/\|A^{-1}y\| \ge 1/\|A^{-1}\|$$

   für $y=Ax$.

2. Sei $\|A^{-1}\| = \|A^{-1}y\|/\|y\|$, also $\|A^{-1}y\| = \|A^{-1}\|\cdot\|y\| > 0$. Man setze $w := \alpha A^{-1}y$, wobei α so gewählt wird, daß

$$w^T A^{-1}y = 1\ .$$

   Man setze:

$$E := -yw^T\ .$$

Es folgt:

$$(A+E)A^{-1}y = y - yw^T A^{-1}y = 0 \ ,$$

so daß $A+E$ singulär ist. Es folgt auch, daß

$$\begin{aligned}
\|E\| &= \max_x \|yw^T x\|/\|x\| \\
&= \|y\| \max_x |w^T x|/\|x\| \\
&= \|y\| \cdot |w^T w|/\|w\| \\
&= \|y\| \cdot |w^T \alpha A^{-1} y|/\|w\| \\
&= \|y\| \cdot |\alpha|/\|w\| \\
&= \|y\| \cdot |\alpha|/(|\alpha| \ \|A^{-1}y\|) \\
&= 1/\|A^{-1}\| \ .
\end{aligned}$$

□

Neue Ergebnisse diesbezüglich findet man in den Arbeiten von Demmel [1985, 1987].

Es gibt leider keine zuverlässige Methode, um eine schlechtkonditionierte Matrix zu erkennen, außer, die Inverse A^{-1} zu berechnen.

Beispiel:

$$A_n := \begin{pmatrix} 1 & -1 & & \dots & -1 \\ & 1 & & & -1 \\ & & \ddots & & -1 \\ & & & 1 & \\ & O & & & \\ & & & & 1 \end{pmatrix} \in \mathrm{Mat}(n,n) \ .$$

Es gilt $\det(A_n) = 1$, und alle Pivotelemente sind 1, aber

$$A_n^{-1} = \begin{pmatrix} 1 & 1 & 2 & \dots & \dots & 2^{n-2} \\ & 1 & 1 & & & 2^{n-3} \\ & & 1 & 1 & 2 & 4 \\ & & & 1 & 1 & 2 \\ & & & & 1 & 1 \\ & & & & & 1 \end{pmatrix}$$

so daß

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = n \ 2^{n-1} \ .$$

Es gilt aber folgendes:

Satz 6.9 *(siehe Golub und van Loan [1983, S. 78]) Sei $P(A+E) = LR$, wobei P eine Permutationsmatrix, L eine linke Dreiecksmatrix mit $|\ell_{ij}| \leq 1 \quad , \quad \ell_{ii} = 1$, und R eine rechte Dreiecksmatrix ist. Dann gilt:*

$$\kappa_\infty(A) \geq \frac{\|A\|_\infty}{\|E\|_\infty + \inf |r_{kk}|} \, .$$

Es folgt aus Satz 6.9, daß A schlechtkonditioniert ist, falls ein kleines Pivotelement bei der Gaußschen Elimination entsteht.

Wilkinson zufolge gibt es drei mögliche Hinweise, daß A schlechtkonditioniert ist:

1. ein kleines Pivotelement
2. eine große Lösung $\hat{x}$
3. ein großes Residuum $r = b - A\hat{x}$.

Es kann aber vorkommen, daß keiner von diesen Hinweisen vorhanden ist, obwohl A schlechtkonditioniert ist.

Ist eine Approximation zu A^{-1} vorhanden, ist es möglich, den Fehler abzuschätzen:

Satz 6.10 *Es seien $\hat{x}$ eine Approximation zur Lösung von x von $Ax = b$ und C eine $n \times n$-Matrix*

$$\begin{aligned} r &:= b - A\hat{x} \\ F &:= I - CA \, . \end{aligned}$$

Dann gilt:

$$(1 - \|F\|) \, \|\hat{x} - x\| \leq \|Cr\| \, .$$

Beweis:

$$A(\hat{x} - x) = -r \, .$$

Nach Multiplikation mit C :

$$\begin{aligned} CA(\hat{x} - x) &= -Cr \, , \\ (\hat{x} - x) &= (I - CA)(\hat{x} - x) - Cr \, . \end{aligned}$$

Es folgt:

$$\|x - \hat{x}\| \leq \|F\| \, \|\hat{x} - x\| + \|Cr\| \, .$$

□

Satz 6.10 bietet die Möglichkeit, den Fehler $\hat{x} - x$ abzuschätzen. Die Fehlerabschätzung erfordert viel Arbeit, sollte aber erwogen werden, falls n nicht zu groß ist und A keine bekannten Eigenschaften hat. Es wird im allgemeinen davon abgeraten, diese Fehlerabschätzung zu benutzen, aber der Verfasser glaubt, daß Satz 6.10 eine interessante Möglichkeit anbietet.

Heuristik IV
Das Gaußsche Eliminationsverfahren erzeugt eine Lösung $\hat{x}$ mit ungefähr

$$t \ \log_{10} d - \log_{10}[\kappa(A)]$$

Dezimalziffern, die genau sind.

Dabei ist d die interne Basiszahl und t die Länge der Mantisse bei Gleitkommaarithmetik.

Begründung: Wegen Satz 6.5 und der Heuristik III ist

$$(A + E)\hat{x} = b \quad , \quad \|E\| \approx d^{-t}\|A\| \, .$$

Es folgt aus Satz 6.7, daß

$$\frac{\|x - \hat{x}\|}{\|\hat{x}\|} \approx \mathrm{cond}(A) \cdot d^{-t} \, ,$$

so daß

$$-\log_{10} \frac{\|x - \hat{x}\|}{\|\hat{x}\|} \approx t \ \log_{10} d - \log_{10}(\mathrm{cond}(A)) \, .$$

Beispiel: Die Hilbert-Matrix H_n ist das bekannteste Beispiel einer schlechtkonditionierten Matrix:

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \cdots & & \frac{1}{n} \\ \frac{1}{2} & & & & \\ \vdots & \cdots & \frac{1}{(i+j-1)} & & \\ \frac{1}{n} & \cdots & \cdots & \cdots & \frac{1}{(2n-1)} \end{pmatrix}$$

Die Matrizen H_n entstehen bei einigen ungeschickten Ausgleichrechnungen. Eigenschaften von H_n:

1. H_n ist symmetrisch

2. H_n ist positiv definit

3. $n \ \max|(H_n)_{ij}| \cdot \max|(H_n^{-1})_{ij}| \sim \mathrm{const.} \ e^{3{,}525n}$

4. $\kappa_2(H_n) = \frac{\rho(H_n)}{\rho(T_n)} \doteq e^{3 \cdot 5n}$ mit $T_n := H_n^{-1}$.

5. Die Matrizen T_n werden tabelliert, da sie einfachere Elemente als die Matrizen H_n besitzen.

n	$\|H_n\|$	$\|T_n\|$	cond(H_n)	Größtes Element vonT_n
2	1.27	$1.52_{10}1$	$1.93_{10}1$	$1.20_{10}1$
3	1.41	$3.72_{10}2$	$5.24_{10}2$	$1.92_{10}2$
4	1.50	$1.03_{10}4$	$1.55_{10}4$	$6.48_{10}3$
5	1.57	$3.04_{10}5$	$4.77_{10}5$	$1.79_{10}5$
6	1.62	$9.24_{10}6$	$1.50_{10}7$	$4.41_{10}6$
7	1.66	$2.86_{10}8$	$4.75_{10}8$	$1.33_{10}8$
8	1.70	$9.00_{10}9$	$1.53_{10}10$	$4.25_{10}9$
9	1.73	$2.86_{10}11$	$4.93_{10}11$	$1.22_{10}11$
10	1.75	$9.15_{10}12$	$1.60_{10}13$	$3.48_{10}12$

Tabelle 6.3: Merkmale der Hilbert-Matrizen H_n und deren Inversen T_n

n	$\|\hat{x}-x\|_\infty$	$\|\hat{F}\|_\infty$	$\|C\|_\infty\|\hat{r}\|_\infty/(1-\|\hat{F}\|_\infty)$	$\|C\hat{r}\|_\infty/(1-\|\hat{F}\|_\infty)$
1	$0,0\cdot 10^{+0}$	$0,0\cdot 10^{+0}$	$0,0\cdot 10^{+0}$	$0,0\cdot 10^{+0}$
2	$0,0\cdot 10^{+0}$	$0,0\cdot 10^{+0}$	$0,0\cdot 10^{+0}$	$0,0\cdot 10^{+0}$
3	$1,5\cdot 10^{-10}$	$1,9\cdot 10^{-10}$	$7,4\cdot 10^{-10}$	$2,2\cdot 10^{-11}$
4	$1,7\cdot 10^{-8}$	$4,7\cdot 10^{-9}$	$5,0\cdot 10^{-8}$	$9,8\cdot 10^{-9}$
5	$4,2\cdot 10^{-7}$	$1,2\cdot 10^{-7}$	$1,6\cdot 10^{-6}$	$4,3\cdot 10^{-7}$
6	$6,2\cdot 10^{-7}$	$4,1\cdot 10^{-6}$	$4,3\cdot 10^{-5}$	$1,4\cdot 10^{-5}$
7	$9,0\cdot 10^{-4}$	$6,2\cdot 10^{-5}$	$2,8\cdot 10^{-3}$	$7,5\cdot 10^{-4}$
8	$7,8\cdot 10^{-3}$	$3,8\cdot 10^{-3}$	$9,1\cdot 10^{-2}$	$2,7\cdot 10^{-2}$
9	$1,5\cdot 10^{-1}$	$9,8\cdot 10^{-2}$	$6,4\cdot 10^{+0}$	$2,4\cdot 10^{-2}$
10	$2,4\cdot 10^{+1}$	$9,1\cdot 10^{+0}$	$-2,2\cdot 10^{+2}$	$-1,7\cdot 10^{+1}$

Tabelle 6.4: Fehler $\|\hat{x}-x\|_\infty$ bei der Lösung auf dem Commodore PC 10 der Gleichung $H_n x = b \quad , \quad x=(1,2,\ldots,n)^T \in \mathbb{R}^n \quad , \quad C \approx A^{-1} \quad , \quad r := b - A\hat{x} \quad , F : I - CA$.

Die Matrizen T_n für $n = 1$ bis 10 sind in Gregory und Karney [1969, S. 34] aufgelistet.

Es folgt aus Tabelle 6.3 und Heuristik IV, daß, wenn das Gleichungssystem

$$H_n x = b$$

mit einem Pascal Programm auf dem Rainbow ($t = 40$, $d = 2$) gelöst wird, wahrscheinlich nur

$$40 \ \log_{10} 2 - \log_{10} e^{3 \cdot 5n} = 40 \ \log_{10} 2 - 3 \cdot 5n \ \log_{10} e = 12,04 - 1,52n$$

Dezimalziffern exakt sind. Insbesondere ist keine Ziffer exakt für $n \geq 8$. Diese heuristische Überlegung wird in der Praxis bestätigt, wie aus Tabelle 6.4 hervorgeht.

Es ist in Tabelle 6.4 bemerkenswert, daß $\|x-\hat{x}\|$ etwas größer ist als die Fehlerabschätzung $\|Cr\|/(1-\|\hat{F}\|)$. Die Erklärung hierfür ist, daß die Koeffizienten von H_n gerundet sind. Es entsteht deshalb ein weiterer Fehler cond$(H)(\|\delta H\|/\|H\|)$.

Literatur

Demmel, J.: On condition numbers and the distance to the nearest ill-posed problem. Technical Report No. 191, Dept. of Computer Science, New York University, December 1985.

Demmel, J.: On condition numbers and the distance to the nearest ill-posed problem. Technical Report No. 293, Dept. of Computer Science, New York University, April 1987.

Gregory, R.T. and **Karney**, D.L.: A Collection of Matrices for Testing Computational Algorithms. New York: John Wiley, 1969.

6.6 Iterative Verbesserung

Der folgende Algorithmus wird als *Iterative Verbesserung* bezeichnet:

1. Man berechne die LR-Zerlegung von A mit t-Arithmetik:

$$A \doteq \hat{L}\hat{R}$$

2. Man berechne eine Lösung $\hat{x}^{(1)}$ der Gleichung $Ax = b$ mit t-Arithmetik:

$$\begin{aligned} \hat{L}\hat{y}^{(1)} &= b \\ \hat{R}\hat{x}^{(1)} &= \hat{y}^{(1)} \end{aligned}$$

3. Man wiederhole folgende Schritte bis die Konvergenz erreicht ist, oder eine festgestellte Anzahl Schritte durchgeführt worden ist:

$$\begin{aligned} \hat{z}^{(k)} &= b - A\hat{x}^{(k)} && \text{(2t- Arithmetik)} \\ \hat{r}^{(k)} &= rd(\hat{z}^{(k)}) && \text{(t-Arithmetik)} \\ \hat{L}\hat{y}^{k+1} &= r^{(k)} && \text{(t-Arithmetik)} \\ \hat{R}\hat{c}^{k+1} &= \hat{y}^{k+1} && \text{(t-Arithmetik)} \\ \hat{x}^{(k+1)} &= \hat{x}^{(k)} + \hat{c}^{k+1} && \text{(t-Arithmetik)} \end{aligned}$$

Unter bestimmten Voraussetzungen, insbesondere, daß $\hat{L}$, $\hat{R}$ und $\hat{x}$ sinnvoll sind, ist es durch dieses Verfahren möglich, die Lösung mit t Ziffern zu berechnen.

Heuristik V
Die Iterative Verbesserung wird auf einem Rechner mit Basiszahl d benutzt. Sei $\kappa_\infty(A) \doteq d^q$. Nach k Iterationen ist zu erwarten, daß $\hat{x}^{(k)}$

$$\min\{t, k(t-q)\}$$

exakte d-Ziffern hat.

Ein wichtiger Vorteil der Iterativen Verbesserung ist, daß die Konvergenz von $x^{(k)}$ zuverlässig die Genauigkeit bringt. **Beispiel:**[Iterative Verbesserung]

$$\begin{pmatrix} 11 & 15 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 3 \end{pmatrix}$$

$d = 10$, $t = 2$ mit Abrundung.

LR-Zerlegung

$$\hat{L} = g\ell \begin{pmatrix} 1 & 0 \\ 0,454.. & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0,45 & 1 \end{pmatrix}$$

$$\hat{R} = g\ell \begin{pmatrix} 11 & 15 \\ 0 & 7 - 15 \cdot 0,45 \end{pmatrix} = g\ell \begin{pmatrix} 11 & 15 \\ 0 & 7 - 6,75 \end{pmatrix} = \begin{pmatrix} 11 & 15 \\ 0 & 0,3 \end{pmatrix}$$

$Ly = b$

$$\hat{y} = g\ell \begin{pmatrix} 7 \\ 3 - 7 \cdot 0,45 \end{pmatrix} = g\ell \begin{pmatrix} 7 \\ 3 - 3,15 \end{pmatrix} = \begin{pmatrix} 7 \\ -0,1 \end{pmatrix}$$

$Rx = y$

$$\hat{x}^{(1)} = g\ell \begin{pmatrix} (7 - 15x_2)/11 \\ -0,1/0,3 \end{pmatrix} = g\ell \begin{pmatrix} (7 + 4,95)/11 \\ -0,33 \end{pmatrix} = \begin{pmatrix} 1,0 \\ -0,33 \end{pmatrix}$$

$$r^{(1)} = b - Ax^{(1)} = \begin{pmatrix} 7 - 11 + 4,95 \\ 3 - 5 + 2,31 \end{pmatrix} = \begin{pmatrix} 0,95 \\ 0,31 \end{pmatrix}$$

$$\hat{r}^{(1)} = g\ell(r^{(1)}) = r^{(1)} .$$

Erste Iteration

$$\hat{L}y^{(2)} = \hat{r}^{(1)}$$

$$\hat{y}^{(2)} = g\ell \begin{pmatrix} 0,95 \\ 0,31 - 0,4275 \end{pmatrix} = \begin{pmatrix} 0,95 \\ -0,11 \end{pmatrix}$$

$$\hat{R}c^{(2)} = \hat{y}^{(2)}$$

$$\hat{c}^{(2)} = g\ell \begin{pmatrix} (0,95 - 15c_2)/11 \\ -0,11/0,3 \end{pmatrix} = g\ell \begin{pmatrix} (0,95 - 15c_2)/11 \\ -0,366\ldots \end{pmatrix}$$

$$= g\ell \begin{pmatrix} (0,95+5,4)/11 \\ -0,36 \end{pmatrix} = g\ell \begin{pmatrix} 6,3/11 \\ -0,36 \end{pmatrix}$$

$$= g\ell \begin{pmatrix} 0,5727\ldots \\ -0,36 \end{pmatrix} = \begin{pmatrix} 0,57 \\ -0,36 \end{pmatrix}$$

$$\hat{x}^{(2)} = \hat{x}^{(1)} + \hat{c}^{(1)} = \begin{pmatrix} 1,5 \\ -0,69 \end{pmatrix}$$

$$r^{(2)} = \begin{pmatrix} 7-16,5+10,35 \\ 3-7,5+4,83 \end{pmatrix} = \begin{pmatrix} 0,85 \\ +0,33 \end{pmatrix}$$

Bemerkung: Würde $r^{(2)}$ mit $t=2$ arithmetisch berechnet, ergäbe sich:

$$\hat{r}^{(2)} = \begin{pmatrix} 1 \\ 0,3 \end{pmatrix}$$

Zweite Iteration

$$\hat{L}\hat{y}^{(3)} = \hat{r}^{(2)}$$

$$\hat{y}^{(3)} = g\ell \begin{pmatrix} 0,85 \\ 0,33-0,45\cdot 0,85 \end{pmatrix} = g\ell \begin{pmatrix} 0,85 \\ 0,33-0,3825 \end{pmatrix} = \begin{pmatrix} 0,85 \\ -0,05 \end{pmatrix}$$

$$\hat{U}c^{(3)} = \hat{y}^{(3)}$$

$$\hat{c}^{(3)} = g\ell \begin{pmatrix} (0,85-15c_2)/11 \\ -0,05/0,3 \end{pmatrix} = g\ell \begin{pmatrix} (0,85+2,4)/11 \\ -0,16 \end{pmatrix}$$

$$= g\ell \begin{pmatrix} +3,2/11 \\ -0,16 \end{pmatrix} = g\ell \begin{pmatrix} 0,2909\ldots \\ -0,16 \end{pmatrix} = \begin{pmatrix} 0,29 \\ -0,16 \end{pmatrix}$$

$$\hat{x}^{(3)} = \hat{c}^{(3)} + \hat{x}^{(2)} = \begin{pmatrix} 1,7 \\ -0,85 \end{pmatrix}$$

$$r^{(3)} = \begin{pmatrix} 7-18,7+12,75 \\ 3-8,5+5,95 \end{pmatrix} = \begin{pmatrix} 1,05 \\ 0,45 \end{pmatrix}$$

Die Approximationen $x^{(k)}$ werden stets besser, aber die Residua $r^{(k)}$ zeigen es nicht.

Daß die Iterative Verbesserung auch gefährlich sein kann, zeigt **Beispiel:**[Kahan]

$$A = \begin{pmatrix} 0,8647 & 0,5766 \\ 0,4322 & 0,2822 \end{pmatrix} \quad , \quad b = \begin{pmatrix} 0,2885 \\ 0,1445 \end{pmatrix}$$

$d = 10$, $t = 4$, Rundung mit einer Kontrollziffer.

$$\hat{L} = \begin{pmatrix} 1 & 0 \\ 0,4998 & 1 \end{pmatrix}, \quad \hat{U} = \begin{pmatrix} 0,8647 & 0,5766 \\ 0 & 0,1\cdot 10^{-3} \end{pmatrix}$$

$$\hat{L}y = b \Rightarrow \hat{y} = \begin{pmatrix} 0,2885 \\ 0,1\cdot 10^{-3} \end{pmatrix}$$

$$\hat{U}x^{(1)} = \hat{y} \Rightarrow \hat{x} = \begin{pmatrix} -0,3331 \\ 1,000 \end{pmatrix}$$

$$r^{(1)} = b - A\hat{x}^{(1)} = \begin{pmatrix} -0,6843 \\ -0,3418 \end{pmatrix} \cdot 10^{-4}$$

$$\hat{L}y^{(2)} = \hat{r}^{(1)}$$

$$\hat{y}^{(2)} = \begin{pmatrix} -0,6843 & \cdot & 10^{-4} \\ +0,2 & \cdot & 10^{-7} \end{pmatrix}$$

$$\hat{U}c^{(2)} = \hat{y}^{(2)}$$

$$\hat{c}^{(2)} = \begin{pmatrix} -0,2124 & 10^{-3} \\ +0,2 & 10^{-3} \end{pmatrix}$$

$$x^{(2)} = \hat{x}^{(1)} + \hat{c}^{(2)} = \begin{pmatrix} -0,3333124 \\ +1,0002000 \end{pmatrix}$$

$$r^{(2)} = b - Ax^{(2)} = \begin{pmatrix} -0,00000008772 \\ +0,00000002072 \end{pmatrix}$$

Aber

$$x = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

Wie Forsythe sagte: „Anyone unlucky enough to encounter this sort of calamity has probably already been run over by a truck.”

6.7 Dünnbesetzte Matrizen

Eine Matrix A heißt *dünnbesetzt*, wenn nur „wenige” Elemente a_{ij} von A nicht null sind. Es gibt eine große Literatur über die Anwendung des Gaußschen Eliminationsverfahrens auf solche Matrizen.

Literatur

Duff, I.: Sparse Matrices.

6.8 Parallelrechner

Es gibt eine große Literatur über die Implementation des Gaußschen Eliminationsverfahrens auf Parallelrechnern, Vektorrechnern usw.

Literatur

Bunse, W., **Bundes-Gerstner**, A.: Numerische lineare Algebra. Teubner, 1985.

de Boor, C.. **Pinkus**, A.: A backward error analysis for totally positive linear systems. Numer. Math. 27(1977), 485-490.

Goldstine, H.H., **von Neumann**, J.: Numerical inverting of matrices of high order. Bull. Amer. math. Soc. 53(1947), 1021-1099.

Golub, G.H., **van Loan**, C.F.: Matrix Computations.

Gregory, R.T., **Karney**, D.L.: A Collection of Matrices for Testing Computational Algorithms.

Kahan, W.: Numerical Linear Algebra. Canadian Math. Bulletin 9, 756-801(1966).

Wilkinson: Error analysis of direct methods of matrix inversion. J. Assoc. Comp. Mach. 8(1961), 281-330.

Kapitel 7

Interpolation und Approximation

7.1 Einleitung

Das *Approximationsproblem* ist folgendes: Gegeben sei ein Raum X, ein Punkt $f \in X$ und eine Teilmenge $V \subset X$. Gesucht ist $\tilde{v} \in V$, damit $\tilde{v}$ eine „gute" Approximation zu f ist, wobei „gut" natürlich genauer definiert sein muß. (In den praktischen Anwendungen ist die Definition von „gut" oft nicht leicht, da mehrere Faktoren betrachtet werden müssen.)

Wenn X ein normierter Raum mit Norm $\|\cdot\|$ ist, dann existiert

$$\rho(f, V) = \inf_{v \in V} \|f - v\| \ .$$

Falls es $v^* \in V$ gibt mit

$$\|v^* - f\| = \rho(f, V) \ ,$$

heißt v^* eine *beste* oder *optimale* Approximation zu f, und $\tilde{v} \in V$ heißt eine *gute* Approximation, wenn $\|f - \tilde{v}\|$ „nicht viel größer" als $\|f - v^*\|$ ist.

Sei $V \subset X$ ein Untervektorraum von X mit $\dim V = n$. Seien $\lambda_1, \dots, \lambda_n$ lineare Funktionale, $\lambda_i : X \longrightarrow \mathbb{R}$. Um eine Approximation zu f zu finden, liegt es nahe, $\tilde{v} \in V$ zu finden mit

$$\lambda_i(\tilde{v}) = \lambda_i(f) \ , \ 1 \le i \le n \ .$$

Ist X ein *Funktionenraum*, so daß jedes $x \in X$ eine Abbildung

$$x : A \longrightarrow B$$

ist, dann liegt es nahe,

$$\lambda_i(x) := x(a_i) \ , \ a_i \in A$$

zu setzen.

Das Problem läßt sich wie in Abbildung 7.1 veranschaulichen, wobei zu beachten ist, daß einige intuitive Annahmen — z.B. daß die Norm $\|\cdot\|$ die euklidische Norm ist — zu vermeiden sind.

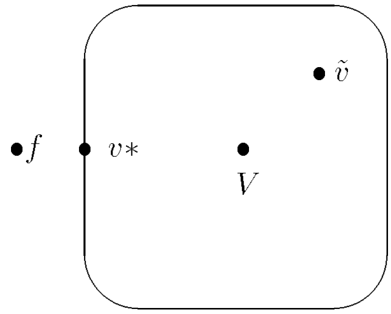


Abbildung 7.1: Das Approximationsproblem

7.1.1 Fragen zu Approximationsaufgaben

Die folgenden Fragen entstehen für gegebene f, V und $\|\cdot\|$:

1. **Existenz:** Gibt es eine beste Approximation v^* ?
2. **Eindeutigkeit:** Wenn v^* existiert, ist v^* eindeutig?
3. Wie läßt sich v^* berechnen?
4. Ist es möglich, $\|v^* - f\|$ auf Grund bekannter Eigenschaften von f und V abzuschätzen?
5. Falls die Berechnung von v^* sehr aufwendig ist, ist es möglich, eine gute Approximation $\tilde{v}$ mit wesentlich weniger Rechenaufwand zu berechnen?
6. n sei gegeben. Mit welchem Unterraum $V \subset X$ von Dimension n ist es möglich, f am besten zu approximieren?

Es ist hier nur möglich, diese Fragen in einfachen, aber wichtigen Fällen zu betrachten.

7.1.2 Anwendungen von Approximationen

1. Auf einer Rechenanlage ist es nur möglich, rationale Funktionen auszuwerten. Die elementaren Funktionen, wie $\cos$, $\sin$, e^x, werden durch Approximationen ersetzt. Z. B.:
$$\frac{1}{2}e^x \doteq \tilde{v}(x) := \frac{1}{2} + x \cdot P(z)/[Q(z) - x \cdot P(z)]\ ,$$
wobei
$$\begin{aligned} z &= x^2 \\ P(z) &= p_o + p_1 z + p_2 z^2 \\ Q(z) &= q_o + q_1 z + q_2 z^2 \end{aligned}$$
$$\begin{aligned} p_0 &= 0.24999\ 99999\ 99999\ 993\ E+0 \\ p_1 &= 0.69436\ 00015\ 11792\ 852\ E-2 \\ p_2 &= 0.16520\ 33002\ 68270\ 130\ E-4 \\ q_0 &= 0.50000\ 00000\ 00000\ 000\ E+0 \\ q_1 &= 0.55553\ 86669\ 69001\ 188\ E-1 \\ q_2 &= 0.49586\ 28849\ 05441\ 294\ E-3 \end{aligned}$$
2. Interpolation in Tabellen.
3. Numerische Integration.
4. Numerische Lösung von gewöhnlichen und partiellen Differentialgleichungen.
5. Computergrafik.

Literatur

Cheney, E.W.: Introduction to Approximation Theory. New York: McGraw-Hill, 1966.

Davis, P.J.: Interpolation and Approximation. New York: Blaisdell, 1963.

Natanson, I.P.: Konstruktive Funktionentheorie. Berlin: 1955.

7.2 Eindimensionale Interpolation

Seien $I \subset \mathbb{R}^1$ ein Intervall und $X = C(I)$. Sei
$$\Phi(\cdot; a_0, \ldots, a_n) \in X\ ,$$

wobei $a_o, a_1, \ldots, a_n$ Parameter sind. Gegeben seien $n+1$ **Stützpunkte** $z_j \in I$. Dann wird $\tilde{v} \in X$ gesucht, $\tilde{v} := \Phi(\cdot; \tilde{a}_o, \ldots, \tilde{a}_n)$, so daß

$$f(z_j) = \tilde{v}(z_j) := \Phi(z_j; \tilde{a}_o, \ldots, \tilde{a}_n)\ ,\ 0 \leq j \leq n\ .$$

Man sagt, $\tilde{v}$ interpoliert f in den Punkten z_j.

Beispiel:

1. Interpolation durch Polynome

$$\begin{aligned} \Phi(z; a_o, \ldots, a_n) &:= \sum_{j=o}^{n} a_j \Phi_j(z) \\ \Phi_j(z) &:= z^j\ , \end{aligned}$$

2. Interpolation durch trigonometrische Interpolation

$$\begin{aligned} \Phi(z; a_o, \ldots, a_n) &:= \sum_{j=o}^{n} a_j \Phi_j(z) \\ \Phi_j(z) &:= e^{ijz} = \cos\ jz + i\ \sin\ jz\ ,\ i = \sqrt{-1}\ , \end{aligned}$$

3. Interpolation durch rationale Funktionen

$$\Phi(z; a_o, \ldots, a_n\ ,\ b_o, \ldots, b_m) = \frac{a_o + a_1 z + \ldots + a_n z^n}{b_o + b_1 z + \ldots + b_m z^m}\ ,$$

4. Interpolation durch Exponentialsummen

$$\Phi(z; a_o, \ldots, a_n\ ;\ \lambda_o, \ldots, \lambda_n) = \sum_{j=o}^{n} a_j e^{\lambda_j z}$$

Interpolation stellt ein verhältnismäßig leichtes Problem dar, da es nur nötig ist, ein System linearer oder nichtlinearer Gleichungen zu lösen. Inwieweit die interpolierende Funktion $\tilde{v}$ von der optimalen Approximation v^* abweicht, werden wir später untersuchen.

