

Hans Rudolf Schwarz | Norbert Köckler

Numerische Mathematik

8. Auflage

► Mit Online-Service

STUDIUM



Hans Rudolf Schwarz | Norbert Köckler

Numerische Mathematik

Hans Rudolf Schwarz | Norbert Köckler

Numerische Mathematik

8., aktualisierte Auflage

STUDIUM



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

Prof. Dr. Hans Rudolf Schwarz
Universität Zürich
Mathematisch-Naturwissenschaftliche Fakultät (MNF)
Winterthurerstrasse 190
8057 Zürich

Prof. Dr. Norbert Köckler
Universität Paderborn
Fakultät EIM – Institut für Mathematik
33098 Paderborn
norbert@upb.de

1. Auflage 1986
- 6., überarbeitete Auflage 2006
- 7., überarbeitete Auflage 2009
- 8., aktualisierte Auflage 2011

Alle Rechte vorbehalten
© Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden GmbH 2011

Lektorat: Ulrike Schmickler-Hirzebruch | Barbara Gerlach

Vieweg+Teubner Verlag ist eine Marke von Springer Fachmedien.
Springer Fachmedien ist Teil der Fachverlagsgruppe Springer Science+Business Media.
www.viewegteubner.de



Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlags unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

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

Umschlaggestaltung: KünkelLopka Medienentwicklung, Heidelberg
Druck und buchbinderische Verarbeitung: AZ Druck und Datentechnik, Berlin
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier
Printed in Germany

ISBN 978-3-8348-1551-4

Vorwort zur 7. und 8. Auflage

Neben kleinen Korrekturen wurde für diese Auflagen das Kapitel 11 über die iterative Lösung von linearen Gleichungssystemen überarbeitet. Der Abschnitt über die ohnehin langsam konvergierenden Relaxationsverfahren wurde gekürzt und ein Abschnitt über Mehrgittermethoden hinzugefügt. Ich freue mich, dass ich damit einen Wunsch vieler Leser erfüllen kann, und bedanke mich bei meiner Tochter Ricarda für die sorgfältige Durchsicht.

Eine elektronische Version dieses Lehrbuchs wird über www.viewegteubner.de angeboten werden. Zusätzlich liegen von den meisten Kapiteln Fassungen im PowerPoint-Stil vor, die als Folien oder für Vorlesungen mit Beamer verwendet werden können. Sie sind für Dozenten kostenlos über das DozentenPLUS-Portal des Verlags erhältlich.

Eine spezielle Internet-Seite ermöglicht den Zugriff auf die Programm-Masken zum Buch und enthält alle aktuellen Informationen zum Stand dieses Projektes:

www.uni-paderborn.de/SchwarzKoeckler/.

Paderborn, im Februar 2011

Norbert Köckler

Aus dem Vorwort zur 5. und 6. Auflage

Mit großer Freude habe ich die Bearbeitung und Fortführung dieses klassischen Lehrbuchs über die numerische Mathematik übernommen. Ich habe versucht, den Inhalt des Buches an die Entwicklung anzupassen, ohne den Anspruch meiner Vorgänger, der Herren Kollegen Prof. Dr. R. Stiefel und Prof. Dr. H. R. Schwarz, auf Vollständigkeit, Rigorosität bei den mathematischen Grundlagen und algorithmische Orientierung aufzugeben.

Inhaltlich habe ich eine Reihe von Änderungen durchgeführt. Das Kapitel über *lineare Optimierung* ist weggefallen, weil dies heute kaum noch in den Kanon der numerischen Ausbildung gehört und es gute Lehrbücher gibt, die sich ausschließlich diesem Thema widmen. Ein Kapitel über Fehlertheorie ist hinzugekommen. Die Kapitel über Interpolation und Funktionsapproximation habe ich zusammengelegt, weil ich glaube, dass in den Anwendungen von der Aufgabe ausgegangen wird Daten oder Funktionen zu approximieren, und erst dann die Entscheidung für die Interpolation oder für die Approximation gefällt wird. Die anderen Kapitel haben sich mal mehr, mal weniger verändert, der Leser der früheren Auflagen sollte insgesamt keine großen Umstellungsprobleme haben. Am Ende der Kapitel gibt es manchmal einen Abschnitt über Anwendungen und immer einen Abschnitt über Software, deren Gebrauch für uns Numeriker unerlässlich ist und die in verwirrender Vielfalt über das Internet oder andere Quellen erreichbar ist.

Herrn Schwarz danke ich für seine kollegiale Unterstützung, den Herren Sandten und Spuhler vom Teubner-Verlag für ihre professionelle und verständnisvolle Betreuung, Frau Karin Senske hat die typographische Umsetzung und manche Durchsicht mit Sorgfalt, Fleiß und Verständnis für mich erledigt; dafür bin ich ihr außerordentlich dankbar. Meinem lieben Kollegen Prof. Dr. Gisbert Stoyan von der ELTE in Budapest bin ich für seine akribische Durchsicht besonders dankbar; sie hat zur Korrektur zahlreicher kleiner Fehler, aber auch zu besserer Verständlichkeit einiger Formulierungen beigetragen.

Paderborn, im Sommer 2006

Norbert Köckler

Aus dem Vorwort zur 4. Auflage

Das Buch entstand auf den seinerzeitigen ausdrücklichen Wunsch meines verehrten Lehrers, Herrn Prof. Dr. E. Stiefel, der mich im Sinne eines Vermächtnisses beauftragte, sein während vielen Jahren wegweisendes Standardwerk [Sti 76] von Grund auf neu zu schreiben und den modernen Erkenntnissen und Bedürfnissen anzupassen. Klarheit und Ausführlichkeit waren stets die Hauptanliegen von Herrn Professor Stiefel. Ich habe versucht, in diesem Lehrbuch dieser von ihm geprägten Philosophie zu folgen, und so werden die grundlegenden Methoden der numerischen Mathematik in einer ausführlichen Darstellung behandelt.

Das Buch ist entstanden aus Vorlesungen an der Universität Zürich. Der behandelte Stoff umfasst im Wesentlichen das Wissen, das der Verfasser seinen Studenten in einem viersemestrigen Vorlesungszyklus zu je vier Wochenstunden vermittelte. Sie sollen damit in die Lage versetzt werden, Aufgaben der angewandten Mathematik mit numerischen Methoden erfolgreich zu lösen oder zumindest die Grundlagen für das Studium von weiterführender spezialisierter Literatur zu haben. Das Buch richtet sich an Mathematiker, Physiker, Ingenieure, Informatiker und Absolventen naturwissenschaftlicher Richtungen. Vorausgesetzt wird diejenige mathematische Vorbildung, die in den unteren Semestern eines Hochschulstudiums oder an Ingenieurschulen vermittelt wird.

Die Darstellung des Stoffes ist algorithmisch ausgerichtet. Zur Begründung einer numerischen Methode werden zuerst die theoretischen Grundlagen vermittelt, soweit sie erforderlich sind, um anschließend das Verfahren so zu formulieren, dass seine Realisierung in einem Programm einfach ist.

Um die speziellen Kenntnisse auf dem Gebiet der numerischen Integralberechnung, die Herr Dr. J. Waldvogel an der ETH Zürich erarbeitet hat, in das Buch einfließen zu lassen, hat er die Abschnitte 7.1 und 7.3 sowie die zugehörigen Aufgaben verfasst. Für diese wertvolle Mitarbeit danke ich ihm hiermit bestens. Meinen beiden Assistenten, den Herren Dipl.-Math. W. Businger und H. P. Märczy verdanke ich viele Anregungen und die kritische Durchsicht des Manuskripts. Schließlich danke ich dem Verlag B. G. Teubner für die Herausgabe des Buches und für die stets freundliche und entgegenkommende Zusammenarbeit.

Mit der vierten Auflage des Buches wurde versucht, eine Aktualisierung des Stoffumfangs zu erreichen, indem in verschiedener Hinsicht Ergänzungen eingefügt wurden. Um eine oft bemängelte Lücke zu schließen, wurden grundlegende Methoden zur Behandlung von Randwertaufgaben bei gewöhnlichen Differenzialgleichungen aufgenommen. Weiter wurde im gleichen Zug die für die Computergraphik zentrale Bézier-Technik zur Darstellung von Kurven und Flächen berücksichtigt. Schließlich fanden die modernen Aspekte der Vektorisierung und Parallelisierung von Algorithmen Aufnahme im Buch. Das notwendige Vorgehen zur Vektorisierung wird am Beispiel der effizienten Lösung von linearen Gleichungssystemen mit vollbesetzter und tridiagonaler Matrix dargelegt. Desgleichen werden die wesentliche Idee und Techniken der Parallelisierung am Beispiel der Lösung von linearen Gleichungssystemen entwickelt und die einschlägigen Algorithmen dargestellt.

Zürich, im Herbst 1996

H. R. Schwarz

Inhalt

Einleitung	13
1 Fehlertheorie	15
1.1 Fehlerarten	15
1.2 Zahldarstellung	16
1.3 Rundungsfehler	18
1.4 Differenzielle Fehleranalyse	21
1.5 Ergänzungen und Beispiele	24
1.6 Software	27
1.7 Aufgaben	28
2 Lineare Gleichungssysteme, direkte Methoden	30
2.1 Der Gauß-Algorithmus	30
2.1.1 Elimination, Dreieckszerlegung und Determinantenberechnung	30
2.1.2 Pivotstrategien	38
2.1.3 Ergänzungen	43
2.2 Genauigkeitsfragen, Fehlerabschätzungen	47
2.2.1 Normen	47
2.2.2 Fehlerabschätzungen, Kondition	52
2.3 Systeme mit speziellen Eigenschaften	56
2.3.1 Symmetrische, positiv definite Systeme	56
2.3.2 Bandgleichungen	62
2.3.3 Tridiagonale Gleichungssysteme	64
2.4 Verfahren für Vektorrechner und Parallelrechner	67
2.4.1 Voll besetzte Systeme	68
2.4.2 Tridiagonale Gleichungssysteme	73
2.5 Anwendungen	82
2.6 Software	87
2.7 Aufgaben	88

3	Interpolation und Approximation	91
3.1	Polynominterpolation	92
3.1.1	Problemstellung	92
3.1.2	Lagrange-Interpolation	95
3.1.3	Newton-Interpolation	95
3.1.4	Hermite-Interpolation	98
3.1.5	Inverse Interpolation	100
3.1.6	Anwendung: Numerische Differenziation	101
3.2	Splines	106
3.2.1	Kubische Splines	107
3.2.2	B-Splines 1. Grades	112
3.2.3	Kubische B-Splines	114
3.3	Zweidimensionale Splineverfahren	119
3.3.1	Bilineare Tensorsplines	120
3.3.2	Bikubische Tensorsplines	123
3.4	Kurveninterpolation	125
3.5	Kurven und Flächen mit Bézier-Polynomen	127
3.5.1	Bernstein-Polynome	127
3.5.2	Bézier-Darstellung eines Polynoms	129
3.5.3	Der Casteljau-Algorithmus	130
3.5.4	Bézier-Kurven	131
3.5.5	Bézier-Flächen	137
3.6	Gauß-Approximation	140
3.6.1	Diskrete Gauß-Approximation	142
3.6.2	Kontinuierliche Gauß-Approximation	144
3.7	Trigonometrische Approximation	145
3.7.1	Fourier-Reihen	145
3.7.2	Effiziente Berechnung der Fourier-Koeffizienten	154
3.8	Orthogonale Polynome	161
3.8.1	Approximation mit Tschebyscheff-Polynomen	162
3.8.2	Interpolation mit Tschebyscheff-Polynomen	170
3.8.3	Die Legendre-Polynome	174
3.9	Software	179
3.10	Aufgaben	179
4	Nichtlineare Gleichungen	183
4.1	Theoretische Grundlagen	183
4.1.1	Problemstellung	183
4.1.2	Konvergenztheorie und Banachscher Fixpunktsatz	185
4.1.3	Stabilität und Kondition	189

Inhalt	9
4.2 Gleichungen in einer Unbekannten	190
4.2.1 Das Verfahren der Bisektion	190
4.2.2 Das Verfahren von Newton	192
4.2.3 Die Sekantenmethode	195
4.2.4 Brents Black-box-Methode	196
4.3 Gleichungen in mehreren Unbekannten	199
4.3.1 Fixpunktiteration und Konvergenz	199
4.3.2 Das Verfahren von Newton	200
4.4 Nullstellen von Polynomen	207
4.4.1 Reelle Nullstellen: Das Verfahren von Newton-Maehly	207
4.4.2 Komplexe Nullstellen: Das Verfahren von Bairstow	211
4.5 Software	215
4.6 Aufgaben	215
5 Eigenwertprobleme	218
5.1 Theoretische Grundlagen	219
5.1.1 Das charakteristische Polynom	219
5.1.2 Ähnlichkeitstransformationen	219
5.1.3 Symmetrische Eigenwertprobleme	220
5.1.4 Elementare Rotationsmatrizen	220
5.2 Das klassische Jacobi-Verfahren	222
5.3 Die Vektoriteration	229
5.3.1 Die einfache Vektoriteration nach von Mises	229
5.3.2 Die inverse Vektoriteration	231
5.4 Transformationsmethoden	232
5.4.1 Transformation auf Hessenberg-Form	233
5.4.2 Transformation auf tridiagonale Form	237
5.4.3 Schnelle Givens-Transformation	239
5.5 QR-Algorithmus	243
5.5.1 Grundlagen zur QR-Transformation	243
5.5.2 Praktische Durchführung, reelle Eigenwerte	248
5.5.3 QR-Doppelschritt, komplexe Eigenwerte	253
5.5.4 QR-Algorithmus für tridiagonale Matrizen	256
5.5.5 Zur Berechnung der Eigenvektoren	260
5.6 Das allgemeine Eigenwertproblem	261
5.6.1 Der symmetrisch positiv definite Fall	261
5.7 Eigenwertschranken, Kondition, Stabilität	264
5.8 Anwendung: Membranschwingungen	268
5.9 Software	270
5.10 Aufgaben	271

6	Ausgleichsprobleme, Methode der kleinsten Quadrate	274
6.1	Lineare Ausgleichsprobleme, Normalgleichungen	274
6.2	Methoden der Orthogonaltransformation	278
6.2.1	Givens-Transformation	279
6.2.2	Spezielle Rechentechniken	284
6.2.3	Householder-Transformation	286
6.3	Singulärwertzerlegung	292
6.4	Nichtlineare Ausgleichsprobleme	296
6.4.1	Gauß-Newton-Methode	297
6.4.2	Minimierungsverfahren	300
6.5	Software	304
6.6	Aufgaben	305
7	Numerische Integration	307
7.1	Newton-Cotes-Formeln	308
7.1.1	Konstruktion von Newton-Cotes-Formeln	308
7.1.2	Verfeinerung der Trapezregel	310
7.2	Romberg-Integration	313
7.3	Transformationsmethoden	315
7.3.1	Periodische Integranden	316
7.3.2	Integrale über \mathbb{R}	318
7.3.3	Variablensubstitution	320
7.4	Gauß-Integration	323
7.4.1	Eingebettete Gauß-Regeln	331
7.5	Adaptive Integration	332
7.6	Mehrdimensionale Integration	336
7.6.1	Produktintegration	336
7.6.2	Integration über Standardgebiete	337
7.7	Software	338
7.8	Aufgaben	339
8	Anfangswertprobleme	342
8.1	Einführung	343
8.1.1	Problemklasse und theoretische Grundlagen	343
8.1.2	Möglichkeiten numerischer Lösung	345
8.2	Einschrittverfahren	350
8.2.1	Konsistenz	350
8.2.2	Runge-Kutta-Verfahren	353
8.2.3	Explizite Runge-Kutta-Verfahren	354

Inhalt	11
8.2.4 Halbimplizite Runge-Kutta-Verfahren	358
8.2.5 Schrittweitensteuerung	359
8.3 Mehrschrittverfahren	363
8.3.1 Verfahren vom Adams-Typ	363
8.3.2 Konvergenztheorie und Verfahrenskonstruktion	368
8.4 Stabilität	376
8.4.1 Inhärente Instabilität	376
8.4.2 Absolute Stabilität bei Einschrittverfahren	378
8.4.3 Absolute Stabilität bei Mehrschrittverfahren	380
8.4.4 Steife Differenzialgleichungen	384
8.5 Anwendung: Lotka-Volterras Wettbewerbsmodell	388
8.6 Software	391
8.7 Aufgaben	392
9 Rand- und Eigenwertprobleme	395
9.1 Problemstellung und Beispiele	395
9.2 Lineare Randwertaufgaben	399
9.2.1 Allgemeine Lösung	399
9.2.2 Analytische Methoden	401
9.2.3 Analytische Methoden mit Funktionenansätzen	404
9.3 Schießverfahren	408
9.3.1 Das Einfach-Schießverfahren	408
9.3.2 Das Mehrfach-Schießverfahren	413
9.4 Differenzenverfahren	418
9.4.1 Dividierte Differenzen	418
9.4.2 Diskretisierung der Randwertaufgabe	419
9.5 Software	424
9.6 Aufgaben	425
10 Partielle Differenzialgleichungen	427
10.1 Differenzenverfahren	427
10.1.1 Problemstellung	427
10.1.2 Diskretisierung der Aufgabe	429
10.1.3 Randnahe Gitterpunkte, allgemeine Randbedingungen	434
10.1.4 Diskretisierungsfehler	444
10.1.5 Ergänzungen	446
10.2 Parabolische Anfangsrandwertaufgaben	448
10.2.1 Eindimensionale Probleme, explizite Methode	448
10.2.2 Eindimensionale Probleme, implizite Methode	454
10.2.3 Diffusionsgleichung mit variablen Koeffizienten	459

10.2.4	Zweidimensionale Probleme	461
10.3	Methode der finiten Elemente	466
10.3.1	Grundlagen	466
10.3.2	Prinzip der Methode der finiten Elemente	469
10.3.3	Elementweise Bearbeitung	471
10.3.4	Aufbau und Behandlung der linearen Gleichungen	477
10.3.5	Beispiele	477
10.4	Software	482
10.5	Aufgaben	483
11	Lineare Gleichungssysteme, iterative Verfahren	487
11.1	Diskretisierung partieller Differenzialgleichungen	487
11.2	Relaxationsverfahren	489
11.2.1	Konstruktion der Iterationsverfahren	489
11.2.2	Einige Konvergenzsätze	494
11.2.3	Optimaler Relaxationsparameter und Konvergenzgeschwindigkeit	505
11.3	Mehrgittermethoden	508
11.3.1	Ein eindimensionales Modellproblem	508
11.3.2	Eigenschaften der gedämpften Jacobi-Iteration	509
11.3.3	Ideen für ein Zweigitterverfahren	511
11.3.4	Eine eindimensionale Zweigittermethode	513
11.3.5	Eine erste Mehrgittermethode	517
11.3.6	Die Mehrgitter-Operatoren für das zweidimensionale Modellproblem	520
11.3.7	Vollständige Mehrgitterzyklen	522
11.3.8	Komplexität	523
11.3.9	Ein Hauch Theorie	524
11.4	Methode der konjugierten Gradienten	530
11.4.1	Herleitung des Algorithmus	530
11.4.2	Eigenschaften der Methode der konjugierten Gradienten	534
11.4.3	Konvergenzabschätzung	538
11.4.4	Vorkonditionierung	541
11.5	Methode der verallgemeinerten minimierten Residuen	547
11.5.1	Grundlagen des Verfahrens	548
11.5.2	Algorithmische Beschreibung und Eigenschaften	551
11.6	Speicherung schwach besetzter Matrizen	556
11.7	Software	559
11.8	Aufgaben	559
Literaturverzeichnis		563
Sachverzeichnis		576

Einleitung

Gegenstand und Ziel

Numerische Mathematik befasst sich damit,
für mathematisch formulierte Probleme
einen rechnerischen Lösungsweg zu finden.
(H. Rutishauser)

Da die meisten Probleme der Natur-, Ingenieur- und Wirtschaftswissenschaften vor ihrer rechnerischen Lösung mathematisch modelliert werden, entwickelt die numerische Mathematik für eine Vielzahl von Problemstellungen rechnerische Lösungswege, so genannte *Algorithmen*, siehe Definition 1.1. Sie muss sich daher neben der Mathematik auch mit der Auswahl von Hard- und Software beschäftigen. Damit ist die numerische Mathematik Teil des Gebietes *wissenschaftliches Rechnen* (Scientific Computing), das Elemente der Mathematik, der Informatik und der Ingenieurwissenschaften umfasst.

Die Entwicklung immer leistungsfähigerer Rechner hat dazu geführt, dass heute Probleme aus Luft- und Raumfahrt, Physik, Meteorologie, Biologie und vielen anderen Gebieten rechnerisch gelöst werden können, deren Lösung lange als unmöglich galt. Dabei gehen die Entwicklung von Algorithmen und Rechnern Hand in Hand. Ziel der Ausbildung in numerischer Mathematik ist deshalb auch die Erziehung zu algorithmischem Denken, d.h. zur Kreativität beim Entwurf von Rechnerlösungen für Anwendungsprobleme.

Vom Problem zur Lösung

Folgende Schritte führen von einem Anwendungsproblem zu seiner numerischen Lösung:

Modellierung: Ein Anwendungsproblem muss zunächst in die Form eines mathematischen Modells gegossen werden. Dies geschieht meistens auf der Grundlage idealisierter Annahmen. Es findet also schon die erste Annäherung statt, damit eine Lösung – exakt analytisch oder angenähert numerisch – möglich wird.

Realisierung: Für das mathematische Modell muss eine Lösungsmethode gefunden werden. Ist diese numerisch, so kann in der Regel zwischen mehreren Verfahren gewählt werden. Zum Verfahren passend wird ein Programm oder ein ganzes Softwaresystem gesucht oder selbst entwickelt, um damit das Problem zu lösen. Dabei entstehen besonders bei größeren Problemen zusätzliche Aufgaben wie Benutzerführung, Datenorganisation und Darstellung der Lösung oft in Form einer Visualisierung.

Validierung: Auf dem Weg vom Anwendungsproblem zu seiner numerischen Lösung sind mehrfach Fehler im Sinne von Annäherungen gemacht worden, zuletzt durch das Rechnen auf einem Rechner mit endlicher Stellenzahl, siehe Kapitel 1. Deswegen müssen das Modell auf seine Gültigkeit, das Programm auf seine Zuverlässigkeit und schließlich das numerische Verfahren auf seine Stabilität, also Fehleranfälligkeit, überprüft werden. Wenn dann mit konkreten Zahlen gerechnet wird, sollte jede einzelne Rechnung von einer Fehleranalyse begleitet werden unbeschadet davon, dass dies in der Praxis nicht immer durchführbar ist.

Für ein Problem aus der Technik können diese Schritte etwa wie folgt aussehen.

Problemschritte	Physikalisches Beispiel
1. Physikalisches Problem	– Brückenbau
2. Physikalisches Modell	– Spannungstheorie
3. Mathematisches Modell	– Differenzialgleichung
4. Numerisches Verfahren	– Auswahl: z.B. Finite Elemente Methode
5. Algorithmus	– Genauer Rechenablauf, auch die Ein- und Ausgabe
6. Programm	– Eigene oder Fremdsoftware (Auswahl)
7. Rechnung	– Datenorganisation
8. Fehlertheorie	– Abschätzung mit den konkreten Daten

Hardware, Software, Pakete, Bibliotheken, Werkzeuge

Die Realisierung numerischer Methoden erfordert eine leistungsfähige Hardware. Oft werden spezielle Rechner wie Vektor- oder Parallelrechner eingesetzt. Dem müssen die numerischen Verfahren und damit auch die Software angepasst werden. Die zuverlässigsten Programme findet man in großen Bibliotheken wie denen der Firmen NAG und IMSL und in Problemlösungsumgebungen wie MATLAB. Sie verwenden Algorithmen, die sorgfältig entwickelt und getestet wurden. Auf solche Pakete wird in den Software-Abschnitten der einzelnen Kapitel Bezug genommen. Gute Möglichkeiten zur Programm-Suche bieten der AMS-Guide to mathematical software <http://gams.nist.gov> und die NETLIB-Liste <http://www.netlib.org>. Darüberhinaus gibt es Programmsammlungen, die oft im Zusammenhang mit Lehrbüchern entwickelt werden, und so genannte *Templates*, denen wir zuerst in Abschnitt 5.9 begegnen. Objekt-orientierte Numerik findet man über <http://www.oonumerics.org/oon/>.

Literaturhinweise

Für die numerische Mathematik gibt es eine große Zahl von Lehrbüchern. Einige sind stärker theoretisch orientiert [Col 68, Häm 94, Hen 82, Hen 72, Lin 01, Stu 82, Sch 05], während andere das wissenschaftliche Rechnen in den Vordergrund stellen [Dew 86, Gol 95, Gol 96a, Koc 90] oder sogar konsequent die Software-Realisierung mit der Vermittlung der numerischen Methoden verknüpfen [EM 90, Gan 92, Hop 88, Ske 01, Übe 95]. Die überwiegende Zahl der neueren Lehrbücher bietet neben einer fundierten Mathematik praktische Anwendungen und den Bezug auf mögliche Software-Lösungen [Bol 04, Dah 03, Deu 08b, Deu 08a, Roo 99, Sto 07, Sto 05]. Diesen Weg versuchen wir mit diesem Lehrbuch auch zu gehen.

1 Fehlertheorie

Die numerische Mathematik löst mathematische Probleme durch die Anwendung von Rechenverfahren (Algorithmen) auf Daten. Dabei entsteht eine vorgegebene Menge von Zahlenwerten als Ergebnis. Die Daten können – z.B. auf Grund von Messungen – fehlerhaft sein, und die Ergebnisse der Rechenoperationen werden durch die endliche Stellenzahl bei der Rechnung ebenfalls verfälscht. Deshalb ist es eine wichtige Aufgabe der Numerik, neben den Methoden, mit denen ein mathematisches Problem numerisch gelöst werden kann, auch die Entstehung von Fehlern bei der konkreten Berechnung von Ergebnissen zu untersuchen.

Definition 1.1. 1. Ein *Algorithmus* ist eine für jeden möglichen Fall eindeutig festgelegte Abfolge von elementaren Rechenoperationen unter Einbeziehung mathematischer Funktionen und Bedingungen.

2. Zu den *elementaren Rechenoperationen* gehören die Grundrechenarten, logische Operationen sowie (rechnerabhängig) weitere Operationen wie die Auswertung von Standardfunktionen, etwa der e -Funktion oder der Wurzelfunktion.

1.1 Fehlerarten

Datenfehler sind Fehler, die auf Grund fehlerhafter Eingabedaten in den Algorithmus einfließen. Sind Messungen der Grund, dann werden sie auch *Messfehler* genannt.

Diskretisierungsfehler, historisch auch *Abbruchfehler*, werden Fehler genannt, die dadurch entstehen, dass ein kontinuierliches mathematisches Problem „diskretisiert“, d.h. endlich bzw. diskret gemacht wird. Dabei wird z. B. die Berechnung unendlicher Summen nach endlich vielen Summanden „abgebrochen“.

Rundungsfehler sind die beim Rechnen mit reellen Zahlen durch Rundung der (Zwischen)-Ergebnisse entstehenden Fehler. Dabei setzen wir voraus, dass alle elementaren Operationen denselben maximalen Standardfehler haben. Die Fortpflanzung der Rundungsfehler von Rechenoperation zu Rechenoperation ist die häufigste Quelle für numerische Instabilität, d.h. für Ergebnisse, die durch Fehlerverstärkung unbrauchbar werden können.

Definition 1.2. Es sei x eine reelle Zahl, $\tilde{x} \in \mathbb{R}$ ein Näherungswert für x . Dann heißen

$$\delta_x := \tilde{x} - x \tag{1.1}$$

absoluter Fehler von x und, falls $x \neq 0$,

$$\varepsilon_x := \frac{\tilde{x} - x}{x} \quad (1.2)$$

relativer Fehler von x .

1.2 Zahldarstellung

Auf einer Rechenanlage ist nur eine Teilmenge \mathcal{M} der Menge \mathbb{R} der reellen Zahlen darstellbar. Nach dem IEEE-Standard wird eine reelle Zahl $x \in \mathcal{M}$ dargestellt als

$$x = \text{sign}(x) \cdot a \cdot E^{e-k}. \quad (1.3)$$

Das Zahlensystem \mathcal{M} ist durch vier ganzzahlige Parameter bestimmt:

- die *Basis* $E \in \mathbb{N}$, $E > 1$, meistens $E = 2$,
- die *Genaugigkeit* $k \in \mathbb{N}$ und
- den *Exponenten-Bereich* $e_{\min} \leq e \leq e_{\max}$, $e_{\min}, e_{\max} \in \mathbb{Z}$.

Die *Mantisse* $a \in \mathbb{N}_0$ ist definiert als

$$a = a_1 E^{k-1} + a_2 E^{k-2} + \cdots + a_{k-1} E^1 + a_k E^0, \quad (1.4)$$

für sie gilt also $0 \leq a \leq E^k - 1$. Dabei ist k die Mantissenlänge und die a_i sind Ziffern des Zahlensystems, also 0 oder 1 im Dualsystem mit der Basis 2. Die Zahl Null ist ein Sonderfall; ist $x \neq 0$, so soll auch die erste Ziffer ungleich null sein, d.h. es ist

$$E^{k-1} \leq a < E^k, \text{ falls } x \neq 0. \quad (1.5)$$

x heißt dann k -stellige *normalisierte Gleitpunktzahl*¹ zur Basis E . Das Rechnen mit solchen Zahlen heißt *Rechnung mit k wesentlichen Stellen*.

Damit ergibt sich als *Bereich* der normalisierten Gleitpunktzahlen $x \neq 0$:

$$E^{e_{\min}-1} \leq |x| \leq E^{e_{\max}}(1 - E^{-k}). \quad (1.6)$$

Eine wichtige Eigenschaft des Zahlensystems \mathcal{M} ist, dass die Zahlen in ihm nicht den gleichen Abstand haben, also nicht äquidistant sind. Im Dualsystem springt ihr Abstand mit jeder Zweierpotenz um den Faktor 2, siehe Abb. 1.1 und Aufgabe 1.1.

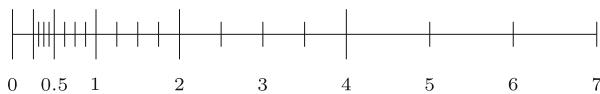


Abb. 1.1 Verteilung der Gleitpunktzahlen für $E = 2$, $k = 3$, $e_{\min} = -1$ und $e_{\max} = 3$.

Nach jeder arithmetischen Rechenoperation wird das Ergebnis x auf einen eindeutig definierten Wert $rd(x)$ gerundet. Außerdem definieren wir die *Maschinengenauigkeit*:

$$\tau := \frac{E}{2} E^{-k}. \quad (1.7)$$

¹Wir schreiben diesen Begriff wie im Angelsächsischen. Im Deutschen ist auch *Gleitkommazahl* gebräuchlich.

Lemma 1.3. Wenn $x \neq 0$ im Bereich der normalisierten Gleitpunktzahlen liegt und $rd(x)$ in \mathcal{M} , dann gilt

$$\begin{aligned} |rd(x) - x| &\leq \frac{E^{e-k}}{2} && (\text{max. absoluter Fehler bei Rundung}), \\ \frac{|rd(x) - x|}{|x|} &\leq \frac{1}{2} E^{1-k} = \tau && (\text{max. relativer Fehler bei Rundung}). \end{aligned} \quad (1.8)$$

Beweis. Sei o.B.d.A. $x > 0$. Dann kann x geschrieben werden als

$$x = \mu E^{e-k}, \quad E^{k-1} \leq \mu \leq E^k - 1.$$

Also liegt x zwischen den benachbarten Gleitpunktzahlen² $y_1 = \lfloor \mu \rfloor E^{e-k}$ und $y_2 = \lceil \mu \rceil E^{e-k}$. Also ist entweder $rd(x) = y_1$ oder $rd(x) = y_2$, und es gilt bei korrektem Runden

$$|rd(x) - x| \leq \frac{y_2 - y_1}{2} \leq \frac{E^{e-k}}{2}.$$

Daraus folgt

$$\frac{|rd(x) - x|}{|x|} \leq \frac{\frac{E^{e-k}}{2}}{\mu E^{e-k}} = \frac{1}{2} E^{1-k} = \tau.$$

□

Beispiel 1.1. Wir wollen für $x = \pi$ den Wert $y = x^2 = 9.8696044\dots$ in zweistelliger dezimaler und in fünfstelliger dualer Gleitpunktdarstellung (mit den Dualziffern 0 und 1) berechnen.

(a) Für $E = 10$ und $k = 2$ ist die Maschinengenauigkeit $\tau = 0.05$ und

$$\begin{aligned} rd(\pi) &= 3.1 = 0.31 \cdot 10^1 \\ rd(\pi)rd(\pi) &= 0.961 \cdot 10^1 \\ y_{10,2} &= rd\{rd(\pi)rd(\pi)\} = 0.96 \cdot 10^1 = 9.6 \\ y_{10,2} - \pi^2 &\approx 0.27. \end{aligned}$$

(b) Für $E = 2$ und $k = 5$ ist die Maschinengenauigkeit $\tau = 2^{-5} = 0.03125$ und

$$\begin{aligned} rd(\pi) &= rd(2^1 + 2^0 + 2^{-3} + 2^{-6} + \dots) = 2^1 + 2^0 + 2^{-3} = 0.11001 \cdot 2^2 \\ rd(\pi)rd(\pi) &= 0.1001110001 \cdot 2^4 \\ y_{2,5} &= rd\{rd(\pi)rd(\pi)\} = 0.10100 \cdot 2^4 = 10 \\ y_{2,5} - \pi^2 &\approx 0.13. \end{aligned}$$

△

Die Parameter der im IEEE-Standard vorgesehenen Gleitpunktzahlen-Systeme werden in Tabelle 1.1 wiedergegeben.

² $\lfloor \mu \rfloor$ ist die größte ganze Zahl kleiner gleich μ , $\lceil \mu \rceil$ ist die kleinste ganze Zahl größer gleich μ .

Tab. 1.1 Die Parameter der IEEE Standard-Arithmetik zur Basis $E = 2$

Genauigkeit	t	e_{\min}	e_{\max}	τ
Einfach	24	-125	128	$2^{-24} \approx 6 \times 10^{-8}$
Doppelt	53	-1021	1024	$2^{-53} \approx 1 \times 10^{-16}$
Erweitert	64	-16381	16384	$2^{-64} \approx 5 \times 10^{-20}$

1.3 Rundungsfehler

Aus Lemma 1.3 folgt

Lemma 1.4. 1. Es gilt (mit der Maschinengenauigkeit τ)

$$rd(x) = x(1 + \varepsilon) \quad \text{mit } |\varepsilon| \leq \tau. \quad (1.9)$$

2. Es sei * eine der elementaren Operationen. Dann gilt:

$$rd(x * y) = (x * y)(1 + \varepsilon) \quad \text{mit } |\varepsilon| \leq \tau. \quad (1.10)$$

3. Die Maschinengenauigkeit τ ist die kleinste positive Zahl g für die gilt:

$$rd(1 + g) > 1. \quad (1.11)$$

Definition 1.5. 1. Die *wesentlichen Stellen* einer Zahl sind ihre Mantissenstellen bei normalisierter Gleitpunktdarstellung.

2. Beim *Rechnen mit k wesentlichen Stellen* müssen alle Eingabegrößen auf k wesentliche Stellen gerundet werden, anschließend werden die Ergebnisse jeder elementaren Operation vor dem Weiterrechnen sowie das Endergebnis auf k wesentliche Stellen gerundet.

3. *Auslöschung* nennt man das Annullieren führender Mantissenstellen bei der Subtraktion zweier Zahlen gleichen Vorzeichens.

Wird ein numerischer Algorithmus instabil, so hat das in den meisten Fällen seinen Grund im Informationsverlust durch Subtraktion, also in der Auslöschung.

Wir wollen den Einfluss von Rundungsfehlern an zwei Beispielen verdeutlichen:

Beispiel 1.2. Es seien $a := 1.2$ und $b := 1.1$. Wir wollen zwei Algorithmen zur Berechnung von $a^2 - b^2 = (a + b)(a - b)$ in zweistelliger dezimaler Gleitpunktrechnung vergleichen. Es ist

$$a^2 = 1.44 \implies rd(a^2) = 1.4$$

$$b^2 = 1.21 \implies rd(b^2) = 1.2$$

$$rd(a^2) - rd(b^2) = 0.2 \quad (\text{Fehler } 13\%).$$

Mit ebenfalls drei Rechenoperationen berechnet man

$$a + b = rd(a + b) = 2.3$$

$$a - b = rd(a - b) = 0.10$$

$$(a + b)(a - b) = 0.23 \quad (\text{Fehler } 0\%).$$

△

Beispiel 1.3. Es ist

$$99 - 70\sqrt{2} = \sqrt{9801} - \sqrt{9800} = \frac{1}{\sqrt{9801} + \sqrt{9800}} = 0.005050633883346584 \dots$$

Diese drei Formeln für einen Zahlenwert sind drei Algorithmen. Ihre unterschiedliche Stabilität zeigt eine Berechnung in k -stelliger dezimaler Gleitpunktrechnung für verschiedene Werte von k .

Anzahl wesentlicher Stellen	$99 - 70\sqrt{2}$	$\sqrt{9801} - \sqrt{9800}$	$1/(\sqrt{9801} + \sqrt{9800})$
2	1.0	0.0	0.0050
4	0.02000	0.01000	0.005051
6	0.00530000	0.00510000	0.00505063
10	0.005050660000	0.005050640000	0.005050633884

Der Grund für die numerische Instabilität der ersten beiden Algorithmen ist die Subtraktion der beiden fast gleichen Zahlen, also Auslöschung. Es gehen alle bzw. 3 bzw. 4 Stellen verloren, was auch durch die abschließenden Nullen deutlich wird. \triangle

Zur systematischen Untersuchung wird der Algorithmus als Funktion dargestellt:

$$\begin{aligned} \text{Eingabedaten: } & x \in \mathbb{R}^m \\ \text{Ergebnisse: } & y \in \mathbb{R}^n \\ \text{Algorithmus: } & y = \varphi(x) \end{aligned} \quad (1.12)$$

Die exakte Analyse des Algorithmus mit Berücksichtigung der Eingabe- oder Datenfehler, der Rundungsfehlerfortpflanzung von Rechenschritt zu Rechenschritt und der Ergebnisrundung wird *differentielle Fehleranalyse* genannt, siehe Abschnitt 1.4. Hier soll zunächst eine einfachere Modellsituation betrachtet werden³

Lemma 1.6. Seien $\delta_x \in \mathbb{R}^m$ der absolute Datenfehler von x und

$$D\varphi = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x_1} & \dots & \frac{\partial \varphi_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varphi_n}{\partial x_1} & \dots & \frac{\partial \varphi_n}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{n,m} \quad (1.13)$$

die Funktionalmatrix der Funktion φ .

Dann gilt unter der Voraussetzung, dass exakt, also ohne den Einfluss von Rundungsfehlern, gerechnet wird, für die Ergebnisfehler des Algorithmus $y = \varphi(x)$ in erster Näherung:

$$\delta_y \doteq D\varphi \delta_x \quad (1.14)$$

$$\varepsilon_{y_i} \doteq \sum_{k=1}^m K_{ik} \varepsilon_{x_k}, \quad i = 1, \dots, n. \quad (1.15)$$

Dabei sind K_{ik} die Konditionszahlen

$$K_{ik} = \frac{\partial \varphi_i(x)}{\partial x_k} \frac{x_k}{\varphi_i(x)}, \quad i = 1, \dots, n, \quad k = 1, \dots, m. \quad (1.16)$$

Beweis. Die Aussage (1.14) für δ_y ergibt sich sofort aus dem Satz von Taylor. Die Aussage (1.15) für ε_y entsteht durch arithmetische Erweiterung von (1.14). \square

³Wir verwenden das Zeichen \doteq für Gleichheit bis auf Glieder zweiter Ordnung, also bis auf Terme, in denen Quadrate oder höhere Potenzen einer Fehlergröße auftreten (in (1.14) und (1.15) sind das δ_x bzw. ε_{x_k}).

Die Konditionszahlen sind die Verstärkungsfaktoren der relativen Fehler *eines* Eingabedatums in *einem* Ergebnis.

Lemma 1.7.

$$|\delta_{y_i}| \dot{\leq} \sum_{k=1}^m |\delta_{x_k}| \left| \frac{\partial \varphi_i(x)}{\partial x_k} \right|, \quad i = 1, \dots, n, \quad (1.17)$$

$$|\varepsilon_{y_i}| \dot{\leq} \sum_{k=1}^m |K_{ik}| |\varepsilon_{x_k}|. \quad (1.18)$$

Beispiel 1.4. Sei

$$y = \varphi(x_1, x_2, x_3) = x_1 + x_2 + x_3,$$

also $\varphi : \mathbb{R}^3 \rightarrow \mathbb{R}$. Es sind

$$\frac{\partial y}{\partial x_j} = 1 \quad j = 1, 2, 3.$$

Also sind die Konditionszahlen gegeben als

$$K_{1j} = \frac{x_j}{x_1 + x_2 + x_3}, \quad j = 1, 2, 3.$$

Das Problem ist deshalb dann gut konditioniert, wenn $\sum x_i \gg x_j$.

Es gilt

$$\delta_y = \delta_{x_1} + \delta_{x_2} + \delta_{x_3}$$

und

$$\varepsilon_y = \frac{x_1}{x_1 + x_2 + x_3} \varepsilon_{x_1} + \frac{x_2}{x_1 + x_2 + x_3} \varepsilon_{x_2} + \frac{x_3}{x_1 + x_2 + x_3} \varepsilon_{x_3}.$$

Unter der Annahme $|\varepsilon_{x_j}| \leq \tau$ gilt

$$|\varepsilon_y| \leq \frac{|x_1| + |x_2| + |x_3|}{|x_1 + x_2 + x_3|} \tau.$$

Die Fehler werden also nur dann sehr verstärkt, wenn die Summe $x_1 + x_2 + x_3$ klein ist relativ zu den Summanden, und das kann nur durch Auslöschung entstehen. \triangle

Für die elementaren Operationen ergibt sich der

Satz 1.8. (a) Absoluter Fehler:

$$y = x_1 \pm x_2 \Rightarrow \delta_y = \delta_{x_1} \pm \delta_{x_2} \quad (1.19)$$

$$y = x_1 \cdot x_2 \Rightarrow \delta_y = x_2 \delta_{x_1} + x_1 \delta_{x_2} + \delta_{x_1} \delta_{x_2} \doteq x_2 \delta_{x_1} + x_1 \delta_{x_2} \quad (1.20)$$

$$y = \frac{x_1}{x_2} \Rightarrow \delta_y \doteq \frac{x_2 \delta_{x_1} - x_1 \delta_{x_2}}{x_2^2} \quad (x_2 \neq 0) \quad (1.21)$$

(b) Relativer Fehler:

$$\varepsilon_{x_1 \pm x_2} = \frac{x_1}{x_1 \pm x_2} \varepsilon_{x_1} \pm \frac{x_2}{x_1 \pm x_2} \varepsilon_{x_2} \quad (x_1 \pm x_2 \neq 0) \quad (1.22)$$

$$\varepsilon_{x_1 \cdot x_2} \doteq \varepsilon_{x_1} + \varepsilon_{x_2} \quad (1.23)$$

$$\varepsilon_{x_1/x_2} \doteq \varepsilon_{x_1} - \varepsilon_{x_2} \quad (1.24)$$

$$\varepsilon_{x^n} \doteq n \varepsilon_x, \quad n \in Q, \quad (\text{z.B. } \varepsilon_{\sqrt{x}} \doteq \frac{1}{2} \varepsilon_x) \quad (1.25)$$

Beweis. Übung. □

Auslöschung entsteht, wenn zwei Zahlen x_1, x_2 mit $x_1 \approx x_2$ voneinander abgezogen werden, weil dann nach (1.22) $\varepsilon_{x_1-x_2} \gg \varepsilon_{x_i}$.

1.4 Differenzielle Fehleranalyse

Wir lassen jetzt die Voraussetzung des exakten Rechnens fallen und berücksichtigen alle Fehler, vom Eingangsfehler (Datenfehler) bis zur Rundung der errechneten Ergebnisse einschließlich aller fortgepflanzten Rundungsfehler. Wir betrachten den in elementare Rechenoperationen zerlegten Algorithmus:

$$\begin{aligned} x = x^{(0)} &\longrightarrow \varphi^{(0)}(x^{(0)}) =: x^{(1)} \rightarrow \dots \rightarrow \varphi^{(r)}(x^{(r)}) =: x^{(r+1)} \equiv y \\ x \in \mathbb{R}^m &\longrightarrow y = \varphi(x) \in \mathbb{R}^n \end{aligned} \quad (1.26)$$

Dabei sind im Allgemeinen alle Zwischenergebnisse Vektoren; die elementaren Rechenoperationen sind elementweise zu verstehen. Wir definieren so genannte *Restabbildungen*

$$\begin{aligned} \psi^{(k)} &:= \varphi^{(r)} \circ \varphi^{(r-1)} \circ \dots \circ \varphi^{(k)} : \mathbb{R}^{m_k} \rightarrow \mathbb{R}^n, \\ \psi^{(k-1)} &= \psi^{(k)} \circ \varphi^{(k-1)}, \quad k = 2, \dots, r. \end{aligned} \quad (1.27)$$

Satz 1.9. Der beim Algorithmus (1.26) aus Datenfehler δ_x und Rundung beim Rechnen mit k wesentlichen Stellen fortgepflanzte Fehler δ_y ergibt sich als

$$\delta_y \doteq D\varphi(x) \delta_x + D\psi^{(1)}(x^{(1)}) E_1 x^{(1)} + \dots + D\psi^{(r)}(x^{(r)}) E_r x^{(r)} + E_{r+1} y$$

oder

$$\delta_y \doteq \underbrace{D\varphi(x) \delta_x}_{\text{Datenfehler}} + \underbrace{\sum_{i=1}^r D\psi^{(i)}(x^{(i)}) E_i x^{(i)}}_{\substack{\text{fortgepflanzte Rundungsfehler} \\ \text{Resultatrundung}}} + \underbrace{E_{r+1} y}_{\text{Rundungsfehler}}. \quad (1.28)$$

Dabei sind $D\varphi, D\psi^{(i)}$ die Funktionalmatrizen der entsprechenden Abbildungen und E_i diagonale Rundungsfehlermatrizen

$$E_i := \begin{pmatrix} \varepsilon_1^{(i)} & & & \\ & \varepsilon_2^{(i)} & & \\ & & \ddots & \\ & & & \varepsilon_{m_i}^{(i)} \end{pmatrix} \quad \text{mit } |\varepsilon_j^{(i)}| \leq \tau. \quad (1.29)$$

Beweis. (nach [Sto 07]):

Aus der Kettenregel für Funktionalmatrizen $D(f \circ g)(x) = Df(g(x)) Dg(x)$ folgen

$$D\varphi(x) = D\varphi^{(r)}(x^{(r)}) D\varphi^{(r-1)}(x^{(r-1)}) \dots D\varphi^{(0)}(x),$$

$$D\psi^{(k)}(x^{(k)}) = D\varphi^{(r)}(\cdot) \cdots D\varphi^{(k)}(\cdot). \quad (1.30)$$

Wir müssen jetzt Schritt für Schritt je zwei Fehleranteile berücksichtigen:

- (a) Die Fortpflanzung der aufgelaufenen Fehler mit $D\varphi^{(i)}$,
- (b) die Rundung des gerade berechneten Zwischenergebnisses.

1. Schritt:

$$\begin{aligned} \delta_{x^{(1)}} &\doteq D\varphi^{(0)}(x) \delta_x + \alpha^{(1)} \quad \text{mit} \\ \alpha^{(1)} &:= rd(\varphi^{(0)}(\tilde{x})) - \varphi^{(0)}(\tilde{x}) \quad \text{oder komponentenweise} \\ \alpha_j^{(1)} &= \varepsilon_j^{(1)} \varphi_j^{(0)}(\tilde{x}) = \varepsilon_j^{(1)} x_j^{(1)} \quad \text{mit} \\ |\varepsilon_j^{(1)}| &\leq \tau. \end{aligned}$$

Fasst man diese $\varepsilon_j^{(1)}$ zu einer Diagonalmatrix $E_1 = \text{diag}(\varepsilon_1^{(1)}, \varepsilon_2^{(1)}, \dots, \varepsilon_{m_1}^{(1)}) \in \mathbb{R}^{m_1, m_1}$ zusammen, so bekommt man

$$\alpha^{(1)} = E_1 x^{(1)}, \quad \alpha^{(1)}, x^{(1)} \in \mathbb{R}_1^m, \quad E_1 \in \mathbb{R}^{m_1, m_1}.$$

2. Schritt:

$$\delta_{x^{(2)}} \doteq D\varphi^{(1)}(x^{(1)}) \underbrace{[D\varphi^{(0)}(x) \delta_x + \alpha^{(1)}]}_{\begin{array}{l} \text{Fehler aus 1. Schritt} \\ \text{fortgepflanzter Fehler} \end{array}} + \underbrace{\alpha^{(2)}}_{\text{neue Rundung}},$$

oder mit der Abkürzung $D_k := D\varphi^{(k)}(x^{(k)})$

$$\delta_{x^{(2)}} \doteq D_1 D_0 \delta_x + D_1 \alpha^{(1)} + \alpha^{(2)}.$$

Entsprechend ergibt sich

$$\delta_{x^{(3)}} \doteq D_2 D_1 D_0 \delta_x + D_2 D_1 \alpha^{(1)} + D_2 \alpha^{(2)} + \alpha^{(3)}.$$

Schließlich bekommt man insgesamt

$$\begin{aligned} \delta_y = \delta_{x^{(r+1)}} &\doteq D_r D_{r-1} \cdots D_0 \delta_x \quad \} \text{ fortgepfl. Datenfehler} \\ &+ D_r D_{r-1} \cdots D_1 \alpha^{(1)} \\ &+ D_r D_{r-1} \cdots D_2 \alpha^{(2)} \\ &+ \cdots \\ &+ D_r \alpha^{(r)} \\ &+ \alpha^{(r+1)} \quad \} \text{ Ergebnisrundung} \end{aligned} \quad \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \text{ fortgepfl. Rundungsfehler}$$

mit $\alpha^{(r+1)} = E_{r+1} x^{(r+1)} = E_{r+1} y$. Mit (1.30) beweist das (1.28). \square

Der Satz gibt in 1. Näherung die exakte Darstellung des Gesamtfehlers, aber nicht in berechenbarer Form, da die Zahlen $\varepsilon_j^{(k)}$ in den Diagonalmatrizen E_k nicht bekannt sind. Schätzt man den Betrag dieser Zahlen durch die Maschinengenauigkeit τ ab, so ergibt sich

$$|\delta_y| \leq \tau \sum_{i=1}^r |D\psi^{(i)}(x^{(i)})| |x^{(i)}| + |D\varphi(x)| |\delta_x| + \tau |y|. \quad (1.31)$$

Dabei sind die Beträge komponenten- bzw. elementweise zu verstehen. Wird ein Element $x_k^{(i)}$ in einem Rechenschritt nur übernommen, so ist das k -te Diagonalelement der Run dungsfehlermatrix E_i gleich null, oder: aus $x_k^{(i+1)} = x_j^{(i)}$ folgt $\varepsilon_k^{(i+1)} = 0$.

Bei längeren Algorithmen führt die differenzielle Fehleranalyse zu einer großen Anzahl von Einzeltermen, die in (1.31) alle nach oben abgeschätzt werden. Dabei können sich Fehler mit verschiedenem Vorzeichen teilweise gegenseitig aufheben. Das kann nur in einer *Fehler schätzung* berücksichtigt werden, bei der z.B. stochastische Methoden angewendet werden.

Beispiel 1.5. Es ist $1 - \frac{1-x}{1+x} = \frac{2x}{1+x}$.

Welche der beiden Berechnungsformeln ist numerisch stabiler?

Algorithmus 1:

$$\begin{aligned} x^{(0)} &= x \in \mathbb{R} \\ x^{(1)} &= \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \end{pmatrix} = \varphi^{(0)}(x) = \begin{pmatrix} 1-x \\ 1+x \end{pmatrix} : \mathbb{R} \rightarrow \mathbb{R}^2 \\ x^{(2)} &= \varphi^{(1)}(x^{(1)}) = \frac{x_1^{(1)}}{x_2^{(1)}} : \mathbb{R}^2 \rightarrow \mathbb{R} \\ x^{(3)} &= \varphi^{(2)}(x^{(2)}) = 1 - x^{(2)} : \mathbb{R} \rightarrow \mathbb{R}. \end{aligned}$$

Daraus ergeben sich als Funktionalmatrizen

$$\begin{aligned} D\varphi^{(0)} &= \begin{pmatrix} -1 \\ +1 \end{pmatrix}, \quad D\varphi^{(1)} = \left(\frac{1}{x_2^{(1)}}, \frac{-x_1^{(1)}}{(x_2^{(1)})^2} \right), \quad D\varphi^{(2)} = -1, \\ D\varphi &= \frac{2}{(1+x)^2}, \\ D\psi^{(2)} &= D\varphi^{(2)} = -1, \\ D\psi^{(1)} &= D\varphi^{(2)} D\varphi^{(1)} = \left(-\frac{1}{x_2^{(1)}}, \frac{x_1^{(1)}}{(x_2^{(1)})^2} \right). \end{aligned}$$

Setzen wir für die $x_k^{(j)}$ die Ausdrücke in x ein, so ergibt das mit (1.28)

$$\begin{aligned} \delta_y &\doteq D\varphi \delta_x + D\psi^{(1)} E_1 x^{(1)} + D\psi^{(2)} E_2 x^{(2)} + E_3 y \\ &= \frac{2\delta_x}{(1+x)^2} + \left(-\frac{1}{1+x}, \frac{1-x}{(1+x)^2} \right) \begin{pmatrix} \varepsilon_1^{(1)} & 0 \\ 0 & \varepsilon_2^{(1)} \end{pmatrix} \begin{pmatrix} 1-x \\ 1+x \end{pmatrix} \\ &+ (-1) \varepsilon_1^{(2)} \frac{1-x}{1+x} + \varepsilon_1^{(3)} \left(1 - \frac{1-x}{1+x} \right) \\ &= \frac{2\delta_x}{(1+x)^2} + \frac{-(1-x)}{1+x} \varepsilon_1^{(1)} + \frac{1-x^2}{(1+x)^2} \varepsilon_2^{(1)} - \varepsilon_1^{(2)} \frac{1-x}{1+x} + \varepsilon_1^{(3)} \left(1 - \frac{1-x}{1+x} \right). \end{aligned}$$

Unter der Annahme $|x| \ll 1$, $\tau \ll |x|$, $|\delta_x| \leq \tau$ kann man $1 - x \approx 1 + x \approx 1$ setzen und bekommt

$$|\delta_y| \overset{<}{\approx} 5\tau.$$

Algorithmus 2:

$$x^0 = x, \quad x^{(1)} = \varphi^{(0)}(x) = \begin{pmatrix} 2x \\ 1+x \end{pmatrix} : \mathbb{R} \rightarrow \mathbb{R}^2,$$

$$y = x^{(2)} = \varphi^{(1)}(x^{(1)}) = \frac{x_1^{(1)}}{x_2^{(1)}} : \mathbb{R}^2 \rightarrow \mathbb{R}.$$

Das ergibt die Funktionalmatrizen

$$D\varphi^{(0)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad D\varphi^{(1)} = \left(\frac{1}{x_2^{(1)}}, -\frac{x_1^{(1)}}{(x_2^{(1)})^2} \right) = \left(\frac{1}{1+x}, \frac{-2x}{(1+x)^2} \right),$$

$$D\psi^{(1)} = D\varphi^{(1)},$$

$$D\varphi = \frac{2}{(1+x)^2}.$$

(1.28) liefert jetzt

$$\begin{aligned} \delta_y &\doteq \frac{2\delta_x}{(1+x)^2} + \left(\frac{1}{x_2^{(1)}}, \frac{-x_1^{(1)}}{(x_2^{(1)})^2} \right) \begin{pmatrix} \varepsilon_1^{(1)} & 0 \\ 0 & \varepsilon_2^{(1)} \end{pmatrix} \begin{pmatrix} 2x \\ 1+x \end{pmatrix} + \varepsilon_1^{(2)} \frac{2x}{1+x} \\ &= \frac{2\delta_x}{(1+x)^2} + \varepsilon_1^{(1)} \frac{2x}{1+x} + \varepsilon_2^{(1)} \frac{(-2x)}{1+x} + \varepsilon_1^{(2)} \frac{2x}{1+x} \end{aligned}$$

Mit denselben Annahmen wie beim Algorithmus 1 ergibt das

$$|\delta_y| \overset{<}{\approx} 2\tau.$$

Für $x = 1 \cdot 10^{-3}$ ergibt sich $y_{\text{exakt}} = 1.998001998 \cdot 10^{-3}$.

$$\begin{aligned} E = 10, k = 8 : \quad &\text{Algorithmus 1: } y = 1.9980000 \cdot 10^{-3}, \\ &\text{Algorithmus 2: } y = 1.9980020 \cdot 10^{-3}. \end{aligned}$$

Algorithmus 2 liefert also ein auf acht Stellen korrektes Ergebnis, das um zwei Stellen genauer ist als das von Algorithmus 1. Das verstärkt noch den in den differenziellen Fehleranalysen fest gestellten Unterschied zwischen den beiden Algorithmen. \triangle

1.5 Ergänzungen und Beispiele

Rückwärts-Fehleranalyse (backward analysis)

Die differentielle Fehleranalyse ist eine Vorwärtsuntersuchung der Fehlerentwicklung. Eine andere Idee ist die Rückwärtsuntersuchung.

Der Algorithmus $y = \varphi(x)$ wird mit Datenfehler und Rundungsfehlerfortpflanzung zu

$$\tilde{y} = \tilde{\varphi}(\tilde{x})$$

Problem: Finde ein δ_x so, dass

$$\tilde{y} = \varphi(x + \delta_x)$$

Hier wird versucht, ausgehend vom erhaltenen Ergebnis \tilde{y} ein $x + \delta_x$ zu finden, das mit dem exakten Algorithmus φ das Ergebnis \tilde{y} liefert. Auch hier bekommt man in der Regel nur Abschätzungen für $\|\delta_x\|$. Kann man zeigen, dass das so geschätzte δ_x in der Größenordnung des tatsächlichen Datenfehlers liegt, also

$$\|\delta_x\| \approx \|x\| \tau$$

gilt, so ist der Algorithmus gutartig.

Viele Beispiele zu dieser Betrachtungsweise findet man in [Wil 94] und [Wil 88].

Numerische Stabilität

In der Literatur ist dieser Begriff nicht eindeutig definiert, da er immer nur relativ zur Genauigkeit ist.

Ein Versuch: Ist bei einem numerischen Verfahren der zu erwartende Genauigkeitsverlust (Konditionszahlen!) durch Rundungsfehlerfortpflanzung groß relativ zur Genauigkeit der Daten und zur Genauigkeitsanforderung (-erwartung) an das mathematische Problem, so spricht man von numerischer Instabilität.

Besser lässt sich numerische Stabilität komparativ erklären (nach [Sto 07]):

“Ein Algorithmus I zur Lösung eines Problems ist numerisch stabiler als ein Algorithmus II zur Lösung desselben Problems, wenn der Gesamteinfluss der Rundungsfehler bei I kleiner ist als bei II.”

Beispiel 1.6. Numerische 0/0-Situationen

Von numerischer 0/0-Situation spricht man, wenn sowohl Zähler als auch Nenner eines Quotienten von sehr kleinem Betrage sind. Es gibt zwei typische Situationen für solche Fälle, die sich aber bezüglich ihrer numerischen Stabilität stark unterscheiden.

(1)

$$y = \frac{x_1 \cdot x_2}{x_3 \cdot x_4} \quad \text{mit } x_1 \cdot x_2 \text{ und } x_3 \cdot x_4 \text{ klein.}$$

Mit Satz 1.8 ergibt sich als relativer Fehler

$$\varepsilon_y \doteq (\varepsilon_{x_1} + \varepsilon_{x_2}) - (\varepsilon_{x_3} + \varepsilon_{x_4}).$$

Dies ist also eine völlig harmlose Situation!

(2)

$$y = \frac{x_1 - x_2}{x_3 - x_4} \quad \text{mit } x_1 \approx x_2, x_3 \approx x_4.$$

Jetzt ergibt Satz 1.8

$$\varepsilon_y \doteq \left(\frac{x_1}{x_1 - x_2} \varepsilon_{x_1} - \frac{x_2}{x_1 - x_2} \varepsilon_{x_2} \right) - \left(\frac{x_3}{x_3 - x_4} \varepsilon_{x_3} - \frac{x_4}{x_3 - x_4} \varepsilon_{x_4} \right).$$

Wegen $x_1 \approx x_2$ und $x_3 \approx x_4$ wird der relative Fehler hier groß relativ zu den Eingangsdaten. Im Gegensatz zu (1) ist dies deshalb die schlimme Situation der Auslöschung, die in einem Algorithmus unbedingt vermieden werden sollte! \triangle

Beispiel 1.7. Rundungsfehler bei Summation

$$S = \sum_{i=0}^n a_i$$

Algorithmus:

$$\begin{aligned} S_0 &:= a_0, \\ S_i &:= S_{i-1} + a_i \text{ für } i = 1, \dots, n, \\ S &:= S_n. \end{aligned}$$

Absoluter Fehler:

$$\begin{aligned} \delta_{S_i} &= \delta_{S_{i-1}} + \delta_{a_i} + \tau_i \text{ mit } |\tau_i| \leq \tau |S_i| \\ \implies \delta_S &= \delta_{S_n} = \sum_{i=0}^n \delta_{a_i} + \sum_{i=1}^n \tau_i. \end{aligned}$$

Sei jetzt noch $|\delta_{a_i}| \leq \tau |a_i|$ (Summanden nur gerundet), dann ist

$$|\delta_S| \leq \tau \left(\sum_{i=0}^n |a_i| + \sum_{i=1}^n |S_i| \right).$$

Daraus ergibt sich der relative Fehler

$$|\varepsilon_S| \leq \frac{\tau}{|S|} \left(\sum_{i=0}^n |a_i| + \sum_{i=1}^n |S_i| \right).$$

Der Rundungsfehler bei Summation wird also besonders groß bei betragsmäßig großen Summanden, aber kleiner Summe (Auslöschung!).

Der Fehler bleibt kleiner bei kleineren Zwischensummen, ist z.B. $a_0 > a_1 > a_2 \dots > 0$, so liefert folgender Algorithmus kleinere Fehler:

$$\begin{aligned} S_0 &:= a_n, \\ S_i &:= S_{i-1} + a_{n-i} \text{ für } i = 1, \dots, n, \\ S &:= S_n. \end{aligned}$$

△

Beispiel 1.8. Berechnung der e-Funktion für sehr kleine negative Werte:

$$e^{-x} \approx \sum_{i=0}^N (-1)^i \frac{x^i}{i!}.$$

Dabei seien $x \gg 0$ und N so groß, dass der Abbruchfehler vernachlässigt werden kann. Die Summanden wachsen betragsmäßig zunächst an bis $x \approx i$, dann wächst $i!$ stärker als x^i . Sei k der Index zum betragsgrößten Term, dann gilt

$$\frac{x^k}{k!} \geq \frac{x^{k+1}}{(k+1)!} \Rightarrow \frac{x}{k+1} \leq 1 \Rightarrow x \approx k.$$

Dieser Term hat also etwa die Größenordnung

$$\frac{x^x}{x!} \approx \frac{e^x}{\sqrt{2\pi x}} \quad (\text{Stirling'sche Formel}).$$

Allein dieser Term liefert also einen relativen Fortpflanzungsfehler-Anteil von

$$\left(\tau \frac{e^x}{\sqrt{2\pi x}} \right) / e^{-x} = \frac{\tau e^{2x}}{\sqrt{2\pi x}}$$

Wegen $x \gg 0$ ist dieser Ausdruck sehr groß! Berechnen wir mit demselben x statt e^{-x} das Reziproke e^x , dann sind alle Fortpflanzungsfehlerfaktoren ≤ 1 :

$$\tau \frac{x^k}{k!} / e^x < \tau$$

Das kann man ausnutzen, indem man algorithmisch $1/e^x$ statt e^{-x} berechnet. Wir wollen den Effekt an einem Beispiel demonstrieren.

Sei $x = 10$, $\tau = 5 \cdot 10^{-10}$. Dann ist

$$\frac{e^{2x}}{\sqrt{2\pi x}} \doteq 6.1 \cdot 10^7$$

Deshalb erwarten wir einen relativen Fehler von etwa 3%. Einige Zwischensummen sind in Tab. 1.2 zu finden, die Endwerte mit acht wesentlichen Stellen; der entsprechend gerundete exakte Wert ist $e^{-10} = 4.5399930 \cdot 10^{-5}$.

Die Originalsumme für e^{-x} links verändert sich nach 47 Summanden nicht mehr. Der Wert hat den erwarteten relativen Fehler von etwa 3%. Die Summe für e^x verändert sich schon nach 32 Summanden nicht mehr. Ihr Reziprokwert stimmt in allen angezeigten Stellen mit dem exakten Wert überein. Der Grund für die mangelnde numerische Stabilität der Originalsumme liegt wieder in der Auslöschung, die durch das Summieren von betragsmäßig großen Summanden zu einem kleinen Summenwert entsteht. \triangle

Tab. 1.2 Zwischensummen der Reihenentwicklung für e^{-x} und e^x .

k	$\exp(-x) \approx \sum_{i=0}^k (-1)^i \frac{x^i}{i!}$	$\exp(x) \approx \sum_{i=0}^k \frac{x^i}{i!}$
0	1.0	1.0
1	-9.0	11.0
2	41.0	61.0
3	-125.6667	227.6667
4	291.0	644.3333
10	1342.587	12842.31
20	54.50004	21991.48
30	0.0009717559	22026.464
32	0.0001356691	22026.466
40	0.000046816665	
47	0.000046814252	

1.6 Software

Software zur Fehlertheorie? Nicht unbedingt, aber es gibt wichtige Funktionen in den guten, großen Bibliotheken wie NAG oder IMSL, die sich auf den Bereich der Rechengenauigkeit

beziehen. So liefert ein Funktionsaufruf den genauest möglichen Näherungswert für die nicht exakt berechenbare Zahl π , ein anderer gibt die Maschinengenauigkeit τ (1.7) aus. Diese und andere *Maschinenkonstanten* findet man bei NAG im Kapitel X.

Die meisten Programmiersprachen kennen für den Bereich der reellen Zahlen unterschiedliche Genauigkeiten, meistens spricht man von einfacher Genauigkeit, wenn $\tau \approx 10^{-8}$, und von doppelter Genauigkeit, wenn $\tau \approx 10^{-16}$ ist.

Auch in MATLAB gibt es vordefinierte Maschinenkonstanten, u.a. die Maschinengenauigkeit $\tau = \text{eps}$ sowie die größte und die kleinste positive Gleitpunktzahl `realmax` und `realmin`.

1.7 Aufgaben

Aufgabe 1.1. Der Abstand zwischen einer normalisierten Gleitpunktzahl x und einer benachbarten normalisierten Gleitpunktzahl y , ($x, y \in \mathcal{M}$) beträgt mindestens $2E^{-1}\tau$ und höchstens $2\tau|x|$ mit der Maschinengenauigkeit τ .

Aufgabe 1.2. Gegeben seien die mathematisch äquivalenten Ausdrücke

$$f(x) = 1 - \frac{1-x}{1+x} \quad \text{und} \quad g(x) = \frac{2x}{1+x}$$

Werten Sie diese Ausdrücke an der Stelle $x = 1/7654321$ aus, wobei Sie mit 5 wesentlichen Stellen rechnen. Bestimmen Sie jeweils die Größenordnung der relativen Fehler. Führen Sie anschließend das gleiche nochmal mit 8 wesentlichen Stellen durch.

Aufgabe 1.3. Für die durch $f(x) := e^x, x \in \mathbb{R}$, erklärte Funktion f sind

$$d_1(h) = \frac{1}{h}(e^{1+h} - e) \quad \text{und} \quad d_2(h) = \frac{1}{2h}(e^{1+h} - e^{1-h})$$

Näherungen für die Ableitung $f'(1)$ ($0 \neq h \in \mathbb{R}$).

Schreiben Sie eine Prozedur, um $d_1(h)$ und $d_2(h)$ zu berechnen. Lassen Sie $d_1(h)$, $d_2(h)$ und die absoluten Fehler $|d_1(h) - f'(1)|$, $|d_2(h) - f'(1)|$ ausgeben für

$$h := 10^{-i}, \quad i = 0, 1, 2, \dots, 18,$$

und wagen Sie eine Erklärung.

(Hinweis: Betrachten Sie die Taylorentwicklung von f und drücken Sie d_1 und d_2 damit aus.)

Aufgabe 1.4. Albert E. möchte ein Tafelwerk erstellen, das Cosinus-Werte für alle Argumente

$$\{x \in [0, 2\pi] \mid x \text{ hat drei Stellen nach dem Komma}\}$$

enthält. Diese Cosinus-Werte sollen auf vier Stellen nach dem Komma gerundet sein. Er will dazu alle Werte zunächst mit doppelter Genauigkeit (acht Stellen) berechnen, und anschließend auf vier Stellen runden.

Erklären Sie Albert am Beispiel

$$\cos(0.614) = 0.8173499969 \dots,$$

dass sein Vorgehen schief gehen kann, indem Sie

$$\text{rd}_t(\cos(0.614)) \quad \text{und} \quad \text{rd}_4(\text{rd}_t(\cos(0.614)))$$

für $t = 4, \dots, 9$ berechnen.

Aufgabe 1.5. Es soll eine Näherung für $\pi \approx 3.141592653589793238462643383305$ über die Beziehung

$$\pi = \lim_{n \rightarrow \infty} 3 \cdot 2^{n-1} s_n = \lim_{n \rightarrow \infty} 3 \cdot 2^{n-1} t_n$$

mit

$$(*) \quad s_1 := 1 \quad \text{und} \quad s_{n+1} := \sqrt{2 - \sqrt{4 - s_n^2}} \quad n = 1, 2, \dots,$$

bzw.

$$(**) \quad t_1 := 1 \quad \text{und} \quad t_{n+1} := \frac{t_n}{\sqrt{2 + \sqrt{4 - t_n^2}}} \quad n = 1, 2, \dots$$

bestimmt werden.

a) Beweisen Sie Rekursionsformel (*) mit Hilfe des Satzes von Pythagoras.