7.3 Polynominterpolation

Definition 7.1 *$\mathcal{P}_n$ bezeichne die Menge aller reellen Polynome vom Grad $\leq n$:*

$$\mathcal{P}_n := \left\{ \sum_{k=o}^{n} a_k x^k\ ,\ a_k \in \mathbb{R} \right\}$$

Als *Polynominterpolation* bezeichnet man die folgende Aufgabe:
Gegeben: $x_o, \ldots, x_n \quad , \quad y_o, \ldots, y_n \in \mathbb{R}$
Gesucht: $P \in \mathcal{P}_n$ mit

$$P(x_j) = y_j \quad , \quad j = 0, \ldots, n$$

Die Werte y_j könnten diskrete Daten sein (z.B. $y_j =$ Durchschnittstemperatur in Münster im Jahre j) oder Auswertungen eines $f \in C[a,b]$ in den Punkten x_j.

Siehe Abbildung 7.2.

Satz 7.1 *Das interpolierende Polynom $P \in P_n$ ist eindeutig bestimmt, falls die n Stützpunkte x_j paarweise verschieden sind.*

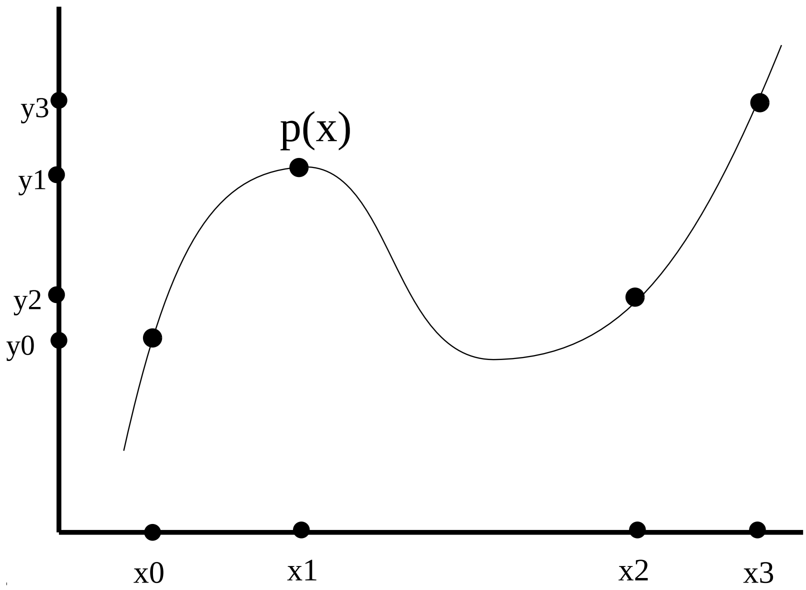


Abbildung 7.2: Interpolation $(n = 3)$

Beweis: Mit

$$P(x_j) = \sum_{k=o}^{n} a_k x_j^k = y_j \qquad , \quad j = 0, \ldots, n$$

erhalten wir ein lineares Gleichungssystem für die Koeffizienten $a_o, \ldots, a_n$ von P. Dieses besitzt genau dann eine eindeutig bestimmte Lösung, wenn das homogene System

$$P(x_j) = \sum_{k=o}^{n} a_k x_j^k = 0 \qquad , \quad j = 0, \ldots, n$$

nur trivial lösbar ist. Dies ist aber offensichtlich der Fall, da ein Polynom vom Grade $\leq n$ nur dann $n+1$ verschiedene Nullstellen haben kann, wenn es identisch verschwindet. $\square$

Bemerkung: Um das gesuchte Polynom zu berechnen, könnte man nun das Gleichungssystem lösen, beispielsweise mit der Cramer'schen Regel. Dies führt auf die *Vandermond'sche Determinante*:

$$V(x_o,\dots,x_n) := \begin{vmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ & & \cdots & \\ 1 & x_n & \cdots & x_n^n \end{vmatrix} = \prod_{k=o}^{n} \prod_{j>k} (x_j - x_k)$$

Für die Koeffizienten a_k gilt dann:

$$v_k := \begin{vmatrix} & & & k-teSpalte & & \\ & & & \downarrow & & \\ 1 & x_0 & \cdots & y_o & \cdots & x_0^n \\ 1 & x_1 & & y_1 & & x^n \\ & & \cdots & & \cdots & \\ 1 & x_n & & y_n & & x_n^n \end{vmatrix}$$

$$a_k = \frac{v_k}{V(x_o,\dots,x_n)}$$

Zur praktischen Berechnung von P stellen wir drei weniger aufwendige Methoden vor:

7.3.1 Die Interpolationsformel von Lagrange

Man setzt

$$\omega_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Es gilt dann:

1. $\omega_i \in \mathcal{P}_n$
2. $\omega_i(x_k) = \delta_{ik} = \begin{cases} 1, & k = i \\ 0, & \text{sonst} \end{cases}$ (Kroneckersymbol)

Damit ist das gesuchte Polynom:

$$P(x) = \sum_{j=o}^{n} y_j \; \omega_j(x)$$

Beispiel:[n = 2]

vorgegebene Werte:

j	x_j	y_j
0	0	1
1	1	3
2	3	2

$$\begin{aligned}
\omega_o(x) &= \frac{(x-x_1)(x-x_2)}{(x_o-x_1)(x_o-x_2)} = \frac{1}{3}\,(x-1)(x-3) \\
\omega_1(x) &= \frac{(x-x_o)(x-x_2)}{(x_1-x_o)(x_1-x_2)} = -\frac{1}{2}\,x(x-3) \\
\omega_2(x) &= \frac{(x-x_o)(x-x_1)}{(x_2-x_o)(x_2-x_1)} = \frac{1}{6}\,x(x-1)
\end{aligned}$$

$$\begin{aligned}
P(x) &= 1\cdot\omega_1(x) + 3\cdot\omega_2(x) + 2\cdot\omega_3(x) \\
&= \frac{1}{3}\,(x-1)(x-3) - \frac{3}{2}\,x(x-3) + \frac{1}{3}\,x(x-1) \\
&= -\frac{5}{6}\,x^2 + \frac{17}{6}\,x + 1
\end{aligned}$$

Die Interpolationsformel von Lagrange ist theoretisch sehr wichtig und wird bei der Herleitung von Integrationsformeln usw. benutzt. Zur Berechnung eines konkreten Interpolationspolynoms ist die Formel von Lagrange ineffizient.

In der Praxis möchte man häufig ein vorhandenes Interpolationspolynom durch Hinzunahme weiterer Stützstellen verbessern. Die Lagrange-Methode ist hierfür nicht geeignet, da man bei neu hinzukommenden Stützstellen mit der Rechnung von vorn beginnen muß. Dies ist bei den zwei folgenden Methoden nicht der Fall.

7.3.2 Die Rekursionsformel von Neville

Bezeichnung: Bei gegebenen $x_o, \ldots, x_n$, $y_o, \ldots, y_n$ bezeichnet $P_{i,i+1,\ldots,k}$ dasjenige Polynom aus $\mathcal{P}_{k-i}$, das an den Stellen $x_i, x_{i+1}, \ldots, x_k$ die Werte $y_i, y_{i+1}, \ldots, y_k$ annimmt. Für paarweise verschiedene $x_i, x_{i+1}, \ldots, x_k$ ist $P_{i,\ldots,k}$ nach Satz 7.1 eindeutig bestimmt.

Satz 7.2 *Seien $x_i, x_{i+1}, \ldots, x_k$ paarweise verschieden. Dann gilt:*

$$P_{i,\ldots,k}(x) = \frac{1}{x_k - x_i}\,\left((x-x_i)P_{i+1,\ldots,k}(x) + (x_k - x)P_{i,\ldots,k-1}(x)\right)\ . \tag{7.1}$$

Beweis: Auf der rechten Seite steht ein Polynom vom Grad $\leq k-i$, das an den Stellen $x_i, \ldots, x_k$ die Werte $y_i, \ldots, y_k$ annimmt. □

(7.1) erlaubt die rekursive Berechnung des Interpolationspolynoms nach dem folgenden Schema ($n = 3$):

$$\begin{array}{lllllll}
x_o & y_o = P_0 & & & & & \\
& & \rangle\ P_{0,1} & & & & \\
x_1 & y_1 = P_1 & & \rangle\ P_{0,1,2} & & & \\
& & \rangle\ P_{1,2} & & \rangle\ P_{0,1,2,3} & & \\
x_2 & y_2 = P_2 & & \rangle\ P_{1,2,3} & & & \\
& & \rangle\ P_{2,3} & & & & \\
x_3 & y_3 = P_3 & & & & &
\end{array}$$

Bei Hinzunahme von weiteren Stützstellen wird das Schema einfach erweitert, ohne daß die bereits berechneten Ergebnisse ungültig werden.

Beispiel: Neville: Berechne $P_{012}(x)$

$$\begin{array}{ccccccccc}
\mathbf{x-x_j} & \mathbf{j} & \mathbf{x_j} & \mathbf{f(x_j)} & & & & & \\
x-1 & 0 & 1 & P_0 = 3 & & & & & \\
 & & & & P_{0,1} = & \frac{(x-1)2+(3-x)3}{3-1} & = & \frac{7-x}{2} & \\
x-3 & 1 & 3 & P_1 = 2 & & & & & P_{012} \\
 & & & & P_{1,2} = & \frac{(x-3)1+(0-x)2}{0-3} & = & \frac{x+3}{3} & \\
x-0 & 2 & 0 & P_2 = 1 & & & & &
\end{array}$$

wobei

$$P_{012} = \frac{(x-1)\cdot\frac{x+3}{3} + (0-x)\cdot\frac{7-x}{2}}{0-1} = \frac{-5x^2+17x+6}{6}$$

Die Formel von Neville ist sehr nützlich, wenn das interpolierende Polynom an einer Stelle x ausgewertet werden sollte.

Beispiel: Neville: Berechne $P(2)$: Sei $\overline{x} = 2$.

$$\begin{array}{cccccccc}
\mathbf{\overline{x}-x} & \mathbf{x} & \mathbf{f(x)} & & & & & \\
+1 & 1 & 3 & & & & & \\
-1 & 3 & 2 & \frac{(2-1)2+(3-2)3}{3-1} & = & \frac{5}{2} & & \\
2 & 0 & 1 & \frac{(2-3)1+(0-2)2}{0-3} & = & \frac{5}{3} & \frac{(2-1)5/3+(0-2)5/2}{0-1} & = \frac{10}{3}
\end{array}$$

Die Diagonalfolge $3, 5/2, 10/3$ gibt die Approximation zu $f(\overline{x})$ an, bei einer zunehmenden Anzahl von Stützpunkten. Wird der Wert von f bis zu einer bestimmten Genauigkeit gesucht, dann werden weitere Stützpunkte soweit hinzugefügt, bis die berechneten Werte mit der gewünschten Genauigkeit übereinstimmen.

7.3.3 Die Newtonsche Form und dividierte Differenzen

Die Nevill'sche Rekursion eignet sich mehr zur Auswertung des Interpolationspolynoms an einer bestimmten Stelle als zur Berechnung seiner Koeffizienten. Hierzu benutzt man die folgende Methode:

Für das Interpolationspolynom $P = P_{o,\dots,n}$ machen wir den Ansatz

$$\begin{aligned}
P_{o,\dots,n}(x) &= A_o + A_1(x-x_o) + A_2(x-x_o)(x-x_1) + \dots \\
&+ A_n(x-x_o)\cdots(x-x_{n-1})
\end{aligned}$$

Dies führt auf ein einfaches Gleichungssystem für die A_i :

$$\begin{array}{lcl} P_{o,\dots,n}(x_o) = A_o & = & y_o \\ P_{o,\dots,n}(x_1) = A_o + A_1(x_1 - x_o) & = & y_1 \\ P_{o,\dots,n}(x_2) = A_o + A_1(x_2 - x_o) + A_2(x_2 - x_o)(x_2 - x_1) & = & y_2 \\ \vdots & & \\ P_{o,\dots,n}(x_n) = A_o + \ldots + A_n(x_n - x_o)\cdots(x_n - x_{n-1}) & = & y_n \end{array}$$

Die A_i können hieraus durch Vorwärtseinsetzen bestimmt werden, wir wollen jedoch explizite Formeln herleiten. Hierzu definieren wir rekursiv die *Dividierten Differenzen* $[y_i \ldots y_k]$:

$$[y_i] := y_i$$

$$[y_i \ldots y_k] := \frac{1}{x_k - x_i} \; ([y_{i+1} \ldots y_k] - [y_i \ldots y_{k-1}])$$

Beispiel:

$$[y_o y_1] = \frac{y_1 - y_o}{x_1 - x_o}$$

Zur bequemen Berechnung der Dividierten Differenzen dient das Differenzenschema:

$$\begin{array}{cccccc} x_o & [y_o] & & & \\ & & [y_o y_1] & & \\ x_1 & [y_1] & & [y_o y_1 y_2] & \\ & & [y_1 y_2] & & [y_o y_1 y_2 y_3] \\ x_2 & [y_2] & & [y_1 y_2 y_3] & \\ & & [y_2 y_3] & & \\ x_3 & [y_3] & & & \end{array}$$

Beispiel: Für das bei der Lagrange-Form benutzte Beispiel lautet das Differenzenschema:

$$\begin{array}{cccc} x_j & y_j & & \\ 0 & 1 & & \\ & & 2 & \\ 1 & 3 & & -5/6 \\ & & -1/2 & \\ 3 & 2 & & \end{array}$$

Satz 7.3 *Für die Koeffizienten* $A_o, \ldots, A_n$ *der Newton'schen Form des Interpolationspolynoms gilt:*

$$A_i = [y_o \ldots y_i] \; , \; i = 0, \ldots, n$$

Beweis: Wir zeigen:

$$\begin{aligned}
P_{i,\dots,k}(x) &= [y_i] + [y_i y_{i+1}](x-x_i) + [y_i y_{i+1} y_{i+2}](x-x_i)(x-x_{i+1}) + \dots \\
&+ [y_i \dots y_k](x-x_i)\cdot\ldots\cdot(x-x_{k-1})
\end{aligned}$$

durch Induktion nach $m = k - i$.

1. Sei $m = 0$, dann gilt: $P_i(x) = [y_i] = y_i$. Die Induktionsaussage $A(0)$ ist somit richtig.

2. Die Behauptung sei richtig für ein $m \geq 0$:

$$\begin{aligned}
P_{i,\dots,k}(x) &= [y_i \dots y_k]x^{k-i} + Q_1 \\
P_{i+1,\dots,k+1}(x) &= [y_{i+1} \dots y_{k+1}]x^{k-i} + Q_2
\end{aligned}$$

mit $Q_1, Q_2 \in \mathcal{P}_{k-i-1}$. Aus Satz 7.2 folgt

$$\begin{aligned}
P_{i,\dots,k+1}(x) &= \frac{1}{x_{k+1}-x_i}\,\left((x-x_i)P_{i+1,\dots,k+1}(x) + (x_{k+1}-x)P_{i,\dots,k}(x)\right) \\
&= \frac{1}{x_{k+1}-x_i}\,\left((x-x_i)[y_{i+1}\dots y_{k+1}] + (x_{k+1}-x)[y_i \dots y_k]\right)x^{k-i} + R_1 \\
&= \frac{1}{x_{k+1}-x_i}\,\left([y_{i+1}\dots y_{k+1}] - [y_i \dots y_k]\right)x^{k-i+1} + R_2 \\
&= [y_i \dots y_{k+1}]x^{k-i+1} + R_2 \\
&= [y_i \dots y_{k+1}](x-x_i)(x-x_{i+1})\dots(x-x_k) + R_3 \qquad (7.2)
\end{aligned}$$

mit $R_1, R_2, R_3 \in \mathcal{P}_{k-i}$. Für R_3 muß gelten:

$$R_3(x_j) = P_{i,\dots,k+1}(x_j) = y_j \qquad \text{für} \qquad j = i, \dots, k\ ,$$

so daß

$$R_3 = P_{i,\dots,k}. \qquad (7.3)$$

Die Richtigkeit von $A(m+1)$ folgt aus (7.2), (7.3), und $A(m)$.

□

Beispiel: Das Newton'sche Interpolationspolynom für das obige Beispiel kann direkt der obersten Diagonale des Differenzenschemas entnommen werden:

$$P(x) = 1 + 2x - 5/6x(x-1)\ .$$

Will man Stützpunkte hinzunehmen, läßt sich das Differenzenschema leicht erweitern. Zu den Stützpunkten unseres Beispiels nehmen wir noch $x_3 = 2$, $y_3 = 4$ hinzu und erhalten:

$$
\begin{array}{llllll}
x_j & y_j & & & & \\
0 & 1 & & & & \\
 & & 2 & & & \\
1 & 3 & & -5/6 & & \\
 & & -1/2 & & -1/3 & \\
3 & 2 & & -3/2 & & \\
 & & -2 & & & \\
2 & 4 & & & &
\end{array}
$$

$$P(x) = 1 + 2x - \frac{5}{6}\, x(x-1) - \frac{1}{3}\, x(x-1)(x-3)$$

Satz 7.4 *Sei $i_o, \ldots, i_n$ eine beliebige Permutation von $0, \ldots, n$. Dann gilt:*

$$[y_{i_o} \ldots y_{i_n}] = [y_o \ldots y_n]\ .$$

Beweis: Wir stellen das eindeutig bestimmte Interpolationspolynom in der Newton'schen Form mit einer anderen Numerierung der Stützstellen nochmals auf:

$$P(x) = [y_o \ldots y_n]x^n + Q = [y_{i_o} \ldots y_{i_n}]x^n + \tilde{Q}\ ;\ Q, \tilde{Q} \in \mathcal{P}_{n-1}$$

und daraus folgt $[y_o \ldots y_n] = [y_{i_o} \ldots y_{i_n}]$. □

7.4 Der Interpolationsfehler bei Polynominterpolation

Seien in einem Intervall $[a, b]$ $n+1$ paarweise verschiedene Stützstellen $x_o, \ldots, x_n$ und eine Funktion $f \in C^{n+1}[a, b]$ gegeben. Wir wollen $f(x)$ für $x \neq x_i$ approximieren.

Betrachten wir das eindeutig bestimmte Polynom $p \in \mathcal{P}_n$ mit $p(x_j) = f(x_j)\ ,\ j = 0, \ldots, n$. Der folgende Satz macht eine Aussage über den Fehler, der bei einer Approximation von f durch p auftritt:

Satz 7.5 *Zu jedem $\overline{x} \in [a, b]$ existiert ein $\tilde{x} \in [a, b]$, so daß gilt:*

$$f(\overline{x}) - p(\overline{x}) = w(\overline{x}) \frac{f^{(n+1)}(\tilde{x})}{(n+1)!}$$

mit

$$w(x) := \prod_{j=0}^{n} (x - x_j)$$

Beweis: Sei $\overline{x} \in [a,b]$ beliebig gewählt mit $\overline{x} \neq x_j$, $j = 0, \ldots, n$. Wir setzen

$$F(x) := f(x) - p(x) - Kw(x)$$

mit einer Konstanten K und erhalten $F(x_j) = 0,\ j = 0, \ldots, n$.

Wir wählen K nun so, daß auch $F(\overline{x})$ verschwindet, also

$$K = \left(\frac{f-p}{w}\right)(\overline{x})$$

Damit hat F in $[a,b]$ mindestens die $n+2$ Nullstellen $\overline{x}, x_o, \ldots, x_n$. Aus dem Satz von Rolle folgt, daß dann $F^{(n+1)}$ mindestens eine Nullstelle $\tilde{x}$ in $[a,b]$ besitzt.

Aus der Beziehung

$$F^{(n+1)}(x) = f^{(n+1)}(x) - K(n+1)!$$

folgt

$$K = \left(\frac{f-p}{w}\right)(\overline{x}) = \frac{f^{(n+1)}(\tilde{x})}{(n+1)!}$$

also:

$$f(\overline{x}) - p(\overline{x}) = w(\overline{x})\frac{f^{(n+1)}(\tilde{x})}{(n+1)!}$$

Diese Beziehung ist offenbar auch für $\overline{x} = x_j$, $j = 0, \ldots, n$ erfüllt. □

Für den Interpolationsfehler erhalten wir die Abschätzung

$$|f(x) - p(x)| \leq |w(x)| \quad \max_{x\in[a,b]} \quad \frac{|f^{(n+1)}(x)|}{(n+1)!}$$

Wir betrachten zuerst den Fall gleichmäßig verteilter Stützstellen:

1. Sei $x_j = a + jh$ mit der Schrittweite $h = \frac{b-a}{n}$. Für $x = a + \theta h$, $0 \leq \theta \leq n$ gilt:

$$w(x) = h^{n+1}\prod_{j=0}^{n}(\theta - j)$$

und

$$|f(x) - p(x)| \leq \frac{h^{n+1}}{(n+1)!}\left(\prod_{j=0}^{n}|\theta - j|\right) \max_{x\in[a,b]} |f^{(n+1)}(x)|.$$

   (a) Für $b - a \to 0$ bei festem n erhalten wir $f(x) - p(x) = 0(h^{n+1})$.

   (b) Der Fall $n \to \infty$ bei fester Intervallänge $b-a$ führt i.a. zu keiner Konvergenz. Wir betrachten hierzu das Beispiel $f(x) = (1+25x^2)^{-1}$ in $[-1,1]$ von Runge (siehe Abbildung 7.3) f wird an den Stellen $x_j = -1 + \frac{2j}{n}$, $j = 0, \ldots, n$ durch ein Polynom vom Grad n interpoliert. Die folgende Tabelle zeigt, daß der Interpolationsfehler für große n stark anwächst.

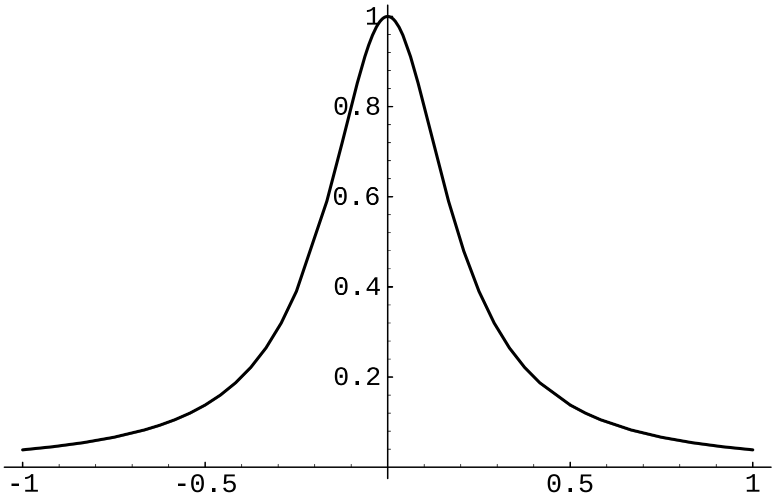


Abbildung 7.3: Runges Funktion

n	$\max_{x\in[-1,1]} \lvert f(x)-p(x)\rvert$
1	0,96
5	0,43
13	1,07
19	8,57

2. Wir wählen die Stützstellen $x_o, \ldots, x_n$ nun so, daß $\max_{[a,b]} |w(x)|$ möglichst klein ist.

   Für $[a,b] = [-1,1]$ erhalten wir

   $$w(x) = 2^{-n} T_{n+1}(x)$$

   wobei für $x \in [-1,1]$ die *Tschebyscheff-Polynome* T_n wie folgt definiert sind:

   $$T_n(x) = \cos nt, \quad x = \cos t, \quad 0 \leq t \leq 2\pi$$

   Es gilt: $T_o(x) = 1$ und $T_1(x) = x$, sowie für $n \geq 1$:

   $$\begin{aligned} T_{n+1}(x) &= \cos(n+1)t = \cos\ n\ t \cdot \cos\ t - \sin\ n\ t \cdot \sin\ t \\ T_{n-1}(x) &= \cos(n-1)t = \cos\ n\ t \cdot \cos\ t + \sin\ n\ t \cdot \sin\ t \end{aligned}$$

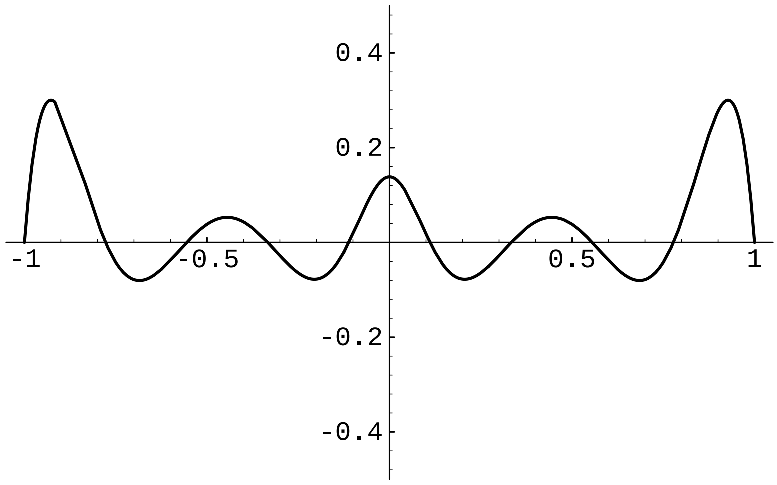


Abbildung 7.4: Der Fehler $f(x) - p(x)$ für $n = 9$

Addition der Gleichungen ergibt:

$$T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x)$$

oder

$$T_{n+1} = 2xT_n - T_{n-1} \ .$$

Da T_o und T_1 Polynome sind, sind es auch alle anderen T_n. Die Rekursion zeigt weiter, daß T_n für $n \geq 1$ die Form

$$T_n(x) = 2^{n-1}x^n + Q_n, \quad Q_n \in \mathcal{P}_{n-1}$$

haben muß. $w(x) = 2^{-n}T_{n+1}(x)$ hat daher den Höchstkoeffizienten 1 und es gilt:

$$|w(x)| \leq 2^{-n} \quad \text{in} \quad [-1, 1] \ .$$

Die Nullstellen von w sind unsere neuen Stützstellen:

$$\begin{aligned} w(x) = 0 \quad &\Leftrightarrow \quad T_{n+1}(x) = 0 \\ &\Leftrightarrow \quad \cos(n+1)t = 0 \\ &\Leftrightarrow \quad t = \frac{(j+\frac{1}{2})\pi}{n+1} \qquad j = 0, \ldots, n \\ &\Rightarrow \quad x_j = \cos\frac{(j+\frac{1}{2})\pi}{n+1} \quad j = 0, \ldots, n \end{aligned}$$

Bei dieser Wahl der Stützstellen ergibt sich die Fehlerabschätzung

$$|f(x) - p(x)| \leq \frac{\max\ |f^{(n+1)}(x)|}{2^n(n+1)!}$$

Für das obige Beispiel ergeben sich folgende Werte:

n	$\max\limits_{[-1,1]} \lvert f(x) - p(x)\rvert$
1	0.93
5	0.56
13	0.12
19	0.04

Die Verbesserung ist erheblich.

In diesem Zusammenhang sind folgende Sätze interessant:

Satz 7.6 *(Weierstrass) Ist $f \in C[a,b]$ und $\epsilon > 0$ beliebig vorgegeben, so existiert ein Polynom $P(x)$ derart, daß*

$$\max_{x\in[a,b]} |f(x) - P(x)| \leq \epsilon\ .$$

Beweis: (Skizze) Sei $[a,b] = [0,1]$ und

$$B_n(x) := \sum_{k=0}^{n} f(\frac{k}{n})\binom{n}{k}\ x^k(1-x)^{n-k}\ .$$

Dann gilt

$$\max_{x\in[0,1]} |B_n(x) - f(x)| \longrightarrow 0 \quad \text{für} \quad n \longrightarrow \infty\ .$$

$B_n(x)$ heißt das *Bernstein-Polynom* der Funktion f. Siehe Natanson oder Forster I, S. 201, Aufgabe 23.5. □

Satz 7.7 (Faber) *Sei*

$$\begin{array}{llll} x_{0,0} & & & \\ x_{1,0} & x_{1,1} & & \\ \cdots & \cdots & & \\ x_{n,0} & x_{n,1} & \cdots & x_{n,n} \\ \cdots & \cdots & & \end{array}$$

eine vorgegebene Menge von Stützpunkten $x_{i,j} \in [a,b]$. Sei $L_n(f;x) \in P_n$ dasjenige Polynom, das die Funktion f in den Stützpunkten $x_{n,j}$, $0 \leq j \leq n$ interpoliert. Dann gibt es $f \in C[a,b]$ mit

$$\liminf_{n\longrightarrow\infty}\ \max_{x\in[a,b]} |f(x) - L_n(f;x)| > 0\ .$$

Beweis: Siehe Korovkin 1960, S. 195. □

7.5 Spline–Interpolation: Einführung

Ein *Spline* ist ein Zeichengerät, das früher beim Schiffsbau eingesetzt wurde. Es besteht aus einer flexiblen langen Latte, die auf einem Zeichenbrett aufgelegt wurde, um glatte Kurven zu erzeugen. (Ein Kurvenlineal wird in ähnlicher Weise benutzt.)