(Hinweis: s_n ist die Länge einer Seite eines regelmäßigen $3 \cdot 2^n$ -Ecks, das dem Einheitskreis einbeschrieben ist.)

b) Zeigen Sie $s_n = t_n$ für alle $n \in \mathbb{N}$.

c) Betrachten Sie die Funktion

$$\varphi(x) := \sqrt{2 - \sqrt{4 - x^2}}.$$

Wie ist das Problem der Berechnung von $\varphi(x)$ konditioniert, insbesondere für kleine Werte $x \neq 0$?

d) Schreiben Sie ein Programm, das die Werte

$$3 \cdot 2^{n-1} s_n \quad \text{und} \quad 3 \cdot 2^{n-1} t_n$$

für $n = 1, \dots, 35$ ausgibt und überlegen Sie sich den Grund für das unterschiedliche Verhalten der beiden Berechnungsarten.

Aufgabe 1.6. Es sei p ein reelles Polynom mit der einfachen Nullstelle $\xi_0 \in \mathbb{R}$, und g ein reelles Störpolynom für p , d.h. für kleines $\epsilon \in \mathbb{R}$ betrachten wir das gestörte Polynom

$$p_\epsilon := p + \epsilon g.$$

a) Zeigen Sie, dass es eine (Intervall-)Umgebung U von Null und eine stetig differenzierbare Funktion $\xi : U \rightarrow \mathbb{R}$ gibt mit

- $\xi(0) = \xi_0$,
- $\xi(\epsilon)$ ist Nullstelle von p_ϵ für alle $\epsilon \in U$, also

$$p(\xi(\epsilon)) + \epsilon g(\xi(\epsilon)) = 0 \quad \text{für alle } \epsilon \in U.$$

• $\xi(\epsilon)$ ist einfache Nullstelle von p_ϵ für alle $\epsilon \in U$.

b) Zeigen Sie, dass für $\xi(\cdot)$ auf U in erster Näherung die Formel

$$\xi(\epsilon) \doteq \xi_0 - \frac{g(\xi_0)}{p'(\xi_0)} \epsilon$$

gilt.

2 Lineare Gleichungssysteme, direkte Methoden

Die numerische Lösung von linearen Gleichungssystemen spielt eine zentrale Rolle in der Numerik. Viele Probleme der angewandten Mathematik führen fast zwangsläufig auf diese Aufgabe, oder sie wird als Hilfsmittel benötigt im Rahmen anderer Methoden. Deshalb stellen wir im Folgenden direkte Lösungsverfahren für lineare Gleichungssysteme bereit und beachten die Tatsache, dass jede Rechnung nur mit endlicher Genauigkeit erfolgt. Zuerst wird ein allgemeiner Algorithmus entwickelt, für dessen Anwendbarkeit nur die Regularität der Koeffizientenmatrix vorausgesetzt wird. Für Gleichungssysteme mit speziellen Eigenschaften, die häufig anzutreffen sind, ergeben sich Modifikationen und Vereinfachungen des Rechenverfahrens.

Ist die Koeffizientenmatrix eines linearen Gleichungssystems nicht regulär oder liegt ein über- oder unterbestimmtes System vor, bei dem also die Koeffizientenmatrix auch nicht mehr quadratisch ist, wird im Allgemeinen auf die Methode der kleinsten Quadrate zurückgegriffen, siehe Kapitel 6.

Eine andere Spezialsituation ist die sehr großer linearer Gleichungssysteme, deren Koeffizientenmatrix aber viele Nullen enthält; sie ist *schwach besetzt*. In diesem Fall werden meistens iterative Methoden verwendet, siehe Kapitel 11.

2.1 Der Gauß-Algorithmus

2.1.1 Elimination, Dreieckszerlegung und Determinantenberechnung

Es sei ein lineares Gleichungssystem

$$\sum_{k=1}^n a_{ik} x_k = b_i, \quad i = 1, 2, \dots, n, \tag{2.1}$$

mit n Gleichungen in n Unbekannten x_k zu lösen. Dabei seien die Koeffizienten a_{ik} und die Konstanten b_i als Zahlen vorgegeben und die numerischen Werte der Unbekannten x_k gesucht. Das Gleichungssystem (2.1) lässt sich in Matrizenform kurz als

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.2}$$

mit der quadratischen $(n \times n)$ -Matrix \mathbf{A} , der rechten Seite \mathbf{b} und dem Vektor \mathbf{x} der Unbekannten schreiben. Für das Folgende setzen wir voraus, dass die Koeffizientenmatrix \mathbf{A} regulär sei, so dass die Existenz und Eindeutigkeit der Lösung \mathbf{x} von (2.2) gesichert ist [Bun 95, Sta 94].

Zur Vereinfachung der Schreibweise und im Hinblick auf die praktische Durchführung auf einem Rechner benutzen wir für die zu lösenden Gleichungen (2.1) eine schematische Darstellung, die im konkreten Fall $n = 4$ die folgende selbsterklärende Form besitzt.

$$\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & 1 \\ \hline a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & b_2 \\ a_{31} & a_{32} & a_{33} & a_{34} & b_3 \\ a_{41} & a_{42} & a_{43} & a_{44} & b_4 \end{array} \quad (2.3)$$

Im Schema (2.3) erscheinen die Koeffizienten a_{ik} der Matrix \mathbf{A} und die Komponenten b_i der rechten Seite \mathbf{b} . Auf das Schema (2.3) dürfen die folgenden drei Äquivalenzoperationen angewandt werden, welche die im gegebenen Gleichungssystem (2.1) enthaltene Information nicht verändern:

1. Vertauschung von Zeilen;
2. Multiplikation einer Zeile mit einer Zahl $\neq 0$;
3. Addition eines Vielfachen einer Zeile zu einer anderen.

Unter der Annahme $a_{11} \neq 0$ subtrahieren wir von den i -ten Zeilen mit $i \geq 2$ das (a_{i1}/a_{11}) -fache der ersten Zeile und erhalten aus (2.3)

$$\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & 1 \\ \hline a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} & b_3^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} & b_4^{(1)} \end{array} \quad (2.4)$$

Mit den Quotienten

$$l_{i1} = a_{i1}/a_{11}, \quad i = 2, 3, \dots, n, \quad (2.5)$$

sind die Elemente in (2.4) gegeben durch

$$a_{ik}^{(1)} = a_{ik} - l_{i1}a_{1k}, \quad i, k = 2, 3, \dots, n, \quad (2.6)$$

$$b_i^{(1)} = b_i - l_{i1}b_1, \quad i = 2, 3, \dots, n. \quad (2.7)$$

Das Schema (2.4) entspricht einem zu (2.1) äquivalenten Gleichungssystem. Die erste Gleichung enthält als einzige die Unbekannte x_1 , die sich somit durch die übrigen Unbekannten ausdrücken lässt gemäß

$$x_1 = \left[b_1 - \sum_{k=2}^n a_{1k}x_k \right] / a_{11}. \quad (2.8)$$

Weiter enthält (2.4) im allgemeinen Fall ein reduziertes System von $(n - 1)$ Gleichungen für die $(n - 1)$ Unbekannten x_2, x_3, \dots, x_n . Der Übergang von (2.3) nach (2.4) entspricht

somit einem *Eliminationsschritt*, mit welchem die Auflösung des gegebenen Systems (2.1) in n Unbekannten auf die Lösung eines Systems in $(n - 1)$ Unbekannten zurückgeführt ist. Wir werden dieses reduzierte System analog weiterbehandeln, wobei die erste Zeile in (2.4) unverändert bleibt. Man bezeichnet sie deshalb als erste Endgleichung, und a_{11} , um welches sich der beschriebene Eliminationsschritt gewissermaßen dreht, als *Pivotelement*.

Unter der weiteren Annahme $a_{22}^{(1)} \neq 0$ ergibt sich aus (2.4) als Resultat eines zweiten Eliminationsschrittes

$$\begin{array}{ccccc|c} x_1 & x_2 & x_3 & x_4 & 1 \\ \hline a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} & b_3^{(2)} \\ 0 & 0 & a_{43}^{(2)} & a_{44}^{(2)} & b_4^{(2)} \end{array} \quad (2.9)$$

Mit den Hilfsgrößen

$$l_{i2} = a_{i2}^{(1)} / a_{22}^{(1)}, \quad i = 3, 4, \dots, n, \quad (2.10)$$

lauten die neuen Elemente in (2.9)

$$a_{ik}^{(2)} = a_{ik}^{(1)} - l_{i2} a_{2k}^{(1)}, \quad i, k = 3, 4, \dots, n, \quad (2.11)$$

$$b_i^{(2)} = b_i^{(1)} - l_{i2} b_2^{(1)}, \quad i = 3, 4, \dots, n. \quad (2.12)$$

Das Schema (2.9) enthält die zweite Endgleichung für x_2

$$x_2 = \left[b_2^{(1)} - \sum_{k=3}^n a_{2k}^{(1)} x_k \right] / a_{22}^{(1)}. \quad (2.13)$$

Die konsequente Fortsetzung der Eliminationsschritte führt nach $(n - 1)$ Schritten zu einem Schema, welches lauter Endgleichungen enthält. Um die Koeffizienten der Endgleichungen einheitlich zu bezeichnen, definieren wir

$$a_{ik}^{(0)} := a_{ik}, \quad i, k = 1, 2, \dots, n; \quad b_i^{(0)} := b_i, \quad i = 1, 2, \dots, n, \quad (2.14)$$

$$\left. \begin{aligned} r_{ik} &:= a_{ik}^{(i-1)}, & k = i, i+1, \dots, n, \\ c_i &:= b_i^{(i-1)}, \end{aligned} \right\} \quad i = 1, 2, \dots, n. \quad (2.15)$$

Damit lautet das Schema der Endgleichungen

$$\begin{array}{ccccc|c} x_1 & x_2 & x_3 & x_4 & 1 \\ \hline r_{11} & r_{12} & r_{13} & r_{14} & c_1 \\ 0 & r_{22} & r_{23} & r_{24} & c_2 \\ 0 & 0 & r_{33} & r_{34} & c_3 \\ 0 & 0 & 0 & r_{44} & c_4 \end{array} \quad (2.16)$$

Aus (2.16) lassen sich die Unbekannten in der Reihenfolge $x_n, x_{n-1}, \dots, x_2, x_1$ gemäß der Rechenvorschrift

$$x_i = \left[c_i - \sum_{k=i+1}^n r_{ik} x_k \right] / r_{ii}, \quad i = n, n-1, \dots, 2, 1, \quad (2.17)$$

berechnen. Man nennt den durch (2.17) beschriebenen Prozess *Rücksubstitution*, da die Endgleichungen in umgekehrter Reihenfolge ihrer Entstehung verwendet werden.

Mit dem dargestellten Rechenverfahren sind die wesentlichen Elemente des Gaußschen Algorithmus erklärt. Er leistet die Reduktion eines gegebenen linearen Gleichungssystems $\mathbf{A}\mathbf{x} = \mathbf{b}$ auf ein System $\mathbf{R}\mathbf{x} = \mathbf{c}$ gemäß (2.16), wo \mathbf{R} eine *Rechtsdreiecksmatrix*

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ 0 & 0 & r_{33} & \cdots & r_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & r_{nn} \end{pmatrix} \quad (2.18)$$

mit von null verschiedenen Diagonalelementen ist, aus dem sich die Unbekannten unmittelbar ermitteln lassen.

Beispiel 2.1. Zu lösen sei das Gleichungssystem

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= -10 \\ 4x_1 + 8x_2 - 3x_3 &= -19 \\ -6x_1 + x_2 + 4x_3 &= -11 \end{aligned}$$

Der Gauß-Algorithmus liefert in zwei Eliminationsschritten

x_1	x_2	x_3	1
2	3	-5	-10
4	8	-3	-19
-6	1	4	-11

x_1	x_2	x_3	1
2	3	-5	-10
0	2	7	1
0	10	-11	-41

x_1	x_2	x_3	1
2	3	-5	-10
0	2	7	1
0	0	-46	-46

Die Rücksubstitution ergibt für die Unbekannten gemäß (2.17) $x_3 = 1$, $x_2 = (1 - 7 \cdot 1) / 2 = -3$ und $x_1 = (-10 - 3 \cdot (-3) + 5) / 2 = 2$. \triangle

Bei rechnerischer Durchführung des Gaußschen Algorithmus werden die Zahlenwerte der sukzessiven Schemata selbstverständlich in einem festen Feld gespeichert, so dass am Schluss das System der Endgleichungen (2.16) verfügbar ist. Es ist dabei nicht sinnvoll, das Feld unterhalb der Diagonale mit Nullen aufzufüllen. Aus einem bald ersichtlichen Grund ist es angezeigt, an den betreffenden Stellen die Werte der Quotienten l_{ik} zu speichern, sodass wir

anstelle von (2.16) das folgende Schema erhalten.

$$\begin{array}{ccccc|c}
 x_1 & x_2 & x_3 & x_4 & 1 \\
 \hline
 r_{11} & r_{12} & r_{13} & r_{14} & c_1 \\
 l_{21} & r_{22} & r_{23} & r_{24} & c_2 \\
 l_{31} & l_{32} & r_{33} & r_{34} & c_3 \\
 l_{41} & l_{42} & l_{43} & r_{44} & c_4
 \end{array} \tag{2.19}$$

Nun wollen wir die im Schema (2.19) enthaltenen Größen mit denjenigen des Ausgangssystems (2.3) in Zusammenhang bringen, wobei die Entstehungsgeschichte zu berücksichtigen ist. Der Wert r_{ik} in der i -ten Zeile mit $i \geq 2$ und $k \geq i$ entsteht nach $(i-1)$ Eliminationsschritten gemäß (2.6), (2.11) etc. und (2.15)

$$\begin{aligned}
 r_{ik} = a_{ik}^{(i-1)} &= a_{ik} - l_{i1}a_{1k}^{(0)} - l_{i2}a_{2k}^{(1)} - \dots - l_{i,i-1}a_{i-1,k}^{(i-2)} \\
 &= a_{ik} - l_{i1}r_{1k} - l_{i2}r_{2k} - \dots - l_{i,i-1}r_{i-1,k}, \quad i \geq 2, k \geq i.
 \end{aligned}$$

Daraus folgt die Beziehung

$$a_{ik} = \sum_{j=1}^{i-1} l_{ij}r_{jk} + r_{ik}, \quad (k \geq i \geq 1), \tag{2.20}$$

die auch für $i = 1$ Gültigkeit behält, weil dann die Summe leer ist. Der Wert l_{ik} in der k -ten Spalte mit $k \geq 2$ und $i > k$ wird im k -ten Eliminationsschritt aus $a_{ik}^{(k-1)}$ erhalten gemäß

$$\begin{aligned}
 l_{ik} &= a_{ik}^{(k-1)} / a_{kk}^{(k-1)} = [a_{ik} - l_{i1}a_{1k}^{(0)} - l_{i2}a_{2k}^{(1)} - \dots - l_{i,k-1}a_{k-1,k}^{(k-2)}] / a_{kk}^{(k-1)} \\
 &= [a_{ik} - l_{i1}r_{1k} - l_{i2}r_{2k} - \dots - l_{i,k-1}r_{k-1,k}] / r_{kk}.
 \end{aligned}$$

Lösen wir diese Gleichung nach a_{ik} auf, so folgt die Beziehung

$$a_{ik} = \sum_{j=1}^k l_{ij}r_{jk}, \quad (i > k \geq 1), \tag{2.21}$$

die wegen (2.5) auch für $k = 1$ gültig ist. Die beiden Relationen (2.20) und (2.21) erinnern uns an die Regeln der Matrizenmultiplikation. Neben der Rechtsdreiecksmatrix \mathbf{R} (2.18) definieren wir noch die *Linksdreiecksmatrix* \mathbf{L} mit Einsen in der Diagonale

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{pmatrix}. \tag{2.22}$$

Dann sind (2.20) und (2.21) tatsächlich gleichbedeutend mit der Matrizengleichung

$$\mathbf{A} = \mathbf{LR}. \tag{2.23}$$

Satz 2.1. Es sei vorausgesetzt, dass für die Pivotelemente

$$a_{11} \neq 0, \quad a_{22}^{(1)} \neq 0, \quad a_{33}^{(2)} \neq 0, \dots, \quad a_{nn}^{(n-1)} \neq 0 \quad (2.24)$$

gilt. Dann leistet der Gaußsche Algorithmus die Produktzerlegung (Faktorisierung) einer regulären Matrix \mathbf{A} in eine Linksdreiecksmatrix \mathbf{L} und eine Rechtsdreiecksmatrix \mathbf{R} .

Für die c_i -Werte mit $i \geq 2$ gelten mit (2.7), (2.12) und (2.15)

$$\begin{aligned} c_i &= b_i^{(i-1)} = b_i - l_{i1}b_1 - l_{i2}b_2^{(1)} - \dots - l_{i,i-1}b_{i-1}^{(i-2)} \\ &= b_i - l_{i1}c_1 - l_{i2}c_2 - \dots - l_{i,i-1}c_{i-1}, \quad (i \geq 2). \end{aligned}$$

Daraus erhalten wir die Beziehungen

$$b_i = \sum_{j=1}^{i-1} l_{ij}c_j + c_i, \quad i = 1, 2, \dots, n, \quad (2.25)$$

die sich mit der Linksdreiecksmatrix \mathbf{L} in folgender Form zusammenfassen lassen

$$\mathbf{L}\mathbf{c} = \mathbf{b}. \quad (2.26)$$

Der Vektor \mathbf{c} im Schema (2.16) kann somit als Lösung des Gleichungssystems (2.26) interpretiert werden. Da \mathbf{L} eine Linksdreiecksmatrix ist, berechnen sich die Unbekannten c_i in der Reihenfolge c_1, c_2, \dots, c_n vermöge des Prozesses der *Vorwärtssubstitution* gemäß

$$c_i = b_i - \sum_{j=1}^{i-1} l_{ij}c_j, \quad i = 1, 2, \dots, n. \quad (2.27)$$

Auf Grund dieser Betrachtung kann die Lösung \mathbf{x} eines linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ unter der Voraussetzung (2.24) des Satzes 2.1 mittels des Gaußschen Algorithmus mit den drei Lösungsschritten berechnet werden:

1. $\mathbf{A} = \mathbf{LR}$ (Zerlegung von \mathbf{A})
2. $\mathbf{Lc} = \mathbf{b}$ (Vorwärtssubstitution $\rightarrow \mathbf{c}$)
3. $\mathbf{Rx} = \mathbf{c}$ (Rücksubstitution $\rightarrow \mathbf{x}$)

Wir wollen nun zeigen, dass die Voraussetzung (2.24) durch geeignete Zeilenumtauschungen, die allenfalls vor jedem Eliminationsschritt auszuführen sind, erfüllt werden kann, so dass das Verfahren theoretisch durchführbar ist.

Satz 2.2. Für eine reguläre Matrix \mathbf{A} existiert vor dem k -ten Eliminationsschritt des Gauß-Algorithmus stets eine Zeilenpermutation derart, dass das k -te Diagonalelement von null verschieden ist.

Beweis. Die vorausgesetzte Regularität der Matrix \mathbf{A} bedeutet, dass ihre Determinante $|\mathbf{A}| \neq 0$ ist. Die im Gauß-Algorithmus auf die Koeffizienten der Matrix \mathbf{A} angewandten Zeilenumoperationen bringen wir in Verbindung mit elementaren Operationen für die zugehörige Determinante.

Angenommen, es sei $a_{11} = 0$. Dann existiert mindestens ein $a_{i1} \neq 0$ in der ersten Spalte, denn andernfalls wäre die Determinante von \mathbf{A} im Widerspruch zur Voraussetzung gleich

null. Die Vertauschung der betreffenden i -ten Zeile mit der ersten Zeile erzeugt an der Stelle $(1, 1)$ ein Pivotelement ungleich null. Eine Zeilenvertauschung in einer Determinante hat einen Vorzeichenwechsel ihres Wertes zur Folge. Wir setzen deshalb $v_1 = 1$, falls vor dem ersten Eliminationsschritt eine Vertauschung von Zeilen nötig ist, und $v_1 = 0$, falls $a_{11} \neq 0$ ist. Die Addition von Vielfachen der ersten Zeile zu den folgenden Zeilen ändert bekanntlich den Wert einer Determinante nicht. Wenn wir zur Entlastung der Schreibweise die Matrixelemente nach einer eventuellen Zeilenvertauschung wieder mit a_{ik} bezeichnen, gilt (wieder für $n = 4$)

$$\begin{aligned} |\mathbf{A}| &= (-1)^{v_1} \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} = (-1)^{v_1} \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{vmatrix} \\ &= (-1)^{v_1} a_{11} \begin{vmatrix} a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{vmatrix} \end{aligned} \quad (2.29)$$

mit den in (2.6) definierten Größen.

Die Überlegungen für den ersten Eliminationsschritt übertragen sich sinngemäß auf die folgenden, reduzierten Systeme, bzw. ihre zugehörigen Determinanten. So existiert mindestens ein $a_{j2}^{(1)} \neq 0$, da andernfalls die dreireihige Determinante in (2.29) und damit $|\mathbf{A}|$ verschwinden würde. Eine Vertauschung der j -ten Zeile mit der zweiten Zeile bringt an der Stelle $(2,2)$ ein Pivotelement ungleich null. \square

Falls wir unter r_{kk} , $k = 1, 2, \dots, n - 1$, das nach einer eventuellen Zeilenvertauschung im k -ten Eliminationsschritt verwendete, in der Diagonale stehende Pivotelement verstehen, und r_{nn} als Pivot für den leeren n -ten Schritt bezeichnen, dann gilt als unmittelbare Folge der Beweisführung für den Satz 2.2 der

Satz 2.3. *Erfolgen im Verlauf des Gauß-Algorithmus insgesamt $V = \sum_{i=1}^{n-1} v_i$ Zeilenvertauschungen, dann ist die Determinante $|\mathbf{A}|$ gegeben durch*

$$|\mathbf{A}| = (-1)^V \prod_{k=1}^n r_{kk}. \quad (2.30)$$

Die Determinante $|\mathbf{A}|$ der Systemmatrix \mathbf{A} ist, abgesehen vom Vorzeichen, gleich dem Produkt der n Pivotelemente des Gauß-Algorithmus. Sie kann somit sozusagen als Nebenprodukt bei der Auflösung eines linearen Gleichungssystems bestimmt werden. Falls keine Zeilenvertauschungen notwendig sind, kann diese Aussage auch direkt aus der Zerlegung (2.23) mit $|\mathbf{L}| = 1$ gefolgert werden, denn dann gilt

$$|\mathbf{A}| = |\mathbf{LR}| = |\mathbf{L}| |\mathbf{R}| = \prod_{k=1}^n r_{kk}.$$

Der Gauß-Algorithmus stellt ganz unabhängig von der Auflösung eines Gleichungssystems ein zweckmäßiges und effizientes Verfahren zur Berechnung einer Determinante dar.

Aus (2.30) folgt noch die wichtige Tatsache, dass das Produkt der Beträge der Pivotelemente für ein gegebenes Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$ im Gauß-Algorithmus eine feste Größe, d.h. eine Invariante ist.

Der Satz 2.2 stellt die Durchführbarkeit des Gaußschen Algorithmus für eine reguläre Matrix \mathbf{A} sicher, falls in seinem Verlauf geeignete Zeilenumtauschungen vorgenommen werden. Die Zeilenumtauschungen, die ja erst auf Grund der anfallenden Zahlenwerte ausgeführt werden, können wir uns schon vor Beginn der Rechnung durchgeführt denken, so dass vor jedem Eliminationsschritt in der Diagonale ein von null verschiedenes Pivotelement verfügbar ist. Diese vorgängigen Zeilenumoperationen lassen sich formal durch eine geeignete *Permutationsmatrix* \mathbf{P} beschreiben. Dies ist eine quadratische Matrix, welche in jeder Zeile und in jeder Spalte genau eine Eins und sonst nur Nullen enthält, und deren Determinante $|\mathbf{P}| = \pm 1$ ist. Auf Grund dieser Betrachtung gilt als Folge der Sätze 2.1 und 2.2 der

Satz 2.4. Zu jeder regulären Matrix \mathbf{A} existiert eine Permutationsmatrix \mathbf{P} , so dass \mathbf{PA} in das Produkt einer Linksdreiecksmatrix \mathbf{L} (2.22) und einer Rechtsdreiecksmatrix \mathbf{R} (2.18) zerlegbar ist gemäß

$$\mathbf{PA} = \mathbf{LR}. \quad (2.31)$$

Die Aussage von Satz 2.4 ist selbstverständlich im Algorithmus (2.28) zu berücksichtigen. Das gegebene Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ ist von links mit der Permutationsmatrix \mathbf{P} zu multiplizieren: $\mathbf{PAx} = \mathbf{Pb}$, so dass die Lösung \mathbf{x} mittels des Gaußschen Algorithmus wie folgt berechnet werden kann:

1. $\mathbf{PA} = \mathbf{LR}$	(Zerlegung von \mathbf{PA})	
2. $\mathbf{Lc} = \mathbf{Pb}$	(Vorwärtssubstitution $\rightarrow \mathbf{c}$)	
3. $\mathbf{Rx} = \mathbf{c}$	(Rücksubstitution $\rightarrow \mathbf{x}$)	

(2.32)

Die Zeilenumpermutationen der Matrix \mathbf{A} sind vor dem Prozess der Vorwärtssubstitution auf die gegebene rechte Seite \mathbf{b} anzuwenden.

Abschließend wollen wir den Rechenaufwand zur Auflösung eines Systems von n linearen Gleichungen in n Unbekannten mit dem Gaußschen Algorithmus bestimmen. Dabei werden wir nur die Multiplikationen und Divisionen als wesentliche, ins Gewicht fallende Operationen ansehen und die Additionen vernachlässigen.

Im allgemeinen j -ten Eliminationsschritt zur Berechnung der Zerlegung $\mathbf{PA} = \mathbf{LR}$ werden gemäß (2.5) und (2.6) die $(n - j)$ Quotienten l_{ij} und die $(n - j)^2$ Werte $a_{ik}^{(j)}$ berechnet. Dazu sind offensichtlich $[(n - j) + (n - j)^2]$ wesentliche Operationen erforderlich, so dass der Rechenaufwand für die Zerlegung

$$\begin{aligned} Z_{LR} &= \{(n - 1) + (n - 2) + \dots + 1\} + \{(n - 1)^2 + (n - 2)^2 + \dots + 1^2\} \\ &= \frac{1}{2}n(n - 1) + \frac{1}{6}n(n - 1)(2n - 1) = \frac{1}{3}(n^3 - n) \end{aligned} \quad (2.33)$$

beträgt. Im Prozess der Vorwärtssubstitution benötigt die Berechnung von c_i auf Grund von (2.27) $(i-1)$ Multiplikationen, so dass sich der totale Rechenaufwand für die Vorwärtssubstitution zu

$$Z_V = \{1 + 2 + \dots + (n-1)\} = \frac{1}{2}n(n-1) = \frac{1}{2}(n^2 - n) \quad (2.34)$$

ergibt. Schließlich erfordert die Berechnung der x_i nach (2.17) $(n-i)$ Multiplikationen und eine Division, weshalb der Rechenaufwand für die Rücksubstitution

$$Z_R = \{1 + 2 + \dots + n\} = \frac{1}{2}n(n+1) = \frac{1}{2}(n^2 + n) \quad (2.35)$$

beträgt. Somit beläuft sich der Rechenaufwand für die beiden, in der Regel zusammen gehörigen Prozesse der Vorwärtss- und Rücksubstitution auf

$$Z_{VR} = n^2 \quad (2.36)$$

wesentliche Operationen. Der vollständige Gauß-Algorithmus zur Lösung von n linearen Gleichungen erfordert somit

$$Z_{\text{Gauß}} = \frac{1}{3}n^3 + n^2 - \frac{1}{3}n = O(n^3) \quad (2.37)$$

wesentliche Rechenoperationen. $O(n^3)$ bedeutet, dass sich der Aufwand und damit die Rechenzeit asymptotisch wie die dritte Potenz der Zahl der Unbekannten verhalten.

2.1.2 Pivotstrategien

Die Sätze 2.2 und 2.4 gewährleisten die Existenz eines von null verschiedenen Pivotelementes für die sukzessiven Eliminationsschritte. Sie lassen aber die konkrete Wahl offen. Für die numerische Durchführung des Gauß-Algorithmus ist die zweckmäßige Auswahl des Pivots von entscheidender Bedeutung für die Genauigkeit der berechneten Lösung. Zudem benötigt jedes Programm eine genau definierte Regel zur Bestimmung der Pivotelemente, die man als *Pivotstrategie* bezeichnet.

Werden die Pivotelemente sukzessive in der Diagonale gewählt, spricht man von *Diagonalstrategie*. In dieser einfachen Strategie werden keine Zeilenvertauschungen in Betracht gezogen. Dabei kann einerseits der Gauß-Algorithmus abbrechen, obwohl die Matrix A regulär ist, andererseits ist zu erwarten, dass diese Strategie nicht in jedem Fall numerisch brauchbar ist, obwohl die Pivotelemente theoretisch zulässig sind.

Beispiel 2.2. Zur Illustration der numerischen Konsequenzen, die ein betragsmäßig kleines Pivotelement nach sich zieht, betrachten wir ein Gleichungssystem in zwei Unbekannten. Es soll mit Diagonalstrategie gelöst werden, wobei die Rechnung mit fünf wesentlichen Dezimalstellen durchgeführt wird, siehe Definition 1.5. Ein Eliminationsschritt des Gauß-Algorithmus liefert

$$\begin{array}{ccc|c} x_1 & x_2 & 1 \\ \hline 0.00035 & 1.2654 & 3.5267 \\ 1.2547 & 1.3182 & 6.8541 \end{array} \longrightarrow \begin{array}{ccc|c} x_1 & x_2 & 1 \\ \hline 0.00035 & 1.2654 & 3.5267 \\ 0 & -4535.0 & -12636 \end{array}$$

Mit $l_{21} = 1.2547/0.00035 \doteq 3584.9$ ergeben sich die beiden Zahlenwerte der zweiten Zeile des zweiten Schemas zu $r_{22} = 1.3182 - 3584.9 \times 1.2654 \doteq 1.3182 - 4536.3 \doteq -4535.0$ und

$c_2 = 6.8541 - 3584.9 \times (3.5267) \doteq 6.8541 - 12643 \doteq -12636$. Der Prozess der Rücksubstitution liefert $x_2 = -12636/(-4535.0) \doteq 2.7863$ und $x_1 = (3.5267 - 1.2654 \times 2.7863)/0.00035 \doteq (3.5267 - 3.5258)/0.00035 = 0.0009/0.00035 \doteq 2.5714$. Durch das kleine Pivotelement sind im einzigen Eliminationsschritt große Zahlen $r_{22} = a_{22}^{(1)}$ und $c_2 = b_2^{(1)}$ entstanden. Ihre Division zur Berechnung von x_2 ist numerisch stabil, siehe Satz 1.8. Bei der Berechnung von x_1 ist eine Differenz von zwei fast gleich großen Zahlen zu bilden; hier tritt Auslöschung auf. Die Berechnung von x_1 ist daher numerisch instabil. Dies bestätigt der Vergleich mit den auf fünf Stellen gerundeten exakten Werten $x_1 \doteq 2.5354$ und $x_2 \doteq 2.7863$; x_2 stimmt in allen wesentlichen Stellen mit der exakten Lösung überein, während x_1 auf Grund der Auslöschung einen relativen Fehler von 1.5 % aufweist. \triangle

In einem Spezialfall von allgemeinen linearen Gleichungssystemen ist die Diagonalstrategie immer anwendbar und sogar numerisch sinnvoll.

Definition 2.5. Eine Matrix \mathbf{A} heißt *(strikt) diagonal dominant*, falls in jeder Zeile der Betrag des Diagonalelementes größer ist als die Summe der Beträge der übrigen Matrixelemente derselben Zeile, falls also gilt

$$|a_{ii}| > \sum_{\substack{k=1 \\ k \neq i}}^n |a_{ik}|, \quad i = 1, 2, \dots, n. \quad (2.38)$$

Sie heißt *schwach diagonal dominant*, wenn

$$|a_{ii}| \geq \sum_{\substack{k=1 \\ k \neq i}}^n |a_{ik}|, \quad i = 1, 2, \dots, n, \quad (2.39)$$

wobei aber für mindestens einen Index i_0 in (2.39) strikte Ungleichheit gilt.

Satz 2.6. Zur Lösung eines regulären Gleichungssystems mit schwach diagonal dominanter Matrix \mathbf{A} ist die Diagonalstrategie anwendbar.

Beweis. Sei ohne Einschränkung die erste Zeile diejenige, für die die strikte Ungleichheit $|a_{11}| > \sum_{k=2}^n |a_{1k}| \geq 0$ gilt. Folglich ist $a_{11} \neq 0$ ein zulässiges Pivotelement für den ersten Eliminationsschritt. Wir zeigen nun, dass sich die Eigenschaft der diagonalen Dominanz auf das reduzierte Gleichungssystem überträgt. Nach Substitution von (2.5) in (2.6) gilt für die reduzierten Elemente

$$a_{ik}^{(1)} = a_{ik} - \frac{a_{i1}a_{1k}}{a_{11}}, \quad i, k = 2, 3, \dots, n. \quad (2.40)$$

Für die Diagonalelemente folgt daraus die Abschätzung

$$|a_{ii}^{(1)}| = \left| a_{ii} - \frac{a_{i1}a_{1i}}{a_{11}} \right| \geq |a_{ii}| - \left| \frac{a_{i1}a_{1i}}{a_{11}} \right|, \quad i = 2, 3, \dots, n. \quad (2.41)$$

Die Summe der Beträge der Nicht-Diagonalelemente der i -ten Zeile, $i = 2, 3, \dots, n$, des reduzierten Systems erfüllt unter Verwendung der Voraussetzung (2.38) und der Abschätzung (2.41) in der Tat die Ungleichung

$$\begin{aligned} \sum_{\substack{k=2 \\ k \neq i}}^n |a_{ik}^{(1)}| &= \sum_{\substack{k=2 \\ k \neq i}}^n \left| a_{ik} - \frac{a_{i1}a_{1k}}{a_{11}} \right| \leq \sum_{\substack{k=2 \\ k \neq i}}^n |a_{ik}| + \left| \frac{a_{i1}}{a_{11}} \right| \sum_{\substack{k=2 \\ k \neq i}}^n |a_{1k}| \\ &= \sum_{\substack{k=1 \\ k \neq i}}^n |a_{ik}| - |a_{i1}| + \left| \frac{a_{i1}}{a_{11}} \right| \left\{ \sum_{k=2}^n |a_{1k}| - |a_{1i}| \right\} \\ &\leq |a_{ii}| - |a_{i1}| + \left| \frac{a_{i1}}{a_{11}} \right| \{|a_{11}| - |a_{1i}|\} = |a_{ii}| - \left| \frac{a_{i1}a_{1i}}{a_{11}} \right| \leq |a_{ii}^{(1)}|. \end{aligned}$$

Wegen $|a_{22}^{(1)}| \geq \sum_{k=3}^n |a_{2k}^{(1)}| \geq 0$ ist $|a_{22}^{(1)}| > 0$, denn sonst wären alle Elemente der zweiten Zeile gleich null, und das widerspräche der vorausgesetzten Regularität des Systems. Also kann $a_{22}^{(1)}$ als Pivotelement gewählt werden, und die Diagonalstrategie ist tatsächlich möglich. \square

Da das Pivotelement von null verschieden sein muss, besteht eine naheliegende Auswahlregel darin, unter den in Frage kommenden Elementen das absolut grösste als Pivot zu wählen. Man spricht in diesem Fall von *Spaltenmaximumstrategie*. Vor Ausführung des k -ten Eliminationsschrittes bestimmt man den Index p so, dass gilt

$$\max_{i \geq k} |a_{ik}^{(k-1)}| = |a_{pk}^{(k-1)}|. \quad (2.42)$$

Falls $p \neq k$ ist, so ist die p -te Zeile mit der k -ten zu vertauschen. Mit dieser Strategie erreicht man, dass die Quotienten $l_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}$ ($i > k$) betragsmäßig durch Eins beschränkt sind. Folglich sind die Faktoren, mit denen die aktuelle k -te Zeile zu multiplizieren ist, dem Betrag nach kleiner oder gleich Eins, was sich auf die Fortpflanzung von Rundungsfehlern günstig auswirken kann.

Beispiel 2.3. Wenn wir das Gleichungssystem vom Beispiel 2.2 mit der Spaltenmaximumstrategie behandeln, so muss vor dem ersten Schritt eine Zeilenumtauschung vorgenommen werden. Bei fünfstelliger Rechnung lauten die Schemata

$$\begin{array}{ccc|c} x_1 & x_2 & 1 \\ \hline 1.2547 & 1.3182 & 6.8541 \\ 0.00035 & 1.2654 & 3.5267 \end{array} \longrightarrow \begin{array}{ccc|c} x_1 & x_2 & 1 \\ \hline 1.2547 & 1.3182 & 6.8541 \\ 0 & 1.2650 & 3.5248 \end{array}$$

Der Quotient $l_{21} = 0.00027895$ ist sehr klein und bewirkt nur geringe Änderungen in den Elementen des reduzierten Schemas. Die Rücksubstitution ergibt nacheinander $x_2 \doteq 2.7864$ und $x_1 = (6.8541 - 1.3182 \times 2.7864) / 1.2547 \doteq (6.8541 - 3.6730) / 1.2547 = 3.1811 / 1.2547 \doteq 2.5353$. Die beiden Lösungswerte weichen nur um je eine Einheit in der letzten Stelle von den richtigen, gerundeten Werten ab. Die Spaltenmaximumstrategie hat die Situation tatsächlich verbessert. \triangle

Beispiel 2.4. Um eine Schwäche der Spaltenmaximumstrategie aufzuzeigen, betrachten wir das folgende Gleichungssystem, das mit fünfstelliger Rechnung gelöst wird und dessen Zeilen so angeordnet wurden, dass die Spaltenmaximumstrategie zur Diagonalstrategie wird. Im zweiten und dritten Schema sind anstelle der Nullen die l -Werte eingesetzt.

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & 1 \\ \hline 2.1 & 2512 & -2516 & 6.5 \\ -1.3 & 8.8 & -7.6 & -5.3 \\ 0.9 & -6.2 & 4.6 & 2.9 \end{array} \rightarrow \begin{array}{cccc|c} x_1 & x_2 & x_3 & 1 \\ \hline 2.1 & 2512 & -2516 & 6.5 \\ -0.61905 & 1563.9 & -1565.1 & -1.2762 \\ 0.42857 & -1082.8 & 1082.9 & 0.11430 \end{array}$$

$$\rightarrow \begin{array}{cccc|c} x_1 & x_2 & x_3 & 1 \\ \hline 2.1 & 2512 & -2516 & 6.5 \\ -0.61905 & 1563.9 & -1565.1 & -1.2762 \\ 0.42857 & -0.69237 & -0.70000 & -0.76930 \end{array}$$

Daraus berechnen sich sukzessive die Lösungen $x_3 \doteq 1.0990$, $x_2 \doteq 1.0990$, $x_1 \doteq 5.1905$, während die exakten Werte $x_3 = x_2 = 1$ und $x_1 = 5$ sind. Die Abweichungen erklären sich mit der Feststellung, dass der erste Eliminationsschritt betragsmäßig große Koeffizienten $a_{ik}^{(1)}$ des reduzierten Schemas erzeugt, womit bereits ein Informationsverlust infolge Rundung eingetreten ist. Zudem ist im zweiten Schritt bei der Berechnung von $a_{33}^{(2)}$ eine katastrophale Auslöschung festzustellen. Der Grund für das schlechte Ergebnis liegt darin, dass das Pivotelement des ersten Eliminationsschrittes klein ist im Vergleich zum Maximum der Beträge der übrigen Matrixelemente der ersten Zeile. Nach (2.40) gehen die Elemente a_{i1} und a_{1k} in symmetrischer Weise in die Reduktionsformel ein. Dieser Feststellung muss Rechnung getragen werden. \triangle

Eine einfache Maßnahme, die Situation zu verbessern, besteht darin, die gegebenen Gleichungen so zu *skalieren*, dass für die neuen Koeffizienten \tilde{a}_{ik} gilt

$$\sum_{k=1}^n |\tilde{a}_{ik}| \approx 1, \quad i = 1, 2, \dots, n. \quad (2.43)$$

Um zusätzliche Rundungsfehler zu vermeiden, wählen wir für die Skalierung nur Faktoren, die Potenzen der Zahlenbasis des Rechners sind, im Dualsystem also Zweierpotenzen.

Durch die Skalierung mit $|\tilde{a}_{ik}| \leq 1$, $i, k = 1, 2, \dots, n$, werden bei der anschließenden Spaltenmaximumstrategie andere Pivotelemente ausgewählt, und diese Auswahl führt zu einer numerisch stabileren Rechnung, wie in Beispiel 2.5 zu sehen ist.

Beispiel 2.5. Die Gleichungen von Beispiel 2.4 lauten nach ihrer Skalierung bei fünfstelliger Rechengenauigkeit

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & 1 \\ \hline 0.00041749 & 0.49939 & -0.50019 & 0.0012922 \\ -0.073446 & 0.49718 & -0.42938 & -0.29944 \\ 0.076923 & -0.52991 & 0.39316 & 0.24786 \end{array}$$

Die Spaltenmaximumstrategie bestimmt a_{31} zum Pivot. Nach entsprechender Zeilenumtauschung lautet das Schema nach dem ersten Eliminationsschritt

$$\begin{array}{cccc|c} x_1 & x_2 & x_3 & 1 \\ \hline 0.076923 & -0.52991 & 0.39316 & 0.24786 \\ -0.95480 & -0.0087800 & -0.053990 & -0.062780 \\ 0.0054274 & 0.50227 & -0.50232 & -0.000053000 \end{array}$$

Die Spaltenmaximumstrategie verlangt eine zweite Zeilenswapfung. Der Eliminationsschritt ergibt

x_1	x_2	x_3	1
0.076923	-0.52991	0.39316	0.24786
0.0054274	0.50227	-0.50232	-0.000053000
-0.95480	-0.017481	-0.062771	-0.062781

Die Rücksubstitution liefert mit $x_3 \doteq 1.0002$, $x_2 \doteq 1.0002$ und $x_1 \doteq 5.0003$ recht gute Näherungswerte für die exakten Lösungen. Die Spaltenmaximumstrategie in Verbindung mit der Skalierung der gegebenen Gleichungen hat sich somit in diesem Beispiel bewährt. \triangle

Die Skalierung der Ausgangsgleichungen gemäß (2.43) überträgt sich natürlich nicht auf die Gleichungen der reduzierten Systeme (vgl. Beispiel 2.5), so dass der für den ersten Schritt günstige Einfluss der Pivotwahl nach der Spaltenmaximumstrategie in den späteren Eliminationsschritten verloren gehen kann. Somit sollten auch die reduzierten Systeme stets wieder skaliert werden. Das tut man aber nicht, weil dadurch der Rechenaufwand auf das Doppelte ansteigt. Um dennoch das Konzept beizubehalten, wird die Skalierung nicht explizit vorgenommen, sondern nur implizit als Hilfsmittel zur Bestimmung eines geeigneten Pivots verwendet, wobei die Spaltenmaximumstrategie auf die skaliert gedachten Systeme Anwendung findet. Unter den in Frage kommenden Elementen bestimmt man dasjenige zum Pivot, welches dem Betrag nach relativ zur Summe der Beträge der Elemente der zugehörigen Zeile am größten ist. Man spricht deshalb von *relativer Spaltenmaximumstrategie*. Vor Ausführung des k -ten Eliminationsschrittes ermittelt man den Index p so, dass gilt

$$\max_{k \leq i \leq n} \left\{ \frac{|a_{ik}^{(k-1)}|}{\sum_{j=k}^n |a_{ij}^{(k-1)}|} \right\} = \frac{|a_{pk}^{(k-1)}|}{\sum_{j=k}^n |a_{pj}^{(k-1)}|}. \quad (2.44)$$

Ist $p \neq k$, wird die p -te Zeile mit der k -ten Zeile vertauscht. Bei dieser Strategie sind selbstverständlich die Quotienten l_{ik} ($i > k$) betragsmäßig nicht mehr durch Eins beschränkt.

Beispiel 2.6. Das Gleichungssystem von Beispiel 2.4 wird jetzt bei fünfstelliger Rechnung nach der relativen Spaltenmaximumstrategie gelöst. Zur Verdeutlichung des Rechenablaufs sind neben dem ersten und zweiten Schema die Summen der Beträge der Matrixelemente $s_i = \sum_{j=k}^n |a_{ij}^{(k-1)}|$ und die für die Pivotwahl ausschlaggebenden Quotienten $q_i = |a_{ik}^{(k-1)}|/s_i$ aufgeführt. Es ist klar, dass im ersten Schritt wie im Beispiel 2.5 das Element a_{31} zum Pivot wird. Dies ist jetzt das absolut kleinste unter den Elementen der ersten Spalte. Im zweiten Schritt ist nochmals eine Zeilenswapfung notwendig.

x_1	x_2	x_3	1	s_i	q_i
2.1	2512	-2516	6.5	5030.1	0.00041749
-1.3	8.8	-7.6	-5.3	17.7	0.073446
0.9	-6.2	4.6	2.9	11.7	0.076923

x_1	x_2	x_3	1	s_i	q_i
0.9	-6.2	4.6	2.9	—	—
-1.4444	-0.15530	-0.95580	-1.1112	1.1111	0.13977
2.3333	2526.5	-2526.7	-0.26660	5053.2	0.49998

x_1	x_2	x_3	1
0.9	-6.2	4.6	2.9
2.3333	2526.5	-2526.7	-0.26660
-1.4444	-0.000061468	-1.1111	-1.1112

Die Unbekannten berechnen sich daraus sukzessive zu $x_3 \doteq 1.0001$, $x_2 \doteq 1.0001$, $x_1 \doteq 5.0001$. Die Determinante der Matrix \mathbf{A} ergibt sich nach (2.30) zu $|\mathbf{A}| = (-1)^2 \times 0.9 \times 2526.5 \times (-1.1111) \doteq -2526.5$. Der exakte Wert ist $|\mathbf{A}| = -2526.504$. \triangle

Nachdem das Gaußsche Eliminationsverfahren mit einer brauchbaren Pivotstrategie vervollständigt worden ist, wollen wir es in Tab. 2.1 algorithmisch so zusammenfassen, dass es leicht auf einem Rechner durchgeführt werden kann. Die Zerlegung, die Vorwärtssubstitution und die Rücksubstitution werden als in sich geschlossene Prozesse getrennt dargestellt. Die Zahlenwerte der aufeinander folgenden Schemata werden im Computer in einem festen Feld gespeichert. Dies ist deshalb möglich, weil der Wert von $a_{ij}^{(k-1)}$ von dem Moment an nicht mehr benötigt wird, wo entweder l_{ij} oder $a_{ij}^{(k)}$ berechnet ist. So werden in der algorithmischen Formulierung die Werte von l_{ij} an die Stelle von a_{ij} gesetzt, und die Koeffizienten der Endgleichungen werden stehen gelassen, genau so, wie es in den Beispielen bereits geschehen ist. Nach beendeter Zerlegung werden somit $a_{ij} = l_{ij}$ für $i > j$, und $a_{ij} = r_{ij}$ für $i \leq j$ bedeuten. Die Information über erfolgte Zeilenvertauschungen wird im Vektor $\mathbf{p} = (p_1, p_2, \dots, p_n)^T$ aufgebaut. Die k -te Komponente enthält den Index derjenigen Zeile, welche vor dem k -ten Eliminationsschritt mit der k -ten Zeile vertauscht worden ist. Es erfolgte keine Vertauschung, falls $p_k = k$ ist. In allen folgenden Beschreibungen sind die Anweisungen stets im dynamischen Sinn zu verstehen, und leere Schleifenanweisungen sollen übersprungen werden.

Bei der Vorwärtssubstitution kann der Hilfsvektor \mathbf{c} mit \mathbf{b} identifiziert werden, da b_i nicht mehr benötigt wird, sobald c_i berechnet ist. Dies ist auch deshalb angezeigt, weil der gegebene Vektor \mathbf{b} ohnehin durch die Permutationen verändert wird. Die analoge Feststellung gilt für die Rücksubstitution, wo der Lösungsvektor \mathbf{x} mit \mathbf{c} (und dann mit $\mathbf{b}!$) identifizierbar ist. Am Schluss steht dann an der Stelle von \mathbf{b} der gesuchte Lösungsvektor \mathbf{x} .

2.1.3 Ergänzungen

Mehrere rechte Seiten, Inversion

In bestimmten Anwendungen sind mehrere Gleichungssysteme mit derselben Koeffizientenmatrix \mathbf{A} , aber verschiedenen rechten Seiten \mathbf{b} entweder gleichzeitig oder nacheinander zu lösen. Die drei Lösungsschritte (2.32) des Gaußschen Algorithmus erweisen sich in dieser Situation als sehr geeignet. Denn die Zerlegung $\mathbf{PA} = \mathbf{LR}$ braucht offenbar nur einmal ausgeführt zu werden, weil dann zusammen mit der Information über die Zeilenvertauschungen

Tab. 2.1 Der vollständige Gauß-Algorithmus.

*Gauß-Elimination
mit relativer Spaltenmaximumstrategie
und Berechnung der Determinante*

$\det = 1$
 für $k = 1, 2, \dots, n - 1$:
 $\max = 0; p_k = 0$
 für $i = k, k + 1, \dots, n$:
 $s = 0$
 für $j = k, k + 1, \dots, n$:
 $s = s + |a_{ij}|$
 $q = |a_{ik}|/s$
 falls $q > \max$:
 $\max = q; p_k = i$
 falls $\max = 0$: STOP
 falls $p_k \neq k$:
 $\det = -\det$
 für $j = 1, 2, \dots, n$:
 $h = a_{kj}; a_{kj} = a_{p_k,j}; a_{p_k,j} = h$
 $\det = \det \times a_{kk}$
 für $i = k + 1, k + 2, \dots, n$:
 $a_{ik} = a_{ik}/a_{kk}$
 für $j = k + 1, k + 2, \dots, n$:
 $a_{ij} = a_{ij} - a_{ik} \times a_{kj}$
 $\det = \det \times a_{nn}$

*Vertauschungen in \mathbf{b}
und Vorwärtssubstitution*

für $k = 1, 2, \dots, n - 1$:
 falls $p_k \neq k$:
 $h = b_k; b_k = b_{p_k}; b_{p_k} = h$
 für $i = 1, 2, \dots, n$:
 $c_i = b_i$
 für $j = 1, 2, \dots, i - 1$:
 $c_i = c_i - a_{ij} \times c_j$

Rücksubstitution

für $i = n, n - 1, \dots, 1$:
 $s = c_i$
 für $k = i + 1, i + 2, \dots, n$:
 $s = s - a_{ik} \times x_k$
 $x_i = s/a_{ii}$

alle notwendigen Zahlenwerte für die Vorwärtssubstitution und Rücksubstitution vorhanden sind. Diese beiden Prozesse sind dann auf die einzelnen rechten Seiten anzuwenden.

Sind etwa gleichzeitig m Gleichungssysteme mit den rechten Seiten $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m$ zu lösen, werden sie zweckmäßigerweise zur Matrix

$$\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m) \in \mathbb{R}^{n,m} \quad (2.45)$$

zusammengefasst. Dann ist eine Matrix $\mathbf{X} \in \mathbb{R}^{n,m}$ gesucht als Lösung der Matrizengleichung

$$\mathbf{AX} = \mathbf{B} \quad (2.46)$$

Die Spalten von \mathbf{X} sind die Lösungsvektoren \mathbf{x}_μ zu den entsprechenden rechten Seiten \mathbf{b}_μ .

Nach (2.33) und (2.36) beträgt der Rechenaufwand zur Lösung von (2.46)

$$Z = \frac{1}{3}(n^3 - n) + mn^2. \quad (2.47)$$

Eine spezielle Anwendung der erwähnten Rechentechnik besteht in der *Inversion* einer regulären Matrix \mathbf{A} . Die gesuchte Inverse $\mathbf{X} = \mathbf{A}^{-1}$ erfüllt die Matrizengleichung

$$\mathbf{AX} = \mathbf{I}, \quad (2.48)$$

wo \mathbf{I} die *Einheitsmatrix* bedeutet. Die Berechnung der Inversen \mathbf{A}^{-1} ist damit auf die gleichzeitige Lösung von n Gleichungen mit derselben Matrix \mathbf{A} zurückgeführt. Der Rechenaufwand an multiplikativen Operationen beläuft sich nach (2.47) auf

$$Z_{\text{Inv}} = \frac{4}{3}n^3 - \frac{1}{3}n. \quad (2.49)$$

Dabei ist allerdings nicht berücksichtigt, dass auch nach Zeilenpermutationen in \mathbf{I} oberhalb der Einselemente Nullen stehen. Der Prozess der Vorwärtssubstitution hat deshalb im Prinzip erst beim jeweiligen Einselement zu beginnen, womit eine Reduktion der Rechenoperationen verbunden wäre. Diese Möglichkeit ist in Rechenprogrammen aber kaum vorgesehen.

Die Berechnung von \mathbf{A}^{-1} nach der beschriebenen Art erfordert neben dem Speicherplatz für \mathbf{A} auch noch denjenigen für \mathbf{I} , an deren Stelle sukzessive die Inverse aufgebaut wird. Der Speicherbedarf beträgt folglich $2n^2$ Plätze.

Nachiteration

Löst man das Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ numerisch mit dem Gauß-Algorithmus, so erhält man auf Grund der unvermeidlichen Rundungsfehler anstelle des exakten Lösungsvektors \mathbf{x} eine Näherung $\tilde{\mathbf{x}}$. Die Einsetzprobe in den gegebenen Gleichungen ergibt im Allgemeinen anstelle des Nullvektors einen *Residuenvektor*

$$\mathbf{r} := \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}. \quad (2.50)$$

Ausgehend vom bekannten Näherungsvektor $\tilde{\mathbf{x}}$ soll die exakte Lösung \mathbf{x} mit Hilfe des *Korrekturansatzes*

$$\mathbf{x} = \tilde{\mathbf{x}} + \mathbf{z} \quad (2.51)$$

ermittelt werden. Der Korrekturvektor \mathbf{z} ist so zu bestimmen, dass die Gleichungen erfüllt sind, d.h. dass gilt

$$\mathbf{Ax} - \mathbf{b} = \mathbf{A}(\tilde{\mathbf{x}} + \mathbf{z}) - \mathbf{b} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{Az} - \mathbf{b} = \mathbf{0}. \quad (2.52)$$

Beachtet man in (2.52) die Gleichung (2.50), so erkennt man, dass der Korrekturvektor \mathbf{z} das Gleichungssystem

$$\mathbf{Az} = -\mathbf{r} \quad (2.53)$$

mit derselben Matrix \mathbf{A} , aber der neuen rechten Seite $-\mathbf{r}$ erfüllen muss. Die Korrektur \mathbf{z} ergibt sich somit durch die Prozesse der Vorwärts- und Rücksubstitution aus dem Residuenvektor \mathbf{r} , der mit doppelter Genauigkeit berechnet werden muss, siehe Beispiel 2.7.

Beispiel 2.7. Die Nachiteration einer Näherungslösung soll am folgenden Gleichungssystem mit vier Unbekannten illustriert werden. Gleichzeitig soll das Beispiel die weiteren Untersuchungen aktivieren.

$$0.29412x_1 + 0.41176x_2 + 0.52941x_3 + 0.58824x_4 = 0.17642$$

$$0.42857x_1 + 0.57143x_2 + 0.71429x_3 + 0.64286x_4 = 0.21431$$

$$0.36842x_1 + 0.52632x_2 + 0.42105x_3 + 0.36842x_4 = 0.15792$$

$$0.38462x_1 + 0.53846x_2 + 0.46154x_3 + 0.38462x_4 = 0.15380$$

In einem ersten Schritt wird nur die Dreieckszerlegung der Matrix A mit der relativen Spaltenmaximumstrategie bei fünfstelliger Rechengenauigkeit ausgeführt. Neben den aufeinander folgenden Schemata sind die Summen s_i der Beträge der Matrixelemente und die Quotienten q_i angegeben.

x_1	x_2	x_3	x_4	s_i	q_i
0.29412	0.41176	0.52941	0.58824	1.8235	0.16129
0.42857	0.57143	0.71429	0.64286	2.3572	0.18181
<u>0.36842</u>	0.52632	0.42105	0.36842	1.6842	0.21875
0.38462	0.53846	0.46154	0.38462	1.7692	0.21740

x_1	x_2	x_3	x_4	s_i	q_i
0.36842	0.52632	0.42105	0.36842	—	—
1.1633	-0.040840	0.22448	0.21428	0.47960	0.085154
0.79833	-0.0084200	0.19327	0.29412	0.49581	0.016982
1.0440	<u>-0.011020</u>	0.021960	-0.00001	0.032980	0.33414

x_1	x_2	x_3	x_4	s_i	q_i
0.36842	0.52632	0.42105	0.36842	—	—
1.0440	-0.011020	0.021960	-0.00001	—	—
0.79833	0.76407	<u>0.17649</u>	0.29413	0.47062	0.37502
1.1633	3.7060	<u>0.14310</u>	0.21432	0.35742	0.40037

x_1	x_2	x_3	x_4
0.36842	0.52632	0.42105	0.36842
1.0440	-0.011020	0.021960	-0.00001
1.1633	3.7060	0.14310	0.21432
0.79833	0.76407	1.2333	<u>0.029810</u>

(2.54)

Bei drei Zeilenvertauschungen ist der Näherungswert für die Determinante $|A| \doteq (-1)^3 \times 0.36842 \times (-0.011020) \times 0.14310 \times 0.029810 \doteq 1.7319 \times 10^{-5}$. In der rechten Seite b sind die drei Zeilenvertauschungen entsprechend auszuführen. Für die Vorwärtssubstitution ist der Vektor $Pb = (0.15792, 0.15380, 0.21431, 0.17642)^T$ zu verwenden. Es resultiert der Vektor $c \doteq (0.15792, -0.011070, 0.071625, -0.029527)^T$, und die Rücksubstitution liefert die Näherungslösung $\tilde{x} \doteq (-7.9333, 4.9593, 1.9841, -0.99051)^T$.

Die Einsetzprobe mit fünfstelliger Rechnung ergibt den Residuenvektor $\tilde{r} \doteq (2, 3, -3, 7)^T \times 10^{-5}$, während zehnstellige Genauigkeit den auf fünf wesentliche Ziffern gerundeten Residuenvektor

$$\mathbf{r} \doteq (2.3951, 7.1948, -4.5999, 5.0390)^T \times 10^{-5}$$

liefert. Da die Ergebnisse recht unterschiedlich sind, indem in $\tilde{\mathbf{r}}$ meistens bereits die erste Ziffer falsch ist, ist $\tilde{\mathbf{r}}$ für eine Nachiteration nicht brauchbar. Der Residuenvektor $\mathbf{r} = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}$ muss stets mit höherer Genauigkeit berechnet werden, damit eine Nachiteration überhaupt sinnvoll sein kann [Wil 69].

Aus der Vorwärtssubstitution mit dem permutierten Vektor

$$\mathbf{P}\mathbf{r} = (-4.5999, 5.0390, 7.1948, 2.3951)^T \times 10^{-5}$$

resultiert

$$\mathbf{c}_\mathbf{r} \doteq (4.5999, -9.8413, 23.926, -28.056)^T \times 10^{-5},$$

und daraus liefert die Rücksubstitution den Korrekturvektor

$$\mathbf{z} \doteq (0.066142, 0.040360, 0.015768, -0.0094116)^T.$$

Da \mathbf{z} genau so wie $\tilde{\mathbf{x}}$ mit Fehlern behaftet ist, erhält man mit $\tilde{\mathbf{x}} + \mathbf{z} = \tilde{\tilde{\mathbf{x}}}$ nur eine weitere Näherungslösung, die unter bestimmten Voraussetzungen die Lösung \mathbf{x} besser approximiert. In unserem Beispiel ist dies tatsächlich der Fall, denn

$$\tilde{\tilde{\mathbf{x}}} \doteq (-7.9994, 4.9997, 1.9999, -0.99992)^T$$

ist eine bessere Näherung für $\mathbf{x} = (-8, 5, 2, -1)^T$. Eine weitere Nachiteration mit dem Residuenvektor $\mathbf{r} \doteq (4.7062, 6.5713, 5.0525, 5.3850)^T \times 10^{-5}$ ergibt die gesuchte Lösung mit fünfstelliger Genauigkeit.

Man beachte übrigens in diesem Zahlenbeispiel die oft typische Situation, dass die Residuenvektoren \mathbf{r} zwar betragsmäßig recht kleine Komponenten aufweisen, dass dies aber nichts über die Güte der zugehörigen Näherungslösungen $\tilde{\mathbf{x}}$ bzw. $\tilde{\tilde{\mathbf{x}}}$ auszusagen braucht. Ferner können, wie dies im ersten Schritt der Nachiteration zutrifft, betragsmäßig kleine Residuenvektoren bedeutend größere Korrekturen bewirken. \triangle

2.2 Genauigkeitsfragen, Fehlerabschätzungen

Wir wollen nun die Genauigkeit einer numerisch berechneten Näherungslösung $\tilde{\mathbf{x}}$ des Systems $\mathbf{Ax} = \mathbf{b}$ untersuchen und insbesondere nach den Gründen forschen, die für die Größe der Abweichung verantwortlich sind. Um Aussagen über den Fehler $\tilde{\mathbf{x}} - \mathbf{x}$ machen zu können, benötigen wir einerseits eine Maßzahl für die Größe eines Vektors und andererseits eine analoge Maßzahl für die Größe einer Matrix.

2.2.1 Normen

Wir betrachten nur den praktisch wichtigen Fall von reellen Vektoren $\mathbf{x} \in \mathbb{R}^n$ und von reellen Matrizen $\mathbf{A} \in \mathbb{R}^{n,n}$.

Definition 2.7. Unter der Vektornorm $\|\mathbf{x}\|$ eines Vektors $\mathbf{x} \in \mathbb{R}^n$ versteht man eine reelle Funktion seiner Komponenten, welche die drei Eigenschaften besitzt:

$$a) \quad \|\mathbf{x}\| \geq 0 \text{ für alle } \mathbf{x}, \text{ und } \|\mathbf{x}\| = 0 \text{ nur für } \mathbf{x} = \mathbf{0}; \quad (2.55)$$

$$b) \quad \|c\mathbf{x}\| = |c| \cdot \|\mathbf{x}\| \text{ für alle } c \in \mathbb{R} \text{ und alle } \mathbf{x}; \quad (2.56)$$

$$c) \quad \|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \text{ für alle } \mathbf{x}, \mathbf{y} \text{ (Dreicksungleichung)}. \quad (2.57)$$

Beispiele von Vektornormen sind

$$\|\mathbf{x}\|_\infty := \max_k |x_k|, \quad (\text{Maximumnorm}) \quad (2.58)$$

$$\|\mathbf{x}\|_2 := \left[\sum_{k=1}^n x_k^2 \right]^{\frac{1}{2}}, \quad (\text{euklidische Norm}) \quad (2.59)$$

$$\|\mathbf{x}\|_1 := \sum_{k=1}^n |x_k|, \quad (L_1\text{-Norm}). \quad (2.60)$$

Man überzeugt sich leicht davon, dass die Eigenschaften der Vektornorm erfüllt sind. Die drei Vektornormen sind in dem Sinn miteinander äquivalent, dass zwischen ihnen für alle Vektoren $\mathbf{x} \in \mathbb{R}^n$ die leicht einzusehenden Ungleichungen gelten

$$\frac{1}{\sqrt{n}} \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty,$$

$$\frac{1}{n} \|\mathbf{x}\|_1 \leq \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n \|\mathbf{x}\|_\infty,$$

$$\frac{1}{\sqrt{n}} \|\mathbf{x}\|_1 \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n} \|\mathbf{x}\|_2.$$

Definition 2.8. Unter der Matrixnorm $\|\mathbf{A}\|$ einer Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ versteht man eine reelle Funktion ihrer Elemente, welche die vier Eigenschaften aufweist:

$$a) \quad \|\mathbf{A}\| \geq 0 \text{ für alle } \mathbf{A}, \text{ und } \|\mathbf{A}\| = 0 \text{ für } \mathbf{A} = \mathbf{0}; \quad (2.61)$$

$$b) \quad \|c\mathbf{A}\| = |c| \cdot \|\mathbf{A}\| \text{ für alle } c \in \mathbb{R} \text{ und alle } \mathbf{A}; \quad (2.62)$$

$$c) \quad \|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\| \text{ für alle } \mathbf{A}, \mathbf{B} \text{ (Dreiecksungleichung);} \quad (2.63)$$

$$d) \quad \|\mathbf{A} \cdot \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|. \quad (2.64)$$

Die geforderte Eigenschaft (2.64) schränkt die Matrixnormen auf die für die Anwendungen wichtige Klasse der *submultiplikativen Normen* ein. Beispiele von gebräuchlichen Matrixnormen sind

$$\|\mathbf{A}\|_G := n \cdot \max_{i,k} |a_{ik}|, \quad (\text{Gesamtnorm}) \quad (2.65)$$

$$\|\mathbf{A}\|_z := \max_i \sum_{k=1}^n |a_{ik}|, \quad (\text{Zeilensummennorm}) \quad (2.66)$$

$$\|\mathbf{A}\|_s := \max_k \sum_{i=1}^n |a_{ik}|, \quad (\text{Spaltensummennorm}) \quad (2.67)$$

$$\|\mathbf{A}\|_F := \left[\sum_{i,k=1}^n a_{ik}^2 \right]^{\frac{1}{2}}, \quad (\text{Frobenius-Norm}). \quad (2.68)$$

Dass die angegebenen Matrixnormen die ersten drei Eigenschaften (2.61), (2.62) und (2.63) erfüllen, ist offensichtlich. Die vierte Eigenschaft (2.64) wollen wir nur für die Gesamtnorm

nachweisen. Für die anderen Matrixnormen verläuft die Verifikation analog.

$$\begin{aligned}\|\mathbf{A} \cdot \mathbf{B}\|_G &= n \cdot \max_{i,k} \left| \sum_{j=1}^n a_{ij} b_{jk} \right| \leq n \cdot \max_{i,k} \sum_{j=1}^n |a_{ij}| \cdot |b_{jk}| \\ &\leq n \cdot \max_{i,k} \sum_{j=1}^n \{\max_{l,m} |a_{lm}|\} \cdot \{\max_{r,s} |b_{rs}|\} \\ &= n^2 \cdot \{\max_{l,m} |a_{lm}|\} \cdot \{\max_{r,s} |b_{rs}|\} = \|\mathbf{A}\|_G \cdot \|\mathbf{B}\|_G.\end{aligned}$$

Die vier Matrixnormen sind ebenfalls miteinander äquivalent. Denn es gelten beispielsweise für alle Matrizen $\mathbf{A} \in \mathbb{R}^{n,n}$ die Ungleichungen

$$\begin{aligned}\frac{1}{n} \|\mathbf{A}\|_G &\leq \|\mathbf{A}\|_{z,s} \leq \|\mathbf{A}\|_G \leq n \|\mathbf{A}\|_{z,s}, \\ \frac{1}{n} \|\mathbf{A}\|_G &\leq \|\mathbf{A}\|_F \leq \|\mathbf{A}\|_G \leq n \|\mathbf{A}\|_F.\end{aligned}$$

Da in den nachfolgenden Betrachtungen Matrizen und Vektoren gemeinsam auftreten, müssen die verwendeten Matrixnormen und Vektornormen in einem zu präzisierenden Zusammenhang stehen, damit man geeignet damit operieren kann.

Definition 2.9. Eine Matrixnorm $\|\mathbf{A}\|$ heißt *kompatibel* oder *verträglich* mit der Vektornorm $\|\mathbf{x}\|$, falls die Ungleichung gilt

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \text{ für alle } \mathbf{x} \in \mathbb{R}^n \text{ und alle } \mathbf{A} \in \mathbb{R}^{n,n}. \quad (2.69)$$

Kombinationen von verträglichen Normen sind etwa

$$\|\mathbf{A}\|_G \text{ oder } \|\mathbf{A}\|_z \text{ sind kompatibel mit } \|\mathbf{x}\|_\infty; \quad (2.70)$$

$$\|\mathbf{A}\|_G \text{ oder } \|\mathbf{A}\|_s \text{ sind kompatibel mit } \|\mathbf{x}\|_1; \quad (2.71)$$

$$\|\mathbf{A}\|_G \text{ oder } \|\mathbf{A}\|_F \text{ sind kompatibel mit } \|\mathbf{x}\|_2. \quad (2.72)$$

Die Verträglichkeit von Normenpaaren soll in zwei Fällen verifiziert werden. So ist wegen

$$\begin{aligned}\|\mathbf{Ax}\|_\infty &= \max_i \left\{ \left| \sum_{k=1}^n a_{ik} x_k \right| \right\} \leq \max_i \left\{ \sum_{k=1}^n |a_{ik}| \cdot |x_k| \right\} \\ &\leq \max_i \left\{ \sum_{k=1}^n [\max_{r,s} |a_{rs}|] \cdot [\max_l |x_l|] \right\} = \|\mathbf{A}\|_G \cdot \|\mathbf{x}\|_\infty\end{aligned}$$

die Gesamtnorm mit der Maximumnorm kompatibel. Desgleichen ist die Frobenius-Norm mit der euklidischen Vektornorm verträglich. Unter Anwendung der Schwarzschen Ungleichung gilt

$$\begin{aligned}\|\mathbf{Ax}\|_2 &= \left[\sum_{i=1}^n \left(\sum_{k=1}^n a_{ik} x_k \right)^2 \right]^{\frac{1}{2}} \leq \left[\sum_{i=1}^n \left\{ \left(\sum_{k=1}^n a_{ik}^2 \right) \left(\sum_{k=1}^n x_k^2 \right) \right\} \right]^{\frac{1}{2}} \\ &= \left[\sum_{i=1}^n \sum_{k=1}^n a_{ik}^2 \right]^{\frac{1}{2}} \left[\sum_{k=1}^n x_k^2 \right]^{\frac{1}{2}} = \|\mathbf{A}\|_F \cdot \|\mathbf{x}\|_2.\end{aligned}$$

Im Allgemeinen wird die rechte Seite der Ungleichung (2.69) echt größer sein als die linke Seite. Deshalb ist es sinnvoll eine Matrixnorm zu definieren, für die in (2.69) mindestens für einen Vektor $\mathbf{x} \neq \mathbf{0}$ Gleichheit gilt. Das gelingt mit der folgenden Definition einer so genannten *zugeordneten* Norm.

Definition 2.10. Der zu einer gegebenen Vektornorm definierte Zahlenwert

$$\|\mathbf{A}\| := \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\| \quad (2.73)$$

heißt die *zugeordnete* oder *natürliche* Matrixnorm. Sie wird auch als *Grenzennorm* oder *lub-Norm* (lowest upper bound) bezeichnet.

Satz 2.11. Der gemäß (2.73) erklärte Zahlenwert stellt eine Matrixnorm dar. Sie ist mit der zu Grunde liegenden Vektornorm kompatibel. Sie ist unter allen mit der Vektornorm $\|\mathbf{x}\|$ verträglichen Matrixnormen die kleinste.