Bezeichnend für die Kurven, die mit Splines erzeugt werden, sind:

1. Es sind stückweise Polynome, d. h. *Polynomzüge* und
2. es gibt *Bruchstellen* (engl.: break points).

7.6 Lineare Splines

Definition 7.2 *Im Intervall* $[a, b] \subset \mathbb{R}$ *seien die Bruchstellen* τ_i , $1 \leq i \leq n$,

$$a = \tau_1 < \tau_2 < \cdots < \tau_n = b$$

vorgegeben. Sei $\Delta = (\tau_1, \ldots, \tau_n)$. *Weiter sei* $\pi_1(\tau_i, \tau_{i+1})$ *die Menge der Restriktionen von Polynomen vom Grad 1 auf das Intervall* (τ_i, τ_{i+1}). s_Δ *heißt* lineare Spline-Funktion *(oder kurz* Linearer Spline*) bzgl.* Δ, *wenn*

1. $s_\Delta : [a, b] \to \mathbb{R}$
2. $s_\Delta|_{(\tau_i,\tau_{i+1})} \in \pi_1(\tau_i, \tau_{i+1})$
3. $s_\Delta \in C[a, b]$.

Die Menge aller linearen Splines wird mit $S_2(\Delta)$ *bezeichnet.*

Sei $f \in C[a, b]$ und Δ wie in Definition 7.17. Es gibt genau ein $s \in S_2(\Delta)$, das f in den Stützpunkten τ_i interpoliert,

$$s(\tau_i) = f(\tau_i) \ , \ 1 \leq i \leq n$$

und zwar ist $s = I_2 f$ mit

$$(I_2 f)(x) := f(\tau_i) + (x - \tau_i)[\tau_i, \tau_{i+1}]f \ , \ x \in [\tau_i, \tau_{i+1}] \ .$$

Ist $f \in C^{(2)}[a, b]$, dann läßt sich der Fehler

$$e(x) := f(x) - (I_2 f)(x)$$

leicht abschätzen:

$$\|e\| \leq \frac{|\Delta|^2}{8} \, \|f^{(2)}\|$$

mit

$$\begin{aligned} |\Delta| &= \max_i |\tau_{i+1} - \tau_i| \\ \|g\| &= \|g\|_\infty = \max_{a \le x \le b} |g(x)| \end{aligned}$$

Satz 7.8

$$\|f - I_2 f\|_\infty \le 2\rho_\infty(f, S_2(\Delta)) ,$$

mit

$$\rho_\infty(f, S_2(\Delta)) := \inf_{s \in S_2(\Delta)} \|f - s\|_\infty .$$

7.7 Kubische Splines: Einführung

Definition 7.3 *Im Intervall* $[a,b] \subset \mathbb{R}$ *seien die Bruchstellen* τ_i , $1 \le i \le n$,

$$a = \tau_1 < \tau_2 \cdots < \tau_n = b$$

vorgegeben. Sei $\Delta := (\tau_1, \ldots, \tau_n)$. *Weiter sei* $\pi_3(\tau_i, \tau_{i+1})$ *die Menge der Restriktionen von Polynomen vom Grad 3 auf das Intervall* (τ_i, τ_{i+1}). s_Δ *heißt eine* Kubische Spline-Funktion *(oder kurz ein* Kubischer Spline*) bzgl.* Δ, *wenn*

1. $s_\Delta : [a,b] \to \mathbb{R}$
2. $s_\Delta|_{(\tau_i, \tau_{i+1})} \in \pi_3(\tau_i, \tau_{i+1})$
3. $s_\Delta \in C^2[a,b]$.

Mit Hilfe der abgeschnittenen Potenzfunktionen (Stützfunktionen)

$$x_+^m = \begin{cases} 0 & \text{für} \quad x \le 0 \\ x^m & \text{für} \quad x > 0 \end{cases}$$

mit

$$x^0 := 1 \quad \text{für} \quad x \ge 0$$

kann man eine Basis des linearen Raumes bilden:

$$S_4(\Delta) := \{ s_\Delta : s_\Delta \quad \text{ist eine Kubische Spline-Funktion bzgl.} \quad \Delta \} ,$$

Satz 7.9 *Jedes* $s_\Delta \in S_4(\Delta)$ *ist eindeutig darstellbar als*

$$s_\Delta(x) = \sum_{j=o}^{2} b_j (x - \tau_1)_+^j + \sum_{i=1}^{n-1} c_i (x - \tau_i)_+^3 \qquad (7.4)$$

mit $b_j, c_i \in \mathbb{R}$.

Beweis: Siehe z.B. Boehmer, S. 17. □

Bemerkung:

1. Es folgt aus Satz 7.9, daß dim $S_4(\Delta) = n+2$. Es ist deshalb zu erwarten, daß $n+2$-Bedingungen nötig sind, um einen Spline s_Δ zu bestimmen.

2. Obwohl die Darstellung (7.4) theoretisch sehr nützlich ist, ist sie praktisch unbrauchbar.

Bei der Definition von Splines höherer Ordnung werden Eigenschaften von dividierten Differenzen benutzt, die wir im folgenden Abschnitt beweisen.

7.8 Dividierte Differenzen

Sei $f : [a,b] \longrightarrow \mathbb{R}$ und $\tau_1 < \tau_2 < \ldots < \tau_n$. Man setzt

$$[\tau_1, \ldots, \tau_n]f := [f(\tau_1, \ldots, f(\tau_n)]$$

wobei $[y_1, \ldots, y_n]$ wie vorher definiert ist. Man merke (siehe Satz 7.25), daß $[\tau_i, \ldots, \tau_k]f$ der Koeffizient von x^{k-i} des Interpolationspolynoms p ist, wobei:

1. $p \in P_{k-i}$

2. $p(\tau_j) = f(\tau_j), \quad i \leq j \leq k.$

Eigenschaften dividierter Differenzen

Satz 7.10 (Symmetrie) *Die dividierte Differenz $[\tau_i, \ldots, \tau_k]f$ ist unabhängig von der Ordnung der τ_j.*

Satz 7.11 (Rekursionsformel)

$$[\tau_i, \ldots, \tau_k]f = \frac{[\tau_{i+1}, \ldots, \tau_k]f - [\tau_i, \ldots, \tau_{k-1}]f}{\tau_k - \tau_i}, \quad \textit{für} \quad \tau_k \neq \tau_i$$

Satz 7.12 (Leibnitzsche Formel) *Sei $f(x) = g(x)h(x)$ für $x \in \mathbb{R}$. Dann gilt:*

$$[\tau_i, \ldots, \tau_{i+k}]f = \sum_{r=i}^{i+k} \left([\tau_i, \ldots, \tau_r]g\right) \cdot \left([\tau_r, \ldots, \tau_{i+k}]h\right)$$

Beweis: Sei

$$\begin{aligned} F(x) &:= \left[\sum_{r=i}^{i+k}(x-\tau_i)\cdots(x-\tau_{r-1})[\tau_i,\ldots,\tau_r]g\right] \\ &\cdot \left[\sum_{s=i}^{i+k}(x-\tau_{s+1})\cdots(x-\tau_{i+k})[\tau_s,\ldots,\tau_{i+k}]h\right] \end{aligned}$$

Dann ist $F(\tau_j) = f(\tau_j)$ für $i \le j \le i+k$.

In dem Produkt tragen Terme mit $r > s$ zum $F(\tau_j)$ nicht bei, da sie den Faktor $\prod_{j=i}^{i+k}(x-\tau_j)$ enthalten. Sei

$$\begin{aligned} \tilde{F}(x) &:= \sum_{s=i}^{i+k}\sum_{r=i}^{s}\prod_{j=i}^{r-1}(x-\tau_j) \\ &\cdot \prod_{\ell=s+1}^{i+k}(x-\tau_\ell)\cdot([\tau_i,\ldots,\tau_r]g)\cdot([\tau_s,\ldots,\tau_{i+k}]h) \end{aligned}$$

Folglich ist $\tilde{F}(\tau_j) = f(\tau_j)$ für $i \le j \le i+k$. $\tilde{F}(x)$ ist ein Polynom vom Grade k, das die Funktion $f(x)$ in den $k+1$ Stützpunkten $\tau_i,\ldots,\tau_{i+k}$ interpoliert. Durch den Vergleich der Koeffizienten von x^k erhält man:

$$[\tau_i,\ldots,\tau_{i+k}]f = \sum_{s=i}^{i+k}[\tau_1,\ldots,\tau_s]g\cdot[\tau_s,\ldots,\tau_{i+k}]h.$$

□

Beispiel:

$$\begin{aligned} [\tau_1\tau_2](gh) &= ([\tau_1]g)([\tau_1\tau_2]h) + ([\tau_1\tau_2]g)([\tau_2]h) \\ &= g(\tau_1)\cdot\frac{h(\tau_2)-h(\tau_1)}{\tau_2-\tau_1} + \frac{g(\tau_2)-g(\tau_1)}{\tau_2-\tau_1}\cdot h(\tau_2) \\ &= \frac{g(\tau_1)\cdot(h(\tau_2)-h(\tau_1)) + (g(\tau_2)-g(\tau_1)\cdot h(\tau_2))}{\tau_2-\tau_1}, \\ &= \frac{(gh)(\tau_2)-(gh)(\tau_1)}{\tau_2-\tau_1} \\ &= [\tau_1,\tau_2]gh \end{aligned}$$

Satz 7.13 (Hermite-Genocchi Formel) *Sei*

$$n \ge 2,\quad f \in C^{(n-2)}[a,b] \quad \textit{und} \quad D^{n-1}\,f \in L_1[a,b].$$

Dann gilt:

$$\begin{aligned} [\tau_1,\ldots,\tau_n]f &= \int D^{n-1}f(\tau_1+t_2\nabla\tau_2+\cdots+t_n\nabla\tau_n)dt \\ &:= \int_0^1 dt_2\int_0^{t_2}dt_3\ldots\int_0^{t_{n-1}}dt_n\; D^{n-1}f(\tau_1+t_2\nabla\cdot\tau_2+\cdots+t_n\nabla\tau_n) \end{aligned}$$

wobei $0 \le t_n \le t_{n-1} \le \ldots \le t_2 \le 1$ *und* ∇ *den Rückwärtsdifferenzenoperator bezeichnet:* $\nabla\tau_j := \tau_j - \tau_{j-1}$.

Beweis: Der Beweis erfolgt durch Induktion nach n. Für $n = 2$ gilt:

$$\int_0^1 Df(\tau_1 + t_2\nabla\tau_2)dt_2 = \left[\frac{f(\tau_1 + t_2\nabla\tau_2)}{\nabla\tau_2}\right]_{t_2=0}^{t_2=1} = \frac{f(\tau_2) - f(\tau_1)}{\tau_2 - \tau_1} = [\tau_1, \tau_2]f,$$

so daß die Behauptung für $n = 2$ richtig ist. Für $n = 3$ gilt dann:

$$\begin{aligned}
& \int_0^1 dt_2 \int_0^{t_3} D^2 f(\tau_1 + t_2\nabla\tau_2 + t_3\nabla\tau_3)dt_3 \\
= \; & \int_0^1 dt_2 \left[\frac{Df(\tau_1 + t_2\nabla\tau_2 + t_3\nabla\tau_3}{\nabla\tau_3}\right]_{t_3=0}^{t_3=t_2} \\
= \; & \int_0^1 dt_2 \frac{[Df(\tau_1 + t_2 \cdot (\tau_3 - \tau_1)) - Df(\tau_1 + t_2 \cdot (\tau_2 - \tau 1))]}{\tau_3 - \tau_2} \\
= \; & \frac{[\tau_1, \tau_3]f - [\tau_1, \tau_2]f}{\tau_3 - \tau_2} = [\tau_1, \tau_2, \tau_3]f,
\end{aligned}$$

wobei im letzten Schritt der Satz mit $n = 2$ angewandt worden ist. Der Beweis für ein allgemeines n folgt auf ähnliche Weise. □

Satz 7.14 (Stetigkeit der dividierten Differenzen bezüglich $\tau_1, \ldots, \tau_n$**)** *Für* $f \in C^{n-1}$ *gilt*

$$\lim_{\tau_1,\ldots,\tau_n \to \tau_0} [\tau_1, \ldots, \tau_n]f = \frac{1}{(n-1)!} D^{n-1} f(\tau_0).$$

Beweis: Aus der Hermite-Genocchi Formel ersieht man sofort, daß $[\tau_1, \ldots, \tau_n]f$ einen eindeutigen Grenzwert besitzt, wenn einige der τ_i zusammenfallen, vorausgesetzt, daß f genügend glatt ist, d. h.: $f \in C^{n-1}$. Insbesondere gilt:

$$\begin{aligned}
& \lim_{\tau_1,\ldots,\tau_n \to \tau_0} [\tau_1, \ldots, \tau_n]f \\
= \; & \lim_{\tau_1,\ldots,\tau_n \to \tau_0} \int_0^1 dt_2 \ldots \int_0^{t_{n-1}} dt_n D^{n-1} f(\tau_1 + t_2\nabla\tau_1 + \cdots + t_n\nabla\tau_{n-1}) \\
= \; & D^{n-1} f(\tau_0) \cdot \int_0^1 dt_2 \cdots \int_0^{t_{n-1}} dt_n \\
= \; & \frac{1}{(n-1)!} D^{n-1} f(\tau_0),
\end{aligned}$$

falls $D^{n-1}f(x)$ in einer Umgebung von $x = \tau_0$ stetig ist. □

Die dividierte Differenz $[\tau_1, \ldots, \tau_n]f$ wurde bisher nur für paarweise ungleiche τ_j definiert. Falls f hinreichend glatt ist, kann diese Definition jetzt wie folgt erweitert werden:

Definition 7.4

$$[\tau_1, \ldots, \tau_n]f = \begin{cases} \frac{D^{n-1}}{(n-1)!} f(\tau_1), & \textit{falls} \quad \tau_1 = \tau_2 = \ldots = \tau_n, \\ \dfrac{[\tau_1, \ldots, \tau_{j-1}, \tau_{j+1}, \ldots, \tau_n]f - [\tau_1, \ldots, \tau_{k-1}, \tau_{k+1}, \ldots, \tau_n]f}{\tau_k - \tau_j}, & \textit{mit} \quad \tau_j \neq \tau_k \end{cases}$$

Satz 7.15 (Oskulierende Interpolation) *Seien* $\tau_0, \ldots, \tau_n \in [a, b]$ *und* $\{\tau_0, \ldots, , \tau_n\} = \{\overline{\tau}_0, \ldots, \overline{\tau}_m\}$, *wobei* $m \leq n$ *und* $i_s \geq 1$ *die Stellen* $\tau_0, \ldots, \tau_n$ *mit* $\overline{\tau}_s$ *zusammenfallen, so daß*

$$\sum_{s=0}^{m} i_s = n + 1$$

Sei $f \in C^{i_s - 1}$ *in einer Umgebung von* $\overline{\tau}_s$ *für* $0 \leq s \leq m$. *Dann gilt: Das Polynom* p *mit*

$$p(x) := \sum \left[\Pi_{k=0}^{j-1}(x - \tau_k)\right] [\tau_0, \ldots, \tau_j] f$$

interpoliert die Funktion f *in folgendem Sinne:*

$$D^k p(\overline{\tau}_s) = D^k f(\overline{\tau}_s), \quad 0 \leq k < i_s$$

für $0 \leq s \leq m$.

Satz 7.16 (Abschätzung dividierter Differenzen) *Sei* $f \in C^n$, *dann existiert* $\overline{x}$ *mit:*

$$[\tau_0, \ldots, \tau_n] f = \frac{D^n f(\overline{x})}{n!}.$$

Beweis: Folgt direkt aus der Hermite-Genocchi Formel. □

Satz 7.17 (Exakte Darstellung des Interpolationsfehlers) *Es sei* $f(x) = p(x) + e(x)$, *wobei* $p(x) \in P_n$ *die Funktion* f *in den Stellen* $\tau_0, \ldots, \tau_n$ *interpoliert und* $e(x)$ *der Fehler dieser Interpolation ist. Für diesen Fehler gilt*

1. $e(x) = \left[\Pi_{j=0}^{n}(x - \tau_j)\right] [\tau_0, \ldots, \tau_n] f.$
2. *Unter der Voraussetzung, daß* $f \in C^{(n+1)}$, *gibt es ein* $\tilde{x}$ *mit*

$$e(x) = \left[\Pi_{j=0}^{n}(x - \tau_j)\right] \cdot \frac{f^{(n+1)}(\tilde{x})}{(n+1)!}.$$

Beweis:

1.

$$\begin{aligned} f(x) &= [\tau_0] f + (x - \tau_0)[\tau_0, \tau_1] f + \cdots \\ &+ (x - \tau_0)(x - \tau_1) \cdots (x - \tau_{n-1}) \cdot [\tau_0, \tau_1, \cdots, \tau_n] f \\ &+ \Pi_{j=0}^{n}(x - \tau_j) \cdot [\tau_0, \ldots, \tau_n] f. \\ &= p(x) + e(x). \end{aligned} \tag{7.5}$$

2. Folgt aus (7.5) und Satz 7.28.

□

7.9 B-Splines

Definition 7.5 *Sei $t := (t_i)$ eine monoton steigende Folge. Der i-te normalisierte B-Spline der Ordnung $k \geq 1$ für die Folge t wird mit $B_{i,k,t}$ bezeichnet:*

$$B_{i,k,t}(x) := \begin{cases} (t_{i+k} - t_i)[t_i, \ldots, t_{i+k}](\cdot - x)_+^{k-1}, & \textit{für} \quad t_{i+k} > t_i \\ 0, & \textit{für} \quad t_{i+k} = t_i \end{cases}$$

Zur Abkürzung wird B_i statt $B_{i,k,t}$ manchmal geschrieben. (Die dividierte Differenz wirkt auf das erste Argument in der Stützfunktion.)

Beispiel:

1.

$$\begin{aligned} B_{i,1,t}(x) &= (t_{i+1} - t_i)[t_i, t_{i+1}](\cdot - x)_+^0 \\ &= (t_{i+1} - x)_+^0 - (t_i - x)_+^0 \\ &= \begin{cases} 0 - 0 = 0 & , \quad x \geq t_{i+1} \\ 1 - 0 = 1 & , \quad t_i \leq x < t_{i+1} \\ 1 - 1 = 0 & , \quad x < t_i \end{cases} \end{aligned}$$

Siehe Abbildung 7.5

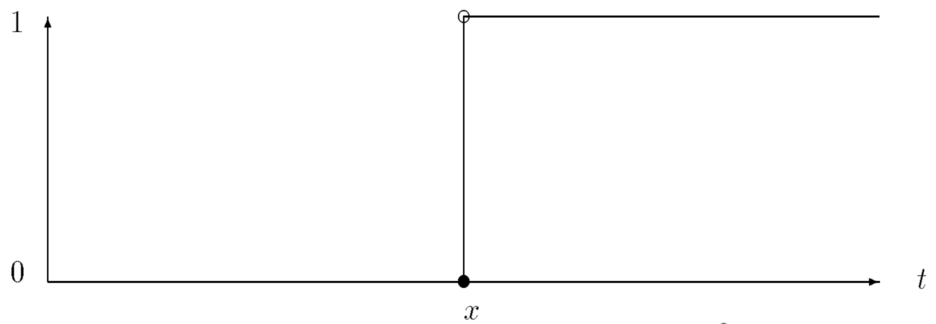


Abbildung 7.5: Die Funktion $(t - x)_+^0$.

2.

$$\begin{aligned} B_{i,2,t} &= (t_{i+2} - t_i)[t_i, t_{i+1}, t_{i+2}](\cdot - x)_+ \\ &= [t_{i+1}, t_{i+2}](\cdot - x)_+ - [t_i, t_{i+1}](\cdot - x)_+ \\ &= \begin{cases} 0 - 0 = 0 & , \quad x \geq t_{i+2} \\ \frac{(t_{i+2} - x) - 0}{t_{i+2} - t_{i+1}} - 0 = \frac{t_{i+2} - x}{t_{i+2} - t_{i+1}} & , \quad t_{i+1} \leq x < t_{i+2} \\ \frac{(t_{i+2} - x) - (t_{i+1} - x)}{t_{i+2} - t_{i+1}} - \frac{(t_{i+1} - x) - 0}{t_{i+1} - t_i} = \frac{x - t_i}{t_{i+1} - t_i} & , \quad t_i \leq x < t_{i+1} \\ 0 & , \quad x < t_i \end{cases} \end{aligned}$$

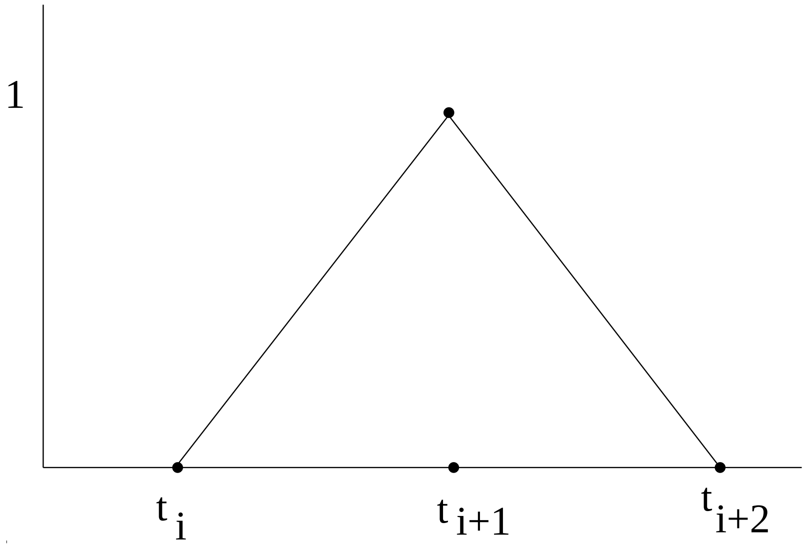


Abbildung 7.6: Die Funktion $B_{i,2,t}$.

Die Funktion $B_{i,2,t}$ wird in der Abbildung 7.6 dargestellt. Wegen ihrer Form wird sie oft als *Hutfunktion* bezeichnet.

Satz 7.18 (Formel von Leibnitz) *Sei* $f(x) = g(x)h(x)$ *für* $x \in \mathbb{R}$. *Dann gilt:*

$$[\tau_i, \ldots, \tau_{i+k}]f = \sum_{r=i}^{i+k} ([\tau_i, \ldots, \tau_r]g) \cdot ([\tau_r, \ldots, \tau_{i+k}]h)$$

Beweis: Sei

$$\begin{aligned} F(x) \quad := \quad & \left[\sum_{r=i}^{i+k} (x-\tau_i)\cdots(x-\tau_{r-1})[\tau_i, \ldots, \tau_r]g\right] \\ \cdot \quad & \left[\sum_{s=i}^{i+k} (x-\tau_{s+1})\cdots(x-\tau_{i+k})[\tau_s, \ldots, \tau_{i+k}]h\right] \end{aligned}$$

Dann ist $F(\tau_j) = f(\tau_j)$ für $i \leq j \leq i+k$.

In dem Produkt tragen Terme mit $r > s$ zum $F(\tau_j)$ nicht bei, da sie den Faktor $\prod_{j=i}^{i+k}(x-\tau_j)$ enthalten. Sei

$$\begin{aligned} \tilde{F}(x) \quad := \quad & \sum_{s=i}^{i+k} \sum_{r=i}^{s} \prod_{j=i}^{r-1} (x-\tau_j) \\ \cdot \quad & \prod_{\ell=s+1}^{i+k} (x-\tau_\ell) \cdot ([\tau_i, \ldots, \tau_r]g) \cdot ([\tau_s, \ldots, \tau_{i+k}]h) \end{aligned}$$

Folglich ist $\tilde{F}(\tau_j) = f(\tau_j)$ für $i \leq j \leq i+k$.
$\tilde{F}(x)$ ist ein Polynom vom Grade k, das die Funktion $f(x)$ in den $k+1$ Stützpunkten

$\tau_i, \ldots, \tau_{i+k}$ interpoliert. Durch den Vergleich der Koeffizienten von x^k erhält man:

$$[\tau_i, \ldots, \tau_{i+k}]f = \sum_{s=i}^{i+k} [\tau_1, \ldots, \tau_s]g \cdot [\tau_s, \ldots, \tau_{i+k}]h.$$

□

Satz 7.19 (Eigenschaften von B-Splines) *1. $B_i(x) = 0$ für $x \notin [t_i, t_{i+k}]$*

2. Auf dem Intervall $[t_j, t_{j+1}]$ sind höchstens k B-Splines ungleich null, und zwar die Splines $B_{i,k,t}$ mit $[t_j, t_{j+1}] \subset [t_i, t_{i+k}]$, d.h. $j \geq i$ und $j + 1 \leq i + k$, oder $j - k + 1 \leq i \leq j$.

3. Die B-Splines erfüllen die Rekursionsformel:

$$B_{i,k,t}(x) = \frac{(x - t_i)}{t_{i+k-1} - t_i} \cdot B_{i,k-1,t}(x) + \frac{(t_{i+k} - x)}{t_{i+k} - t_{i+1}} \cdot B_{i+1,k-1,t}(x), k \geq 2$$

falls $t_{i+k} > t_{i+1}$ und $t_{i+k-1} > t_i$.
Ist $t_{i+k} = t_{i+1}$ und $t_{i+k-1} > t_i$, fällt der Term mit $B_{i+1,k-1,t}(x)$ weg. Ist $t_{i+k} > t_{i+1}$ und $t_{i+k-1} = t_i$ fällt der Term mit $B_{i,k-1,t}(x)$ weg. Ist $t_{i+k} = t_i$, fallen beide Terme weg.

4. $B_{i,k,t}(x)$ ist nichtnegativ.

5.

$$\sum_{i=j-k+1}^{j} B_{i,k,t}(x) = 1 \,, \quad \text{für} \quad t_j < x < t_{j+1}$$

Beweis:

1. Für $x \geq t_{i+k}$ ist $(t_j - x)_+^{k-1} = 0$ für $i \leq j \leq i + k$. Für $x < t_i$ ist $(t_j - x)_+^{k-1} = (t_j - x)^{k-1}$ für $i \leq j \leq i + k$, so daß

$$[t_i, \ldots, t_{i+k}](\cdot - x)_+^{k-1} = [t_i, \ldots, t_{i+k}](\cdot - x)^{k-1} = 0$$

2. Klar.

3. Man wendet die Leibnitz-Formel auf das Produkt $(t-x)(t-x)_+^{k-2} = (t-x)_+^{k-1}$ an und erhält:

$$
\begin{aligned}
\frac{B_{i,k,t}(x)}{t_{i+k}-t_i} &= ([t_i](\cdot - x))([t_i,\dots,t_{i+k}](\cdot - x)_+^{k-2}) \\
&+ ([t_i,t_{i+1}](\cdot - x))([t_{i+1},\dots,t_{i+k}](\cdot - x)_+^{k-2}) \\
&= \frac{(t_i-x)}{t_{i+k}-t_i}\cdot([t_{i+1},\dots,t_{i+k}](\cdot - x)_+^{k-2} - [t_i,\dots,t_{i+k-1}](\cdot - x)_+^{k-2}) \\
&+ \frac{(t_{i+1}-x)-(t_i-x)}{t_{i+1}-t_i}\cdot\frac{1}{t_{i+k}-t_{i+1}}\cdot B_{i+1,k-1,t}(x) \\
&= \frac{(t_i-x)}{t_{i+k}-t_i}\cdot\left(\frac{1}{t_{i+k}-t_{i+1}}\cdot B_{i+1,k-1,t}(x) - \frac{1}{t_{i+k-1}-t_i}\cdot B_{i,k-1,t}(x)\right) \\
&+ \frac{1}{t_{i+k}-t_{i+1}}\cdot B_{i+1,k-1,t}(x) \\
&= \frac{x-t_i}{t_{i+k}-t_i}\cdot\frac{B_{i,k-1,t}(x)}{t_{i+k-1}-t_i} + \frac{t_{i+k}-x}{t_{i+k}-t_i}\cdot\frac{B_{i+1,k-1,t}(x)}{t_{i+k}-t_{i+1}}
\end{aligned}
$$

4. Folgt sofort aus der Rekursionsformel.

5. Für $j-k+1 \le i \le$ gilt $t_i \le t_j < t_{j+1} \le t_{i+k}$ so daß

$$B_{i,k,t}(x) = [t_{i+1},\dots,t_{i+k}](\cdot - x)_+^{k-1} - [t_i,\dots,t_{i+k-1}](\cdot - x)_+^{k-1}$$

Durch Summation ergibt sich:

$$\sum_{i=j-k+1}^{j} B_{i,k,t}(x) = [t_{j+1},\dots,t_{j+k}](\cdot - x)_+^{k-1} - [t_{j-k+1},\dots,t_j](\cdot - x)_+^{k-1}$$

Für $t \ge t_{j+1} > x$ ist $(t-x)_+^{k-1} = (t-x)^{k-1}$ und deshalb gilt:

$$[t_{j+1},\dots,t_{j+k}](\cdot - x)_+^{k-1} = \frac{1}{(k-1)!}\left(\frac{d}{dt}\right)^{k-1}(t-x)^{k-1} = 1\ ,$$

während für $t \le t_j < x$ ist $(t-x)_+^{k-1} = 0$, so daß

$$[t_{j-k+1},\dots,t_j](\cdot - x)_+^{k-1} = 0\ .$$

Die Behauptung folgt.

□

Definition 7.6 *Sei* $l \ge 0$, $k \ge 1, \xi := (\xi_i)_1^{l+1}, \xi_1 < \xi_2 < \cdots < \xi_{l+1}$ *(die Bruchstellenfolge) und* $\nu := (\nu_i)_2^l$, $0 \le \nu_i \le k$, *die Anzahl der Glattheitsbedingungen an den Bruchstellen. Man setzt:*

$$P_{k,\xi} := \{f : \cup_{i=1}^{l}(\xi_i,\xi_{i+1}) \longrightarrow \mathbb{R} : f|_{(\xi_i,\xi_{i+1})} \quad \textit{ist ein Polynom vom Grad}\, k-1\}$$

$f \in P_{k,\xi}$ *heißt* Polynomzug.

Bemerkung: $P_{k,\xi}$ besteht aus Polynomen vom Grade $\leq k-1$, d. h. $P_{k,\xi} \not\subset P_k$.

Definition 7.7 $P_{k,\xi,\nu} := \{f \in P_{k,\xi} \;:\; f \in\ C^{\nu_i - 1}$ *bei* $\;\xi_i,\ 2 \leq i \leq l\}$.

Ein typisches $f \in P_{k,\xi,\nu}$ wird in Abbildung 7.7 gezeigt.

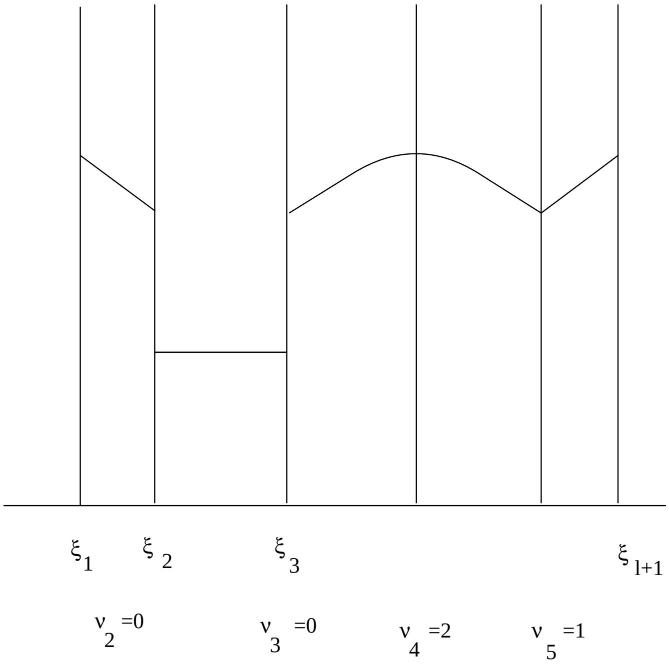


Abbildung 7.7: Polynomzüge

Die Dimension des Raumes $P_{k,\xi,\nu}$ ist:

$$\dim P_{k,\xi,\nu} = kl - \sum_{i=2}^{l} \nu_i = k + \sum_{i=2}^{l} (k - \nu_i).$$

Satz 7.20 (Curry und Schoenberg) *Sei* $P_{k,\xi,\nu}$ *mit Dimension* n *gegeben. Sei* $t := (t_i)_1^{n+k}$ *eine monoton zunehmende Folge mit:*

1. $t_1 \leq t_2 \leq \ldots t_k \leq \xi_1$
2. $\xi_{l+1} \leq t_{n+1} \leq \ldots \leq t_{n+k}$
3. *Für* $i = 2, \ldots, l$ *kommt die Zahl* ξ_i *genau* $(k - \nu_i)$*–mal in der Folge* t *vor.*

Dann sind die B-Splines $B_{i,k,t}$, $1 \leq i \leq n$*, eine Basis für den Raum* $P_{k,\xi,\nu}$.

Beweis: Siehe de Boor [1978, S. 113, Satz IX.1] oder de Boor [1990, S. 57 Satz b.d]. $\square$

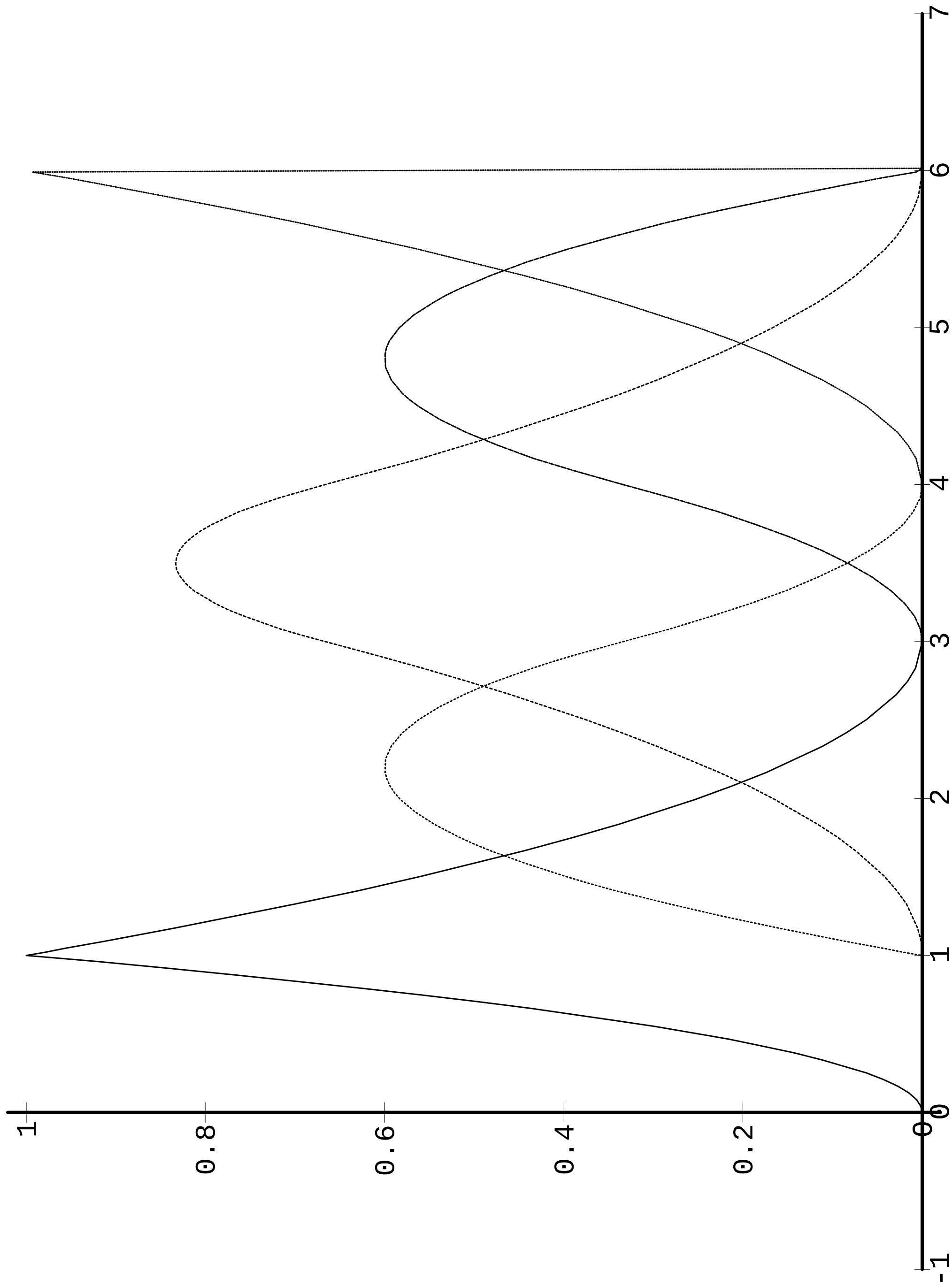


Abbildung 7.8: Quadratische B-Splines mit $t = (0, 1, 1, 3, 4, 6, 6, 6, 6)$

Wenn mehrere Knoten t_i zusammenfallen, dann erniedrigt sich die Glattheit der B-Splines wie aus der Abbildung 7.8 ersichtlich. (wie Fig. IX.2 de Boor [1978, S. 112])

Für den Fall $k = 2, \nu_i = 1$ für $2 \leq k \leq l$, ist $P_{k,\xi,\nu}$ gleich den Raum $S_2(\Delta)$ der linearen Splines.
Für den Fall $k = 4, \nu_i = 3$ für $2 \leq k \leq l$ ist $P_{k,\xi,\nu}$ gleich dem Raum $S_4(\Delta)$ der kubischen Splines.
Damit sehen wir daß die B-Splines als Basisfunktionen für linearen Splines als auch für die kubischen Splines gelten.

7.10 Anwendung der B-Splines

Die Rekursionsformel (siehe Gleichung: B-Splines-Rekursionsformel) bietet die Möglichkeit an, die B-Splines effizient und stabil zu berechnen. Seien $t_i \leq x < t_{i+1}$. Man möchte die nicht-Null B-Splines $B_{j,k,t}(x)$ berechnen, die in Abb. 7.10 gezeigt werden:

$$
\begin{array}{cccccc}
 & & O & \vdots & O \\
 & O & & \vdots & B_{i-k+1,k,t}, \\
O & & B_{i-2,3,t} & \vdots & \vdots \\
 & & B_{i-1,2,t} & \vdots & \vdots \\
B_{i,1,t} & & B_{i-1,3,t} & \vdots & \vdots \\
 & B_{i,2,t} & & \vdots & \vdots \\
O & & & \vdots & \vdots \\
 & & B_{i,3,t} & \vdots & \vdots \\
 & O & & \vdots & \vdots \\
 & & O & \vdots & \vdots \\
 & & & \vdots & B_{i-1,k,t} \\
 & & & \vdots & B_{i,k,t} \\
 & & & \vdots & O
\end{array}
$$

Abbildung 7.10: Splineberechnung

Das folgende Programm (aus dem Programmpaket FITPACK von Dierckx) berechnet die letzte Spalte in der Abbildung 7.10.

```
      subroutine splev(t,n,c,k,x,y,m,ier)
c  subroutine splev evaluates in a number of points x(i),i=1,2,...,m
c  a spline s(x) of degree k, given in its b-spline representation.
```

```
c
c  calling sequence:
c     call splev(t,n,c,k,x,y,m,ier)
c
c  input parameters:
c    t    : array,length n, which contains the position of the knots.
c    n    : integer, giving the total number of knots of s(x).
c    c    : array,length n, which contains the b-spline coefficients.
c    k    : integer, giving the degree of s(x).
c    x    : array,length m, which contains the points where s(x) must
c           be evaluated.
c    m    : integer, giving the number of points where s(x) must be
c           evaluated.
c
c  output parameter:
c    y    : array,length m, giving the value of s(x) at the different
c           points.
c    ier  : error flag
c      ier = 0 : normal return
c      ier =10 : invalid input data (see restrictions)
c
c  restrictions:
c    m >= 1
c    t(k+1) <= x(i) <= x(i+1) <= t(n-k), i=1,2,...,m-1.
c
c  other subroutines required: fpbspl.
c
c  references :
c    de boor c  : on calculating with b-splines, j. approximation theory
c                 6 (1972) 50-62.
c    cox m.g.   : the numerical evaluation of b-splines, j. inst. maths
c                 applics 10 (1972) 134-149.
c    dierckx p. : curve and surface fitting with splines, monographs on
c                 numerical analysis, oxford university press, 1993.
c
c  author :
c    p.dierckx
c    dept. computer science, k.u.leuven
c    celestijnenlaan 200a, b-3001 heverlee, belgium.
c    e-mail : Paul.Dierckx@cs.kuleuven.ac.be
c
c  latest update : march 1987
c
c..scalar arguments..
      integer n,k,m,ier
```

```
c..array arguments..
      real t(n),c(n),x(m),y(m)
c..local scalars..
      integer i,j,k1,l,ll,l1,nk1
      real arg,sp,tb,te
c..local array..
      real h(6)
c..
c  before starting computations a data check is made. if the input data
c  are invalid control is immediately repassed to the calling program.
      ier = 10
      if(m-1) 100,30,10
  10  do 20 i=2,m
        if(x(i).lt.x(i-1)) go to 100
  20  continue
  30  ier = 0
c  fetch tb and te, the boundaries of the approximation interval.
      k1 = k+1
      nk1 = n-k1
      tb = t(k1)
      te = t(nk1+1)
      l = k1
      l1 = l+1
c  main loop for the different points.
      do 80 i=1,m
c  fetch a new x-value arg.
        arg = x(i)
        if(arg.lt.tb) arg = tb
        if(arg.gt.te) arg = te
c  search for knot interval t(l) <= arg < t(l+1)
  40    if(arg.lt.t(l1).or. l.eq.nk1) go to 50
        l = l1
        l1 = l+1
        go to 40
c  evaluate the non-zero b-splines at arg.
  50    call fpbspl(t,n,k,arg,l,h)
c  find the value of s(x) at x=arg.
        sp = 0.
        ll = l-k1
        do 60 j=1,k1
          ll = ll+1
          sp = sp+c(ll)*h(j)
  60    continue
        y(i) = sp
  80  continue
```

```
 100  return
      end

c     cwc deleted this routine since it is also in curfit.f
c     subroutine fpbspl(t,n,k,x,l,h)
c  subroutine fpbspl evaluates the (k+1) non-zero b-splines of
c  degree k at t(l) <= x < t(l+1) using the stable recurrence
c  relation of de boor and cox.
c..
c..scalar arguments..
c     real x
c     integer n,k,l
c..array arguments..
c     real t(n),h(6)
c..local scalars..
c     real f,one
c     integer i,j,li,lj
c..local arrays..
c     real hh(5)
c..
c     one = 0.1e+01
c     h(1) = one
c     do 20 j=1,k
c       do 10 i=1,j
c         hh(i) = h(i)
c 10    continue
c       h(1) = 0.
c       do 20 i=1,j
c         li = l+i
c         lj = li-j
c         f = hh(i)/(t(li)-t(lj))
c         h(i) = h(i)+f*(t(li)-x)
c         h(i+1) = f*(x-t(lj))
c 20  continue
c     return
c      end
```

Der folgende Satz gibt Auskunft darüber, unter welchen Bedingungen eine Funktion f durch Splines interpoliert werden kann.

Es seien die Werte $f(\tau_1), \ldots, f(\tau_n)$ gegeben. Man fragt sich, ob es Konstanten $\alpha_1, \ldots, \alpha_n$ gibt mit:

$$\sum_{j=1}^{n} \alpha_j B_{j,k,t}(\tau_i) = f(\tau_i), \quad 1 \leq i \leq n.$$

Satz 7.21 (Schoenberg-Whitney) *Die Matrix* $(B_j(\tau_i))$, $\quad 1 \leq i, j \leq n$ *ist regulär genau dann, wenn*

$$B_{i,k,t}(\tau_i) \neq 0, \quad 1 \leq i \leq n, \tag{7.6}$$

d. h. genau dann, wenn

$$t_i < \tau_i < t_{i+k}\ , \quad 1 \leq i \leq n.$$

Beweis: de Boor [1978, S. 200, Satz XIII.1] oder de Boor [1990, S. 94, Satz 10b]. □

Seien nun die Schoenberg-Whitney Bedingungen (7.6) erfüllt. Folgende Eigenschaften der Matrix $B := (B_{j,k,t}(\tau_i)) = (b_{ij})$ sind bekannt:

Satz 7.22 *Die Matrix* $B = (B_{j,k,t}(\tau_i)) = (bj)$ *hat die Bandbreite* $2k+1$*, d. h.*

$$b_{ij} = 0 \qquad \textit{für} \quad |j-i| > k.$$

Beweis: Wir wissen, daß

$$t_i < \tau_i < t_{i+k}.$$

Ist $j > i + k$, so folgt

$$\tau_i < t_{i+k} \leq t_j,$$

so daß $\tau_i \notin [t_j, t_{j+k}]$ und folglich $b_{ij} = B_{j,k,t}(\tau_i) = 0$. Auf die gleiche Weise sieht man, daß $\tau_i \notin [t_j, t_{j+k}]$ für $j < i - k$ □

Definition 7.8 *Eine* $n \times n$ *reelle Matrix* A *heißt* total positiv, *wenn jede* $r \times r$ *Unterdeterminante,* $1 \leq r \leq n$*, nicht negativ ist.*

Satz 7.23 *Die* $n \times n$ *Matrix* $(B_{j,k,t}(\tau_i))$ *ist total positiv.*

Beweis: Siehe de Boor [1990, S. 102]. □

Es ist bekannt, daß, wenn eine Matrix A regulär und total positiv ist, die Lösung der linearen Gleichungen $Ax = b$ mit Hilfe der Gaußschen Eliminationsverfahren *ohne Permutation* möglich und numerisch stabil ist. Man folgert, daß, wenn die Schoenberg-Whitney Bedingungen erfüllt sind, das Gleichungssystem mit der Bandmatrix $(B_{j,k,t}(\tau_i))$ effizient und problemlos gelöst werden kann.

Beispiel: Die Ergebnisse einiger Volkszählungen in den U.S.A. sind:

Jahr	Volkszählung
1900	75994575
1910	91972266
1920	105710620
1930	122775046
1940	131669275
1950	150697361
1960	179323175
1970	203235298

Man möchte diese Daten durch kubische Splines interpolieren, d. h. man möchte die Funktion f an die Stellen τ_i durch eine Funktion $s \in P_{4,\xi,\nu}$ mit $\nu = (3, 3, \ldots, 3) \in \mathbb{R}^{l-1}$ interpolieren. Es gibt 8 Stützpunkte τ_i, d.h. 8 Bedingungen. Die Dimension von $P_{4,\xi,\nu}$ ist $(l+1)+2$. Wir wählen $l = 5$, damit die Dimension von $P_{4,\xi,\nu}$ gleich die Anzahl der Bedingungen ist. Als Stützpunkte ξ_i wählen wir:

$$
\begin{array}{llcl}
\mathbf{i} & \boldsymbol{\xi}_\mathbf{i} & & \\
1 & 1900 & = & \tau_1 \\
2 & 1920 & = & \tau_3 \\
3 & 1930 & = & \tau_4 \\
4 & 1940 & = & \tau_5 \\
5 & 1950 & = & \tau_6 \\
6 & 1970 & = & \tau_8
\end{array}
$$

Die Stellen τ_2 und τ_7 kommen in $\xi = (\xi_i)$ nicht vor, so daß die Interpolationsfunktion s an diversen Stellen glatt ist (die sogenannte „not-a-knot"-Bedingung).

Die Koeffizienten c_i, wo

$$s(x) = \sum_{j=1}^{12} c_j B_{j,4,\nu}(x)$$

werden mit Hilfe des FITPACK Programms curfit.f (siehe Dierckx [1993]) mit dem Eingabeparameter $ier = -1$ berechnet:

i	c	t
1	75994575	1900
2	90186238	1900
3	96663990	1900
4	125963130	1900
5	128808980	1920
6	155485805	1930
7	192420282	1940
8	203235298	1950
9	0	1970
10	0	1970
11	0	1970
12	0	1970

Mit Hilfe dieser Approximation wird die Volkszählung für 1980 als 203235298 geschätzt. Die wirkliche Zahl 1980 war 226547082.

Bemerkung: Die Schoenberg-Whitney-Bedingungen werden durch τ_1 und τ_8 *nicht* erfüllt. Es gibt sicherlich eine Erweiterung des Satzes von Schoenberg und Whitney, so daß die schwächeren Bedingungen

$$t_1 = t_2 = \cdots = t_k = \xi_1 \leq \tau_1 < \xi_2 < \xi_l < \tau_n \leq \xi_{l+1} = t_{n+1} = t_{n+2} \ldots = t_{n+k}$$

erlaubt sind.

7.11 Vollständige Kubische Splines

Sei $f \in C^{(1)}[a,b]$. Der *vollständige* (=*complete*) *Kubische Spline* $I_4 f$, der f interpoliert, ist die Spline-Funktion s_Δ mit

$$\begin{aligned} s_\Delta(\tau_i) &= f(\tau_i)\,, \quad 1 \leq i \leq n \\ s'_\Delta(\tau_1) &= f'(\tau_1)\,, \\ s'_\Delta(\tau_n) &= f'(\tau_n)\,. \end{aligned}$$

Um $I_4 f$ darzustellen, ist es nützlich, die Steigungen von s_Δ

$$s_i := s'_\Delta(\tau_i)\ ,\ 1 \leq i \leq n$$

in den Knotenpunkten als Hilfsmittel einzuführen.

Die Spline-Funktion s_Δ läßt sich im Intervall (τ_i, τ_{i+1}), $1 \le i \le n-1$ folgendermaßen darstellen:

$$\begin{array}{c|cccc}
 & [\,]s & [,]s & [,,]s & [,,,]s \\ \hline
\tau_i & f_i & & & \\
 & & s_i & & \\
\tau_i & f_i & & \frac{[\tau_i,\tau_{i+1}]f - s_i}{\Delta_i} & \\
 & & [\tau_i, \tau_{i+1}]f & & \frac{s_{i+1}+s_i-2[\tau_i,\tau_{i+1}]f}{(\Delta_i)^2} \\
\tau_{i+1} & f_{i+1} & & \frac{s_{i+1}-[\tau_i,\tau_{i+1}]f}{\Delta_i} & \\
 & & s_{i+1} & & \\
\tau_{i+1} & f_{i+1} & & &
\end{array}$$

Es folgt, daß im Intervall (τ_i, τ_{i+1}) :

$$\begin{aligned}
s_\Delta(x) &= f_i + (x-\tau_i)s_i + (x-\tau_i)^2 \left(\frac{[\tau_i, \tau_{i+1}]f - s_i}{\Delta_i} \right) \\
&+ (x-\tau_i)^2 (x-\tau_{i+1}) \left(\frac{s_{i+1} + s_i - 2[\tau_i, \tau_{i+1}]f}{\Delta_i^2} \right)
\end{aligned}$$

Genauso folgt, daß im Intervall (τ_{i-1}, τ_i) :

$$\begin{aligned}
s_\Delta(x) &= f_i + (x-\tau_i)s_i + (x-\tau_i)^2 \frac{s_i - [\tau_i, \tau_{i-1}]f}{\Delta_{i-1}} \\
&+ (x-\tau_i)^2 (x-\tau_{i-1}) \left(\frac{s_{i-1} + s_i - 2[\tau_i, \tau_{i-1}]f}{(\Delta_{i-1}^2} \right)
\end{aligned}$$

Die Bedingung $s''_\Delta(\tau_i + 0) = s''_\Delta(\tau_i - 0)$ ergibt für $1 < i < n$:

$$\begin{aligned}
& 2 \left(\frac{[\tau_i, \tau_{i+1}]f - s_i}{\Delta_i} \right) - 2 \left(\frac{s_{i+1} + s_i - 2[\tau_i, \tau_{i+1}]f}{\Delta_i} \right) \\
= \; & 2 \left(\frac{s_i - [\tau_i, \tau_{i-1}]f}{\Delta_{i-1}} \right) + 2 \left(\frac{s_{i-1} + s_i - 2[\tau_i, \tau_{i-1}]f}{\Delta_{i-1}} \right)
\end{aligned}$$

oder, da $[\tau_i, \tau_{i-1}]f = [\tau_{i-1}, \tau_i]f$,

$$\Delta_i s_{i-1} + \Delta_{i-1} s_{i+1} + 2\, s_i (\Delta_i + \Delta_{i-1}) = b_i$$

mit

$$b_i := 3(\Delta_{i-1}[\tau_i, \tau_{i+1}]f + \Delta_i [\tau_{i-1}, \tau_i]f) \,.$$

Der $(n-2)$-Vektor

$$s = (s_2, \ldots, s_{n-1})$$

ist folglich eine Lösung der Gleichung

$$A\, s = b \; - (\Delta_2 f'(\tau_1)\,, 0, \ldots, 0, \Delta_{n-2} f'(\tau_n))^T$$

wobei A eine $(n-2) \times (n-2)$ tridiagonale Matrix ist.

Beispiel: Um die Methode zu verdeutlichen, werden die Berechnungen in einem sehr einfachen Fall durchgeführt:

$$\begin{aligned} f(x) &= \sqrt{1+x} \\ \tau_1 = 0 \quad , \quad \tau_2 = 3 \quad &, \quad \tau_3 = 8 \quad , \quad n = 3\,. \\ s_1 = f'(\tau_1) = \frac{1}{2} \quad &, \quad s_3 = f'(\tau_3) = \frac{1}{6} \end{aligned}$$

Für das Intervall (τ_2, τ_3) erhalten wir die dividierten Differenzen.

$$\begin{array}{llcllll}
\tau_2 = 3 & [\tau_2]s_\Delta & = & f(\tau_2) = 2 & & & \\
 & & & [\tau_2,\tau_2]s_\Delta = f'(\tau_2) = s_2 & & & \\
\tau_2 = 3 & [\tau_2]s_\Delta & = & f(\tau_2) = 2 & [\tau_2,\tau_2,\tau_3]s_\Delta = & \dfrac{\frac{1}{5} - s_2}{8-3} & = \dfrac{\frac{1}{5} - s_2}{5} \\
 & & & [\tau_2,\tau_3]s_\Delta = \dfrac{3-2}{8-3} = \dfrac{1}{5} & & [\tau_2,\tau_2,\tau_3,\tau_3]s_\Delta = & \dfrac{s_3 + s_2 - \frac{2}{5}}{25} \\
\tau_3 = 8 & [\tau_3]s_\Delta & = & f(\tau_3) = 3 & [\tau_2,\tau_3,\tau_3]s_\Delta = & \dfrac{s_3 - \frac{1}{5}}{8-3} & = \dfrac{s_3 - \frac{1}{5}}{5} \\
 & & & [\tau_3,\tau_3]s_\Delta = \quad f'(\tau_3) = s_3 & & & \\
\tau_3 = 8 & [\tau_3]s_\Delta & = & f(\tau_3) = 3 & & &
\end{array}$$

Das Polynom $p_2(x)$, das die Bedingungen erfüllt,

$$\begin{aligned} p_2(\tau_2) &= f(\tau_2) = 2 \\ p_2(\tau_3) &= f(\tau_3) = 3 \\ p_2'(\tau_2) &= s_2 \\ p_2'(\tau_3) &= s_3 \end{aligned}$$

ist:

$$\begin{aligned} p_2(x) &= [\tau_2]f + (x-\tau_2)[\tau_2,\tau_2]f + (x-\tau_2)^2[\tau_2,\tau_2,\tau_3]f \\ &+ (x-\tau_2)^2(x-\tau_3)[\tau_2,\tau_2,\tau_3,\tau_3]f \\ &= 2 + s_2(x-3) + \frac{(\frac{1}{5} - s_2)}{5}(x-3)^2 + \frac{(s_3 + s_2 - \frac{2}{5})}{25}(x-3)^2(x-8) \end{aligned}$$

wie direkte Berechnungen bestätigen.

Ebenso gilt für das Polynom $p_1(x)$, das die Bedingungen erfüllt:

$$\begin{aligned}
p_1(\tau_1) &= f(\tau_1) = 1 \\
p_1(\tau_2) &= f(\tau_2) = 2 \\
p_1^{'}(\tau_1) &= f^{'}(\tau_1) = s_1 \\
p_1^{'}(\tau_2) &= f^{'}(\tau_2) = s_2 \\
p_1(x) &= [\tau_2]f + (x-\tau_2)[\tau_2\tau_2]f + (x-\tau_2)^2[\tau_2,\tau_2,\tau_1]f \\
&+ (x-\tau_2)^3(x-\tau_1)[\tau_2,\tau_2,\tau_1,\tau_1]f \\
&= 2 + s_2(x-3) + \frac{s_2-\frac{1}{3}}{3}(x-3)^2 + \frac{s_1+s_2-\frac{2}{3}}{9}(x-3)^2(x-0)
\end{aligned}$$

wie direkte Berechnungen bestätigen.

$$\begin{array}{llllll}
\tau_2 = 3 & [\tau_2]s_\Delta & = & f(\tau_2) = 2 & & \\
 & & & [\tau_2,\tau_2]s_\Delta = f^{'}(\tau_2) = s_2 & & \\
\tau_2 = 3 & [\tau_2]s_\Delta & = & f(\tau_2) = 2 \quad [\tau_2,\tau_2,\tau_1]s_\Delta = & \dfrac{\frac{1}{3}-s_2}{0-3} & = \dfrac{s_2-\frac{1}{3}}{3} \\
 & & & [\tau_2,\tau_1]s_\Delta = \dfrac{1-2}{0-3} = \dfrac{1}{3} & [\tau_2,\tau_2,\tau_1,\tau_1]s_\Delta = & \dfrac{s_1+s_2-\frac{2}{3}}{9} \\
\tau_1 = 0 & [\tau_1]s_\Delta & = & f(\tau_1) = 1 \quad [\tau_2,\tau_1,\tau_1]s_\Delta = & \dfrac{s_1-\frac{1}{3}}{0-3} & = \dfrac{\frac{1}{3}-s_1}{3} \\
 & & & [\tau_1,\tau_1]s_\Delta = \quad f^{'}(\tau_1) = s_1 & & \\
\tau_1 = 0 & [\tau_1]s_\Delta & = & f(\tau_1) = 1 & &
\end{array}$$

Die Bedingung

$$p_1^{''}(\tau_2) = p_2^{''}(\tau_2)$$

ergibt die folgende Gleichung:

$$\begin{aligned}
p_1^{''}(3) &= \frac{2}{3}\left(s_2 - \frac{1}{3}\right) + \frac{6}{9}\left(s_1 + s_2 - \frac{2}{3}\right) \\
&= p_2^{''}(3) \\
&= \frac{2}{5}\left(\frac{1}{5} - s_2\right) - \frac{10}{25}\left(s_3 + s_2 - \frac{2}{5}\right)
\end{aligned}$$

oder

$$\frac{2}{3}(s_1 + 2s_2 - 1) = \frac{2}{5}\left(-s_3 - 2s_2 + \frac{3}{5}\right)$$

oder

$$5s_1 + 3s_3 + 2s_2(5+3) = \frac{34}{5}\ .$$

Da $s_1 = \frac{1}{2}$ und $s_3 = \frac{1}{6}$, folgt $s_2 = \frac{19}{80}$, d.h.

$$\begin{aligned} p_1(x) &= (2160 + 1080x - 171x^2 + 17x^3)/2160 \, , \\ p_2(x) &= (7248 + 1752x - 59x^2 + x^3)/6000 \, . \end{aligned}$$

Diese Terme wurden mit Hilfe eines Symbolmanipulationssystems erzeugt.