Beweis. Wir verifizieren die Eigenschaften einer Matrixnorm.

a) Mit $\mathbf{x} \neq \mathbf{0}$ gelten $\|\mathbf{Ax}\| \geq 0$ für alle $\mathbf{A} \in \mathbb{R}^{n,n}$ und $\|\mathbf{x}\| > 0$. Folglich ist $\max_{\mathbf{x} \neq \mathbf{0}} \|\mathbf{Ax}\|/\|\mathbf{x}\| \geq 0$.

Weiter ist zu zeigen, dass aus $\|\mathbf{A}\| = 0 \quad \mathbf{A} = \mathbf{0}$ folgt. Wir nehmen das Gegenteil an, d.h. es sei $\mathbf{A} \neq \mathbf{0}$. Dann existiert mindestens ein $a_{pq} \neq 0$. Für \mathbf{x} wählen wir den q -ten Einheitsvektor $\mathbf{e}_q \neq \mathbf{0}$, für den $\mathbf{Ae}_q \neq \mathbf{0}$ ist. Für diesen Vektor ist $\|\mathbf{Ae}_q\|/\|\mathbf{e}_q\| > 0$. Damit ist das Maximum in (2.73) erst recht größer als null, womit ein Widerspruch vorliegt.

b) Auf Grund der zweiten Eigenschaft der Vektornorm gilt

$$\|c\mathbf{A}\| := \max_{\|\mathbf{x}\|=1} \|c\mathbf{Ax}\| = \max_{\|\mathbf{x}\|=1} \{ |c| \cdot \|\mathbf{Ax}\| \} = |c| \cdot \|\mathbf{A}\|.$$

c) Unter Benutzung der Dreiecksungleichung für Vektornormen folgt

$$\begin{aligned} \|\mathbf{A} + \mathbf{B}\| &:= \max_{\|\mathbf{x}\|=1} \|(\mathbf{A} + \mathbf{B})\mathbf{x}\| \leq \max_{\|\mathbf{x}\|=1} \{ \|\mathbf{Ax}\| + \|\mathbf{Bx}\| \} \\ &\leq \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\| + \max_{\|\mathbf{x}\|=1} \|\mathbf{Bx}\| = \|\mathbf{A}\| + \|\mathbf{B}\|. \end{aligned}$$

d) Um die Submultiplikativität der Norm nachzuweisen, setzen wir $\mathbf{A} \neq \mathbf{0}$ und $\mathbf{B} \neq \mathbf{0}$ voraus. Andernfalls ist die Ungleichung (2.64) trivialerweise erfüllt. Dann gelten

$$\begin{aligned} \|\mathbf{A} \cdot \mathbf{B}\| &:= \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{ABx}\|}{\|\mathbf{x}\|} = \max_{\substack{\mathbf{x} \neq \mathbf{0} \\ \mathbf{Bx} \neq \mathbf{0}}} \frac{\|\mathbf{A}(\mathbf{Bx})\| \|\mathbf{Bx}\|}{\|\mathbf{Bx}\| \|\mathbf{x}\|} \\ &\leq \max_{\mathbf{Bx} \neq \mathbf{0}} \frac{\|\mathbf{A}(\mathbf{Bx})\|}{\|\mathbf{Bx}\|} \cdot \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Bx}\|}{\|\mathbf{x}\|} \\ &\leq \max_{\mathbf{y} \neq \mathbf{0}} \frac{\|\mathbf{Ay}\|}{\|\mathbf{y}\|} \cdot \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Bx}\|}{\|\mathbf{x}\|} = \|\mathbf{A}\| \cdot \|\mathbf{B}\|. \end{aligned}$$

Die Kompatibilität der so erklärten Matrixnorm mit der gegebenen Vektornorm ist eine unmittelbare Folge der Definition (2.73), und die letzte Aussage von Satz 2.11 ist offensichtlich, da ein Vektor $\mathbf{x} \neq \mathbf{0}$ so existiert, dass $\|\mathbf{Ax}\| = \|\mathbf{A}\| \cdot \|\mathbf{x}\|$ gilt. \square

Gemäß Definition 2.10 ist die der Maximumn norm $\|\mathbf{x}\|_\infty$ zugeordnete Matrixnorm $\|\mathbf{A}\|_\infty$ gegeben durch

$$\begin{aligned}\|\mathbf{A}\|_\infty &:= \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|_\infty = \max_{\|\mathbf{x}\|=1} \left\{ \max_i \left| \sum_{k=1}^n a_{ik} x_k \right| \right\} \\ &= \max_i \left\{ \max_{\|\mathbf{x}\|=1} \left| \sum_{k=1}^n a_{ik} x_k \right| \right\} = \max_i \sum_{k=1}^n |a_{ik}| = \|\mathbf{A}\|_z.\end{aligned}$$

Der Betrag der Summe wird für festes i dann am größten, falls $x_k = \text{sign}(a_{ik})$ ist. Auf Grund von Satz 2.11 ist deshalb die Zeilensummennorm die kleinste, mit der Maximumn norm verträgliche Matrixnorm.

Um die zur euklidischen Vektornorm $\|\mathbf{x}\|_2$ zugehörige natürliche Matrixnorm $\|\mathbf{A}\|_2$ herzuleiten, sind einige fundamentale Kenntnisse aus der Theorie der linearen Algebra erforderlich.

$$\|\mathbf{A}\|_2 := \max_{\|\mathbf{x}\|_2=1} \|\mathbf{Ax}\|_2 = \max_{\|\mathbf{x}\|_2=1} \{(\mathbf{Ax})^T (\mathbf{Ax})\}^{\frac{1}{2}} = \max_{\|\mathbf{x}\|_2=1} \{\mathbf{x}^T \mathbf{A}^T \mathbf{Ax}\}^{\frac{1}{2}}$$

Die im letzten Ausdruck auftretende Matrix $\mathbf{A}^T \mathbf{A}$ ist offenbar symmetrisch und positiv semidefinit, weil für die zugehörige quadratische Form $Q(\mathbf{x}) := \mathbf{x}^T (\mathbf{A}^T \mathbf{A}) \mathbf{x} \geq 0$ für alle $\mathbf{x} \neq \mathbf{0}$ gilt. Folglich sind die Eigenwerte μ_i von $\mathbf{A}^T \mathbf{A}$ reell und nicht negativ, und die n Eigenvektoren $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ bilden eine vollständige, orthonormierte Basis im \mathbb{R}^n .

$$\mathbf{A}^T \mathbf{Ax}_i = \mu_i \mathbf{x}_i, \quad \mu_i \in \mathbb{R}, \quad \mu_i \geq 0; \quad \mathbf{x}_i^T \mathbf{x}_j = \delta_{ij} \tag{2.74}$$

Mit der eindeutigen Darstellung eines beliebigen Vektors $\mathbf{x} \in \mathbb{R}^n$ als Linearkombination der Eigenvektoren \mathbf{x}_i

$$\mathbf{x} = \sum_{i=1}^n c_i \mathbf{x}_i \tag{2.75}$$

ergibt sich einmal unter Berücksichtigung von (2.74)

$$\begin{aligned}\mathbf{x}^T \mathbf{A}^T \mathbf{Ax} &= \left(\sum_{i=1}^n c_i \mathbf{x}_i \right)^T \mathbf{A}^T \mathbf{A} \left(\sum_{j=1}^n c_j \mathbf{x}_j \right) \\ &= \left(\sum_{i=1}^n c_i \mathbf{x}_i \right)^T \left(\sum_{j=1}^n c_j \mu_j \mathbf{x}_j \right) = \sum_{i=1}^n c_i^2 \mu_i.\end{aligned}$$

Die Eigenwerte μ_i seien der Größe nach nummeriert, so dass $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n \geq 0$ gilt.

Aus der Bedingung $\|\mathbf{x}\|_2 = 1$ folgt noch $\sum_{i=1}^n c_i^2 = 1$, und somit für die zugeordnete Matrixnorm

$$\|\mathbf{A}\|_2 = \max_{\|\mathbf{x}\|_2=1} \left\{ \sum_{i=1}^n c_i^2 \mu_i \right\}^{\frac{1}{2}} \leq \max_{\|\mathbf{x}\|_2=1} \left\{ \mu_1 \sum_{i=1}^n c_i^2 \right\}^{\frac{1}{2}} = \sqrt{\mu_1}.$$

Der maximal mögliche Wert $\sqrt{\mu_1}$ wird für $\mathbf{x} = \mathbf{x}_1$ mit $c_1 = 1, c_2 = \dots = c_n = 0$ angenommen. Damit haben wir das Ergebnis

$$\|\mathbf{A}\|_2 := \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 = \sqrt{\mu_1}, \quad (2.76)$$

wobei μ_1 der größte Eigenwert von $\mathbf{A}^T \mathbf{A}$ ist. Man bezeichnet die der euklidischen Vektornorm zugeordnete Matrixnorm $\|\mathbf{A}\|_2$ auch als *Spektralnorm*. Nach Satz 2.11 ist sie die kleinste, mit der euklidischen Vektornorm verträgliche Matrixnorm.

Die Bezeichnung als Spektralnorm wird verständlich im Spezialfall einer *symmetrischen Matrix* \mathbf{A} . Bedeuten $\lambda_1, \lambda_2, \dots, \lambda_n$ die reellen Eigenwerte von \mathbf{A} , dann besitzt die Matrix $\mathbf{A}^T \mathbf{A} = \mathbf{A}\mathbf{A} = \mathbf{A}^2$ bekanntlich die Eigenwerte $\mu_i = \lambda_i^2 \geq 0$, so dass aus (2.76) folgt

$$\|\mathbf{A}\|_2 = |\lambda_1|, \quad |\lambda_1| = \max_i |\lambda_i|. \quad (2.77)$$

Die Spektralnorm einer symmetrischen Matrix \mathbf{A} ist durch ihren betragsgrößten Eigenwert λ_1 gegeben.

Als Vorbereitung für die nachfolgende Anwendung soll die Spektralnorm der Inversen \mathbf{A}^{-1} einer regulären Matrix \mathbf{A} angegeben werden. Nach (2.76) ist $\|\mathbf{A}^{-1}\|_2 = \sqrt{\psi_1}$, wo ψ_1 gleich dem größten Eigenwert von $\mathbf{A}^{-1T} \mathbf{A}^{-1} = (\mathbf{A}\mathbf{A}^T)^{-1}$ ist. Da aber die inverse Matrix \mathbf{C}^{-1} bekanntlich die reziproken Eigenwerte von \mathbf{C} besitzt, ist ψ_1 gleich dem reziproken Wert des kleinsten (positiven) Eigenwertes der positiv definiten Matrix $\mathbf{A}\mathbf{A}^T$. Die letzte Matrix ist aber ähnlich zur Matrix $\mathbf{A}^T \mathbf{A}$, denn es gilt $\mathbf{A}^{-1}(\mathbf{A}\mathbf{A}^T)\mathbf{A} = \mathbf{A}^T \mathbf{A}$, so dass $\mathbf{A}\mathbf{A}^T$ und $\mathbf{A}^T \mathbf{A}$ die gleichen Eigenwerte haben. Deshalb gilt

$$\|\mathbf{A}^{-1}\|_2 = 1/\sqrt{\mu_n}, \quad (2.78)$$

wo μ_n der kleinste Eigenwert der positiv definiten Matrix $\mathbf{A}^T \mathbf{A}$ ist. Für eine symmetrische, reguläre Matrix ist weiter

$$\|\mathbf{A}^{-1}\|_2 = 1/|\lambda_n|, \quad \mathbf{A}^T = \mathbf{A}, \quad |\lambda_n| = \min_i |\lambda_i|. \quad (2.79)$$

2.2.2 Fehlerabschätzungen, Kondition

Wir wollen nun zwei Fragestellungen untersuchen, welche die Genauigkeit einer berechneten Näherung $\tilde{\mathbf{x}}$ der Lösung \mathbf{x} des quadratischen, regulären Systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ betreffen.

Zuerst wollen wir das Problem betrachten, welche Rückschlüsse aus der Größe des Residuenvektors $\mathbf{r} = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}$ auf den Fehler $\mathbf{z} := \mathbf{x} - \tilde{\mathbf{x}}$ gezogen werden können. Dazu sei $\|\mathbf{A}\|$ eine beliebige Matrixnorm und $\|\mathbf{x}\|$ eine dazu verträgliche Vektornorm. Da nach (2.53) der Fehlervektor \mathbf{z} das Gleichungssystem $\mathbf{A}\mathbf{z} = -\mathbf{r}$ erfüllt, folgt aus den Beziehungen

$$\|\mathbf{b}\| = \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \|\mathbf{x}\|, \quad \|\mathbf{z}\| = \|-\mathbf{A}^{-1}\mathbf{r}\| \leq \|\mathbf{A}^{-1}\| \|\mathbf{r}\| \quad (2.80)$$

die Abschätzung für den relativen Fehler

$$\frac{\|z\|}{\|x\|} = \frac{\|\tilde{x} - x\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|r\|}{\|b\|} =: \kappa(A) \frac{\|r\|}{\|b\|}. \quad (2.81)$$

Definition 2.12. Die Größe

$$\kappa(A) := \|A\| \|A^{-1}\| \quad (2.82)$$

heißt *Konditionszahl* für die Lösung des linearen Gleichungssystems $Ax = b$ mit der Matrix A als Koeffizientenmatrix. $\kappa(A)$ ist abhängig von der verwendeten Matrixnorm.

Die Konditionszahl $\kappa(A)$ ist mindestens gleich Eins, denn es gilt stets

$$1 \leq \|I\| = \|AA^{-1}\| \leq \|A\| \|A^{-1}\| = \kappa(A).$$

Die Abschätzung (2.81) bedeutet konkret, dass neben einem kleinen Residuenvektor r , bezogen auf die Größe der rechten Seite b die Konditionszahl ausschlaggebend für den relativen Fehler der Näherung \tilde{x} ist. Nur bei kleiner Konditionszahl kann aus einem relativ kleinen Residuenvektor auf einen kleinen relativen Fehler geschlossen werden!

Beispiel 2.8. Wir betrachten das System von linearen Gleichungen von Beispiel 2.7, und wollen die Fehlerabschätzung (2.81) anwenden. Als Normen sollen der Einfachheit halber die Maximum-norm $\|x\|_\infty$ und die ihr zugeordnete Zeilensummennorm $\|A\|_\infty = \|A\|_z$ verwendet werden. Zur Bestimmung der Konditionszahl benötigen wir die Inverse A^{-1} .

$$A^{-1} \doteq \begin{pmatrix} 168.40 & -235.80 & -771.75 & 875.82 \\ -101.04 & 138.68 & 470.63 & -528.07 \\ -50.588 & 69.434 & 188.13 & -218.89 \\ 33.752 & -41.659 & -112.88 & 128.73 \end{pmatrix}$$

Somit sind $\|A\|_\infty \doteq 2.3572$, $\|A^{-1}\|_\infty \doteq 2051.77$ und $\kappa_\infty(A) \doteq 4836.4$. Mit $\|x\|_\infty = 8$, $\|r\|_\infty = 7.1948 \cdot 10^{-5}$ und $\|b\|_\infty = 0.21431$ schätzt (2.81) den absoluten Fehler ab zu $\|\tilde{x} - x\|_\infty \leq 12.99$. Tatsächlich ist $\|\tilde{x} - x\|_\infty = 0.0667$, also wesentlich kleiner. \triangle

Das Rechnen mit endlicher Genauigkeit hat zur Folge, dass in der Regel bereits die Koeffizienten a_{ik} und b_i des zu lösenden Gleichungssystems im Rechner nicht exakt darstellbar sind. Sie sind zudem oft mit Rundungsfehlern behaftet, falls sie ihrerseits das Ergebnis einer Rechnung sind. Deshalb soll nun der mögliche Einfluss von Fehlern in den Ausgangsdaten auf die Lösung x untersucht werden, d.h. die *Empfindlichkeit* der Lösung x auf *Störungen* in den Koeffizienten. Unsere Fragestellung lautet deshalb: Wie groß kann die Änderung δx der Lösung x von $Ax = b$ sein, falls die Matrix A um δA und die rechte Seite b um δb geändert werden? Dabei sollen δA und δb kleine Störungen bedeuten derart, dass auch die Matrix $A + \delta A$ regulär ist. Der Vektor $x + \delta x$ soll also Lösung von

$$(A + \delta A)(x + \delta x) = (b + \delta b) \quad (2.83)$$

sein. Nach Ausmultiplikation ergibt sich

$$Ax + A\delta x + \delta Ax + \delta A\delta x = b + \delta b,$$

und wegen $Ax = b$ erhalten wir weiter

$$\begin{aligned}\mathbf{A} \delta \mathbf{x} &= \delta \mathbf{b} - \delta \mathbf{A} \mathbf{x} - \delta \mathbf{A} \delta \mathbf{x}, \\ \delta \mathbf{x} &= \mathbf{A}^{-1} \{\delta \mathbf{b} - \delta \mathbf{A} \mathbf{x} - \delta \mathbf{A} \delta \mathbf{x}\}.\end{aligned}$$

Für verträgliche Normen folgt daraus

$$\begin{aligned}\|\delta \mathbf{x}\| &\leq \|\mathbf{A}^{-1}\| \|\delta \mathbf{b} - \delta \mathbf{A} \mathbf{x} - \delta \mathbf{A} \delta \mathbf{x}\| \\ &\leq \|\mathbf{A}^{-1}\| \{\|\delta \mathbf{b}\| + \|\delta \mathbf{A}\| \|\mathbf{x}\| + \|\delta \mathbf{A}\| \|\delta \mathbf{x}\|\}\end{aligned}$$

und weiter

$$(1 - \|\mathbf{A}^{-1}\| \|\delta \mathbf{A}\|) \|\delta \mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \{\|\delta \mathbf{b}\| + \|\delta \mathbf{A}\| \|\mathbf{x}\|\}. \quad (2.84)$$

An dieser Stelle treffen wir die Zusatzannahme, dass die Störung $\delta \mathbf{A}$ so klein sei, dass $\|\mathbf{A}^{-1}\| \|\delta \mathbf{A}\| < 1$ gilt. Dann folgt aus (2.84) die Abschätzung für die Norm der Änderung $\delta \mathbf{x}$:

$$\|\delta \mathbf{x}\| \leq \frac{\|\mathbf{A}^{-1}\|}{1 - \|\mathbf{A}^{-1}\| \|\delta \mathbf{A}\|} \{\|\delta \mathbf{b}\| + \|\delta \mathbf{A}\| \|\mathbf{x}\|\}. \quad (2.85)$$

Anstelle der absoluten Fehlerabschätzung sind wir mehr an einer relativen Abschätzung interessiert. Aus $\mathbf{A} \mathbf{x} = \mathbf{b}$ folgt aber

$$\|\mathbf{b}\| = \|\mathbf{A} \mathbf{x}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \quad \text{oder} \quad \|\mathbf{x}\| \geq \|\mathbf{b}\| / \|\mathbf{A}\|,$$

und somit erhalten wir aus (2.85)

$$\begin{aligned}\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} &\leq \frac{\|\mathbf{A}^{-1}\|}{1 - \|\mathbf{A}^{-1}\| \|\delta \mathbf{A}\|} \left\{ \frac{\|\delta \mathbf{b}\|}{\|\mathbf{x}\|} + \|\delta \mathbf{A}\| \right\} \\ &\leq \frac{\|\mathbf{A}^{-1}\| \|\mathbf{A}\|}{1 - \|\mathbf{A}^{-1}\| \|\delta \mathbf{A}\|} \left\{ \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} \right\}.\end{aligned}$$

Mit $\|\mathbf{A}^{-1}\| \|\delta \mathbf{A}\| = \kappa(\mathbf{A}) \|\delta \mathbf{A}\| / \|\mathbf{A}\| < 1$ lautet das Ergebnis

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\kappa(\mathbf{A})}{1 - \kappa(\mathbf{A}) \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|}} \left\{ \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} \right\}$$

(2.86)

Die Konditionszahl $\kappa(\mathbf{A})$ der Koeffizientenmatrix \mathbf{A} ist die entscheidende Größe, welche die Empfindlichkeit der Lösung \mathbf{x} gegenüber Änderungen $\delta \mathbf{A}$ und $\delta \mathbf{b}$ beschreibt. Wir wollen nun die praktische Bedeutung und die numerischen Konsequenzen dieser Abschätzungen darlegen. Bei einer d -stelligen dezimalen Gleitpunktrechnung können die relativen Fehler der Ausgangsdaten für beliebige, kompatible Normen von der Größenordnung

$$\|\delta \mathbf{A}\| / \|\mathbf{A}\| \approx 5 \cdot 10^{-d}, \quad \|\delta \mathbf{b}\| / \|\mathbf{b}\| \approx 5 \cdot 10^{-d}$$

sein. Ist die Konditionszahl $\kappa(\mathbf{A}) \approx 10^\alpha$ mit $5 \cdot 10^{\alpha-d} \ll 1$, so ergibt (2.86) die qualitative Abschätzung

$$\|\delta \mathbf{x}\| / \|\mathbf{x}\| \leq 10^{\alpha-d+1}.$$

Mit dieser Schätzung der Empfindlichkeit gelangen wir zu folgender

Daumenregel. Wird ein lineares Gleichungssystem $\mathbf{A} \mathbf{x} = \mathbf{b}$ mit d -stelliger dezimaler Gleitpunktrechnung gelöst, und beträgt die Konditionszahl $\kappa(\mathbf{A}) \approx 10^\alpha$, so sind auf Grund der

im Allgemeinen unvermeidlichen Eingangsfehler in der berechneten Lösung \tilde{x} , bezogen auf die betragsgrößte Komponente, nur $d - \alpha - 1$ Dezimalstellen sicher.

Auch wenn diese Regel oft eine pessimistische Aussage liefert, so ist ein weiterer wesentlicher Punkt zu beachten. Da die Abschätzung (2.86) die Normen betrifft, kann die Änderung δx alle Komponenten von x betreffen. Falls die Werte der Unbekannten starke Größenunterschiede aufweisen, können die betragskleinsten bedeutend größere relative Fehler enthalten, die so groß sein können, dass nicht einmal das Vorzeichen richtig ist.

Beispiel 2.9. Wir betrachten ein lineares Gleichungssystem $Ax = b$ in zwei Unbekannten mit

$$A = \begin{pmatrix} 0.99 & 0.98 \\ 0.98 & 0.97 \end{pmatrix}, \quad b = \begin{pmatrix} 1.97 \\ 1.95 \end{pmatrix}, \quad x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Die Konditionszahl der symmetrischen Matrix A bezüglich der Spektralnorm berechnet sich nach (2.77) und (2.79) als Quotient des betragsgrößten und betragskleinsten Eigenwertes von A zu $\kappa(A) = |\lambda_1|/|\lambda_2| \doteq 1.96005/0.000051019 \doteq 38418 \doteq 3.8 \cdot 10^4$. Sind die angegebenen Zahlenwerte bei fünfstelliger Rechnung mit entsprechenden Rundungsfehlern behaftet, so sind nach der Dauermenregel mit $d = 5$, $\alpha = 4$ gar keine richtigen Dezimalstellen zu erwarten. Dies ist auch tatsächlich der Fall, denn das gestörte Gleichungssystem $(A + \delta A)(x + \delta x) = (b + \delta b)$ mit

$$A + \delta A = \begin{pmatrix} 0.990005 & 0.979996 \\ 0.979996 & 0.970004 \end{pmatrix}, \quad b + \delta b = \begin{pmatrix} 1.969967 \\ 1.950035 \end{pmatrix}$$

besitzt die Lösung

$$x + \delta x \doteq \begin{pmatrix} 1.8072 \\ 0.18452 \end{pmatrix}, \quad \text{also} \quad \delta x \doteq \begin{pmatrix} 0.8072 \\ -0.81548 \end{pmatrix}.$$

Die Abschätzung (2.86) ist in diesem konstruierten Beispiel sehr realistisch. Denn mit $\|\delta A\|_2 \doteq 8.531 \cdot 10^{-6}$, $\|A\|_2 \doteq 1.960$, $\|\delta b\|_2 \doteq 4.810 \cdot 10^{-5}$, $\|b\|_2 \doteq 2.772$ liefert sie

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \frac{3.842 \cdot 10^4}{1 - 0.1672} \{4.353 \cdot 10^{-6} + 1.735 \cdot 10^{-5}\} = 1.001,$$

während tatsächlich $\|\delta x\|_2/\|x\|_2 \doteq 0.8114$ ist. \triangle

Die Abschätzung (2.86) für den relativen Fehler besitzt auch dann eine Anwendung, wenn die Ausgangsdaten als exakt anzusehen sind. Die numerisch berechneten Koeffizienten des ersten reduzierten Systems können als die exakten Werte eines gestörten Ausgangssystems betrachtet werden. Diese Betrachtungsweise lässt sich auf die weiteren Eliminationsschritte fortsetzen. So kann die resultierende Dreieckszerlegung als die exakte Zerlegung einer geänderten Ausgangsmatrix aufgefasst werden, so dass gilt $P(A + \delta A) = \tilde{L}\tilde{R}$, wo \tilde{L} und \tilde{R} die mit Gleitpunktarithmetik erhaltenen Dreiecksmatrizen bedeuten. Die Analyse der Rundungsfehler gestattet, die Beträge der Elemente von δA abzuschätzen. Die Idee dieser *Rückwärts-Fehleranalyse* lässt sich auch auf die Prozesse der Vorwärts- und Rücksubstitution ausdehnen, und liefert Abschätzungen für δb . Leider sind die theoretischen Ergebnisse im allgemeinen Fall allzu pessimistisch und entsprechen nicht den praktischen Erfahrungen [Sto 07, Stu 82, Wil 88, Wil 69]. Die tatsächlichen Änderungen δA und δb einer Rückwärtsfehlerrechnung sind im allgemeinen Fall bei kleinen Systemen etwa von der Größenordnung der unvermeidbaren Eingangsfehler und sind bei größeren Systemen nur um Faktoren von

wenigen Zehnerpotenzen größer. Da die Abschätzung (2.86) auf Grund ihrer allgemeinen Gültigkeit oft zu pessimistisch ist, so bleibt die oben formulierte Daumenregel zumindest als Richtlinie auch unter Berücksichtigung der Rückwärts-Fehleranalyse anwendbar.

Zur *Nachiteration* kann auf Grund der obigen Betrachtungen noch eine heuristisch gültige Aussage hinzugefügt werden. Damit jeder Nachiterationsschritt wenigstens eine Verbesserung der Näherungslösung bringt, muss mindestens eine Ziffer, immer bezogen auf die absolut größte Komponente des Korrekturvektors, richtig sein. Damit dies bei d -stelliger dezimaler Gleitpunktrechnung zutrifft, muss $\alpha < d - 1$ sein. Dann wird die Nachiteration in jedem Schritt weitere $(d - \alpha - 1)$ Dezimalstellen richtigstellen, und die Folge von Näherungslösungen konvergiert tatsächlich. Beispiel 2.7 illustriert diese Tatsache sehr schön. Bei einer Konditionszahl $\kappa(\mathbf{A}) \doteq 2.82 \cdot 10^3$ bezüglich der Spektralnorm sind in $\tilde{\mathbf{x}}$ sogar zwei wesentliche Dezimalstellen richtig, und die Nachiteration liefert zwei weitere richtige Stellen.

Soll etwa ein Computerprogramm zu einer berechneten Näherungslösung $\tilde{\mathbf{x}}$ eine präzise Angabe über die Genauigkeit mitliefern, oder soll es entscheiden können, ob eine Nachiteration notwendig oder überhaupt sinnvoll ist, ist die Kenntnis der Konditionszahl $\kappa(\mathbf{A})$ erforderlich. Dazu braucht man aber entweder die Inverse von \mathbf{A} oder den größten und kleinsten Eigenwert von $\mathbf{A}^T \mathbf{A}$. Um diese im Vergleich zur Gleichungsauflösung recht aufwändigen Prozesse zu vermeiden, sind Verfahren entwickelt worden, die mit vertretbarem Rechenaufwand einen brauchbaren Schätzwert für $\kappa(\mathbf{A})$ liefern [Cli 79, For 77, Hig 02, Kie 88].

2.3 Systeme mit speziellen Eigenschaften

In vielen Anwendungen sind lineare Gleichungssysteme mit besonderen Strukturen zu lösen. Deren Berücksichtigung kann Rechen- und Speicheraufwand reduzieren und die Stabilität erhöhen. Wir betrachten einige wichtige Fälle solcher Gleichungssysteme, die in verschiedenen Zusammenhängen als Teilaufgabe zu lösen sind, so z.B. in den Kapiteln 10 und 11. Wir stellen die einschlägigen Algorithmen und die zweckmäßigen Maßnahmen für eine geeignete Realisierung auf einem Rechner zusammen.

2.3.1 Symmetrische, positiv definite Systeme

Oft ist die Koeffizientenmatrix \mathbf{A} in $\mathbf{Ax} = \mathbf{b}$ nicht nur symmetrisch, sondern auch positiv definit. Diese Eigenschaft ist z.B. typisch für Anwendungen, denen ein Energieerhaltungssatz zu Grunde liegt.

Definition 2.13. Eine symmetrische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ heisst positiv definit, falls die zugehörige quadratische Form positiv definit ist; d.h. falls gilt

$$\begin{aligned} Q(\mathbf{x}) := \mathbf{x}^T \mathbf{Ax} &= \sum_{i=1}^n \sum_{k=1}^n a_{ik} x_i x_k \geq 0 \quad \text{für alle } \mathbf{x} \in \mathbb{R}^n, \\ &= 0 \quad \text{nur für } \mathbf{x} = \mathbf{0}. \end{aligned} \tag{2.87}$$

Satz 2.14. Ist eine symmetrische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ positiv definit, so erfüllen ihre Elemente notwendigerweise die Bedingungen

$$a) \quad a_{ii} > 0 \text{ für } i = 1, 2, \dots, n; \quad (2.88)$$

$$b) \quad a_{ik}^2 < a_{ii}a_{kk} \quad \text{für } i \neq k; \quad i, k = 1, 2, \dots, n; \quad (2.89)$$

$$c) \quad \text{es existiert ein } k \text{ mit } \max_{i,j} |a_{ij}| = a_{kk}. \quad (2.90)$$

Beweis. Die beiden ersten Eigenschaften zeigen wir auf Grund von (2.87) durch spezielle Wahl von $\mathbf{x} \neq \mathbf{0}$. So folgt (2.88) mit $\mathbf{x} = \mathbf{e}_i$ (i -ter Einheitsvektor) wegen $Q(\mathbf{x}) = a_{ii} > 0$. Wählen wir $\mathbf{x} = \xi \mathbf{e}_i + \mathbf{e}_k$, $\xi \in \mathbb{R}$ beliebig, $i \neq k$, so reduziert sich die quadratische Form $Q(\mathbf{x})$ auf $a_{ii}\xi^2 + 2a_{ik}\xi + a_{kk} > 0$ für alle $\xi \in \mathbb{R}$. Die quadratische Gleichung $a_{ii}\xi^2 + 2a_{ik}\xi + a_{kk} = 0$ hat keine reellen Lösungen in ξ , folglich ist ihre Diskriminante $4a_{ik}^2 - 4a_{ii}a_{kk} < 0$, woraus sich (2.89) ergibt. Nimmt man schließlich an, das betragsgrößte Matrixelement liege nicht in der Diagonale, so steht diese Annahme im Widerspruch zu (2.89). \square

Eine notwendige und hinreichende Bedingung für die positive Definitheit einer symmetrischen Matrix gewinnt man mit der Methode der Reduktion einer quadratischen Form auf eine Summe von Quadraten. Wir dürfen $a_{11} > 0$ annehmen, denn andernfalls wäre \mathbf{A} nach Satz 2.14 nicht positiv definit. Folglich lassen sich alle Terme in (2.87), welche x_1 enthalten, zu einem vollständigen Quadrat ergänzen:

$$\begin{aligned} Q(\mathbf{x}) &= a_{11}x_1^2 + 2 \sum_{i=2}^n a_{i1}x_1x_i + \sum_{i=2}^n \sum_{k=2}^n a_{ik}x_ix_k \\ &= \left[\sqrt{a_{11}}x_1 + \sum_{i=2}^n \frac{a_{i1}}{\sqrt{a_{11}}}x_i \right]^2 + \sum_{i=2}^n \sum_{k=2}^n \left(a_{ik} - \frac{a_{i1}a_{1k}}{a_{11}} \right) x_ix_k \\ &= \left[\sum_{i=1}^n l_{i1}x_i \right]^2 + \sum_{i=2}^n \sum_{k=2}^n a_{ik}^{(1)}x_ix_k = \left[\sum_{i=1}^n l_{i1}x_i \right]^2 + Q^{(1)}(\mathbf{x}^{(1)}) \end{aligned} \quad (2.91)$$

Dabei bedeuten

$$l_{11} = \sqrt{a_{11}}; \quad l_{i1} = \frac{a_{i1}}{\sqrt{a_{11}}} = \frac{a_{i1}}{l_{11}}, \quad i = 2, 3, \dots, n; \quad (2.92)$$

$$a_{ik}^{(1)} = a_{ik} - \frac{a_{i1}a_{1k}}{a_{11}} = a_{ik} - l_{i1}l_{1k}, \quad i, k = 2, 3, \dots, n. \quad (2.93)$$

$Q^{(1)}(\mathbf{x}^{(1)})$ ist eine quadratische Form in den $(n-1)$ Variablen x_2, x_3, \dots, x_n mit den Koeffizienten $a_{ik}^{(1)}$ (2.93), wie sie sich im Gaußschen Algorithmus im ersten Eliminationsschritt mit dem Pivotelement a_{11} ergeben. Sie gehört zur Matrix des reduzierten Gleichungssystems.

Satz 2.15. Die symmetrische Matrix $\mathbf{A} = (a_{ik})$ mit $a_{11} > 0$ ist genau dann positiv definit, falls die reduzierte Matrix $\mathbf{A}^{(1)} = (a_{ik}^{(1)}) \in \mathbb{R}^{(n-1),(n-1)}$ mit den Elementen $a_{ik}^{(1)}$ gemäß (2.93) positiv definit ist.

Beweis. a) Notwendigkeit:

Es sei \mathbf{A} positiv definit. Zu jedem Vektor $\mathbf{x}^{(1)} = (x_2, x_3, \dots, x_n)^T \neq \mathbf{0}$ kann der Wert x_1

wegen $a_{11} > 0$ und damit $l_{11} \neq 0$ so bestimmt werden, dass $\sum_{i=1}^n l_{i1}x_i = 0$ ist. Für den zugehörigen Vektor $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \neq \mathbf{0}$ ist dann wegen (2.91) $0 < Q^{(1)}(\mathbf{x}^{(1)})$, und folglich muss $\mathbf{A}^{(1)}$ notwendigerweise positiv definit sein.

b) *Hinlänglichkeit:*

Es sei $\mathbf{A}^{(1)}$ positiv definit. Somit gilt für alle $\mathbf{x} \neq \mathbf{0}$ wegen (2.91) $Q(\mathbf{x}) \geq 0$. Es kann $Q(\mathbf{x}) = 0$ nur dann sein, falls in (2.91) beide Summanden gleichzeitig verschwinden. Aus $Q^{(1)}(\mathbf{x}^{(1)}) = 0$ folgt jetzt aber $x_2 = x_3 = \dots = x_n = 0$, und der erste Summand verschwindet wegen $l_{11} \neq 0$ dann nur für $x_1 = 0$. Die Matrix \mathbf{A} ist somit notwendigerweise positiv definit. \square

Aus Satz 2.15 folgen unmittelbar weitere Aussagen, die für die praktische Lösung von symmetrischen, positiv definiten Systemen bedeutungsvoll sind.

Satz 2.16. *Eine symmetrische Matrix $\mathbf{A} = (a_{ik}) \in \mathbb{R}^{n,n}$ ist genau dann positiv definit, wenn der Gaußsche Eliminationsprozess mit Diagonalstrategie durchführbar ist und die Pivotelemente positiv sind.*

Beweis. Ist \mathbf{A} positiv definit, so steht notwendigerweise mit $a_{11} > 0$ ein erstes Pivotelement in der Diagonale zur Verfügung. Die reduzierte Matrix $\mathbf{A}^{(1)}$ ist dann nach Satz 2.15 wieder positiv definit, und es ist $a_{22}^{(1)} > 0$ das zweite zulässige Pivotelement. Das gilt analog für alle weiteren reduzierten Matrizen $\mathbf{A}^{(k)}$, $k = 2, 3, \dots, n-1$, und es ist insbesondere auch $a_{nn}^{(n-1)}$ als letztes Pivot positiv.

Sind umgekehrt alle Pivotelemente $a_{11} > 0, a_{22}^{(1)} > 0, \dots, a_{nn}^{(n-1)} > 0$, so ist die Matrix $\mathbf{A}^{(n-1)} = (a_{nn}^{(n-1)})$ positiv definit und deshalb sind dann unter sukzessiver und sinngemäßer Anwendung von Satz 2.15 auch die Matrizen $\mathbf{A}^{(n-2)}, \dots, \mathbf{A}^{(1)}, \mathbf{A}$ positiv definit. \square

Nach Satz 2.16 ist für die Klasse der symmetrischen und positiv definiten Matrizen \mathbf{A} die Dreieckszerlegung mit dem Gauß-Algorithmus ohne Zeilenvertauschungen durchführbar. Da aber nach (2.93) die Matrizen der reduzierten Gleichungssysteme wieder symmetrisch sind, bedeutet dies für die Rechenpraxis eine Reduktion des Rechenaufwandes für die Zerlegung auf etwa die Hälfte.

Satz 2.17. *Eine symmetrische Matrix $\mathbf{A} = (a_{ik}) \in \mathbb{R}^{n,n}$ ist genau dann positiv definit, falls die Reduktion der quadratischen Form $Q(\mathbf{x})$ auf eine Summe von n Quadraten*

$$Q(\mathbf{x}) = \sum_{i=1}^n \sum_{k=1}^n a_{ik}x_i x_k = \sum_{k=1}^n \left[\sum_{i=k}^n l_{ik}x_i \right]^2 \quad (2.94)$$

im Körper der reellen Zahlen vollständig durchführbar ist.

Beweis. Auf Grund von Satz 2.16 ist die Behauptung offensichtlich, falls wir in Ergänzung zu (2.92) und (2.93) die folgenden Größen erklären, die im allgemeinen k -ten Reduktionsschritt

anfallen:

$$l_{kk} = \sqrt{a_{kk}^{(k-1)}}; \quad l_{ik} = \frac{a_{ik}^{(k-1)}}{l_{kk}}, \quad i = k+1, k+2, \dots, n, \quad (2.95)$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik}l_{jk}, \quad i, j = k+1, k+2, \dots, n. \quad (2.96)$$

Die Radikanden in (2.95) sind nach Satz 2.16 genau dann positiv, falls \mathbf{A} positiv definit ist. \square

Mit den Größen l_{ik} , welche durch (2.92) und (2.95) für $i \geq k$ eingeführt worden sind, definieren wir die Linksdreiecksmatrix

$$\mathbf{L} = \begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix} \quad (2.97)$$

Satz 2.18. Die Reduktion einer positiv definiten quadratischen Form auf eine Summe von Quadraten (2.94) leistet die Produktzerlegung der zugehörigen Matrix \mathbf{A} in

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T. \quad (2.98)$$

Beweis. Nach (2.94) besitzt die quadratische Form $Q(\mathbf{x})$ zwei verschiedene Darstellungen, die mit der Linksdreiecksmatrix \mathbf{L} (2.97) lauten

$$Q(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} = (\mathbf{L}^T \mathbf{x})^T (\mathbf{L}^T \mathbf{x}) = \mathbf{x}^T \mathbf{L} \mathbf{L}^T \mathbf{x}. \quad (2.99)$$

Wegen der Eindeutigkeit der Darstellung gilt (2.98). \square

Man nennt (2.98) die *Cholesky-Zerlegung* der symmetrischen, positiv definiten Matrix \mathbf{A} . Sie geht auf den Geodäten Cholesky [Ben 24] zurück.

Mit Hilfe der Cholesky-Zerlegung (2.98) lassen sich symmetrische, positiv definite Gleichungssysteme wie folgt lösen: Durch Substitution von $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ in $\mathbf{A}\mathbf{x} = \mathbf{b}$ ergibt sich

$$\mathbf{L}\mathbf{L}^T \mathbf{x} = \mathbf{b} \quad \text{oder} \quad \mathbf{L}(\mathbf{L}^T \mathbf{x}) = \mathbf{b}. \quad (2.100)$$

Mit dem Hilfsvektor $\mathbf{c} = \mathbf{L}^T \mathbf{x}$ kann somit die Auflösung von $\mathbf{A}\mathbf{x} = \mathbf{b}$ mittels der *Methode von Cholesky* in den drei Schritten erfolgen:

- | | |
|----|--|
| 1. | $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ (Cholesky-Zerlegung) |
| 2. | $\mathbf{L}\mathbf{c} = \mathbf{b}$ (Vorwärtssubstitution $\rightarrow \mathbf{c}$) |
| 3. | $\mathbf{L}^T \mathbf{x} = \mathbf{c}$ (Rücksubstitution $\rightarrow \mathbf{x}$) |
- (2.101)

Obwohl die Cholesky-Zerlegung zur Lösung von linearen Gleichungssystemen n Quadratwurzeln benötigt, die man im Gauß-Algorithmus nicht braucht, da ja bekanntlich die Berechnung der Lösung ein rationaler Prozess ist, hat sie den Vorteil, dass die Zerlegung unter Wahrung der Symmetrie erfolgt.

Beachtet man, dass in (2.96) nur die Matrixelemente $a_{ij}^{(k)}$ in und unterhalb der Diagonale zu berechnen sind, setzt sich der Rechenaufwand im k -ten Reduktionsschritt zusammen aus einer Quadratwurzelberechnung, $(n - k)$ Divisionen und $(1 + 2 + \dots + (n - k)) = \frac{1}{2}(n - k + 1)(n - k)$ Multiplikationen. Die vollständige Cholesky-Zerlegung erfordert somit neben der nicht ins Gewicht fallenden Berechnung von n Quadratwurzeln

$$\begin{aligned} Z_{LL}^T &= \{(n-1) + (n-2) + \dots + 1\} \\ &\quad + \frac{1}{2}\{n(n-1) + (n-1)(n-2) + \dots + 2 \cdot 1\} \\ &= \frac{1}{2}n(n-1) + \frac{1}{2}\left\{\frac{1}{6}n(n-1)(2n-1) + \frac{1}{2}n(n-1)\right\} \\ &= \frac{1}{6}(n^3 + 3n^2 - 4n) \end{aligned}$$

wesentliche Operationen. Die Prozesse der Vorwärts- und Rücksubstitution erfordern je den gleichen Rechenaufwand, weil die Diagonalelemente $l_{ii} \neq 1$ sind, nämlich

$$Z_V = Z_R = \frac{1}{2}(n^2 + n)$$

multiplikative Operationen. Damit beträgt der Rechenaufwand zur Lösung von n linearen Gleichungen nach der Methode von Cholesky

$$Z_{\text{Cholesky}} = \frac{1}{6}n^3 + \frac{3}{2}n^2 + \frac{1}{3}n = O(n^3) \quad (2.102)$$

wesentliche Operationen. Für größere n ist dieser Aufwand im Vergleich zu (2.37) etwa halb so groß.

Die detaillierte, algorithmische Zusammenfassung der drei Lösungsschritte (2.101) lautet wie folgt, wobei vorausgesetzt wird, dass nur die Elemente a_{ik} in und unterhalb der Diagonale vorgegeben sind. Die gegebenen Ausgangswerte werden durch den Algorithmus verändert.

für $k = 1, 2, \dots, n$:

falls $a_{kk} \leq 0$: STOP

$l_{kk} = \sqrt{a_{kk}}$

für $i = k + 1, k + 2, \dots, n$:

$l_{ik} = a_{ik}/l_{kk}$

für $j = k + 1, k + 2, \dots, i$:

$a_{ij} = a_{ij} - l_{ik} \times l_{jk}$

(2.103)

für $i = 1, 2, \dots, n$:

$s = b_i$

für $j = 1, 2, \dots, i - 1$:

$s = s - l_{ij} \times c_j$

$c_i = s/l_{ii}$

(2.104)

$$\begin{aligned}
 & \text{für } i = n, n-1, \dots, 1 : \\
 & \quad s = c_i \\
 & \text{für } k = i+1, i+2, \dots, n : \\
 & \quad s = s - l_{ki} \times x_k \\
 & \quad x_i = s / l_{ii}
 \end{aligned} \tag{2.105}$$

Da die gegebenen Matrixelemente a_{ik} in (2.103) verändert werden, und da der Wert von a_{ik} zuletzt bei der Berechnung von l_{ik} benötigt wird, kann die Matrix \mathbf{L} an der Stelle von \mathbf{A} aufgebaut werden. Dazu genügt es, das Feld l mit a zu identifizieren. Desgleichen kann in (2.104) der Vektor \mathbf{b} mit \mathbf{c} identifiziert werden, und in (2.105) ist der Lösungsvektor \mathbf{x} mit \mathbf{c} identifizierbar, so dass an der Stelle von \mathbf{b} die Lösung \mathbf{x} steht.

Um auch im Rechner von der Tatsache Nutzen zu ziehen, dass in der Methode von Cholesky nur mit der unteren Hälfte der Matrizen \mathbf{A} und \mathbf{L} gearbeitet wird, sind die relevanten Matrixelemente zeilenweise aufeinanderfolgend in einem eindimensionalen Feld zu speichern, wie dies in (2.106) angedeutet ist. Das Matrixelement a_{ik} findet sich als r -te Komponente in dem eindimensionalen Feld mit $r = \frac{1}{2}i(i-1) + k$. Der Speicherbedarf beträgt jetzt nur $S = \frac{1}{2}n(n+1)$, also gut die Hälfte im Vergleich zur normalen Speicherung einer Matrix.

$$A : \begin{array}{|cccccc|cccccc|ccc|} \hline & a_{11} & a_{21} & a_{22} & a_{31} & a_{32} & a_{33} & a_{41} & a_{42} & a_{43} & a_{44} & \cdot & \cdot & \cdot \\ \hline \end{array} \quad (2.106)$$

Speicherung der unteren Hälfte einer symmetrischen, positiv definiten Matrix.

Beispiel 2.10. Das Cholesky-Verfahren für $Ax = b$ mit

$$\mathbf{A} = \begin{pmatrix} 5 & 7 & 3 \\ 7 & 11 & 2 \\ 3 & 2 & 6 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

liefert bei fünfstelliger, dezimaler Gleitpunktarithmetik die beiden reduzierten Matrizen

$$\mathbf{A}^{(1)} = \begin{pmatrix} 1.2000 & -2.1999 \\ -2.1999 & 4.2001 \end{pmatrix}, \quad \mathbf{A}^{(2)} = (0.16680)$$

und die Linksdreiecksmatrix \mathbf{L} , den Vektor \mathbf{c} als Ergebnis der Vorwärtssubstitution und die Näherungslösung $\tilde{\mathbf{x}}$

$$L = \begin{pmatrix} 2.2361 & 0 & 0 \\ 3.1305 & 1.0954 & 0 \\ 1.3416 & -2.0083 & 0.40841 \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ 0 \\ 2.4485 \end{pmatrix}, \quad \tilde{x} = \begin{pmatrix} -18.984 \\ 10.991 \\ 5.9952 \end{pmatrix}.$$

Die Einsetzprobe ergibt den Residuenvektor $r = (2.6, 3.4, 1.2)^T \cdot 10^{-3}$. Die Konditionszahl bezüglich der Spektralnorm beträgt $\kappa(\mathbf{A}) \doteq 1.50 \cdot 10^3$. Ein Nachiterationsschritt liefert mit der Korrektur $z \doteq (-15.99, 8.99, 4.80)^T \cdot 10^{-3}$ eine verbesserte Näherung, die auf fünf wesentliche Stellen mit der exakten Lösung $x = (-19, 11, 6)^T$ übereinstimmt. Die Näherungslösung \tilde{x} ist bedeutend genauer ausgefallen, als die Daumenregel erwarten ließe. \triangle

2.3.2 Bandgleichungen

Man spricht von einer *Bandmatrix* \mathbf{A} , falls alle von null verschiedenen Elemente a_{ik} in der Diagonale und in einigen dazu benachbarten Nebendiagonalen liegen. Für die Anwendungen sind die symmetrischen, positiv definiten Bandmatrizen besonders wichtig.

Definition 2.19. Unter der *Bandbreite* m einer symmetrischen Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ versteht man die kleinste natürliche Zahl $m < n$, so dass gilt

$$a_{ik} = 0 \text{ für alle } i \text{ und } k \text{ mit } |i - k| > m. \quad (2.107)$$

Die Bandbreite m gibt somit die Anzahl der Nebendiagonalen unterhalb, bzw. oberhalb der Diagonalen an, welche die i.a. von null verschiedenen Matrixelemente enthalten.

Satz 2.20. Die Linksdreiecksmatrix \mathbf{L} der Cholesky-Zerlegung $\mathbf{A} = \mathbf{LL}^T$ (2.98) einer symmetrischen, positiv definiten Bandmatrix mit der Bandbreite m besitzt dieselbe Bandstruktur, denn es gilt

$$l_{ik} = 0 \text{ für alle } i \text{ und } k \text{ mit } i - k > m. \quad (2.108)$$

Beweis. Es genügt zu zeigen, dass der erste Reduktionsschritt, beschrieben durch (2.92) und (2.93), in der ersten Spalte von \mathbf{L} unterhalb der Diagonale nur in den m Nebendiagonalen von null verschiedene Elemente produziert, und dass die reduzierte Matrix $\mathbf{A}^{(1)} = (a_{ik}^{(1)})$ dieselbe Bandbreite m aufweist. Die erste Behauptung ist offensichtlich wegen (2.92) richtig, denn es ist $l_{i1} = 0$ für alle i mit $i - 1 > m$, da dann nach Voraussetzung $a_{i1} = 0$ ist. Für die zweite Behauptung brauchen wir aus Symmetriegründen nur Elemente $a_{ik}^{(1)}$ unterhalb der Diagonale zu betrachten. Für eine beliebige Stelle (i, k) mit $i \geq k \geq 2$ und $i - k > m$ ist einerseits nach Voraussetzung $a_{ik} = 0$ und andererseits auf Grund der eben gemachten Feststellung $l_{i1} = 0$, denn es ist $i - 1 > i - k > m$. Damit gilt wegen (2.93) in der Tat $a_{ik}^{(1)} = 0$ für alle $i, k \geq 2$ mit $|i - k| > m$. \square

Nach Satz 2.20 verläuft die Cholesky-Zerlegung einer symmetrischen, positiv definiten Bandmatrix vollständig innerhalb der Diagonale und den m unteren Nebendiagonalen, so dass die Matrix \mathbf{L} genau den Platz des wesentlichen gegebenen Teils der Matrix \mathbf{A} einnehmen kann. Zudem ist klar, dass jeder Reduktionsschritt nur die Elemente im Band innerhalb eines dreieckigen Bereiches erfasst, der höchstens die m nachfolgenden Zeilen umfasst. Zur Verdeutlichung ist in Abb. 2.1 ein allgemeiner Reduktionsschritt im Fall einer Bandmatrix mit $m = 4$ schematisch dargestellt.

Der Rechenaufwand für einen allgemeinen Reduktionsschritt setzt sich zusammen aus einer Quadratwurzelberechnung für l_{kk} , m Divisionen für die Werte l_{ik} und $\frac{1}{2}m(m+1)$ Multiplikationen für die eigentliche Reduktion der Elemente. Der totale Aufwand für die Cholesky-Zerlegung einer Bandmatrix der Ordnung n und der Bandbreite m beträgt also n Quadratwurzelberechnungen und weniger als $\frac{1}{2}nm(m+3)$ wesentliche Operationen. Er ist somit nur noch proportional zur Ordnung n und zum Quadrat der Bandbreite m . Die Prozesse der

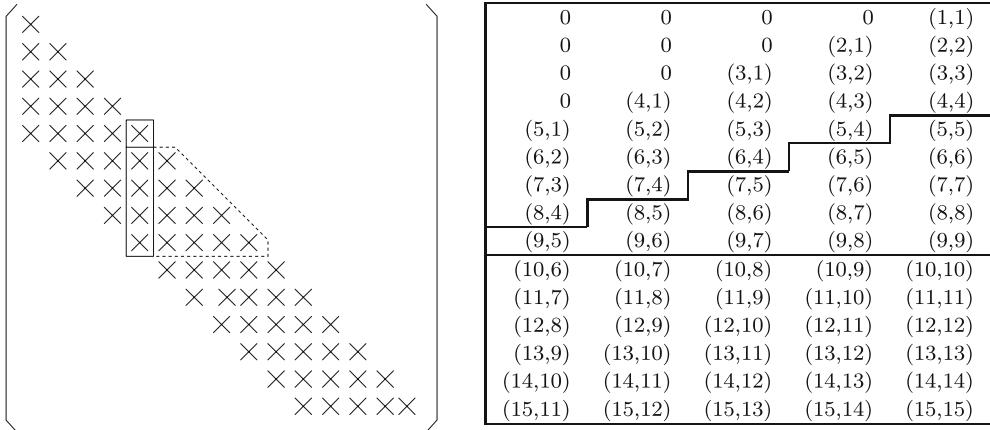


Abb. 2.1 Zur Reduktion und Speicherung einer symmetrischen, positiv definiten Bandmatrix.

Vorwärts- und Rücksubstitution sind mit je höchstens $n(m + 1)$ multiplikativen Operationen durchführbar. Es gilt folglich die Abschätzung des Aufwands zur Lösung von n linearen Gleichungen mit symmetrischer, positiv definiter Bandmatrix der Bandbreite m

$$Z_{\text{Cholesky}}^{\text{(Band)}} \leq \frac{1}{2}nm(m + 3) + 2n(m + 1) = O(nm^2) \quad (2.109)$$

Um die Speicherung der Bandmatrix zu optimieren, wird ihre untere Hälfte in der Art von Abb. 2.1 in einem rechteckigen Feld von n Zeilen und $(m + 1)$ Spalten gespeichert. Dabei soll vereinbart werden, dass die einzelnen Nebendiagonalen von \mathbf{A} als Spalten erscheinen und zwar so, dass der i -ten Zeile von \mathbf{A} auch die i -te Zeile in dem Feld entspricht. Die Diagonalelemente von \mathbf{A} sind in der $(m+1)$ -ten Spalte des Feldes zu finden, und das Element a_{ik} von \mathbf{A} mit $\max(i - m, 1) \leq k \leq i$ steht in der $(k - i + m + 1)$ -ten Spalte des Feldes. Die im linken oberen Dreieksbereich des Feldes undefinierten Elemente können zweckmäßigerweise gleich null gesetzt werden. Rechts in Abb. 2.1 findet sich am Speicherplatz des Elementes a_{ik} der entsprechende Doppelindex.

Die algorithmische Fassung der Cholesky-Zerlegung $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ einer Bandmatrix der Ordnung n und der Bandbreite m in der Speicherung nach Abb. 2.1 lautet:

für $k = 1, 2, \dots, n :$	
falls $a_{k,m+1} \leq 0$: STOP	
$l_{k,m+1} = \sqrt{a_{k,m+1}}$	
$p = \min(k + m, n)$	
für $i = k + 1, k + 2, \dots, p :$	
$l_{i,k-i+m+1} = a_{i,k-i+m+1} / l_{k,m+1}$	
für $j = k + 1, k + 2, \dots, i :$	
$a_{i,j-i+m+1} = a_{i,j-i+m+1} - l_{i,k-i+m+1} \times l_{j,k-j+m+1}$	

2.3.3 Tridiagonale Gleichungssysteme

Als besonders einfach zu behandelnde Gleichungssysteme werden wir in verschiedenen Anwendungen solche mit *tridiagonaler* Koeffizientenmatrix \mathbf{A} antreffen. Die allgemeine i -te Gleichung enthält nur die Unbekannten x_{i-1}, x_i und x_{i+1} . Um der sehr speziellen Struktur der Matrix Rechnung zu tragen, und um auch die Realisierung auf einem Rechner zu vereinfachen, gehen wir von folgendem Gleichungssystem mit $n = 5$ Unbekannten aus.

$$\begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & 1 \\ \hline a_1 & b_1 & & & & d_1 \\ c_2 & a_2 & b_2 & & & d_2 \\ c_3 & a_3 & b_3 & & & d_3 \\ c_4 & a_4 & b_4 & & & d_4 \\ c_5 & a_5 & b_5 & & & d_5 \end{array} \quad (2.110)$$

Wir setzen zunächst voraus, der Gauß-Algorithmus sei mit Diagonalstrategie, d.h. ohne Zeilenvertauschungen durchführbar, weil \mathbf{A} beispielsweise diagonal dominant oder symmetrisch und positiv definit ist. Dann existiert also die Dreieckszerlegung $\mathbf{A} = \mathbf{L}\mathbf{R}$, und man verifiziert leicht, dass \mathbf{L} eine *bidiagonale* Linksdreiecksmatrix und \mathbf{R} eine bidiagonale Rechtsdreiecksmatrix ist. Wir können deshalb für die Dreieckszerlegung direkt den Ansatz verwenden.

$$\begin{pmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ c_3 & a_3 & b_3 & & \\ c_4 & a_4 & b_4 & & \\ c_5 & a_5 & b_5 & & \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & l_2 & 1 & & \\ & & l_3 & 1 & \\ & & & l_4 & 1 \end{pmatrix} \cdot \begin{pmatrix} m_1 & r_1 & & & \\ m_2 & r_2 & & & \\ m_3 & r_3 & & & \\ m_4 & r_4 & & & \\ m_5 & r_5 & & & \end{pmatrix}, \quad (2.111)$$

und die unbekannten Größen l_i, m_i, r_i durch Koeffizientenvergleich bestimmen. Man erhält die Bestimmungsgleichungen

$$\begin{aligned} a_1 &= m_1, & b_1 &= r_1, \\ c_2 = l_1 m_1, & a_2 = l_1 r_1 + m_2, & b_2 &= r_2, \\ c_3 = l_2 m_2, & a_3 = l_2 r_2 + m_3, & b_3 &= r_3, \\ c_4 = l_3 m_3, & a_4 = l_3 r_3 + m_4, & b_4 &= r_4, \\ c_5 = l_4 m_4, & a_5 = l_4 r_4 + m_5. & & \end{aligned} \quad (2.112)$$

Aus (2.112) bestimmen sich die Unbekannten sukzessive in der Reihenfolge $m_1; r_1, l_1, m_2; r_2, l_2, m_3; \dots; r_4, l_4, m_5$. Da $r_i = b_i$ für alle i gilt, lautet der Algorithmus zur Zerlegung der tridiagonalen Matrix \mathbf{A} (2.110) für allgemeines n :

$$\begin{aligned} m_1 &= a_1 \\ \text{für } i &= 1, 2, \dots, n-1 : \\ l_i &= c_{i+1}/m_i \\ m_{i+1} &= a_{i+1} - l_i \times b_i \end{aligned} \quad (2.113)$$

Die Vorwärts- und Rücksubstitution $\mathbf{Ly} = \mathbf{d}$ und $\mathbf{Rx} = \mathbf{y}$ lassen sich in den folgenden einfachen Rechenvorschriften zusammenfassen:

$y_1 = d_1$ für $i = 2, 3, \dots, n$: $y_i = d_i - l_{i-1} \times y_{i-1}$	(2.114)
---	---------

$x_n = y_n/m_n$ für $i = n-1, n-2, \dots, 1$: $x_i = (y_i - b_i \times x_{i+1})/m_i$	(2.115)
---	---------

Ein Programm zur Lösung eines tridiagonalen Gleichungssystems mit dem Gaußschen Algorithmus mit Diagonalstrategie besteht also im Wesentlichen aus drei simplen Schleifenanweisungen. Die Zahl der wesentlichen Rechenoperationen beträgt für die drei Lösungsschritte insgesamt

$Z_{\text{Gauss}}^{(\text{trid})} = 2(n-1) + (n-1) + 1 + 2(n-1) = 5n - 4 = O(n)$	(2.116)
--	---------

Der Rechenaufwand ist somit nur proportional zur Zahl der Unbekannten. Selbst große tridiagonale Gleichungssysteme lassen sich mit relativ kleinem Rechenaufwand lösen. Daselbe gilt auch, falls der Gauß-Algorithmus mit Zeilenvertauschungen durchgeführt werden muss. Wir erklären das Prinzip wieder am System (2.110) und legen den Betrachtungen die relative Spaltenmaximumstrategie zu Grunde. Als Pivotelemente für den ersten Eliminationsschritt kommen nur die beiden Matrixelemente a_1 und c_2 in Betracht. Wir berechnen die beiden Hilfsgrößen

$$\alpha := |a_1| + |b_1|, \quad \beta := |c_2| + |a_2| + |b_2|. \quad (2.117)$$

Falls $|a_1|/\alpha \geq |c_2|/\beta$ gilt, ist a_1 Pivotelement, andernfalls ist eine Zeilenvertauschung erforderlich. In diesem Fall entsteht an der Stelle (1,3) ein im Allgemeinen von null verschiedenes Element, das außerhalb des tridiagonalen Bandes zu liegen kommt. Um den Eliminationsschritt einheitlich beschreiben zu können, werden die folgenden Größen definiert.

$$\begin{aligned} \text{Falls } a_1 \text{ Pivot : } & \left\{ \begin{array}{llll} r_1 := a_1, & s_1 := b_1, & t_1 := 0, & f_1 := d_1 \\ u := c_2, & v := a_2, & w := b_2, & z := d_2 \end{array} \right. \\ \text{Falls } c_2 \text{ Pivot : } & \left\{ \begin{array}{llll} r_1 := c_2, & s_1 := a_2, & t_1 := b_2, & f_1 := d_2 \\ u := a_1, & v := b_1, & w := 0, & z := d_1 \end{array} \right. \end{aligned} \quad (2.118)$$

Mit diesen Variablen erhält (2.110) die Gestalt

	x_1	x_2	x_3	x_4	x_5	1	
	r_1	s_1	t_1			f_1	
	u	v	w			z	
	c_3	a_3	b_3			d_3	
		c_4	a_4	b_4		d_4	
			c_5	a_5		d_5	

(2.119)

Der erste Eliminationsschritt ergibt

x_1	x_2	x_3	x_4	x_5	1
r_1	s_1	t_1	f_1		
l_1	a'_2	b'_2			d'_2
	c_3	a_3	b_3		d_3
		c_4	a_4	b_4	d_4
			c_5	a_5	d_5

(2.120)

mit den Zahlenwerten

$$l_1 := u/r_1; \quad a'_2 := v - l_1 s_1, \quad b'_2 := w - l_1 t_1, \quad d'_2 := z - l_1 f_1. \quad (2.121)$$

Für das reduzierte System ergibt sich damit die gleiche Situation, denn die reduzierte Matrix ist wiederum tridiagonal. Die Überlegungen für den ersten Schritt lassen sich sinngemäß anwenden. Damit die Formeln (2.117) und (2.118) ihre Gültigkeit auch für den letzten ($n - 1$)-ten Eliminationsschritt behalten, muss $b_n = 0$ vereinbart werden. Die konsequente Fortsetzung der Elimination führt zum Schlussschema

x_1	x_2	x_3	x_4	x_5	1
r_1	s_1	t_1	f_1		
l_1	r_2	s_2	t_2		f_2
l_2	r_3	s_2	t_3		f_3
l_3	r_4		s_4		f_4
	l_4		r_5		f_5

(2.122)

Der Gauß-Algorithmus für ein tridiagonales Gleichungssystem (2.110) unter Verwendung der relativen Spaltenmaximumstrategie lautet auf Grund der Formeln (2.117), (2.118) und (2.121) unter Einschluss der Vorrwärtssubstitution:

für $i = 1, 2, \dots, n - 1$:

$$\alpha = |a_i| + |b_i|; \quad \beta = |c_{i+1}| + |a_{i+1}| + |b_{i+1}|$$

falls $|a_i|/\alpha \geq |c_{i+1}|/\beta$:

$$r_i = a_i; \quad s_i = b_i; \quad t_i = 0; \quad f_i = d_i;$$

$$u = c_{i+1}; \quad v = a_{i+1}; \quad w = b_{i+1}; \quad z = d_{i+1};$$

sonst

$$r_i = c_{i+1}; \quad s_i = a_{i+1}; \quad t_i = b_{i+1}; \quad f_i = d_{i+1};$$

$$u = a_i; \quad v = b_i; \quad w = 0; \quad z = d_i$$

$$l_i = u/r_i; \quad a_{i+1} = v - l_i \times s_i$$

$$b_{i+1} = w - l_i \times t_i; \quad d_{i+1} = z - l_i \times f_i$$

$$r_n = a_n; \quad f_n = d_n$$
(2.123)

In der algorithmischen Beschreibung (2.123) sind die Bezeichnungen der Rechenschemata (2.119) und (2.122) verwendet worden. Da die Koeffizienten a_i, b_i, c_i, d_i der gegebenen Gleichungen verändert werden, können in (2.123) die folgenden Variablen gleichgesetzt werden: $r_i = a_i, s_i = b_i, l_i = c_{i+1}, f_i = d_i$. Damit kann Speicherplatz gespart werden, und (2.123) kann etwas vereinfacht werden.

Die Unbekannten x_i werden durch Rücksubstitution wie folgt geliefert:

$$\boxed{\begin{aligned}x_n &= f_n/r_n \\x_{n-1} &= (f_{n-1} - s_{n-1} \times x_n)/r_{n-1} \\&\text{für } i = n-2, n-3, \dots, 1 : \\x_i &= (f_i - s_i \times x_{i+1} - t_i \times x_{i+2})/r_i\end{aligned}} \quad (2.124)$$

Der totale Rechenaufwand an multiplikativen Operationen zur Auflösung eines allgemeinen tridiagonalen Gleichungssystems in n Unbekannten beträgt unter Einschluss der beiden Divisionen für die Pivotbestimmung

$$Z_{\text{Gauss}}^{(\text{trid,allg})} = 5(n-1) + (n-1) + 3(n-1) = 9(n-1) = O(n).$$

Im Vergleich zu (2.116) verdoppelt sich der Rechenaufwand etwa, falls eine Pivotierung notwendig ist. Die Linksdreiecksmatrix \mathbf{L} der Zerlegung $\mathbf{PA} = \mathbf{LR}$ ist zwar noch *bidiagonal*, aber \mathbf{R} ist eine Rechtsdreiecksmatrix, in der zwei obere Nebendiagonalen im Allgemeinen von null verschiedene Matrixelemente enthalten.

2.4 Verfahren für Vektorrechner und Parallelrechner

Die in den vorangegangenen Abschnitten behandelten klassischen Verfahren zur Lösung von linearen Gleichungssystemen sind für sequentiell arbeitende Skalarrechner konzipiert worden und eignen sich deshalb schlecht für eine Implementierung auf modernen Superrechnern, weil sie ihrer sehr speziellen Arbeitsweise nicht Rechnung tragen. Deshalb mussten für Vektor- und Parallelrechner verschiedener Architekturen angepasste Rechenverfahren entwickelt werden, um die technischen Möglichkeiten der Computer effizient auszunutzen. Auf die wichtigsten Software-Pakete gehen wir in Abschnitt 2.6 ein. Das Kunststück besteht darin, die wichtigsten Algorithmen-Teile so zu formulieren, dass sie für die allgemeine Anwendung auf Rechnern stark unterschiedlicher Charakteristik geeignet sind, und diese Teile von den speziellen rechnerabhängigen Teilen zu trennen. Die Zusammenhänge zwischen Architektur-Parametern wie Cache-Größe, Anzahl der Prozessoren, optimale Feldgröße bei Vektorprozessoren und den algorithmischen Parametern wie der gewählten Methode (z.B. Zeilen- oder Spalten-orientiert) oder der möglichen Blockgröße sind sehr komplex. Andererseits ist die Portabilität von guten Softwaresystemen von großer Bedeutung. Wie viele technische Details von Software und Hardware dabei eine Rolle spielen, ist z.B. gut nachvollziehen beim Studium der LAPACK Working Note 100 [Cho 95].

Mehr über die Architektur von Höchstleistungsrechnern sowie über parallele und Vektor-Algorithmen findet man in [Ale 02, Bre 92, Cos 95, Cul 99, Fre 92, Fro 90, Don 93, Gal 90a, Hoc 88, Lei 97, Ort 88, vdV 90]. Im Folgenden sollen an zwei typischen Aufgabenstellungen die notwendigen Modifikationen oder aber grundsätzlich neue Lösungswege beschrieben werden. Auf den Gauß-Algorithmus für parallele und Vektorrechner gehen die meisten der genannten Referenzen ein, einen ganzen Band widmet ihm Robert [Rob 91].

2.4.1 Voll besetzte Systeme

Voll besetzte Systeme auf Parallelrechnern

Um die gängigen Algorithmen zu Matrix-Zerlegungen und zur Lösung linearer Gleichungssysteme mit Hilfe dieser Zerlegungen an die Architekturen von Parallelrechnern anzupassen, muss besonders auf effizienten Speicherzugriff geachtet werden. Die Erweiterung SCALAPACK der Programmmbibliothek LAPACK [And 99], siehe auch Abschnitt 2.6, tut dies, indem sie die Algorithmen so reorganisiert, dass in den innersten Schleifen des Algorithmus Blockmatrix-Operationen wie Matrix-Multiplikation verwendet werden. Diese Operationen können dann auf den verschiedenen Architekturen unabhängig von den LAPACK-Routinen optimiert werden. Für diese Reorganisation wollen wir ein Beispiel geben, in dem wir die *LR*-Zerlegung mit dem Gauß-Algorithmus nach [Cho 96] betrachten.

Zu Grunde gelegt wird ein Gitter von $P \times Q$ Prozessoren mit eigenem Speicherplatz und der Möglichkeit zur Kommunikation untereinander. Wie sehr Einzelheiten dieser Struktur den algorithmischen Ablauf und seine Effizienz bestimmen, wurde schon erwähnt. Hier soll deshalb nur ein Algorithmus beschrieben werden, der die Gauß-Elimination blockweise durchführt und die Daten zyklisch verteilt.