7.12 Vollständige Kubische Splines: Fortsetzung

Satz 7.24 *Für jedes $f \in C^{(1)}[a,b]$ gibt es genau eine vollständige Kubische Spline-Funktion $s_\Delta = I_4(f) \in S_4(\Delta)$ mit*

$$\begin{aligned} s_\Delta(\tau_i) &= f(\tau_i) \, , \quad 1 \le i \le n \\ s_\Delta^{'}(\tau_1) &= f^{'}(\tau_1) \, , \\ s_\Delta^{'}(\tau_n) &= f^{'}(\tau_n) \, . \end{aligned}$$

Beweis: Es genügt zu zeigen, daß die Matrix

$$A = \begin{pmatrix} 2(\Delta_1 + \Delta_2) & \Delta_1 & & \\ \Delta_3 & & \ddots & \\ \ddots & & & \Delta_{n-2} \\ & \Delta_n & 2(\Delta_{n-1} + \Delta_n) & \end{pmatrix}$$

regulär ist, was sicherlich der Fall ist, wenn $\mathrm{Kern}(A) = \{0\}$. □

Satz 7.25 *Seien $f \in C^{(2)}[a,b]$ und $e := f - I_4 f$. Dann ist $e^{''}$ orthogonal zu $S_2(\Delta)$ in $L_2[a,b]$, d.h.*

$$\int_a^b e^{''}(x)s(x)dx = 0 \, , \quad \textit{für alle} \quad s \in S_2(\Delta). \tag{7.7}$$

Beweis: Sei $h \in C^2[a,b]$. Es gilt (Taylorsche Formel):

$$\begin{aligned} h(x) &= h(a) + (x-a)h^{'}(a) + \int_a^x (x-t)h^{''}(t)dt \\ &= h(a) + (x-a)h^{'}(a) + \int_a^b (x-t)_+ h^{''}(t)dt \end{aligned}$$

Es folgt, da die zweite dividierte Differenz einer linearen Funktion gleich null ist:

$$[\tau_{i-1}, \tau_i, \tau_{i+1}]h(x) = \int_a^b [\tau_{i-1}, \tau_i, \tau_{i+1}](\cdot - t)_+ h^{''}(t)dt = \int_a^b \hat{H}_i(t)h^{''}(t)dt \, ,$$

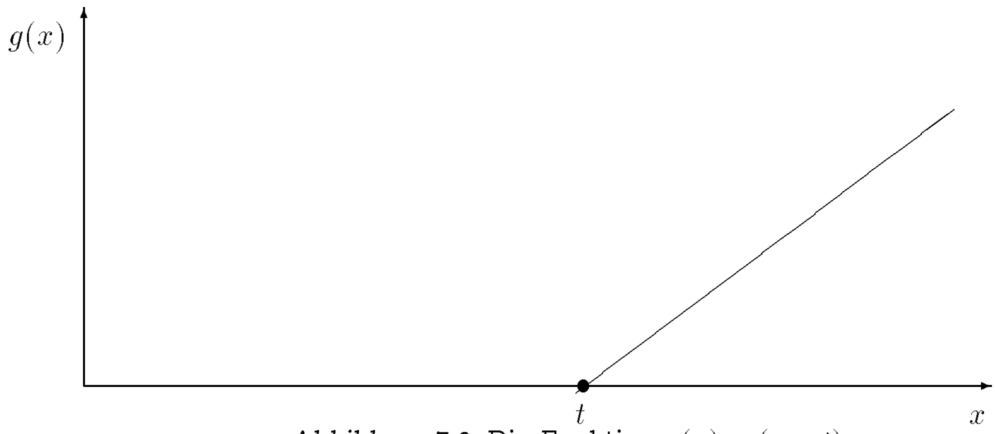


Abbildung 7.9: Die Funktion $g(x) = (x-t)_+$

wobei

$$\hat{H}_i(x) = \frac{2}{\Delta_i + \Delta_{i-1}} \begin{cases} \frac{(x-\tau_{i-1})}{\Delta_{i-1}} & , \text{ für } \quad \tau_{i-1} < x \leq \tau_i \\ \frac{(\tau_{i+1}-x)}{\Delta_i} & , \text{ für } \quad \tau_i < x < \tau_{i+1} \\ 0 & , \qquad \text{sonst.} \end{cases}$$

Diese Gleichung gilt auch für $i = 1$ und $i = n$ mit $\tau_o := \tau_1$ und $\tau_{n+1} := \tau_n$, was unter Berücksichtigung der Eigenschaften von $(x-t)_+$ (siehe Abbildung 7.9) sofort einzusehen ist. In diese Gleichung setzen wir $h = e = f - I_4 f$:

$$0 = [\tau_{i-1}, \tau_i, \tau_{i+1}]e = \int_a^b \hat{H}_i(t) e''(t) dt \ , \quad 1 \leq i \leq n \ .$$

Die Funktionen

$$H_i(x) = \frac{\Delta_i + \Delta_{i-1}}{2} \hat{H}_i(x) \ , \quad 1 \leq i \leq n \tag{7.8}$$

bilden eine Basis für $S_2(\Delta)$, und (7.7) folgt sofort. □

Bemerkung: $H_i(x)$ ist die B-Spline $B_{i-1,2,t}(x)$.

Satz 7.26 *Sei $f \in C^2[a,b]$. Dann gilt:*

$$\int_a^b [f''(x)]^2 dx = \int_a^b [(I_4 f)''(x)]^2 dx + \int_a^b [(f - I_4(f))''(x)]^2 dx \ .$$

Beweis:

$$\begin{aligned}(f'',f'') &= ((f-I_4f)''+(I_4f)'' \quad , \quad (f-I_4f)''+(I_4f)'')\\ &= ((f-I_4f)'' \quad , \quad (f-I_4f)'')+((I_4f)'' \quad , \quad (I_4f)'')+2((f-I_4f)'' \quad , \quad (I_4f)'')\\ &= (e'',e'')+((I_4f)'' \quad , \quad (I_4f)'')\ ,\end{aligned}$$

da $e'' \perp S_2(\Delta)$ (Satz 7.25) und $(I_4f)'' \in S_2(\Delta)$. □

Satz 7.27 *Sei $f \in C^2[a,b]$. Sei $\phi \in C^2[a,b]$ mit:*

$$\begin{aligned}\phi(\tau_i) &= f(\tau_i)\ , \quad 1 \le i \le n\\ \phi'(a) &= f'(a)\ ,\\ \phi'(b) &= f'(b)\ .\end{aligned}$$

Dann gilt:

$$\int_a^b \phi''(x)^2 dx \ge \int_a^b [(I_4(f))''(x)]^2 dx. \tag{7.9}$$

mit Gleichheit genau dann, wenn $\phi = I_4f$ gilt.

Beweis: Es folgt aus Satz 7.27, daß

$$\begin{aligned}\int_a^b \phi''(x)^2 dx &\ge \int_a^b [(I_4\phi)''(x)]^2 dx + \int_a^b [(\phi - I_4(\phi))''(x)]^2 dx\\ &= \int_a^b [(I_4f)''(x)]^2 + \int_a^b [(\phi - I_4(f))''(x)]^2 dx\end{aligned}$$

da $I_4f = I_4\phi$.

Die Ungleichung (7.9) folgt sofort. Es gilt Gleichheit genau dann, wenn

$$(\phi - I_4(f))''(x) = 0\ , \quad \text{für} \quad x \in (a,b)\ .$$

Durch Integration ergibt sich:

$$\phi = I_4f + Ax + B\ ,$$

wobei A und B Konstanten sind. Aus den Bedingungen $\phi(a) = f(a)$ und $\phi'(a) = f'(a)$ schließt man, daß $A = B = 0$ und $\phi = I_4f$. □

Bemerkung: Die Energie eines elastischen Stabes $y = \phi(x)$ ist

$$\int_a^b \frac{[\phi''(x)]^2}{[1+(\phi'(x))^2]^{3/2}}\, dx\ .$$

7.13 Ausgleichrechnung

Eine symmetrische Bilinearform s auf einem reellen Vektorraum V heißt *Skalarprodukt*, falls:

$$\begin{aligned}
s &: V \times V \longrightarrow \mathbb{R}\,, \\
s(v,\cdot) &: V \longrightarrow \mathbb{R} \qquad \text{linear}\,, \\
s(\cdot,w) &: V \longrightarrow \mathbb{R} \qquad \text{linear}\,, \\
s(v,w) &= s(w,v)\,, \qquad \text{für alle} \quad v,w \in V\,, \\
s(v,v) &> 0\,, \qquad \text{für alle} \quad v \neq 0\,.
\end{aligned}$$

Ein reeller Vektorraum H heißt *Prähilbert-Raum*, wenn ein Skalarprodukt $(\cdot,\cdot)$ auf H erklärt ist. Er heißt *Hilbert-Raum*, wenn er bezüglich der Norm

$$\|x\| = (x,x)^{1/2}$$

vollständig ist, d.h. wenn jede Cauchy-Folge einen Limes in H hat.

Satz 7.28 *Seien H ein Prähilbert-Raum und $x_1,\ldots,x_n \in H$ linear unabhängig. Seien*

$$V = span\{x_i : 1 \leq i \leq n\}$$

und $f \in H$.

Es gibt eine eindeutige beste Approximation $f^* \in V$ zu f bzgl. der Norm $\|\cdot\|$. Weiter gilt:

$$f^* = \sum_{k=1}^{n} a_k x_k\,,$$

wobei die Koeffizienten a_k die eindeutige Lösung der Gleichung $A\,a = b$ bilden mit

$$\begin{aligned}
A &= (a_{ij})\,, \qquad 1 \leq i\,,\; j \leq n \\
a_{ij} &:= (x_i.x_j) \\
a &:= (a_1,\ldots,a_n) \in \mathbb{R}^n \\
b &:= (b_1,\ldots,b_n) \in \mathbb{R}^n \\
b_k &:= (f,x_k)\,.
\end{aligned}$$

Beweis: Man setze

$$F(a_1,\ldots,a_n) := \|f - \sum_{k=1}^{n} a_k x_k\|^2$$

und minimiere. Es folgt:

$$\begin{aligned}
\frac{\partial F}{\partial a_i} &= \frac{\partial}{\partial a_i}\left[(f,f) - 2\sum_{k=1}^{n} a_k(f,x_k) + \sum_{k=1}^{n}\sum_{j=1}^{n} a_k a_j (x_k,x_j)\right] \\
&= -2(f,x_i) + 2\sum_{j=1}^{n} a_j(x_i,x_j)
\end{aligned}$$

Die notwendigen Bedingungen dafür, daß F im Punkt $\tilde{a}$ ein Minimum hat, sind

$$\frac{\partial F}{\partial \tilde{a}_i} = 0 , \qquad 1 \leq i \leq n ,$$

d.h.

$$A\,\tilde{a} = b. \tag{7.10}$$

Die Matrix A ist symmetrisch und positiv definit: Sei $a \in \mathbb{R}^n, a \neq 0$. Definiere:

$$v := \sum_{k=1}^{n} a_k x_k,$$

dann gilt:

$$a^T A a = (v, v) > 0.$$

Daß die Bedingungen (7.10) hinreichend sind, folgt aus der Identität:

$$\begin{aligned} F(a) - F(\tilde{a}) &= -2a^T b + 2\tilde{a}^T b + a^T A a - \tilde{a}^T A \tilde{a} \\ &= (a-\tilde{a})^T A (a-\tilde{a}) + 2(a-\tilde{a})^T (A\tilde{a} - b) \\ &= (a-\tilde{a})^T A (a-\tilde{a}) \\ &> 0 \qquad \text{falls} \quad a \neq \tilde{a}. \end{aligned}$$

□

Bemerkung: Die Gleichung $A\,a = b$ heißt *Normal-Gleichung.*

Bemerkung: $(f - f^*) \perp V$.

7.14 Fehlerabschätzungen für vollständige Kubische Spline-Funktionen

Definition 7.9 *Sei $H = L_2(a,b)$ mit*

$$\begin{aligned} \|f\| &= \int_a^b [f(x)]^2 dx , \\ (f,g) &= \int_a^b f(x) g(x) dx . \end{aligned}$$

Sei $V = S_2(\Delta)$ mit $\Delta = \{\tau_1, \ldots, \tau_n\}$. Die beste Approximation $f^* \in V$ an $f \in H$ wird mit $L_2(\Delta)f$ bezeichnet, so daß

$$L_2(\Delta) : L_2(a,b) \longrightarrow S_2(\Delta) , \qquad f \longmapsto f^*$$

Hilfssatz 7.1 *Sei* $s = L_2(\Delta)f$,

$$s(x) = \sum_{k=1}^{n} a_k H_k(x) \ .$$

Dann ist

$$A\,a = b$$

wobei $H_k(x)$ *die B-Spline* $B_{k-1,2,t}(x)$ *ist; siehe (7.8).*

$$\begin{aligned} a &= (a_1, \dots, a_n) \ , \\ b_k &= \int_a^b H_k(x) f(x) dx \ , \end{aligned}$$

$$A = \begin{pmatrix} [\Delta_o + \Delta_1]/3 & \Delta_1/6 & & \bigcirc \\ \Delta_1/6 & [\Delta_1 + \Delta_2]/3 & \Delta_2/6 & \\ & & \ddots & \\ & & \Delta_{n-1}/6 & [\Delta_{n-1} + \Delta_n]/3 \end{pmatrix}$$

mit

$$\begin{aligned} \Delta_i &:= \tau_{i+1} - \tau_i \\ \tau_o &:= \tau_1 \ , \quad \tau_{n+1} := \tau_n \end{aligned}$$

Beweis: Direkte Berechnung der Normalen–Gleichung. □

Satz 7.29 *Sei* $g \in C[a,b]$. *Es gilt:*

$$\|L_2(\Delta)g\|_\infty \le 3\|g\|_\infty$$

und

$$\|g - L_2(\Delta)g\|_\infty \le 4\ \rho_\infty(g, S_2(\Delta))$$

Beweis: de Boor [1978, S. 43]. □

Satz 7.30 *Die zweite Ableitung der Kubischen Spline-Approximation an* $f \in C^2[a,b]$ *ist die beste Approximation an* f'' *in* $L_2(a,b)$:

$$(I_4 f)'' = L_2(\Delta) f'' \ .$$

Beweis: Sei $e := f - I_4 f$. Es gilt: $e'' \perp S_2(\Delta)$, d.h.

$$(I_4 f)'' = \sum \alpha_i H_i$$

mit

$$(f'' - (I_4 f)'', H_k) = 0\ , \quad 1 \leq k \leq n$$

oder

$$\sum_{i=1}^{n} \alpha_i (H_i, H_k) = (f'', H_k)\ .$$

$\square$

Folgerungen: de Boor [1978, S. 68]

1. $\|e''\|_\infty = \|f'' - (I_4 f)''\|_\infty \leq 4\ \rho_\infty(f'', S_2(\Delta)) \leq \frac{1}{2}|\Delta|^2 \|f^{(4)}\|_\infty$
2. $\|e\|_\infty \leq \frac{|\Delta|^2}{8}\ \|e''\|_\infty \leq \frac{|\Delta|^4}{16}\ \|f^{(4)}\|_\infty$, wenn $f \in C^4[a,b]$.

Literatur

Böhmer, K.: Spline-Funktionen. Stuttgart: Teubner, 1974.

Cox, M. G.: On the numerical evaluation of B-Splines. J. Inst. Math. Applics. 10 (1972), 134-149.

de Boor, C.: A Practical Guide to Splines. New York, Springer, 1978.

de Boor, C.: Splinefunktionen. Birkhäuser, 1990.

Dierckx, P.: Curve and Surface Fitting with Splines. Oxford: Clarendon Press, 1993.

Forster, O.: Analysis I, 4. Auflage. Braunschweig: Vieweg, 1983.

Korovkin, P.P.: Linear Operators and Approximation Theory. New York: Gordon and Breach, 1960.

Schumaker, L.L.: Spline-Functions: Basic Theory. New York: John Wiley.

Kapitel 8

Nichtlineare Gleichungen

8.1 Einleitung

Wir betrachten das Problem: Sei $y \in \mathbb{R}^n$ gegeben und $F : G \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$. Finde $x^* \in G$ mit

$$F(x^*) = y \ . \tag{8.1}$$

x^* heißt *Lösung* der Gleichung (8.1).

(8.1) läßt sich in zwei äquivalente Gestalten transformieren:

1. Sei $f : G \longrightarrow \mathbb{R}^n$ mit $f(x) := F(x) - y$. Dann ist x^* eine Lösung der Gleichung (8.1) genau dann, wenn

   $$f(x^*) = 0 \ , \tag{8.2}$$

   d.h., wenn x^* eine *Nullstelle* von f ist.

2. Sei $g : G \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$ mit $g(x) := f(x) + x$. x heißt *Fixpunkt* von g, falls

   $$g(x) = x \ . \tag{8.3}$$

   Dann ist x^* eine Nullstelle von f genau dann, wenn x^* ein Fixpunkt von g ist.

Die Gleichungen (8.1), (8.2) und (8.3) sind mathematisch äquivalent, aber es kann vorkommen, daß eine Darstellungsmethode für ein bestimmtes Problem oder numerisches Verfahren besonders geeignet ist.

In Einzelfällen sind die Lösungen der Gleichung (8.1) analytisch darstellbar. Beispiele dafür sind (für $n = 1$):

a) f sei ein Polynom vom Grad $m \leq 4$. In der Praxis werden die analytischen Formeln nur für $m \leq 2$ benutzt, da sie für $m = 3, 4$ sehr umständlich und numerisch ungeeignet sind. Es läßt sich hier erwähnen, daß infolge der Galoisschen Theorie die Nullstellen eines Polynoms vom Grad größer als 4 nicht analytisch darstellbar sind.

b) f sei eine „elementare" analytische Funktion, deren Inverse f^{-1} häufig benutzt und deshalb als „bekannt" betrachtet wird:

$$\begin{aligned} f(x) &= \tan x \,, \\ f^{-1}(x) &= \arctan x \,. \end{aligned}$$

Im allgemeinen liegt eine brauchbare analytische Darstellung der Lösungen der Gleichung (8.1) nicht vor. Die Lösung $\mathbf{x}^*$ wird dann iterativ berechnet. Eine Iterationsfolge $\{x^k : k \in \mathbb{N}_0\}$ wird definiert, so daß:

a) $x^k \longrightarrow x^*$ für $k \longrightarrow \infty$

b) x^k ist berechenbar.

Auf diese Weise läßt sich die Lösung der Gleichung (8.1) mit beliebiger Genauigkeit berechnen.

Iterationsfolgen werden häufig mit Hilfe von *Iterationsfunktionen* Φ_k berechnet. Sei $s_0 \in \mathbb{N}_0$ gegeben. Seien auch

$$x^0, x^1, \ldots, x^{s_0} \in G$$

gegeben, und

$$\begin{aligned} \Phi_k &: G^{k+1} \times \mathbb{N}_0^{k+1} \longrightarrow \mathbb{N}_0 \\ s_k &:= \Phi_k(x_0, \ldots, x_k \,,\, s_0, \ldots, s_{k-1}; f) \\ \Phi_k &: G^{s_{k+1}} \longrightarrow G \,. \end{aligned}$$

Dann ist

$$x^{k+1} := \Phi_k(x^k, \ldots, x^{k-s_k}; f) \,.$$

In den meisten Fällen ist $s_k = s$ und $\Phi_k = \Phi$ für alle k, so daß

$$x^{k+1} = \Phi(x^k, \ldots, x^{k-s_k}; f) \,. \tag{8.4}$$

In diesem Kapitel werden wir fast ausschließlich Iterationsfolgen betrachten, die auf der Gleichung (8.4) basieren.

Es entstehen mehrere Fragen:

1. Unter welchen Bedingungen ist die Folge $\{x^k\}$ definiert?

2. Wie findet man passende Anfangspunkte $x^0, \dots, x^{s_0}$?

3. Unter welchen Bedingungen konvergiert die Folge $\{x^k\}$?

4. Wie schnell konvergiert die Folge $\{x^k\}$?

5. Kann der Fehler $|x^k - x^*|$ abgeschätzt werden?

6. Wie verhält sich die Reihe $\{g\ell(x^k)\}$?

7. Wie findet man passende Iterationsfunktionen Φ_k ?

8. Welche Iterationsfunktionen Φ_k sind am besten?

Literatur

Byrne, G.D. und **Hall**, C.A.: Numerical Solution of Systems of nonlinear Algebraic Equations. New York. Academic Press, 1973.

Dennis, J.E. und **Schnabel**, R.B.: Numerical methods for unconstrained optimization and nonlinear equations. Prentice Hall, 1983.

Heitzinger, W., **Troch**, I., **Valentin**, G.: Praxis nichtlinearer Abbildungen. München: Hanser, 1984.

Householder, A.S.: The Numerical Treatment of a Single Nonlinear Equation. New York: McGraw Hill, 1970.

Ortega, J.M., **Rheinboldt**, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. New York, Academic Press, 1970.

Ostrowski, A.M.: Solution of Equations and Systems of Equations. New York, Academic Press, 1966.

Rall, L.: Computational Solution of Nonlinear Operator Equations. New York: Wiley, 1969.

Traub, J.F.: Iterative Methods for the Solution of Equations. Englewood Cliffs: Prentice-Hall, 1964.

Wait, R.: The Numerical Solution of Algebraic Equations. Chichester, Wiley, 1979.

8.2 Einige klassische Iterationsverfahren für $n = 1$

Es gibt eine Vielzahl von Iterationsverfahren für $n = 1$, wovon wir hier nur eine kleine beispielhafte Auswahl beschreiben können.

Als Grundbeispiel nehmen wir:

$$\begin{aligned} f_1 : (\frac{\pi}{2}, \frac{3\pi}{2}) \longrightarrow \mathbb{R}, \quad f_1(x) &:= x - \tan x \\ g_1 : (\frac{\pi}{2}, \frac{3\pi}{2}) \longrightarrow \mathbb{R}, \quad g_1(x) &:= \tan x \\ y_1 &:= 0 \\ F_1 &:= f_1 \end{aligned}$$

8.2.1 Fixpunkt-Iteration

Wird die Gleichung $F(x) = y$ in Fixpunktgestalt $x = g(x)$ umgeformt, dann liegt folgende Methode nahe:

Fixpunkt-Iteration
Wähle $x^0 \in G$. Berechne die Folge $\{x^n\}$ rekursiv mit Hilfe der Gleichung

$$x^{n+1} = g(x^n), \quad n \in \mathbb{N}_0 .$$

Die Abbildung g heißt *Iterationsfunktion.*

Fixpunkt-Iteration hat mehrere wichtige theoretische und praktische Anwendungen:

1. Viele bekannte Verfahren, wie etwa das Newton-Verfahren, sind spezielle Fälle der Fixpunkt-Iteration.

2. Fixpunkt-Iteration wird besonders oft bei der Lösung von linearen Gleichungssystemen angewandt.

3. Fixpunkt-Iteration ist eine wichtige Methode für Gleichungen in Banachräumen.

Beispiel: Die Iteration

$$x^{k+1} = \tan x^k$$

divergiert mit $x^0 = 2$. Die Iteration

$$x^{k+1} = \pi + \arctan x^k$$

konvergiert (siehe unten).

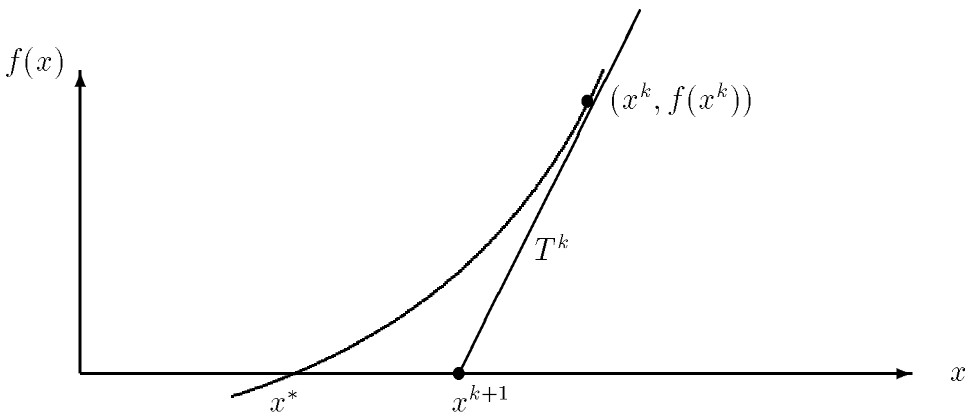


Abbildung 8.1: Das Newton-Verfahren

8.2.2 Das Newton-Verfahren

Das Newton-Verfahren läßt sich geometrisch erklären: Sei x^k eine Näherung für x^*. Im Punkt $(x^k, f(x^k))$ wird die Tangente T^k an die Kurve $y = f(x)$ konstruiert. Dann wird x^{k+1} als Schnittpunkt von T^k mit der x-Achse gegeben (siehe Abbildung 8.1) :

$$x^{k+1} = x^k - f(x^k)/f'(x^k) .$$

Das Newton-Verfahren läßt sich leicht mittels Iterationsfunktionen schreiben:

$$\begin{aligned} x^0 & \qquad \text{sei gegeben} \\ s_k & := s_0 := 0 , \quad \text{für} \quad k \geq 0 , \\ \Phi_k(x; f) & := x - f(x)/f'(x) . \end{aligned}$$

Über die Konvergenz des Newton-Verfahrens wird im nächsten Abschnitt diskutiert. Es läßt sich unmittelbar sehen, daß das Newton-Verfahren nicht immer konvergiert, wie dies aus 8.2 und 8.3 hervorgeht. In dem Beispiel von 8.2 ist $x^k = x^{k-2}$ für $k \geq 2$. In dem Beispiel von 8.3 ist x^1 nicht definiert.

Beispiel: $f(x) = x - \tan x$ (VIC-20-"NEWTON")

1. $x^0 = 2$

k	x^k	$f(x^k)$	$f'(x^k)$
0	2	4.18×10^0	-4.77×10^0
1	2.87	3.14×10^0	-7.36×10^{-2}
2	45.6	6.37×10^1	-3.27×10^2

Ergebnis: $\{x^k\}$ divergiert.

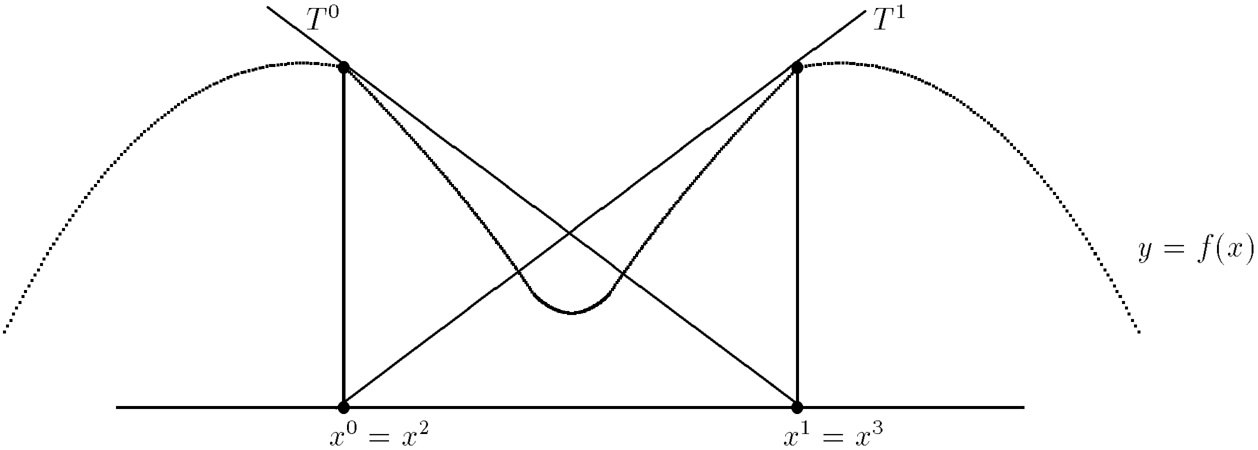


Abbildung 8.2: Das Newton-Verfahren $x^k - x^{k-2}$

2. $x^0 = 4$

k	x^k	$f(x^k)$	$f'(x^k)$
0	4.0	2.84×10^0	-1.34×10^0
1	6.12	6.28×10^0	-0.27×10^{-1}
2	238.4	2.38×10^2	-0.13×10^0

Ergebnis: $\{x^k\}$ divergiert.