Sei $m_b \times n_b$ die Blockgröße der Untermatrizen. Blöcke mit einem festen Abstand in Spalten- und Zeilenrichtung werden demselben Prozessor zugeordnet. Die Zuordnung wird für ein 2×3 -Prozessoren-Gitter und eine Matrix mit 12×12 Blöcken in Abb. 2.2 beschrieben. Die kursiv gedruckten Zahlen links und oben bezeichnen die Block-Indizes der entsprechenden Zeilen und Spalten. Die kleinen Rechtecke in der linken Abbildung enthalten die Nummern der Prozessoren, denen die entsprechenden Blöcke zugeordnet sind. Die rechte Abbildung zeigt diese Verteilung aus der Sicht der Prozessoren; jedem sind 6×4 Blöcke zugeordnet. Diese zyklische Verteilung ist die einzige, die SCALAPACK unterstützt. Sie kann die meisten Datenverteilungen in Algorithmen der numerischen linearen Algebra reproduzieren. Für $P = Q = 1$ kehrt sie zur normalen Zeilen-Spalten-Struktur einer Matrix zurück.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	0	1	2	0	1	2	0	1	2
1	3	4	5	3	4	5	3	4	5	3	4	5
2	0	1	2	0	1	2	0	1	2	0	1	2
3	0	1	2	0	1	2	0	1	2	0	1	2
4	3	4	5	3	4	5	3	4	5	3	4	5
5	0	1	2	0	1	2	0	1	2	0	1	2
6	3	4	5	3	4	5	3	4	5	3	4	5
7	0	1	2	0	1	2	0	1	2	0	1	2
8	3	4	5	3	4	5	3	4	5	3	4	5
9	0	1	2	0	1	2	0	1	2	0	1	2
10	3	4	5	3	4	5	3	4	5	3	4	5
11	0	1	2	0	1	2	0	1	2	0	1	2

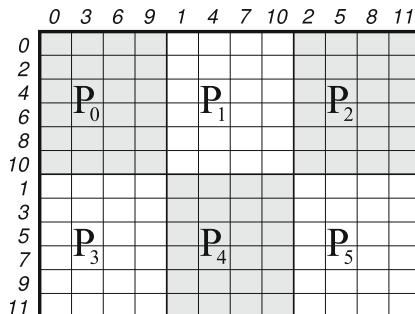


Abb. 2.2 Beispiel für die zyklische Verteilung der Matrix-Block-Daten.

Abhängig von den schon erwähnten Architektur-Parametern gibt es jetzt verschiedene Versionen des Gauß-Algorithmus, [Gal 90b, Cho 96, Fro 90]. Hier soll nur eine dieser Versionen

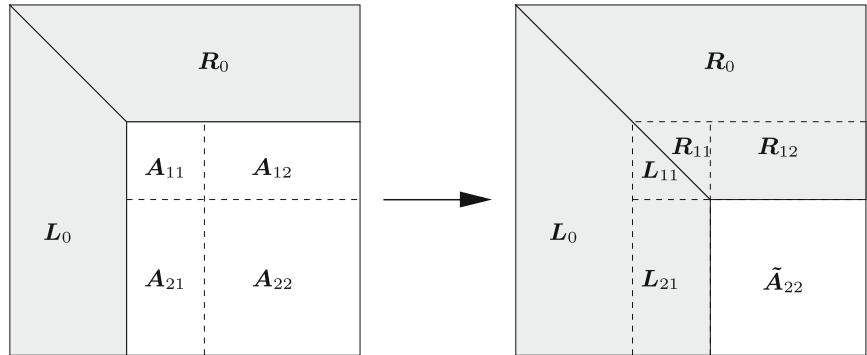


Abb. 2.3 Ein Block-Eliminationsschritt.

vorgestellt werden, wobei die eventuell notwendigen Zeilenvertauschungen im Nachhinein behandelt werden. Der Block-Algorithmus besteht im Wesentlichen aus den Schritten des skalaren Gauß-Algorithmus, jetzt für Blöcke aufgeschrieben; er soll aber hier rekursiv definiert werden.

Sei dazu $\mathbf{A} \in \mathbb{R}^{n,n}$, sei n durch n_b teilbar, $m = n/n_b$, weiter sei $P = Q$ und P durch m teilbar. Abb. 2.3 symbolisiert den k -ten Block-Eliminationsschritt; dazu sind die schon behandelten Matrixteile schattiert; sie ändern sich allenfalls noch durch Zeilenvertauschungen. Links sieht man die Matrix \mathbf{A} nach $k-1$ Block-Eliminationsschritten, der k -te Eliminationsschritt erzeugt Abb. 2.3 rechts. $\mathbf{A}^{(k)}$ sei die im k -ten Schritt noch zu behandelnde Matrix. Für sie und ihre LR -Zerlegung gilt

$$\begin{aligned} \mathbf{A}^{(k)} &= \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{L}_{11}\mathbf{R}_{11} & \mathbf{L}_{11}\mathbf{R}_{12} \\ \mathbf{L}_{21}\mathbf{R}_{11} & \mathbf{L}_{21}\mathbf{R}_{12} + \mathbf{L}_{22}\mathbf{R}_{22} \end{pmatrix}. \end{aligned} \quad (2.125)$$

Dabei sind $\mathbf{A}_{11} \in \mathbb{R}^{n_b, n_b}$ und $\mathbf{A}_{22} \in \mathbb{R}^{n-kn_b, n-kn_b}$; die anderen Dimensionen ergeben sich dementsprechend.

Für den k -ten Zerlegungsschritt ergeben sich nach (2.125) folgende Teilschritte:

- (1) Zerlege $\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{R}_{11} \rightarrow \mathbf{A}_{11}$.
- (2) Löse $\mathbf{R}_{11}^T \mathbf{L}_{21}^T = \mathbf{A}_{21}^T$ nach $\mathbf{L}_{21} \rightarrow \mathbf{A}_{21}$.
- (3) Löse $\mathbf{L}_{11}\mathbf{R}_{12} = \mathbf{A}_{12}$ nach $\mathbf{R}_{12} \rightarrow \mathbf{A}_{12}$.
- (4) Passe an $\mathbf{A}_{22} - \mathbf{L}_{21}\mathbf{R}_{12} \rightarrow \mathbf{A}_{22}$.

Der angepasste Block \mathbf{A}_{22} ist der weiter zu behandelnde Teil $\mathbf{A}^{(k+1)}$.

Wir betrachten jetzt die parallele Verarbeitung der Schritte (1)–(4) bei zyklischer Verteilung der Blöcke nach Abb. 2.2. Dabei wird zusätzlich mit Spaltenpivotisierung gearbeitet. Dies macht entsprechende Zeilenvertauschungen auch in \mathbf{L}_0 (Abb. 2.3) notwendig.

- (a) Finde das absolute Maximum in der ersten Spalte von $\mathbf{A}^{(k)}$. Dazu sucht jeder beteiligte Prozessor das absolute Maximum in seinem Teil dieser Spalte und durch Versenden von Wert und Ort wird das Maximum und sein Prozessor-Ort bestimmt.
- (b) Vertausche die entsprechenden Zeilen durch Versenden der Pivot-Information und der Zeileninhalte durch alle beteiligten Prozessoren.
- (c) Führe den Zerlegungsschritt (1) durch und löse (verteilt) die Gleichungssysteme aus Schritt (2) in der aktuellen $(n - (k - 1)n_b) \times n_b$ -Prozessoren-Spalte.
- (d) Versende \mathbf{L}_{11} in der aktuellen $n_b \times (n - (k - 1)n_b)$ -Prozessoren-Zeile, löse (verteilt) die Gleichungssysteme aus Schritt (3).
- (e) Verteile die Spaltenblöcke von \mathbf{L}_{21} zeilenweise, die Zeilenblöcke von \mathbf{R}_{12} spaltenweise und bilde (verteilt) das neue \mathbf{A}_{22} nach Schritt (4).

Es ist zu sehen, dass die Zeilenvertauschungen einen wesentlichen zusätzlichen Kommunikationsbedarf zwischen den Prozessoren erzeugen.

In [Cho 96] wird dieser Algorithmus mit dem Paket SCALAPACK, basierend auf PBLAS und kommunizierend mit BLACS auf verschiedenen Parallelrechnern realisiert. Auf einem Paragon-System mit 50 MHz i860XP Prozessoren wurde bei einer Blockgröße $n_b = 8$ auf einem 16×32 -Prozessorfeld schon 1994 eine Rechenleistung von mehr als 15 Gigaflops bei Matrixgrößen zwischen $n = 20\,000$ und $n = 35\,000$ erreicht (engl. flops = floating point operations). Diese Leistung kann gesteigert werden, wenn auf die Modularität der genannten Programme und ihre leichte Portabilität auf andere Parallelrechner verzichtet wird.

Voll besetzte Systeme auf Vektorrechnern

Der Gauß-Algorithmus zur Lösung eines linearen Gleichungssystems mit voll besetzter Matrix in vielen Unbekannten von Abschnitt 2.1 kann im Prinzip auf einfache Weise so modifiziert werden, dass Operationen für Vektoren ausführbar werden. Zur Vereinfachung der Notation soll die rechte Seite \mathbf{b} als $(n + 1)$ -te Spalte einer $n \times (n + 1)$ -Matrix \mathbf{A} aufgefasst werden, so dass das zu lösende System

$$\sum_{k=1}^n a_{ik}x_k = a_{i,n+1}, \quad i = 1, 2, \dots, n,$$

lautet. Die Rechenvorschrift (2.5) ist bereits eine typische Vektoroperation für die $(n - 1)$ Matrixelemente der ersten Spalte von \mathbf{A} , welche mit dem reziproken Wert des Pivotelementes a_{11} zu multiplizieren sind. Dieser Teilschritt wird zur vektorisierbaren Vorschrift

$$h = 1/a_{11}$$

für $i = 2, 3, \dots, n : \quad l_{i1} = h \times a_{i1}.$

(2.126)

Die Formeln (2.6) und (2.7) lassen sich wie folgt zusammenfassen, falls die Operationen *spaltenweise* ausgeführt werden:

$$\boxed{\begin{aligned} \text{für } k &= 2, 3, \dots, n+1 : \\ \text{für } i &= 2, 3, \dots, n : \quad a_{ik}^{(1)} = a_{ik} - l_{i1} \times a_{1k} \end{aligned}} \quad (2.127)$$

Die zweite Zeile von (2.127) ist eine typische *Triade*, bei der von den letzten ($n-1$) Elementen der k -ten Spalte das a_{1k} -fache der entsprechenden l -Werte zu subtrahieren sind. Wenn wir davon ausgehen, dass die Matrix \mathbf{A} spaltenweise gespeichert ist, und wenn die l_{i1} anstelle der a_{i1} und die Werte $a_{ik}^{(1)}$ anstelle von a_{ik} gespeichert werden, so betreffen die beiden Vektoroperationen (2.126) und (2.127) aufeinanderfolgend im Speicher angeordnete Zahlenwerte. Dies ist für eine optimale Ausführung der Operationen wichtig.

Die Fortsetzung des Gauß-Algorithmus betrifft in den weiteren Eliminationsschritten kleinere werdende, reduzierte Systeme, und die Vektoroperationen betreffen Vektoren mit abnehmender Anzahl von Komponenten. Dies ist für eine effiziente Ausführung ein Nachteil, weil ab einer kritischen Länge der Vektoroperationen infolge der notwendigen Aufstartzeit die Rechenzeit größer wird im Vergleich zur skalaren Ausführung.

In der Rechenvorschrift (2.17) der Rücksubstitution zur Berechnung der Unbekannten x_i

$$x_i = \left[c_i - \sum_{k=i+1}^n r_{ik} x_k \right] / r_{ii}, \quad i = n, n-1, \dots, 1,$$

tritt zwar ein vektorisierbares Skalarprodukt auf, jedoch mit Matrixelementen r_{ik} der i -ten Zeile, die im Speicher nicht aufeinanderfolgend angeordnet sind, so dass die Ausführung der Operation infolge von so genannten Bankkonflikten eine erhebliche Erhöhung der Rechenzeit zur Folge haben kann. Aus diesem Grund ist der Prozess der Rücksubstitution so zu modifizieren, dass nach Berechnung der Unbekannten x_i das x_i -fache der i -ten Teilspalte von \mathbf{R} zur Konstantenspalte addiert wird, was auf eine Triade führt, die auf je aufeinanderfolgend gespeicherte Vektorelemente auszuführen ist. In der Notation (2.16) der Endgleichungen erhält die Rücksubstitution die folgende vektorisierbare Form:

$$\boxed{\begin{aligned} \text{für } i &= n, n-1, \dots, 2 : \\ x_i &= c_i / r_{ii} \\ \text{für } j &= 1, 2, \dots, i-1 : \quad c_j = c_j - x_i \times r_{ji} \\ x_1 &= c_1 / r_{11} \end{aligned}} \quad (2.128)$$

Auch in (2.128) nimmt die Länge der Vektoroperationen von ($n-1$) auf 1 ab. Diesen Nachteil von Vektoroperationen abnehmender Länge kann man durch die Modifikation des Gauß-Algorithmus zum *Gauß-Jordan-Verfahren* vermeiden. Dazu wird die Gleichung, mit welcher eine Elimination der betreffenden Unbekannten erfolgt, durch den Koeffizienten (= Pivotelement) dividiert; außerdem wird im p -ten Eliminationsschritt mit $p \geq 2$ die Unbekannte x_p in allen anderen Gleichungen eliminiert. Aus (2.3) entsteht so mit den anders

bezeichneten Konstanten unter der Annahme $a_{11} \neq 0$ das Schema

$$\begin{array}{ccccc|c} x_1 & x_2 & x_3 & x_4 & 1 \\ \hline 1 & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} & a_{15}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & a_{25}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} & a_{35}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} & a_{45}^{(1)} \end{array} \quad (2.129)$$

Die Elemente von (2.129) sind wie folgt definiert:

$$\begin{aligned} a_{1k}^{(1)} &= a_{1k}/a_{11}, & k = 2, 3, \dots, n+1, \\ a_{ik}^{(1)} &= a_{ik} - a_{1i} \cdot a_{1k}^{(1)}, & i = 2, 3, \dots, n; k = 2, 3, \dots, n+1. \end{aligned} \quad (2.130)$$

Sei $a_{pp}^{(p-1)} \neq 0$. Dann lautet die Vorschrift für den p -ten Eliminationsschritt mit $p \geq 2$

$$\begin{aligned} a_{pk}^{(p)} &= a_{pk}^{(p-1)} / a_{pp}^{(p-1)}, & k = p+1, p+2, \dots, n+1, \\ a_{ik}^{(p)} &= a_{ik}^{(p-1)} - a_{ip}^{(p-1)} \cdot a_{pk}^{(p)}, & i = 1, 2, \dots, p-1, p+1, \dots, n; \\ && k = p+1, p+2, \dots, n+1. \end{aligned} \quad (2.131)$$

Als Folge dieser Modifikation steht im resultierenden Endschema die Einheitsmatrix, und die Unbekannten sind gegeben durch

$$x_k = a_{k,n+1}^{(n)}, \quad k = 1, 2, \dots, n.$$

Bei dieser Variante der Gleichungslösung ist keine Rücksubstitution erforderlich, doch steigt die Anzahl der wesentlichen Rechenoperationen auf rund $0.5n^3$ an, so dass der Rechenaufwand im Vergleich zum Gauß-Algorithmus etwa 50% größer ist. Da aber die Vektoroperationen in der Form von Triaden die konstante Länge n aufweisen, ist trotz des höheren Rechenaufwandes die Rechenzeit des Gauß-Jordan-Verfahrens dank der besseren Vektorsierbarkeit schon für relativ kleine n kleiner als diejenige des Gauß-Algorithmus. In einer zweckmäßigen algorithmischen Formulierung von (2.131) wird der Ausnahmeindex $i = p$ so behandelt, dass zunächst der falsche Wert $a_{pk}^{(p)} = 0$ berechnet wird, um ihn anschließend durch den richtigen Wert zu ersetzen. So ergibt sich der Algorithmus

$$\begin{aligned} &\text{für } p = 1, 2, \dots, n : \\ &\quad \text{für } k = p+1, p+2, \dots, n+1 : \\ &\quad\quad h = a_{pk}/a_{pp} \\ &\quad\quad \text{für } i = 1, 2, \dots, n : \quad a_{ik} = a_{ik} - h \times a_{ip} \\ &\quad\quad a_{pk} = h \\ &\quad \text{für } k = 1, 2, \dots, n : \quad x_k = a_{k,n+1} \end{aligned} \quad (2.132)$$

Der Algorithmus (2.132) arbeitet mit Diagonalstrategie. Muss eine andere Pivotstrategie verwendet werden, so ist das Pivotelement aus der p -ten Zeile zu bestimmen, und es sind Spaltenvertauschungen vorzunehmen, um so die Ausführung verlangsamtende Bankkonflikte zu vermeiden.

2.4.2 Tridiagonale Gleichungssysteme

Tridiagonale Gleichungssysteme auf Vektorrechnern

Wir betrachten im Folgenden den für viele Anwendungen wichtigen Fall, große tridiagonale Gleichungssysteme zu lösen, für welche Diagonalstrategie möglich ist. Dies trifft zu für Systeme mit diagonal dominanter oder symmetrischer und positiv definiter Matrix. Sowohl der Zerlegungsalgorithmus (2.113) wie auch die Prozesse der Vorwärts- und Rücksubstitution (2.114) und (2.115) sind in dem Sinn rekursiv, dass beispielsweise in (2.113) der Wert m_{i+1} von m_i abhängt. Aus diesem Grund können diese drei Prozesse nicht vektorisiert werden.

Um solche Gleichungssysteme auf Vektorrechnern effizient lösen zu können, wird das Prinzip der *zyklischen Reduktion* angewandt [Hoc 65]. Wir betrachten ein tridiagonales Gleichungssystem der Gestalt

$$\begin{aligned} a_1 x_1 + b_1 x_2 &= d_1, \\ c_i x_{i-1} + a_i x_i + b_i x_{i+1} &= d_i, \quad i = 2, 3, \dots, n-1, \\ c_n x_{n-1} + a_n x_n &= d_n. \end{aligned} \quad (2.133)$$

Die Idee besteht darin, aus je drei aufeinander folgenden Gleichungen zwei der fünf Unbekannten so zu eliminieren, dass in der resultierenden Gleichung nur Unbekannte mit Indizes derselben Parität auftreten. Wir wollen das prinzipielle Vorgehen unter der Annahme darlegen, dass n gerade sei, also $n = 2m, m \in \mathbb{N}^*$. Sei $i \in \mathbb{N}^*$ mit $2 < i < n$ gerade. Aus den drei Gleichungen

$$\begin{aligned} c_{i-1} x_{i-2} + a_{i-1} x_{i-1} + b_{i-1} x_i &= d_{i-1} \\ c_i x_{i-1} + a_i x_i + b_i x_{i+1} &= d_i \\ c_{i+1} x_i + a_{i+1} x_{i+1} + b_{i+1} x_{i+2} &= d_{i+1} \end{aligned}$$

eliminieren wir x_{i-1} und x_{i+1} , indem wir zur mittleren Gleichung das $(-c_i/a_{i-1})$ -fache der ersten und das $(-b_i/a_{i+1})$ -fache der dritten Gleichung addieren. Das Ergebnis lautet

$$\begin{aligned} -\frac{c_{i-1} c_i}{a_{i-1}} x_{i-2} + \left\{ -\frac{b_{i-1} c_i}{a_{i-1}} + a_i - \frac{b_i c_{i+1}}{a_{i+1}} \right\} x_i - \frac{b_i b_{i+1}}{a_{i+1}} x_{i+2} \\ = -\frac{d_{i-1} c_i}{a_{i-1}} + d_i - \frac{b_i d_{i+1}}{a_{i+1}}, \quad i = 2, 4, \dots, n. \end{aligned} \quad (2.134)$$

Damit (2.134) auch für $i = 2$ und $i = n$ gilt, definiert man die zusätzlichen Werte

$$c_1 = c_{n+1} = b_n = b_{n+1} = d_{n+1} = 0, \quad a_{n+1} = 1. \quad (2.135)$$

Mit dieser Vereinbarung führen wir die Koeffizienten der neuen Gleichungen für gerade Indexwerte i ein:

$$\left. \begin{aligned} c_i^{(1)} &:= -c_{i-1} c_i / a_{i-1} \\ a_i^{(1)} &:= -b_{i-1} c_i / a_{i-1} + a_i - b_i c_{i+1} / a_{i+1} \\ b_i^{(1)} &:= -b_i b_{i+1} / a_{i+1} \\ d_i^{(1)} &:= -d_{i-1} c_i / a_{i-1} + d_i - b_i d_{i+1} / a_{i+1} \end{aligned} \right\} \quad i = 2, 4, \dots, n. \quad (2.136)$$

Das System (2.133) hat nach diesen Operationen im Fall $n = 8$ die Gestalt

$$\begin{array}{ccccccccc|c} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & 1 \\ \hline a_1 & b_1 & & & & & & & d_1 \\ 0 & a_2^{(1)} & 0 & b_2^{(1)} & & & & & d_2^{(1)} \\ c_3 & a_3 & b_3 & & & & & & d_3 \\ c_4^{(1)} & 0 & a_4^{(1)} & 0 & b_4^{(1)} & & & & d_4^{(1)} \\ & & c_5 & a_5 & b_5 & & & & d_5 \\ & & c_6^{(1)} & 0 & a_6^{(1)} & 0 & b_6^{(1)} & & d_6^{(1)} \\ & & & c_7 & a_7 & b_7 & & & d_7 \\ & & c_8^{(1)} & 0 & a_8^{(1)} & & & & d_8^{(1)} \end{array} \quad (2.137)$$

Aus (2.137) erkennt man einerseits, dass die Gleichungen mit geraden Indizes ein lineares Gleichungssystem mit tridiagonaler Matrix für die geradzahlig indizierten Unbekannten darstellt, und dass sich andererseits die Unbekannten x_1, x_3, x_5, \dots in der angegebenen Reihenfolge bei bekannten Werten für x_2, x_4, x_6, \dots aus den anderen Gleichungen berechnen lassen. Das durch zyklische Reduktion resultierende Gleichungssystem der halben Ordnung m lautet im Fall $m = 4$

$$\begin{array}{ccccc} x_2 & x_4 & x_6 & x_8 & 1 \\ \hline a_2^{(1)} & b_2^{(1)} & & & d_2^{(1)} \\ c_4^{(1)} & a_4^{(1)} & b_4^{(1)} & & d_4^{(1)} \\ c_6^{(1)} & a_6^{(1)} & b_6^{(1)} & & d_6^{(1)} \\ c_8^{(1)} & a_8^{(1)} & & & d_8^{(1)} \end{array} \quad (2.138)$$

Die Formeln (2.136) des Reduktionsschrittes lassen sich mit Operationen an geeignet definierten Vektoren realisieren. Dazu führen wir die folgenden Teilvektoren der gegebenen Koeffizienten ein.

$$\begin{aligned} \mathbf{c}^{(u)} &:= \begin{pmatrix} c_1 \\ c_3 \\ c_5 \\ \vdots \\ c_{n+1} \end{pmatrix}, \quad \mathbf{a}^{(u)} := \begin{pmatrix} a_1 \\ a_3 \\ a_5 \\ \vdots \\ a_{n+1} \end{pmatrix}, \quad \mathbf{b}^{(u)} := \begin{pmatrix} b_1 \\ b_3 \\ b_5 \\ \vdots \\ b_{n+1} \end{pmatrix}, \quad \mathbf{d}^{(u)} := \begin{pmatrix} d_1 \\ d_3 \\ d_5 \\ \vdots \\ d_{n+1} \end{pmatrix} \in \mathbb{R}^{m+1} \\ \mathbf{c}^{(g)} &:= \begin{pmatrix} c_2 \\ c_4 \\ c_6 \\ \vdots \\ c_n \end{pmatrix}, \quad \mathbf{a}^{(g)} := \begin{pmatrix} a_2 \\ a_4 \\ a_6 \\ \vdots \\ a_n \end{pmatrix}, \quad \mathbf{b}^{(g)} := \begin{pmatrix} b_2 \\ b_4 \\ b_6 \\ \vdots \\ b_n \end{pmatrix}, \quad \mathbf{d}^{(g)} := \begin{pmatrix} d_2 \\ d_4 \\ d_6 \\ \vdots \\ d_n \end{pmatrix} \in \mathbb{R}^m. \end{aligned}$$

Daraus bilden wir die Hilfsvektoren unter der Vereinbarung, dass \otimes die komponentenweise Multiplikation zweier Vektoren und der Index +1 eine Verschiebung des Anfangsindexwertes

um 1 bedeuten:

$$\begin{aligned} \mathbf{r} &:= \begin{pmatrix} -1/a_1 \\ -1/a_3 \\ \vdots \\ -1/a_{n+1} \end{pmatrix} \in \mathbb{R}^{m+1}; \quad \mathbf{p} := \mathbf{c}^{(g)} \otimes \mathbf{r} = \begin{pmatrix} -c_2/a_1 \\ -c_4/a_3 \\ \vdots \\ -c_n/a_{n-1} \end{pmatrix} \\ \mathbf{q} &:= \mathbf{b}^{(g)} \otimes \mathbf{r}_{+1} = \begin{pmatrix} -b_2/a_3 \\ -b_4/a_5 \\ \vdots \\ -b_n/a_{n+1} \end{pmatrix} \in \mathbb{R}^m. \end{aligned}$$

Mit den Vektoren für das reduzierte System (2.138)

$$\mathbf{c}_1 := \begin{pmatrix} c_2^{(1)} \\ c_4^{(1)} \\ \vdots \\ c_m^{(1)} \end{pmatrix}, \quad \mathbf{a}_1 := \begin{pmatrix} a_2^{(1)} \\ a_4^{(1)} \\ \vdots \\ a_m^{(1)} \end{pmatrix}, \quad \mathbf{b}_1 := \begin{pmatrix} b_2^{(1)} \\ b_4^{(1)} \\ \vdots \\ b_m^{(1)} \end{pmatrix}, \quad \mathbf{d}_1 := \begin{pmatrix} d_2^{(1)} \\ d_4^{(1)} \\ \vdots \\ d_m^{(1)} \end{pmatrix} \in \mathbb{R}^m$$

lautet (2.136) in vektorisierter Form

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{c}^{(u)} \otimes \mathbf{p}; \\ \mathbf{a}_1 &= \mathbf{a}^{(g)} + \mathbf{b}^{(u)} \otimes \mathbf{p} + \mathbf{c}_{+1}^{(u)} \otimes \mathbf{q}; \\ \mathbf{b}_1 &= \mathbf{b}_{+1}^{(u)} \otimes \mathbf{q}; \\ \mathbf{d}_1 &= \mathbf{d}^{(g)} + \mathbf{d}^{(u)} \otimes \mathbf{p} + \mathbf{d}_{+1}^{(u)} \otimes \mathbf{q}. \end{aligned} \tag{2.139}$$

Dieser Reduktionsschritt erfordert somit m Divisionen, $8m$ Multiplikationen und $4m$ Additionen, insgesamt also $13m$ arithmetische Operationen mit Gleitpunktzahlen (flops).

Untersuchen wir an dieser Stelle noch das Rückwärtsrechnen der ungerade indizierten Unbekannten x_1, x_3, \dots, x_{n-1} aus den als bekannt vorausgesetzten Werten für x_2, x_4, \dots, x_n . Das zuständige Gleichungssystem lautet nach (2.137)

$$\begin{aligned} a_1 x_1 + b_1 x_2 &= d_1 \\ c_{2i+1} x_{2i} + a_{2i+1} x_{2i+1} + b_{2i+1} x_{2i+2} &= d_{2i+1}, \quad i = 1, 2, \dots, m-1. \end{aligned}$$

Daraus folgt

$$x_{2i+1} = (d_{2i+1} - c_{2i+1} x_{2i} - b_{2i+1} x_{2i+2})/a_{2i+1}, \quad i = 0, 1, 2, \dots, m-1, \tag{2.140}$$

wenn man $c_1 = 0$ beachtet und zusätzlich $x_0 = x_{n+1} = x_{n+2} := 0$ festsetzt. Um auch diese Vorschrift in vektorieller Form korrekt formulieren zu können, führen wir die beiden erweiterten Vektoren

$$\begin{aligned} \mathbf{x}^{(g)} &:= (x_0, x_2, x_4, \dots, x_n, x_{n+2})^T \in \mathbb{R}^{m+2}, \\ \mathbf{x}^{(u)} &:= (x_1, x_3, x_5, \dots, x_{n-1}, x_{n+1})^T \in \mathbb{R}^{m+1} \end{aligned}$$

ein. Dann lautet (2.140)

$$\mathbf{x}^{(u)} = (\mathbf{c}^{(u)} \otimes \mathbf{x}^{(g)} + \mathbf{b}^{(u)} \otimes \mathbf{x}_{+1}^{(g)} - \mathbf{d}^{(u)}) \otimes \mathbf{r}. \tag{2.141}$$

Hier weisen alle Vektoroperationen die gleiche Länge ($m+1$) auf, und es gehen entsprechende

Teilvektoren von $\mathbf{x}^{(g)}$ in die Rechnung ein. Der Rechenaufwand für diesen Schritt beläuft sich auf $3(m+1)$ Multiplikationen und $2(m+1)$ Additionen, zusammen also rund $5m$ flops.

Wir halten fest, dass der Reduktionsschritt und die Berechnung der eliminierten Unbekannten insgesamt einen Rechenaufwand von etwa $18m$ flops erfordert.

Ist m wieder eine gerade Zahl, so kann der Prozess der Reduktion auf das erste reduzierte tridiagonale Gleichungssystem analog weitergeführt werden. Ist im besonderen $n = 2^q$, $q \in \mathbb{N}^*$, so endet die zyklische Reduktion nach q Schritten mit einem Gleichungssystem mit der einzigen Unbekannten x_n , die jetzt trivialerweise berechnet werden kann. Jetzt setzt der Prozess der Rücksubstitution ein, welcher sukzessive die Unbekannten liefert. Da jeder weitere Reduktionsschritt und das Rückwärtsrechnen den halben Rechenaufwand im Vergleich zum vorangehenden erfordert, ergibt die Summation im Fall $n = 2^q$ einen Rechenaufwand von insgesamt rund $18n$ flops. Im Vergleich dazu benötigt der Gauß-Algorithmus (2.113), (2.114), (2.115) etwa $8n$ arithmetische Operationen, so dass der Aufwand der vektorisierbaren Methode der zyklischen Reduktion etwa 2.25mal größer ist.

Eine effiziente Implementierung des Algorithmus auf einem Vektorrechner erfordert eine sehr geschickte Datenstruktur derart, dass die Komponenten der zu verarbeitenden Vektoren entweder aufeinanderfolgend oder wenigstens nur mit einer Indexdifferenz zwei angeordnet sind. Zudem sind auch die sehr spezifischen Eigenschaften des Vektorrechners zu beachten und geeignet auszunutzen, um eine Verkürzung der Rechenzeit im Vergleich zur skalaren Ausführung zu erreichen. Auch kann es nötig sein, Modifikationen am Prozess vorzunehmen, um speziellen Gegebenheiten des Rechners gerecht zu werden [Reu 88]. Weiter ist zu beachten, dass die Längen der Vektoroperationen im Prozess der zyklischen Reduktion wie eine geometrische Folge rasch abnehmen. Es ist deshalb sinnvoll, den Prozess abzubrechen, sobald die Ordnung des letzten reduzierten tridiagonalen Gleichungssystems eine vom Rechner abhängige kritische Grenze unterschritten hat, für welche die Lösung des Systems mit dem Gauß-Algorithmus in skalarer Weise schneller ist als die weitere vektorielle Bearbeitung.

Tridiagonale Gleichungssysteme auf Parallelrechnern

Um tridiagonale lineare Gleichungssysteme auf einem *Parallelrechner* mit mehreren Recheneinheiten effizient zu lösen, sind andere Methoden entwickelt worden. Das Prinzip dieser *Divide-and-Conquer-Methoden* besteht darin, das gegebene Problem in einem ersten Schritt in Teilprobleme zu zerlegen, welche unabhängig voneinander, d.h. parallel auf den einzelnen Recheneinheiten, gelöst werden können, um schließlich die Teillösungen zur Gesamtlösung des gegebenen Problems zusammenzusetzen.

Eine erste Idee solcher Divide-and-Conquer-Methoden geht auf Van der Vorst [vdV 87a, vdV 87b] zurück unter der Annahme, dass der Parallelrechner zwei Recheneinheiten besitze. Anstelle der *LR*-Zerlegung (2.111) der tridiagonalen Matrix \mathbf{A} kann sie ebenso gut auf Grund einer *RQ*-Zerlegung in das Produkt von zwei Matrizen \mathbf{R} und \mathbf{Q} zerlegt werden, wobei je zwei Untermatrizen von \mathbf{R} und \mathbf{Q} *bidiagonal* sind.

Im Fall $n = 8$ lautet der Ansatz [Bon 91]

$$\mathbf{R} := \begin{pmatrix} 1 & & & & & & & \\ r_2 & 1 & & & & & & \\ & r_3 & 1 & & & & & \\ & & r_4 & 1 & r_5 & & & \\ & & & 1 & r_6 & & & \\ & & & & 1 & r_7 & & \\ & & & & & 1 & r_8 & \\ & & & & & & 1 & \end{pmatrix}, \quad \mathbf{Q} := \begin{pmatrix} q_1 & b_1 & & & & & & \\ & q_2 & b_2 & & & & & \\ & & q_3 & b_3 & & & & \\ & & & q_4 & & & & \\ & & & & c_5 & q_5 & & \\ & & & & & c_6 & q_6 & \\ & & & & & & c_7 & q_7 \\ & & & & & & & c_8 & q_8 \end{pmatrix}. \quad (2.142)$$

Aus der Bedingung $\mathbf{A} = \mathbf{R}\mathbf{Q}$ ergeben sich folgende wesentliche Bedingungsgleichungen für die unbekannten Nebendiagonalelemente von \mathbf{R} und für die Diagonalelemente von \mathbf{Q} :

$$\begin{aligned} q_1 &= a_1; \\ r_2 q_1 &= c_2, & r_2 b_1 + q_2 &= a_2; \\ r_3 q_2 &= c_3, & r_3 b_2 + q_3 &= a_3; \\ r_4 q_3 &= c_4, & r_4 b_3 + q_4 + r_5 c_5 &= a_4; \\ \hline r_5 q_5 &= b_4; \\ q_5 + r_6 c_6 &= a_5, & r_6 q_6 &= b_5; \\ q_6 + r_7 c_7 &= a_6, & r_7 q_7 &= b_6; \\ q_7 + r_8 c_8 &= a_7, & r_8 q_8 &= b_7; \\ q_8 &= a_8. \end{aligned} \quad (2.143)$$

Aus (2.143) lassen sich aus dem ersten Satz von Gleichungen sukzessive die Matrixelemente $q_1, r_2, q_2, r_3, q_3, r_4$ vollständig berechnen, während sich die Elemente $q_8, r_8, q_7, r_7, q_6, r_6, q_5, r_5$ in absteigender Reihenfolge bestimmen. Dann kann noch der letzte Wert q_4 berechnet werden. Die RQ -Zerlegung kann unter der vereinfachenden Annahme, dass $n = 2m, m \in \mathbb{N}^*$ gilt, wie folgt zusammengefasst werden:

$$\begin{aligned} q_1 &= a_1 \\ \text{für } i &= 2, 3, \dots, m : \\ r_i &= c_i / q_{i-1} \\ q_i &= a_i - r_i \times b_{i-1} \end{aligned}$$

(2.144)

$$\begin{aligned} q_n &= a_n \\ \text{für } i &= n-1, n-2, \dots, m+1 : \\ r_{i+1} &= b_i / q_{i+1} \\ q_i &= a_i - r_{i+1} \times c_{i+1} \\ r_{m+1} &= b_m / q_{m+1} \end{aligned}$$

(2.145)

$$q_m = q_m - c_{m+1} \times r_{m+1}$$

(2.146)

Die Rechenschritte (2.144) und (2.145) sind unabhängig voneinander auf zwei verschiedenen Recheneinheiten durchführbar, wobei beiden Prozessoren die gleiche Arbeitslast zugeteilt wird. Die Zerlegung wird durch (2.146) vervollständigt, wobei das Teilresultat q_m aus (2.144) und der Wert r_{m+1} aus (2.145) verwendet werden.

Für ungerades $n = 2m + 1, m \in \mathbb{N}^*$, kann die RQ -Zerlegung analog mit Untermatrizen der Ordnungen m und $(m+1)$ in der Diagonale von \mathbf{R} und \mathbf{Q} durchgeführt werden. In [Bon 91] wird die Existenz der RQ -Zerlegung für diagonal dominante tridiagonale Matrizen gezeigt. Die RQ -Zerlegung existiert auch für symmetrische, positiv definite Matrizen oder für so genannte M -Matrizen [vdV 87a, vdV 87b].

Der Rechenaufwand der RQ -Zerlegung setzt sich aus je $(n-1)$ Divisionen, Multiplikationen und Additionen zusammen und entspricht somit demjenigen einer LR -Zerlegung.

Auf Grund der RQ -Zerlegung der tridiagonalen Matrix \mathbf{A} geht das gegebene Gleichungssystem $\mathbf{Ax} = \mathbf{d}$ über in $\mathbf{RQx} = \mathbf{d}$. Die Lösung \mathbf{x} berechnet sich aus den beiden sukzessive zu lösenden Systemen

$$\mathbf{Ry} = \mathbf{d}, \quad \mathbf{Qx} = \mathbf{y}. \quad (2.147)$$

Wegen der besonderen Struktur von \mathbf{R} ist eine Vorwärts- und Rücksubstitution für je einen Teil der Komponenten von \mathbf{y} auszuführen.

$y_1 = d_1$ für $i = 2, 3, \dots, m-1 :$ $y_i = d_i - r_i \times y_{i-1}$	$y_n = d_n$ für $i = n-1, n-2, \dots, m+1 :$ $y_i = d_i - r_{i+1} \times y_{i+1}$
---	---

(2.148)

Aus der m -ten Gleichung

$$r_my_{m-1} + y_m + r_{m+1}y_{m+1} = d_m$$

ergibt sich mit den aus den beiden Teilprozessen bekannten Werten y_{m-1} und y_{m+1} noch

$y_m = d_m - r_m \times y_{m-1} - r_{m-1} \times y_{m+1}.$	(2.149)
--	--

Die Rücksubstitution des Gauß-Algorithmus wird ersetzt durch die Rück- und Vorwärtssubstitution für je einen Teil der Unbekannten x_i , sobald die Unbekannte x_m als gemeinsamer Startwert für diese beiden Prozesse als Lösung der trivialen m -ten Gleichung von $\mathbf{Qx} = \mathbf{y}$ vorliegt.

$x_m = y_m / q_m$	(2.150)
-------------------	--

für $i = m-1, m-2, \dots, 1 :$ $x_i = (y_i - b_i \times x_{i+1}) / q_i$	für $i = m+1, m+2, \dots, n :$ $x_i = (y_i - c_i \times x_{i-1}) / q_i$
--	--

(2.151)

Die Lösung der beiden Systeme (2.147) geschieht wiederum durch je zwei weitgehend unabhängige Prozesse, wobei der Rechenaufwand mit n Divisionen und je $(2n-2)$ Multiplikationen und Additionen gleich demjenigen der Vorwärts- und Rücksubstitution des Gauß-Algorithmus ist. Die Parallelisierung ist also mit keiner Erhöhung der Zahl der arithmetischen Operationen verbunden. Zudem bestätigen Experimente auf Parallelrechnern mit zwei

Prozessoren, dass diese in der Tat optimal eingesetzt werden können und sich die Rechenzeit halbiert [Bon 91]. Die numerische Stabilität des Verfahrens der *RQ*-Zerlegung ist für eine Klasse von tridiagonalen Gleichungssystemen gleich derjenigen des Gauß-Algorithmus [vdV 87a, vdV 87b].

Die bestechend einfache Idee der Zerlegung lässt sich leider nicht in dem Sinn verallgemeinern, dass eine feinere Aufteilung in mehr Teilaufgaben und damit eine höhere Parallelisierung erzielt werden kann. Falls überdies die Prozessoren des Parallelrechners auch Vektorarithmetik anbieten, so kann davon kein direkter Gebrauch gemacht werden, weil die einzelnen Teilprozesse rekursiven Charakter haben und in dieser Form nicht vektorisierbar sind.

Um die Auflösung von großen tridiagonalen Gleichungssystemen auf mehr als zwei Recheinheiten verteilen zu können, sind verschiedene Methoden entwickelt worden. Wir behandeln die *Odd-Even-Eliminationsmethode* (Odd-Even Reduction), welche dem Divide-and-Conquer-Konzept entspricht. Die Anzahl $n = 2m, m \in \mathbb{N}^*$, der Unbekannten sei gerade. In einem vorbereitenden Schritt werden im gegebenen linearen Gleichungssystem (2.133) die Gleichungen und die Unbekannten so vertauscht, dass zuerst jene mit ungeraden Indizes und dann jene mit geraden Indizes aufgeschrieben werden. Im Fall $n = 8$ erhält das Gleichungssystem die folgende Gestalt:

$$\begin{array}{ccccccccc|c} x_1 & x_3 & x_5 & x_7 & x_2 & x_4 & x_6 & x_8 & 1 \\ \hline a_1 & & & b_1 & & & & & d_1 \\ & a_3 & & c_3 & b_3 & & & & d_3 \\ & & a_5 & & c_5 & b_5 & & & d_5 \\ & & & a_7 & & c_7 & b_7 & & d_7 \\ \hline c_2 & b_2 & & a_2 & & & & & d_2 \\ & c_4 & b_4 & & a_4 & & & & d_4 \\ & & c_6 & b_6 & & a_6 & & & d_6 \\ & & & c_8 & & & a_8 & & d_8 \end{array} \quad (2.152)$$

Mit der zugehörigen Permutationsmatrix $\mathbf{P} \in \mathbb{R}^{n,n}$ geht das gegebene Gleichungssystem $\mathbf{Ax} = \mathbf{d}$ über in $\mathbf{PAP}^T(\mathbf{Px}) = \mathbf{Pd}$, wobei die zeilen- und spaltenpermutierte Matrix \mathbf{PAP}^T die spezielle Blockstruktur

$$\mathbf{PAP}^T = \left(\begin{array}{cc} \mathbf{A}_1 & \mathbf{B} \\ \mathbf{C} & \mathbf{A}_2 \end{array} \right) \quad (2.153)$$

aufweist, in der $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{m,m}$ je Diagonalmatrizen, \mathbf{B} eine untere bidiagonale und \mathbf{C} eine

obere bidiagonale ($m \times m$)-Matrix sind. Bezeichnen wir weiter mit

$$\begin{aligned} \mathbf{y}_1 &:= \begin{pmatrix} x_1 \\ x_3 \\ \vdots \\ x_{n-1} \end{pmatrix}, \quad \mathbf{y}_2 := \begin{pmatrix} x_2 \\ x_4 \\ \vdots \\ x_n \end{pmatrix}, \\ \mathbf{v}_1 &:= \begin{pmatrix} d_1 \\ d_3 \\ \vdots \\ d_{n-1} \end{pmatrix}, \quad \mathbf{v}_2 := \begin{pmatrix} d_2 \\ d_4 \\ \vdots \\ d_n \end{pmatrix} \in \mathbb{R}^m \end{aligned} \quad (2.154)$$

die Teilvektoren, so gelten die Beziehungen

$$\mathbf{A}_1 \mathbf{y}_1 + \mathbf{B} \mathbf{y}_2 = \mathbf{v}_1, \quad (2.155)$$

$$\mathbf{C} \mathbf{y}_1 + \mathbf{A}_2 \mathbf{y}_2 = \mathbf{v}_2. \quad (2.156)$$

Für diagonal dominante oder symmetrische, positiv definite Matrizen \mathbf{A} sind \mathbf{A}_1 und \mathbf{A}_2 regulär, so dass aus (2.155) und (2.156) folgen

$$\mathbf{y}_1 = \mathbf{A}_1^{-1} \mathbf{v}_1 - \mathbf{A}_1^{-1} \mathbf{B} \mathbf{y}_2, \quad \mathbf{y}_2 = \mathbf{A}_2^{-1} \mathbf{v}_2 - \mathbf{A}_2^{-1} \mathbf{C} \mathbf{y}_1. \quad (2.157)$$

Wir setzen \mathbf{y}_2 aus (2.157) in (2.155) und \mathbf{y}_1 aus (2.157) in (2.156) ein und erhalten

$$(\mathbf{A}_1 - \mathbf{B} \mathbf{A}_2^{-1} \mathbf{C}) \mathbf{y}_1 = (\mathbf{v}_1 - \mathbf{B} \mathbf{A}_2^{-1} \mathbf{v}_2), \quad (2.158)$$

$$(\mathbf{A}_2 - \mathbf{C} \mathbf{A}_1^{-1} \mathbf{B}) \mathbf{y}_2 = (\mathbf{v}_2 - \mathbf{C} \mathbf{A}_1^{-1} \mathbf{v}_1). \quad (2.159)$$

(2.158) ist ein Gleichungssystem für die m Unbekannten x_1, x_3, \dots, x_{n-1} , und (2.159) eines für die m Unbekannten x_2, x_4, \dots, x_n . Die Matrizen

$$\mathbf{A}_1^{(1)} := \mathbf{A}_1 - \mathbf{B} \mathbf{A}_2^{-1} \mathbf{C}, \quad \mathbf{A}_2^{(1)} := \mathbf{A}_2 - \mathbf{C} \mathbf{A}_1^{-1} \mathbf{B} \quad (2.160)$$

sind wieder tridiagonal, weil das Produkt einer unteren und einer oberen Bidiagonalmatrix eine Tridiagonalmatrix ergibt. Diese beiden tridiagonalen Gleichungssysteme in je $n/2$ Unbekannten können unabhängig voneinander in paralleler Weise auf zwei Prozessoren gelöst werden, womit die Zielsetzung der Divide-and-Conquer-Methode erreicht ist. Mit

$$\begin{aligned} \mathbf{A}_1^{(1)} &:= \begin{pmatrix} a_1^{(1)} & b_1^{(1)} & & & & \\ c_3^{(1)} & a_3^{(1)} & b_3^{(1)} & & & \\ & c_5^{(1)} & a_5^{(1)} & b_5^{(1)} & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-1}^{(1)} & a_{n-1}^{(1)} & \end{pmatrix}, \\ \mathbf{d}_1^{(1)} &:= \mathbf{v}_1 - \mathbf{B} \mathbf{A}_2^{-1} \mathbf{v}_2 = \begin{pmatrix} d_1^{(1)} \\ d_3^{(1)} \\ d_5^{(1)} \\ \vdots \\ d_{n-1}^{(1)} \end{pmatrix} \end{aligned}$$

und analog für $\mathbf{A}_2^{(1)}$ und $\mathbf{d}_2^{(1)}$ mit den geradzahlig indizierten Koeffizienten lauten die beiden

tridiagonalen Gleichungssysteme

$$\mathbf{A}_1^{(1)} \mathbf{y}_1 = \mathbf{d}_1^{(1)}, \quad \mathbf{A}_2^{(1)} \mathbf{y}_2 = \mathbf{d}_2^{(1)}. \quad (2.161)$$

Die Matrix- und Vektorelemente von $\mathbf{A}_1^{(1)}$ und $\mathbf{d}_1^{(1)}$ sind gegeben durch

$$\left. \begin{array}{l} a_i^{(1)} = a_i - b_{i-1}c_i/a_{i-1} - b_ic_{i+1}/a_{i+1} \\ b_i^{(1)} = -b_ib_{i+1}/a_{i+1}, \quad c_i^{(1)} = -c_{i-1}c_i/a_{i-1} \\ d_i^{(1)} = d_i - d_{i-1}c_i/a_{i-1} - b_id_{i+1}/a_{i+1} \end{array} \right\} \quad i = 1, 3, \dots, n-1,$$

falls man die zusätzlichen Werte $a_0 = 1, b_0 = b_n = c_0 = c_1 = d_0 = 0$ festlegt. Diese Formeln sind identisch mit jenen von (2.136), welche für das zweite System von (2.161) gültig bleiben, falls noch die weiteren Werte $a_{n+1} = 1, b_{n+1} = c_{n+1} = d_{n+1} = 0$ vereinbart werden. Wie im Abschnitt 2.4.2 beschrieben ist, kann die Berechnung der Koeffizienten der Systeme (2.161) vektorisiert und für die ungeraden und geraden Indexwerte unabhängig voneinander parallel auf zwei Prozessoren ausgeführt werden.

Die beiden Gleichungssysteme (2.161) können aber auf die gleiche Weise in je zwei Teilprobleme aufgeteilt werden, und diese Aufteilung kann so oft wiederholt werden, bis entweder ein bestimmter Grad der Parallelisierung erreicht ist oder die Teilprobleme keine weitere Aufspaltung mehr zulassen. Da bei diesem Vorgehen die Zahl der Teilprobleme sukzessive verdoppelt wird, spricht man vom *Prinzip der rekursiven Verdoppelung*. Das Prinzip ist in Abb. 2.4 anhand der Baumstruktur veranschaulicht.

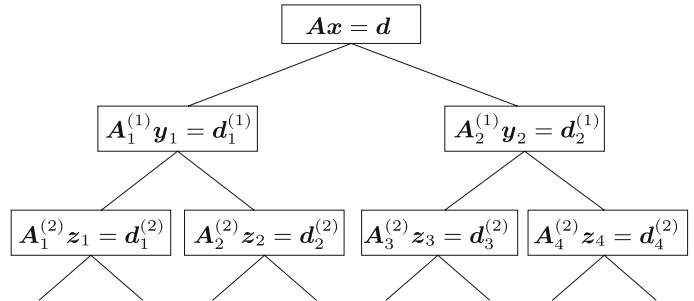


Abb. 2.4 Prinzip der rekursiven Verdoppelung.

Sind nach genügend vielen Verdoppelungsschritten die Lösungen der vielen kleinen tridiagonalen Teilgleichungssysteme berechnet, dann sind die Teillösungen nur noch durch eine einfache Umordnung zur Gesamtlösung zusammenzusetzen.

Ist im speziellen $n = 2^p, p \in \mathbb{N}^*$, so kann die rekursive Verdoppelung p mal fortgesetzt werden, so dass am Schluss nur noch je eine Gleichung für eine Unbekannte übrig bleibt, aus der die betreffende Unbekannte resultiert. Der Rechenaufwand an arithmetischen Operationen beträgt $(4np - 4n + 4)$ Additionen, $(6np - 8n + 8)$ Multiplikationen und $(2np - n + 2)$ Divisionen [Bon 91]. Für den Odd-Even-Eliminationsalgorithmus ist in [Bon 91] ein MATLAB-Programm angegeben und es werden auch Varianten der Methode diskutiert.

Zur Parallelisierung der Lösung von tridiagonalen Gleichungssystemen sind zahlreiche andere Methoden und davon abgeleitete Varianten entwickelt worden. Zu erwähnen wäre das *Verfahren von Stone* [Bon 91, Hoc 88, Sto 73, Sto 75], welches darauf beruht, die rekursiven Formeln des Gauß-Algorithmus in lineare Differenzengleichungen zweiter Ordnung zu transformieren, die sodann mit der Methode der zyklischen Reduktion gelöst werden. Soll die Behandlung des Problems auf eine bestimmte Zahl p von Prozessoren aufgeteilt werden, so eignen sich *Partitionsverfahren* gut. Nach dem Vorschlag von *Wang* [Wan 81] wird die tridiagonale Matrix \mathbf{A} längs der Diagonale in p etwa gleich große Blöcke unterteilt. Zuerst werden in jedem Diagonalblock die Elemente der unteren Nebendiagonale eliminiert und anschließend diejenigen der oberen Nebendiagonale. In zwei weiteren Schritten wird die Matrix auf obere Dreiecksgestalt und schließlich auf Diagonalf orm gebracht, so dass jetzt die Lösung berechnet werden kann. Alle Operationen sind parallel ausführbar. Für Details vergleiche man etwa [Bon 91, Fro 90, Ort 88]. Das Verfahren hat in den letzten Jahren verschiedene Erweiterungen erfahren zur GECR-Methode [Joh 87] der Wraparound-Partitionierungsmethode [Heg 91] und Verfeinerungen der Divide-and-Conquer-Methoden [Bon 91].

2.5 Anwendungen

Schaltkreistheorie

Auf Grund der Gesetze von Kirchhoff und Ohm ergeben sich für ein Netzwerk von elektrischen Verbindungen mit Spannungsquellen und Widerständen verschiedene Beziehungen, die man in ein symmetrisches, positiv definites Gleichungssystem mit den Potenzialen als Unbekannten umformen kann. Ein Zweig Z_j in einem solchen Netzwerk besteht aus einer Stromquelle U_j und einem Widerstand R_j . Für einen Widerstand R_j ohne Stromquelle setzt man $U_j = 0V$. Aus den berechneten Potenzialen kann man dann auch die Stromstärken und Spannungen in allen Zweigen des Netzes berechnen. Die technischen Einzelheiten findet man z.B. in [Mag 85]. Wir betrachten als Beispiel das Netzwerk der Abb. 2.5.

Gegeben seien die Spannungen (unter Berücksichtigung der Polung)

$$U_1 = 10V, \quad U_2 = -15V, \quad U_3 = 20V, \quad U_4 = U_5 = U_6 = 0V$$

und die elektrischen Widerstände

$$R_1 = R_2 = \dots = R_6 = 10\Omega.$$

Jetzt stellt man die Inzidenzmatrix $\mathbf{T} = (t_{ij}) \in \mathbb{R}^{4,6}$ auf, in der für jeden Knoten K_i der Beginn eines Zweiges (Z_j) mit -1 und das Ende eines Zweiges mit $+1$ gekennzeichnet wird, also für unser Beispiel:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \end{pmatrix}.$$

Dann wird die Inverse der Diagonalmatrix der Widerstände aufgestellt:

$$\mathbf{R}^{-1} := \text{diag}(R_1^{-1}, \dots, R_6^{-1}) = \text{diag}(0.1, \dots, 0.1), \quad \mathbf{R}^{-1} \in \mathbb{R}^{6,6}.$$

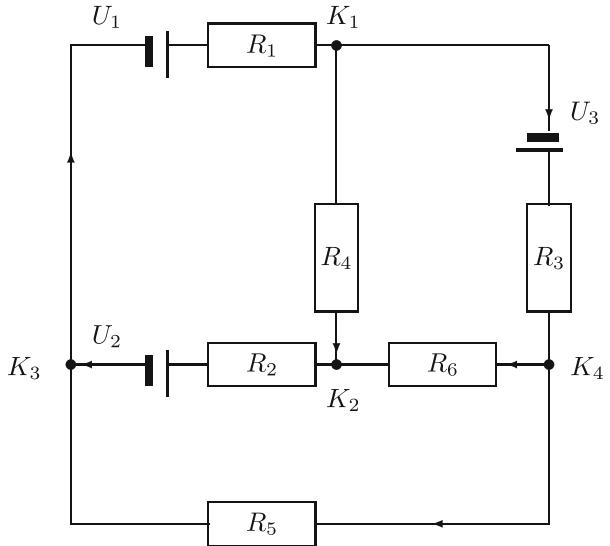


Abb. 2.5 Stromkreis mit vier Knoten und sechs Zweigen.

Wenn man mit den Gesetzen von Kirchhoff und Ohm die Ströme berechnet, die im Stromkreis fließen, erhält man ein lineares Gleichungssystem mit den Potenzialen P_j in den Knoten K_j als Unbekannten. Die Matrix des gesuchten Gleichungssystems ergibt sich aus

$$\mathbf{B} = \mathbf{T} \cdot \mathbf{R}^{-1} \cdot \mathbf{T}^T,$$

indem man für den Bezugspunkt, hier Knoten K_4 , das Potenzial null setzt und die entsprechende Zeile und Spalte der Matrix \mathbf{B} streicht. Das ergibt

$$\mathbf{A} = \begin{pmatrix} 0.3 & -0.1 & -0.1 \\ -0.1 & 0.3 & -0.1 \\ -0.1 & -0.1 & 0.3 \end{pmatrix}.$$

Mit $\mathbf{U} := (U_1, \dots, U_6)^T$ ergibt sich die rechte Seite als $\mathbf{b} = -\mathbf{T}\mathbf{R}^{-1}\mathbf{U}$ mit anschließender Streichung der letzten Komponente:

$$\mathbf{b} = (1.0, -1.5, 2.5)^T.$$

Entsprechend kann man bei komplizierteren Netzen vorgehen. Es ist auch leicht möglich, ein Programm zur Konstruktion solcher Netze (z.B. im Dialog) mit Lösung der entstehenden Gleichungssysteme zu schreiben. Die Matrix muss dabei symmetrisch und positiv definit werden, sonst hat man etwas falsch gemacht, z.B. einen Widerstand null gesetzt.

Die Lösung des Gleichungssystems sind die Potenziale $P_1 = 7.5V$, $P_2 = 1.25V$, $P_3 = 11.25V$; das Bezugspotenzial ist $P_4 = 0V$. Verschiedene Spannungen U_j für dasselbe Netz ergeben ein Gleichungssystem mit mehreren rechten Seiten.

Fluss durch ein Rohrnetz mit Pumpe

Der Fluss durch ein Pumpen-Netzwerk (siehe [Hil 88]) gehorcht unter gewissen Voraussetzungen den Gesetzen für laminare Strömung:

1. An Verzweigungen ist das Volumen der einströmenden Flüssigkeit gleich dem der ausströmenden Flüssigkeit.
2. Der Druckverlust Δp entlang eines Rohres der Länge l ist gegeben als

$$\Delta p = \frac{8\eta l}{\pi r^4} q.$$

Dabei sind r der Radius des Rohres, η die Viskosität der Flüssigkeit und q die unbekannte Durchflussleistung.

3. Der Druckverlust in geschlossenen Schleifen ist null.
4. Entgegen der Fließrichtung erhöht sich der Druck, also ist der entsprechende Druckverlust negativ.

In Abb. 2.6 betrachten wir ein einfaches Rohrnetz mit drei Verzweigungen **A**, **B** und **C** und acht Rohren **1** bis **8**.

Seien r_i , l_i , q_i und k_i , $i = 1, 2, \dots, 8$, Radius, Länge, Durchflussleistung und effektiver Widerstand im i -ten Rohr, wobei

$$k_i := \frac{8\eta l_i}{\pi r_i^4}.$$

Die Zisterne in Abb. 2.6 ist ein Flüssigkeitsreservoir, welches das Pumpennetzwerk mit neuer Flüssigkeit versorgt; die Zisterne ist ausreichend gefüllt. Dann kann man das Pumpennetzwerk als einen geschlossenen Flüssigkeitskreislauf betrachten. Dabei wird neben der Pumpe **P** auch die Zisterne **Z** zu einem Verzweigungsknoten. Außerdem muß man sich die Abflüsse des Pumpennetzwerks mit der Zisterne verbunden denken. Unter diesen Voraussetzungen gilt $q_1 = q_8$.

Es entstehen die inneren Schleifen **PAZP**, **AZBA**, **BZCB** und **CZC**. In der Schleife **PAZP** baut die Pumpe einen Druck P auf. Weitere Schleifen liefern zusätzliche, aber redundante Gleichungen, die nicht berücksichtigt werden müssen. Die Anwendung der Gesetze für laminare Strömung unter Beachtung von $q_1 = q_8$ ermöglicht die Aufstellung eines Systems von sieben linearen Gleichungen für die Unbekannten q_i :

$$\begin{aligned}
 q_1 - q_2 - q_3 &= 0 \\
 q_3 - q_4 - q_5 &= 0 \\
 q_4 - q_6 - q_7 &= 0 \\
 (k_1 + k_8)q_1 + k_2q_2 &= P \\
 k_2q_2 - k_3q_3 - k_5q_5 &= 0 \\
 -k_4q_4 + k_5q_5 - k_7q_7 &= 0 \\
 k_6q_6 - k_7q_7 &= 0
 \end{aligned}$$

Dabei ist P die Druckerhöhung durch die Pumpe **P**. Für die Zahlenwerte $P = 350\,000$, $\eta = 0.015$, $r_i \equiv r = 0.06$ und $l_i = (2, 21, 3, 4, 6, 10, 6, 34)$ lässt sich das System mit dem Gauß-Algorithmus lösen. Die auf vier wesentliche Stellen gerundeten Ergebnisse lauten: $q_1 = q_8 = 2.904$, $q_2 = 0.6768$, $q_3 = 2.227$, $q_4 = 0.9718$, $q_5 = 1.255$, $q_6 = 0.3644$, $q_7 = 0.6074$.

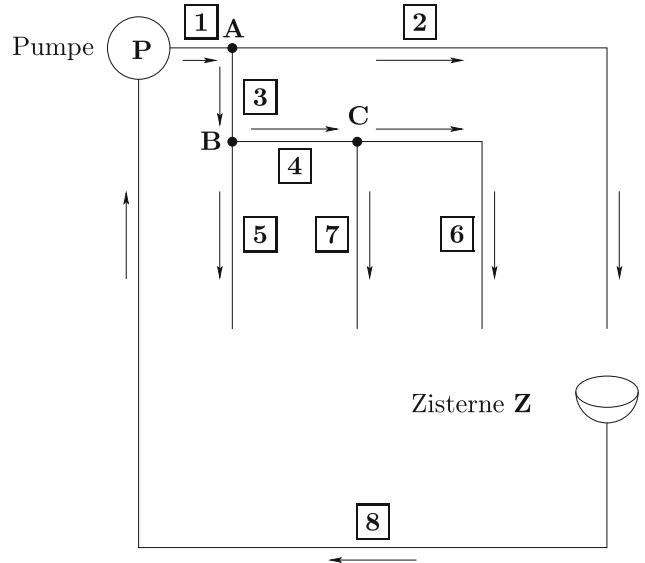


Abb. 2.6 Fluß durch ein Pumpennetzwerk.

Ebenes Fachwerk

Die Berechnung der Kräfte in einem ebenen Fachwerk führt unter gewissen Voraussetzungen auf ein System von linearen Gleichungen in Bandform, da für jeden Knoten im Fachwerk nur Beziehungen zu den Nachbarknoten berücksichtigt werden. Die Bandbreite hängt von der Nummerierung der Knoten ab. Wir wollen ein Beispiel mit sechs Knoten durchrechnen, das in Abb. 2.7 dargestellt ist.

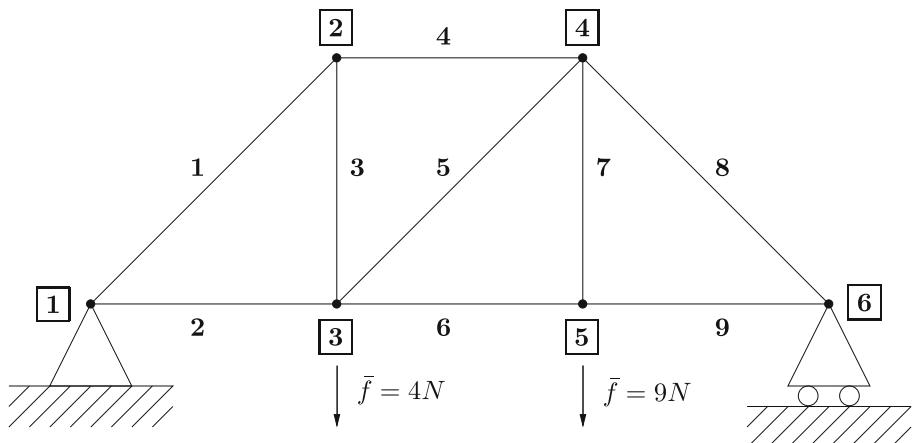


Abb. 2.7 Ein ebenes Fachwerk mit sechs Knoten.

Wir nehmen an, dass das Fachwerk reibungsfrei verbunden ist. Ein Satz der Mechanik besagt, dass das Fachwerk statisch bestimmt ist, wenn die Anzahl der Verbindungen m und die Anzahl der Knoten j die Gleichung $2j - 3 = m$ erfüllen; das ist hier der Fall. Die Kräfte sind dann über statische Gleichgewichtsbedingungen determiniert. Ihre horizontalen und vertikalen Komponenten seien F_x bzw. F_y . Um die Kräfte über schiefe Verbindungen in ihre horizontalen und vertikalen Komponenten zerlegen zu können, benötigt man $\cos \varphi$ und $\sin \varphi$, wobei der Winkel φ die Neigung der schießen Komponente angibt. Da in diesem Beispiel $\varphi = \pi/4$ gilt, verwenden wir folgende Abkürzung: $t := \sin \pi/4 = \cos \pi/4 = 1/\sqrt{2}$. Wir stellen nun die Gleichgewichtsgleichungen für die nicht abgestützten Knoten 2 bis 5 und den vertikal abgestützten Knoten 6 auf. Da der Knoten 1 vollständig abgestützt ist, findet dort kein Kräfteausgleich statt. Die Summe aller Kräfte in einem nicht abgestützten Knoten muß sowohl in horizontaler als auch in vertikaler Richtung gleich null sein, da sich das Fachwerk in einem stabilen Gleichgewichtszustand befindet. Mit den Kräften f_i , die jeweils entlang der i -ten Verbindung wirken, erhält man folgende Gleichungen (beachte: f_7 kann direkt berechnet werden):

$$\text{Knoten 2: } \begin{cases} -tf_1 + f_4 = 0 \\ -tf_1 - f_3 = 0 \end{cases}$$

$$\text{Knoten 3: } \begin{cases} f_2 - tf_5 - f_6 = 0 \\ f_3 + tf_5 - 4 = 0 \end{cases}$$

$$\text{Knoten 4: } \begin{cases} -f_4 - tf_5 + tf_8 = 0 \\ -tf_5 - f_7 - tf_8 = 0 \end{cases}$$

$$\text{Knoten 5: } \begin{cases} -f_6 + f_9 = 0 \\ f_7 - 9 = 0 \end{cases}$$

$$\text{Knoten 6: } \begin{cases} -tf_8 - f_9 = 0 \end{cases}$$

Wenn die Kraft f_7 , die direkt ausgerechnet werden kann, weggelassen wird, lassen sich die Gleichgewichtskräfte als Lösungen des linearen Gleichungssystems $\mathbf{A}\tilde{\mathbf{f}} = \mathbf{b}$ darstellen mit

$$\mathbf{A} = \begin{pmatrix} -t & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -t & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -t & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & t & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -t & 0 & t & 0 \\ 0 & 0 & 0 & 0 & -t & 0 & -t & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -t & -1 \end{pmatrix}$$

$$\tilde{\mathbf{f}} = (f_1, f_2, f_3, f_4, f_5, f_6, f_8, f_9)^T$$

$$\mathbf{b} = (0, 0, 0, 4, 0, 9, 0, 0)^T.$$

Die Matrix \mathbf{A} ist schwach besetzt und hat die Bandbreite $m = 3$. Die Lösungen des Gleichgewichtsproblems sind – wieder auf vier wesentliche Stellen gerundet – die Kräfte

$$\mathbf{f} = (-8.014, 5.667, 5.667, -5.667, -2.357, 7.333, 9, -10.37, 7.333)^T.$$

Dabei ist \mathbf{f} um die direkt berechnete Kraft f_7 ergänzt worden.

2.6 Software

Historische Quelle aller Software zur numerischen linearen Algebra ist das Handbuch von Wilkinson und Reinsch, [Wil 86], das bezüglich übersichtlicher Programmierung und umfassender Dokumentation Maßstäbe setzt. Es ist insofern ein frühes (erstes) Template, siehe Abschnitt 5.9. Die Algorithmen dieses Bandes, seine ALGOL-Programme und seine Vorgehensweise bei Erstellung und Dokumentation eines Programmpaketes sind Grundlage der FORTRAN-Bibliotheken LINPACK [Don 79] zur Lösung von linearen Gleichungssystemen und EISPACK [Smi 88] zur Lösung von Matrix-Eigenwertproblemen, die in Kapitel 5 behandelt werden. Diese Bibliotheken haben ihrerseits Eingang in die wichtigen großen Numerik-Bibliotheken NAG [NAGb, NAGa] und IMSL [IMSL] sowie in MATLAB [Mol] gefunden. Weiterentwicklungen von LINPACK und EISPACK, die sich auch besonders für Vektor- und Parallelrechner eignen, finden sich in dem viel benutzten Paket LAPACK [And 99] (*A Portable Linear Algebra Library for High-Performance Computers*). Dazu wurden zunächst die Grundoperationen der linearen Algebra vom Skalarprodukt über ebene Rotation bis zum Matrixprodukt im Unterprogramm-Paket BLAS (*Basic Linear Algebra Subprograms*) zusammengefasst. BLAS besteht aus drei Stufen, den Vektor-Vektor-, Matrix-Vektor- und den Matrix-Matrix-Operationen. BLAS wurde weiterentwickelt zu PBLAS (*A Set of Parallel BLAS*), PB-BLAS (*A Set of Parallel Block BLAS*) und ergänzt um BLACS (*Basic Linear Algebra Communication Subprograms*), das für die Kommunikation auf Parallelrechnern oder in verteilten Systemen verantwortlich ist. Zu demselben Zweck wurde LAPACK weiterentwickelt zu SCALAPACK (*A Portable Linear Algebra Library for Distributed Memory Computers*). Statt zu all diesen Entwicklungen und den dazu entwickelten Algorithmen einzelne Literaturhinweise zu geben, verweisen wir auf die instruktiven und ständig ergänzten LAPACK Working Notes (<http://www.netlib.org/lapack/lawns/>).

Unsere Problemlöseumgebung PAN (<http://www.upb.de/SchwarzKoeckler/>) verfügt über drei Programme zur Lösung linearer Gleichungssysteme mit regulärer Matrix, symmetrisch positiv-definiter Matrix und mit einer Bandmatrix als Koeffizientenmatrix.

2.7 Aufgaben

Aufgabe 2.1. Man löse mit dem Gauß-Algorithmus

$$\begin{aligned} 2x_1 - 4x_2 + 6x_3 - 2x_4 &= 3 \\ 3x_1 - 6x_2 + 10x_3 - 4x_4 &= -2 \\ x_1 + 3x_2 + 13x_3 - 6x_4 &= 3 \\ 5x_2 + 11x_3 - 6x_4 &= -5. \end{aligned}$$

Wie lautet die *LR*-Zerlegung der Systemmatrix?

Aufgabe 2.2. Das lineare Gleichungssystem

$$\begin{aligned} 6.22x_1 + 1.42x_2 - 1.72x_3 + 1.91x_4 &= 7.53 \\ 1.44x_1 + 5.33x_2 + 1.11x_3 - 1.82x_4 &= 6.06 \\ 1.59x_1 - 1.23x_2 - 5.24x_3 - 1.42x_4 &= -8.05 \\ 1.75x_1 - 1.69x_2 + 1.57x_3 + 6.55x_4 &= 8.10 \end{aligned}$$

ist unter Ausnutzung der diagonalen Dominanz mit dem Gauß-Algorithmus bei fünfstelliger Rechnung zu lösen.

Aufgabe 2.3. Man berechne den Wert der Determinante

$$\left| \begin{array}{ccc} 0.596 & 0.497 & 0.263 \\ 4.07 & 3.21 & 1.39 \\ 0.297 & 0.402 & 0.516 \end{array} \right|.$$

a) Zunächst benutze man dazu die Definitionsgleichung (Regel von Sarrus), einmal mit voller und dann mit dreistelliger Rechengenauigkeit. Im zweiten Fall hängt das numerische Ergebnis von der Reihenfolge der Operationen ab. Um das unterschiedliche Resultat erklären zu können, sind alle Zwischenergebnisse zu vergleichen.

b) Jetzt rechne man mit dem Gauß-Algorithmus bei dreistelliger Rechnung unter Verwendung der Diagonalstrategie und der relativen Spaltenmaximumstrategie.

Aufgabe 2.4. Das lineare Gleichungssystem

$$\begin{aligned} 10x_1 + 14x_2 + 11x_3 &= 1 \\ 13x_1 - 66x_2 + 14x_3 &= 1 \\ 11x_1 - 13x_2 + 12x_3 &= 1 \end{aligned}$$

ist mit dem Gauß-Algorithmus und relativer Spaltenmaximumstrategie bei fünfstelliger Rechnung zu lösen. Wie lauten die Permutationsmatrix P und die Matrizen L und R der Zerlegung? Mit den mit höherer Genauigkeit berechneten Residuen führe man einen Schritt der Nachiteration durch. Wie groß sind die Konditionszahlen der Systemmatrix A für die Gesamtnorm, die Zeilensummennorm und die Frobenius-Norm

Aufgabe 2.5. Man zeige die Submultiplikativität $\|AB\|_F \leq \|A\|_F \|B\|_F$ der Frobenius-Norm.

Aufgabe 2.6. Man verifizierte, dass sowohl die Frobenius-Norm als auch die Gesamtnorm mit der euklidischen Vektornorm kompatibel sind.

Aufgabe 2.7. Welches ist die der L_1 -Vektornorm (2.60) zugeordnete Matrixnorm?

Aufgabe 2.8. Man zeige, dass für die Konditionszahlen folgende Beziehungen gelten:

- a) $\kappa(\mathbf{AB}) \leq \kappa(\mathbf{A})\kappa(\mathbf{B})$ für alle Matrixnormen;
- b) $\kappa(c\mathbf{A}) = \kappa(\mathbf{A})$ für alle $c \in \mathbb{R}$;
- c) $\kappa_2(\mathbf{Q}) = 1$ für eine orthogonale Matrix \mathbf{Q} ;
- d) $\kappa_2(\mathbf{A}) \leq \kappa_F(\mathbf{A}) \leq \kappa_G(\mathbf{A}) \leq n^2 \kappa_\infty(\mathbf{A})$;
- e) $\kappa_2(\mathbf{QA}) = \kappa_2(\mathbf{A})$, falls \mathbf{Q} eine orthogonale Matrix ist.

Aufgabe 2.9. Das folgende Gleichungssystem mit symmetrischer und positiv definiter Matrix \mathbf{A} soll mit dem Gauß-Algorithmus (Diagonalstrategie und Spaltenmaximumstrategie) und nach der Methode von Cholesky bei fünfstelliger Rechnung gelöst werden.

$$\begin{aligned} 5x_1 + 7x_2 + 6x_3 + 5x_4 &= 12 \\ 7x_1 + 10x_2 + 8x_3 + 7x_4 &= 19 \\ 6x_1 + 8x_2 + 10x_3 + 9x_4 &= 17 \\ 5x_1 + 7x_2 + 9x_3 + 10x_4 &= 25 \end{aligned}$$

Man führe einen Schritt Nachiteration aus und bestimme die Konditionszahl von \mathbf{A} zur qualitativen Erklärung der gemachten Feststellungen.

Aufgabe 2.10. Welche der symmetrischen Matrizen

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & -2 \\ -1 & 3 & -2 & 4 \\ 0 & -2 & 4 & -3 \\ -2 & 4 & -3 & 5 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 4 & -2 & 4 & -6 \\ -2 & 2 & -2 & 5 \\ 4 & -2 & 13 & -18 \\ -6 & 5 & -18 & 33 \end{pmatrix}$$

ist positiv definit?

Aufgabe 2.11. Man zeige, dass die Hilbert-Matrix

$$\mathbf{H} := \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & \dots \\ 1/2 & 1/3 & 1/4 & 1/5 & \dots \\ 1/3 & 1/4 & 1/5 & 1/6 & \dots \\ 1/4 & 1/5 & 1/6 & 1/7 & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{pmatrix} \in \mathbb{R}^{n,n}, \quad h_{ik} = \frac{1}{i+k-1},$$

für beliebige Ordnung n positiv definit ist.

Dann berechne man ihre Inverse für $n = 3, 4, 5, \dots, 12$, welche ganzzahlige Matrixelemente besitzt, mit der Methode von Cholesky. Daraus bestimme man das Wachstum der Konditionszahl $\kappa(\mathbf{H})$ für zunehmendes n .

Mit Verfahren von Kapitel 5 ermittle man die Konditionszahl $\kappa_2(\mathbf{H})$.

Aufgabe 2.12. Das symmetrische, tridiagonale System

$$\begin{array}{rrr} -0.24x_1 & +1.76x_2 & = 1.28 \\ -1.05x_1 & +1.26x_2 & = 0.48 \\ & 1.12x_2 & -2.12x_3 & = 1.16 \\ & 1.34x_3 & +0.36x_4 & = -0.46 \\ & 1.29x_4 & +1.05x_5 & +0.66x_6 = -0.66 \\ & 0.96x_5 & +2.04x_6 & = -0.57 \end{array}$$

ist mit der relativen Spaltenmaximumstrategie und fünfstelliger Rechnung zu lösen. Welches sind die Matrizen \mathbf{L} und \mathbf{R} der zeilenpermutierten Matrix \mathbf{A} des Systems?

Aufgabe 2.13. Das tridiagonale Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ mit

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 3 & 1 & & \\ & & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 3 & 1 \\ & & & & & 1 & 2 & 1 \\ & & & & & & 1 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

ist nach dem Prinzip der zyklischen Reduktion für Vektorrechner, auf Grund der RQ -Zerlegung und der Odd-Even-Elimination für Parallelrechner zu lösen.

3 Interpolation und Approximation

Dieses Kapitel soll die wichtigsten Möglichkeiten beschreiben, mit denen eine reellwertige Funktion $f(x)$ der reellen Variablen x oder eine vorliegende Datentabelle (x_i, y_i) durch eine einfache Funktion – sagen wir $g(x)$ – in geschlossener Form angenähert werden können.

Bei der Approximation wird diese Funktion so bestimmt, dass eine Norm der Differenz zwischen den Werten der gesuchten Funktion $g(x_i)$ und der gegebenen Datentabelle

$$\|g - y\| \quad \text{mit den Vektoren } g = \{g(x_i)\} \text{ und } y = \{y_i\}$$

bzw. zwischen der gesuchten und der gegebenen Funktion $\|g(x) - f(x)\|$ minimal wird.

Bei der Interpolation wird die approximierende Funktion so konstruiert, daß sie an vorgegebenen Stellen mit der gegebenen Funktion oder den Daten der Tabelle übereinstimmt:

$$g(x_i) = f(x_i) \quad \text{bzw.} \quad g(x_i) = y_i.$$

Darüber hinaus hat die Interpolation eine gewisse Bedeutung als Hilfsmittel zur Entwicklung numerischer Verfahren, so bei der numerischen Differenziation, siehe Abschnitt 3.1.6, oder bei der numerischen Integration von Funktionen, siehe Kapitel 7.

Die Qualität der Approximation oder Interpolation hängt von der Entscheidung für eine der Methoden und von der Wahl des approximierenden Funktionensystems ab, also von der Frage, ob z.B. $g(x)$ ein Polynom oder eine trigonometrische Funktion ist. Deshalb sollen die unterschiedlichen Verfahren in einem Kapitel behandelt werden. Alle Methoden lassen sich von einer auf mehrere Dimensionen übertragen. Auf mehrdimensionale Aufgaben werden wir auch kurz eingehen. Zunächst lautet die wichtigste Frage

Interpolation oder Approximation ?

Für die Approximation einer Tabelle von Daten (x_i, y_i) , $i = 0, 1, \dots, n$, gibt es zwei typische Situationen:

1. Es sind sehr viele Daten gegeben, d.h. n ist sehr groß. Dann ist Interpolation nicht sinnvoll, besonders dann nicht, wenn die Daten Mess- und Beobachtungsfehlern unterliegen. Es sollte dann eine möglichst glatte Kurve durch die “Datenwolke” gelegt werden wie in Abb. 3.1 links.
2. Es sind nur wenige Daten gegeben, und es ist sinnvoll oder sogar wichtig, dass die approximierende Funktion an den gegebenen Stellen x_i die gegebenen Funktionswerte y_i bzw. $f(x_i)$ auch annimmt. Dann wird man ein Interpolationsverfahren wählen wie in Abb. 3.1 rechts.

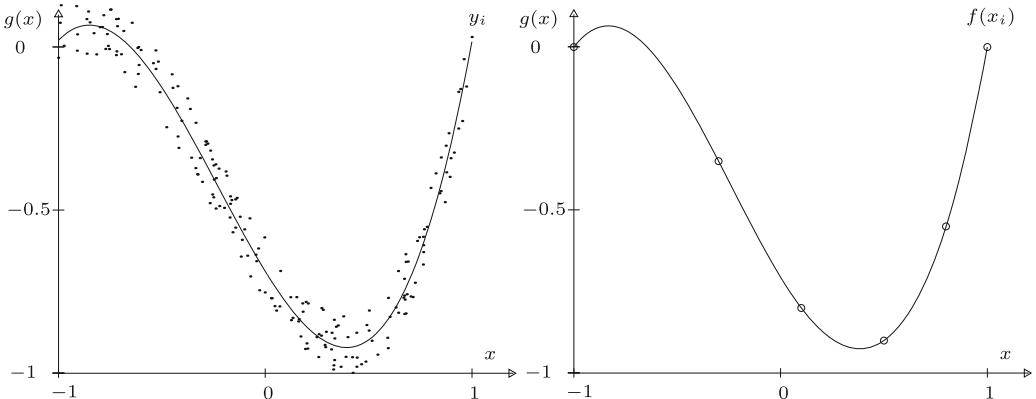


Abb. 3.1 Polynomapproximation und Interpolation mit kubischen Splines.

3.1 Polynominterpolation

3.1.1 Problemstellung

Gegeben seien $n + 1$ voneinander verschiedene reelle *Stützstellen* x_0, x_1, \dots, x_n und zugehörige beliebige *Stützwerte* y_0, y_1, \dots, y_n . Gesucht ist ein Polynom n -ten Grades

$$P_n(x) := a_0 + a_1x + \dots + a_nx^n, \quad (3.1)$$

welches die Interpolationsbedingung

$$P_n(x_i) = y_i, \quad i = 0, 1, \dots, n \quad (3.2)$$

erfüllt.

Satz 3.1. Zu $n+1$ beliebigen Wertepaaren (x_i, y_i) , $i = 0, 1, \dots, n$, mit paarweise verschiedenen reellen Stützstellen $x_i \neq x_j$ für alle $i \neq j$ existiert genau ein Interpolationspolynom $P_n(x)$ mit der Eigenschaft (3.2), dessen Grad höchstens gleich n ist.

Beweis. Die Aussage dieses Satzes ergibt sich aus der Konstruktion der in Abschnitt 3.1.2 vorgestellten Lagrange-Interpolation. \square

Die gegebenen Stützwerte y_i können Werte einer gegebenen Funktion f sein, die an den Stützstellen x_i interpoliert werden soll,

$$f(x_i) = y_i, \quad i = 0, 1, \dots, n, \quad (3.3)$$

oder diskrete Daten einer Tabelle

$$(x_i, y_i), \quad i = 0, 1, \dots, n.$$

Im ersten Fall sind gute Aussagen über den Fehler bei der Interpolation der Daten möglich. Im anderen Fall benötigt man gewisse Annahmen über den zu Grunde liegenden Prozess, um entsprechende Aussagen zu bekommen. Diese sind dann natürlich nicht mit derselben

Sicherheit zu treffen. Für den Fehler ist auch die Stützstellenverteilung von großem Einfluss. Ist eine gegebene Funktion zu interpolieren, die überall im Intervall $[a, b]$ ausgewertet werden kann, so besteht die Möglichkeit, eine optimale Stützstellenverteilung zu wählen.

Satz 3.2. Alle Stützstellen liegen im Intervall $[a, b]$. f sei eine im Intervall $[a, b]$ $(n+1)$ mal stetig differenzierbare Funktion: $f \in C^{n+1}[a, b]$, und es sei $y_i := f(x_i)$, $i = 0, 1, \dots, n$. Sei P_n das die Tabelle (x_i, y_i) interpolierende Polynom und

$$\omega(x) := (x - x_0)(x - x_1) \cdots (x - x_n). \quad (3.4)$$

Dann gibt es zu jedem $\tilde{x} \in [a, b]$ ein $\xi \in (a, b)$ mit

$$f(\tilde{x}) - P_n(\tilde{x}) = \frac{\omega(\tilde{x}) f^{(n+1)}(\xi)}{(n+1)!}. \quad (3.5)$$

Damit gilt

$$|f(\tilde{x}) - P_n(\tilde{x})| \leq \frac{|\omega(\tilde{x})|}{(n+1)!} \max_{\xi \in [a, b]} |f^{(n+1)}(\xi)|. \quad (3.6)$$

Beweis. Wenn $\tilde{x} = x_k$ für ein $k \in \{0, 1, \dots, n\}$, dann verschwindet der Fehler.

Sei deshalb $\tilde{x} \neq x_i$ für alle i , sei

$$F(x) := f(x) - P_n(x) - K \omega(x), \quad F \in C^{n+1}[a, b]$$

und K so bestimmt, dass $F(\tilde{x}) = 0$. Das geht, weil

$$\omega(\tilde{x}) \neq 0 \Rightarrow K = \frac{f(\tilde{x}) - P_n(\tilde{x})}{\omega(\tilde{x})}.$$

Dann besitzt $F(x)$ in $[a, b]$ mindestens die $(n+2)$ verschiedenen Nullstellen $x_0, \dots, x_n, \tilde{x}$. Daraus folgt mit dem Satz von Rolle:

- $F'(x)$ hat mindestens $n+1$ Nullstellen.
- $F''(x)$ hat mindestens n Nullstellen.
- ...
- $F^{(n+1)}(x)$ hat mindestens 1 Nullstelle.

Für jede Nullstelle ξ von $F^{(n+1)}$ gilt:

$$\begin{aligned} 0 = F^{(n+1)}(\xi) &= f^{(n+1)}(\xi) - \underbrace{0}_{P^{(n+1)}(x)} - K \underbrace{(n+1)!}_{\omega^{(n+1)}(x)} \\ \Rightarrow K &= \frac{f^{(n+1)}(\xi)}{(n+1)!} \\ \Rightarrow f(\tilde{x}) - P_n(\tilde{x}) &= \omega(\tilde{x}) \frac{f^{(n+1)}(\xi)}{(n+1)!} \end{aligned}$$

□

Der Satz zeigt, dass man den Fehlerverlauf im Intervall $[a, b]$ gut studieren kann, wenn man sich den Funktionsverlauf von ω ansieht. ω wird bei äquidistanten Stützstellen in den Randintervallen sehr viel größer als in der Mitte des Stützstellenbereichs.

Einen gleichmäßigeren Verlauf der Fehlerkurve erhält man, wenn statt äquidistanter Stützstellen die so genannten *Tschebyscheff-Punkte* als Stützstellen gewählt werden:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{i}{n}\pi\right), \quad i = 0, 1, \dots, n. \quad (3.7)$$

Dies sind die Extremalstellen der Tschebyscheff-Polynome, siehe Abschnitt 3.8.1. Die Tschebyscheff-Punkte sind zum Rand des Intervalls $[x_0, x_n]$ hin dichter verteilt als in der Mitte. Deshalb ist der Verlauf der Fehlerkurve ω ausgeglichener siehe Abb. 3.2.

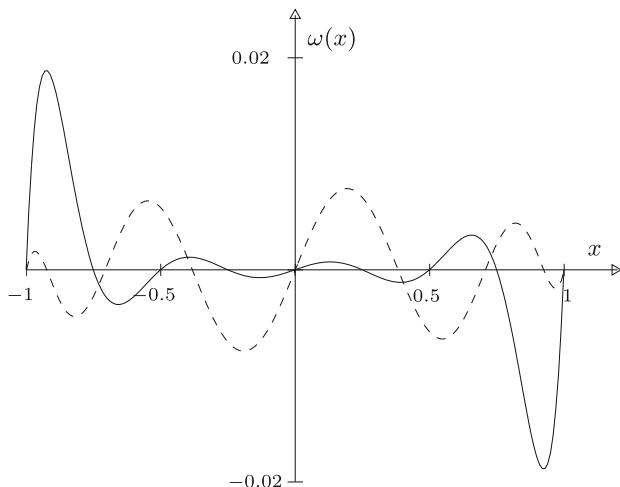


Abb. 3.2 ω für äquidistante (—) und Tschebyscheff-Punkte (- -), $n=8$.

Numerische Verfahren

Wir wollen die folgenden Verfahren behandeln:

- Die *Lagrange-Interpolation* zeigt am schönsten, dass immer ein Polynom nach Satz 3.1 konstruiert werden kann. Numerisch ist sie nicht empfehlenswert, da die Auswertung des berechneten Polynoms aufwändiger ist als bei der Newton-Interpolation.
- Die *Newton-Interpolation* konstruiert mit geringem Aufwand und numerisch stabil das Interpolationspolynom. Das rekursive Berechnungsschema ist sowohl für eine rasche manuelle Berechnung als auch für eine elegante rechnerische Umsetzung besonders geeignet.
- Bei der *Hermite-Interpolation* werden zusätzlich zu Funktionswerten auch Ableitungswerte interpoliert. Das Schema der Newton-Interpolation lässt sich leicht auf die Lösung dieser Aufgabe verallgemeinern.

3.1.2 Lagrange-Interpolation

Das Interpolationspolynom P lässt sich mit Hilfe der $n + 1$ Lagrange-Polynome

$$\begin{aligned} L_i(x) &:= \prod_{\substack{j=0 \\ j \neq i}}^n \frac{(x - x_j)}{(x_i - x_j)} \\ &= \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \end{aligned} \quad (3.8)$$

angeben:

$$P_n(x) := \sum_{i=0}^n y_i L_i(x). \quad (3.9)$$

Es erfüllt (3.2), da ja für die Lagrange-Polynome gilt

$$L_i(x_k) = \delta_{ik} = \begin{cases} 1, & \text{falls } i = k, \\ 0, & \text{falls } i \neq k. \end{cases} \quad (3.10)$$

Das Polynom $P_n(x)$ nach (3.9) ist auch eindeutig bestimmt. Gäbe es nämlich ein zweites Polynom $Q_n(x)$ vom Höchstgrad n mit

$$P_n(x_k) = Q_n(x_k) = y_k, \quad k = 0, 1, \dots, n, \quad (3.11)$$

dann wäre auch das Differenzpolynom $D(x) := P_n(x) - Q_n(x)$ höchstens vom Grad n und besäße die $n + 1$ paarweise verschiedenen Nullstellen x_0, x_1, \dots, x_n . Nach dem Fundamentalsatz der Algebra muss damit $D(x) = 0$ und damit $P_n(x) = Q_n(x)$ sein.

3.1.3 Newton-Interpolation

Die Idee zu diesem Verfahren geht von einer anderen Darstellung des Interpolationspolynoms aus:

$$\begin{aligned} P_n(x) &:= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots \\ &\quad + c_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{aligned} \quad (3.12)$$

Die Koeffizienten c_i dieser mit (3.1) äquivalenten Darstellung lassen sich sukzessive berechnen, wenn nacheinander die Stützstellen und Stützwerte eingesetzt werden:

$$\begin{aligned} P_n(x_0) &= c_0 \implies c_0 = y_0 \\ P_n(x_1) &= c_0 + c_1(x_1 - x_0) \implies c_1 = \frac{y_1 - y_0}{x_1 - x_0} \\ &\quad \dots \end{aligned}$$

Bei der Fortführung dieser rekursiven Auflösung entstehen weiterhin Brüche von Differenzen von vorher berechneten Brüchen. Diese so genannten *dividierten Differenzen* lassen sich rekursiv berechnen:

0. dividierte Differenz:

$$[y_k] := y_k, \quad k = 0, 1, \dots, n.$$

1. dividierte Differenz für $k = 0, 1, \dots, n - 1$:

$$[y_k, y_{k+1}] := \frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{[y_{k+1}] - [y_k]}{x_{k+1} - x_k}.$$

Dies wird entsprechend fortgeführt mit den j -ten dividierten Differenzen

$$\begin{aligned} [y_k, y_{k+1}, \dots, y_{k+j}] &:= \frac{[y_{k+1}, \dots, y_{k+j}] - [y_k, \dots, y_{k+j-1}]}{x_{k+j} - x_k}, \\ j &= 2, 3, \dots, n, \quad k = 0, 1, \dots, n - j. \end{aligned} \tag{3.13}$$

Die dividierten Differenzen werden jetzt mit gewissen Interpolationspolynomen in Verbindung gebracht. Wenn wir mit $P_{k,k+1,\dots,k+j}^*$ das Interpolationspolynom j -ten Grades zu den $(j + 1)$ Stützstellen $x_k, x_{k+1}, \dots, x_{k+j}$ bezeichnen, dann gilt definitionsgemäß

$$P_{k,k+1,\dots,k+j}^*(x_{k+i}) = y_{k+i}, \quad i = 0, 1, \dots, j, \tag{3.14}$$

und speziell ist mit dieser Bezeichnungsweise

$$P_k^*(x_k) = y_k = [y_k]. \tag{3.15}$$

Satz 3.3. Für $1 \leq j \leq n$ gilt die Rekursionsformel

$$P_{k,k+1,\dots,k+j}^*(x) = \frac{(x - x_k)P_{k+1,\dots,k+j}^*(x) - (x - x_{k+j})P_{k,\dots,k+j-1}^*(x)}{x_{k+j} - x_k} \tag{3.16}$$

Beweis. Die Interpolationspolynome $P_{k+1,\dots,k+j}^*(x)$ und $P_{k,\dots,k+j-1}^*(x)$ sind vom Höchstgrad $(j - 1)$. Es ist zu zeigen, dass die rechte Seite von (3.16) die Stützwerte y_k, \dots, y_{k+j} an den Stützstellen x_k, \dots, x_{k+j} interpoliert. Wegen der Interpolationseigenschaften der Polynome rechts in (3.16) gilt trivialerweise

$$P_{k,k+1,\dots,k+j}^*(x_k) = y_k \quad \text{und} \quad P_{k,k+1,\dots,k+j}^*(x_{k+j}) = y_{k+j},$$

und weiter für $i = 1, 2, \dots, j - 1$

$$P_{k,k+1,\dots,k+j}^*(x_{k+i}) = \frac{(x_{k+i} - x_k)y_{k+i} - (x_{k+i} - x_{k+j})y_{k+j}}{x_{k+j} - x_k} = y_{k+i}.$$

Nach Satz 3.1 stellt damit die rechte Seite von (3.16) das eindeutig bestimmte Interpolationspolynom $P_{k,k+1,\dots,k+j}^*(x)$ dar. \square

Wegen (3.13) ist eine unmittelbare Folge von Satz 3.3, dass die dividierten Differenzen $[y_0, y_1, \dots, y_j]$ mit den Koeffizienten c_j in (3.12) identisch sind, da ja c_j Höchstkoeffizient des Interpolationspolynoms $P_{0,1,\dots,j}^*(x)$ zu den $(j + 1)$ Stützpunkten (x_i, y_i) , $i = 0, 1, \dots, j$ ist.

Jetzt kann die Berechnung der Koeffizienten in dem so genannten Newton-Schema zusammengefasst werden, das wir nur für $n = 3$ angeben wollen:

$$\begin{array}{c|ccccc} x_0 & y_0 = c_0 & & & & \\ x_1 & y_1 & [y_0, y_1] = c_1 & & & \\ x_2 & y_2 & [y_1, y_2] & [y_0, y_1, y_2] = c_2 & & \\ x_3 & y_3 & [y_2, y_3] & [y_1, y_2, y_3] & [y_0, y_1, y_2, y_3] = c_3 & \end{array} \quad (3.17)$$

Diese rekursive Berechnung der Koeffizienten c_j stellt den ersten Schritt des Algorithmus Tab. 3.1 dar. Dabei wird von Spalte zu Spalte von unten nach oben gerechnet, damit die nicht mehr benötigten Werte überschrieben werden können.

Für die Berechnung von Polynomwerten eignet sich auch die Form (3.12) besonders gut, da gemeinsame Faktoren ausgeklammert werden können:

$$\begin{aligned} P_n(x) &= c_0 + c_1(x - x_0) + \cdots + c_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}) \\ &= c_0 + (x - x_0) \{ c_1 + (x - x_1) [c_2 + (x - x_2) (\\ &\quad \cdots c_{n-1} + (x - x_{n-1}) c_n)] \}. \end{aligned} \quad (3.18)$$

Diese Art der Polynomwertberechnung wird *Horner-Schema* genannt. Das Horner-Schema stellt den zweiten Schritt des Algorithmus Tab. 3.1 dar.

Tab. 3.1 Algorithmus zur Newton-Interpolation.

1. Berechnung der Koeffizienten
Für $k = 0, 1, \dots, n$:
$c_k := y_k$
Für $k = 1, 2, \dots, n$:
für $i = n, n-1, \dots, k$:
$c_i := \frac{c_i - c_{i-1}}{x_i - x_{i-k}}$
2. Horner-Schema: Berechnung von Polynomwerten
$p := c_n$
Für $k = n-1, n-2, \dots, 0$:
$p := c_k + (x - x_k)p$
Für den so berechneten Wert p gilt $p = P_n(x)$.

Beispiel 3.1. Berechnet werden soll $p := \ln(1.57)$. Es liegt eine Tafel für den natürlichen Logarithmus vor mit folgenden Werten:

x	1.4	1.5	1.6	1.7
$\ln(x)$	0.3364722366	0.4054651081	0.4700036292	0.5306282511

Für diese Tabelle erstellen wir das Newton-Schema:

1.4	<u>0.3364722366</u>			
1.5	<u>0.4054651081</u>	<u>0.689928715</u>	<u>-0.22271752</u>	<u>0.0900752</u>
1.6	0.4700036292	0.645385211	-0.19569496	
1.7	0.5306282511	0.606246219		

Aus diesem Schema können wir die Koeffizienten mehrerer Interpolationspolynome mit Stützstellen, die den Punkt 1.57 umgeben, ablesen. Da ist zunächst das lineare Polynom mit den eingekästelten Koeffizienten

$$P_1(x) = 0.4054651081 + 0.645385211(x - 1.5).$$

Bei Berücksichtigung der ersten drei Tabellenwerte ergibt sich das quadratische Polynom mit den doppelt unterstrichenen Koeffizienten

$$\begin{aligned} P_2(x) &= 0.3364722366 + 0.689928715(x - 1.4) - 0.22271752(x - 1.4)(x - 1.5) \\ &= 0.3364722366 + (x - 1.4)[0.689928715 - (x - 1.5)0.22271752]. \end{aligned}$$

Alle unterstrichenen Werte in der oberen Schrägzeile sind schließlich die Koeffizienten des gesuchten Polynoms dritten Grades

$$\begin{aligned} P_3(x) &= 0.3364722366 + 0.689928715(x - 1.4) - 0.22271752(x - 1.4)(x - 1.5) \\ &\quad + 0.0900752(x - 1.4)(x - 1.5)(x - 1.6) \\ &= 0.3364722366 + (x - 1.4)[0.689928715 \\ &\quad + (x - 1.5)\{-0.22271752 + 0.0900752(x - 1.6)\}] \\ &= P_2(x) + 0.0900752(x - 1.4)(x - 1.5)(x - 1.6). \end{aligned}$$

Man sieht an der letzten Zeile, dass man durch Hinzufügen eines Tabellenwertpaars und Nachberechnung einer unteren Schrägzeile im Newton-Schema das Polynom des nächsthöheren Grades erhält. Auch das liegt an der Form (3.12) der Interpolationspolynome.

Wir wollen jetzt die Genauigkeit der drei Polynome vergleichen. Es ist (auf zehn Stellen genau) $\ln(1.57) = 0.4510756194$. Damit ergibt sich:

$$\begin{aligned} P_1(1.57) &= 0.4506420729, & \text{Fehler : } & 4.3 \cdot 10^{-4}, \\ P_2(1.57) &= 0.4511097797, & \text{Fehler : } & -3.4 \cdot 10^{-5}, \\ P_3(1.57) &= 0.4510776229, & \text{Fehler : } & -2.0 \cdot 10^{-6}. \end{aligned} \quad \triangle$$

Sind die Stützstellen äquidistant wie in diesem Beispiel, so kann das Schema der dividierten Differenzen zu einem reinen Differenzenschema reduziert werden, wobei die dann berechneten Koeffizienten c_j noch durch $j! h^j$ dividiert werden müssen. Insgesamt ergibt sich eine vereinfachte Darstellung [Sch 97].

3.1.4 Hermite-Interpolation

Zusätzlich zu den Stützwerten sollen jetzt an den Stützstellen noch Ableitungswerte interpoliert werden. Das können Ableitungen verschiedener Ordnung, gegeben an unterschiedlichen Stützstellen, sein. Das Schema der dividierten Differenzen kann weiterhin angewendet werden, wenn wir uns bei mehrfachen Stützstellen einen Grenzübergang vorstellen, etwa bei

einer doppelten Stützstelle x_k mit dem Stützwert $y_k := f(x_k)$

$$[y_k, y_k] = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{(x_k + h) - x_k} = \frac{f(x_k + h) - f(x_k)}{h} = f'(x_k). \quad (3.19)$$

Allgemein ergibt sich bei m -facher Stützstelle x_k

$$[y_k, y_k, \dots, y_k] := \frac{1}{(m-1)!} f^{(m-1)}(x_k). \quad (3.20)$$

Wir wollen aber diese Aufgabe nur in folgender spezieller Form behandeln:

Gegeben seien $n+1$ voneinander verschiedene Stützstellen x_0, x_1, \dots, x_n , diesen zugeordnete Stützwerte y_0, y_1, \dots, y_n und Ableitungswerte y'_0, y'_1, \dots, y'_n .

Gesucht ist ein Polynom P_{2n+1} vom Höchstgrad $2n+1$, für das gilt

$$\begin{aligned} P_{2n+1}(x_i) &= y_i, \quad i = 0, 1, \dots, n, \\ P'_{2n+1}(x_i) &= y'_i, \quad i = 0, 1, \dots, n. \end{aligned} \quad (3.21)$$

Die Lösung dieser Aufgabe ergibt sich aus (3.19), indem das Newton-Schema (3.17) entsprechend erweitert wird. Das ergibt für $n=1$ das folgende Schema:

$$\begin{array}{c|cccc} x_0 & y_0 = c_0 & [y_0, y_0] := y'_0 = c_1 & [y_0, y_0, y_1] = c_2 & [y_0, y_0, y_1, y_1] = c_3 \\ x_0 & y_0 & [y_0, y_1] & [y_0, y_1, y_1] & \\ x_1 & y_1 & [y_1, y_1] := y'_1 & & \\ x_1 & y_1 & & & \end{array}$$

Das Interpolationspolynom enthält entsprechend quadratische Terme:

$$P_3(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)^2 + c_3(x - x_0)^2(x - x_1)$$

oder allgemein

$$\begin{aligned} P_{2n+1}(x) = & c_0 + c_1(x - x_0) + c_2(x - x_0)^2 + c_3(x - x_0)^2(x - x_1) \\ & + c_4(x - x_0)^2(x - x_1)^2 + \dots \\ & + c_{2n+1}(x - x_0)^2(x - x_1)^2 \cdots (x - x_{n-1})^2(x - x_n). \end{aligned}$$

Beispiel 3.2. Wir wollen die Funktion

$$\begin{aligned} f(x) &= e^x \sin(5x) \quad \text{mit} \\ f'(x) &= e^x (\sin(5x) + 5 \cos(5x)) \end{aligned}$$

im Intervall $[0, 1]$ nach Hermite interpolieren und die Genauigkeit mit der der Newton-Interpolation vergleichen. Wir wählen als Stützpunkte $n+1$ äquidistante Punkte x_i . Für $n=1$ ergibt sich mit auf vier Stellen hinter dem Komma gerundeten Werten folgendes Hermite-Schema

$$\begin{array}{c|cccc} 0 & 0 = c_0 & 5 = c_1 & -7.6066 = c_2 & 11.4620 = c_3 \\ 0 & 0 & -2.6066 & 3.8554 & \\ 1 & -2.6066 & 1.2487 & & \\ 1 & -2.6066 & & & \end{array}$$

Das ergibt ein Polynom, das die Funktion und ihre Ableitung an den Rändern des Intervalls $[0, 1]$ interpoliert.

$$P_H(x) = 5x - 7.6066x^2 + 11.4620x^2(x - 1).$$

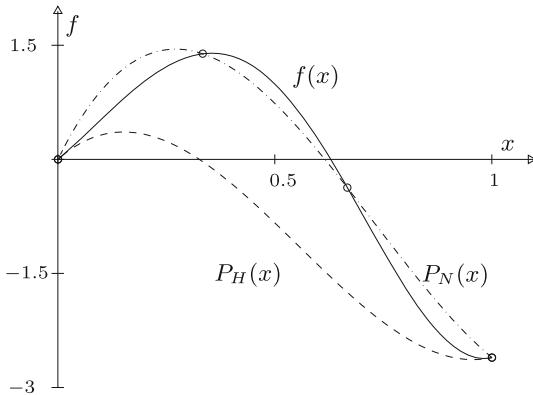


Abb. 3.3 Hermite- (- -) und Newton-Interpolation (· –) 3. Grades.

In Abb. 3.3 sehen wir diese naturgemäß schlechte Approximation der Funktion f neben der Newton-Interpolierenden $P_N(x)$ gleichen Grades, also mit doppelt so vielen Stützstellen.

Wegen der Auswertung von Funktion und Ableitung ist der Aufwand der Hermite-Interpolation für $n + 1$ Stützstellen etwa so groß wie der der Newton-Interpolation mit $2n + 2$ Stützstellen. Deshalb wollen wir die entsprechenden Interpolationspolynome gleichen Grades mit wachsendem n über den maximalen Fehler in $[0, 1]$ vergleichen.

$n + 1$	Hermite-Fehler	$2n + 2$	Newton-Fehler
2	1.8	4	0.40
3	0.077	6	0.038
4	0.0024	8	0.0018
5	0.000043	10	0.000044

Die Entscheidung zwischen Newton- und Hermite-Interpolation sollte also nicht nach Genauigkeitsgesichtspunkten, sondern nach Datenlage erfolgen. \triangle

3.1.5 Inverse Interpolation

Die Aufgabe, zu der Datentabelle $(x_i, y_i = f(x_i))$, $i = 0, 1, \dots, n$, einer vorgegebenen Funktion f eine Stelle \bar{x} zu finden, für die näherungsweise $f(\bar{x}) = \bar{y}$ gilt mit gegebenem \bar{y} , kann mit inverser Interpolation gelöst werden. Das Vorgehen kann aber nur dann aus Gründen der Eindeutigkeit sinnvoll sein, falls die Funktion $f(x)$ im Interpolationsintervall *monoton* ist und dementsprechend alle y_i voneinander verschieden sind. Unter dieser Voraussetzung ist der Newtonsche Algorithmus geeignet. Im Unterschied zur normalen Polynominterpolation bezeichnen wir das Interpolationspolynom der inversen Aufgabe mit $x = Q(y)$. Für sie werden einfach im Interpolationsschema die Werte (Spalten im Newton-Schema) für x_i und y_i ausgetauscht.

Diese Methode ist bei der Suche nach Nullstellen $\bar{y} = 0$ hilfreich, wie wir in Abschnitt 4.2.4 sehen werden. Wir wollen ein einführendes Beispiel rechnen.

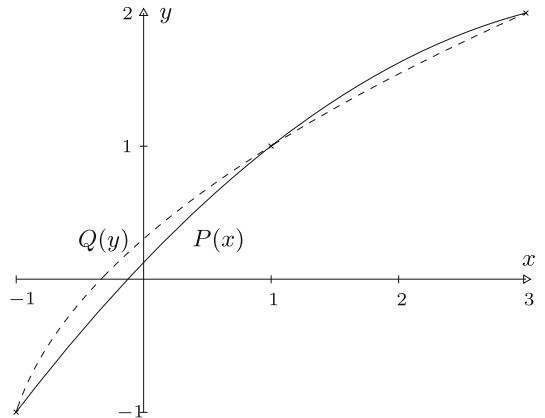


Abb. 3.4 Inverse Interpolation und Interpolation.

Beispiel 3.3. Für drei Punkte ($n = 2$) soll die direkte der inversen Interpolation gegenübergestellt werden.

x_i	y_i
-1	-1
1	1
3	2

Dieses normale Newton-Schema ergibt das interpolierende Polynom

$$y = P_2(x) = -1 + (x + 1) - \frac{1}{8}(x + 1)(x - 1).$$

Inverse Interpolation ergibt

y_i	x_i
-1	-1
1	1
2	3

$$x = Q_2(y) = -1 + (y + 1) + \frac{1}{3}(y + 1)(y - 1).$$

Beide Funktionen interpolieren dieselbe Tabelle, siehe auch Abb. 3.4, aber nur in die der inversen Interpolation kann ein y -Wert wie $\bar{y} = 0$ eingesetzt werden, um den zugehörigen Wert $\bar{x} = Q_2(\bar{y}) = -1/3$ zu finden. \triangle

3.1.6 Anwendung: Numerische Differenziation

Interpolationspolynome zu tabellarisch gegebenen Funktionen stellen gleichzeitig die Grundlage dar, Ableitungen der Funktionen näherungsweise zu berechnen. Die auf diese Weise gewonnenen Formeln zur *numerischen Differenziation* sind dann selbstverständlich auch zur genäherten Berechnung von Ableitungen analytisch berechenbarer Funktionen anwendbar. Die Darstellung des Interpolationspolynoms nach Lagrange (3.9) eignet sich besonders gut

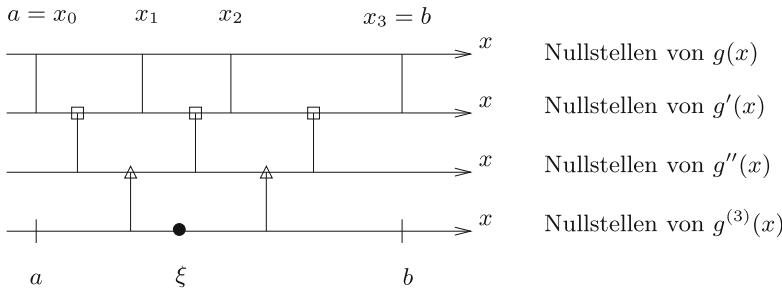


Abb. 3.5 Existenz von Nullstellen nach dem Satz von Rolle.

zur Herleitung der gewünschten Differenzierungsregeln. Differenzieren wir (3.9) n mal nach x und definieren noch den Nenner des i -ten Lagrange-Polynoms als

$$\lambda_i := \frac{1}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)},$$

dann ist allgemein

$$\frac{d^n P_n(x)}{dx^n} = \sum_{i=0}^n y_i n! \lambda_i \approx f^{(n)}(x) \quad (3.22)$$

eine mögliche Approximation der n -ten Ableitung $f^{(n)}(x)$. Es gilt dazu der

Satz 3.4. Sei $f(x)$ im Intervall $[a, b]$ mit $a = \min_i(x_i), b = \max_i(x_i)$ eine mindestens n mal stetig differenzierbare Funktion, dann existiert ein $\xi \in (a, b)$ so, dass

$$f^{(n)}(\xi) = \sum_{i=0}^n y_i n! \lambda_i. \quad (3.23)$$

Beweis. Wir betrachten die Funktion $g(x) := f(x) - P_n(x)$, wo $P_n(x)$ das Interpolationspolynom zu den Stützstellen x_i , $i = 0, 1, \dots, n$, mit den Stützwerten $y_i = f(x_i)$ sei. Dann hat aber $g(x)$ wegen der Interpolationseigenschaft mindestens die $(n + 1)$ Nullstellen x_0, x_1, \dots, x_n . Wenden wir auf $g(x)$ n mal den Satz von Rolle an, folgt die Existenz einer Stelle ξ im Innern des kleinstmöglichen Intervalls, das alle Stützstellen enthält derart, dass $g^{(n)}(\xi) = f^{(n)}(\xi) - P_n^{(n)}(\xi) = 0$. Das ist aber wegen (3.22) die Behauptung (3.23). In Abb. 3.5 ist die Situation im konkreten Fall $n = 3$ dargestellt. \square

Die durch (3.22) erklärte *Regel der numerischen Differenziation* zur Approximation der n -ten Ableitung einer (tabellarisch) gegebenen Funktion wird im Allgemeinen nur für äquidistante Stützstellen $x_i = x_0 + ih$, $i = 0, 1, \dots, n$, verwendet. Dann wird

$$\begin{aligned} \lambda_i &= \frac{1}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \\ &= \frac{1}{(-1)^{n-i} h^n [i(i-1) \cdots 1][1 \cdot 2 \cdots (n-i)]} = \frac{(-1)^{n-i}}{h^n n!} \binom{n}{i}, \end{aligned} \quad (3.24)$$

und es ergibt sich für (3.22) speziell

$$f^{(n)}(x) \approx \frac{1}{h^n} \left[(-1)^n y_0 + (-1)^{n-1} \binom{n}{1} y_1 + (-1)^{n-2} \binom{n}{2} y_2 + \dots - \binom{n}{n-1} y_{n-1} + y_n \right]. \quad (3.25)$$

Die Approximation der n -ten Ableitung $f^{(n)}(x)$ berechnet sich nach (3.25) als eine durch h^n zu dividierende Linearkombination der Stützwerte y_i , die mit den Binomialkoeffizienten alternierenden Vorzeichens multipliziert werden, wobei der letzte positiv ist. Der Ausdruck (3.25) wird als der n -te *Differenzenquotient* der $(n+1)$ Stützwerte y_i bezeichnet. Für $n = 1, 2$ und 3 lauten die entsprechenden Formeln der numerischen Differenziation

$f'(x) \approx \frac{y_1 - y_0}{h}$	1. Differenzenquotient
$f''(x) \approx \frac{y_2 - 2y_1 + y_0}{h^2}$	2. Differenzenquotient
$f^{(3)}(x) \approx \frac{y_3 - 3y_2 + 3y_1 - y_0}{h^3}$	3. Differenzenquotient

Die Stelle ξ , an welcher der n -te Differenzenquotient die n -te Ableitung einer mindestens n mal stetig differenzierbaren Funktion $f(x)$ exakt liefert, liegt oft in der Nähe des Mittelpunktes $x_M = (x_0 + x_n)/2$. Im Fall von rein tabellarisch definierten Funktionen wird deshalb ein Differenzenquotient (3.26) die entsprechende Ableitung in der Regel in der Mitte des Interpolationsintervalls am besten approximieren.

Die p -te Ableitung einer Funktion kann aber ebenso gut auf Grund eines höhergradigen Interpolationspolynoms $P_n(x)$ mit $n > p$ approximiert werden. Da jetzt die p -te Ableitung $P_n^{(p)}(x)$ nicht konstant ist, muss die Stelle x genau definiert werden. Zur Illustration soll die erste Ableitung $f'(x)$ auf Grund eines quadratischen Interpolationspolynoms approximiert werden. Mit (3.8)/(3.9) und (3.24) ergibt sich

$$\begin{aligned} P_2(x) &= \frac{y_0(x - x_1)(x - x_2) - 2y_1(x - x_0)(x - x_2) + y_2(x - x_0)(x - x_1)}{2h^2}, \\ P'_2(x) &= \frac{y_0(2x - x_1 - x_2) - 2y_1(2x - x_0 - x_2) + y_2(2x - x_0 - x_1)}{2h^2}. \end{aligned}$$

Für die erste Ableitung zu den Stützstellen x_0, x_1 und x_2 ergeben sich daraus

$$f'(x_0) \approx \frac{1}{2h}[-3y_0 + 4y_1 - y_2] \quad (3.27)$$

$$f'(x_1) \approx \frac{1}{2h}[-y_0 + y_2] \quad (3.28)$$

$$f'(x_2) \approx \frac{1}{2h}[y_0 - 4y_1 + 3y_2] \quad (3.29)$$

Der Ausdruck (3.28) heißt *zentraler Differenzenquotient*. Er stellt die Steigung der Sekanten der Interpolationsparabel $P_2(x)$ durch die beiden Stützpunkte $(x_0, y_0), (x_2, y_2)$ dar. Sie ist bekanntlich gleich der Steigung der Tangente an die Parabel im mittleren Stützpunkt

(x_1, y_1) . Der zentrale Differenzenquotient (3.29) zeichnet sich dadurch aus, dass er die erste Ableitung tatsächlich besser approximiert als der erste Differenzenquotient (3.26).

Desgleichen können die erste und zweite Ableitung an bestimmten Stellen mit dem kubischen Interpolationspolynom angenähert werden. Die betreffende Rechnung ergibt die folgenden ausgewählten Formeln der numerischen Differenziation:

$$\boxed{\begin{aligned} f'(x_0) &\approx \frac{1}{6h}[-11y_0 + 18y_1 - 9y_2 + 2y_3] \\ f'(x_1) &\approx \frac{1}{6h}[-2y_0 - 3y_1 + 6y_2 - y_3] \\ f'(x_M) &\approx \frac{1}{24h}[y_0 - 27y_1 + 27y_2 - y_3] \\ f''(x_M) &\approx \frac{1}{2h^2}[y_0 - y_1 - y_2 + y_3] \\ x_M &= \frac{1}{2}(x_0 + x_3) \end{aligned}} \quad (3.30)$$

Beispiel 3.4. Um eine Problematik der numerischen Differenziation aufzuzeigen, wollen wir die zweite Ableitung der hyperbolischen Sinusfunktion $f(x) = \sinh(x)$ an der Stelle $x = 0.6$ auf Grund des zweiten Differenzenquotienten (3.26) berechnen, und zwar für eine Folge von Schrittweiten h , die gegen null strebt. Wir verwenden auf neun Stellen gerundete Funktionswerte, und die Auswertung des zweiten Differenzenquotienten erfolgt mit zehnstelliger Gleitpunktrechnung. In der folgenden Tabelle sind die sich ändernden Zahlenwerte h, x_0, x_2, y_0, y_2 und die resultierende Näherung für $f''(0.6) = \sinh(0.6) = 0.636653582$ zusammengefasst.

h	x_0	x_2	y_0	y_2	$f''(x_1) \approx$
0.1	0.50	0.7	0.521095305	0.758583702	0.637184300
0.01	0.59	0.61	0.624830565	0.648540265	0.636660000
0.001	0.599	0.601	0.635468435	0.637839366	0.637000000
0.0001	0.5999	0.6001	0.636535039	0.636772132	0.700000000
0.00001	0.59999	0.60001	0.636641728	0.636665437	10.00000000

Mit abnehmender Schrittweite h wird die Auslöschung immer katastrophaler, so dass für das kleinste $h = 0.00001$ ein ganz falscher Näherungswert resultiert. \triangle

Die numerische Differenziation ist ganz allgemein ein gefährlicher Prozess, der infolge der Auslöschung bei kleiner werdender Schrittweite Ergebnisse mit wachsenden relativen Fehlern liefert.

Um genauere Werte von Ableitungen zu bekommen, kann die Methode der *Extrapolation* angewendet werden. Um das Prinzip darzulegen, benötigt man eine Analyse des Fehlers. Dazu muss vorausgesetzt werden, dass die Funktion $f(x)$, deren p -te Ableitung durch numerische Differenziation zu berechnen ist, beliebig oft stetig differenzierbar ist in dem in Betracht kommenden abgeschlossenen Intervall, und dass sie sich dort in konvergente Taylor-Reihen entwickeln lässt.

Beginnen wir mit dem 1. Differenzenquotienten (3.26), in welchem wir für $y_1 = f(x_1) = f(x_0 + h)$ die Taylor-Reihe

$$y_1 = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f^{(3)}(x_0) + \frac{h^4}{4!}f^{(4)}(x_0) + \dots$$

einsetzen und erhalten mit $y_0 = f(x_0)$

$$\frac{y_1 - y_0}{h} = f'(x_0) + \frac{h}{2!}f''(x_0) + \frac{h^2}{3!}f^{(3)}(x_0) + \frac{h^3}{4!}f^{(4)}(x_0) + \dots \quad (3.31)$$

Der Differenzenquotient stellt $f'(x_0)$ dar mit einem Fehler, der sich als Potenzreihe in h erfassen lässt. Für den zentralen Differenzenquotienten (3.29) erhalten wir mit den Taylor-Reihen

$$\begin{aligned} y_2 &= f(x_2) = f(x_1 + h) \\ &= f(x_1) + hf'(x_1) + \frac{h^2}{2!}f''(x_1) + \frac{h^3}{3!}f^{(3)}(x_1) + \frac{h^4}{4!}f^{(4)}(x_1) + \dots \\ y_0 &= f(x_0) = f(x_1 - h) \\ &= f(x_1) - hf'(x_1) + \frac{h^2}{2!}f''(x_1) - \frac{h^3}{3!}f^{(3)}(x_1) + \frac{h^4}{4!}f^{(4)}(x_1) - \dots \\ \frac{1}{2h}[y_2 - y_0] &= f'(x_1) + \frac{h^2}{3!}f^{(3)}(x_1) + \frac{h^4}{5!}f^{(5)}(x_1) + \frac{h^6}{7!}f^{(7)}(x_1) + \dots \end{aligned} \quad (3.32)$$

Der zentrale Differenzenquotient (3.29) liefert nach (3.32) den Wert der ersten Ableitung $f'(x_1)$ mit einem Fehler, der sich in eine Potenzreihe nach h entwickeln lässt, in der nur *gerade Potenzen* auftreten. Der Fehler ist hier von *zweiter Ordnung*, während der Fehler (3.31) im ersten Differenzenquotienten von *erster Ordnung* ist. Der Approximationsfehler des zentralen Differenzenquotienten ist somit kleiner.

Desgleichen erhalten wir für den 2. Differenzenquotienten (3.26) das Resultat

$$\frac{y_2 - 2y_1 + y_0}{h^2} = f''(x_1) + \frac{2h^2}{4!}f^{(4)}(x_1) + \frac{2h^4}{6!}f^{(6)}(x_1) + \frac{2h^6}{8!}f^{(8)}(x_1) + \dots \quad (3.33)$$

Den drei Ergebnissen (3.31), (3.32) und (3.33) ist gemeinsam, dass eine *berechenbare Größe* $B(t)$, die von einem Parameter t abhängt, *einen gesuchten Wert A* (hier ein Ableitungswert) approximiert mit einem Fehler, der sich als Potenzreihe in t darstellt, so dass mit festen Koeffizienten a_1, a_2, \dots gilt

$$B(t) = A + a_1t + a_2t^2 + a_3t^3 + \dots + a_nt^n + \dots \quad (3.34)$$

Aus numerischen Gründen oder aber aus Gründen des Aufwands ist es oft nicht möglich, die berechenbare Größe $B(t)$ für einen so kleinen Parameterwert t zu bestimmen, dass $B(t)$ eine hinreichend gute Approximation für A darstellt. Das *Prinzip der Extrapolation* besteht nun darin, für einige Parameterwerte $t_0 > t_1 > t_2 > \dots > t_n > 0$ die Werte $B(t_k)$ zu berechnen und dann die zugehörigen Interpolationspolynome $P_k(t)$ sukzessive an der außerhalb der Stützstellen liegenden Stelle $t = 0$ auszuwerten. Mit zunehmendem k stellen die Werte $P_k(0)$ bessere Näherungswerte für den gesuchten Wert $B(0) = A$ dar. Die Extrapolation wird abgebrochen, sobald die extrapolierten Werte $P_k(0)$ den gesuchten Wert A mit der vorgegebenen Genauigkeit darstellen. Die Durchführung des Extrapolationsprozesses kann mit der Newton-Interpolation erfolgen.

Beispiel 3.5. Die zweite Ableitung der Funktion $f(x) = \sinh(x)$ an der Stelle $x = 0.6$ soll mit der Methode der Extrapolation möglichst genau berechnet werden. Wegen (3.33) ist der Parameter $t = h^2$. Damit die berechenbaren Werte $B(t_k) := [y_2 - 2y_1 + y_0]/t_k$ einen möglichst kleinen relativen Fehler aufweisen, darf die Schrittweite nicht zu klein sein. In der nachfolgenden Tabelle sind die wesentlichen Zahlenwerte zusammengestellt. Als Funktionswerte $y_i = \sinh(x_i)$ wurden auf neun wesentliche Stellen gerundete Zahlenwerte verwendet. Das Newton-Schema wurde mit sechzehnstelliger Genauigkeit durchgerechnet ebenso wie die Berechnung der Werte $P_k(0)$. Die Vorgehensweise entspricht der von Beispiel 3.1.

h_k	$t_k = h_k^2$	Newton-Schema für t_k und $B(t_k)$			
0.30	0.09	0.6414428889	0.05328577800	0.001797515344	0.001356560849
0.20	0.04	0.6387786000	0.05316444571	0.001688990476	
0.15	0.0225	0.6378482222	0.05311377600		
0.10	0.01	0.6371843000			

In der oberen Schräzeile stehen die Koeffizienten c_j , $j = 0, 1, 2, 3$, der Polynome $P_k(t)$, $k = 0, 1, 2, 3$. Es ist $f''(0.6) = \sinh(0.6) = 0.6366535821482413$. Damit ergibt sich

k	1	2	3
$P_k(0)$	0.6366471689	0.6366536399	0.6366535300
$ P_k(0) - \sinh(0) $	$6.4133 \cdot 10^{-6}$	$5.7787 \cdot 10^{-8}$	$5.2094 \cdot 10^{-8}$

Alle extrapolierten Werte $P_k(0)$ sind bedeutend bessere Näherungen als die Ausgangsdaten $B(t_k)$. Im letzten Wert $P_3(0)$ macht sich bereits die begrenzte Stellenzahl bei der Berechnung der y -Werte bemerkbar. \triangle

3.2 Splines

Die Polynominterpolation ist ein klassisches Verfahren für Aufgaben kleinen Umfangs. Für zu große Polynomgrade weisen die interpolierenden Polynome Oszillationen auf, sie werden unbrauchbar für den Zweck der Interpolation. Die Interpolationsqualität kann auch durch das Aneinanderstückeln einer einfachen Vorschrift verbessert werden. Das entspricht der üblichen Vorgehensweise bei der Bestimmung eines Funktionswertes mit Hilfe einer ‘‘Logarithmentafel’’. Ist man aber nicht nur an einzelnen Werten, sondern an der approximierenden Funktion interessiert, so fällt negativ auf, dass diese an den Stückelungsstellen Knicke aufweist; sie ist dort nicht differenzierbar, nicht *glatt*. Ein Ausweg aus dieser Zwickmühle sind die *Splinefunktionen*, kurz Splines genannt.

In ihrer allgemeinen Definition erfüllen sie drei Bedingungen:

- Sie sind stückweise Polynome vom Höchstgrad k .
- Sie sind p mal stetig differenzierbar.
- Unter den Funktionen, die die ersten beiden Bedingungen erfüllen und eine gegebene Tabelle interpolieren, sind sie die ‘‘glattesten’’ (s.u.).

Am häufigsten werden die kubischen Splines mit $k = 3$ und $p = 2$ benutzt.

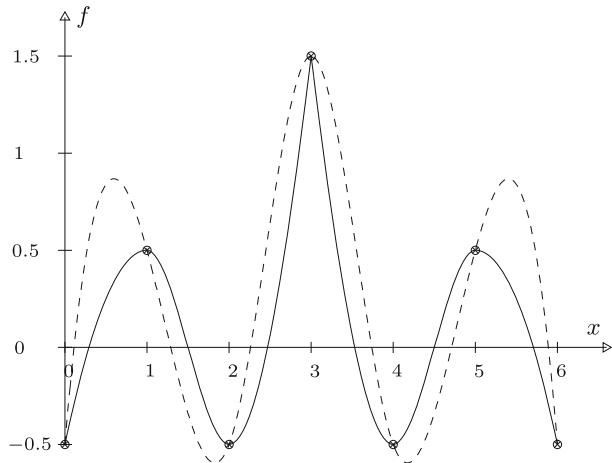


Abb. 3.6 Stückweise Interpolation: Zwei Polynome (—) oder ein Spline (- -).

3.2.1 Kubische Splines

Physikalisch: Schon zu Beginn des 20. Jahrhunderts benutzte man im Schiffs- und etwas später im Flugzeugbau so genannte Straklatten (dünne Balsaholzstäbe, engl. splines), um glatte Flächen für den Schiffs- oder Flugzeugkörper oder die Flügelform zu konstruieren. An vorgegebenen (Interpolations-) Punkten werden die Straklatten fixiert. Auf Grund des natürlichen Energieminimierungsprinzips ist dann die Form der Latte praktisch identisch mit der Kurve kleinsten Krümmung, die diese Punkte interpoliert.

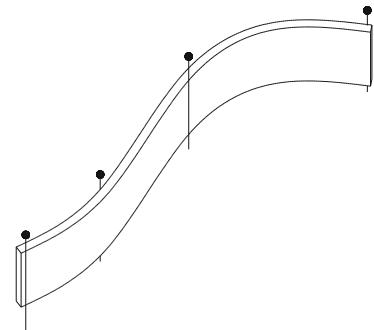


Abb. 3.7
Straklatte = Spline.

Wir wollen diese physikalische Aufgabe mathematisch formulieren:

Gegeben seien $n + 1$ voneinander verschiedene Stützstellen im Intervall $[a, b]$

$$a = x_0 < x_1 < \dots < x_n = b \quad (3.35)$$

und diesen zugeordnete Stützwerte y_0, y_1, \dots, y_n .

Gesucht ist eine Funktion s , die folgende Forderungen erfüllt:

(i) s interpoliert die gegebene Tabelle:

$$s(x_i) = y_i \quad i = 0, 1, \dots, n. \quad (3.36)$$

(ii) s ist im Intervall $[a, b]$ mindestens einmal stetig differenzierbar.

(iii) Im Innern jeden Teilintervalls (x_i, x_{i+1}) , $i = 0, 1, \dots, n - 1$, ist s mindestens viermal stetig differenzierbar. Für die Ableitungen existieren links- und rechtsseitige Grenzwerte an den Stützstellen x_i , die aber nicht notwendig übereinstimmen müssen.

(iv) s minimiert das Funktional

$$J[s] := \frac{1}{2} \int_a^b s''(x)^2 dx. \quad (3.37)$$

Dies entspricht für kleine Krümmungen der Minimierung der Energie, die die Straklatte zum Verbiegen verbraucht und ist damit eine natürliche Bedingung im Gegensatz z.B. zu der auch vorstellbaren Minimierung von $\int_a^b |s''(x)| dx$.

Satz 3.5. Es existiert genau eine Funktion s , die die Bedingungen (i)–(iv) erfüllt.

Wir folgen dem schönen Beweis in [Hen 82], der die klassische Variationsrechnung benutzt. Sie liefert die notwendigen Bedingungen, die s als Lösung erfüllen muss. Die Eindeutigkeit folgt dann aus der Eindeutigkeit der Hermite-Interpolation, siehe 3.1.4, weil die Forderungen zu eindeutig festgelegten Ableitungen in den Stützstellen führen. Algorithmisch wollen wir uns auf die Konstruktion von Splinefunktion aus Basisfunktionen, den so genannten B-Splines, beschränken, siehe die Abschnitte 3.2.2 und 3.2.3.

Beweis. Zunächst wollen wir *voraussetzen*, dass eine Funktion s existiert, die die Forderungen (i)–(iv) erfüllt. Die Variationstechnik, die auf Euler zurückgeht, liefert damit andere Bedingungen, die s eindeutig charakterisieren. Damit konstruieren wir ein s und zeigen, dass es die Forderungen (i)–(iv) erfüllt.

Sei s also eine Funktion, die (i)–(iv) erfüllt. Sei s_1 eine andere Funktion, die (i)–(iii) erfüllt. Dann gilt für s_1

$$J[s] \leq J[s_1]. \quad (3.38)$$

Für s_1 wählen wir die spezielle Form

$$s_1(x) = s(x) + \varepsilon h(x). \quad (3.39)$$

Dabei ist ε ein reeller Parameter, und weil s_1 die Bedingungen (i)–(iii) erfüllt, gelten für h die Eigenschaften (ii), (iii) und statt (i) die Gleichungen

$$h(x_i) = 0, \quad i = 0, 1, \dots, n. \quad (3.40)$$

Jedes h , das diese Bedingungen erfüllt, heißt *zulässig*, weil es ein s_1 liefert, das (i)–(iii) erfüllt. Sei nun

$$J(\varepsilon) := J[s_1] = \frac{1}{2} \int_a^b (s''(x) + \varepsilon h''(x))^2 dx. \quad (3.41)$$

$J(\varepsilon)$ ist ein quadratisches Polynom in ε . Wegen (3.38) nimmt dieses Polynom sein Minimum für $\varepsilon = 0$ an. Eine notwendige Bedingung dafür ist, dass

$$J'(0) = \frac{dJ}{d\varepsilon} \Big|_{\varepsilon=0} = 0 \quad \text{für jede zulässige Funktion } h. \quad (3.42)$$

Es ist leicht nachzurechnen, dass

$$J'(0) = \int_a^b s''(x)h''(x) dx. \quad (3.43)$$

Dieser Ausdruck wird zweimal partiell integriert, was wegen (iii) getrennt in jedem Teilintervall geschehen muss. Es sind

$$\begin{aligned} \int_{x_i}^{x_{i+1}} s''(x)h''(x) dx &= s''(x)h'(x) \Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} s^{(3)}(x)h'(x) dx, \\ \int_{x_i}^{x_{i+1}} s^{(3)}(x)h'(x) dx &= s^{(3)}(x)h(x) \Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} s^{(4)}(x)h(x) dx. \end{aligned}$$

Beachten wir jetzt, dass $h(x_i) = 0$ an allen Stützstellen und summieren alle Teilsummen auf, dann bekommen wir

$$\begin{aligned} J'(0) &= -s''(x_0+)h'(x_0) + (s''(x_1-) - s''(x_1+))h'(x_1) + \\ &\quad (s''(x_2-) - s''(x_2+))h'(x_2) + \cdots + s''(x_n-)h'(x_n) \\ &\quad + \int_a^b s^{(4)}(x)h(x) dx. \end{aligned} \quad (3.44)$$

Dabei sind $s''(x_i+)$ bzw. $s''(x_i-)$ die Grenzwerte von s'' , wenn x von rechts bzw. von links gegen x_i läuft.

Da h eine beliebige zulässige Funktion sein kann, folgen aus $J'(0) = 0$ zwei Bedingungen, erstens

$$s^{(4)}(x) = 0 \quad \text{für alle } x \neq x_0, x_1, \dots, x_n. \quad (3.45)$$

Wäre dies nicht so, wäre also z.B. $s^{(4)}(\xi) > 0$ an einem inneren Punkt eines Teilintervalls, dann könnten wir ein zulässiges h finden mit $h(\xi) > 0$ nur in einer kleiner Umgebung des Punktes ξ . Dann würde nur das entsprechende Teilintervall einen positiven Anteil im Integral in (3.44) liefern und damit wäre $J'(0) > 0$ im Widerspruch zu (3.42).

Zweitens folgt

$$s''(x_0+) = s''(x_n-) = 0 \quad \text{und} \quad (3.46)$$

$$s''(x_i-) = s''(x_i+), \quad i = 1, 2, \dots, n-1; \quad (3.47)$$

d.h. s'' ist stetig und verschwindet an den Endpunkten¹. Wäre dies nicht so, wäre also z.B. $s''(x_k-) - s''(x_k+) \neq 0$ für irgendein k , dann gäbe es ein zulässiges h mit $h'(x_i) = 0$ an allen Punkten außer für x_k , was wiederum $J'(0) \neq 0$ zur Folge hätte im Widerspruch zu (3.42).

Aus der ersten Bedingung (3.45) folgt, dass s in jedem Teilintervall $[x_i, x_{i+1}]$ als Polynom dritten Grades $s_i(x)$ dargestellt werden kann. Aus (i), (ii) und (3.47) folgt, dass diese Polynome sowie ihre ersten und zweiten Ableitungen an den inneren Stützstellen stetig ineinander übergehen:

$$\left. \begin{array}{lcl} s_i(x_i) & = & s_{i-1}(x_i), \\ s'_i(x_i) & = & s'_{i-1}(x_i), \\ s''_i(x_i) & = & s''_{i-1}(x_i), \end{array} \right\} \quad i = 1, 2, \dots, n-1. \quad (3.48)$$

Außerdem folgt aus (3.47)

$$s''_0(x_0) = s''_{n-1}(x_n) = 0. \quad (3.49)$$

Damit sind die Eigenschaften der Splinefunktion s hergeleitet. Jetzt ist noch zu zeigen, dass genau eine solche Funktion s bzw. eine Menge kubischer Polynome $\{s_0, s_1, \dots, s_{n-1}\}$ existiert, die diese Eigenschaften besitzen. Da ein Polynom dritten Grades vier Koeffizienten besitzt, liegen vier Freiheitsgrade pro Teilintervall vor, die durch die Bedingungen benutzt werden können, um s_i eindeutig festzulegen. Zwei Bedingungen sind die aus Forderung (i):

$$s_i(x_i) = y_i, \quad s_i(x_{i+1}) = y_{i+1}. \quad (3.50)$$

Um zwei weitere Bedingungen zu formulieren, führen wir die Größen

$$c_i := s'(x_i) \quad (3.51)$$

ein. Sie stellen zunächst unsere Unbekannten dar und liefern die beiden Bedingungen

$$s'_i(x_i) = c_i, \quad s'_i(x_{i+1}) = c_{i+1}. \quad (3.52)$$

Wir transformieren jetzt mit

$$t := \frac{x - x_i}{h_i}, \quad h_i := x_{i+1} - x_i \quad (3.53)$$

das Teilintervall $[x_i, x_{i+1}]$ auf das Intervall $[0, 1]$ und damit s_i auf

$$q_i(t) = s_i(x) = s_i(x_i + h_i t). \quad (3.54)$$

¹Für die Straklatte ist dies ohnehin klar: Wenn sie sich an den Enden frei verbiegen kann, wird sie das ungekrümmt tun.

Auch die q_i sind kubische Polynome und unsere vier Bedingungen werden zu

$$q_i(0) = y_i, \quad q_i(1) = y_{i+1}, \quad (3.55)$$

$$q'_i(0) = h_i c_i, \quad q'_i(1) = h_i c_{i+1}. \quad (3.56)$$

Dies sind die Daten, die eine Hermite-Interpolation an zwei Punkten erfordert, siehe Abschnitt 3.1.4. Es ist leicht zu sehen, dass die Lösung das kubische Polynom

$$\begin{aligned} q_i(t) = & (1-t)^2 y_i + t^2 y_{i+1} + \\ & t(1-t) \{(1-t)(2y_i + h_i c_i) + t(2y_{i+1} - h_i c_{i+1})\} \end{aligned} \quad (3.57)$$

ist. Durch die Interpolation ist die Stetigkeit der Funktion s und ihrer ersten Ableitung s' gesichert. Deshalb betrachten wir noch die zweite Ableitung an den Intervallenden

$$q''_i(0) = 6(y_{i+1} - y_i) - 2h_i(2c_i + c_{i+1}), \quad (3.58)$$

$$q''_i(1) = -6(y_{i+1} - y_i) + 2h_i(c_i + 2c_{i+1}). \quad (3.59)$$

Mit $s''_i(x) = q''_i(t)(dt/dx)^2 = q''_i(t)h_i^{-2}$ ergeben sich für das Verschwinden der zweiten Ableitung von s an den Endpunkten und für ihre Stetigkeit an den inneren Stützstellen die Bedingungen

$$(2c_0 + c_1)/h_0 = 3(y_1 - y_0)/h_0^2, \quad (3.60)$$

$$(c_{n-1} + 2c_n)/h_{n-1} = 3(y_n - y_{n-1})/h_{n-1}^2,$$

und für $i = 1, \dots, n-1$:

$$(c_{i-1} + 2c_i)/h_{i-1} + (2c_i + c_{i+1})/h_i = 3(y_i - y_{i-1})/h_{i-1}^2 + 3(y_{i+1} - y_i)/h_i^2.$$

Diese Beziehungen stellen ein System von $n+1$ linearen Gleichungen für die $n+1$ Unbekannten c_0, c_1, \dots, c_n dar. Seine Koeffizientenmatrix

$$\left(\begin{array}{cccccc} \frac{2}{h_0} & \frac{1}{h_0} & & & & & 0 \\ \frac{1}{h_0} & \frac{2}{h_0} + \frac{2}{h_1} & \frac{1}{h_1} & & & & \\ & \frac{1}{h_1} & \frac{2}{h_1} + \frac{2}{h_2} & \frac{1}{h_2} & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \frac{1}{h_{n-2}} & \frac{2}{h_{n-2}} + \frac{2}{h_{n-1}} & \frac{1}{h_{n-1}} & & \\ 0 & & & \frac{1}{h_{n-1}} & \frac{2}{h_{n-1}} & & \end{array} \right) \quad (3.61)$$

ist symmetrisch und diagonal dominant. Folglich ist sie nach dem Kreise-Satz von Gershgorin 5.17 nicht singulär, sie ist sogar positiv definit. Also hat das Gleichungssystem (3.60)

eine eindeutige Lösung. Daraus folgt, dass es genau eine Funktion s gibt, die die Forderungen (i)–(iv) erfüllt. In jedem Teilintervall $[x_i, x_{i+1}]$ wird diese Funktion durch ein kubisches Polynom $s_i(x)$ repräsentiert. \square

Die Funktion s heißt *natürliche Spline-Interpolierende*. Die durch die Minimierung des Funktionalen (3.37) entstehenden *natürlichen* Randbedingungen sind nicht für jede Anwendung der Spline-Interpolation passend. Sie können leicht durch zwei andere Bedingungen ersetzt werden. Damit ergeben sich die drei am häufigsten benutzten Formen:

$$\begin{aligned} \text{Natürlicher Spline: } & s''(x_0) = 0 & s''(x_n) = 0, \\ \text{Vollständiger Spline: } & s'(x_0) = y'_0, & s'(x_n) = y'_n, \\ \text{Periodischer Spline: } & s'(x_0) = s'(x_n) & s''(x_0) = s''(x_n). \end{aligned}$$

Die zusätzlichen Randbedingungen können vermieden werden, wenn als Basis einer Interpolations- oder Approximationsaufgabe die so genannten B-Splines gewählt werden, mit denen wir uns in den nächsten Abschnitten beschäftigen wollen. Sie eignen sich als Funktionensystem und durch die Möglichkeit rekursiver Konstruktion auch besser für die algorithmische Darstellung der Lösung von Spline-Aufgaben wie Interpolation oder Approximation.