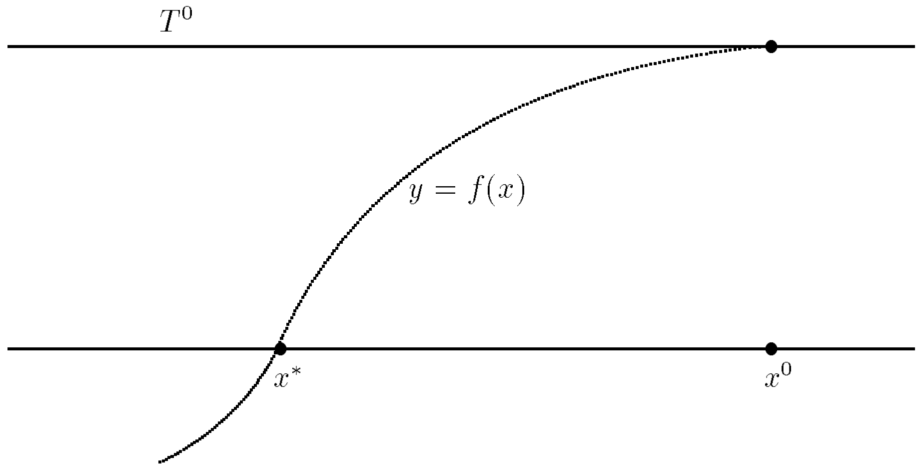


Abbildung 8.3: Das Newton-Verfahren im Fall x^1 nicht definiert

3. $x^0 = 4.5$

k	x^k	$f(x^k)$	$f'(x^k)$
0	4.5	-1.37×10^{-1}	-2.15×10^{1}
1	4.4936	-4.13×10^{-3}	-2.02×10^{1}
2	4.49340965	-3.97×10^{-6}	-2.01×10^{1}
3	4.49340945	$+1.07 \times 10^{-8}$	-2.01×10^{1}

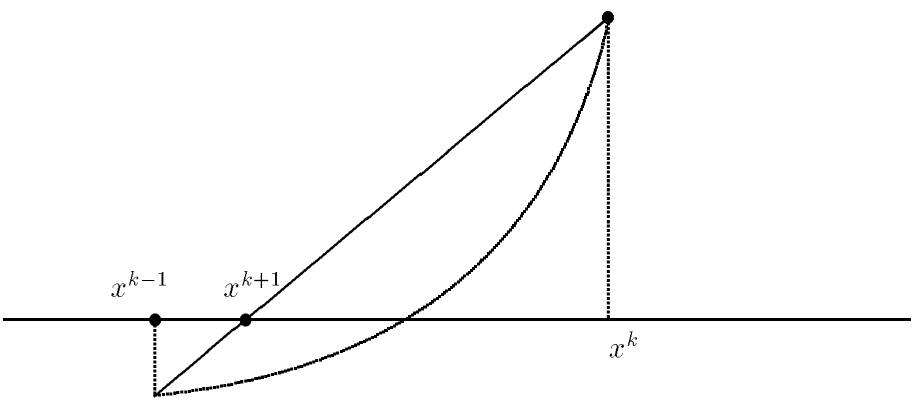
Ergebnis: $x^k \longrightarrow x^* = 4.49340945$

8.2.3 Die Sekantenmethode

x^0, x^1 sei gegeben.

$$\begin{aligned} s_k &= s_0 = 1 \\ x^{k+1} &= \Phi(x^k, x^{k-1}; f) := x^k - \frac{f(x^k)}{[x^k, x^{k-1}]f} = x^k - f(x^k) \cdot \frac{x^k - x^{k-1}}{f(x^k) - f(x^{k-1})} \, . \end{aligned}$$

Die Sekantenmethode läßt sich als Approximation zum Newton-Verfahren bezeichnen, wobei die Tangente an $(x^k, f(x^k))$ durch die Sekante zwischen $(x^k, f(x^k))$ und $(x^{k-1}, f(x^{k-1}))$ ersetzt wird.



Beispiel: $f(x) = x - \tan x$ (VIC-20-“SEKANT”)

1. $x^0 = 2,\ x^1 = 4.6$

k	x^{k-1}	x^k	$f(x^k)$	$\frac{f(x^k)-f(x^{k-1})}{x^k-x^{k-1}}$
1	2	4	2.84×10^0	-6.7×10^1
2	4	8.23×10^1	1.74×10^1	1.86×10^0
3	8.23	2.47×10^0	3.26×10^0	1.30×10^0
4	2.47×10^0	-3.01×10^{-1}	9.12×10^{-6}	1.30×10^1
5	-3.01×10^{-1}	-3.01×10^{-1}	9.12×10^{-6}	-9.08×10^{-4}
6	-3.01×10^{-1}	-2.00×10^{-1}	2.70×10^{-6}	-6.30×10^{-4}
37	-1.43×10^{-4}	-1.56×10^{-4}	0	-5.92×10^{-7}

Das Programm endet mit einer Fehler-Division durch Null.

Ergebnis: $x^k \longrightarrow 0$. Numerische Schwierigkeiten entstehen, wenn $|x^k - 0|$ klein ist.

2. $x^0 = 4, x^1 = 4.6$

k	x^k	x^{k-1}	$f(x^k)$	$\frac{f(x^k)-f(x^{k-1})}{x^k-x^{k-1}}$
1	4.0	4.6	-4.26×10^0	-1.18×10^1
2	4.6	4.24	$+2.28 \times 10^0$	-1.81×10^1
3	4.24	4.3656	$+1.59 \times 10^0$	-5.45×10^0
4	4.3656	4.6587	-1.39×10^1	-5.31×10^1
5	4.6587	4.3957	$+1.34 \times 10^0$	-5.82×10^1
6	4.3957	4.4187	$+1.11 \times 10^0$	-1.00×10^1
7	4.4187	4.5288	-8.58×10^{-1}	-1.78×10^1
8	4.5288	4.4808	2.38×10^{-1}	-2.28×10^1
9	4.4808	4.491323	$+4.17 \times 10^{-2}$	-1.88×10^1
10	4.491323	4.4935329	-2.49×10^{-3}	-2.00×10^1
11	4.4935329	4.49340824	2.45×10^{-5}	-2.02×10^1
12	4.49340824	4.49340945	1.07×10^{-8}	-2.01×10^1
13	Division durch Null			

Erfolg: $x^k \longrightarrow x^* \doteq 4.49340945$

8.2.4 Intervallschachtelung (Bisektion)

Sei $f : [a, b] \longrightarrow \mathbb{R}$ stetig mit $f(a) \cdot f(b) < 0$. Auf Grund des Mittelwertsatzes der Differentialrechnung hat f mindestens eine Nullstelle im Intervall $[a, b]$. Zur Berechnung einer Nullstelle dient folgender Algorithmus:

$$x_p^0 := \begin{cases} a & , \quad \text{falls} \quad f(a) > 0 \\ b & , \quad \text{sonst} \end{cases}$$

$$x_n^0 := \begin{cases} a & , \quad \text{falls} \quad f(a) < 0 \\ b & , \quad \text{sonst} \end{cases}$$

Für $k \geq 0$:

$$x_m^k := (x_p^k + x_n^k)/2 \,, \quad z^k := \text{signum}(f(x_m^k))$$

$$x_p^{k+1} := \begin{cases} x_m^k & , \text{ falls } \quad z^k \geq 0 \,, \\ x_p^k & , \text{ sonst} \end{cases}$$

$$x_n^{k+1} := \begin{cases} x_m^k & , \text{ falls } \quad z^k < 0 \,, \\ x_n^k & , \text{ sonst} \end{cases}$$

Dieser Algorithmus heißt *Intervallschachtelung* oder *Bisektion* oder *Intervallhalbierung.*

Intervallhalbierung läßt sich mittels Iterationsfunktionen schreiben:

$$x^0 := \begin{cases} a & , \text{ falls } \quad f(a) \geq 0 \\ b & , \text{ sonst} \end{cases}$$

$$x^1 := \begin{cases} a & , \text{ falls } \quad f(a) < 0 \\ b & , \text{ sonst} \end{cases}$$

$$\begin{aligned} x^{k+1} &:= \Phi_k(x^k, \ldots, x^{k-s_k}; q) \\ s_{2k} &= s_0 = 1 \,, \quad s_{2k+1} = s_1 = 2 \,, , \quad k \geq 0 \\ \Phi_{2k} &= \Phi_0 \,, \quad \Phi_{2k+1} = \Phi_1 \,, \quad k \geq 0 \,, \quad g(x) := f(x) + x \end{aligned}$$

$$\Phi_0(u, v; f) := \begin{cases} \frac{u+v}{2} & , \text{ falls } \quad f(\frac{u+v}{2}) \geq 0 \\ u & , \text{ sonst} \end{cases}$$

$$\Phi_1(u, v, w; f) := \begin{cases} \frac{v+w}{2} & , \text{ falls } \quad g(\frac{v+w}{2}) < \frac{v+w}{2} \\ v & , \text{ sonst} \end{cases}$$

Beispiel: (VIC 20 “INTERV-HALB”)

$$\begin{aligned} f(x) &= x - \tan x \\ a &= 2 \qquad b = 4.6 \\ f(a) &\doteq 4.18 \qquad f(b) \doteq -4.26 \end{aligned}$$

k	x_p^k	x_n^k	x_m^k	$f(x_m^k)$	z^k
0	2	4.6	3.3	$+3.14 \times 10^0$	$+1$
1	3.3	4.6	3.95	$+2.90 \times 10^0$	$+1$
2	3.95	4.6	4.275	$+2.13 \times 10^0$	$+1$
3	4.275	4.6	4.4375	$+8.91 \times 10^{-1}$	$+1$
4	4.4375	4.6	4.51875	-5.80×10^{-1}	-1
5	4.4375	4.51875	4.47812	$+2.87 \times 10^{-1}$	$+1$
10	4.493359	4.49589	4.49462	-2.47×10^{-2}	-1
20	4.493408	4.493411	4.493410	-1.51×10^{-5}	-1

Für $k = 100$:

$$\begin{aligned} x_p^{100} &= 4.49340945 \qquad x_n^{100} = 4.49340946 \\ x_m^{100} &= 4.49340946 \qquad f(x_m^{100}) = -1.72294616 \times 10^{-10} \end{aligned}$$

und keine weitere Verbesserung ist möglich.

8.2.5 Andere bekannte Verfahren

Regula falsi: Es wird angenommen, daß $f(x^0) \cdot f(y^0) < 0$. Dann:

$$x^{k+1} := x^k - \frac{x^k - y^k}{f(x^k) - f(y^k)} \cdot f(x^k) \, ,$$

$$y^{k+1} := \begin{cases} x^k & , \text{ falls } \; f(x^k) \cdot f(x^{k+1}) < 0 \\ y^k & , \text{ falls } \; f(x^k) \cdot f(x^{k+1}) > 0 \end{cases}$$

Eine Lösung liegt zwischen x^k und y^k für alle k.

Halley (1694)

$$x^{k+1} = x^k - \frac{2f(x^k)f'(x^k)}{2f'(x^k) \cdot f'(x^k) - f(x^k)f''(x^k)}$$

Muller

$$\begin{aligned} x^{k+1} &= x^k - \frac{2f(x^k)}{\gamma_k \mp (\gamma_k^2 - 4f(x^k)g_k)^{1/2}} \\ g_k &:= [x^k, x^{k-1}, x^{k-2}]f \\ &= \frac{[x^k, x^{k-1}]f - [x^{k-1}, x^{k-2}]f}{x^k - x^{k-2}} \\ \gamma_k &:= [x^k, x^{k-1}]f + (x^k - x^{k-1})g_k \end{aligned}$$

8.3 Konvergenz von Iterationsfolgen: Der Banachsche Fixpunktsatz

Einführung: Dieser Abschnitt ist grundsätzlichen Konvergenzbeweisen gewidmet.

Wir betrachten den Fall $s_k = 0$, $\Phi_k = \Phi$, für alle k, also Iterationen folgender Gestalt: $x^0 \in \mathbb{R}^n$ sei gegeben,

$$x^{k+1} := \Phi(x^k; f) \, , \; k \geq 0 \, .$$

Definition 8.1 *Sei $\|\cdot\|$ eine Norm auf dem $\mathbb{R}^n$. $g : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$ heißt* kontrahierend in D *bzgl. $\|\cdot\|$, falls gilt: Es gibt $q \in [0,1)$, so daß*

$$\|g(x) - g(y)\| \leq q\|x - y\| \, , \; \forall x, y \in D \, .$$

Bemerkung: Eine Abbildung $g : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ *erfüllt eine Lipschitz Bedingung mit Konstante L*, falls

$$\|g(x) - g(y)\| \leq L\|x - y\| \, , \; \forall x, y \in D \, .$$

Die Abbildung $g : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ ist deshalb kontrahierend genau dann, wenn g eine Lipschitz Bedingung mit Konstante $L < 1$ erfüllt.

Satz 8.1 (Kontraktionssatz, Fixpunktsatz von Banach) *Sei $D \subseteq \mathbb{R}^n$ abgeschlossen und $g : D \longrightarrow D$ kontrahierend in D. Sei $x^0 \in D$ beliebig. Dann konvergiert die Folge*

$$x^{k+1} := g(x^k)\ ,\ k \in \mathbb{N}_0$$

gegen den eindeutig bestimmten Fixpunkt x^* von g in D, und es gilt:

$$\|x^* - x^k\| \leq \frac{q}{1-q}\|x^k - x^{k-1}\|\ .$$

Beweis: Es gilt für $k \geq 1$:

$$\|x^{k+1} - x^k\| = \|g(x^k) - g(x^{k-1})\| \leq q\|x^k - x^{k-1}\| \leq q^k\|x^1 - x^0\|$$

und damit für $j > 1$:

$$\begin{aligned} \|x^j - x^\ell\| &= \|\sum_{k=\ell}^{j-1}(x^{k+1} - x^k)\| \leq \sum_{k=\ell}^{j-1} q^k\|x^1 - x^0\| \\ &\leq q^\ell\frac{1}{1-q}\|x^1 - x^0\| \longrightarrow 0\ , \quad \text{für} \quad k \to \infty\ . \end{aligned} \tag{8.5}$$

Die x^k bilden damit eine Cauchy-Folge.

Sei x^* der Grenzwert der Folge $\{x^k\}$. Da D abgeschlossen ist und $x^k \in D$ für alle D, ist $x^* \in D$. Es gilt:

$$g(x^*) \longleftarrow g(x^k) = x^{k+1} \longrightarrow x^*\ , \quad \text{für} \quad k \to \infty\ ,$$

so daß

$$g(x^*) = x^*\ .$$

Bildet man in (8.5) den Grenzwert für $j \to \infty$, erhält man:

$$\|x^* - x^\ell\| \leq \frac{q^\ell}{1-q}\|x^1 - x^0\|\ .$$

Für $\ell = 1$ folgt

$$\|x^* - x^1\| \leq \frac{q}{1-q}\|x^1 - x^0\|\ .$$

Dies ist die behauptete Abschätzung für $k = 1$. Setzt man $x^{k-1} = x^0$, so folgt die Behauptung für beliebiges k.

Zu zeigen bleibt noch die Eindeutigkeit von x^*. Seien x^*, x^{**} Fixpunkte von g. Dann gilt:

$$\|x^* - x^{**}\| = \|g(x^*) - g(x^{**})\| \leq q\|x^* - x^{**}\|\ .$$

Da $q < 1$, folgt $\|x^* - x^{**}\| = 0$, d.h. $x^* = x^{**}$. □

Jetzt werden einige einfache aber nützliche Bedingungen hergeleitet, die dafür ausreichend sind, daß eine Abbildung g kontrahierend ist.

Hilfssatz 8.1 *Seien $D = [a, b]$, $g : D \longrightarrow D$. g ist kontrahierend, falls g stetig differenzierbar ist und*

$$q := \max_{x \in D} |g'(x)| < 1 \ .$$

Beweis: Nach dem Mittelwertsatz der Differentialrechnung gilt: Es gibt $\Theta \in [0, 1]$:

$$|g(x) - g(y)| = |g'(\Theta x + (1 - \Theta)y)(x - y)| \leq q|x - y| \ .$$

□

Wir wollen jetzt Hilfssatz 8.1 für den Fall $n > 1$ formulieren.

Definition 8.2 *Seien $U \subset \mathbb{R}^n$ eine offene Teilmenge des $\mathbb{R}^n$ und $g : U \longrightarrow \mathbb{R}^n$.$g$ heißt* stetig differenzierbar auf U, *wenn g stetig ist und die partiellen Ableitungen*

$$\frac{\partial g_i(x)}{\partial x_j} \ , \ 1 \leq i \ , \ j \leq n \ , \ x \in U$$

existieren und stetig sind.

Definition 8.3 *Sei D eine abgeschlossene Teilmenge des $\mathbb{R}^n$ und $f : D \longrightarrow \mathbb{R}^n$. Sei $U \subset D$ mit U offen und $\overline{U} = D$. Sei $g : U \longrightarrow \mathbb{R}^n$, $g := f|_U$.*
f heißt stetig differenzierbar auf D, *falls:*

1. *f sei stetig*

2. *g sei stetig differenzierbar*

3. *Es existiere die stetige Abbildung $f_{ij} : D \longrightarrow \mathbb{R}$ mit*

$$f_{ij}|_U = \frac{\partial g_i}{\partial x_j} \ .$$

Die n^2 „Ableitungen" $f_{ij}(x)$ bilden die Funktionalmatrix

$$f'(x) := Df(x) := \begin{pmatrix} f_{11}(x) & \cdots & f_{1n}(x) \\ \vdots & \ddots & \\ f_{n1}(x) & \cdots & f_{nn}(x) \end{pmatrix}$$
$$= \left(\frac{\partial f_i(x)}{\partial x_j} \right) \ ,$$

deren Determinante die Funktionaldeterminante *(englisch: Jacobian) heißt.*

Definition 8.4 *Sei $\| \cdot \|$ eine Norm auf dem $\mathbb{R}^n$ und $A : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ eine lineare Abbildung. Die entsprechende verträgliche Norm $\|A\|$ wird wie folgt definiert:*

$$\|A\| := \sup_{\substack{x \in \mathbb{R}^n \\ x \neq 0}} \frac{\|Ax\|}{\|x\|} \ .$$

Hilfssatz 8.2 *$g : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ sei stetig differenzierbar auf einer konvexen abgeschlossenen Menge D. Für alle $x, y \in D$ gilt:*

$$\|g(y) - g(x)\| \leq \sup_{0 \leq t \leq 1} \|g'(x + t(y-x))\| \, \|x - y\|. \tag{8.6}$$

Beweis: Sei

$$M := \sup_{0 \leq t \leq 1} \|g'(x + t(y-x))\| \; .$$

Falls $M = \infty$ oder $x = y$, dann gilt (8.6).
Falls $M < \infty$ und $x \neq y$, betrachte, für $\epsilon > 0$ und $x, y \in D$,

$$\gamma := \sup\{t \in [0,1] : \|g((1-t)x + ty) - g(x)\| \leq (M + \epsilon)t\|y - x\|\} \; .$$

Da g stetig ist, folgt

$$\|g((1-\gamma)x + \gamma y) - g(x)\| \leq (M + \epsilon)\gamma\|y - x\| \; .$$

Es gilt: $\gamma = 1$. Wäre nämlich $\gamma \neq 1$, dann würde $\gamma \in [0, 1)$ gelten. Seien $1 \leq i \leq n$ und $\phi_i : [0,1] \longrightarrow \mathbb{R}$

$$\phi_i(t) := g_i(x + t(y-x)) \; .$$

Dann ist ϕ_i im Punkt $t = \gamma$ differenzierbar und nach der Kettenregel

$$\lim_{\delta \to 0} \frac{\phi_i(\gamma + \delta) - \phi_i(\gamma)}{\delta} = \phi_i'(\gamma) = \sum_{j=1}^{n} \frac{\partial g_i}{\partial x_j} (x + \gamma(y-x)) \cdot (y-x)_j \; .$$

Es folgt

$$\lim_{\delta \to 0} \frac{g(x + (\gamma+\delta)(y-x)) - g(x + \gamma(y-x))}{\delta} = g'(x + \gamma(y-x)) \cdot (y-x)$$

und

$$\begin{aligned}
& \lim_{\delta \to 0} \frac{\|g(x + (\gamma+\delta)(y-x)) - g(x + \gamma(y-x))\|}{\delta} \\
= \; & \|g'(x + \gamma(y-x)) \cdot (y-x)\| \\
\leq \; & \|g'(x + \gamma(y-x))\| \cdot \|y - x\| \\
\leq \; & M\|y - x\|.
\end{aligned}$$

Folglich gibt es $\delta > 0$ mit

1. $\delta < (1-\gamma)/2$
2. $\|g(x + (\gamma+\delta)(y-x)) - g(x + \gamma(y-x))\| \leq (M + \epsilon)\delta\|y - x\|$,

d.h. $1 > \beta := \gamma + \delta > \gamma$ und

$$\begin{aligned}\|g(x+\beta(y-x)) - g(x)\| &\leq \|g(x+\beta(y-x)) - g(x+\gamma(y-x))\| + \|g(x+\gamma(y-x)) - g(x)\| \\ &\leq (M+\epsilon)\delta\|y-x\| + (M+\epsilon)\gamma\|y-x\| \\ &\leq (M+\epsilon)\beta\|y-x\| ,\end{aligned}$$

was der Definition von γ widerspricht.

Da $\gamma = 1$ ist, gilt für beliebiges $\epsilon > 0$

$$\|g(y) - g(x)\| \leq (M+\epsilon)\|y-x\| .$$

$\square$

Folgerung Sei $D \subset \mathbb{R}^n$ eine konvexe abgeschlossene Menge. Sei $g : D \longrightarrow \mathbb{R}^n$ stetig differenzierbar. Sei

$$g'(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} \end{pmatrix} .$$

Sei $\sup \|g'(z)\| \leq q < 1$, wobei $\|g'(z)\|$ die verträgliche Norm zu $\|\cdot\|$ ist,

$$\|g'(z)\| = \sup_{v \neq 0} \frac{\|g'(z) \cdot v\|}{\|v\|} .$$

Dann ist g kontrahierend in D mit der Konstante $q < 1$.

Beispiel: $f(x) = x - \tan x$, $D = [\pi, \frac{3}{2}\,\pi]$. Setze $g(x) := \tan x$. Dann ist

$$g'(x) = \frac{1}{\cos^2 x} \geq 1 \quad \text{für} \quad x \in D .$$

Folglich ist g nicht kontrahierend, und der Fixpunkt $x^* \in D$ kann möglicherweise nicht durch die einfache Iteration $x^{k+1} = g(x^k)$ berechnet werden.

Dieses Beispiel verdeutlicht eine Hauptschwierigkeit bei der Anwendung des Kontraktionssatzes: Man muß das Problem mit Hilfe einer geeigneten kontrahierenden Funktion formulieren.

Beispiel: $f(x) = x - \tan x$, $D = [\pi, \frac{3}{2}\,\pi]$.
Aus der Gleichung $f(x) = 0$ folgt

$$\arctan x = x + k\pi \ , \ k \in \mathbb{Z} .$$

Da $\arctan : (-\infty, +\infty) \longrightarrow (-\frac{\pi}{2}, \frac{\pi}{2})$ und wir eine Lösung $x^* \in D$ suchen, wählen wir $k = -1$ und setzen

$$g(x) := \pi + \arctan x .$$

Offenbar gilt: $g(D) \subseteq D$ und

$$\max_{x\in D} |g'(x)| = \frac{1}{1+\pi^2} \doteq 0,092 < 1$$

g ist also kontrahierend in D.

Nachdem $x^* \doteq 4.49340945$, ist $g'(x^*) \doteq \cdot 04719$. Wie schnell x^k gegen x^* konvergiert, entnimmt man aus Tabelle 8.1.

k	x^k	$\frac{q}{1-q}\lvert x^k - x^{k-1}\rvert$	$x^* - x^k$	$\frac{x^*-x^k}{x^*-x^{k-1}}$
0	3.14159265	-	-	-
1	4.40421991	0.1351	0.0892	-
2	4.48911945	0.008918	0.0043	· 0672
3	4.49320683	0.0004280	0.0002	· 0481
4	4.4933999	-	-	· 0472

Tabelle 8.1: Einfache Iteration (VIC-20-"KONTRAK")

Satz 8.2 (Lokaler Konvergenzsatz) *Sei $g : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ mit $g(x^*) = x^*$. Ist g in einer Umgebung von x^* stetig differenzierbar und $\|g'(x^*)\| < 1$, dann gibt es eine Umgebung U von x^*, so daß das Iterationsverfahren*

$$x^{k+1} = g(x^k)\ ,\ x^0 \in U\ ,\ k \in \mathbb{N}_0$$

gegen x^* konvergiert.

Beweis: Es gibt eine Kugel V mit Radius r um x^* und ein $q \in [0,1)$, so daß

$$\|g'(x)\| \le q < 1 \quad \text{für} \quad x \in V\ .$$

Sei $x \in V$. Dann:

$$\|g(x) - x^*\| = \|g(x) - g(x^*)\| \le q\|x - x^*\| \le r\ .$$

Folglich ist $g(x) \in V$. Damit ist g kontrahierend in V, und es gilt $g(V) \subseteq V$. Mit dem Fixpunktsatz von Banach folgt dann die Behauptung. □

Der folgende Satz ist naheliegend:

Satz 8.3 (Intervallschachtelung) *Sei $f : [a,b] \longrightarrow \mathbb{R}$ stetig mit $f(a)\cdot f(b) < 0$. Sei $\{x^k\}$ durch Intervallhalbierung definiert mit $\{x^0, x^1\} = \{a, b\}$. Dann strebt x^k gegen eine Nullstelle $x^* \in [a,b]$ von f und*

$$|x_m^k - x^*| \le (b-a)/2^{k+1}\ .$$

8.4 Konvergenzordnung

In diesem Abschnitt führen wir ein Maß ein, womit wir die Konvergenzgeschwindigkeit verschiedener Iterationsmethoden vergleichen können.

Definition 8.5 *$\{x^k\}$ sei eine Folge in $\mathbb{R}^n$, $x^k \longrightarrow x^*$. Sei p eine reelle Zahl und $C \neq 0$ eine Konstante. Es gelte*

$$\frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^p} \longrightarrow C\,, \quad \textit{für} \quad k \to \infty\,.$$

Dann heißt Ord($\{x^k\}$) $:= p$ die Ordnung *der Folge $\{x^k\}$ und C die* asymptotische Fehlerkonstante.

Definition 8.6 *Sei φ eine Iterationsfunktion für die Gleichung $f(x) = 0$ mit Nullstelle x^*,*

$$x^{k+1} := \varphi(x^k, \dots, x^{k-s}; f)\,.$$

Sei

$$\mathit{Ord}(\varphi) := \inf_f \inf_{\{x^k\}\ \text{konvergent}} \mathit{Ord}(\{x^k\})\,.$$

Ord(φ) heißt die Ordnung der Iterationsfunktion φ *(oder* Ordnung *der entsprechenden Methode).*

In einigen Fällen werden spezielle Ausdrücke benutzt (siehe Tabelle 8.2).

p	Bezeichnung
1	lineare Konvergenz
2	quadratische Konvergenz
> 1	superlineare Konvergenz

Tabelle 8.2: Einige spezielle Konvergenzbezeichnungen

Es ist natürlich möglich, daß die Folge $\{x^k\}$ konvergent, aber daß Ord($\{x^k\}$) nicht definiert ist.