3.2.2 B-Splines 1. Grades

Die beste algorithmisch zu verarbeitende Form einer Splinefunktion ist ihr Aufbau aus sehr einfachen Basisfunktionen, den *B-Splines*. Leider ist deren allgemeine Darstellung etwas komplex und nicht sehr anschaulich. Wir wollen deshalb den Begriff der B-Splines an einem Beispiel mit wenigen Stützstellen und in ihrer einfachsten Form kennen lernen, den B-Splines 1. Grades, die anschaulicher auch Hutfunktionen genannt werden. Es handelt sich um stetige, stückweise lineare Funktionen.

Beispiel 3.6. Zu den Stützstellen

$$x_0 < x_1 < x_2 < x_3 \tag{3.62}$$

suchen wir die vier Funktionen, die

- in jedem der Intervalle $[x_0, x_1]$, $[x_1, x_2]$ und $[x_2, x_3]$ lineare Polynome sind,
- in genau einer der vier Stützstellen den Wert 1 annehmen, und
- in den anderen Stützstellen verschwinden.

Offensichtlich sind das die vier Funktionen der Abb. 3.8.

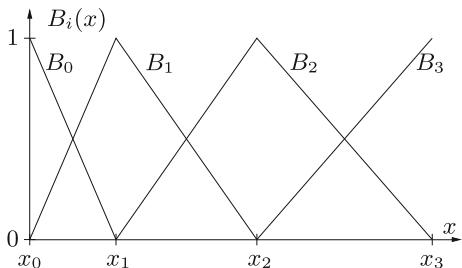


Abb. 3.8
Lineare B-Splines für vier Stützstellen.

Diese vier einfachen Funktionen wollen wir jetzt genauso (kompliziert) herleiten, wie das bei allgemeinen B-Splines getan wird. Zunächst werden die B-Splines nicht über der Menge der Stützstellen (auf der man z.B. interpolieren will), erklärt, sondern über einer Knotenpunktmenge, die wir hier für die Interpolation speziell festlegen wollen als

$$t_0 := t_1 := x_0, \quad t_2 := x_1, \quad t_3 := x_2, \quad t_4 := t_5 := x_3. \quad (3.63)$$

Außerdem definieren wir die abgeschnittene Potenzfunktion 1. Grades

$$F_x(t) := \begin{cases} (t - x) & \text{falls } t \geq x \\ 0 & \text{falls } t < x \end{cases} \quad (3.64)$$

$F_x(t)$ ist für jedes (feste) x eine Funktion von t . Ihre Werte in den Knotenpunkten t_i seien die Stützwerte $f_i := F_x(t_i)$. Die B-Splines werden jetzt mit Hilfe der dividierten Differenzen definiert, die wir in Abschnitt 3.1.3 kennen gelernt haben. Wir wollen das nur für B_0 genauer aufschreiben:

$$B_0(x) := (t_2 - t_0)[f_0, f_1, f_2]. \quad (3.65)$$

Hier sind noch zwei Besonderheiten zu beachten. Einmal sind die dividierten Differenzen ja noch Funktionen von x , denn sie werden mit t -Werten gebildet, die f_i hängen aber noch von x ab, also auch die B_i . Zum anderen gibt es wieder gleiche Stützstellen und Werte im Newton-Schema, die wie bei der Hermite-Interpolation in Abschnitt 3.1.4 durch Ableitungswerte ersetzt werden. Für B_0 ergibt sich folgendes Newton-Schema:

t_i	$F_x(t_i)$
x_0	0
x_0	0
x_1	$x_1 - x$

x_0	0	$\frac{x_1 - x}{x_1 - x_0}$
x_1	$x_1 - x$	$\frac{x_1 - x}{x_1 - x_0}$

Also ist

$$B_0(x) = \begin{cases} \frac{x_1 - x}{x_1 - x_0} & \text{falls } x \in [x_0, x_1] \\ 0 & \text{sonst} \end{cases}. \quad (3.66)$$

Noch umständlicher wird die Bestimmung von

$$B_1(x) := (t_3 - t_1)[f_1, f_2, f_3],$$

weil man jetzt wegen der Fallunterscheidung in der Definition der abgeschnittenen Potenzfunktion zwei Newton-Schemata braucht. Anschaulich ergibt sich ja ganz leicht, dass

$$B_1(x) = \begin{cases} 1 - \frac{x_1 - x}{x_1 - x_0} = \frac{x - x_0}{x_1 - x_0} & \text{falls } x \in [x_0, x_1] \\ \frac{x_2 - x}{x_2 - x_1} & \text{falls } x \in [x_1, x_2] \\ 0 & \text{sonst} \end{cases}$$

sein muss. Die komplizierte Definition ist nur für allgemeine Aufgaben von Vorteil, wie wir noch sehen werden. \triangle

Hat man die B-Splines für gegebene Stützstellen und für eine Knotenpunktmenge einmal berechnet, so lässt sich die folgende Interpolationsaufgabe leicht lösen:

Bestimme Koeffizienten α_i , $i = 0, \dots, n$, so, dass

$$\sum_{i=0}^n \alpha_i B_i(x_k) = y_k, \quad k = 0, 1, \dots, n.$$

Wegen

$$B_i(x_k) = \begin{cases} 1 & \text{falls } i = k \\ 0 & \text{falls } i \neq k \end{cases}$$

ist einfach (wie bei der Lagrange-Interpolation, siehe Abschnitt 3.1.2)

$$\alpha_i := y_i .$$

3.2.3 Kubische B-Splines

B-Splines höherer Ordnung sind von Hand kaum noch zu berechnen. Sie eignen sich aber um so besser für die Computer-Berechnung, weil sie sich aus B-Splines niedrigerer Ordnung rekursiv berechnen lassen. Für zweimal stetig differenzierbare Splinefunktionen dritten Grades begnügen wir uns damit ihre rekursive Definition anzugeben und ihre wesentlichen Eigenschaften ohne Beweise zusammenzustellen. Dazu definieren wir zunächst wieder mit Hilfe der Stützstellenmenge $\{x_i\}$ eine spezielle Knotenpunktmenge $\{t_i\}$:

$$\begin{aligned} \{t_0, t_1, t_2, t_3, t_4, t_5, \dots, t_n, t_{n+1}, t_{n+2}, t_{n+3}, t_{n+4}\} &:= \\ \{x_0, x_0, x_0, x_0, x_2, x_3, \dots, x_{n-2}, x_n, x_n, x_n, x_n\}. \end{aligned}$$

Für diese Knotenpunkte t_0, t_1, \dots, t_{n+4} definieren wir die B-Splines 0. Grades

$$B_{i,0}(x) = \begin{cases} 1 & \text{falls } t_i \leq x < t_{i+1} \\ 1 & \text{falls } x = t_{i+1} = x_n \\ 0 & \text{sonst} \end{cases} \quad i = 0, 1, \dots, n+3. \quad (3.67)$$

Es ist also $B_{i,0}(x) \equiv 0$, falls $t_i = t_{i+1}$. Die B-Splines höheren Grades ergeben sich aus denen 0. Grades rekursiv:

$$B_{i,k}(x) =^* \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x). \quad (3.68)$$

($=^*$ bedeutet, dass Terme mit Nenner null weggelassen werden.)

Aus der oben angegebenen Knotenpunktmenge und den B-Splines 0. Grades lassen sich also folgende B-Splines rekursiv berechnen

$$\begin{array}{ccccccccc} B_{0,0} & B_{1,0} & B_{2,0} & \dots & B_{n,0} & B_{n+1,0} & B_{n+2,0} & B_{n+3,0} \\ \downarrow & \swarrow & \downarrow & \swarrow & \dots & \downarrow & \swarrow & \downarrow & \swarrow \\ B_{0,1} & B_{1,1} & \dots & \dots & B_{n,1} & B_{n+1,1} & B_{n+2,1} & & \\ \downarrow & \swarrow & \downarrow & \dots & \dots & \downarrow & \swarrow & \downarrow & \swarrow \\ B_{0,2} & B_{1,2} & \dots & \dots & B_{n,2} & B_{n+1,2} & & & \\ \downarrow & \swarrow & \downarrow & \dots & \dots & \downarrow & \swarrow & & \\ B_{0,3} & B_{1,3} & \dots & \dots & B_{n,3} & & & & \end{array} \quad (3.69)$$

Zu einem Punkt $x \in [t_i, t_{i+1}]$ mit $t_i < t_{i+1}$ benötigt man zur Berechnung einer Splinesumme mit den kubischen B-Splines alle B-Spline-Werte $B_{i,k}(x)$ für $k = 0, 1, 2, 3$, die in x einen Wert ungleich null haben. Diese kann man in einem Dreiecksschema anordnen und mit rekursiven Algorithmen elegant berechnen:

$$\begin{matrix} & & & 0 \\ & & 0 & \\ 0 & 0 & B_{i-2,2} & B_{i-2,3} \\ B_{i,0} & B_{i-1,1} & B_{i-1,2} & B_{i-1,3} \\ 0 & B_{i,1} & B_{i,2} & B_{i,3} \\ 0 & 0 & 0 & \\ & & & 0 \end{matrix} \quad (3.70)$$

Die B-Splines werden also nur in den entsprechenden Teilintervallen berechnet.

Da die Konstruktion der B-Splines nicht besonders anschaulich ist, wollen wir die Funktionen durch ihre Eigenschaften näher charakterisieren. Die B-Splines $B_i(x) := B_{i,3}(x)$ erfüllen folgende Bedingungen :

1. Sie haben kleinstmögliche Trägerintervalle:

$$\begin{aligned} B_0(x) > 0 & \text{ falls } x \in [x_0, x_2], \\ B_1(x) > 0 & \text{ falls } x \in (x_0, x_3), \\ B_2(x) > 0 & \text{ falls } x \in (x_0, x_4), \\ B_3(x) > 0 & \text{ falls } x \in (x_0, x_5), \\ B_i(x) > 0 & \text{ falls } x \in (x_{i-2}, x_{i+2}), \quad i = 4, \dots, n-4, \\ B_{n-3}(x) > 0 & \text{ falls } x \in (x_{n-5}, x_n), \\ B_{n-2}(x) > 0 & \text{ falls } x \in (x_{n-4}, x_n), \\ B_{n-1}(x) > 0 & \text{ falls } x \in (x_{n-3}, x_n), \\ B_n(x) > 0 & \text{ falls } x \in (x_{n-2}, x_n). \end{aligned} \quad (3.71)$$

Außerhalb dieser Intervalle sind die B-Splines identisch null.

$$2. \quad \sum_{i=0}^n B_i(x) = 1 \quad \forall x \in [x_0, x_n]$$

3. Die Matrix B mit den Koeffizienten

$$b_{ik} := B_k(x_i) \quad (3.72)$$

hat Fünfbandgestalt und ist regulär. Oft ist sie symmetrisch und dann auch positiv definit. Sie kann (bis zu) Siebenbandgestalt haben, wenn die Knotenpunkte t_i beliebig, d.h. unabhängig von den Stützstellen x_i gewählt werden.

4. Die Werte der B-Splines wie die ihrer Ableitungen und ihrer Integrale lassen sich rekursiv und numerisch stabil berechnen.

Auf Grund dieser positiven Eigenschaften eignen sich B-Splines auch gut als Ansatzfunktionen bei der Lösung von Differenzial- und Integralgleichungen.

Für äquidistante Stützstellen wollen wir die inneren kubischen B-Splines angeben. Sei also

$$x_i := x_0 + ih, \quad i = 0, 1, \dots, n.$$

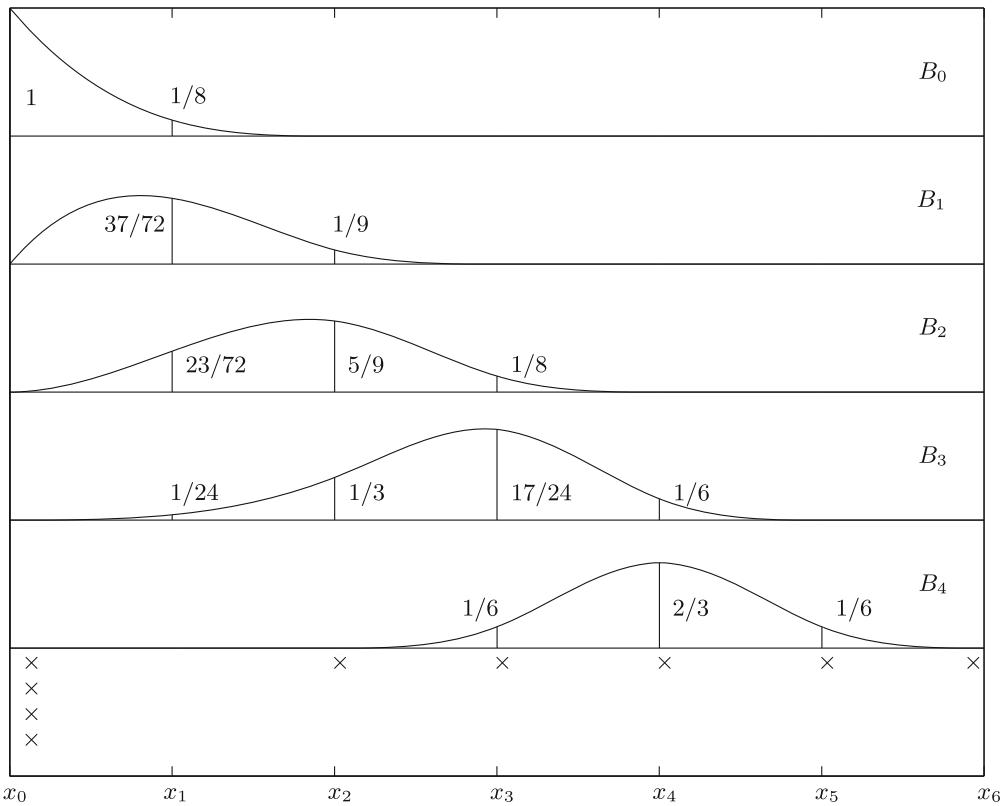


Abb. 3.9 Kubische B-Splines bei äquidistanten Stützstellen.

Dann ist für $i = 4, 5, \dots, n - 4$:

$$B_i(x) = \frac{1}{6h^3} \cdot \begin{cases} (x - x_{i-2})^3, & x \in [x_{i-2}, x_{i-1}], \\ h^3 + 3h^2(x - x_{i-1}) + 3h(x - x_{i-1})^2 - 3(x - x_{i-1})^3, & x \in [x_{i-1}, x_i], \\ h^3 + 3h^2(x_{i+1} - x) + 3h(x_{i+1} - x)^2 - 3(x_{i+1} - x)^3, & x \in [x_i, x_{i+1}], \\ (x_{i+2} - x)^3, & x \in [x_{i+1}, x_{i+2}], \\ 0, & \text{sonst.} \end{cases}$$

In Abb. 3.9 sind die ersten fünf kubischen B-Splines bei mindestens neun Stützstellen dargestellt. Die Knotenpunkte sind mit \times gekennzeichnet. Der einzige “innere” B-Spline mit einem Träger von vier Intervallen und symmetrischem Verlauf ist B_4 . Diese inneren B-Splines sind interpolierende natürliche kubische Splines mit äquidistanten Stützstellen $\{x_{i-2}, \dots, x_{i+2}\}$ zu der Wertetabelle

$$y_i = \{0, 1/6, 2/3, 1/6, 0\}.$$

Für die Konstruktion der B-Splines haben wir bisher die Knotenpunkte sehr speziell festgelegt. Werden die Knotenpunkte freier festgelegt, so wird eine große Flexibilität bei der Definition der Basisfunktionen erreicht. Diese Flexibilität soll noch an einem gegenüber dem allgemeinen Fall stark vereinfachten Lemma und an einer Beispiel-Zeichnung deutlich gemacht werden:

Lemma 3.6. *Ist t_j ein m -facher innerer Knoten, dann ist ein kubischer B-Spline B_i an der Stelle t_j mindestens $(3 - m)$ mal stetig differenzierbar.*

Durch diese Eigenschaft können Splines mit gewünschten Ecken oder Sprüngen konstruiert werden, wie Abb. 3.10 zeigt.

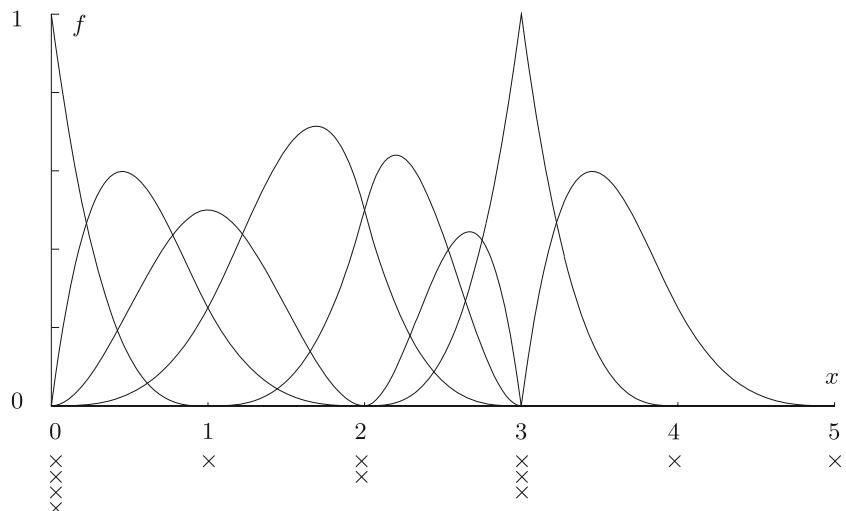


Abb. 3.10 Kubische B-Splines mit mehrfachen inneren Knoten.

Interpolation

Die gesuchte Splinefunktion $s(x)$ wird als Linearkombination der B-Splines dargestellt:

$$s(x) = \sum_{k=0}^n \alpha_k B_k(x). \quad (3.73)$$

Dies ist ein Ansatz mit $n+1$ Freiheitsgraden. Für $n+1$ Interpolationsbedingungen ergibt sich daher eine eindeutige Lösung. Zusätzliche Bedingungen am Rand sind nicht zu erfüllen. Das ist ein Vorteil des B-Spline-Ansatzes gegenüber der üblichen kubischen Spline-Interpolation (Abschnitt 3.2.1).

Durch Lösung des linearen Gleichungssystems $B\alpha = y$ werden die unbekannten Koeffizienten α_k bestimmt; α und y sind die Vektoren mit den α_k bzw. y_i als Komponenten, also:

$$\sum_{k=0}^n \alpha_k B_k(x_i) = y_i, \quad i = 0, 1, \dots, n. \quad (3.74)$$

Diese Lösung ist auf Grund der oben genannten Eigenschaften stabil und schnell möglich. Weitere Einzelheiten zu B-Splines findet man in [Boo 01].

Approximation

Sollen m Stützwerte ausgeglichen werden mit $m > n + 1$, so ist eine diskrete Gauß-Approximation durchzuführen, siehe Abschnitt 3.6. Es ergibt sich das entsprechende Gleichungssystem $B\alpha = y$, wo allerdings B jetzt eine rechteckige Matrix ist. Dieses wird mit der Methode der kleinsten Quadrate, siehe Kapitel 6, gelöst:

$$\sum_{i=0}^m \left(\sum_{k=0}^n \alpha_k B_k(x_i) - y_i \right)^2 \rightarrow \min_{\alpha_i} \quad (3.75)$$

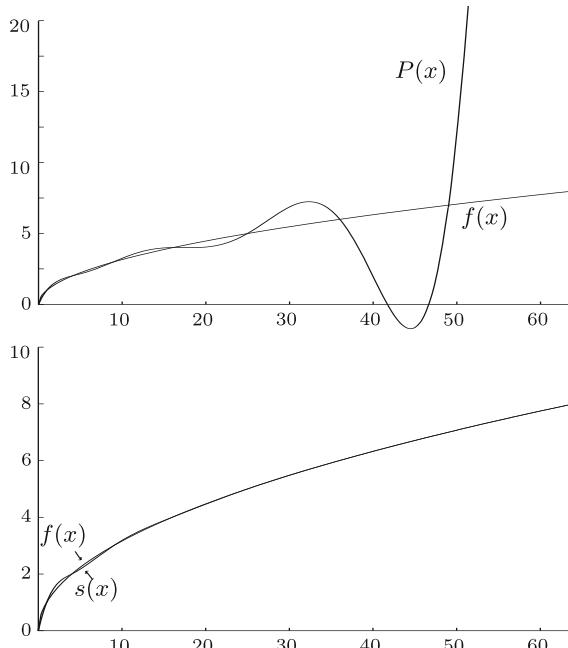


Abb. 3.11 Vergleich der Interpolationsmethoden.

Tab. 3.2 Werte im Vergleich.

x	$P(x)$	$s(x)$	$f(x) = \sqrt{x}$
0.1	0.128820	0.124346	0.316227
0.3	0.364987	0.355691	0.547722
0.5	0.574768	0.564967	0.707106
0.7	0.760697	0.753454	0.836660
0.9	0.925141	0.922429	0.948683
51.0	18.501183	7.141938	7.141428
53.0	35.466596	7.281021	7.280109
55.0	56.773501	7.417362	7.416198
57.0	78.917289	7.551075	7.549834
59.0	94.778908	7.682272	7.681145
61.0	92.051716	7.811066	7.810249
63.0	51.276268	7.937570	7.937253

Beispiel 3.7. Die Interpolation mit Polynomen oder mit Splines soll an einem Beispiel verglichen werden. Die Funktion

$$f(x) = \sqrt{x}$$

soll im Intervall $[0, 64]$ an folgenden Stellen interpoliert werden:

x	0	1	4	9	16	25	36	49	64
y	0	1	2	3	4	5	6	7	8

Natürlich ist dies ein sehr künstliches Beispiel, aber es verdeutlicht den Qualitätsunterschied der betrachteten Interpolationsverfahren bei sehr schwach variierenden Tabellenwerten sehr schön.

Uns interessieren besonders die Werte der Interpolationsfunktionen in den Randintervallen $(0, 1)$ und $(49, 64)$. In Tab. 3.2 stellen wir die Werte der Funktion $f(x) = \sqrt{x}$ den Werten aus der Polynom- und der Splineinterpolation gegenüber. In Abb. 3.11 finden wir oben die Funktion f und das interpolierende Polynom P . P oszilliert stark und verschwindet bei $x \approx 50$ sogar aus der Zeichnung, weil es Werte bis zu 100 annimmt. Im unteren Teil sind f und die interpolierende Splinefunktion s eingezeichnet. Sie stimmen bis auf ganz leichte Oszillationen von s am Anfang des Intervalls gut überein. \triangle

3.3 Zweidimensionale Splineverfahren

Wir betrachten zwei Problemstellungen:

Interpolation

Gegeben sei eine Tabelle mit Daten

$$(x_k, y_k, f_k) \quad k = 1, 2, \dots, m. \quad (3.76)$$

Gesucht ist eine Funktion

$$S(x, y) := \sum_{\nu=1}^n \sum_{\mu=1}^l c_{\nu\mu} s_{\nu}(x) t_{\mu}(y) \quad \text{mit } l n = m \text{ und} \quad (3.77)$$

$$S(x_k, y_k) = f_k, \quad k = 1, 2, \dots, m.$$

Dabei sind die Funktionen s_{ν} und t_{μ} die eindimensionalen Ansatzfunktionen.

Approximation

Gegeben sei eine Tabelle mit Daten

$$(x_k, y_k, f_k) \quad k = 1, 2, \dots, m. \quad (3.78)$$

Gesucht ist eine Funktion

$$S(x, y) := \sum_{\nu=1}^n \sum_{\mu=1}^l c_{\nu\mu} s_{\nu}(x) t_{\mu}(y) \quad \text{mit } l n < m \text{ und} \quad (3.79)$$

$$\sum_{k=1}^m (S(x_k, y_k) - f_k)^2 \rightarrow \min_{c_{\nu\mu}}.$$

Bei der Interpolation wird die Lösung stark vereinfacht, wenn man sich auf die Gitterpunkte eines Rechteckgitters beschränkt:

Gegeben seien zwei Koordinatenlisten und eine Liste mit Funktionswerten:

$$\begin{aligned} x_i, \quad i = 1, 2, \dots, n \quad \text{und} \quad y_j, \quad j = 1, 2, \dots, l \\ f_{ij}, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, l. \end{aligned} \quad (3.80)$$

Gesucht ist eine Funktion

$$S(x, y) := \sum_{\nu=1}^n \sum_{\mu=1}^l c_{\nu\mu} s_{\nu}(x) t_{\mu}(y) \quad (3.81)$$

$$\begin{aligned} \text{mit } S(x_i, y_j) &= f_{ij}, \\ i = 1, 2, \dots, n, \quad j &= 1, 2, \dots, l. \end{aligned} \quad (3.82)$$

In den folgenden Unterabschnitten wollen wir für bilineare und bikubische TensorSplines zwei Konstruktionsprinzipien kennen lernen.

3.3.1 Bilineare TensorSplines

Hier wollen wir Flächen im \mathbb{R}^3 durch Produkte stückweise linearer Polynome konstruieren (nach [Loc 93]). Dieses Prinzip lässt sich auf bikubische Splines übertragen, die wir allerdings nach einem etwas anderen Konstruktionsprinzip im nächsten Unterabschnitt herleiten wollen.

Sei eine Funktion $f : Q_k \rightarrow \mathbb{R}$ gegeben. Dabei ist $Q_k = \{(x, y) | -k \leq x, y \leq k\}$ ein Quadrat mit der Seitenlänge $2k$.

Wir bilden jetzt mit der Grundfunktion

$$B_1(t) := \begin{cases} 1+t & -1 \leq t < 0 \\ 1-t & 0 \leq t < 1 \\ 0 & \text{sonst} \end{cases}$$

die zweidimensionale Produktfunktion

$$D_{11}(x, y) := B_1(x)B_1(y), \quad (x, y) \in \mathbb{R}^2,$$

die ungleich null nur für $-1 < x, y < 1$ ist. Hält man eine Variable fest, so entsteht eine stückweise lineare Funktion der anderen Variablen. Insgesamt erhält man also eine auf der Ebene $z = 0$ aufsitzende Pyramide. Die Seitenflächen sind allerdings nicht eben, sondern leicht gekrümmmt, siehe Abb. 3.12.

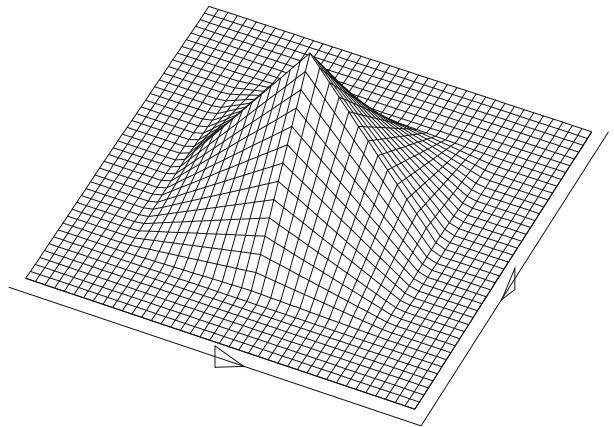


Abb. 3.12 Bilinearer Basisspline.

Satz 3.7. Zu den Daten $f_{ij}, -k \leq i, j \leq k, k \in \mathbb{N}^*$, existiert ein eindeutig bestimmter bilinearer Spline S mit

$$S(i, j) = f_{ij}, \quad -k \leq i, j \leq k.$$

S hat die Darstellung

$$S(x, y) = \sum_{i=-k}^k \sum_{j=-k}^k f_{ij} D_{11}(x - i, y - j), \quad (x, y) \in \mathbb{R}^2.$$

Diesen Satz kann man leicht in allgemeiner Form beweisen:

Satz 3.8. Das Interpolationsproblem

$$\sum_{\nu=0}^n a_\nu \varphi_\nu(k) = f_k, \quad k = 0, \dots, n,$$

sei für jeden Datensatz $(f_k)_{k=0,\dots,n} \in \mathbb{R}^{n+1}$ eindeutig lösbar.

Dann hat auch das durch Tensorproduktbildung definierte Interpolationsproblem

$$\sum_{\nu=0}^n \sum_{\mu=0}^m a_{\nu\mu} D_{\nu\mu}(k, l) = f_{kl} \quad \begin{cases} k = 0, \dots, n \\ l = 0, \dots, m \end{cases}$$

mit

$$D_{\nu\mu}(x, y) := \varphi_\nu(x) \varphi_\mu(y) \quad \begin{cases} \nu = 0, \dots, n \\ \mu = 0, \dots, m \end{cases}$$

für jeden Datensatz $(f_{kl}) \in \mathbb{R}^{(n+1)(m+1)}$ eine eindeutig bestimmte Lösung.

Beweis.

$$\sum_{\nu=0}^n \sum_{\mu=0}^m a_{\nu\mu} D_{\nu\mu}(k, l) = \sum_{\nu=0}^n \left(\sum_{\mu=0}^m a_{\nu\mu} \varphi_\mu(l) \right) \varphi_\nu(k).$$

Setzen wir

$$\tau_{\nu l} := \sum_{\mu=0}^m a_{\nu\mu} \varphi_\mu(l), \quad \begin{cases} \nu = 0, \dots, n \\ l = 0, \dots, m, \end{cases}$$

dann muß gelten

$$\sum_{\nu=0}^n \tau_{\nu l} \varphi_\nu(k) = f_{kl} \quad \begin{cases} k = 0, \dots, n \\ l = 0, \dots, m \end{cases}.$$

Das sind $m + 1$ eindeutig lösbare eindimensionale Interpolationsprobleme. Mit den $\tau_{\nu l}$ bekommt man die $n + 1$ ebenfalls eindeutig lösbar Interpolationsprobleme

$$\sum_{\mu=0}^m a_{\nu\mu} \varphi_\mu(l) = \tau_{\nu l} \quad \begin{cases} l = 0, \dots, m, \\ \nu = 0, \dots, n, \end{cases}$$

die gesuchten Koeffizienten eindeutig liefern. □

Durch die Tensorproduktbildung, d.h. durch den Aufbau der mehrdimensionalen Interpolation als Produkt aus eindimensionalen Interpolationen wird eine Aufwandsersparnis erreicht, ebenfalls dadurch, dass die B-Splines minimalen Träger haben.

Beispiel 3.8. Interpoliere bilinear mit $k = 10$, also auf dem Quadrat $[-10, 10] \times [-10, 10]$ die Funktion

$$f(x, y) = \exp\left(\frac{x}{10}\right) \exp\left(\frac{\sqrt{|y|}}{10}\right).$$

Das Ergebnis sehen wir in Abb. 3.13. △

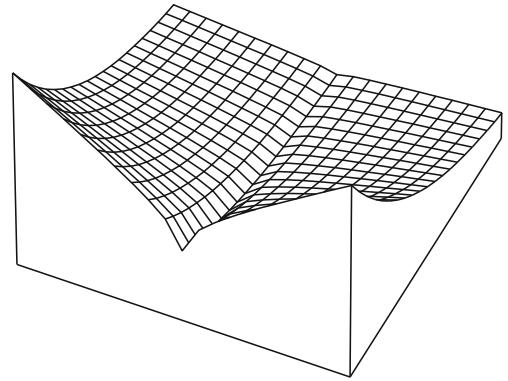


Abb. 3.13 Bilineare Spline-Interpolation von $f(x, y) = \exp\left(\frac{x}{10}\right) \exp\left(\frac{\sqrt{|y|}}{10}\right)$.

3.3.2 Bikubische Tensorsplines

Sei jetzt in (3.77)

$$\begin{aligned} s_\nu(x) &:= B_\nu(x), \quad \nu = 1, 2, \dots, n, \\ t_\mu(y) &:= B_\mu(y), \quad \mu = 1, 2, \dots, l. \end{aligned} \tag{3.83}$$

Dabei sind $B_\nu(x)$ bzw. $B_\mu(y)$ die in Abschnitt 3.2.3 eingeführten kubischen B-Splines, jetzt über den zwei Stützstellenmengen $\{x_i\}$ bzw. $\{y_j\}$ definiert² und in der Nummerierung um Eins verschoben. Einen typischen zweidimensionalen B-Spline auf einem 10×10 -Gitter kann man in Abb. 3.14 sehen.

Mit (3.83) ergeben sich die zweidimensionalen Ansatzfunktionen

$$\hat{B}_{\nu\mu}(x, y) := B_\nu(x)B_\mu(y). \tag{3.84}$$

In jedem Rechteck R_η des Gitters sind dies Polynome vom Höchstgrad 6:

$$\hat{B}_{\nu\mu}(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j, \quad \text{falls } (x, y) \in R_\eta. \tag{3.85}$$

Dabei hängen die Koeffizienten a_{ij} noch von den Spline-Indizes (ν, μ) und dem Rechteck R_η ab, müssten also eigentlich fünf Indizes haben. Die Funktionen gehen an den Rechteckseiten zweimal stetig differenzierbar ineinander über, wenn alle inneren Knotenpunkte verschieden sind. Das entspricht dem eindimensionalen Fall.

Die Lösung des Interpolationsproblems (3.77) auf dem Rechteckgitter (3.80) liefert jetzt das

²Wir wollen beide mit B bezeichnen, da uns das anschaulicher und weniger verwirrend erscheint als eine korrektere Bezeichnung wie etwa B und \tilde{B} .

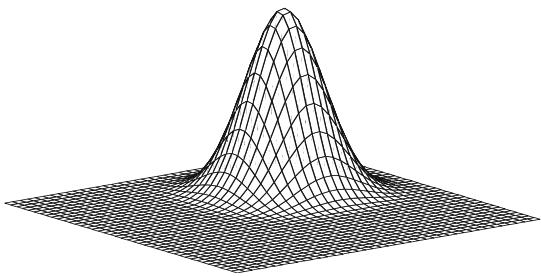


Abb. 3.14 Zweidimensionaler bikubischer B-Spline.

lineare Gleichungssystem

$$\sum_{\nu=1}^n \sum_{\mu=1}^l c_{\nu\mu} B_\nu(x_i) B_\mu(y_j) = f_{ij}, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, l. \quad (3.86)$$

Dies ist ein System von $n \times l$ linearen Gleichungen mit ebenso vielen Unbekannten. Es ist schwach besetzt und hat Block-Band-Struktur. Zur Lösung wird man dementsprechend ein spezielles Verfahren verwenden, z.B. eine spezielle QR-Zerlegung (siehe Kapitel 6). Aus Satz 3.8 folgt, dass die Lösung eindeutig bestimmt ist.

Ein lineares Gleichungssystem kann man auch bei beliebiger Stützstellenvorgabe ohne Kopplung an ein Gitter aufstellen. Seien also die Daten wie in (3.76) gegeben:

$$(x_i, y_i, f_i), \quad i = 1, 2, \dots, m.$$

Dann wird (3.86) zu

$$\sum_{\nu=1}^n \sum_{\mu=1}^l c_{\nu\mu} B_\nu(x_i) B_\mu(y_i) = f_i, \quad i = 1, 2, \dots, m. \quad (3.87)$$

Nun wird man aber die B-Splines nicht mehr über die Stützstellen $\{x_i\}$ und $\{y_i\}$ konstruieren, sondern mit einer nur noch vom Gesamtgebiet, in dem die Stützstellen liegen, abhängigen Knotenpunktmenge. Dabei sollte die Verteilung der Knotenpunkte abhängen von der Dichte der Stützstellen und der Variation der Stützwerte. Die Struktur des linearen Gleichungssystems (3.87) ist nicht mehr so klar vorher bestimmbar wie die von (3.86). Entsprechend wachsen Zeit- und Specheraufwand. Das Gleichungssystem (3.87) ist nur für $m = nl$ quadratisch und regulär. Um diese Bedingung nicht einhalten zu müssen, löst man es nach der Methode der kleinsten Quadrate und bekommt für $m \leq nl$ eine Interpolationslösung und für $m > nl$ eine Approximationslösung.

Beispiel 3.9. In Abb. 3.15 stellt die zweidimensionale Funktion $F(x, y)$ zwei zusammen treffende Flachwasserwellen dar. Mathematisch handelt es sich um die Lösung einer Kortevég-de Vries-Gleichung, die 2-Soliton genannt wird. Die Kortevég-de Vries-Gleichung ist eine partielle Differenzialgleichung, die in gewissen Spezialfällen analytisch lösbar ist, [Fuc 87]. Die in Abb. 3.15 dargestellte

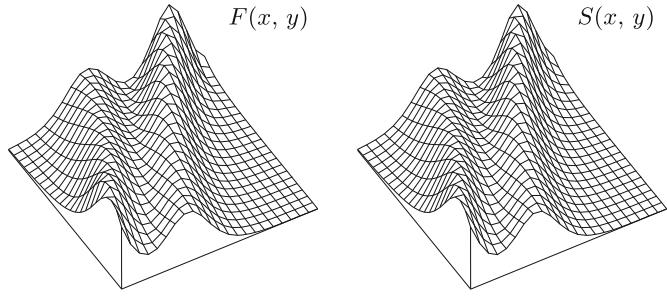


Abb. 3.15 2-Soliton und seine Splineapproximation.

Lösung hat die Form (hier ist y die Zeitvariable):

$$F(x, y) := -2 \frac{(a_1 + a_2 + \frac{5}{168}a_{12})^2}{(1 + \frac{5}{8}a_1 + \frac{5}{6}a_2 + \frac{25}{2352}a_{12})^2} + 2 \frac{\frac{8}{5}a_1 + \frac{6}{5}a_2 + \frac{1}{12}a_{12}}{1 + \frac{5}{8}a_1 + \frac{5}{6}a_2 + \frac{25}{2352}a_{12}} \quad (3.88)$$

mit

$$a_1 := \exp\left(\frac{128}{125}y - \frac{8}{5}x\right), \quad a_2 := \exp\left(\frac{54}{125}y - \frac{6}{5}x\right), \quad a_{12} := \exp\left(\frac{182}{125}y - \frac{14}{5}x\right).$$

Diese Funktion soll im Rechteck $[-8, 7] \times [-8, 7]$ von einer bikubischen Splinefunktion interpoliert werden. Wir geben die Funktionswerte auf allen ganzzahligen Gitterpunkten dieses Rechtecks vor. Die Knotenpunkte wählen wir in x - und y -Richtung entsprechend Abschnitt 3.2.3 (vierfache Randpunkte und alle inneren Stützstellen außer der zweiten und der zweitletzten). Das ergibt zusammen $20 \times 20 = 400$ Knotenpunkte. Die interpolierende Splinefläche werten wir auf 31×31 Punkten aus und zeichnen sie (Abb. 3.15 rechts). Diese Interpolation ist mit einem maximalen Fehler von etwa 10% in der unteren linken Ecke behaftet. Der mittlere Fehler ist wesentlich kleiner. \triangle

3.4 Kurveninterpolation

Problemstellung

Gegeben seien $n + 1$ Punkte $\mathbf{x}^{(i)} \in \mathbb{R}^m$

$$\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}), \quad i = 0, 1, \dots, n. \quad (3.89)$$

Sie sollen durch eine Kurve verbunden werden. Diese wird in Abhängigkeit von einem Parameter $t \in \mathbb{R}$, z.B. der Bogenlänge der Kurve, dargestellt.

Gesucht sind also m Parameterfunktionen

$$x_1(t), x_2(t), \dots, x_m(t), \quad t \in [0, 1], \quad (3.90)$$

die die zugehörigen Komponenten der gegebenen Punkte an den Stellen t_i interpolieren, für die gilt:

$$x_k(t_i) = x_k^{(i)}, \quad i = 0, 1, \dots, n, \quad k = 1, 2, \dots, m. \quad (3.91)$$

Den $n + 1$ Punkten $\mathbf{x}^{(i)}$ können folgende Parameterwerte t_i zugeordnet werden:

$$\begin{aligned} t_0 &:= 0 \\ t_i &:= t_{i-1} + \sqrt{\sum_{k=1}^m (x_k^{(i)} - x_k^{(i-1)})^2}, \quad i = 1, \dots, n. \end{aligned} \tag{3.92}$$

Meistens werden die t_i noch auf das Intervall $[0, 1]$ normiert:

$$t_i := \frac{t_i}{t_n}, \quad i = 1, \dots, n. \tag{3.93}$$

Numerische Lösung mit Splines

Es werden m eindimensionale Splineinterpolationen durchgeführt. Stützstellen sind immer die Parameterwerte t_i , Stützwerte für den k -ten Spline $x_k(t)$ die Werte $x_k^{(i)}$. Dann ist die Gleichung (3.91) erfüllt. Die Kurve im \mathbb{R}^m ist festgelegt durch ihre Parameterdarstellung

$$(x_1(t), x_2(t), \dots, x_m(t)), \quad t \in [0, 1]. \tag{3.94}$$

Beispiel 3.10. Der Effekt glatter Splineinterpolation soll an einem “schönen” Beispiel demonstriert werden. Wir wollen eine verzerrte Archimedesspirale zeichnen:

$$r(t) = ct \left(1 + \frac{bct}{d} \sin(at)\right). \tag{3.95}$$

Dabei sind (r, t) Polarkoordinaten, d.h. der Kurvenparameter t ist hier identisch mit dem Winkel φ der üblichen Polarkoordinatendarstellung. Als Zahlen wurden gewählt:

- $d = 500$: Verzerrungsverhältnis
- $c = 2$: Spiralenöffnung
- $b = 0.2$: Überlagerungsfaktor
- $a = 5.03$: Überlagerungsfrequenz

Die Parameterdarstellung ergibt sich als

$$\begin{aligned} x(t) &= r \cos(t), \\ y(t) &= r \sin(t). \end{aligned} \tag{3.96}$$

Es sollen 32 Umdrehungen dieser Spirale gezeichnet werden. Das haben wir mit 503 und 1006 Punkten versucht. Die Ergebnisse sind in Abb. 3.16 festgehalten. Die beiden unteren Kurven sind durch 503 Punkte gezeichnet, die oberen durch 1006 Punkte. Die linken beiden Kurven wurden als Polygonzug gezeichnet, also als gerade Verbindung der Stützpunkte, die beiden rechten mit Splines. An dem Unterschied kann man erkennen, wie wichtig eine gute Zeichen-Software in einem Graphik-Paket ist. In diesem Zusammenhang seien noch folgende Stichworte erwähnt: CAD, Bézier-Kurven und -Flächen (siehe Abschnitt 3.5), Beta-Splines und Coons-Flächen. \triangle

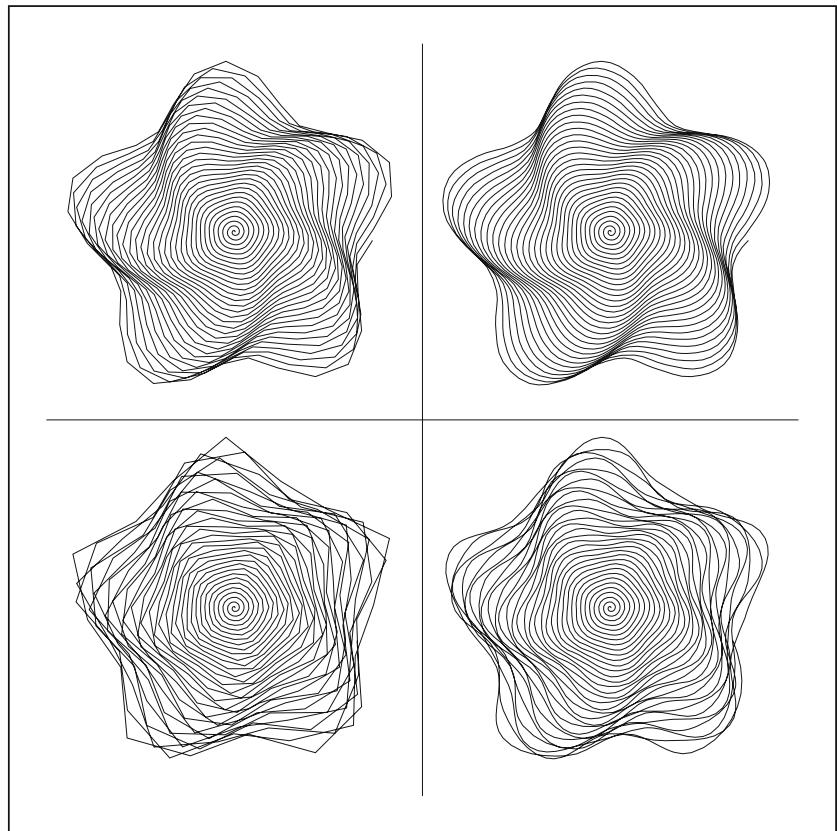


Abb. 3.16 Polygonzug und Spline durch 503 und 1006 Kurvenpunkte.

3.5 Kurven und Flächen mit Bézier-Polynomen

3.5.1 Bernstein-Polynome

Nach dem binomischen Lehrsatz gilt

$$1 = ((1-t) + t)^n = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i. \quad (3.97)$$

Die *Bernstein-Polynome* n -ten Grades in t bezüglich des Intervalls $[0, 1]$

$$B_{in}(t) := \binom{n}{i} (1-t)^{n-i} t^i, \quad i = 0, 1, \dots, n, \quad (3.98)$$

bilden deshalb eine Zerlegung der Eins. Der Mathematiker S. Bernstein hat mit ihnen 1912 den Weierstraßschen Approximationssatz konstruktiv bewiesen.

Das i -te Bernstein-Polynom vom Grad n bezüglich des Intervalls $[a, b]$ ist dann vermöge der Transformation

$$u \in [a, b] \rightarrow t \in [0, 1] : \quad t = \frac{u - a}{b - a}$$

für $i = 0, 1, \dots, n$ gegeben durch

$$B_{in}(u; a, b) := B_{in}\left(\frac{u - a}{b - a}\right) = \frac{1}{(b - a)^n} \binom{n}{i} (b - u)^{n-i} (u - a)^i. \quad (3.99)$$

Wir werden mit wenigen Ausnahmen und ohne wesentliche Einschränkung auf dem Intervall $[0, 1]$ arbeiten. Für die meisten Sätze wollen wir auf die Beweise verzichten und verweisen auf [Deu 08b, Loc 93].

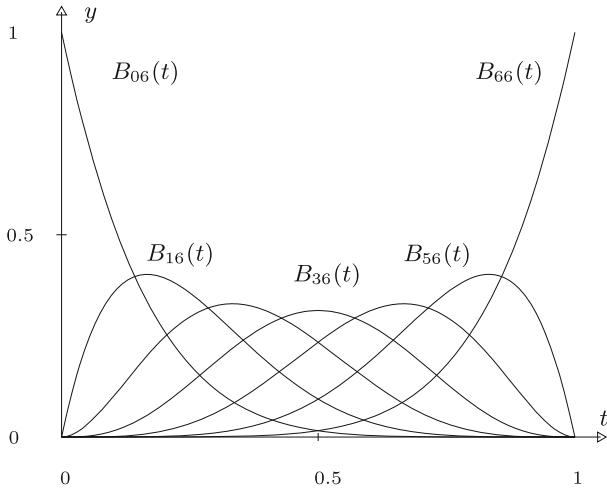


Abb. 3.17 Bernstein-Polynome 6. Grades.

Satz 3.9. Sei $n \in \mathbb{N}^*$, $0 \leq i \leq n$, dann gilt für $B_{in}(t)$:

$$t = 0 \text{ ist } i\text{-fache Nullstelle von } B_{in}. \quad (3.100)$$

$$t = 1 \text{ ist } (n - i)\text{-fache Nullstelle von } B_{in}. \quad (3.101)$$

$$B_{in}(t) = B_{n-i,n}(1-t) \quad (\text{Symmetrie}). \quad (3.102)$$

$$0 \leq B_{in}(t) \leq 1 \quad \forall t \in [0, 1], \quad B_{in}(t) > 0 \quad \forall t \in (0, 1). \quad (3.103)$$

Satz 3.10. Das Bernstein-Polynom B_{in} hat in $[0, 1]$ genau ein Maximum, und zwar bei $t_{\max} = i/n$.

Satz 3.11. Die Bernstein-Polynome genügen der Rekursionsformel

$$B_{in}(t) = t B_{i-1,n-1}(t) + (1-t) B_{i,n-1}(t), \quad i = 1, \dots, n-1, \quad (3.104)$$

$$B_{0n}(t) = (1-t) B_{0,n-1}(t),$$

$$B_{nn}(t) = t B_{n-1,n-1}(t).$$

Satz 3.12. Die $\{B_{in}(t)\}_{i=0}^n$ sind linear unabhängig und bilden eine Basis des Raumes Π_n der Polynome n -ten Grades.

Satz 3.13. Die Ableitungen der Bernstein-Polynome ($n \geq 1$) sind gegeben durch

$$B'_{in}(t) = \begin{cases} -n B_{0,n-1}(t) & \text{für } i = 0, \\ n [B_{i-1,n-1}(t) - B_{i,n-1}(t)] & \text{für } i = 1, 2, \dots, n-1, \\ n B_{n-1,n-1}(t) & \text{für } i = n. \end{cases} \quad (3.105)$$

3.5.2 Bézier-Darstellung eines Polynoms

Da die Bernstein-Polynome eine Basis des Raumes Π_n bilden, lässt sich jedes Polynom $p \in \Pi_n$ darstellen als

$$p(t) = \sum_{i=0}^n \beta_i B_{in}(t). \quad (3.106)$$

Dies nennt man die *Bézier*³-Darstellung von p . Die β_i heißen *Bézier-Koeffizienten* und die Punkte

$$(i/n, \beta_i)^T \in \mathbb{R}^2, \quad i = 0, 1, \dots, n,$$

Bézier-Punkte.

Die Verbindung der Bézier-Punkte durch Geraden heißt *Bézier-Polygon*.

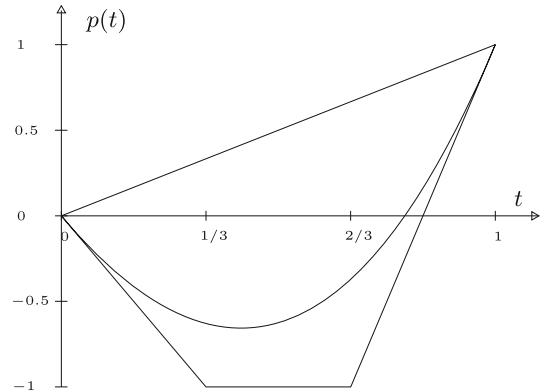


Abb. 3.18 Bézier-Polygon mit Bézier-Polynom vom Grad 3.

Beispiel 3.11. Abb. 3.18 zeigt das Bézier-Polynom zu den Bézier-Punkten $(0, 0)$, $(1/3, -1)$, $(2/3, -1)$, $(1, 1)$

$$p(t) = 0 \cdot B_{03}(t) - B_{13}(t) - B_{23}(t) + B_{33}(t) = t^3 + 3t^2 - 3t$$

zusammen mit seinem Bézier-Polygon. \triangle

³Bézier: zeitgenössischer französischer Industrie-Mathematiker.

Satz 3.14. 1. Der Graph eines Polynoms liegt in der konvexen Hülle seiner Bézier-Punkte.
2. Der Graph eines Polynoms und sein Bézier-Polygon berühren sich tangential in $t = 0$ und in $t = 1$.

3. Für die Ableitungen eines Polynoms mit Bézier-Koeffizienten β_i gilt:

$$p^{(k)}(t) = n(n-1)\cdots(n-k+1) \sum_{i=0}^{n-k} \Delta^k \beta_i B_{i,n-k}(t), \quad (3.107)$$

wenn Δ^k die Vorwärtsdifferenzen $\Delta\beta_i := \beta_{i+1} - \beta_i$, $\Delta^k\beta_i := \Delta^{k-1}\beta_{i+1} - \Delta^{k-1}\beta_i$ sind.

3.5.3 Der Casteljau-Algorithmus

Die Rekursionsformel (3.104) führt für ein Bézier-Polynom (3.106) im ersten Schritt zu

$$\begin{aligned} p(t) &= \beta_0(1-t)B_{0,n-1}(t) + \beta_1[(1-t)B_{1,n-1}(t) + tB_{0,n-1}(t)] + \cdots \\ &\quad \cdots + \beta_n t B_{n-1,n-1}(t) \\ &= [\beta_0(1-t) + \beta_1 t]B_{0,n-1}(t) + \cdots + [\beta_{n-1}(1-t) + \beta_n t]B_{n-1,n-1}(t), \end{aligned}$$

also

$$\begin{aligned} p(t) &= \sum_{i=0}^{n-1} \beta_i^{(1)} B_{i,n-1}(t) \quad \text{mit} \\ \beta_i^{(1)} &= \beta_i(1-t) + \beta_{i+1}t, \quad i = 0, \dots, n-1, \quad \beta_i^{(1)} = \beta_i^{(1)}(t). \end{aligned}$$

Führt man das rekursiv fort, kommt man zu

$$p(t) = \beta_0^{(n)}. \quad (3.108)$$

Diese rekursive Berechnung eines Polynomwertes $p(t)$ entspricht dem Dreiecksschema

$$\begin{array}{ccccccc} \beta_0 & = & \beta_0^{(0)} & & & & \\ & & \beta_0^{(1)} & & & & \\ & & & \ddots & & & \\ & & \vdots & \vdots & \vdots & \ddots & \beta_0^{(n)} \\ & & & & & \ddots & \\ & & & & & & \beta_{n-1}^{(1)} \\ \beta_n & = & \beta_n^{(0)} & & & & \end{array}$$

und heißt *Algorithmus von de Casteljau*.

Die Koeffizienten $\beta_i^{(1)}$ des Algorithmus von de Casteljau besitzen folgende geometrische Bedeutung. Der Punkt (x_i, y_i) mit

$$\begin{aligned} x_i &= (1-t) \frac{i}{n} + t \frac{i+1}{n} = \frac{i+t}{n} \\ \beta_i^{(1)} = y_i &= (1-t)\beta_i + t\beta_{i+1}, \quad i = 0, \dots, n-1 \end{aligned} \quad (3.109)$$

liegt auf der Geraden

$$\binom{i/n}{\beta_i} \rightarrow \binom{(i+1)/n}{\beta_{i+1}},$$

also auf dem Bézier-Polygon.

Rekursiv entsteht ein System von Zahlen $\beta_i^{(k)}$, $k = 1, \dots, n$, $i = 0, \dots, n - k$, die durch lineare Interpolation aus den Zahlen $\beta_i^{(k-1)}$ entstehen, bis der Punkt $p(t) = \beta_0^{(n)}$ erreicht ist. Dieser Prozess kann am Beispiel 3.12 nachvollzogen werden.

Beispiel 3.12. Wir wollen für $t = 0.5$ den Polynomwert des Bézier-Polynoms $p(t) = 0 \cdot B_{03}(t) - B_{13}(t) - B_{23}(t) + B_{33}(t)$ aus Beispiel 3.11 berechnen. In Abb. 3.19 sind die Punkte verschieden markiert, die innerhalb des Casteljau-Algorithmus für den Wert $t = 0.5$ entstehen. Aus den vier Bézier-Punkten $(0, 0)$, $(1/3, -1)$, $(2/3, -1)$, $(1, 1)$ werden wegen $t = 0.5$ die drei Mittelpunkte $(x_i, \beta_i^{(1)})$ der Teilgeraden des Bézier-Polygons (\circ). Die beiden Geraden, die diese Punkte verbinden, werden wieder mittig geteilt (+). Der Mittelpunkt der verbleibenden Verbindungsstrecke (\bullet) ist der gesuchte Punkt auf dem Polynom $p(0.5)$. \triangle

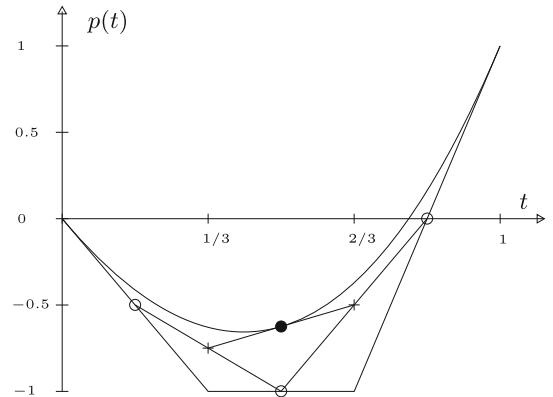


Abb. 3.19 Der Casteljau-Algorithmus geometrisch.

3.5.4 Bézier-Kurven

In der graphischen Datenverarbeitung ist die Einschränkung einer Kurve auf eine Funktion nicht sinnvoll, da Kurven unabhängig von ihrer Darstellung bezüglich eines Koordinaten- systems sein sollten.

Der Übergang von Bézier-Polynomen zu Bézier-Kurven geschieht über einen Parameter (hier einfach wieder t) wie beim Übergang von Splines zu Parameter-Splines in Abschnitt 3.4.

Definition 3.15. Seien für $0 \leq t \leq 1$ die Funktionen $x_k(t)$, $k = 1, 2, \dots, d$, Bézier-Polynome. Dann ist $(x_1(t), x_2(t), \dots, x_d(t))^T$ eine *Bézier-Kurve* im Raum \mathbb{R}^d .

Wir werden uns auf $d = 2$ beschränken und die Bézier-Kurve vereinfachend mit $\mathbf{P}(t) = (x(t), y(t))^T$ bezeichnen. Bei der geometrischen Konstruktion kann man weiterhin den de-Casteljau-Algorithmus verwenden, die Kurvendarstellung geschieht jetzt allerdings über die

Bézier-Punkte

$$\mathbf{P}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \in \mathbb{R}^2, \quad (3.110)$$

deren x -Werte nicht mehr äquidistant sein müssen.

Satz 3.16. $\mathbf{P}_0, \dots, \mathbf{P}_n \in \mathbb{R}^2$ seien $n + 1$ Punkte in der Ebene. Dann ist die Bézier-Kurve mit diesen Punkten als Bézier-Punkten gegeben als

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \mathbf{P}(t) = \sum_{i=0}^n B_{in}(t) \mathbf{P}_i. \quad (3.111)$$

$x(t)$ und $y(t)$ sind Bézier-Polynome.

Der Algorithmus von de Casteljau lässt sich vollständig übertragen. Wir tun dies nur für $n = 3$. Sei also

$$\mathbf{P}_{i0} := \mathbf{P}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \in \mathbb{R}^2, \quad i = 0, 1, 2, 3.$$

Dann werden wie oben für einen festen Wert t folgende Punkte berechnet:

$$\begin{aligned} \mathbf{P}_{i1} &:= (1-t) \mathbf{P}_{i0} + t \mathbf{P}_{i+1,0}, \quad i = 0, 1, 2 \\ \mathbf{P}_{i2} &:= (1-t) \mathbf{P}_{i1} + t \mathbf{P}_{i+1,1}, \quad i = 0, 1 \\ \mathbf{P}_{03} &:= (1-t) \mathbf{P}_{02} + t \mathbf{P}_{12}. \end{aligned} \quad (3.112)$$

oder

$$\begin{aligned} \mathbf{P}_{03} &= (1-t) \mathbf{P}_{02} + t \mathbf{P}_{12} \\ &= (1-t)((1-t) \mathbf{P}_{01} + t \mathbf{P}_{11}) + t((1-t) \mathbf{P}_{11} + t \mathbf{P}_{21}) \\ &= (1-t)^3 \mathbf{P}_{00} + 3(1-t)^2 t \mathbf{P}_{10} + 3(1-t)t^2 \mathbf{P}_{20} + t^3 \mathbf{P}_{30} \end{aligned}$$

Ein Beispiel für die geometrische Konstruktion der Kurve für den Punkt $t = 1/2$ ist in Abb. 3.20 zu sehen.

Aus den Eigenschaften der Bernstein- und Bézier-Polynome kann jetzt auf entsprechende Eigenschaften der Bézier-Kurven geschlossen werden. So folgt aus Satz 3.14 der

Satz 3.17. Die Menge der Punkte einer Bézier-Kurve

$$M := \left\{ \mathbf{P}(t) = \sum_{i=0}^n B_{in}(t) \mathbf{P}_i, \quad t \in [0, 1] \right\} \quad (3.113)$$

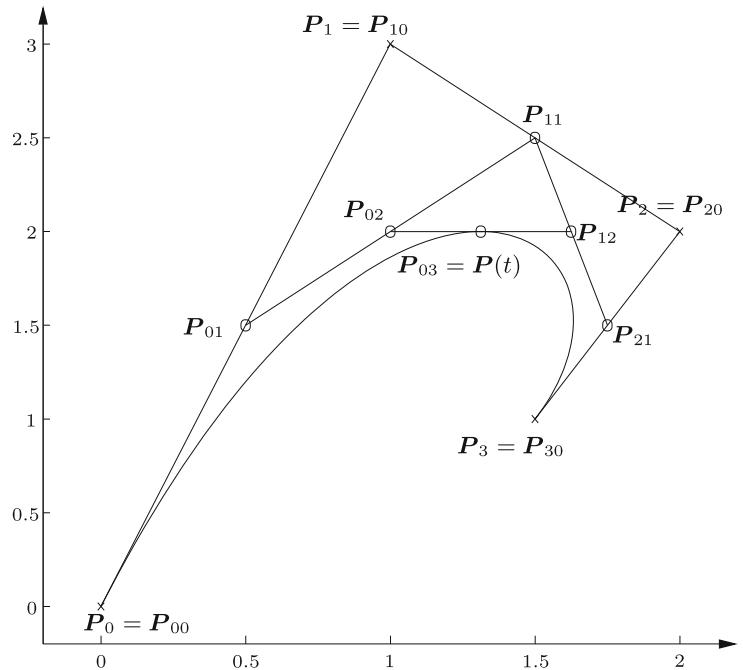
liegt in der konvexen Hülle der $n + 1$ Bézier-Punkte $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$.

Aus (3.100), (3.101) und Satz 3.13 folgt der

Satz 3.18. Für die Randpunkte einer Bézier-Kurve $\mathbf{P}(t) = \sum_{i=0}^n B_{in}(t) \mathbf{P}_i$, $n \geq 2$, gelten:

$$\mathbf{P}(0) = \mathbf{P}_0, \quad \mathbf{P}(1) = \mathbf{P}_n, \quad (3.114)$$

$$\mathbf{P}'(0) = n(\mathbf{P}_1 - \mathbf{P}_0), \quad \mathbf{P}'(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1}). \quad (3.115)$$

Abb. 3.20 Punktweise Kurven-Konstruktion, hier für $t = 1/2$.

Die Berechnung von $\mathbf{P}(t)$ mit dem Casteljau-Algorithmus kann für festes n algorithmisch besonders günstig als Matrix*Matrix*Vektor dargestellt werden.

Lemma 3.19. Für die Bézier-Kurve $\mathbf{P}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$, $0 \leq t \leq 1$, n -ten Grades gilt

$$\mathbf{P}(t) = (\mathbf{P}_0 \ \mathbf{P}_1 \ \dots \ \mathbf{P}_n) \ \mathbf{B}_n \begin{pmatrix} t^n \\ t^{n-1} \\ \vdots \\ t \\ 1 \end{pmatrix} \quad (3.116)$$

mit der für jedes n konstanten Matrix \mathbf{B}_n der Ordnung $n+1$ mit den Elementen

$$b_{i+1,k+1} = \begin{cases} (-1)^{i+k+n} \binom{n-i}{k} \binom{n}{i} & \text{für } \begin{cases} i = 0, 1, \dots, n, \\ k = 0, 1, \dots, n-i \end{cases} \\ 0 & \text{sonst} \end{cases} \quad (3.117)$$

Beispiel 3.13.

$$\mathbf{P}_3(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}.$$

△

Zur Darstellung einer Kurve mit gewissen erwünschten Eigenschaften reicht *eine* Bézier-Kurve nicht aus, sondern die Kurve wird stückweise aus Bézier-Kurvensegmenten zusammengesetzt. Entsprechend der Zielsetzung soll die zusammengesetzte Kurve neben der Stetigkeit an den Segmentteilknoten gegebenenfalls auch bestimmte Glattheitseigenschaften besitzen. Die Stetigkeit ist gewährleistet, wenn der Endpunkt des einen Kurvensegments mit dem Anfangspunkt des nächsten Segmentes übereinstimmt, d.h. wenn die Bézier-Punkte zusammenfallen. Nach Satz 3.18 wird die Stetigkeit der ersten Ableitung dadurch erreicht, dass der vorletzte Bézier-Punkt, der erste nachfolgende Bézier-Punkt und der gemeinsame Bézier-Punkt in einem bestimmten Abstandsverhältnis auf einer Geraden liegen. An dieser Stelle kommen die Parameterintervalle der Kurvensegmente, die der Situation problemgerecht angepasst werden sollten, ins Spiel. Ihre Längen sollten ungefähr den Längen der Kurvenstücke entsprechen. Die ganze Kurve werde mit dem Parameter u parametrisiert, sie sei aus m Bézier-Kurvensegmenten zusammengesetzt, und die Segmente sollen eine polynomiale Parameterdarstellung vom gleichen Grad n aufweisen. Das Parameterintervall sei unterteilt durch

$$u_0 < u_1 < u_2 < u_3 < \dots < u_{m-1} < u_m, \quad (3.118)$$

und das j -te Bézier-Kurvensegment habe die Darstellung

$$\mathbf{P}_j(u) = \sum_{i=0}^n \mathbf{P}_{ij} B_{in}(u; u_{j-1}, u_j), \quad u \in [u_{j-1}, u_j], \quad (3.119)$$

mit den Bézier-Punkten $\mathbf{P}_{0j}, \mathbf{P}_{1j}, \dots, \mathbf{P}_{nj}$, $j = 1, 2, \dots, m$. Die Länge des j -ten Parameterintervalls $[u_{j-1}, u_j]$ sei $h_j := u_j - u_{j-1}$. Das j -te und $(j+1)$ -te Bézier-Kurvensegment sind an der Stelle $u = u_j$ stetig, wenn

$$\mathbf{P}_{n,j} = \mathbf{P}_{0,j+1} \quad (3.120)$$

gilt. Für $u = u_j$ besteht C^1 -Stetigkeit genau dann, wenn $\mathbf{P}'_j(u_j) = \mathbf{P}'_{j+1}(u_j)$ erfüllt ist. Wegen (3.99) gilt für die Ableitung der Bernstein-Polynome

$$\frac{d}{du} B_{in}(u; u_{j-1}, u_j) = \frac{d}{du} B_{in} \left(\frac{u - u_{j-1}}{h_j} \right) = \frac{d}{dt} B_{in}(t) \cdot \frac{1}{h_j},$$

und das führt zusammen mit (3.115) auf die Bedingungsgleichung

$$n(\mathbf{P}_{n,j} - \mathbf{P}_{n-1,j})/h_j = n(\mathbf{P}_{1,j+1} - \mathbf{P}_{0,j+1})/h_{j+1}. \quad (3.121)$$

Weil aus Stetigkeitsgründen $\mathbf{P}_{n,j} = \mathbf{P}_{0,j+1}$ ist, so kann die Bedingung (3.121) der C^1 -Stetigkeit für $u = u_j$ wie folgt formuliert werden:

$$\mathbf{P}_{n,j} = \frac{h_{j+1}}{h_j + h_{j+1}} \mathbf{P}_{n-1,j} + \frac{h_j}{h_j + h_{j+1}} \mathbf{P}_{1,j+1} \quad (3.122)$$

Der dem j -ten und $(j+1)$ -ten Bézier-Kurvensegment gemeinsame Bézier-Punkt $\mathbf{P}_{n,j} = \mathbf{P}_{0,j+1}$ muss eine bestimmte lineare Konvexitätskombination der beiden benachbarten Bézier-Punkte $\mathbf{P}_{n-1,j}$ und $\mathbf{P}_{1,j+1}$ des Bézier-Polygons sein. Geometrisch bedeutet (3.122), dass die Strecke von $\mathbf{P}_{n-1,j}$ nach $\mathbf{P}_{1,j+1}$ durch $\mathbf{P}_{n,j}$ im Verhältnis $h_j : h_{j+1}$ unterteilt sein muss.

Mit zusätzlichen Bedingungen an die Bézier-Punkte kann für die Kurve sogar zweimalige stetige Differenzierbarkeit erreicht werden; hierauf wollen wir nicht eingehen.

Zur datenmäßigen Erfassung einer Bézier-Kurve, die aus m Kurvensegmenten n -ten Grades zusammengesetzt ist, werden $m \cdot n + 1$ Bézier-Punkte, d.h. d -dimensionale Vektoren benötigt. Um in der bisherigen Indizierung drei Indizes zu vermeiden und um der Stetigkeitsbedingung (3.120) einfache Rechnung zu tragen, werden die Daten in einer d -zeiligen Matrix mit $m \cdot n + 1$ Spalten gespeichert.

$$\mathbf{B} := (\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \dots, \mathbf{P}_{m \cdot n - 1}, \mathbf{P}_{m \cdot n}) \quad (3.123)$$

Das j -te Kurvensegment besitzt dann zu den ebenfalls vorgegebenen Parameterwerten u_j (3.118) die Bézier-Darstellung

$$\mathbf{P}_j(u) = \sum_{i=0}^n \mathbf{P}_{(j-1) \cdot n + i} B_{in}(u; u_{j-1}, u_j), \quad j = 1, 2, \dots, m. \quad (3.124)$$

Erfüllen die Spalten von \mathbf{B} die einschlägigen Bedingungen der C^p -Stetigkeit mit $p \geq 1$, so ist die resultierende Bézier-Kurve bezüglich des Kurvenparameters u p -mal stetig differenzierbar. Dies kann etwa bei der Steuerung einer Werkzeugmaschine wichtig sein, wenn durch gleichmäßige Änderung von u eine Bewegung mit stetiger Geschwindigkeit erzielt werden soll. Ist man hingegen nur an der Bézier-Kurve als solche interessiert, dann kann anstelle von (3.124) die einfachere Form

$$\mathbf{P}_j(t) = \sum_{i=0}^n \mathbf{P}_{(j-1) \cdot n + i} B_{in}(t), \quad t \in [0, 1], \quad j = 1, 2, \dots, m. \quad (3.125)$$

verwendet werden, welche die gleiche Punktmenge definiert.

Beispiel 3.14. Es soll ein Viertelkreis vom Radius $r = 1$ durch eine Bézier-Kurve dritten Grades approximiert werden unter der Bedingung, dass die Tangenten der Bézier-Kurve in den Endpunkten mit den Tangenten an den Kreis übereinstimmen. Dadurch ist die Lage der Bézier-Punkte \mathbf{P}_1 und \mathbf{P}_2 bereits auf die Tangenten festgelegt, und aus Symmetriegründen ist die Bézier-Kurve bis auf einen freien Parameter ξ bestimmt (siehe Abb. 3.21).

Der Ansatz lautet somit für die gesuchte Bézier-Kurve

$$\mathbf{P}(t) = \sum_{i=0}^3 \mathbf{P}_i B_{i3}(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (1-t)^3 + \begin{pmatrix} 1 \\ \xi \end{pmatrix} 3t(1-t)^2 + \begin{pmatrix} \xi \\ 1 \end{pmatrix} 3t^2(1-t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} t^3.$$

Zur Bestimmung von ξ fordern wir beispielsweise, dass der Punkt $\mathbf{P}(0.5)$ der Bézier-Kurve auf dem Kreis liegt. Dies liefert den Wert $\xi = 0.552285$. Die in Abb. 3.21 dargestellte Bézier-Kurve weicht vom Kreis um maximal 0.00027 ab und stellt somit für zeichnerische Zwecke eine hervorragende Näherung dar.

Mit diesem Kurvensegment lässt sich eine Ellipse mit den Halbachsen $a = 1/\xi \doteq 1.81066$ und $b = 1$ durch den folgenden Satz von 13 Bézier-Punkten mit derselben Genauigkeit und C^1 -Stetigkeit

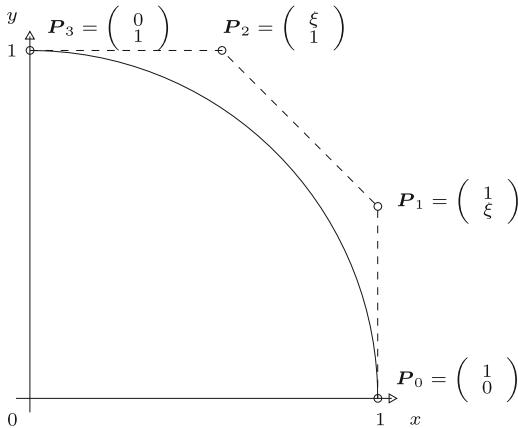


Abb. 3.21 Bézier-Kurve für den Viertelkreis.

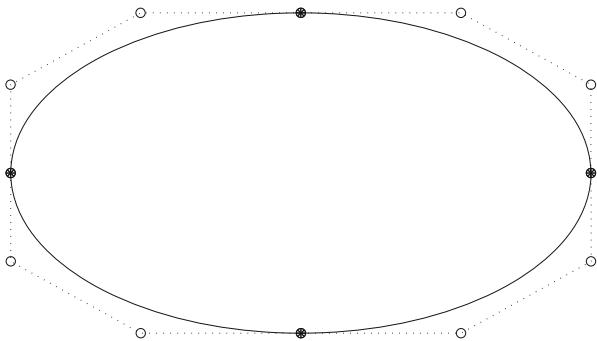


Abb. 3.22 Ellipse aus vier kubischen Segmenten.

approximieren (siehe Abb. 3.22).

$$\mathbf{B} = \begin{pmatrix} a & a & 1 & 0 & -1 & -a & -a & -a & -1 & 0 & 1 & a & a \\ 0 & \xi & 1 & 1 & 1 & \xi & 0 & -\xi & -1 & -1 & -1 & -\xi & 0 \end{pmatrix}$$

△

Beispiel 3.15. Das Profil eines Tragflügels lässt sich etwa durch sechs kubische Bézier-Kurvensegmente mit C^1 -Stetigkeit recht gut approximieren. Die unterschiedlich langen Segmente sind bei der Festlegung der Bézier-Punkte zu beachten, um die C^1 -Stetigkeit sicherzustellen. Die Bézier-Punkte sind auf Grund der Unterteilung in $u_0 = 0$, $u_1 = 1.6$, $u_2 = 2.6$, $u_3 = 3.1$, $u_4 = 3.6$, $u_5 = 4.6$, $u_6 = 6.2$ festgelegt worden. Abb. 3.23 zeigt das aus sechs Segmenten bestehende Bézier-Polygon (gepunktet) und die zugehörige Bézier-Kurve. △

Beispiel 3.16. Bézier-Kurven finden auch Anwendung beim Zeichenentwurf für Textverarbeitungsprogramme, um auf diese Weise den Umriss von Zeichen zu definieren.

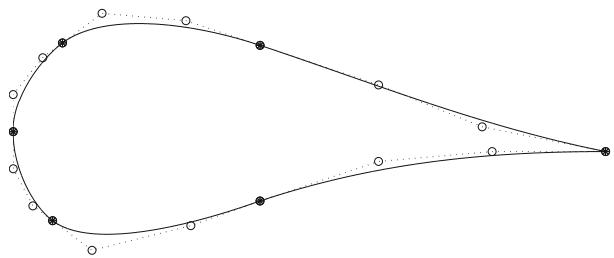


Abb. 3.23 Tragflügelprofil.

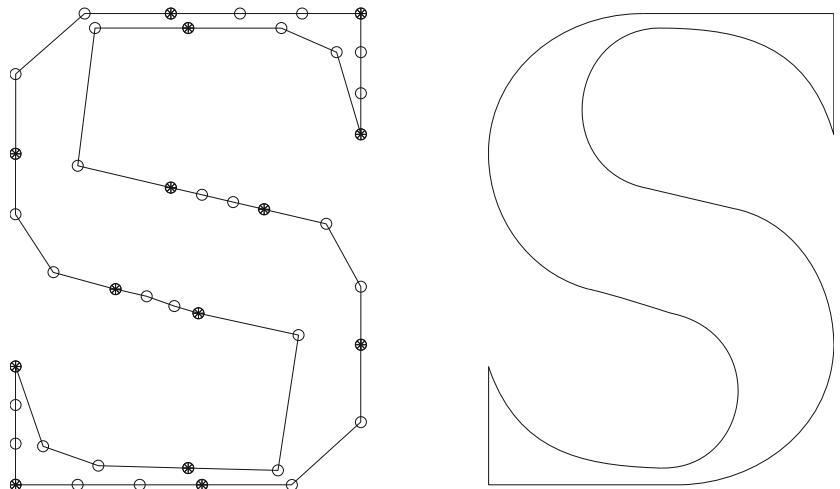


Abb. 3.24 Zeichenentwurf: Der Buchstabe S.

In Abb. 3.24 sind das Bézier-Polygon und der Umriss für den Großbuchstaben S wiedergegeben. Dieses Beispiel zeigt, wie flexibel die Konstruktion von Bézier-Kurven ist. Sie eignen sich deshalb sehr gut zum Zeichnen von Kurven, die sowohl glatte Verläufe als auch Ecken enthalten müssen. \triangle

3.5.5 Bézier-Flächen

Der Schritt von den Bézier-Kurven zu Bézier-Flächen ist auf Grund der bekannten Tatsachen einfach. Die zwei Parameter, die für eine Parameterdarstellung einer Fläche erforderlich sind, bezeichnen wir im Folgenden mit s und t , wenn sie im Einheitsintervall variieren, und mit u und v für allgemeinere Intervalle. Die Grundidee zur Definition von Bézier-Flächen besteht darin, dass man von zwei räumlichen Kurven ausgeht, welche Bézier-Darstellungen vom gleichen Grad m und mit demselben Parameterintervall haben. Längs dieser Bézier-Kurven lässt man zu gleichen Parameterwerten eine zweite Bézier-Kurve vom Grad n gleiten, deren Verlauf durchaus vom ersten Parameter abhängen kann. Das so erzeugte Flächenstück besitzt eine Parameterdarstellung, die als Tensorprodukt von Bernstein-Polynomen dargestellt

werden kann, ganz entsprechend dem Vorgehen bei Tensorsplines in Abschnitt 3.3.1.

$$\mathbf{x}(s, t) := \sum_{i=0}^n \sum_{j=0}^m \mathbf{P}_{ij} B_{in}(s) B_{jm}(t), \quad s, t \in [0, 1]. \quad (3.126)$$

Darin stellen die \mathbf{P}_{ij} , $i = 0, 1, \dots, n$; $j = 0, 1, \dots, m$, dreidimensionale Koordinatenvektoren dar, die man die *Bézier-Punkte* der *Bézier-Fläche* $\mathbf{x}(s, t)$ nennt. Üblicherweise fasst man diese $(n+1)(m+1)$ Bézier-Punkte \mathbf{P}_{ij} in einer Bézier-Punktematrix

$$\begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \dots & \mathbf{P}_{0m} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \dots & \mathbf{P}_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{n0} & \mathbf{P}_{n1} & \dots & \mathbf{P}_{nm} \end{pmatrix} \quad (3.127)$$

zusammen mit Elementen $\mathbf{P}_{ij} \in \mathbb{R}^3$, so dass für eine rechnerische Verarbeitung ein dritter Index $k = 1, 2, 3$ hinzukommt.

Eine unmittelbare Folge von (3.100) und (3.101) ist der

Satz 3.20. Für eine Bézier-Fläche $\mathbf{x}(s, t)$ (3.126) gelten

$$\mathbf{x}(0, 0) = \mathbf{P}_{00}, \quad \mathbf{x}(0, 1) = \mathbf{P}_{0m}, \quad \mathbf{x}(1, 0) = \mathbf{P}_{n0}, \quad \mathbf{x}(1, 1) = \mathbf{P}_{nm}, \quad (3.128)$$

$$\mathbf{x}(0, t) = \sum_{j=0}^m \mathbf{P}_{0j} B_{jm}(t), \quad \mathbf{x}(1, t) = \sum_{j=0}^m \mathbf{P}_{nj} B_{jm}(t), \quad (3.129)$$

$$\mathbf{x}(s, 0) = \sum_{i=0}^n \mathbf{P}_{i0} B_{in}(s), \quad \mathbf{x}(s, 1) = \sum_{i=0}^n \mathbf{P}_{im} B_{in}(s). \quad (3.130)$$

Geometrisch bedeuten (3.128), dass die Bézier-Punkte \mathbf{P}_{00} , \mathbf{P}_{0m} , \mathbf{P}_{n0} und \mathbf{P}_{nm} die Ecken des Bézier-Flächenstückes $\mathbf{x}(s, t)$ sind. Die vier durch (3.129) und (3.130) definierten, das Flächensegment $\mathbf{x}(s, t)$ (3.126) begrenzenden Randkurven sind Bézier-Kurven. Dabei sind die erste bzw. die letzte Zeile der Bézier-Punktematrix (3.127) die Bézier-Punkte der Randkurven (3.129), und die erste und letzte Spalte von (3.127) sind die Bézier-Punkte der Randkurven (3.130). Auf Grund dieser Feststellung wird das stetige Zusammensetzen von Flächensegmenten einfach sein.

Satz 3.21. Die Menge der Punkte der Bézier-Fläche (3.126)

$$M := \{\mathbf{x}(s, t) | s, t \in [0, 1]\}$$

liegt in der konvexen Hülle ihrer $(n+1)(m+1)$ Bézier-Punkte.

Wegen Satz 3.13 über die Ableitungen der Bernstein-Polynome gilt für die ersten partiellen Ableitungen

$$\frac{\partial \mathbf{x}}{\partial s} = n \sum_{i=0}^{n-1} \sum_{j=0}^m (\mathbf{P}_{i+1,j} - \mathbf{P}_{i,j}) B_{i,n-1}(s) B_{jm}(t),$$

$$\frac{\partial \mathbf{x}}{\partial t} = m \sum_{i=0}^n \sum_{j=0}^{m-1} (\mathbf{P}_{i,j+1} - \mathbf{P}_{i,j}) B_{in}(s) B_{jm}(t),$$

und daraus erhalten wir wegen (3.100), (3.101) den für die glatte Zusammensetzung von Flächensegmenten bedeutsamen

Satz 3.22. Für die partiellen Ableitungen einer Bézier-Fläche $\mathbf{x}(s, t)$ (3.126) längs der Randkurven gelten

$$\begin{aligned} \frac{\partial \mathbf{x}(0, t)}{\partial s} &= n \sum_{j=0}^m (\mathbf{P}_{1j} - \mathbf{P}_{0j}) B_{jm}(t), \\ \frac{\partial \mathbf{x}(1, t)}{\partial s} &= n \sum_{j=0}^m (\mathbf{P}_{n,j} - \mathbf{P}_{n-1,j}) B_{jm}(t), \end{aligned} \quad (3.131)$$

$$\begin{aligned} \frac{\partial \mathbf{x}(s, 0)}{\partial t} &= m \sum_{i=0}^n (\mathbf{P}_{i1} - \mathbf{P}_{i0}) B_{in}(s), \\ \frac{\partial \mathbf{x}(s, 1)}{\partial t} &= m \sum_{i=0}^n (\mathbf{P}_{i,m} - \mathbf{P}_{i,m-1}) B_{in}(s). \end{aligned} \quad (3.132)$$

Schließlich erfolgt die effiziente und numerisch stabile Berechnung eines Wertes $\mathbf{x}(s, t)$ grundsätzlich durch zweimaliges Anwenden des Algorithmus von de Casteljau. Soll etwa die Kurve der Bézier-Fläche für $s = \text{const}$ bestimmt und dargestellt werden, so erhalten wir mit

$$\mathbf{x}(s, t) = \sum_{j=0}^m \left\{ \sum_{i=0}^n \mathbf{P}_{ij} B_{in}(s) \right\} B_{jm}(t) =: \sum_{j=0}^m \mathbf{Q}_j(s) B_{jm}(t) \quad (3.133)$$

ihre Bézier-Darstellung mit den $(m + 1)$ von s abhängigen Bézier-Punkten

$$\mathbf{Q}_j(s) := \sum_{i=0}^n \mathbf{P}_{ij} B_{in}(s), \quad j = 0, 1, \dots, m. \quad (3.134)$$

Diese erhält man dadurch, dass der Algorithmus von de Casteljau für den festen Wert s auf die $(m + 1)$ Spalten der Bézier-Punktematrix angewendet wird. Mit diesen Hilfspunkten $\mathbf{Q}_j(s)$ ergibt sich die gesuchte Flächenkurve, indem man für jedes t den Algorithmus von de Casteljau durchführt. Analoges gilt für die Berechnung der Kurven $\mathbf{x}(s, t)$ für $t = \text{const}$. Jetzt wird der Algorithmus von de Casteljau zuerst auf die $(n + 1)$ Zeilen der Bézier-Punktematrix ausgeführt, um so die Bézier-Punkte der Flächenkurve $t = \text{const}$ zu erhalten.

Beispiel 3.17. Wir wollen eine einfache glatte Bézier-Fläche für 5×4 Bézier-Punkte erzeugen. Die drei Komponenten der Bézier-Punktematrix sind

$$\begin{pmatrix} 0.0 & 0.333 & 0.666 & 1.0 \\ 0.0 & 0.333 & 0.666 & 1.0 \\ 0.0 & 0.333 & 0.666 & 1.0 \\ 0.0 & 0.333 & 0.666 & 1.0 \\ 0.0 & 0.333 & 0.666 & 1.0 \end{pmatrix} \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.75 & 0.75 & 0.75 & 0.75 \\ 1.0 & 1.0 & 1.0 & 1.0 \end{pmatrix} \begin{pmatrix} 0.0 & 0.9 & 0.7 & 0.0 \\ 1.0 & 2.4 & 2.2 & 1.0 \\ 2.2 & 3.4 & 3.1 & 2.1 \\ 1.9 & 2.9 & 2.7 & 1.9 \\ 1.7 & 2.4 & 2.3 & 1.8 \end{pmatrix}.$$

In Abb. 3.25 sind die Bézier-Punkte zusammen mit dem zugehörigen Bézier-Netz und die Flächenkurven für $s = 0, 1/15, 2/15, \dots, 1.0$ sowie für $t = 0, 1/11, 2/11, \dots, 1.0$ des resultierenden Bézier-Flächensegments dargestellt. Die Aussagen von Satz 3.20 wie auch die früheren Aussagen über Tangentenrichtungen an die Bézier-Kurven des Randes werden ersichtlich. \triangle

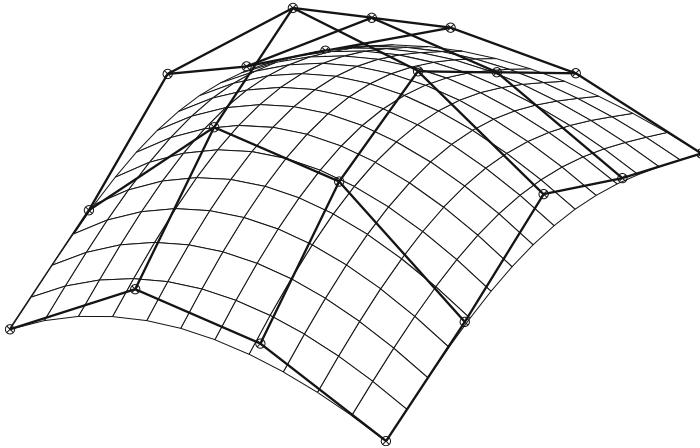


Abb. 3.25 Bézier-Flächensegment zu den 5×4 -Punktematrizen.

Größere Flächen mit vorgegebenen Eigenschaften werden aus einzelnen Bézier-Flächensegmenten geeignet zusammengesetzt. Hier sind – insbesondere für einen glatten C^1 -Übergang – eine Reihe von Bedingungen zu erfüllen, auf die wir nicht mehr eingehen wollen. Dies und manches mehr zu Kurven und Flächen findet sich in Büchern zur geometrischen Datenverarbeitung (*computational geometry*), etwa in [Bar 88b, Bar 95, Bun 02, Dew 88, Enc 96, Far 94, Hos 92].

3.6 Gauß-Approximation

Der wahrscheinlichste Wert einer beobachteten Größe hat die Eigenschaft, dass die Summe der quadratischen Abweichungen der Beobachtungen von diesem Wert in ihm ihr Minimum findet. (Legendre, 1752–1833)

Zunächst wird die allgemeine Gauß-Approximation hergeleitet, die dann sowohl diskret, d.h. auf Datentabellen, als auch kontinuierlich, d.h. auf Funktionen, angewendet wird.

Wir formulieren die Aufgabe in einem Hilbert-Raum H . Das ist ein Raum, in dem ein Skalarprodukt (\cdot, \cdot) und eine zugehörige Norm $\|\cdot\|$ definiert sind. Zugehörig ist eine Norm, für die $(f, f) = \|f\|^2 \forall f \in H$ gilt. Wir betrachten ein in H linear unabhängiges Funktionensystem $\{\varphi_0, \varphi_1, \dots, \varphi_n\}$.

Gegeben sei ein Element $f \in H$.

Gesucht ist eine lineare Approximation g von f im quadratischen Mittel mit Hilfe des gegebenen Funktionensystems. Die gesuchte Funktion g wird repräsentiert durch ihre Koeffizienten α_j in der Linearkombination

$$g(x) := g(x; \alpha_0, \alpha_1, \dots, \alpha_n) := \sum_{j=0}^n \alpha_j \varphi_j(x). \quad (3.135)$$

Approximation im quadratischen Mittel heißt dann:

Bestimme die Koeffizienten α_j so, dass

$$F(\alpha_0, \alpha_1, \dots, \alpha_n) := \|f - g\|^2 \rightarrow \min_{\alpha_j}. \quad (3.136)$$

Satz 3.23. *Die Ansatzfunktionen φ_j , $j = 0, \dots, n$, seien linear unabhängig.*

Dann ist die Minimalaufgabe (3.136) eindeutig lösbar und ihre Lösung ist die Lösung des linearen Gleichungssystems

$$\begin{pmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ (\varphi_n, \varphi_0) & \cdots & \cdots & (\varphi_n, \varphi_n) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} (f, \varphi_0) \\ (f, \varphi_1) \\ \vdots \\ (f, \varphi_n) \end{pmatrix}. \quad (3.137)$$

Es wird Normalgleichungssystem (NGS) genannt.

Seine Koeffizientenmatrix ist positiv definit und symmetrisch.

Beweis. Notwendig für ein Minimum ist, dass die Ableitungen der Funktion F aus (3.136) nach den α_j verschwinden. Es ist

$$\begin{aligned} F(\alpha_0, \dots, \alpha_n) &= \|f - g\|_2^2 = (f - g, f - g) \\ &= (f - \sum_{j=0}^n \alpha_j \varphi_j, f - \sum_{j=0}^n \alpha_j \varphi_j) \\ &= (f, f) - 2 \sum_j \alpha_j (f, \varphi_j) + \sum_j \sum_k \alpha_j \alpha_k (\varphi_j, \varphi_k). \end{aligned}$$

Die notwendigen Bedingungen für ein Minimum sind also

$$\frac{1}{2} \frac{\partial F}{\partial \alpha_k} = -(f, \varphi_k) + \sum_{j=0}^n \alpha_j (\varphi_j, \varphi_k) = 0, \quad k = 0, 1, \dots, n.$$

Hinreichend für die Existenz eines Minimums ist, dass die Matrix der zweiten Ableitungen

$$\frac{\partial^2 F}{\partial \alpha_j \partial \alpha_k} = ((\varphi_j, \varphi_k))$$

positiv definit ist. Dazu wird zunächst gezeigt:

Lemma 3.24. Das Funktionensystem $\{\varphi_j\}_{j=0}^n$ ist genau dann linear abhängig, wenn die Determinante der Matrix aus (3.137) gleich null ist:

$$\det(\varphi_j, \varphi_k) = 0$$

Beweis. Wenn die φ_j linear abhängig sind, dann gibt es einen Koeffizientensatz α_j , sodass $\sum_j \alpha_j \varphi_j = 0$ (Nullelement in H) mit mindestens einem $\alpha_k \neq 0$. Das heißt, dass das homogene lineare Gleichungssystem

$$\sum_j (\alpha_j \varphi_j, \varphi_k) = \sum_j \alpha_j (\varphi_j, \varphi_k) = 0, \quad k = 0, \dots, n \quad (3.138)$$

eine nicht-triviale Lösung hat. Daraus folgt $\det((\varphi_j, \varphi_k)) = 0$.

Ist umgekehrt diese Determinante gleich null, dann hat das homogene lineare Gleichungssystem (3.138) eine nicht-triviale Lösung $\{\alpha_j\}$: $\sum_j \alpha_j (\varphi_j, \varphi_k) = 0$. Somit folgt:

$$\sum_j \sum_k \alpha_j (\varphi_j, \varphi_k) \alpha_k = (\sum_j \alpha_j \varphi_j, \sum_k \alpha_k \varphi_k) = \left\| \sum_k \alpha_k \varphi_k \right\|^2 = 0,$$

und daraus folgt $\sum_k \alpha_k \varphi_k = 0$ in H . Das bedeutet, dass die $\{\varphi_j\}_{j=0}^n$ linear abhängig sind. \square

Aus der linearen Unabhängigkeit des Funktionensystems $\{\varphi_j\}$ folgt also, dass die Matrix (φ_j, φ_k) regulär ist. Deshalb gibt es genau eine Lösung des NGS. Da die Matrix darüber hinaus positiv definit ist (Übung!), repräsentiert die Lösung des NGS ein Minimum, d.h. das Minimalproblem (3.136) ist eindeutig lösbar. \square

Trotz der eindeutigen Existenz eines Minimums kann die numerische Lösung der Approximationsaufgabe noch instabil sein, siehe dazu die Bemerkung am Ende des Abschnitts 3.6.2.

3.6.1 Diskrete Gauß-Approximation

Wenn eine Datentabelle (x_i, f_i) , $i = 1, 2, \dots, m$, approximiert werden soll, dann ist der Hilbert-Raum $H = \mathbb{R}^m$ und als Skalarprodukt wird ein gewichtetes euklidisches Skalarprodukt gewählt

$$(f, g) := \sum_{i=1}^m f_i g(x_i) \omega_i, \quad \omega_i > 0. \quad (3.139)$$

Hier ist zu beachten, dass jede Funktion $g(x)$, die an allen Stellen x_i , $i = 1, 2, \dots, m$ verschwindet, das Nullelement in H darstellt.

Die Aufgabe lautet also jetzt:

Gegeben seien Stützstellen x_i , Funktionswerte f_i und positive Gewichte ω_i :

$$(x_i, f_i, \omega_i), \quad i = 1, \dots, m.$$

Gesucht sind die Koeffizienten α_j , $j = 0, 1, \dots, n$, des linearen Ansatzes

$$g(x) := g(x; \alpha_0, \alpha_1, \dots, \alpha_n) := \sum_{j=0}^n \alpha_j \varphi_j(x), \quad (3.140)$$

so dass

$$F(\alpha_0, \alpha_1, \dots, \alpha_n) := \sum_{i=1}^m (f_i - g(x_i))^2 \omega_i \rightarrow \min_{\alpha_j}. \quad (3.141)$$

Die gesuchte Lösung ist die Lösung des NGS (3.137). Die Skalarprodukte, die die Elemente der Koeffizientenmatrix des NGS darstellen, sind die Summen

$$(\varphi_j, \varphi_k) = \sum_{i=1}^m \varphi_j(x_i) \varphi_k(x_i) \omega_i. \quad (3.142)$$

Die Punktmenge $\{x_i\}$ kann oft so gewählt werden, dass die Orthogonalität des Funktionensystems $\{\varphi_k\}$ für das eingeführte Skalarprodukt gilt:

$$\sum_{i=1}^m \varphi_j(x_i) \varphi_k(x_i) \omega_i = 0 \quad \text{für alle } j, k = 0, \dots, n \quad \text{mit } j \neq k.$$

Beispiel 3.18. Lineare Ausgleichung wird die diskrete Gauß-Approximation mit einem linearen Polynom genannt.

$$g(x; \alpha_0, \alpha_1) = \alpha_0 + \alpha_1 x, \quad \omega_i = 1. \quad (3.143)$$

Hier sind also $\varphi_0(x) = 1$, $\varphi_1(x) = x$ und

$$F(\alpha_0, \alpha_1) = \sum_{i=1}^m (f_i - \alpha_0 - \alpha_1 x_i)^2,$$

und es ergibt sich mit (3.142) das Normalgleichungssystem

$$\begin{pmatrix} m & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} \sum f_i \\ \sum x_i f_i \end{pmatrix}.$$

Es ist regulär, wenn $m \geq 2$ ist und es mindestens zwei Werte x_i, x_j mit $x_i \neq x_j$ gibt. Dann ist nämlich $m \sum x_i^2 > (\sum x_i)^2$ (Cauchy-Schwarzsche Ungleichung).

Beispiel

Aus der Datentabelle	x_i	0	0.1	0.2	0.3	0.4	0.5
	f_i	0	0.11	0.22	0.28	0.315	0.565

ergeben sich mit $m = 6$, $\sum x_i = 1.5$, $\sum f_i = 1.49$, $\sum x_i^2 = 0.55$ und $\sum x_i f_i = 0.5475$ die Koeffizienten $\alpha_0 = -0.0016$ und $\alpha_1 = 1.00000$. Also ist die approximierende Funktion

$$g(x) = -0.0016 + x.$$

Die Approximationsgüte kann durch die Fehlerquadratsumme

$$\sum (f_i - g(x_i, \alpha_0, \alpha_1))^2 = 0.0123$$

beschrieben werden. Datenpunkte (\times) und Ausgleichsgerade sind in Abb. 3.26 zu sehen. \triangle

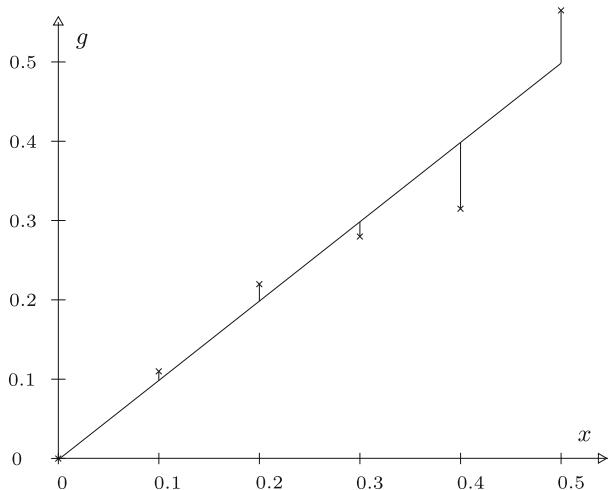


Abb. 3.26 Lineare Ausgleichsrechnung.

3.6.2 Kontinuierliche Gauß-Approximation

Hier wird eine Funktion f in einem Intervall (a, b) durch das Funktionensystem $\{\varphi_i\}$ approximiert. Der Hilbert-Raum H wird zum Funktionenraum L_2 , der durch das Skalarprodukt

$$(f, g) := \int_a^b f(x) g(x) w(x) dx \quad (3.144)$$

festgelegt wird. Dabei ist w eine gegebene positive Gewichtsfunktion: $w(x) > 0$ in (a, b) .

Genauer lautet die Approximationsaufgabe:

Gegeben seien eine auf (a, b) quadratintegrierbare, stückweise stetige Funktion f , und eine in (a, b) integrierbare, positive *Gewichtsfunktion* w .

Gesucht sind die Koeffizienten α_i , $i = 0, 1, \dots, n$, des Funktionenansatzes

$$g(x) := g(x; \alpha_0, \alpha_1, \dots, \alpha_n) := \sum_{i=0}^n \alpha_i \varphi_i(x), \quad (3.145)$$

so dass

$$F(\alpha_0, \alpha_1, \dots, \alpha_n) := \int_a^b (f(x) - g(x))^2 w(x) dx \rightarrow \min_{\alpha_i}. \quad (3.146)$$

Lösung ist wieder die Lösung des NGS (3.137). Allerdings sind jetzt die Elemente der Koeffizientenmatrix des NGS Integrale

$$(\varphi_i, \varphi_j) := \int_a^b \varphi_i(x) \varphi_j(x) w(x) dx \quad (3.147)$$

ebenso wie die rechten Seiten

$$(f, \varphi_j) := \int_a^b f(x) \varphi_j(x) w(x) dx. \quad (3.148)$$

Die Auswertung dieser Integrale stellt einen wesentlichen Teil der Arbeit zur Lösung der Approximationsaufgabe dar. Diese ist besonders dann aufwändig und rundungsfehleranfällig, wenn die Koeffizientenmatrix voll besetzt ist. Deshalb wird fast ausschließlich mit *orthogonalen* Funktionensystemen gearbeitet. Für sie gilt bekanntlich

$$(\varphi_i, \varphi_j) = 0, \quad \text{falls } i \neq j. \quad (3.149)$$

Dadurch wird das NGS ein Diagonalsystem mit den Lösungen

$$\alpha_i = \frac{(f, \varphi_i)}{(\varphi_i, \varphi_i)}, \quad i = 0, 1, \dots, n. \quad (3.150)$$

Wir werden die kontinuierliche Gauß-Approximation mit Polynomen in Abschnitt 3.8 und die mit trigonometrischen Funktionen in Abschnitt 3.7 kennen lernen.

Beispiel 3.19. Wie wichtig der Einsatz orthogonaler Funktionensysteme ist, kann man leicht sehen, wenn man als Funktionensystem die Grundpolynome

$$\varphi_j(x) := x^j, \quad x \in [0, 1], \quad j = 0, 1, \dots, n,$$

und als Gewichtsfunktion die Konstante $w(x) \equiv 1$ wählt. Für die Elemente der Koeffizientenmatrix des NGS ergibt sich damit

$$\int_0^1 \varphi_i(x) \varphi_j(x) w(x) dx = \int_0^1 x^{i+j} dx = \frac{1}{i+j+1}.$$

Das sind aber gerade die Elemente der *Hilbert-Matrix*, von der wir in Aufgabe 2.11 gesehen haben, wie sehr schlecht sie konditioniert ist. \triangle

3.7 Trigonometrische Approximation

3.7.1 Fourier-Reihen

Gegeben sei eine stückweise stetige Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ mit der Periode 2π

$$f(x + 2\pi) = f(x) \quad \text{für alle } x \in \mathbb{R}.$$

(3.151)

Die Funktion $f(x)$ darf Sprungstellen aufweisen, wenn für eine Unstetigkeitsstelle x_0 die Grenzwerte y_0^- und y_0^+

$$\lim_{h \rightarrow +0} f(x_0 - h) = y_0^-, \quad \lim_{h \rightarrow +0} f(x_0 + h) = y_0^+ \quad (3.152)$$

existieren und endlich sind. Die Funktion $f(x)$ soll durch eine Linearkombination der (2π) -periodischen trigonometrischen Funktionen

$$1, \cos(x), \sin(x), \cos(2x), \sin(2x), \dots, \cos(nx), \sin(nx) \quad (3.153)$$

in der Form

$$g_n(x) = \frac{1}{2}a_0 + \sum_{k=1}^n \{a_k \cos(kx) + b_k \sin(kx)\} \quad (3.154)$$

im *quadratischen Mittel* approximiert werden, so dass gilt

$$\|g_n(x) - f(x)\|_2 := \left\{ \int_{-\pi}^{\pi} \{g_n(x) - f(x)\}^2 dx \right\}^{1/2} \rightarrow \min \quad (3.155)$$

Das entspricht bis auf die Nummerierung der Koeffizienten der kontinuierlichen Gauß-Approximation (3.146).

Satz 3.25. Die trigonometrischen Funktionen (3.153) bilden für das Intervall $[-\pi, \pi]$ ein System von paarweise orthogonalen Funktionen. Es gelten folgende Beziehungen

$$\int_{-\pi}^{\pi} \cos(jx) \cos(kx) dx = \begin{cases} 0 & \text{für alle } j \neq k \\ 2\pi & \text{für } j = k = 0 \\ \pi & \text{für } j = k > 0 \end{cases} \quad (3.156)$$

$$\int_{-\pi}^{\pi} \sin(jx) \sin(kx) dx = \begin{cases} 0 & \text{für alle } j \neq k, j, k > 0 \\ \pi & \text{für } j = k > 0 \end{cases} \quad (3.157)$$

$$\int_{-\pi}^{\pi} \cos(jx) \sin(kx) dx = 0 \quad \text{für alle } j \geq 0, k > 0 \quad (3.158)$$

Beweis. Auf Grund von bekannten trigonometrischen Identitäten gilt

$$\int_{-\pi}^{\pi} \cos(jx) \cos(kx) dx = \frac{1}{2} \int_{-\pi}^{\pi} [\cos\{(j+k)x\} + \cos\{(j-k)x\}] dx. \quad (3.159)$$

Für $j \neq k$ ergibt sich daraus

$$\frac{1}{2} \left[\frac{1}{j+k} \sin\{(j+k)x\} + \frac{1}{j-k} \sin\{(j-k)x\} \right]_{-\pi}^{\pi} = 0$$

die erste Relation (3.156). Für $j = k > 0$ folgt aus (3.159)

$$\frac{1}{2} \left[\frac{1}{j+k} \sin\{(j+k)x\} + x \right]_{-\pi}^{\pi} = \pi$$

der dritte Fall von (3.156), während die zweite Beziehung von (3.156) trivial ist. Die Aussage (3.157) zeigt man völlig analog auf Grund der Identität

$$\int_{-\pi}^{\pi} \sin(jx) \sin(kx) dx = -\frac{1}{2} \int_{-\pi}^{\pi} [\cos\{(j+k)x\} - \cos\{(j-k)x\}] dx,$$

während (3.158) aus der Tatsache folgt, dass der Integrand als Produkt einer geraden und einer ungeraden Funktion ungerade ist, so dass der Integralwert verschwindet. \square

Die Orthogonalitätsrelationen (3.156) bis (3.158) gelten wegen der Periodizität der trigonometrischen Funktionen selbstverständlich für irgendein Intervall der Länge 2π . Für ein beliebiges endliches Intervall $[a, b]$ führt die lineare Transformation $t = 2\pi(x-a)/(b-a)$ auf das Funktionensystem 1, $\cos(kt)$, $\sin(kt)$,

Wegen der Orthogonalität ist das NGS diagonal und seine Lösungen sind die Koeffizienten $a_i = (f, \varphi_i)/(\varphi_i, \varphi_i)$ aus (3.150). Das ergibt hier mit (3.156) bis (3.158) auf Grund der anderen Bezeichnungsweise und Nummerierung die gesuchten Koeffizienten

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad k = 0, 1, \dots, n,$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx, \quad k = 1, 2, \dots, n.$$

(3.160)

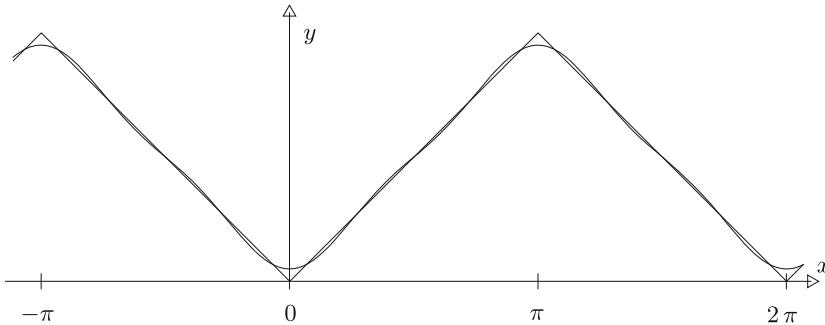
Sie heißen die *Fourier-Koeffizienten* der (2π) -periodischen Funktion $f(x)$ und die mit ihnen gebildete Funktion $g_n(x)$ (3.154) das *Fourier-Polynom*. Die Fourier-Koeffizienten und das zugehörige Fourier-Polynom können für beliebig großes n definiert werden, so dass zu jeder (2π) -periodischen, stückweise stetigen Funktion $f(x)$ durch Grenzübergang die unendliche *Fourier-Reihe*

$$g(x) := \frac{1}{2} a_0 + \sum_{k=1}^{\infty} \{a_k \cos(kx) + b_k \sin(kx)\} \quad (3.161)$$

formal gebildet werden kann. Ohne Beweis [Cou 93, Heu 08, Smi 90] zitieren wir den

- Satz 3.26.** *Es sei $f(x)$ eine (2π) -periodische, stückweise stetige Funktion mit stückweise stetiger erster Ableitung. Dann konvergiert die zugehörige Fourier-Reihe $g(x)$ (3.161) gegen*
- a) *den Wert $f(x_0)$, falls $f(x)$ an der Stelle x_0 stetig ist,*
 - b) *$\frac{1}{2}\{y_0^- + y_0^+\}$ mit den Grenzwerten (3.152), falls $f(x)$ an der Stelle x_0 eine Sprungstelle besitzt.*

Die Voraussetzungen des Satzes 3.26 schließen solche stückweise stetigen Funktionen aus, deren Graph eine vertikale Tangente aufweist wie beispielsweise $f(x) = \sqrt{|x|}$ für $x = 0$.

Abb. 3.27 Periodische Dachkurve mit approximierendem Fourier-Polynom $g_3(x)$.

Beispiel 3.20. Die (2π) -periodische Funktion $f(x)$ sei im Grundintervall $[-\pi, \pi]$ definiert als

$$f(x) = |x|, \quad -\pi \leq x \leq \pi. \quad (3.162)$$

Sie wird wegen ihres Graphen als Dachfunktion bezeichnet (vgl. Abb. 3.27). Sie ist eine stetige Funktion, und da sie gerade ist, sind alle Koeffizienten $b_k = 0$. Zudem kann die Berechnung der a_k vereinfacht werden.

$$\begin{aligned} a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} |x| dx = \frac{2}{\pi} \int_0^{\pi} x dx = \pi \\ a_k &= \frac{2}{\pi} \int_0^{\pi} x \cos(kx) dx = \frac{2}{\pi} \left[\frac{1}{k} x \sin(kx) \Big|_0^\pi - \frac{1}{k} \int_0^{\pi} \sin(kx) dx \right] \\ &= \frac{2}{\pi k^2} \cos(kx) \Big|_0^\pi = \frac{2}{\pi k^2} [(-1)^k - 1], \quad k > 0 \end{aligned}$$

Die zugehörige Fourier-Reihe lautet deshalb

$$g(x) = \frac{1}{2}\pi - \frac{4}{\pi} \left\{ \frac{\cos(x)}{1^2} + \frac{\cos(3x)}{3^2} + \frac{\cos(5x)}{5^2} + \dots \right\}. \quad (3.163)$$

In Abb. 3.27 ist das Fourier-Polynom $g_3(x)$ eingezeichnet. Es approximiert die Funktion $f(x)$ bereits recht gut. Da f die Voraussetzungen des Satzes 3.26 erfüllt, gilt nach Satz 3.26 $g(0) = 0$ und es ergibt sich

$$\frac{\pi^2}{8} = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \dots$$

△

Beispiel 3.21. Die (2π) -periodische Funktion $f(x)$ sei im Intervall $(0, 2\pi)$ definiert durch

$$f(x) = x^2, \quad 0 < x < 2\pi. \quad (3.164)$$

Sie besitzt für $x_k = 2\pi k, (k \in \mathbb{Z})$ Sprungstellen, erfüllt aber die Voraussetzungen von Satz 3.26. Die zugehörigen Fourier-Koeffizienten erhält man mit Hilfe partieller Integration.

$$a_0 = \frac{1}{\pi} \int_0^{2\pi} x^2 dx = \frac{8\pi^2}{3}$$

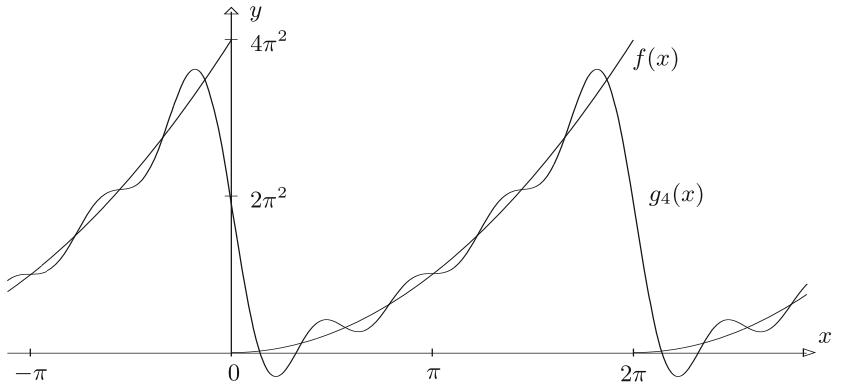


Abb. 3.28 Periodische Funktion mit Sprungstellen.

$$a_k = \frac{1}{\pi} \int_0^{2\pi} x^2 \cos(kx) dx = \frac{4}{k^2}, \quad k = 1, 2, \dots$$

$$b_k = \frac{1}{\pi} \int_0^{2\pi} x^2 \sin(kx) dx = -\frac{4\pi}{k}, \quad k = 1, 2, \dots$$

Als Fourier-Reihe ergibt sich damit

$$g(x) = \frac{4\pi^2}{3} + \sum_{k=1}^{\infty} \left\{ \frac{4}{k^2} \cos(kx) - \frac{4\pi}{k} \sin(kx) \right\}. \quad (3.165)$$

In Abb. 3.28 ist zur Illustration der Approximation von $f(x)$ das Fourier-Polynom $g_4(x)$ angegeben. Die vorhandenen Sprungstellen bewirken eine lokal schlechtere Konvergenz als bei Beispiel 3.20. \triangle

Im allgemeinen Fall ergeben sich für die Fourier-Koeffizienten a_k und b_k nach (3.160) keine geschlossenen Formeln. Dann müssen sie durch numerische Integration bestimmt werden. Dazu wählen wir als Integrationsformel die *Trapezregel*, denn sie liefert bei äquidistanten, fest vorgegebenen Integrationsstützstellen einen verschiebungsinvarianten Wert des Integrals. Zudem besitzt sie zur approximativen Berechnung eines Integrals einer periodischen Funktion über ein Periodenintervall besondere Eigenschaften (vgl. Abschnitt 7.3). Wird das Intervall $[0, 2\pi]$ in N Teilintervalle unterteilt, so dass für die Schrittweite h und die Integrationsstützstellen x_j

$$h = \frac{2\pi}{N}, \quad x_j = hj = \frac{2\pi}{N}j, \quad j = 0, 1, 2, \dots, N$$

(3.166)

gelten, dann erhalten wir aus der Trapezregel (7.13)

$$a_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(kx) dx$$

$$\approx \frac{1}{\pi} \frac{2\pi}{2N} \left\{ f(x_0) \cos(kx_0) + 2 \sum_{j=1}^{N-1} f(x_j) \cos(kx_j) + f(x_N) \cos(kx_N) \right\}.$$

Berücksichtigen wir die (2π) -Periodizität von $f(x) \cos(kx)$, so ergeben sich für die a_k und analog für die b_k die Näherungswerte

$$\begin{aligned} a_k^* &:= \frac{2}{N} \sum_{j=1}^N f(x_j) \cos(kx_j), \quad k = 0, 1, 2, \dots, \\ b_k^* &:= \frac{2}{N} \sum_{j=1}^N f(x_j) \sin(kx_j), \quad k = 1, 2, 3, \dots. \end{aligned} \tag{3.167}$$

Fourier-Polynome, welche mit den Koeffizienten a_k^* und b_k^* gebildet werden, haben Eigenarten, die wir jetzt herleiten wollen. Als Vorbereitung dazu benötigen wir die zu (3.156) bis (3.158) analogen diskreten Orthogonalitätsrelationen der trigonometrischen Funktionen.

Satz 3.27. Für die diskreten Stützstellen x_j (3.166) gelten

$$\sum_{j=1}^N \cos(kx_j) = \begin{cases} 0 & \text{falls } k/N \notin \mathbb{Z} \\ N & \text{falls } k/N \in \mathbb{Z} \end{cases} \tag{3.168}$$

$$\sum_{j=1}^N \sin(kx_j) = 0 \quad \text{für alle } k \in \mathbb{Z}. \tag{3.169}$$

Beweis. Wir betrachten die komplexe Summe

$$S := \sum_{j=1}^N \{\cos(kx_j) + i \sin(kx_j)\} = \sum_{j=1}^N e^{ikx_j} = \sum_{j=1}^N e^{ijkh}, \tag{3.170}$$

welche eine endliche geometrische Reihe mit dem (komplexen) Quotienten $q := e^{ikh} = e^{2\pi ik/N}$ darstellt. Eine Fallunterscheidung ist nötig, um ihren Wert zu bestimmen.

Ist $k/N \notin \mathbb{Z}$, dann ist $q \neq 1$, und die Summenformel ergibt

$$S = e^{ikh} \frac{e^{ikhN} - 1}{e^{ikh} - 1} = e^{ikh} \frac{e^{2\pi ki} - 1}{e^{ikh} - 1} = 0, \quad k/N \notin \mathbb{Z}.$$

Ist hingegen $k/N \in \mathbb{Z}$, so folgt wegen $q = 1$, dass $S = N$ ist. Aus diesen beiden Ergebnissen folgen aus dem Realteil und dem Imaginärteil von S (3.170) die Behauptungen (3.168) und (3.169). \square

Satz 3.28. Die trigonometrischen Funktionen (3.153) erfüllen für die äquidistanten Stützstellen x_j (3.166) die diskreten Orthogonalitätsrelationen

$$\sum_{j=1}^N \cos(kx_j) \cos(lx_j) = \begin{cases} 0, & \text{falls } \frac{k+l}{N} \notin \mathbb{Z} \text{ und } \frac{k-l}{N} \notin \mathbb{Z} \\ \frac{N}{2}, & \text{falls entweder } \frac{k+l}{N} \in \mathbb{Z} \text{ oder } \frac{k-l}{N} \in \mathbb{Z} \\ N, & \text{falls } \frac{k+l}{N} \in \mathbb{Z} \text{ und } \frac{k-l}{N} \in \mathbb{Z} \end{cases} \quad (3.171)$$

$$\sum_{j=1}^N \sin(kx_j) \sin(lx_j) = \begin{cases} 0, & \text{falls } \frac{k+l}{N} \notin \mathbb{Z} \text{ und } \frac{k-l}{N} \notin \mathbb{Z} \\ & \text{oder } \frac{k+l}{N} \in \mathbb{Z} \text{ und } \frac{k-l}{N} \in \mathbb{Z} \\ -\frac{N}{2}, & \text{falls } \frac{k+l}{N} \in \mathbb{Z} \text{ und } \frac{k-l}{N} \notin \mathbb{Z} \\ \frac{N}{2}, & \text{falls } \frac{k+l}{N} \notin \mathbb{Z} \text{ und } \frac{k-l}{N} \in \mathbb{Z} \end{cases} \quad (3.172)$$

$$\sum_{j=1}^N \cos(kx_j) \sin(lx_j) = 0, \quad \text{für alle } k, l \in \mathbb{N} \quad (3.173)$$

Beweis. Zur Verifikation der Behauptungen sind die Identitäten für trigonometrische Funktionen

$$\cos(kx_j) \cos(lx_j) = \frac{1}{2}[\cos\{(k+l)x_j\} + \cos\{(k-l)x_j\}]$$

$$\sin(kx_j) \sin(lx_j) = \frac{1}{2}[\cos\{(k-l)x_j\} - \cos\{(k+l)x_j\}]$$

$$\cos(kx_j) \sin(lx_j) = \frac{1}{2}[\sin\{(k+l)x_j\} - \sin\{(k-l)x_j\}]$$

und dann die Aussagen von Satz 3.27 anzuwenden, um die Beziehungen (3.171) bis (3.173) zu erhalten. \square

Die angeneherte Bestimmung der Fourier-Koeffizienten a_k und b_k mit der Trapezregel hat zu der Formel (3.167) für die Näherungen a_k^* und b_k^* geführt. Wenn wir (3.167) als Skalarprodukte (f, φ_j) auffassen mit $\varphi_{2k}(x) = \cos(kx)$ bzw. $\varphi_{2k+1}(x) = \sin(kx)$, $k = 0, 1, 2, \dots$, und wenn wir die in den Sätzen 3.27 und 3.28 gezeigte diskrete Orthogonalität der Ansatzfunktionen für dieses Skalarprodukt berücksichtigen, dann hat sich mit der numerischen Integration mit der Trapezregel der Übergang von der kontinuierlichen zur diskreten trigonometrischen Approximation ergeben, und das unter Beibehaltung der Orthogonalität. Über die Approximationsqualität bei Verwendung der angeneherten Fourier-Koeffizienten a_k^* und b_k^* geben die folgenden Sätze und Beispiele Auskunft.

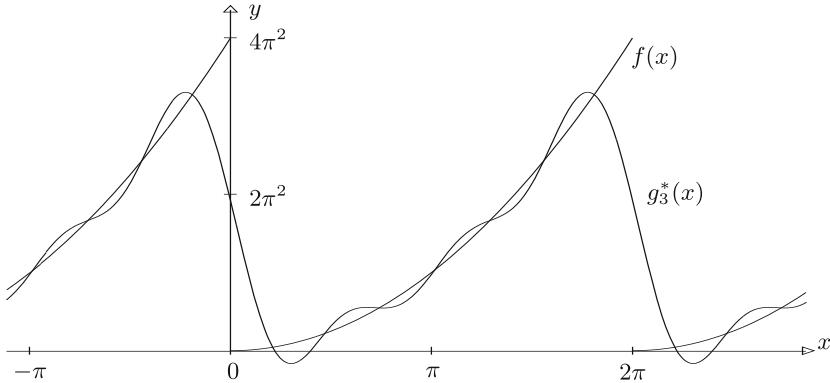


Abb. 3.29 Approximation im diskreten quadratischen Mittel.

Satz 3.29. Es sei $N = 2n, n \in \mathbb{N}^*$. Das Fourier-Polynom

$$g_m^*(x) := \frac{1}{2}a_0^* + \sum_{k=1}^m \{a_k^* \cos(kx) + b_k^* \sin(kx)\} \quad (3.174)$$

vom Grad $m < n$ mit den Koeffizienten (3.167) approximiert die Funktion $f(x)$ im diskreten quadratischen Mittel der N Stützstellen x_j (3.166) derart, dass die Summe der Quadrate der Abweichungen

$$F := \sum_{j=1}^N \{g_m^*(x_j) - f(x_j)\}^2 \quad (3.175)$$

minimal ist.

Beispiel 3.22. Für die (2π) -periodisch fortgesetzte Funktion $f(x) = x^2, 0 < x < 2\pi$, für welche an den Sprungstellen $f(0) = f(2\pi) = 2\pi^2$ festgesetzt sei, ergeben sich für $N = 12$ die Koeffizienten

$$\begin{aligned} a_0^* &\doteq 26.410330, & a_1^* &\doteq 4.092652, & a_2^* &\doteq 1.096623, & a_3^* &\doteq 0.548311, \\ b_1^* &\doteq -12.277955, & b_2^* &\doteq -5.698219, & b_3^* &\doteq -3.289868. \end{aligned}$$

Das zugehörige Fourier-Polynom $g_3^*(x)$ ist zusammen mit $f(x)$ in Abb. 3.29 dargestellt. \triangle

Für eine bestimmte Wahl von m wird die Approximation zur Interpolation. Dieses Ergebnis wollen wir noch ohne Beweis anfügen (siehe [Sch 97]).

Satz 3.30. Es sei $N = 2n, n \in \mathbb{N}^*$. Das spezielle Fourier-Polynom

$$g_n^*(x) := \frac{1}{2}a_0^* + \sum_{k=1}^{n-1} \{a_k^* \cos(kx) + b_k^* \sin(kx)\} + \frac{1}{2}a_n^* \cos(nx) \quad (3.176)$$

mit den Koeffizienten (3.167) ist das eindeutige, interpolierende Fourier-Polynom zu den Stützstellen x_j (3.166) mit den Stützstellen $f(x_j), j = 1, 2, \dots, N$.

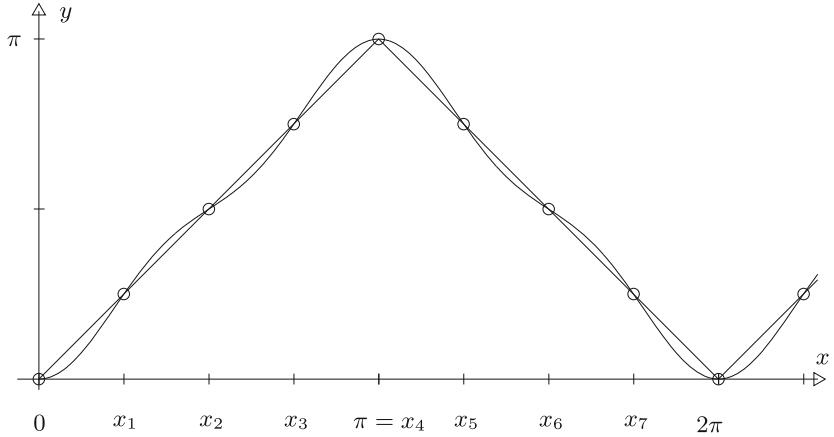


Abb. 3.30 Periodische Dachkurve mit interpolierendem Fourier-Polynom $g_4(x)$.

Als Ergänzung zu Satz 3.30 sei noch bemerkt, dass der Funktionswert $f(x_j)$ in Anlehnung an Satz 3.26 im Fall einer Sprungstelle x_j als das arithmetische Mittel der beiden Grenzwerte y_j^- und y_j^+ festzulegen ist. Auch für ungerades N kann ein interpolierendes Fourier-Polynom mit den Koeffizienten a_k^* und b_k^* gebildet werden [You 90].

Beispiel 3.23. Für die Dachfunktion $f(x)$ (3.162) ergeben sich für $N = 8$, $h = \pi/4$, $x_j = \pi j/4$, $j = 1, 2, \dots, 8$, die Werte

$$a_0^* = \pi, \quad a_1^* \doteq -1.34076, \quad a_2^* = 0, \quad a_3^* \doteq -0.230038, \quad a_4^* = 0, \quad b_1^* = b_2^* = b_3^* = 0.$$

Das interpolierende Fourier-Polynom $g_4^*(x)$ lautet

$$g_4^*(x) \doteq 1.57080 - 1.34076 \cos(x) - 0.230038 \cos(3x)$$

und ist in Abb. 3.30 dargestellt. Die Approximation der Dachkurve ist im Vergleich zu Abb. 3.27 wegen der Interpolationseigenschaft natürlich verschieden. \triangle

Es bleibt die Frage des Unterschiedes zwischen kontinuierlicher und diskreter trigonometrischer Approximation bzw. nach den Fehlern $|a_k^* - a_k|$ und $|b_k^* - b_k|$. Die Antwort findet sich im folgenden Satz, auf dessen Beweis wir verzichten wollen.

Satz 3.31. Es sei $f(x)$ eine 2π -periodische Funktion. Sie erfülle die Voraussetzungen von Satz 3.26. Weiter sei $N = 2n$, $n \in \mathbb{N}^*$. Wird $f(x_j)$ an einer Sprungstelle x_j als arithmetisches Mittel der beiden Grenzwerte (3.152) definiert, dann gelten für die Koeffizienten a_k^* und b_k^* die Darstellungen

$$a_k^* = a_k + a_{N-k} + a_{N+k} + a_{2N-k} + a_{2N+k} + \dots, \quad (0 \leq k \leq n), \quad (3.177)$$

$$b_k^* = b_k - b_{N-k} + b_{N+k} - b_{2N-k} + b_{2N+k} + \dots, \quad (1 \leq k \leq n-1), \quad (3.178)$$

$$|a_k^* - a_k| \leq \sum_{\mu=1}^{\infty} \{|a_{\mu N-k}| + |a_{\mu N+k}|\}, \quad (0 \leq k \leq n), \quad (3.179)$$

$$|b_k^* - b_k| \leq \sum_{\mu=1}^{\infty} \{|b_{\mu N-k}| + |b_{\mu N+k}|\}, \quad (1 \leq k \leq n-1). \quad (3.180)$$

Die Fehler lassen sich daher zu gegebenem N abschätzen, falls bekannt ist, wie die Fourier-Koeffizienten a_k und b_k für eine gegebene Funktion $f(x)$ gegen null konvergieren. Umgekehrt lässt sich in diesem Fall der Wert N schätzen, der nötig ist, damit zu vorgegebenem $\varepsilon > 0$ $|a_k^* - a_k| \leq \varepsilon$ und $|b_k^* - b_k| \leq \varepsilon$ für $k = 0, 1, 2, \dots, m < n$ gelten.

Beispiel 3.24. Für die Dachfunktion $f(x)$ (3.162) wurde für die Fourier-Koeffizienten im Beispiel 3.20 gefunden

$$a_0 = \pi; \quad a_k = \begin{cases} -\frac{4}{\pi k^2}, & \text{falls } k \text{ ungerade,} \\ 0, & \text{falls } k > 0 \text{ gerade.} \end{cases} \quad (3.181)$$

Mit $N = 8$ (vgl. Beispiel 3.23) gelten beispielsweise

$$\begin{aligned} a_0^* &= a_0 + 2(a_8 + a_{16} + a_{24} + a_{32} + \dots) = a_0, \\ a_1^* &= a_1 + a_7 + a_9 + a_{15} + a_{17} + \dots = a_1 - \frac{4}{\pi} \left\{ \frac{1}{49} + \frac{1}{81} + \frac{1}{225} + \frac{1}{289} + \dots \right\}. \end{aligned}$$

Es soll nun der Wert von N abgeschätzt werden, so dass die ersten von null verschiedenen Koeffizienten a_k durch die entsprechenden a_k^* mit einem maximalen, absoluten Fehler von $\varepsilon = 10^{-6}$ approximiert werden. Für k ungerade und $k \ll N$ gilt

$$\begin{aligned} a_k^* - a_k &= -\frac{4}{\pi} \left\{ \frac{1}{(N-k)^2} + \frac{1}{(N+k)^2} + \frac{1}{(2N-k)^2} + \frac{1}{(2N+k)^2} + \dots \right\} \\ &= -\frac{4}{\pi} \left\{ \frac{2N^2+2k^2}{(N^2-k^2)^2} + \frac{8N^2+2k^2}{(4N^2-k^2)^2} + \dots \right\} \\ &\approx -\frac{8}{\pi} \left\{ \frac{1}{N^2} + \frac{1}{(2N)^2} + \frac{1}{(3N)^2} + \dots \right\} \\ &= -\frac{8}{\pi N^2} \left\{ \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \dots \right\} = -\frac{8}{\pi N^2} \cdot \frac{\pi^2}{6} = -\frac{4\pi}{3N^2}. \end{aligned}$$

Die Summe der Kehrwerte der Quadrate der ganzen Zahlen ergibt sich aus der Fourier-Reihe (3.165) für $x = 0$. Aus der Bedingung $|a_k^* - a_k| \leq \varepsilon = 10^{-6}$ folgt $N > 2046$. \triangle

3.7.2 Effiziente Berechnung der Fourier-Koeffizienten

Wir haben die Formeln (3.167) zur genäherten Berechnung der Fourier-Koeffizienten a_k und b_k einer (2π) -periodischen Funktion $f(x)$ verwendet. Die Summen (3.167) treten auch im Zusammenhang mit der *diskreten Fourier-Transformation* auf, die von großer Bedeutung ist beispielsweise in der Physik, Elektrotechnik, Bildverarbeitung und Statistik, siehe etwa [Bri 95, Pic 86]. Dabei ist N oft sehr groß, und deshalb ist es besonders wichtig, diese Summen mit möglichst geringem Rechenaufwand zu bestimmen. Die nahe liegende, direkte Realisierung der Berechnung der Summen erfordert für die Bestimmung der $N = 2n$ Koeffizienten a_k^* , $k = 0, 1, \dots, n$, und b_k^* , $k = 1, 2, \dots, n-1$, etwa N^2 Multiplikationen und N^2 trigonometrische Funktionsauswertungen. Für größere Werte von N ($N \gg 1000$) wird der Rechenaufwand prohibitiv groß. Wir werden einen Algorithmus darstellen, der

die diskrete Fourier-Transformation sehr effizient durchführt. Interessant ist auch der *Algorithmus von Reinsch* [Sau 68, Sto 07], welcher die Fourier-Koeffizienten etwas weniger effizient, jedoch numerisch stabil und mit minimalem Speicherbedarf liefert. Hinweisen wollen wir noch auf einen historischen Algorithmus, den Runge schon 1903 und 1905 vorstellt, [Run 03, Run 05, Run 24, JE 82, Sch 97].

Die schnelle Fourier-Transformation

Zur Berechnung der Summen

$$\boxed{\begin{aligned} a'_k &:= \sum_{j=0}^{N-1} f(x_j) \cos(kx_j), \quad k = 0, 1, 2, \dots, \frac{N}{2}, \\ b'_k &:= \sum_{j=0}^{N-1} f(x_j) \sin(kx_j), \quad k = 1, 2, \dots, \frac{N}{2} - 1, \end{aligned}} \quad (3.182)$$

mit $x_j = \frac{2\pi}{N} j$ kann im Spezialfall, in dem N eine Potenz von 2 ist, ein sehr effizienter Algorithmus entwickelt werden, falls man zu einer komplexen Fourier-Transformation übergeht. Unter Ausnutzung der Periodizität läuft in (3.182) der Summationsindex j von 0 bis $N - 1$. Aus je zwei aufeinander folgenden Stützwerten bilden wir die $n = N/2$ komplexen Zahlenwerte

$$\boxed{y_j := f(x_{2j}) + i f(x_{2j+1}), \quad j = 0, 1, \dots, n - 1, \quad n = \frac{N}{2}.} \quad (3.183)$$

Zu diesen komplexen Daten y_j definieren wir die *diskrete, komplexe Fourier-Transformation* der Ordnung n

$$\boxed{\begin{aligned} c_k &:= \sum_{j=0}^{n-1} y_j e^{-ijk\frac{2\pi}{n}} = \sum_{j=0}^{n-1} y_j w_n^{jk}, \quad k = 0, 1, \dots, n - 1 \\ \text{mit } w_n &:= e^{-i\frac{2\pi}{n}} = \cos\left(\frac{2\pi}{n}\right) - i \sin\left(\frac{2\pi}{n}\right). \end{aligned}} \quad (3.184)$$

Die komplexe Größe w_n stellt eine n -te Einheitswurzel dar. Der Zusammenhang zwischen den komplexen Fourier-Transformierten c_k und den gesuchten reellen Größen a'_k und b'_k folgt aus dem

Satz 3.32. *Die reellwertigen trigonometrischen Summen a'_k und b'_k sind gegeben durch die komplexen Fourier-Transformierten c_k gemäß*

$$a'_k - ib'_k = \frac{1}{2}(c_k + \bar{c}_{n-k}) + \frac{1}{2i}(c_k - \bar{c}_{n-k})e^{-ik\pi/n} \quad (3.185)$$

$$a'_{n-k} - ib'_{n-k} = \frac{1}{2}(\bar{c}_k + c_{n-k}) + \frac{1}{2i}(\bar{c}_k - c_{n-k})e^{ik\pi/n} \quad (3.186)$$

für $k = 0, 1, \dots, n$, falls $b'_0 = b'_n = 0$ und $c_n = c_0$ gesetzt wird.

Beweis. Für den ersten Summanden von (3.185) erhalten wir

$$\frac{1}{2}(c_k + \bar{c}_{n-k}) = \frac{1}{2} \sum_{j=0}^{n-1} \{y_j w_n^{jk} + \bar{y}_j \overline{w_n^{j(n-k)}}\} = \frac{1}{2} \sum_{j=0}^{n-1} (y_j + \bar{y}_j) w_n^{jk},$$

und für den Klammerausdruck des zweiten Summanden

$$\frac{1}{2i}(c_k - \bar{c}_{n-k}) = \frac{1}{2i} \sum_{j=0}^{n-1} \{y_j w_n^{jk} - \bar{y}_j \overline{w_n^{j(n-k)}}\} = \frac{1}{2i} \sum_{j=0}^{n-1} (y_j - \bar{y}_j) w_n^{jk}.$$

Verwenden wir die Definition (3.183), so ergibt sich

$$\begin{aligned} & \frac{1}{2}(c_k + \bar{c}_{n-k}) + \frac{1}{2i}(c_k - \bar{c}_{n-k}) e^{-ik\pi/n} \\ &= \sum_{j=0}^{n-1} \left\{ f(x_{2j}) e^{-ijk2\pi/n} + f(x_{2j+1}) e^{-ik(2j+1)\pi/n} \right\} \\ &= \sum_{j=0}^{n-1} \left\{ f(x_{2j}) [\cos(kx_{2j}) - i \sin(kx_{2j})] \right. \\ &\quad \left. + f(x_{2j+1}) [\cos(kx_{2j+1}) - i \sin(kx_{2j+1})] \right\} = a'_k - ib'_k. \end{aligned}$$

Damit ist (3.185) verifiziert. (3.186) ergibt sich aus (3.185), indem k durch $n-k$ substituiert wird. \square

Zur Berechnung der reellen Werte a'_k und b'_k können für einen festen Index k durch Addition und Subtraktion der Gleichungen (3.185) und (3.186) einfache Beziehungen für die Summen und Differenzen der Wertepaare a'_k und a'_{n-k} , bzw. b'_k und b'_{n-k} gewonnen werden.

Wir wollen jetzt das Grundprinzip des effizienten Algorithmus zur Durchführung der diskreten Fourier-Transformation (3.184) schrittweise darstellen. Da w_n eine n -te Einheitswurzel darstellt, liegen die Potenzen w_n^{jk} auf dem Einheitskreis der komplexen Ebene und bilden die Eckpunkte eines regelmäßigen n -Ecks. Die Exponenten lassen sich deshalb mod n reduzieren. Nach dieser Vorbemerkung betrachten wir den Fall $n = 4$. Die Fourier-Transformation (3.184) stellen wir mit einer Matrix $\mathbf{W}_4 \in \mathbb{C}^{4,4}$ als lineare Transformation dar.

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & w^1 & w^2 & w^3 \\ 1 & w^2 & 1 & w^2 \\ 1 & w^3 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}, \quad w = w_4; \quad \mathbf{c} = \mathbf{W}_4 \mathbf{y} \quad (3.187)$$

Vertauschen wir in (3.187) die zweite und dritte Komponente im Vektor \mathbf{c} und damit auch die zweite und dritte Zeile in \mathbf{W}_4 , so lässt sich diese zeilenpermutierte Matrix offensichtlich

als Produkt von zwei Matrizen schreiben. Aus (3.187) erhalten wir

$$\begin{pmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \\ \tilde{c}_3 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_2 \\ c_1 \\ c_3 \end{pmatrix} = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & w^2 & 1 & w^2 \\ \hline 1 & w & w^2 & w^3 \\ 1 & w^3 & w^2 & w^1 \end{array} \right) \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} =$$

$$\left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & w^2 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & w^2 \end{array} \right) \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & w^2 & 0 \\ 0 & w^1 & 0 & w^3 \end{array} \right) \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}. \quad (3.188)$$

Die erste Faktormatrix der Produktzerlegung, unterteilt in vier Untermatrizen der Ordnung zwei, hat Block-Diagonalgestalt, wobei die beiden Untermatrizen in der Diagonale identisch sind. Die zweite Faktormatrix baut sich aus vier Diagonalmatrizen je der Ordnung zwei auf, wobei zweimal die Einheitsmatrix vorkommt. Auf Grund der Darstellung (3.188) führen wir die lineare Transformation auch in zwei Teilschritten aus. Wir multiplizieren den Vektor \mathbf{y} mit der zweiten Faktormatrix und erhalten als Ergebnis den Vektor \mathbf{z} mit den Komponenten

$$z_0 = y_0 + y_2, \quad z_1 = y_1 + y_3, \quad (3.189)$$

$$z_2 = (y_0 - y_2)w^0, \quad z_3 = (y_1 - y_3)w^1, \quad (3.190)$$

wenn wir berücksichtigen, dass $w^2 = -1$ und $w^3 = -w^1$ gelten. Im Hinblick auf die Verallgemeinerung ist in (3.190) die triviale Multiplikation mit $w^0 = 1$ aufgeführt. Im zweiten Teilschritt wird der Hilfsvektor \mathbf{z} mit der ersten Faktormatrix multipliziert. Wir erhalten

$$\tilde{c}_0 = c_0 = z_0 + z_1, \quad \tilde{c}_1 = c_2 = z_0 + w^2 z_1, \quad (3.191)$$

$$\tilde{c}_2 = c_1 = z_2 + z_3, \quad \tilde{c}_3 = c_3 = z_2 + w^2 z_3. \quad (3.192)$$

Die Formelsätze (3.191) und (3.192) sind identisch gebaut. Im Hinblick auf die Verallgemeinerung wird die triviale Multiplikation mit $w^2 = -1$ mitgeführt. Wenn wir $w_4^2 = w_2^1$ berücksichtigen, erkennt man, dass (3.191) und (3.192) je eine komplexe Fourier-Transformation der Ordnung zwei darstellen. Die Fourier-Transformation (3.187) der Ordnung vier wurde somit mittels (3.189) und (3.190) auf zwei Fourier-Transformationen der Ordnung zwei zurückgeführt.