Ortega und Rheinboldt, Seite 281ff, geben eine Verallgemeinerung des Begriffs Ordnung:

Definition 8.7 *$\{x^k\}$ sei eine konvergente Folge mit Grenzwert x^*.*

$$Q_p\{x^k\} := \begin{cases} 0 & , \textit{ falls } \quad x^k = x^* \quad \textit{für alle} \quad k \quad \textit{außer endlich vielen} \\ \limsup\limits_{k\to\infty} \frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|^p} & , \textit{ falls } \quad x^k \neq x^* \quad \textit{für alle} \quad k \quad \textit{außer endlich vielen} \\ \infty & , \textit{ sonst} \end{cases}$$

Definition 8.8 $C(\varphi, x^*)$ *sei die Menge aller konvergenten Folgen mit Grenzwert* x^*, *die durch* φ *erzeugt sind. Dann:*

$$Q_p(\varphi, x^*) := \sup\{Q_p\{x^k\} : \{x^k\} \in C(\varphi, x^*)\} .$$

Satz 8.4 *Genau eine der folgenden Möglichkeiten gilt:*

a) $Q_p(\varphi, x^*) = 0 \ , \ \forall p \in [1, \infty]$,

b) $Q_p(\varphi, x^*) = \infty \ , \ \forall p \in [1, \infty]$,

c) Es gibt $p_0 \in [1, \infty]$:

$$\begin{aligned} Q_p(\varphi, x^*) &= 0 \ , \ p \in [1, p_0) \\ Q_p\{\varphi, x^*\} &= \infty \ , \ \textit{für} \quad p \in (p_0, \infty) \ . \end{aligned}$$

Bemerkung: Wenn Ord(φ) definiert ist, dann gilt:

$$\mathrm{Ord}(\varphi) = \inf\{p | Q_p(\varphi, x^*) = \infty\}$$

Beispiel:[Das Newton-Verfahren $\varphi(x) := x - f(x)/f'(x)$] Sei $f : G \subset \mathbb{R} \longrightarrow \mathbb{R}$ mit einer Nullstelle x^*. Sei x^* eine einfache Nullstelle von f, d.h. $f'(x^*) \neq 0$. Sei $\{x^k\}$ die Iterationsfolge, die durch φ mit Startwert x^0 erzeugt wird. Konvergiere $\{x^k\}$ gegen x^*. Dann gilt:

$$\begin{aligned} x^k \longrightarrow x^* \qquad & \text{für} \quad k \to \infty \\ \varphi(x^*) &= x^* \\ \varphi'(x^*) &= 1 - \frac{f'(x^*)}{f'(x^*)} + \frac{f(x^*)}{[f'(x^*)]^2} \cdot f''(x^*) \\ \varphi''(x^*) &= f''(x^*)/f'(x^*) \end{aligned}$$

Aus $x^{k+1} = \varphi(x^k)$ und $x^* = \varphi(x^*)$ folgt für $\epsilon_k := x^k - x^*$

$$\epsilon_{k+1} = \frac{1}{2}\, \varphi''(\Theta_k)\epsilon_k^2 \ , \ \Theta_k \in [x^k, x^*] \ .$$

Es folgt

$$\mathrm{Ord}(\varphi) = 2, \quad C = \frac{1}{2}\, \frac{f''(x^*)}{f'(x^*)} \ .$$

Beispiel:[Sekantenmethode] Die Ordnung der Sekantenmethode ist $p = \frac{\sqrt{5}+1}{2} \doteq 1,62$. Ein rigoroser Beweis erfordert erheblichen Aufwand (s. Ostrowski, S. 30). Wir geben deshalb einen Beweis, der an einigen Stellen heuristisch ist. Sei

$$x^{k+1} := x^k - f(x^k) \cdot \frac{x^k - x^{k-1}}{f(x^k) - f(x^{k-1})} \tag{8.7}$$

$$\epsilon_k := x^k - x^* \tag{8.8}$$

Es gilt:

$$\begin{aligned} f(x^k) &= f(x^*) + \epsilon_k f'(x^*) + \frac{\epsilon_k^2}{2} f''(x^*) + \ldots \, , \\ f(x^{k-1}) &= f(x^*) + \epsilon_{k-1} f'(x^*) + \frac{\epsilon_{k-1}^2}{2} f''(x^*) + \ldots \end{aligned} \tag{8.9}$$

Nun folgt aus (8.7), (8.8) und (8.9):

$$\begin{aligned} \epsilon_{k+1} &\doteq \epsilon_k - \frac{(\epsilon_k - \epsilon_{k-1})[\epsilon_k f' + \frac{1}{2}\,\epsilon_k^2 f'']}{(\epsilon_k - \epsilon_{k-1})[f' + \frac{1}{2}(\epsilon_k - \epsilon_{k-1}) f'']} \\ &\doteq \epsilon_k - \frac{\epsilon_k}{f'}[f' + \frac{1}{2}\,\epsilon_k f''][1 - \frac{1}{2}(\epsilon_k + \epsilon_{k-1}) f''/f'] \; , \; (\frac{1}{1-\alpha} = 1 + \alpha + O(\alpha^2)) \\ &\doteq \epsilon_k \epsilon_{k-1} \, \frac{f''(x^*)}{2 f'(x^*)} \\ &= \epsilon_k \epsilon_{k-1} C \end{aligned}$$

Setze $\eta_k := \ell n \; \epsilon_k$, dann folgt:

$$\eta_{k+1} = \eta_k + \eta_{k-1} + \ell n \; C \; .$$

Diese Differenzengleichung hat die Lösung:

$$\eta_k = -\ell n \; C + A s_1^k + B s_2^k \; ,$$

wobei s_1 und s_2 die Nullstellen des charakteristischen Polynoms sind:

$$\begin{aligned} s^2 &= s + 1 \\ s_1 &= \frac{+1 + \sqrt{5}}{2} \; , \; s_2 = \frac{1 - \sqrt{5}}{2} \end{aligned}$$

Für $k \to \infty$ gilt i.A. $(A \neq 0)$.

$$\begin{aligned} \eta_k &\sim A s_1^k \\ \epsilon_k &\sim e^{A s_1^k} \end{aligned}$$

Aus

$$\frac{\epsilon_{k+1}}{\epsilon_k^p} \longrightarrow \text{const}$$

folgt

$$\eta_{k+1} - p \; \eta_k \longrightarrow \ell n(\text{const.})$$

oder

$$A s_1^{k+1} - p \; A s_1^k \longrightarrow \ell n(\text{const.})$$

oder

$$p = s_1 = \frac{\sqrt{5} + 1}{2} \; .$$

8.5 Vergleich von Algorithmen

Einführung: Wir betrachten eine Iterationsfunktion φ mit der Ordnung p :

$$\begin{aligned} \epsilon_{k+1} &\doteq K\epsilon_k^p \\ \epsilon_k &:= x^k - x^* \end{aligned}$$

und möchten feststellen, wieviel Iterationen nötig sind, um den Fehler um einen Faktor e^m zu verringern. Wir suchen also k :

$$|\epsilon_k| \leq |\epsilon_0|e^{-m} \ .$$

Wir setzen $s_k := \ell n|\epsilon_k|$. Es folgt: $s_{k+1} = ps_k + \ell n|K|$, so daß

$$s_k = \begin{cases} s_0 + k\ \ell n|K| & ,\ \text{falls}\quad p = 1 \\ p^k\left[s_0 + \frac{\ell n|K|}{p-1}\right] - \frac{\ell n|K|}{p-1} & ,\ \text{falls}\quad p > 1 \end{cases}$$

Folgerungen

1. Für $p = 1$ konvergiert die Folge $\{x^k\}$ nur, wenn $\ell n|K| < 0$, d.h. $|K| < 1$.
   Für $p = 1$ dagegen konvergiert die Folge für jedes K, vorausgesetzt, daß $s_0 = \ell n|\epsilon_0|$ genügend klein ist:
   $$s_0 + \frac{\ell n|K|}{p-1} < 0 \ .$$

2. Ein Vergleich des Newton-Verfahrens und der Sekantenmethode ist besonders leicht, da:
   $$\begin{aligned} |K_N| &= \left|\frac{f''(x^*)}{2f'(x^*)}\right| \ ,\ p_N = 2 \\ |K_S| &= \left|\frac{f''(x^*)}{2f'(x^*)}\right|^{p_S-1} \ ,\ p_S = \frac{\sqrt{5}+1}{2} \ . \end{aligned}$$
   Ist $s_0^N = s_0^S$, dann ist $s_k^N = s_k^S$, wenn $p_N^{k_N} = 2^{k_N} = p_S^{k_S}$, d.h.
   $$k_S = k_N\ \ell n\ 2/\ell n\ p_S = 1,4404k_N.$$
   Es folgt, daß die Sekantenmethode effizienter ist als das Newton-Verfahren, falls der Rechenaufwand für die Berechnung von f' mehr als 0,44-mal größer ist als der Rechenaufwand für die Berechnung von f.

3. Im allgemeinen ist ein direkter Vergleich von Algorithmen nicht leicht. Für großes k gilt aber
   $$s_k \sim \text{const}p^k \ .$$

Der Rechenaufwand von zwei Methoden a und b läßt sich grob vergleichen:

$$\frac{R_a}{R_b} \sim \frac{\Theta_a/\ell n\ p_a}{\Theta_b/\ell n\ p_b}\ ,$$

wobei

$$\begin{aligned} R_a, R_b &= \text{Totaler Rechenaufwand von Methode a bzw. b} \\ \Theta_a, \Theta_b &= \text{Rechenaufwand pro Schritt für Methode a bzw. b} \end{aligned}$$

Es gibt mehrere Maßstäbe der Effizienz. Ein solcher Maßstab, der von Ostrowski vorgeschlagen wurde, ist

$$EFF := p^{1/d}\ ,$$

wobei

EFF = Effizienzfaktor
d = Anzahl der Ableitungen von f, die benutzt werden.

Bei der Definition von EFF wird angenommen, daß die Auswertung jeder Ableitung $f^{(j)}$ den gleichen Rechenaufwand erfordert.

Es gibt natürlich viel mehr Faktoren, die berücksichtigt werden müssen:

(a) Ob die Ableitungen $f^{(j)}$ vorhanden oder überhaupt berechenbar sind.

(b) Unter welchen Bedingungen ein Verfahren konvergiert.

(c) Ob Rundungsfehler Schwierigkeiten verursachen können.

Kronsjö [1987, S. 61] hat eine gute Zusammenfassung dieses Themas.

Literatur

Kronsjö, L.: Algorithms: Their Complexity and Efficiency. Chichester: John Wiley, 1987.

8.6 Theorie von Iterationsfunktion höherer Ordnung

In diesem Abschnitt werden einige Ergebnisse aus der allgemeinen Theorie von Iterationsfunktionen zitiert.

8.6.1 Iterationsfunktionen ohne Gedächtnis

Satz 8.5 (Schröder 1870) *$\varphi(x)$ sei eine Iterationsfunktion für die Funktion f, $p \in \mathbb{N}$. $\varphi^{(p)}$ sei stetig in einer Umgebung von x^* , $e_k := x^k - x^*$. Dann gilt: φ ist von der Ordnung p genau dann, wenn*

$$\begin{aligned} \varphi(x^*) &= x^* \ , \ \varphi^{(j)}(x^*) = 0 \ , \ 1 \le j \le p-1 \\ \varphi^{(p)}(x^*) &\neq 0 \end{aligned}$$

Weiter gilt:

$$\frac{e_{k+1}}{e_k^p} \longrightarrow \frac{\varphi^{(p)}(x^*)}{p!} \ .$$

Satz 8.6 (Schröder) *Sei $s \in \mathbb{N}$, $f(x^*) = 0$, $f'(x^*) \neq 0$, $f \in C^s$, $s \ge 1$. Es existiert eine Iterationsfunktion φ, wobei:*

1. *$\varphi(x)$ ist eine Funktion von x , $f(x), \ldots, f^{(s-1)}(x)$.*
2. *φ ist von der Ordnung s.*

Beweis: Nachdem $f'(x^*) \neq 0$, ist $f'(x) \neq 0$ für x in einer Umgebung von x^*. Wir betrachten die Gleichung $f(x) = y$. Es existiert die inverse Funktion $g \in C^s$, $x = g(y)$.

Sei jetzt $x = x^k$ eine Approximation zu x^*. Wir setzen

$$x^{k+1} := \sum_{j=0}^{s-1} g^{(j)}(y) \frac{(0-y)^j}{j!} \ , \ y = f(x^k) \ .$$

Es folgt

$$\begin{aligned} x^{k+1} - x^* &= g^{(s)}(\tilde{y}) \cdot y^s \frac{(-1)^s}{s!} \ , \tilde{y} \in [0, y] \ , \\ &= g^{(s)}(\tilde{y}) \cdot \frac{(f(x^k))^s}{s!} \ (-1)^s \ , \\ &= \frac{1}{s!} \ g^{(s)}(\tilde{y})[(x^k - x^*)f'(\tilde{x}^*)]^s \ , \ \tilde{x} \in [x^k, x^*] \ , \\ &= C(x^k - x^*)^s + 0(x^k - x^*)^s \ , \\ &= C\epsilon_k^s + 0(\epsilon_k^s) \qquad \text{mit} \\ C &:= \frac{g^{(s)}(0)}{s!} \ [f'(x^*)]^s (-1)^s \ . \end{aligned}$$

Berechnung von $g^{(j)}(y)$:

$$\begin{aligned} & x = g(y) \ , \\ & y = f(x) \ , \\ & g(f(x)) \equiv x \ , \\ \Rightarrow \ & g'(y) \cdot f'(x) = 1 \ , \\ \Rightarrow \ & g''(y) \cdot (f'(x))^2 + g'(y) f''(x) = 0 \quad \text{usw.} \end{aligned}$$

Folglich

$$\begin{aligned} g' &= 1/f' , \\ g'' &= -f''/(f')^3 \quad \text{usw.} \end{aligned}$$

□

Beispiel:$[s = 2]$ $x^{k+1} = g(y) - yg'(y) = x^k - f(x^k)/f'(x^k)$.
Wir erhalten das Newton Verfahren.

Satz 8.7 (Traub, Satz 5-3, Seite 98) *$\varphi(x)$ sei eine Iterationsfunktion der Ordnung s. Dann muß φ von den ersten $s-1$ Ableitungen von f abhängig sein.*

8.6.2 Iterationsfunktionen mit Gedächtnis

Es folgt aus den Sätzen 2 und 3 des vorherigen Abschnitts, daß eine Iterationsfunktion $\varphi(x)$, die die Werte $f(x), \ldots, f^{(s-1)}(x)$ benutzt, die Ordnung s erreichen, aber nicht überschreiten kann. Es liegt deshalb nahe, die vorherigen Approximationswerte zu benutzen. Z.B. die Sekantenmethode benutzt die Werte $f(x^k)$ und $f(x^{k-1})$ bei der Berechnung von x^{k+1}.

Satz 8.8 *Sei $f(x)$ eine* glatte *Funktion mit einer einfachen Nullstelle x^*.*

$$x^{k+1} := \varphi_k(x^k, \ldots, x^0, f) ,$$

wobei φ_k die Funktionswerte $f^{(j)}(x^\ell)$, $0 \le \ell \le k$, $0 \le j \le s$ benutzt. Dann ist die Ordnung von $\{x^k\}$ nicht größer als $s + 2$.

Beweis: Siehe Brent, Winograd und Wolfe: Optimal iterative processes for root-finding. Numer. Math. 20(1973) 327-341. □

Insbesondere für $s = 0$ ist die Maximalordnung $0 + 2 = 2$. Es folgt, daß die Sekantenmethode mit $p = 1,62$ eine einfache Methode ist, für die die Ordnung nicht weit unter der möglichen Maximalordnung liegt.

Siehe auch Kronsjö [1987, S. 72].

Literatur

Brent, R., **Winograd**, S., **Wolfe**, P.: Optimal Iterative Processes for Root-finding. Numer. Math. 20(1973) 327-341.

Kung, H.T. und **Traub**, J.F.: Optimal order and efficiency for iterations with two evaluations. SIAM J. Numer. Anal. 13(1976) 84-95.

Smale, S.: On the efficiency of algorithms of analysis. Bulletin AMS 13(1985) 87-121.

Smale, S.: The Fundamental Theorem of Algebra and Complexity Theory. Bulletin (New Series) on the American Mathematical Society, Volume 4, Number 1, 1981.

Traub, J.F. und **Wozniakowski**, H.: A General Theory of Optimal Algorithms. New York: Academic Press, 1980.

8.7 Berechnung der Nullstellen von Polynomen

Wie schon öfter erwähnt und wie durch das Wilkinson Polynom bestätigt, ist die Berechnung der Nullstellen von Polynomen oft ein schlecht konditioniertes Problem. Trotzdem ist es von Interesse. Es gibt mehrere Verfahren, die für Polynome maßgeschnitten sind.

8.7.1 Das Routh-Hurwitz Problem

Für viele physikalische, volkswirtschaftliche und biologische Systeme ist das System stabil genau dann, wenn alle Nullstellen eines Polynoms, das für das System charakteristisch ist, negative Realteile haben. Es gibt zwei alte und berühmte Methoden, um festzustellen, ob alle Nullstellen eines Polynoms negative Realteile haben, ohne die Nullstellen explizit zu berechnen.

Satz 8.9 (Hurwitz) *Das Polynom*

$$f(x) = a_0 + a_1 x + \ldots + a_n x^n (a_0 > 0)$$

mit reellen Koeffizienten $a_k (k = 0, \ldots, n)$ *hat genau dann nur Nullstellen mit negativem Realteil, wenn die* $k \times k$ *Determinanten*

$$D_k^{(n)} := \begin{vmatrix} a_1 & a_0 & 0 & 0 & 0 & 0 & 0 & \ldots & \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 & 0 & \ldots & \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 & \ldots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{2k-1} & a_{2k-2} & \ldots & \ldots & \ldots & \ldots & & \ldots & a_k \end{vmatrix}$$

für $1 \leq k \leq n$ *positiv sind; dabei ist* $a_k = 0$ *für* $k > n$ *zu setzen.*

Beweis: Siehe z.B. N. Obreschkoff, Seite 108. □

Beispiel: $a_0 + a_1 x + a_2 x^2 = 0$.
In diesem Fall sind die Bedingungen $a_1 > 0$, $a_2 > 0$, was leicht überprüfbar ist.

Beispiel: (Heitzinger, Troch, Valentin, *Praxis nichtlinearer Gleichungen*, Carl Hauser Verlag, München, S. 264.)

Drehzahlregelung einer Kraftmaschine: Wächst bei der in Abb. 8.4 schematisch skizzierten Maschine die Drehzahl an, so heben die Schwungmassen das Fliehkraftpendel b etwas an und verkleinern dadurch die Öffnung des Dampfventils c. Durch die redezierte Dampfzufuhr sinkt die Drehzahl wieder ab. Fällt (z.B. durch höhere Belastung) hingegen die Drehzahl unter den Normalwert, so wird das Ventil c stärker geöffnet, durch die erhöhte Dampfzufuhr kann die Drehzahl wieder auf den Normalwert ansteigen. Die stets im System vorhandene Trägheit kann zu periodischen Schwankungen in der Drehzahl, verursacht durch ein sog. „Reglertanzen", führen. Dies kann man beispielsweise als Flackern des Lichts beobachten, wenn eine solche Maschine einen Generator treibt.

Das System des Fliehkraft-Reglers besteht aus einer Masse, einer Feder und einer Dämpfung. Die Kopplung mit der Maschine ergibt infolge der Fliehkraft der Schwingmasse eine zusätzliche nach oben gerichtete Kraft. Somit lautet die Bewegungsgleichung des Reglers:

$$m\ddot{x} + k_1\dot{x} + c_1 x = A_1\dot{\varphi} \ .$$

Hierin bedeuten x die Aufwärtsbewegungen des Reglerschiebers, bezogen auf Normalstellung (d.h. im Normalbetrieb gilt $x = 0$), m die gesamte bewegte Masse, $\dot{\varphi}$ die Abweichung der momentanen Drehzahl von der Soll-Drehzahl, k_1, c_1 und A_1, die die Dämpfung, Steifigkeit und Kopplung mit der Maschine beschreibenden positiven Parameter. Bezeichnet I das polare Trägheitsmoment, so ist das Drehmoment durch $I\ddot{\varphi}$ gegeben und die Bewegungsgleichung der Maschine kann daher mit ($A_2 > 0, \ldots$ Kopplungsparameter)

$$I\ddot{\varphi} = -A_2 x$$

angesetzt werden. Zusammenfassen beider Beziehungen führt auf

$$m\dddot{x} + k_1\ddot{x} + c_1\dot{x} = -(A_1 A_2/I)x$$

mit der charakteristischen Gleichung

$$mIz^3 + K_1 I z^2 + c_1 I z + A_1 A_2 = 0 \ .$$

Es sind alle Parameter positiv, so daß nach dem Hurwitz-Satz als einzige wesentliche Stabilitätsbedingung

$$k_1 I c_1 - m A_1 A_2 > 0 \Leftrightarrow k_1 > m A_1 A_2/(c_1 I)$$

bleibt. Erfüllt die Dämpfungskonstante k_1 diese Bedingung, so kommt das System nach plötzlichen Belastungsänderungen zur Ruhe, andernfalls werden durch solche Änderungen im System anhaltende, bzw. aufklingende Schwingungen erregt.

8.7.2 Der Sturmsche Satz

Der Satz von Sturm bietet die Möglichkeit, die Anzahl der Wurzeln eines Polynoms in einem gegebenen Intervall genau zu bestimmen. Eine wichtige Rolle spielt hierbei der

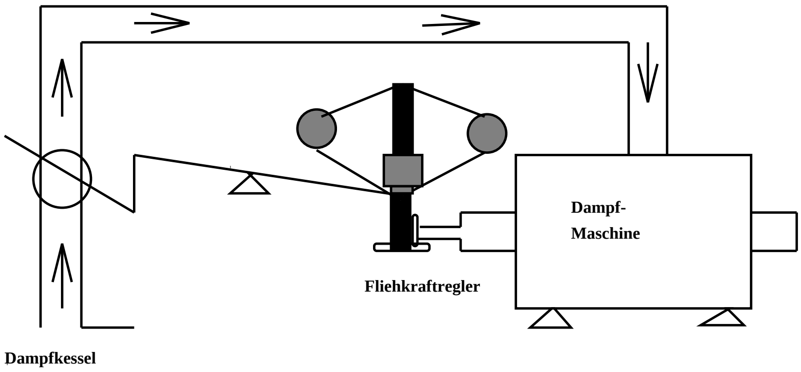


Abbildung 8.4: Drehzahlregelung einer Kraftmaschine

Begriff der Sturmschen Kette; eine solche Kette erhält man auf folgende Weise: Man wendet den Euklidischen Algorithmus zur Ermittlung des größten gemeinsamen Teilers zweier Polynome auf $p(x)$ und $p'(x)$ an; die bei diesem Verfahren auftretenden Reste seien

$$-R_1, -R_2, -R_3, \dots, -R_k\ ,$$

so daß

$$\left.\begin{array}{rcl} p(x) &=& p'(x)Q_1(x) - R_1\ , \\ p'(x) &=& R_1 Q_2(x) - R_2\ , \\ \multicolumn{3}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ R_{k-3} &=& R_{k-2}Q_{k-1}(x) - R_{k-1}\ , \\ R_{k-2} &=& R_{k-1}Q_k(x) - R_k \end{array}\right\}$$

ist. Die Folge

$$p(x), p'(x), R_1, R_2, \dots, R_k \tag{8.10}$$

bezeichnet man als Sturmsche Kette.

Wir betrachten zunächst den Fall, daß die Gleichung $p(x) = 0$ keine mehrfachen Wurzeln hat. Die Polynome $p(x)$ und $p'(x)$ haben in diesem Fall keinen gemeinsamen Teiler, wenn man von konstanten Teilern absieht, und der Rest R_k ist eine von Null verschiedene Konstante. Der Satz von Sturm lautet dann:

Satz 8.10 (Sturm) *Wenn die Gleichung $p(x) = 0$ nur einfache Wurzeln hat, dann ist die Anzahl ihrer reellen Wurzeln im Intervall (α, β) genau gleich der Anzahl der in diesem Intervall verlorengehenden Zeichenwechsel der Sturmschen Kette (8.10).*

Beweis: Siehe Obreschkoff, S. 88. □

Satz 8.11 (Sturm) *Die Gleichung $p(x) = 0$ hat in dem Intervall (α, β) so viele Wurzeln, wie die Anzahl der in diesem Intervall verlorengehenden Zeichenwechsel der Folge (8.10) angibt. Dabei werden die mehrfachen Wurzeln als einfache gezählt.*

Beweis: Siehe Obreschkoff, S. 91. □

Beispiel: $f(x) = x^4 - x^2 - 2x - 1 = 0$. Es folgt:

$$\begin{aligned} \frac{1'}{2} &= 2x^3 - x - 1 \\ 2 \cdot R_1 &= x^2 + 3x + 2 \\ R_2 &= -13x - 11 \\ R_3 &< 0 \end{aligned}$$

x	f	f'	R_1	R_2	R_3	Anzahl Zeichenwechsel
$-\infty$	$+$	$-$	$+$	$+$	$-$	3
$+\infty$	$+$	$+$	$+$	$-$	$-$	1

Es gibt also $3 - 1 = 2$ reelle Nullstellen.

8.7.3 Das Laguerresche Verfahren

Satz 8.12 (Laguerre) *Hat die algebraische Gleichung $f(x) = 0$ n-ten Grades n einfache reelle Wurzeln α_i und sind α_k, α_{k+1} zwei solche, die unmittelbar aufeinander folgen, so muß die Gleichung*

$$[(n-2)f'^2 - (n-1)ff''](X-x)^2 - 2ff'(X-x) - nf^2 = 0 \tag{8.11}$$

für jede reelle Zahl x mit $\alpha_k < x < \alpha_{k+1}$ je eine Wurzel mit $\alpha_k \leq X < x$ und $x < X \leq \alpha_{k+1}$ haben.

Beweis: siehe Obreschkoff, S. 261. Wie man leicht nachprüft, ergeben sich für die Wurzeln von (8.11) die Ausdrücke

$$X_{1,2} = x + \frac{nf(x)}{-f'(x) \pm sf(x)}$$

wobei

$$s = \sqrt{(n-1)[(n-1)(f'(x))^2 - nf(x)f''(x)]/f(x)^2}.$$

Der Beweis des Satzes beruht auf einer Hilfsungleichung. Es seien $a_1, a_2, \ldots, a_n$ beliebige reelle Zahlen, und es werde

$$\begin{aligned} a_1 &+ a_2 + \ldots + a_n = a\,, \\ a_1^2 &+ a_2^2 + \ldots + a_n^2 = b \end{aligned}$$

gesetzt. Nach der Cauchy-Schwarzschen Ungleichung wird

$$(a - a_1)^2 = (a_2 + a_3 + \ldots + a_n)^2 \leq (n-1)(a_2^2 + a_3^2 + \ldots + a_n^2) = (n-1)(b - a_1^2)$$

Wir setzen

$$p(z) := nz^2 - 2az + a^2 - (n-1)b$$

und damit

$$p(a_1) = na_1^2 - 2a_1 a + a^2 - (n-1)b \leq 0 \; .$$

Da $p(z)$ positiv ist für $|z|$ groß, ergibt sich, daß jede der Zahlen $a_k (k = 1, \ldots, n)$ zwischen den Wurzeln der Gleichung

$$p(x) = nx^2 - 2ax + a^2 - (n-1)b = 0$$

liegen muß, d.h., es ist

$$\frac{a - \sqrt{(n-1)(nb-a^2)}}{n} \leq a_k \leq \frac{a + \sqrt{(n-1)(nb-a^2)}}{n} \; . \tag{8.12}$$

Weiter gilt:

$$\frac{f'(x)}{f(x)} = \sum_{k=1}^{n} \frac{1}{x-\alpha_k} \; , \; \left[\frac{f'(x)}{f(x)}\right]^2 - \frac{f''(x)}{f(x)} = \sum_{k=1}^{n} \frac{1}{(x-\alpha_k)^2}. \tag{8.13}$$

Man setze nun

$$a_k := \frac{1}{x-\alpha_k} \; , \; a := \frac{f'(x)}{f(x)} \; , \; b := \left[\frac{f'(x)}{f(x)}\right]^2 - \frac{f''(x)}{f(x)} \tag{8.14}$$

Es ergibt sich aus (8.12) bis (8.14):

$$nb - a^2 = \frac{(n-1)(f')^2 - nff''}{f^2}$$

woraus

$$u_1 := \frac{f' - sf}{nf} \leq \frac{1}{x-\alpha_j} \leq \frac{f' + sf}{nf} =: u_2 \; , \; 1 \leq j \leq n \tag{8.15}$$

folgt. Wegen $\alpha_k < x < \alpha_{k+1}$ ist $x - \alpha_k > 0$ und $x - \alpha_{k+1} < 0$, woraus

$$u_1 < 0 \; , \; u_2 > 0 \tag{8.16}$$

folgt. Sei $\alpha_j < x$. Dann gilt wegen (8.15) und (8.16)

$$\begin{aligned} x - \alpha_j \quad &\geq \quad \frac{1}{u_2} \\ \Rightarrow \alpha_j \leq x \quad &- \quad \frac{1}{u_2} = x_1 < x \end{aligned}$$

Sei $\alpha_j > x$. Dann folgt ebenfalls

$$\begin{aligned} a_j - x \quad &\geq \quad -\frac{1}{u_1} \\ \Rightarrow \alpha_j \geq x \quad &- \quad \frac{1}{u_1} = x_2 > x \end{aligned}$$

Insbesondere gilt

$$\begin{aligned} \alpha_k &\leq X_1 < x \\ \alpha_{k+1} &\geq X_2 > x \end{aligned}$$

und der Satz ist bewiesen. $\square$

Bemerkung: Da die Größen

$$\begin{aligned} X_1 - x &= -\frac{1}{u_2} \\ X_2 - x &= -\frac{1}{u_1} \end{aligned}$$

entgegengesetzte Vorzeichen haben, ist $s > f'/f$. Es folgt:

$$\begin{aligned} X_1 &= x - \frac{nf(x)}{f'(x) + \operatorname{sign}(f) \cdot \sigma} \\ X_2 &= x - \frac{nf(x)}{f'(x) - \operatorname{sign}(f) \cdot \sigma} \\ \sigma &:= |f(x)|s = \sqrt{(n-1)((n-1)(f'(x))^2 - nf(x)f''(x))} \end{aligned}$$

Satz 8.13 *$p(x)$ sei ein Polynom vom Grad n mit n reellen einfachen Wurzeln. x^0 sei ein beliebiger Punkt. Die Folge $\{x^k\}$ wird durch das Laguerresche Verfahren definiert:*

$$\begin{aligned} x^{k+1} &:= x^k - \frac{nf(x^k)}{f'(x^k) + \mathit{sign}(f'(x^k)) \cdot \sigma_k} \\ \sigma_k &:= \sqrt{(n-1)((n-1)(f'(x^k))^2 - nf(x^k) \cdot f''(x^k))} \end{aligned}$$

Dann konvergiert x^k gegen eine Nullstelle α von p. Die Konvergenz ist von der Ordnung 3.

Beweis: f' hat nur eine Nullstelle ξ zwischen zwei nacheinander folgenden Nullstellen von f. Es folgt:

a) $\alpha_k < \xi < x^0 < \alpha_{k+1}$, also

$$\begin{aligned} & f(x^0) \cdot f'(x^0) < 0 \\ \Rightarrow\ & x^1 > x^0 \\ \Rightarrow\ & x^k \uparrow \alpha_{k+1}\ . \end{aligned}$$

b) $\alpha_k < x^0 < \xi < \alpha_{k+1}$, also

$$\begin{aligned} & f(x^0) \cdot f'(x^0) > 0 \\ \Rightarrow \quad & x^1 < x^0 \\ \Rightarrow \quad & x^k \downarrow \alpha_k \end{aligned}$$

Der Beweis, daß die Konvergenz von der Ordnung drei ist, folgt aus leichten aber mühseligen Berechnungen. □

8.7.4 Das Hornerschema für $p_0(x) = a_0 x^n + \ldots + a_n$

Wir setzen

$$\begin{aligned} b_0 &:= a_0 \\ b_i &:= b_{i-1}\alpha + a_i \qquad i = 1, 2, \ldots, n\ ,\ \alpha \in \mathbb{R} \\ \Rightarrow b_n &= p_0(\alpha) \end{aligned}$$

Dann gilt:

$$p_0(x) = (x - \alpha) p_1(x) + b_n\ ,$$

wobei

$$p_1(x) := b_0 x^{n-1} + \ldots + b_{n-1}\ .$$

Auf diese Weise ist es erst möglich, die Ableitungen von $p(x)$ effizient zu berechnen.

8.7.5 Deflation

Wenn eine Nullstelle α von $p(x)$ berechnet worden ist, bleibt die Frage, wie die übrigen Nullstellen zu berechnen sind. Es gibt zwei Möglichkeiten:

1. Man berechne die Nullstellen der rationalen Funktion

$$q(x) := \frac{p(x)}{x - \alpha}\ .$$

2. Man berechne das Quotientenpolynom $p_1(x)$ und berechne dann die Nullstellen von $p_1(x)$. Dieses Verfahren hat mehrere Vorteile, aber Vorsicht ist notwendig.

   Peters und **Wilkinson**, *Practical Problems Arising in the Solution of Polynomial Equations*, J.Inst.Maths.Applics 8(1971) 16-35, schlagen die folgende Methode vor:

(a) Man berechne eine Nullstelle α des Polynoms $f(x)$. Dabei ist $f(\alpha)$ nicht Null, sondern

$$|g\ell(f(\alpha))| \leq K\delta \ ,$$

wobei bekannt ist, daß

$$|g\ell(f(\alpha)) - f(\alpha)| \leq \delta$$

und die Konstante $K = 0(1)$.

(b) Man berechne die Koeffizienten

$$\begin{aligned}
p_{n-1} \quad &, \dots , \quad p_0, p_{-1} \\
q_n \quad &, \dots , \quad q_1, q_0 \ , \\
f(x) = (a_n x^n + \ldots + a_0) \quad &= \quad (x-\alpha)(p_{n-1}x^{n-1} + \ldots + p_0) + p_{-1} \ , \ a_0 \neq 0 \\
(a_0 + \ldots + a_n x^n) \quad &= \quad (-\alpha + x)(q_0 + q_1 x + \ldots + q_{n-1}x^{n-1}) + q_n x^n \ .
\end{aligned}$$

(c) Man bestimme r mit $|a_r\alpha^r| = \max_k \ a_k\alpha^k$ und setze

$$g(x) = (p_{n-1}x^{n-1} + \ldots + p_r x^r) + (q_{r-1}x^{r-1} + \ldots + q_0) \ .$$

(d) Dann gilt:

$$\begin{aligned}
(x-\alpha)g(x) &= f(x) + (q_{r-1} - \alpha p_r - a_r)x^r \\
&= f(x) - x^r f(\alpha)/\alpha^r \\
&= f(x) - x^r a_r \frac{f(\alpha)}{a_r\alpha^r} \ .
\end{aligned}$$

Es folgt aus der Rundungsfehleranalyse, daß

$$\begin{aligned}
|f(\alpha)| &\leq K\delta = K\sum_{k=0}^{n} |a_k\alpha^k\delta_k| \\
&\leq K(n+1)|a_r\alpha^r| \max|\delta_k| \\
\Rightarrow \left|\frac{f(\alpha)}{a_r\alpha^r}\right| &\leq (n+1)K \ \max|\delta_k| \ .
\end{aligned}$$

Wir sehen deshalb, daß das Produkt $(x-\alpha)g(x)$ einer geringen relativen Veränderung eines Koeffizienten von f entspricht.

Beispiel: (um die Definition von $g(x)$ zu erläutern)

$$\begin{aligned}
f(x) &= (x+1)^2(x+3) \\
&= x^3 + 5x^2 + 7x + 3 \\
&= a_3x^3 + a_2x^2 + a_1x + a_0 \ .
\end{aligned}$$

Wir nehmen $\alpha = -2$.

$$\begin{aligned} f(x) &= (x+2)(x^2+3x+1)+1 \\ &= (x+2)(p_2x^2+p_1x+p_0)+p_{-1} \\ f(x) &= (2+x)\left(\frac{3}{2}+\frac{11}{4}x+\frac{9}{8}x^2\right)-\frac{x^3}{8} \\ &= (2+x)(q_0+q_1x+q_2x^2)+q_3x^3 \end{aligned}$$

$$\left.\begin{aligned} |a_0\alpha^0| &= 3 \\ |a_1\alpha^1| &= 14 \\ |a_2\alpha^2| &= 20 \\ |a_3\alpha^3| &= 8 \end{aligned}\right\} \Rightarrow r = 2$$

$$\begin{aligned} g(x) &= p_2x^2+q_1x+q_0 \\ &= \left(x^2+\frac{11}{4}x+\frac{3}{2}\right) \\ (x+2)g(x) &= x^3+\left(\frac{11}{4}+2\right)x^2+7x+3 \\ &= f(x)-\frac{1}{4}x^2 \\ &= f(x)-x^2f(-2)/(-2)^2 \,. \end{aligned}$$

Literatur

Marden, M.: Geometry of Polynomials. Providence, AMS 1966.

Obreschkoff, N.: Verteilung und Berechnung der Nullstellen reeller Polynome. Berlin: Deutscher Verlag der Wissenschaften, 1963.

Peters, G., **Wilkinson**, J.H.: Practical Problems arising in the solution of polynomial equations. J.I.M.A. 8(1971) 16-35.

Ralston, A.: A first course in numerical analysis. McGraw Hill.

Wilkinson, J.H.: The evaluation of the zeros of ill-conditioned polynomials I. Numer. Math. 1(1959) 150-166.

8.8 Berechnung der Nullstellen von Systemen nichtlinearer Gleichungen

8.8.1 Beispiele

Beispiel:[Carnahan, Luther, Wilkes [1969], S. 321] Eine Gleichgewichtslösung des folgenden schemischen Systems wird gesucht:

$$
\begin{aligned}
CH_4 &+ \frac{1}{2}\, 0_2 \rightleftharpoons CO + 2H_2 \\
CH_4 &+ H_2 0 \rightleftharpoons CO + 3H_2 \\
H_2 &+ CO_2 \rightleftharpoons CO + H_2 0
\end{aligned}
$$

Dies führt zu folgendem Gleichungssystem für $x \in \mathbb{R}^7$:

$$
\begin{aligned}
f_1(x) &= \frac{1}{2}\, x_1 + x_2 + \frac{1}{2}\, x_3 - \frac{x_6}{x_7} = 0 \\
f_2(x) &= x_3 + x_4 + 2x_5 - \frac{2}{x_7} = 0 \\
f_3(x) &= x_1 + x_2 + x_5 - \frac{1}{x_7} = 0 \\
f_4(x) &= -28837x_1 - 139009x_2 - 78213x_3 + 18927x_4 + 8427x_5 + \frac{13492}{x_7} - 10690\frac{x_6}{x_7} = 0 \\
f_5(x) &= x_1 + x_2 + x_3 + x_4 + x_5 - 1 = 0 \\
f_6(x) &= 400x_1x_4^3 - 1.7837 \times 10^5 x_3x_5 = 0 \\
f_7(x) &= x_1x_3 - 2.6058x_2x_4 = 0\ .
\end{aligned}
$$

Beispiel:[Holden, Peel und Thompson [1982], S. 41)] Ein kleines Modell der Wirtschaft

Großbritanniens:

$$
\begin{aligned}
Y &= C + I + G + X - IM \\
C &= 0.4YD + 0.01FA/P + 0.5C_{-1} \\
I &= 450 - 3200(R - PEXP) + 0.8I_{-1} \\
G &= GEX + (UB.UN)/P + 0.25(R.B)/P \\
X &= 3000 - 3000(E.P/PW) + 100WT + 0.5_{-1} \\
IM &= 1160(E.P/PW) + 0.12Y + 0.5IM_{-1} - 2600 \\
T &= -2000 + TR(Y.P) \\
M &= 5555P - 65000R.P + 0.39Y.P + 0.5P(M/P)_{-1} \\
H &= 0.1M \\
dP &= -0.005(UN - \overline{U}) + PEXP \\
LD &= 0.001Y - 0.075TIME + 0.5LD_{-1} \\
E &= 0.5(PW/P) + 5\Delta(R - WR) + 0.5(R - WR) + 0.5E_{-1} \\
PEXP &= dP_{-1} \\
P &= P_{-1}(1 + dP)^{0.25} \\
PSBR &= P.G - T \\
\Delta B &= PSBR - \Delta H \\
\Delta H &= H - H_{-1} \\
B &= B_{-1} + \Delta B \\
FA &= FA_{-1} + \Delta FA \\
UN &= LS - LD \\
\Delta FA &= PSBR + P.X - (PW.IM)/E \\
YD &= Y - T/P
\end{aligned}
$$

Es gibt 22 Veränderliche, die an der linken Seite der obigen Gleichungen stehen. Jede Veränderliche hat volkswirtschaftliche Bedeutung, z.B.:

Y = Sozialprodukt
UN = Anzahl der Arbeitslosen
P = Preisniveau

Das Modell wird benutzt, um die Wirtschaftslage in dreimonatigen Zeitintervallen zu berechnen. Es enthält einige Werte aus dem vorherigen Zeitintervall:

$$C_{-1},\ I_{-1},\ X_{-1},\ M_{-1},\ P_{-1},\ D_{-1},\ E_{-1},\ H_{-1},\ B_{-1},\ DP_{-1}$$

Sind diese gegeben, kann das Modell benutzt werden, um die 22 Veränderlichen im jetzigen Zeitintervall zu berechnen.

In der Praxis werden ähnliche aber größere Modelle benutzt, z.B. enthält das Modell, das von der britischen Regierung benutzt wird, mehr als 1000 Gleichungen.

Weitere Beispiele
Volkswirtschaftslehre, Kraftwerke, Flüssigkeitslehre, gewöhnliche Differentialgleichungen, partielle Differentialgleichungen, Integralgleichungen.

Literatur

Carnahan, B., **Luther**, H.A., **Wilkes**, J.O.: Applied Numerical Methods. New York, Wiley, 1969.

Holden, K, **Peel**, D.A., **Thompson**, J.L.: Modelling the UK Economy. Martin Robertson: Oxford, 1982.

8.8.2 Das Newton-Verfahren: Definition

Im Falle $n = 1$ ist das Newton-Verfahren mit geometrischer Erklärung gegeben:

$$x^{k+1} = x^k - f(x^k)/f'(x^k) \ . \tag{8.17}$$

Im allgemeinen Fall wird das Newton-Verfahren folgendermaßen definiert:

$$f : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$$
$$x^0 \in D \quad \text{sei gegeben}$$
$$x^{k+1} = x^k - [f'(x^k)]^{-1} f(x^k) \ . \tag{8.18}$$

Das Newton-Verfahren läßt sich auf verschiedene Weise erklären:

1. Als Verallgemeinerung von (8.17)

2. Zu betrachten ist die folgende Frage: Gibt es einen linearen Operator T, damit

$$g(x) = x + Tf(x)$$

kontrahierend ist?

Wie schon bewiesen, ist g kontrahierend, falls $\|g'\| < 1$ gilt. Nötig ist deshalb, daß $\|g'(x^*)\| < 1$. Je kleiner $\|g'(x^*)\|$, desto schneller die Konvergenz. Deshalb sollte T so gewählt werden, daß

$$g'(x^*) = I + Tf'(x^*) = 0 \ .$$

Daraus folgt

$$T = -[f'(x^*)]^{-1} \ .$$

Da x^* unbekannt ist, wird x^* durch x^k ersetzt und man erhält (8.18)

3. Nach dem Satz von Taylor gilt, falls f zweimal stetig differenzierbar ist,

$$0 = f(x^*) = f(x^k) + f'(x^k)(x^* - x^k) + 0(\|x^* - x^k\|^2) .$$

Durch Vernachlässigung des quadratischen Gliedes erhält man

$$0 \doteq f(x^k) + f'(x^k)(x^* - x^k) .$$

Hieraus ergibt sich die Gleichung

$$0 = f(x^k) + f'(x^k)(x^{k+1} - x^k) ,$$

was zu (8.18) äquivalent ist.

Beispiel:[Powell - siehe auch Brown 1973, S. 314]

$$f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$$

$$x \longmapsto \begin{pmatrix} 10^4 x_1 x_2 - 1 \\ e^{-x_1} + e^{-x_2} - 1.0001 \end{pmatrix}$$

$$f'(x) = \begin{pmatrix} 10^4 x_2 & 10^4 x_1 \\ -e^{-x_1} & -e^{-x_2} \end{pmatrix}$$

$$x_0^T := (0, 1)$$

(VIC-20-NEWTON-POWELL.)

k	x_1^k	x_2^k	$f_1(x^k)$	$f_2(x^k)$
0	0	1	-1	0.36
1	1.0×10^4	1.999	.999	.135
2	3.6×10^{-8}	2.999	$-.999$	.04
3	3.3×10^{-5}	3.996	.333	.018
4	1.6×10^{-4}	4.989	$-.164$	6.69×10^{-3}
5	1.67×10^{-5}	5.97	$.89 \times 10^{-5}$	2.4×10^{-3}
6	1.40×10^{-5}	6.92	$-.025$	8.6×10^{-4}
7	1.26×10^{-5}	1.81	$-.012$	2.9×10^{-4}
8	1.16×10^{-5}	8.53	-7.3×10^{-3}	8.4×10^{-5}
9	1.11×10^{-5}	8.97022	-2.1×10^{-3}	1.6×10^{-5}
10	1.09×10^{-5}	9.0972	-1.6×10^{-4}	9.8×10^{-7}
11	1.098×10^{-5}	9.106	-7.8×10^{-7}	4.1×10^{-9}
12	1.0981595×10^{-5}	9.10614	4.6×10^{-10}	0
13	1.0981595×10^{-5}	9.10614	0	0

Die Lösung ist

$$f'(x^*) \doteq \begin{pmatrix} 9.1 \times 10^4 & .11 \\ -1 & -1.1 \times 10^{-4} \end{pmatrix}$$

mit Konditionszahl

$$\kappa = \|f'\|_\infty \cdot \|(f')^{-1}\|_\infty = 0(10^9) \; .$$

Bei der Berechnung von $f_2(x)$ entsteht Auslöschung, und $f_2(x)$ läßt sich in folgender Gestalt besser berechnen:

$$e^{-x_1} + e^{-x_2} - 1.0001 = (e^{-x_1} - 1) + (e^{-x_2} - 10^{-4}) \doteq (-x_1 + (x_1)^2/2) + (x^{-x_2} - 10^{-4}) \; .$$

8.8.3 Das Newton-Verfahren: Konvergenz

Jetzt betrachten wir mathematische Eigenschaften des Newton-Verfahrens. Für gegebene $f : D \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $x^0 \in D$, gibt es mehrere Fragen:

1. Ist die Folge $\{x^k\}$ wohldefiniert, also $x^k \in D$ für alle k und $f'(x^k)$ nichtsingulär für alle k ?
2. Konvergiert $\{x^k\}$?
3. Wie schnell konvergiert $\{x^k\}$?
4. Falls $x^k \longrightarrow x^* \in D$, ist x^* die einzige Nullstelle von f in D ?

Die erste Frage ist am schwierigsten. Um diese Frage zu beantworten, ist es nötig, Information über $f'(x)$ für $x \in D$ zur Verfügung zu haben. Es gibt mehrere Sätze über das Newton-Verfahren, und die Information über f' wird auf verschiedene Methoden gefordert:

1. Es wird verlangt, daß $f'(x)$ nichtsingulär ist für $x \in D$.
2. Es wird verlangt, daß $f'(x^*)$ nichtsingulär ist und daß $f'(x)$ eine Lipschitz-Bedingung für $x \in D$ erfüllt.
3. Es wird verlangt, daß $f'(x^*)$ nichtsingulär ist und daß $f''(x)$ für $x \in D$ beschränkt ist.

Zu bemerken ist, daß Information über $f'(x)$ für alle $x \in D$ nötig ist.

Hilfssatz 8.3 *A und B seien lineare Abbildungen von $\mathbb{R}^n$ nach $\mathbb{R}^n$. A sei nichtsingulär, und es gelte:*

$$\|A^{-1}(B-A)\| = r < 1 \ .$$

Dann ist B nichtsingulär und

$$\|B^{-1}\| \le \frac{\|A^{-1}\|}{1-r} \ .$$

Beweis: Zuerst beweisen wir, daß B nichtsingulär ist. Sonst gibt es $x \in \mathbb{R}^n$ mit $x \neq 0$ und $Bx = 0$. Dann gilt

$$\|x\| = \|A^{-1}Ax\| = \|A^{-1}(A-B)x\| \le r\|x\|$$

was unmöglich ist.
Da B nichtsingulär ist, gilt für alle $x \in \mathbb{R}^n$

$$\begin{aligned} y &:= B^{-1}x \\ y &= A^{-1}(A-B)y + A^{-1}By = A^{-1}(A-B)y + A^{-1}x \end{aligned}$$

Es folgt

$$\|y\| \le r\|y\| + \|A^{-1}\|\|x\|$$

und

$$\|B^{-1}x\| = \|y\| \le \frac{\|A^{-1}\|}{1-r}\, \|x\| \ .$$

□

Hilfssatz 8.4 *$g : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ sei zweimal differenzierbar auf einer konvexen Menge $D_0 \subset D$. Für alle $x, y \in D_0$ gilt:*

$$\|g(y) - g(x) - g'(x)(y-x)\| \le \sup_{0\le t\le 1} \|g''(x+t(y-x))\|\ \|y-x\|^2 \ .$$

Beweis: Sei für $w \in D_0$ $F(w) := g(w) - g'(x)(w-x)$. Nach Hilfssatz 8.2 gilt

$$\|F(y) - F(x)\| \le \sup_{0\le t\le 1} \|F'(x+t(y-x))\| \cdot \|y-x\|$$

und

$$\begin{aligned} \|F'(x+t(y-x))\| &= \|g'(x+t(y-x)) - g'(x)\| \\ &\le \sup_{0\le s\le 1} \|g''(x+st(y-x))\|t\|y-x\| \end{aligned}$$

Zusammenfassend

$$\begin{aligned} \|g(y) - g(x) - g'(x)(y-x)\| &= \|F(y) - F(x)\| \\ &\le \sup_{0\le t\le 1}\ \sup_{0\le s\le 1} \|g''(x+st(y-x))\|t\|y-x\|^2 \ . \end{aligned}$$

□

Satz 8.14 *Es sei*

a) $U = \{x \in \mathbb{R}^n : \|x - x^*\| < d\}$, *mit* $d > 0$.

b) $f : U \longrightarrow \mathbb{R}^n$ *sei zweimal stetig differenzierbar.*

c) $f(x^*) = 0$.

d) $f'(x^*)$ *sei nichtsingulär.*

Dann gibt es positive Konstanten K *und* δ : *Wenn* $\|x^0 - x^*\| \leq \delta$, *dann ist die Folge* $\{x^k\}$

$$x^{k+1} := x^k - [f'(x^k)]^{-1} f(x^k)\ , \ k \geq 0$$

wohldefiniert, $x^k \longrightarrow x^*$ *für* $k \to \infty$, *und*

$$\|x^{k+1} - x^*\| \leq K \|x^k - x^*\|^2\ .$$

Beweis: Sei $U_0 \subset \overline{U}_0 \subset U$, $M_2 := \sup_{x \in U_0} \|f''(x)\|$, und $A := f'(x^*)$. Da $f'(x)$ stetig ist, gibt es eine Konstante $\delta_1 < d$:

$$\|f'(x) - A\| \leq 1/(2\|A^{-1}\|)\ , \quad \text{falls} \quad \|x - x^*\| \leq \delta_1\ .$$

Es folgt aus dem Hilfssatz 8.3, daß $f'(x)$ nichtsingulär ist und

$$\begin{aligned} \|[f'(x)]^{-1}\| &\leq K_1 := 2\|A^{-1}\|\ , \quad \text{für} \quad x \in U_1\ , \\ U_1 &= \{x \in \mathbb{R}^n : \|x - x^*\| \leq \delta_1\}\ . \end{aligned}$$

Es folgt aus Hilfssatz 8.4, daß

$$f(x^*) - f(x^k) - f'(x^k)(x^* - x^k) = \varphi(x^*, x^k)\ ,$$

wobei

$$\|\varphi(x^*, x^k)\| \leq M_2 \|x^* - x^k\|^2\ .$$

Dann gilt, für $x^k \in U_1$,

$$\begin{aligned} \|x^{k+1} - x^*\| &= \|x^k - x^* - [f'(x^k)]^{-1} f(x^k)\| \\ &= \|x^k - x^* - [f'(x^k)]^{-1}[f(x^*) - f'(x^k)(x^* - x^k) - \varphi(x^*, x^k)]\| \\ &= \|[f'(x^k)]^{-1} \varphi(x^*, x^k)\| \\ &\leq K_1 M_2 \|x^* - x^k\|^2\ . \end{aligned}$$

Sei

$$\begin{aligned} \delta_2 &= \min\{\delta_1, 1/(2K_1M_2)1\}\ , \\ U_2 &= \{x \in \mathbb{R}^n : \|x - x^*\| \leq \delta_2\}\ . \end{aligned}$$

Dann gilt:

1. Sei $x^k \in U_2$. Dann gilt
$$\|x^{k+1} - x^*\| \leq \frac{1}{2} \|x^k - x^*\| ,$$
so daß $x^{k+1} \in U_2$. Insbesondere sei $x^0 \in U_2$, dann ist $x^k \in U_2$ für alle k.

2. $\|x^{k+1} - x^*\| \leq K\|x^k - x^*\|^2$, mit $K := K_1 M_2$.

□

Ein weiterer Satz, worin die Existenz von x^* nicht vorausgesetzt ist, ist folgender:

Satz 8.15 (Newton-Kantorovitch) *(Stoer, S. 227) Es sei eine offene Menge $D \subseteq \mathbb{R}^n$ gegeben, ferner eine konvexe Menge D_0 mit $\overline{D}_0 \subseteq D$, und $f :\longrightarrow \mathbb{R}^n$ sei eine für alle $x \in D_0$ differenzierbare und für alle $x \in D$ stetige Funktion. Für ein $x^0 \in D_0$ gebe es positive Konstanten $r, \alpha, \beta, \gamma, h$ mit den folgenden Eigenschaften:*

$$\begin{aligned} S_r(x^0) &:= \{x : \|x - x^0\| < r\} \subseteq D_0 \\ h &:= \alpha\beta\gamma/2 < 1 \\ r &:= \alpha/(1-h) . \end{aligned}$$

$f(x)$ habe die Eigenschaften

a) $\|Df(x) - Df(y)\| \leq \gamma\|x - y\|$ für alle $x, y \in D_0$,

b) $(Df(x))^{-1}$ existiert und es gilt $\|(Df(x))^{-1}\| \leq \beta$ für alle $x \in D_0$,

c) $\|(Df(x^0))^{-1} f(x^0)\| \leq \alpha$, wobei $Df := f'$.

Dann gilt

A) Ausgehend von x^0 ist jedes
$$x^{k+1} := x^k - Df(x^k)^{-1} f(x^k) , \quad k = 0, 1, \ldots$$
wohldefiniert und es gilt $x^k \in S_r(x^0)$ für alle $k \geq 0$.

B) $\lim_{k\to\infty} x^k = x^*$ existiert und es gilt
$$x^* \in \overline{S_r(x^0)} \quad \textit{und} \quad f(x^*) = 0 .$$

C) Für alle $k \geq 0$ gilt
$$\|x^k - x^*\| \leq \alpha \, \frac{h^{2^k} - 1}{1 - h^{2^k}} .$$

Wegen $0 < h < 1$ ist also das Newton-Verfahren mindestens quadratisch-konvergent.

Beweis: A) Da $Df(x)^{-1}$ für $x \in D_0$ existiert, ist x^{k+1} für alle k dann wohldefiniert, wenn $x^k \in S_r(x^0)$ für alle $k \geq 0$. Dies ist richtig für $k = 0$ und $k = 1$ wegen Vor.c). Wenn nun $x^j \in S_r(x^0)$ für $j = 0, 1, \ldots, k$ für ein $k \geq 1$, so gilt zunächst wegen b)

$$\begin{aligned} \|x^{k+1} - x^k\| &= \|-Df(x^k)^{-1}f(x^k)\| \leq \beta\|f(x^k)\|\,, \\ &= \beta\|f(x^k) - f(x^{k-1}) - Df(x^{k-1})(x^k - x^{k-1})\|\,. \end{aligned}$$

da nach Definition von x^k gilt,

$$f(x^{k-1}) + Df(x^{k-1})(x^k - x^{k-1}) = 0\,.$$

Die gewünschte Abschätzung folgt aus den Voraussetzungen (a) - (c). Siehe Stoer für den Rest des Beweises. □

Dieser Satz ermöglicht konstruktive Existenzbeweise. Siehe z.B. Rall (1969).

Literatur

Rall, L.: Computational Solution of Nonlinear Operator Equations. New York: Wiley, 1969.

8.8.4 Erweiterungen des Newton-Verfahrens

Das Newton-Verfahren ist sehr intensiv studiert worden, und es gibt mehrere Erweiterungen, von denen einige jetzt kurz erwähnt werden.

Zunächst wird das Newton-Verfahren wiedergegeben:

$$A(x^k)[x^{k+1} - x^k] = -f(x^k) \tag{8.19}$$

oder

$$x^{k+1} = x^k - [A(x^k)]^{-1}f(x^k) \tag{8.20}$$

oder

$$A(x^k) = f'(x^k) = J(x^k) \tag{8.21}$$

mit

$$J(x) = \text{ Funktionalmatrix } = \left(\frac{\partial f_i}{\partial x_j}\right)\,. \tag{8.22}$$

Approximation von $f'(x)$ durch Differenzen

Die Berechnung von $J(x^k)$ ist sehr zeitraubend. In einigen Fällen ist es möglich, die Berechnung von $J(x)$ zu automatisieren (siehe Rall).

Es liegt nahe, $J(x^k)$ numerisch zu approximieren. Eine Möglichkeit ist, $f(x)$ in der Nähe von x^k auszuwerten. Dann gibt es:

$$\frac{\partial f_i(x)}{\partial x_j}|_{x=x^k} \doteq \Delta_j f_i := \frac{f_i(x^k + he_j) - f_i(x^k)}{h} ,$$

d.h. durch die Berechnung von $f(x^k + he_j)$ für $j = 1, 2, \dots, n$ und h klein wird eine Annäherung zu $f'(x^k)$ gewonnen. Diese Methode erfordert $n+1$ Berechnungen von f für jeden Schritt des Newton-Verfahrens, und dies ist für n groß sehr ungünstig.

Das Broyden-Verfahren
Siehe Dennis und Schnabel, Kapitel 8.

Literatur

Griewank, A.: On Solving nonlinear equations with simple singularities or nearly singular solutions. SIAM Review 27 (1985) 537 -.

Kelley,C.T und **Suresh**,R.: A New Acceleration Method for Newton's Method at Singular Points. SIAM J. Numer. Anal., Vol. 20, No. 5, October 1983.

8.9 Homotopiemethoden

Literatur

Allgower,E. und **Georg**,K.: Simplicial and Continuation Methods for Approximating Fixed Points and Solutions to Systems of Equations. SIAM Review, Vol. 22, No. 1, Jan. 1980.

Chow,S.-N., **Mallet-Paret**,J. und **Yorke**,J.: Finding Zeroes of Maps: Homotopy Methods That Are Constructive With Probality One. Mathematics of Computation, Vol. 32, Number 143, 1978.

Eaves,B.C, **Gould**,F.J.m **Peitgen**,H.O. und **Todd**,M.J.: Homotopy Methods and Global Convergence. New York: Plenum Press, 1983.

Kellogg,R.B, **Li**,T.Y. und **Yorke**,J: A Constructive Proof of the Brouwer Fixed-Point Theorem and Computational Results. SIAM J. Numer. Anal. Vol. 13, No. 4, Sept. 1976.

Lloyd,N.G.: Degree Theory. Cambridge Tracts in Math., Vol. 73, Cambridge Univ. Press, Cambridge, Great Britain, 1978.

Scarf, H.: The Approximation of Fixed Points of a Continuous Mapping. SIAM J. Appl. Math., Vol. 15, No. 5, Sept. 1967.

Todd, M.J.: The Computation of Fixed Points and Applications. Lecture Notes in Economics and Mathematical Systems, 1976.

8.10 Chaos, Globale Konvergenz usw.

Literatur

Li, T.Y., **Yorke**, James A.: Period Three Implies Chaos. Am. Math. Monthly 82 (1975).

Pickover, C.A.: A note on chaos and Halley's method. Comm. ACM 31 (1988) 1326 - 1329.

Smale