Die Reduktion einer komplexen Fourier-Transformation von gerader Ordnung auf zwei Fourier-Transformationen je der halben Ordnung ist stets möglich. Das kann mittels einer analogen Faktorisierung der zeilenpermutierten Transformationsmatrix gezeigt werden. Statt dessen zeigen wir dies mit Hilfe von algebraischen Umformungen. Es sei $n = 2m$, $m \in \mathbb{N}^*$. Dann gilt für die komplexen Transformierten c_k (3.184) mit geraden Indizes $k = 2l$, $l = 0, 1, \dots, m-1$,

$$c_{2l} = \sum_{j=0}^{2m-1} y_j w_n^{2lj} = \sum_{j=0}^{m-1} (y_j + y_{m+j}) w_n^{2lj} = \sum_{j=0}^{m-1} (y_j + y_{m+j})(w_n^2)^{lj}.$$

Dabei wurde die Identität $w_n^{2l(m+j)} = w_n^{2lj} w_n^{2lm} = w_n^{2lj}$ verwendet. Mit den m Hilfswerten

$$z_j := y_j + y_{m+j}, \quad j = 0, 1, \dots, m-1,$$

(3.193)

und wegen $w_n^2 = w_m$ sind die m Koeffizienten

$$c_{2l} = \sum_{j=0}^{m-1} z_j w_m^{jl}, \quad l = 0, 1, \dots, m-1, \quad (3.194)$$

gemäß (3.184) die Fourier-Transformierten der Ordnung m der Hilfswerte z_j (3.193). Für die c_k mit ungeraden Indizes $k = 2l+1$, $l = 0, 1, \dots, m-1$, gilt

$$\begin{aligned} c_{2l+1} &= \sum_{j=0}^{2m-1} y_j w_n^{(2l+1)j} = \sum_{j=0}^{m-1} \{y_j w_n^{(2l+1)j} + y_{m+j} w_n^{(2l+1)(m+j)}\} \\ &= \sum_{j=0}^{m-1} \{y_j - y_{m+j}\} w_n^{(2l+1)j} = \sum_{j=0}^{m-1} \{(y_j - y_{m+j}) w_n^j\} w_n^{2lj}. \end{aligned}$$

Mit den weiteren m Hilfswerten

$$z_{m+j} := (y_j - y_{m+j}) w_n^j, \quad j = 0, 1, \dots, m-1, \quad (3.195)$$

sind die m Koeffizienten

$$c_{2l+1} = \sum_{j=0}^{m-1} z_{m+j} w_m^{jl}, \quad l = 0, 1, \dots, m-1, \quad (3.196)$$

die Fourier-Transformierten der Ordnung m der Hilfswerte z_{m+j} (3.195). Die Zurückführung einer komplexen Fourier-Transformation der Ordnung $n = 2m$ auf zwei komplexe Fourier-Transformationen der Ordnung m erfordert wegen (3.195) als wesentlichen Rechenaufwand m komplexe Multiplikationen. Ist die Ordnung $n = 2^\gamma$, $\gamma \in \mathbb{N}^*$, so können die beiden Fourier-Transformationen der Ordnung m selbst wieder auf je zwei Fourier-Transformationen der halben Ordnung zurückgeführt werden, so dass eine systematische Reduktion möglich ist. Im Fall $n = 32 = 2^5$ besitzt dieses Vorgehen die formale Beschreibung

$$FT_{32} \xrightarrow{16} 2(FT_{16}) \xrightarrow{2:8} 4(FT_8) \xrightarrow{4:4} 8(FT_4) \xrightarrow{8:2} 16(FT_2) \xrightarrow{16:1} 32(FT_1), \quad (3.197)$$

in welcher FT_k eine Fourier-Transformation der Ordnung k bedeutet, und die Zahl der erforderlichen komplexen Multiplikationen für die betreffenden Reduktionsschritte angegeben ist. Da die Fourier-Transformierten der Ordnung Eins mit den zu transformierenden Werten übereinstimmen, stellen die nach dem letzten Reduktionsschritt erhaltenen Zahlenwerte die gesuchten Fourier-Transformierten c_k dar.

Eine komplexe Fourier-Transformation (3.184) der Ordnung $n = 2^\gamma$, $\gamma \in \mathbb{N}^*$ ist in Verallgemeinerung von (3.197) mit γ Reduktionsschritten auf n Fourier-Transformationen der Ordnung Eins zurückführbar. Da jeder Schritt $n/2$ komplexe Multiplikationen erfordert, beträgt der totale Rechenaufwand

$$Z_{FTn} = \frac{1}{2} n \gamma = \frac{1}{2} n \log_2 n \quad (3.198)$$

komplexe Multiplikationen. Der Rechenaufwand nimmt somit nur etwa linear mit der Ordnung n zu. Deshalb wird die skizzierte Methode als *schnelle Fourier-Transformation* (fast Fourier transform=FFT) bezeichnet. Sie wird *Cooley* und *Tukey* [Coo 65] zugeschrieben. Sie wurde aber bereits von *Good* formuliert [Goo 60]. Die hohe Effizienz der schnellen Fourier-Transformation ist in der Zusammenstellung (3.199) illustriert, wo für verschiedene Ordnungen n die Anzahl n^2 der komplexen Multiplikationen, die bei der direkten Berechnung der Summen (3.184) erforderlich ist, dem Aufwand Z_{FTn} (3.198) gegenübergestellt ist.

$\gamma =$	5	6	8	9	10	11	12	
$n =$	32	64	256	512	1024	2048	4096	
$n^2 =$	1024	4096	65536	$2.62 \cdot 10^5$	$1.05 \cdot 10^6$	$4.19 \cdot 10^6$	$1.68 \cdot 10^7$	
$Z_{FTn} =$	80	192	1024	2304	5120	11264	24576	
Faktor	12.8	21.3	64	114	205	372	683	(3.199)

Der Rechenaufwand der schnellen Fourier-Transformation wird gegenüber n^2 um den Faktor in der letzten Zeile von (3.199) verringert. Eine mögliche algorithmische Realisierung der schnellen Fourier-Transformation, welche auf den vorhergehenden Betrachtungen beruht, besteht aus einer sukzessiven Transformation der gegebenen y -Werte in die gesuchten c -Werte. Die erforderlichen Rechenschritte sind in Tab. 3.3 für den Fall $n = 16 = 2^4$ dargestellt. Für das ganze Schema gilt $w := e^{-i2\pi/16} = e^{-i\pi/8} = \cos(\pi/8) - i \sin(\pi/8)$. Die zu berechnenden Hilfsgrößen z_j (3.193) und z_{m+j} (3.195) werden wieder mit y_j und y_{m+j} bezeichnet, wobei die Wertzuweisungen so zu verstehen sind, dass die rechts stehenden Größen stets die Werte vor dem betreffenden Schritt bedeuten.

Im ersten Schritt sind gemäß (3.193) und (3.195) gleichzeitig Summen und Differenzen von y -Werten zu bilden, wobei die letzteren mit Potenzen von w zu multiplizieren sind. Aus der ersten Gruppe von acht y -Werten resultieren die Fourier-Transformierten c_k mit geraden Indizes und aus der zweiten Gruppe die restlichen c_k . Die betreffenden c -Werte sind zur Verdeutlichung des Prozesses angegeben. Im zweiten Schritt sind für jede der beiden Gruppen von y -Werten entsprechende Summen und Differenzen zu bilden, wobei wegen $w_8 = w_{16}^2 = w^2$ nur gerade Potenzen von w als Multiplikatoren auftreten. Zudem sind auch die Fourier-Transformierten c_k den entsprechenden Permutationen innerhalb jeder Gruppe zu unterwerfen. Im dritten Schritt sind die vier Gruppen analog mit der Einheitswurzel $w_4 = w_{16}^4 = w^4$ zu behandeln, wobei die Fourier-Transformierten c_k entsprechend zu (3.188) zu permutieren sind. Im vierten und letzten Schritt erfolgen noch die letzten Reduktionen mit $w_2 = w_{16}^8 = w^8$.

Die resultierenden y -Werte sind identisch mit den c -Werten, wobei allerdings die Zuordnung der Indexwerte eine zusätzliche Betrachtung erfordert. Den Schlüssel für die richtige Zuordnung liefert die binäre Darstellung der Indexwerte. In Tab. 3.4 sind die Binärdarstellungen der Indizes der c -Werte nach den einzelnen Reduktionsschritten zusammengestellt.

Die Binärdarstellungen der letzten Spalte sind genau umgekehrt zu denjenigen der ersten Spalte. Ein Beweis dieser Feststellung beruht darauf, dass die Permutation der Indexwerte im ersten Schritt einer zyklischen Vertauschung der γ Binärstellen entspricht. Im zweiten Schritt ist die Indexpermutation in den beiden Gruppen äquivalent zu einer zyklischen Vertauschung der ersten ($\gamma - 1$) Binärstellen und so fort. Die eindeutige Wertzuordnung

Tab. 3.3 Eine algorithmische Realisierung der schnellen Fourier-Transformation

y_0	$y_0 := y_0 + y_8$	c_0	$y_0 := y_0 + y_4$	c_0	$y_0 := y_0 + y_2$	c_0	$y_0 := y_0 + y_1 = c_0$
y_1	$y_1 := y_1 + y_9$	c_2	$y_1 := y_1 + y_5$	c_4	$y_1 := y_1 + y_3$	c_8	$y_1 := (y_0 - y_1)w^0 = c_8$
y_2	$y_2 := y_2 + y_{10}$	c_4	$y_2 := y_2 + y_6$	c_8	$y_2 := (y_0 - y_2)w^0$	c_4	$y_2 := y_2 + y_3 = c_4$
y_3	$y_3 := y_3 + y_{11}$	c_6	$y_3 := y_3 + y_7$	c_{12}	$y_3 := (y_1 - y_3)w^4$	c_{12}	$y_3 := (y_2 - y_3)w^0 = c_{12}$
y_4	$y_4 := y_4 + y_{12}$	c_8	$y_4 := (y_0 - y_4)w^0$	c_2	$y_4 := y_4 + y_6$	c_2	$y_4 := y_4 + y_5 = c_2$
y_5	$y_5 := y_5 + y_{13}$	c_{10}	$y_5 := (y_1 - y_5)w^2$	c_6	$y_5 := y_5 + y_7$	c_{10}	$y_5 := (y_4 - y_5)w^0 = c_{10}$
y_6	$y_6 := y_6 + y_{14}$	c_{12}	$y_6 := (y_2 - y_6)w^4$	c_{10}	$y_6 := (y_4 - y_6)w^0$	c_6	$y_6 := y_6 + y_7 = c_6$
y_7	$y_7 := y_7 + y_{15}$	c_{14}	$y_7 := (y_3 - y_7)w^6$	c_{14}	$y_7 := (y_5 - y_7)w^4$	c_{14}	$y_7 := (y_6 - y_7)w^0 = c_{14}$
y_8	$y_8 := (y_0 - y_8)w^0$	c_1	$y_8 := y_8 + y_{12}$	c_1	$y_8 := y_8 + y_{10}$	c_1	$y_8 := y_8 + y_9 = c_1$
y_9	$y_9 := (y_1 - y_9)w^1$	c_3	$y_9 := y_9 + y_{13}$	c_5	$y_9 := y_9 + y_{11}$	c_9	$y_9 := (y_8 - y_9)w^0 = c_9$
y_{10}	$y_{10} := (y_2 - y_{10})w^2$	c_5	$y_{10} := y_{10} + y_{14}$	c_9	$y_{10} := (y_8 - y_{10})w^0$	c_5	$y_{10} := y_{10} + y_{11} = c_5$
y_{11}	$y_{11} := (y_3 - y_{11})w^3$	c_7	$y_{11} := y_{11} + y_{15}$	c_{13}	$y_{11} := (y_9 - y_{11})w^4$	c_{13}	$y_{11} := (y_{10} - y_{11})w^0 = c_{13}$
y_{12}	$y_{12} := (y_4 - y_{12})w^4$	c_9	$y_{12} := (y_8 - y_{12})w^0$	c_3	$y_{12} := y_{12} + y_{14}$	c_3	$y_{12} := y_{12} + y_{13} = c_3$
y_{13}	$y_{13} := (y_5 - y_{13})w^5$	c_{11}	$y_{13} := (y_9 - y_{13})w^2$	c_7	$y_{13} := y_{13} + y_{15}$	c_{11}	$y_{13} := (y_{12} - y_{13})w^0 = c_{11}$
y_{14}	$y_{14} := (y_6 - y_{14})w^6$	c_{13}	$y_{14} := (y_{10} - y_{14})w^4$	c_{11}	$y_{14} := (y_{12} - y_{14})w^0$	c_7	$y_{14} := y_{14} + y_{15} = c_7$
y_{15}	$y_{15} := (y_7 - y_{15})w^7$	c_{15}	$y_{15} := (y_{11} - y_{15})w^6$	c_{15}	$y_{15} := (y_{13} - y_{15})w^4$	c_{15}	$y_{15} := (y_{14} - y_{15})w^0 = c_{15}$

 $\text{FT}_{16} \rightarrow 2(\text{FT}_8)$ $2(\text{FT}_8) \rightarrow 4(\text{FT}_4)$ $4(\text{FT}_4) \rightarrow 8(\text{FT}_2)$ $8(\text{FT}_2) \rightarrow 16(\text{FT}_1)$

der resultierenden y_i zu den gesuchten Fourier-Transformierten c_k erfolgt somit auf Grund einer *Bitumkehr* der Binärdarstellung von j . Man sieht übrigens noch leicht ein, dass es genügt, die Werte y_i und y_k zu vertauschen, falls auf Grund der Bitumkehr $k > j$ gilt, um die richtige Reihenfolge der c_k zu erhalten.

Tab. 3.4 Folge der Binärdarstellungen der Indexwerte.

j	y_i	$c_k^{(1)}$	$c_k^{(2)}$	$c_k^{(3)}$	$c_k^{(4)}$	k
0	0000	0000	0000	0000	0000	0
1	000L	00L0	0L00	L000	L000	8
2	00L0	0L00	L000	0L00	0L00	4
3	00LL	0LL0	LL00	LL00	LL00	12
4	0L00	L000	00L0	00L0	00L0	2
5	0L0L	L0L0	0LL0	L0L0	L0L0	10
6	0LL0	LL00	L0L0	OL00	OL00	6
7	OLLL	LLL0	LLL0	LLL0	LLL0	14
8	L000	000L	000L	000L	000L	1
9	L00L	00LL	0L0L	L00L	L00L	9
10	L0L0	0L0L	L00L	OLOL	OLOL	5
11	LOLL	OLLL	LL0L	LL0L	LL0L	13
12	LL00	L00L	00LL	00LL	00LL	3
13	LL0L	LOLL	0LLL	L0LL	L0LL	11
14	LLL0	LL0L	L0LL	OLLL	OLLL	7
15	LLLL	LLLL	LLLL	LLLL	LLLL	15

Beispiel 3.25. Die schnelle Fourier-Transformation wird angewandt zur Berechnung der approximativen Fourier-Koeffizienten a_k^* und b_k^* der (2π) -periodischen Funktionen $f(x) = x^2, 0 < x < 2\pi$ von Beispiel 3.21. Bei $N = 16$ Stützstellen ist eine FFT der Ordnung $n = 8$ auszuführen. Die aus den reellen Stützwerten $f(x_l)$ gebildeten komplexen Werte werden mit $y_j^{(0)}$ bezeichnet. In Tab. 3.5 sind die gerundeten Zahlenwerte im Verlauf der FFT zusammengestellt sowie die aus den Fourier-Transformierten c_k berechneten reellen Fourier-Koeffizienten a_k^* und b_k^* . \triangle

Eine effiziente Realisierung der komplexen Fourier-Transformation (3.184) ist auch für eine allgemeinere Ordnung n möglich [Boo 80, Bri 95, Win 78]. Softwarebibliotheken erlauben nahezu alle Zahlen für N . Der Aufwand hängt dann von der Primfaktorzerlegung von N ab. Die entsprechende NAG-Routine, [NAGa, NAGb], ist besonders schnell für Vielfache von 2, 3 und 5, erlaubt aber bis zu 20 Faktoren aus Primzahlen nicht größer als 19.

3.8 Orthogonale Polynome

Wir betrachten im Folgenden zwei Systeme von orthogonalen Polynomen, die für bestimmte Anwendungen von Bedeutung sind. Wir stellen die Eigenschaften nur soweit zusammen, wie sie für die späteren Zwecke benötigt werden. Die sehr speziellen Eigenschaften der

Tab. 3.5 Schnelle Fourier-Transformation.

j	$y_j^{(0)}$	$y_j^{(1)}$	$y_j^{(2)}$	
0	$19.73921 + 0.15421i$	$29.60881 + 12.64543i$	$54.28282 + 42.56267i$	
1	$0.61685 + 1.38791i$	$16.03811 + 20.04763i$	$51.81542 + 62.30188i$	
2	$2.46740 + 3.85531i$	$24.67401 + 29.91724i$	$4.93480 - 17.27181i$	
3	$5.55165 + 7.55642i$	$35.77732 + 42.25424i$	$-22.20661 + 19.73921i$	
4	$9.86960 + 12.49122i$	$9.86960 - 12.33701i$	$-12.33701 + 7.40220i$	
5	$15.42126 + 18.65972i$	$-22.68131 - 1.74472i$	$-24.42602 + 34.89432i$	
6	$22.20661 + 26.06192i$	$-22.20661 + 19.73921i$	$32.07621 - 32.07621i$	
7	$30.22566 + 34.69783i$	$-1.74472 + 36.63904i$	$-38.38375 + 20.93659i$	
j, k	$y_j^{(3)}$	c_k	a_k^*	
0	$106.09825 + 104.86455i$	$106.09825 + 104.86455i$	26.370349	—
1	$2.46740 - 19.73921i$	$-36.76303 + 42.29652i$	4.051803	-12.404463
2	$-17.27181 + 2.46740i$	$-17.27181 + 2.46740i$	1.053029	-5.956833
3	$27.14141 - 37.01102i$	$-6.30754 - 11.13962i$	0.499622	-3.692727
4	$-36.76303 + 42.29652i$	$2.46740 - 19.73921i$	0.308425	-2.467401
5	$12.08902 - 27.49212i$	$12.08902 - 27.49212i$	0.223063	-1.648665
6	$-6.30754 - 11.13962i$	$27.14141 - 37.01102i$	0.180671	-1.022031
7	$70.45997 - 53.01281i$	$70.45997 - 53.01281i$	0.160314	-0.490797
8	—	—	0.154213	—

Tschebyscheff-Polynome machen diese geeignet zur Approximation von Funktionen, wobei sich eine Beziehung zu den Fourier-Reihen ergeben wird. Die *Legendre-Polynome* werden hauptsächlich die Grundlage bilden für die Herleitung von bestimmten Quadraturformeln in Kapitel 7.

3.8.1 Approximation mit Tschebyscheff-Polynomen

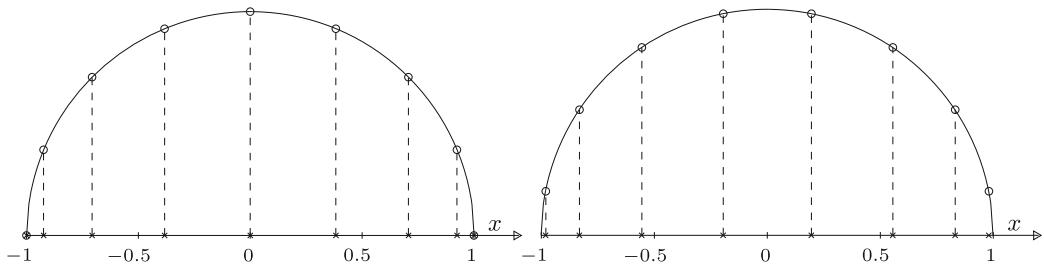
Satz 3.33. Für $n \in \mathbb{N}$ ist $\cos(n\varphi)$ als Polynom n -ten Grades in $\cos(\varphi)$ darstellbar.

Beweis. Für $n = 0$ und $n = 1$ ist die Aussage trivial. Aus dem Additionstheorem für die Kosinusfunktion folgt

$$\begin{aligned} \cos[(n+1)\varphi] + \cos[(n-1)\varphi] &= 2 \cos(\varphi) \cos(n\varphi), \quad n \in \mathbb{N}^*, \\ \implies \cos[(n+1)\varphi] &= 2 \cos(\varphi) \cos(n\varphi) - \cos[(n-1)\varphi]. \end{aligned} \tag{3.200}$$

Damit bekommt man sukzessive

$$\begin{aligned} \cos(2\varphi) &= 2 \cos^2(\varphi) - 1, \\ \cos(3\varphi) &= 2 \cos(\varphi) \cos(2\varphi) - \cos(\varphi) = 4 \cos^3(\varphi) - 3 \cos(\varphi), \\ \cos(4\varphi) &= 2 \cos(\varphi) \cos(3\varphi) - \cos(2\varphi) = 8 \cos^4(\varphi) - 8 \cos^2(\varphi) + 1. \end{aligned} \tag{3.201}$$

Abb. 3.31 Extremal- und Nullstellen von $T_8(x)$.

Ein Beweis der Behauptung ist jetzt durch vollständige Induktion nach n offensichtlich. \square

Das n -te Tschebyscheff-Polynom $T_n(x)$ wird auf der Grundlage von Satz 3.33 definiert durch

$$\cos(n\varphi) =: T_n(\cos(\varphi)) = T_n(x), \quad x = \cos(\varphi), \quad x \in [-1, 1], \quad n \in \mathbb{N}. \quad (3.202)$$

Der wegen $x = \cos(\varphi)$ zunächst nur für das Intervall $[-1, 1]$ gültige Definitionsbereich der Polynome $T_n(x)$ kann natürlich auf ganz \mathbb{R} erweitert werden. Die lineare Transformation $t = a + (b - a)(x + 1)/2$ führt auf ein beliebiges Intervall $[a, b]$. Wir werden aber die T-Polynome nur im Intervall $[-1, 1]$ betrachten. Die ersten T-Polynome lauten gemäß (3.201)

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1,$$

$$T_3(x) = 4x^3 - 3x, \quad T_4(x) = 8x^4 - 8x^2 + 1.$$

Aus der Definition (3.202) folgt sofort die Eigenschaft

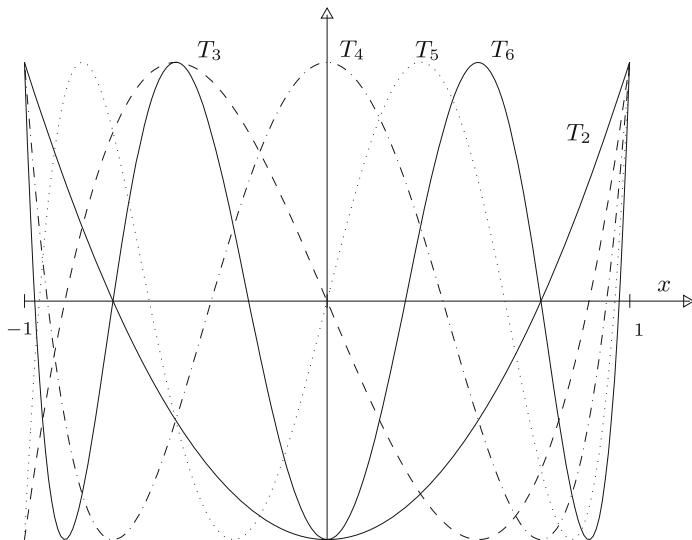
$$|T_n(x)| \leq 1 \quad \text{für} \quad x \in [-1, 1], \quad n \in \mathbb{N}. \quad (3.203)$$

Das n -te Polynom $T_n(x)$ nimmt die Extremalwerte ± 1 dann an, falls $n\varphi = k\pi, k = 0, 1, \dots, n$, gilt. Deshalb sind die $(n+1)$ Extremalstellen von $T_n(x)$ gegeben durch

$$x_k^{(e)} = \cos\left(\frac{k\pi}{n}\right), \quad k = 0, 1, 2, \dots, n, \quad n \geq 1. \quad (3.204)$$

Sie sind nicht äquidistant im Intervall $[-1, 1]$, sondern liegen gegen die Enden des Intervalls dichter. Geometrisch können sie als Projektionen von regelmäßig auf dem Halbkreis mit Radius Eins verteilten Punkten interpretiert werden. In Abb. 3.31 links ist die Konstruktion für $n = 8$ dargestellt. Die Extremalstellen $x_k^{(e)}$ liegen im Intervall $[-1, 1]$ symmetrisch bezüglich des Nullpunktes. Wegen des oszillierenden Verhaltens von $\cos(n\varphi)$ werden die Extremalwerte mit alternierendem Vorzeichen angenommen, falls x und damit auch φ das Intervall monoton durchläuft.

Zwischen zwei aufeinander folgenden Extremalstellen von $T_n(x)$ liegt notwendigerweise je eine Nullstelle. Aus der Bedingung $\cos(n\varphi) = 0$ ergibt sich $n\varphi = (2k-1)\frac{\pi}{2}, k = 1, 2, \dots, n$.

Abb. 3.32 Tschebyscheff-Polynome $T_2(x)$ bis $T_6(x)$.

Folglich sind die n Nullstellen von $T_n(x)$

$$x_k = \cos\left(\frac{2k-1}{n}\frac{\pi}{2}\right), \quad k = 1, 2, \dots, n, \quad n \geq 1. \quad (3.205)$$

Die Nullstellen von $T_n(x)$ sind reell und einfach und liegen im Innern des Intervalls $[-1, 1]$ symmetrisch zum Nullpunkt. Sie sind gegen die Enden des Intervalls dichter verteilt. Ihre geometrische Konstruktion ist in Abb. 3.31 rechts für $n = 8$ dargestellt. Wegen ihrer Bedeutung im Zusammenhang mit der Approximation von Funktionen bezeichnet man sie als die *Tschebyscheff-Abszissen* zum n -ten T-Polynom. Wegen (3.200) und (3.202) erfüllen die T-Polynome die Drei-Term-Rekursion

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1; \quad T_0(x) = 1, \quad T_1(x) = x. \quad (3.206)$$

Mit vollständiger Induktion nach n zeigt man mit Hilfe von (3.206), dass der Koeffizient von x^n des Polynoms $T_n(x)$ für $n \geq 1$ gleich 2^{n-1} ist. Analog folgt aus (3.206) die Eigenschaft

$$T_n(-x) = (-1)^n T_n(x), \quad n \geq 0. \quad (3.207)$$

Somit ist $T_n(x)$ entsprechend der Parität von n ein gerades oder ungerades Polynom. In Abb. 3.32 sind die Tschebyscheff-Polynome $T_2(x)$ bis $T_6(x)$ dargestellt.

Satz 3.34. Die Polynome $T_n(x)$, ($n = 0, 1, 2, \dots$) bilden für das Intervall $[-1, 1]$ mit dem L_2 -Skalarprodukt und der Gewichtsfunktion $w(x) = 1/\sqrt{1-x^2}$ ein System von orthogonalen

Polynomen. Es gelten die Beziehungen

$$\boxed{\int_{-1}^1 T_k(x)T_j(x) \frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0 & \text{falls } k \neq j \\ \frac{1}{2}\pi & \text{falls } k = j > 0 \\ \pi & \text{falls } k = j = 0 \end{cases}, \quad k, j \in \mathbb{N}} \quad (3.208)$$

Beweis. Mit den Substitutionen $x = \cos(\varphi)$, $T_k(x) = \cos(k\varphi)$, $T_j(x) = \cos(j\varphi)$ und $dx = -\sin(\varphi)d\varphi$ ergibt sich

$$\begin{aligned} \int_{-1}^1 T_k(x)T_j(x) \frac{dx}{\sqrt{1-x^2}} &= - \int_{\pi}^0 \cos(k\varphi) \cos(j\varphi) \frac{\sin(\varphi)d\varphi}{\sin(\varphi)} \\ &= \int_0^{\pi} \cos(k\varphi) \cos(j\varphi) d\varphi = \frac{1}{2} \int_{-\pi}^{\pi} \cos(k\varphi) \cos(j\varphi) d\varphi. \end{aligned}$$

Wenden wir auf das letzte Integral die Relationen (3.156) an, so erhalten wir die Aussage (3.208). \square

Jetzt betrachten wir die Aufgabe, eine im Intervall $[-1, 1]$ gegebene, stetige Funktion $f(x)$ im quadratischen Mittel durch ein Polynom $g_n(x)$ n -ten Grades in der Darstellung mit T-Polynomen

$$\boxed{g_n(x) = \frac{1}{2}c_0T_0(x) + \sum_{k=1}^n c_kT_k(x)} \quad (3.209)$$

zu approximieren. Dies stellt eine kontinuierliche Gauß-Approximation dar mit dem Skalarprodukt

$$(f, g) := \int_{-1}^1 f(x)g(x)w(x) dx := \int_{-1}^1 f(x)g(x) \frac{dx}{\sqrt{1-x^2}}. \quad (3.210)$$

Die gesuchten Koeffizienten c_k sind deshalb wegen der Orthogonalitätsbeziehungen (3.208) und des Ansatzes (3.209) gegeben durch

$$\boxed{c_k = \frac{2}{\pi} \int_{-1}^1 f(x)T_k(x) \frac{dx}{\sqrt{1-x^2}}, \quad k = 0, 1, 2, \dots, n.} \quad (3.211)$$

Die Integraldarstellung (3.211) vereinfacht sich stark, falls wir die Variablensubstitution $x = \cos(\varphi)$ durchführen und (3.202) berücksichtigen. Nach elementaren Umformungen ergibt

sich aus (3.211)

$$c_k = \frac{2}{\pi} \int_0^\pi f(\cos \varphi) \cos(k\varphi) d\varphi, \quad k = 0, 1, \dots, n. \quad (3.212)$$

Nun ist aber $F(\varphi) := f(\cos \varphi)$ offensichtlich eine gerade und (2π) -periodische Funktion des Arguments φ . Deshalb gilt auch

$$c_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(\cos \varphi) \cos(k\varphi) d\varphi, \quad k = 0, 1, 2, \dots, n. \quad (3.213)$$

Folgerung. Die Koeffizienten c_k in der Entwicklung (3.209) der approximierenden Funktion $g_n(x)$ sind die Fourier-Koeffizienten a_k (3.160) der geraden, (2π) -periodische Funktion $F(\varphi) := f(\cos \varphi)$.

Damit ist die betrachtete Aufgabe auf die Berechnung von Fourier-Koeffizienten zurückgeführt worden, und die im Abschnitt 3.7 entwickelten Methoden sind anwendbar. Die Koeffizienten c_k und das zugehörige Polynom $g_n(x)$ können für beliebig großes n definiert werden, und zu jeder im Intervall $[-1, 1]$ stetigen Funktion $f(x)$ kann durch Grenzübergang die unendliche *Tschebyscheff-Entwicklung* (kurz: T-Entwicklung)

$$g(x) = \frac{1}{2} c_0 T_0(x) + \sum_{k=1}^{\infty} c_k T_k(x) \quad (3.214)$$

gebildet werden. Die Frage der Konvergenz der Reihe (3.214) lässt sich mit Hilfe von Satz 3.26 beantworten. Dazu stellen wir fest, dass die (2π) -periodische Funktion $F(\varphi)$ als Zusammensetzung von zwei stetigen Funktionen stetig ist. Besitzt die gegebene Funktion $f(x)$ im Intervall $[-1, 1]$ eine stückweise stetige erste Ableitung, dann trifft dies auch für $F(\varphi)$ zu. Unter dieser Zusatzvoraussetzung konvergiert die zu $f(x)$ gehörige T-Entwicklung (3.214) für alle $x \in [-1, 1]$ gegen den Wert $f(x)$.

Der Approximationsfehler $f(x) - g_n(x)$ kann unter einer zusätzlichen Voraussetzung abgeschätzt werden.

Satz 3.35. Es seien c_k die Koeffizienten der T-Entwicklung einer stetigen und einmal stückweise stetig differenzierbaren Funktion $f(x)$. Falls die Reihe $\sum_{k=1}^{\infty} |c_k|$ konvergiert, dann konvergiert die T-Entwicklung (3.214) für alle $x \in [-1, 1]$ gleichmäßig gegen $f(x)$ und es gilt

$$|f(x) - g_n(x)| \leq \sum_{k=n+1}^{\infty} |c_k| \quad \text{für } x \in [-1, 1]. \quad (3.215)$$

Beweis. Nach Voraussetzung konvergiert die unendliche T-Reihe (3.214) punktweise gegen $f(x)$ für $|x| \leq 1$. Für den Beweis der gleichmäßigen Konvergenz benutzen wir die Tatsache,

dass zu jedem $\varepsilon > 0$ ein N existiert mit

$$\sum_{k=n}^{\infty} |c_k| < \varepsilon \quad \text{für alle } n > N. \quad (3.216)$$

Dann gilt für jedes $n > N$ und $|x| \leq 1$ wegen $|T_k(x)| \leq 1$

$$|f(x) - g_n(x)| = \left| \sum_{k=n+1}^{\infty} c_k T_k(x) \right| \leq \sum_{k=n+1}^{\infty} |c_k| < \varepsilon. \quad (3.217)$$

Damit ist auch die Fehlerabschätzung (3.215) gezeigt. \square

Beispiel 3.26. Die Funktion $f(x) = e^x$ soll im Intervall $[-1, 1]$ durch ein Polynom $g_n(x)$ (3.209) approximiert werden. Die Entwicklungskoeffizienten sind nach (3.212) gegeben durch

$$c_k = \frac{2}{\pi} \int_0^\pi e^{\cos \varphi} \cos(k\varphi) d\varphi = 2I_k(1), \quad k = 0, 1, 2, \dots, \quad (3.218)$$

wo $I_k(x)$ die modifizierte k -te Bessel-Funktion darstellt [Abr 71]. Auf Grund der Potenzreihe für $I_k(x)$ erhalten wir

$$\begin{aligned} c_0 &\doteq 2.5321317555, & c_1 &\doteq 1.1303182080, & c_2 &\doteq 0.2714953395, \\ c_3 &\doteq 0.0443368498, & c_4 &\doteq 0.0054742404, & c_5 &\doteq 0.0005429263, \\ c_6 &\doteq 0.0000449773, & c_7 &\doteq 0.0000031984, & c_8 &\doteq 0.0000001992, \\ c_9 &\doteq 0.0000000110, & c_{10} &\doteq 0.0000000006. \end{aligned} \quad (3.219)$$

Die Koeffizienten c_k bilden eine rasch konvergente Nullfolge, und die zugehörige Reihe ist gleichmäßig konvergent. Die zu approximierende Funktion $f(x)$ ist beliebig oft stetig differenzierbar, so dass die Voraussetzungen von Satz 3.35 erfüllt sind. Für $g_6(x)$ erhalten wir nach (3.215) als Schranke für den Approximationsfehler

$$|f(x) - g_6(x)| \leq 3.409 \cdot 10^{-6} \quad \text{für alle } |x| \leq 1.$$

Die Approximation ist so gut, dass nur bis $n = 2$ graphisch ein Unterschied zwischen Funktion und Approximation zu sehen ist, siehe Abb. 3.33. Der Fehlerverlauf ist im Intervall $[-1, 1]$ sehr gleichmäßig. \triangle

Die Entwicklungskoeffizienten c_k können nur in seltenen Fällen explizit dargestellt werden, und so sind sie näherungsweise zu berechnen. Da die c_k gleich den Fourier-Koeffizienten a_k der (2π) -periodischen Funktion $F(\varphi) = f(\cos \varphi)$ sind, stehen im Prinzip die Formeln (3.167) zur Verfügung. In jenen Formeln ist x durch φ zu ersetzen, und so erhalten wir

$$\begin{aligned} c_k^* &= \frac{2}{N} \sum_{j=1}^N F(\varphi_j) \cos(k\varphi_j) = \frac{2}{N} \sum_{j=0}^{N-1} f(\cos \varphi_j) \cos(k\varphi_j), \\ \varphi_j &= \frac{2\pi}{N} j, \quad N \in \mathbb{N}^*, \quad k = 0, 1, 2, \dots, \left[\frac{N}{2} \right]. \end{aligned} \quad (3.220)$$

Die Näherungswerte c_k^* lassen sich mit der schnellen Fourier-Transformation effizient berechnen. Zur direkten Berechnung der Koeffizienten c_k^* für nicht zu großes $N = 2m$, $m \in \mathbb{N}^*$,

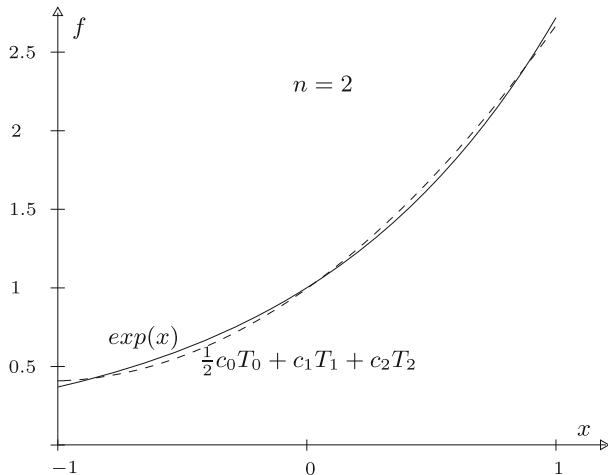


Abb. 3.33 Tschebyscheff-Approximation der Exponentialfunktion, $n = 2$.

kann die Summation vereinfacht werden, indem die Symmetrie der cos-Funktion ausgenutzt wird. Es ergibt sich die Darstellung

$$\begin{aligned} c_k^* &= \frac{2}{m} \left\{ \frac{1}{2}(f(1) + f(-1) \cos(k\pi)) + \sum_{j=1}^{m-1} f\left(\cos\left(\frac{j\pi}{m}\right)\right) \cos\left(\frac{kj\pi}{m}\right) \right\}, \\ k &= 0, 1, 2, \dots, n, \quad n \leq m. \end{aligned} \quad (3.221)$$

Die Rechenvorschrift (3.221) ist interpretierbar als summierte Trapezregel zur genäherten Berechnung des Integrals in (3.212) bei Unterteilung des Integrationsintervalls $[0, \pi]$ in m Teilintervalle.

Falls die im Intervall $[-1, 1]$ stetige Funktion $f(x)$ auch eine stückweise stetige erste Ableitung besitzt, so besteht zwischen den Näherungen c_k^* und den exakten Koeffizienten c_k nach (3.177) die Beziehung

$$\begin{aligned} c_k^* &= c_k + c_{N-k} + c_{N+k} + c_{2N-k} + c_{2N+k} + \dots, \\ k &= 0, 1, \dots, n \leq m, \quad N = 2m. \end{aligned} \quad (3.222)$$

Unter der Voraussetzung, dass das asymptotisch gültige Gesetz über das Verhalten der Koeffizienten c_k einer gegebenen Funktion bekannt ist, kann der Fehler $|c_k^* - c_k|$ bei gegebenem N abgeschätzt werden oder der erforderliche Wert von N bestimmt werden, um den Fehler hinreichend klein zu machen.

Beispiel 3.27. Die Koeffizienten c_k der T-Entwicklung der Funktion $f(x) = e^x$ konvergieren rasch gegen null. Da $c_{11} \doteq 2.50 \cdot 10^{-11}$ und $c_{12} \doteq 1.04 \cdot 10^{-12}$ sind und für die weiteren Koeffizienten $|c_k| < 10^{-13}$, $k \geq 13$ gilt, folgt aus (3.222), dass bereits für $m = 12$, d.h. $N = 24$ die Formel (3.221) die ersten elf Entwicklungskoeffizienten mit einer Genauigkeit von zehn Dezimalstellen nach dem Komma liefert. Es gilt sogar

$$|c_k^* - c_k| \leq |c_{24-k}| + |c_{24+k}| + \dots < 10^{-13} \quad \text{für } k = 0, 1, \dots, 10. \quad \triangle$$

Wir betrachten noch die Aufgabe, den Wert eines approximierenden Polynoms $g_n(x)$ in der Darstellung (3.209) effizient und numerisch stabil zu berechnen. Die nahe liegende Idee, zu gegebenem x die Werte der T-Polynome mit Hilfe der Rekursionsformel (3.206) sukzessive zu ermitteln und damit die Partialsummen zu bilden, führt zu einer instabilen Methode. Es zeigt sich, dass es besser ist, die endliche T-Reihe (3.209) unter Verwendung der Rekursionsformel (3.206) rückwärts abzubauen, d.h. mit dem letzten Term zu beginnen. Wir stellen den Algorithmus im Fall $n = 5$ dar.

$$\begin{aligned} g_5(x) &= \frac{1}{2}c_0T_0 + c_1T_1 + c_2T_2 + c_3T_3 + c_4T_4 + c_5T_5 \\ &= \frac{1}{2}c_0T_0 + c_1T_1 + c_2T_2 + (c_3 - c_5)T_3 + (c_4 + 2xc_5)T_4 \end{aligned}$$

Mit der Substitution $d_4 := c_4 + 2xc_5$ ergibt der nachfolgende Schritt

$$g_5(x) = \frac{1}{2}c_0T_0 + c_1T_1 + (c_2 - d_4)T_2 + (c_3 + 2xd_4 - c_5)T_3.$$

Jetzt setzen wir $d_3 := c_3 + 2xd_4 - c_5$ und erhalten in analoger Fortsetzung mit den entsprechenden Definitionen weiter

$$\begin{aligned} g_5(x) &= \frac{1}{2}c_0T_0 + (c_1 - d_3)T_1 + (c_2 + 2xd_3 - d_4)T_2 \\ &= \left(\frac{1}{2}c_0 - d_2\right)T_0 + (c_1 + 2xd_2 - d_3)T_1 \\ &= \left(\frac{1}{2}c_0 - d_2\right)T_0 + d_1T_1. \end{aligned}$$

Wegen $T_0(x) = 1$ und $T_1(x) = x$ erhalten wir schließlich

$$g_5(x) = \frac{1}{2}c_0 + xd_1 - d_2 = \frac{1}{2}\{(c_0 + 2xd_1 - d_2) - d_2\} = \frac{1}{2}(d_0 - d_2).$$

Aus den gegebenen T-Koeffizienten c_0, c_1, \dots, c_n sind die Werte $d_{n-1}, d_{n-2}, \dots, d_0$ rekursiv zu berechnen. Der Wert des Polynoms $g_n(x)$ ergibt sich dann als halbe Differenz von d_0 und d_2 . Dieser *Algorithmus von Clenshaw* [Cle 55] zur Berechnung des Wertes $g_n(x)$ zu gegebenem x lautet zusammengefasst:

$$\begin{aligned} d_n &= c_n; \quad y = 2 \times x; \quad d_{n-1} = c_{n-1} + y \times c_n \\ \text{für } k &= n-2, n-3, \dots, 0 : \\ d_k &= c_k + y \times d_{k+1} - d_{k+2} \\ g_n(x) &= (d_0 - d_2)/2 \end{aligned} \tag{3.223}$$

Der Rechenaufwand zur Auswertung von $g_n(x)$ für einen Argumentwert x beträgt nur $(n+2)$ Multiplikationen. Der Algorithmus (3.223) ist stabil, da gezeigt werden kann, dass der totale Fehler in $g_n(x)$ höchstens gleich der Summe der Beträge der Rundungsfehler ist, die bei der Berechnung der d_k auftreten [Fox 79].

Beispiel 3.28. Wir wollen noch ein Beispiel durchrechnen, das zeigt, dass die Konvergenz der T-Entwicklung und die Genauigkeit der Approximation stark von der Glattheit der Funktion f abhängen. Wir wollen die stetige, aber nicht differenzierbare Dachkurve

$$f(x) = 1 - |x|, \quad -1 \leq x \leq 1$$

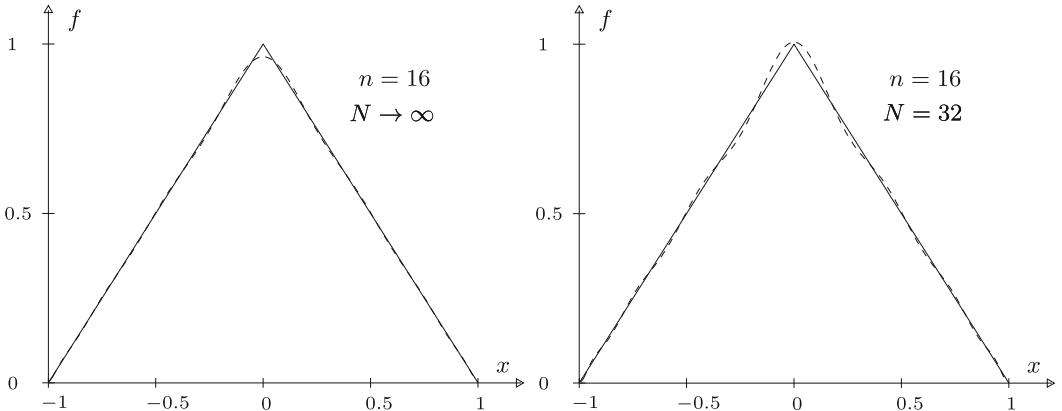


Abb. 3.34 Tschebyscheff-Approximation der Dachfunktion mit c_k (links) und c_k^* (rechts).

mit Tschebyscheff-Polynomen approximieren. Die Koeffizienten c_k gehen nur langsam mit N gegen null. Wir können sie nicht exakt berechnen, sondern müssen mit den diskret angenähernten Koeffizienten c_k^* arbeiten. Die Funktion f ist gerade, deshalb sind alle $c_{2k+1} = 0$. Für $n = 16$ ($N = 32$) sind die Koeffizienten auf fünf wesentliche Stellen

$$\begin{aligned} c_0^* &\doteq 0.73085, & c_2^* &\doteq -0.42854, & c_4^* &\doteq 0.089106, \\ c_6^* &\doteq -0.040773, & c_8^* &\doteq 0.024864, & c_{10}^* &\doteq -0.017885, \\ c_{12}^* &\doteq 0.014448, & c_{14}^* &\doteq -0.012803, & c_{16}^* &\doteq 0.012311. \end{aligned}$$

Wegen (3.222) ist klar, dass bei langsamer Konvergenz der T-Entwicklung auch der Fehler $|c_k - c_k^*|$ nur langsam mit N gegen null geht. Das lässt sich feststellen durch Rechnen mit sehr großem N . Für $N \rightarrow \infty$ ergeben sich die ersten neun Koeffizienten ungleich null als

$$\begin{aligned} c_0 &\doteq 0.72676, & c_2 &\doteq -0.42441, & c_4 &\doteq 0.084883, \\ c_6 &\doteq -0.036378, & c_8 &\doteq 0.020210, & c_{10} &\doteq -0.012861, \\ c_{12} &\doteq 0.0089038, & c_{14} &\doteq -0.0065294, & c_{16} &\doteq 0.0049931. \end{aligned}$$

Entsprechend unterschiedlich sind die Approximationen, siehe Abb. 3.34. \triangle

3.8.2 Interpolation mit Tschebyscheff-Polynomen

Zur Funktionsapproximation mit Hilfe von T-Polynomen ist ein bestimmtes Interpolationspolynom im Vergleich zur endlichen T-Reihe oft ebenso zweckmäßig. Um das Vorgehen zu motivieren, benötigen wir folgenden

Satz 3.36. Unter allen Polynomen $P_n(x)$ von Grad $n \geq 1$, deren Koeffizient von x^n gleich Eins ist, hat $T_n(x)/2^{n-1}$ die kleinste Maximumnnorm im Intervall $[-1, 1]$, d.h. es gilt

$$\min_{P_n(x)} \left\{ \max_{x \in [-1, 1]} |P_n(x)| \right\} = \max_{x \in [-1, 1]} \left| \frac{1}{2^{n-1}} T_n(x) \right| = \frac{1}{2^{n-1}}. \quad (3.224)$$

Beweis. Die Minimax-Eigenschaft (3.224) des n -ten T-Polynoms $T_n(x)$ zeigen wir indirekt. Deshalb nehmen wir an, es existiere ein Polynom $P_n(x)$ mit Höchstkoeffizient Eins, so dass $|P_n(x)| < 1/2^{n-1}$ für alle $x \in [-1, 1]$ gilt. Unter dieser Annahme gelten für die $(n+1)$ Extremstellen $x_k^{(e)}$ (3.204) von $T_n(x)$ die folgenden Ungleichungen

$$\begin{aligned} P_n(x_0^{(e)}) &< T_n(x_0^{(e)})/2^{n-1} = 1/2^{n-1}, \\ P_n(x_1^{(e)}) &> T_n(x_1^{(e)})/2^{n-1} = -1/2^{n-1}, \\ P_n(x_2^{(e)}) &< T_n(x_2^{(e)})/2^{n-1} = 1/2^{n-1}, \quad \text{usw.} \end{aligned}$$

Folglich nimmt das Differenzpolynom

$$Q(x) := P_n(x) - T_n(x)/2^{n-1}$$

an den in abnehmender Reihenfolge angeordneten $(n+1)$ Extremstellen $x_0^{(e)} > x_1^{(e)} > \dots > x_n^{(e)}$ Werte mit alternierenden Vorzeichen an. Aus Stetigkeitsgründen besitzt $Q(x)$ (mindestens) n verschiedene Nullstellen. Da aber sowohl $P_n(x)$ als auch $T_n(x)/2^{n-1}$ den Höchstkoeffizienten Eins haben, ist der Grad von $Q(x)$ höchstens gleich $(n-1)$. Wegen des Hauptsatzes der Algebra ist dies ein Widerspruch. \square

Die Minimax-Eigenschaft der T-Polynome besitzt eine wichtige Anwendung im Zusammenhang mit der Polynominterpolation. Wird eine mindestens $(n+1)$ -mal stetig differenzierbare Funktion $f(x)$ im Intervall $[-1, 1]$ durch ein Polynom $P_n(x)$ interpoliert, so gilt nach (3.6) mit $M_{n+1} := \max_{-1 \leq \xi \leq 1} |f^{(n+1)}(\xi)|$ für den Interpolationsfehler die Ungleichung

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |(x-x_0)(x-x_1)\dots(x-x_n)|, \quad x \in [-1, 1].$$

Die $(n+1)$ Stützstellen x_0, x_1, \dots, x_n sollen jetzt so gewählt werden, dass

$$\max_{-1 \leq x \leq 1} |(x-x_0)(x-x_1)\dots(x-x_n)| = \text{Min!}$$

gilt. Die Funktion $\omega(x) := (x-x_0)(x-x_1)\dots(x-x_n)$ stellt ein Polynom vom Grad $(n+1)$ mit dem Höchstkoeffizienten Eins dar. Nach Satz 3.36 ist sein Betrags-Maximum für alle $x \in [-1, 1]$ genau dann minimal, falls die $(n+1)$ Stützstellen gleich den $(n+1)$ Nullstellen von $T_{n+1}(x)$ sind. Es gilt dann $\max |\omega(x)| = 2^{-n}$. Für das Interpolationspolynom $P_n^*(x)$, dessen Stützstellen gleich den *Tschebyscheff-Abszissen* zum $(n+1)$ -ten T-Polynom sind, erhält man somit die kleinstmögliche Schranke für den Interpolationsfehler

$$|f(x) - P_n^*(x)| \leq \frac{M_{n+1}}{2^n \cdot (n+1)!}, \quad x \in [-1, 1]. \tag{3.225}$$

Das Ergebnis (3.225) bedeutet aber nicht, dass das Interpolationspolynom $P_n^*(x)$ das Polynom bester Approximation im Tschebyscheffschen Sinn darstellt, denn in der jetzt gültigen Darstellung des Interpolationsfehlers (3.5) ist ξ von x abhängig:

$$f(x) - P_n^*(x) = \frac{f^{(n+1)}(\xi)}{2^n \cdot (n+1)!} T_{n+1}(x), \quad x \in [-1, 1] \tag{3.226}$$

Im Abschnitt 3.1 sind wir bereits ausführlich auf die Polynominterpolation eingegangen. Für das Polynom $P_n^*(x)$ zu den Tschebyscheff-Abszissen ist jetzt die folgende Darstellung als Linearkombination von T-Polynomen besonders vorteilhaft:

$$P_n^*(x) = \frac{1}{2}\gamma_0 T_0(x) + \sum_{k=1}^n \gamma_k T_k(x). \quad (3.227)$$

Zur Bestimmung der Koeffizienten $\gamma_0, \gamma_1, \dots, \gamma_n$ aus den $(n+1)$ Interpolationsbedingungen

$$\frac{1}{2}\gamma_0 T_0(x_l) + \sum_{k=1}^n \gamma_k T_k(x_l) = f(x_l), \quad l = 1, 2, \dots, n+1, \quad (3.228)$$

an den Tschebyscheff-Abszissen $x_l = \cos\left(\frac{2l-1}{n+1}\frac{\pi}{2}\right)$ von $T_{n+1}(x)$ benötigen wir eine diskrete Orthogonalitätseigenschaft der T-Polynome.

Satz 3.37. Es seien x_l die $(n+1)$ Nullstellen von $T_{n+1}(x)$. Dann gelten

$$\sum_{l=1}^{n+1} T_k(x_l) T_j(x_l) = \begin{cases} 0 & \text{falls } k \neq j \\ \frac{1}{2}(n+1) & \text{falls } k = j > 0 \\ n+1 & \text{falls } k = j = 0 \end{cases} \quad 0 \leq k, j \leq n \quad (3.229)$$

Beweis. Wegen (3.202) und (3.205) sind die Werte der T-Polynome an den Tschebyscheff-Abszissen mit $h := \pi/(n+1)$

$$T_k(x_l) = \cos(k \cdot \arccos(x_l)) = \cos\left(k \frac{2l-1}{n+1} \frac{\pi}{2}\right) = \cos\left(k \left(l - \frac{1}{2}\right) h\right).$$

Aus bekannten trigonometrischen Identitäten folgt damit

$$\begin{aligned} \sum_{l=1}^{n+1} T_k(x_l) T_j(x_l) &= \sum_{l=1}^{n+1} \cos\left(kh\left(l - \frac{1}{2}\right)\right) \cos\left(jh\left(l - \frac{1}{2}\right)\right) \\ &= \frac{1}{2} \sum_{l=1}^{n+1} \left\{ \cos\left((k-j)h\left(l - \frac{1}{2}\right)\right) + \cos\left((k+j)h\left(l - \frac{1}{2}\right)\right) \right\} \\ &= \frac{1}{2} \operatorname{Re} \left\{ \sum_{l=1}^{n+1} e^{i(k-j)h(l-\frac{1}{2})} + \sum_{l=1}^{n+1} e^{i(k+j)h(l-\frac{1}{2})} \right\}. \end{aligned} \quad (3.230)$$

Die beiden Summen stellen je endliche geometrische Reihen mit den Quotienten $q_1 = e^{i(k-j)h}$ beziehungsweise $q_2 = e^{i(k+j)h}$ dar. Wir betrachten zuerst den Fall $k \neq j$. Da $0 \leq k \leq n$ und $0 \leq j \leq n$ gelten, folgen die Ungleichungen $0 < |k-j| \leq n$ und $0 < k+j < 2n$. Deshalb sind wegen

$$\frac{\pi}{n+1} \leq |(k-j)h| \leq \frac{n\pi}{n+1}, \quad \frac{\pi}{n+1} \leq (k+j)h < \frac{2\pi n}{n+1}$$

$q_1 \neq 1$ und $q_2 \neq 1$. Für die erste Summe ergibt sich somit

$$\sum_{l=1}^{n+1} e^{i(k-j)h(l-\frac{1}{2})} = e^{\frac{1}{2}i(k-j)h} \cdot \frac{e^{i(k-j)h(n+1)} - 1}{e^{i(k-j)h} - 1} = \frac{(-1)^{k-j} - 1}{2i \sin\left(\frac{1}{2}(k-j)\frac{\pi}{n+1}\right)}$$

ein rein imaginärer oder verschwindender Wert. Dasselbe gilt für die zweite Summe, so dass der Realteil in (3.230) auf jeden Fall gleich null ist. Damit ist die erste Zeile von (3.229) gezeigt.

Für $k = j > 0$ ist die erste Summe in (3.230) gleich $(n + 1)$, während die zweite den Wert null hat. Im Fall $k = j = 0$ sind alle Summanden der beiden Summen in (3.230) gleich Eins. Damit sind die beiden letzten Aussagen von (3.229) gezeigt. \square

Die Relationen (3.229) bedeuten für die Matrix des linearen Gleichungssystems (3.228), dass ihre Spaltenvektoren paarweise orthogonal sind. Die Unbekannten $\gamma_0, \gamma_1, \dots, \gamma_n$ lassen sich aus diesem Grund explizit als Lösung von (3.228) angeben. Dazu multiplizieren wir die l -te Gleichung von (3.228) mit $T_j(x_l)$, wo j ein fester Index mit $0 \leq j \leq n$ sein soll. Dann addieren wir alle $(n + 1)$ Gleichungen und erhalten unter Berücksichtigung von (3.229) nach einer Indexsubstitution für die Koeffizienten γ_k die Darstellung

$$\begin{aligned} \gamma_k &= \frac{2}{n+1} \sum_{l=1}^{n+1} f(x_l) T_k(x_l) \\ &= \frac{2}{n+1} \sum_{l=1}^{n+1} f\left(\cos\left(\frac{2l-1}{n+1}\frac{\pi}{2}\right)\right) \cos\left(k\frac{2l-1}{n+1}\frac{\pi}{2}\right), \\ &\quad k = 0, 1, \dots, n. \end{aligned} \tag{3.231}$$

Die Entwicklungskoeffizienten γ_k des interpolierenden Polynoms $P_n^*(x)$ bezüglich der Tschebyscheff-Abszissen von $T_{n+1}(x)$ unterscheiden sich von den angenäherten Koeffizienten c_k^* (3.221). Auch mit diesen Koeffizienten kann ein Polynom n -ten Grades

$$g_n^*(x) := \frac{1}{2}c_0^*T_0(x) + \sum_{k=1}^{n-1} c_k^*T_k(x) + \frac{1}{2}c_n^*T_n(x) \tag{3.232}$$

gebildet werden, welches im Fall $n = m = \frac{N}{2}$ wegen Satz 3.30 die interpolierende Eigenschaft an den $(n + 1)$ Extremalstellen $x_j^{(e)} = \cos\left(\frac{j\pi}{n}\right)$ von $T_n(x)$ hat. In diesem Fall bilden die Intervallendpunkte ± 1 Stützstellen des Interpolationspolynoms $g_n^*(x)$. Für $g_n^*(x)$ gilt die Abschätzung (3.225) des Interpolationsfehlers nicht.

Beispiel 3.29. Für die Funktion $f(x) = e^x$ erhalten wir für die Koeffizienten γ_k des an den Tschebyscheff-Abszissen interpolierenden Polynoms $P_6^*(x)$ nach (3.231) die Werte

$$\begin{aligned} \gamma_0 &\doteq 2.5321317555, & \gamma_1 &\doteq 1.1303182080, & \gamma_2 &\doteq 0.2714953395, \\ \gamma_3 &\doteq 0.0443368498, & \gamma_4 &\doteq 0.0054742399, & \gamma_5 &\doteq 0.0005429153, \\ \gamma_6 &\doteq 0.0000447781. \end{aligned}$$

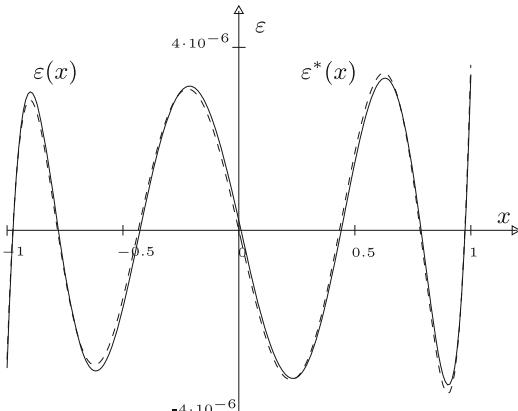


Abb. 3.35
Approximationfehler der T-Polynome für e^x .

Im Vergleich zu den Entwicklungskoeffizienten c_k (3.219) unterscheiden sich nur die letzten drei Koeffizienten γ_4, γ_5 und γ_6 von c_4, c_5 und c_6 innerhalb von zehn Dezimalstellen nach dem Komma. Der maximale Interpolationsfehler von $P_6^*(x)$ beträgt nach (3.225) wegen $M_7 = \max_{-1 \leq x \leq 1} |f^{(7)}(x)| = \max_{-1 \leq x \leq 1} |e^x| \doteq 2.7183$ höchstens

$$|e^x - P_6^*(x)| \leq 8.43 \cdot 10^{-6} \quad \text{für alle } |x| \leq 1.$$

Der Interpolationsfehler wird naturgemäß etwas überschätzt. Der maximale Betrag der Abweichung beträgt tatsächlich $3.620 \cdot 10^{-6}$. Das Interpolationspolynom $P_6^*(x)$ liefert eine vergleichbar gute Approximation wie $g_6(x)$. In Abb. 3.35 sind die beiden Fehlerfunktionen $\varepsilon^*(x) := e^x - P_6^*(x)$ und $\varepsilon(x) := e^x - g_6(x)$ dargestellt. \triangle

3.8.3 Die Legendre-Polynome

In diesem Abschnitt bedeutet $P_n(x)$ das n -te *Legendre-Polynom*, welches definiert ist durch

$$\boxed{P_n(x) := \frac{1}{2^n \cdot n!} \frac{d^n}{dx^n} [(x^2 - 1)^n], \quad n \in \mathbb{N}.} \quad (3.233)$$

Da der Ausdruck in der eckigen Klammer ein Polynom vom echten Grad $2n$ ist, stellt seine n -te Ableitung ein Polynom vom Grad n dar.

Satz 3.38. *Die Legendre-Polynome $P_n(x)$, $n = 0, 1, 2, \dots$ bilden für das Intervall $[-1, 1]$ mit dem L_2 -Skalarprodukt und der Gewichtsfunktion $w(x) = 1$ ein Orthogonalsystem. Es gilt*

$$\boxed{\int_{-1}^1 P_m(x) P_n(x) dx = \begin{cases} 0 & \text{falls } m \neq n \\ \frac{2}{2n+1} & \text{falls } m = n \end{cases} \quad m, n \in \mathbb{N}.} \quad (3.234)$$

Beweis. Wir zeigen zuerst die Orthogonalität der Legendre-Polynome und setzen ohne Einschränkung $m < n$ voraus. Dann gilt für das Integral nach partieller Integration

$$\begin{aligned} I_{m,n} &:= 2^m m! 2^n n! \int_{-1}^1 P_m(x) P_n(x) dx = \int_{-1}^1 \frac{d^m}{dx^m} [(x^2 - 1)^m] \cdot \frac{d^n}{dx^n} [(x^2 - 1)^n] dx \\ &= \left. \frac{d^m}{dx^m} [(x^2 - 1)^m] \cdot \frac{d^{n-1}}{dx^{n-1}} [(x^2 - 1)^n] \right|_{-1}^1 \\ &\quad - \int_{-1}^1 \frac{d^{m+1}}{dx^{m+1}} [(x^2 - 1)^m] \cdot \frac{d^{n-1}}{dx^{n-1}} [(x^2 - 1)^n] dx. \end{aligned} \tag{3.235}$$

Nun ist zu beachten, dass das Polynom $(x^2 - 1)^n$ für $x = \pm 1$ je eine n -fache Nullstelle besitzt. Demzufolge gilt

$$\frac{d^{n-k}}{dx^{n-k}} [(x^2 - 1)^n] = 0 \text{ für } x = \pm 1 \text{ und für } k = 1, 2, \dots, n. \tag{3.236}$$

Nach weiteren $(n - 1)$ analogen partiellen Integrationen erhält man, da die ausintegrierten Teile jeweils gleich null sind

$$I_{m,n} = (-1)^n \int_{-1}^1 \frac{d^{m+n}}{dx^{m+n}} [(x^2 - 1)^m] \cdot (x^2 - 1)^n dx. \tag{3.237}$$

Nach unserer Annahme ist $m + n > 2m$, und somit verschwindet der erste Faktor des Integranden, und es gilt $I_{m,n} = 0$. Der zweite Teil der Behauptung (3.234) folgt aus der Darstellung (3.237) des Integrals, welche auch für $m = n$ gültig ist. Mit

$$\frac{d^{2n}}{dx^{2n}} [(x^2 - 1)^n] = (2n)!$$

ergibt sich aus (3.237) durch n -malige partielle Integration

$$\begin{aligned} I_{n,n} &= (-1)^n (2n)! \int_{-1}^1 (x - 1)^n (x + 1)^n dx \\ &= (-1)^n (2n)! \left[(x - 1)^n \frac{1}{n+1} (x + 1)^{n+1} \right]_{-1}^1 \\ &\quad - \frac{n}{n+1} \int_{-1}^1 (x - 1)^{n-1} (x + 1)^{n+1} dx = \dots \end{aligned}$$

$$\begin{aligned}
&= (-1)^{2n} (2n!) \frac{n(n-1)(n-2)\dots 1}{(n+1)(n+2)(n+3)\dots(2n)} \int_{-1}^1 (x+1)^{2n} dx \\
&= (n!)^2 \cdot \frac{2^{2n+1}}{2n+1}.
\end{aligned}$$

Wegen (3.235) folgt daraus die zweite Aussage von (3.234). \square

Aus der Definition (3.233) der Legendre-Polynome ist klar, dass $P_n(x)$ eine gerade oder ungerade Funktion in x ist entsprechend der Parität von n . Denn der Ausdruck in der eckigen Klammer von (3.233) ist eine gerade Funktion in x . Da die Ableitung einer geraden Funktion ungerade ist und umgekehrt, hat die n -te Ableitung die genannte Eigenschaft und es gilt

$$P_n(-x) = (-1)^n P_n(x), \quad n \in \mathbb{N}. \quad (3.238)$$

Satz 3.39. Das Legendre-Polynom $P_n(x)$, $n \geq 1$, besitzt im Intervall $(-1, 1)$ n einfache Nullstellen.

Beweis. Ausgehend von der Tatsache, dass $(x^2 - 1)^n$ für $x = \pm 1$ je eine n -fache Nullstelle besitzt, folgt die Aussage durch n -malige Anwendung des Satzes von Rolle. Dabei ist zu beachten, dass $\frac{d^k}{dx^k}[(x^2 - 1)^n]$ für $x = \pm 1$ und $k = 1, 2, \dots, n-1$ je eine $(n-k)$ -fache Nullstelle aufweist. Daraus folgt die Existenz von mindestens n paarweise verschiedenen Nullstellen im Innern von $[-1, 1]$. Da ein Polynom n -ten Grades genau n Nullstellen (unter Berücksichtigung ihrer Vielfachheiten) besitzt, folgt die Behauptung. \square

Die Nullstellen von $P_n(x)$ können für allgemeines n nicht wie im Fall der T-Polynome durch eine geschlossene Formel angegeben werden. Ihre Werte findet man beispielsweise in [Abr 71, Sch 76, Str 74] tabelliert.

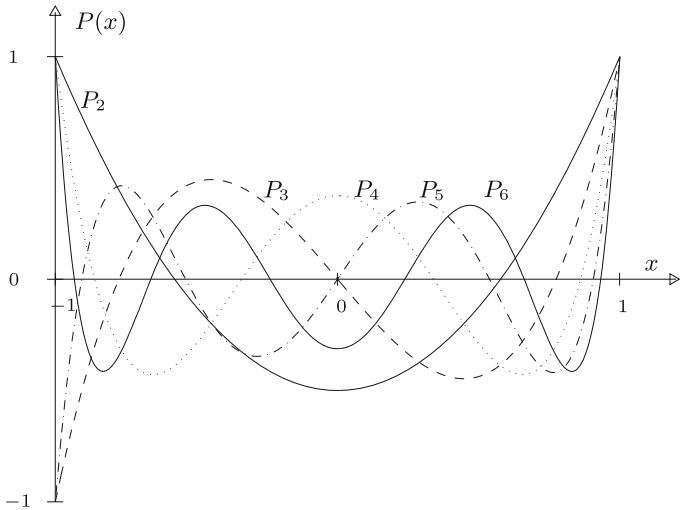
Satz 3.40. Für die Legendre-Polynome gilt die Drei-Term-Rekursion

$$\boxed{
\begin{aligned}
P_0(x) &= 1, \quad P_1(x) = x, \\
P_{n+1}(x) &= \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x), \quad n = 1, 2, \dots
\end{aligned}
} \quad (3.239)$$

Beweis. Es kann allgemeiner gezeigt werden, dass es zu jedem gewichteten L_2 -Skalarprodukt ein System orthogonaler Polynome gibt. Alle diese Systeme genügen einer Drei-Term-Rekursion. Die Systeme sind unter gewissen Zusatzbedingungen eindeutig bestimmt, siehe etwa [Deu 08b]. \square

Auf Grund der Rekursionsformel (3.239) ergeben sich die weiteren Legendre-Polynome

$$P_2(x) = \frac{1}{2}(3x^2 - 1), \quad P_3(x) = \frac{1}{2}(5x^3 - 3x),$$

Abb. 3.36 Legendre-Polynome $P_2(x)$ bis $P_6(x)$.

$$\begin{aligned} P_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3), & P_5(x) &= \frac{1}{8}(63x^5 - 70x^3 + 15x), \\ P_6(x) &= \frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5). \end{aligned}$$

In Abb. 3.36 sind die Legendre-Polynome $P_2(x)$ bis $P_6(x)$ dargestellt. Ohne Beweis sei erwähnt, dass die Legendre-Polynome im Intervall $[-1, 1]$ betragsmäßig durch Eins beschränkt sind. Zur weiteren Charakterisierung der Legendre-Polynome zeigt man mit vollständiger Induktion nach n mit Hilfe von (3.239)

$$P_n(1) = 1, \quad P_n(-1) = (-1)^n, \quad n = 0, 1, 2, \dots$$

(3.240)

Die kontinuierliche Gauß-Approximation mit Legendre-Polynomen

$$g_n(x) = \sum_{k=0}^n c_k P_k(x) \quad (3.241)$$

ergibt analog zu der entsprechenden Aufgabe mit T-Polynomen wegen (3.234) die Koeffizienten

$$c_k = \frac{2k+1}{2} \int_{-1}^1 f(x) P_k(x) dx, \quad k = 0, 1, \dots, n.$$

(3.242)

Die exakte, analytische Berechnung der Integrale (3.242) ist meistens nicht möglich, und die erhaltenen Formeln unterliegen bei ihrer numerischen Auswertung oft einer starken Auslösung. Zur genäherten Berechnung der Integrale ist die *Gaußquadratur* (vgl. Abschnitt 7.4) gut geeignet, die auf den Legendre-Polynomen beruht.

Beispiel 3.30. Es soll die Funktion $f(x) = e^x$ im Intervall $[-1, 1]$ durch ein Polynom sechsten Grades $g_6(x)$ im quadratischen Mittel approximiert werden. Die Entwicklungskoeffizienten c_k des Polynoms $g_6(x)$ nach den Legendre-Polynomen sind gegeben durch (3.242)

$$c_k = \frac{2k+1}{2} \int_{-1}^1 e^x P_k(x) dx, \quad k = 0, 1, 2, \dots, 6. \quad (3.243)$$

Um diese Integrale zu berechnen, bestimmen wir zur Vorbereitung die Hilfsintegrale

$$I_n := \int_{-1}^1 x^n e^x dx, \quad n = 0, 1, 2, \dots, 6, \quad (3.244)$$

aus denen sich die c_k durch Linearkombinationen ergeben. Durch partielle Integration erhalten wir die Rekursionsformel

$$\begin{aligned} I_n &= x^n e^x \Big|_{-1}^1 - n \int_{-1}^1 x^{n-1} e^x dx = \left(e - (-1)^n \frac{1}{e} \right) - n I_{n-1}, \quad n \geq 1, \\ \text{also } I_n &= \begin{cases} \left(e - \frac{1}{e} \right) - n I_{n-1}, & \text{falls } n \text{ gerade,} \\ \left(e + \frac{1}{e} \right) - n I_{n-1}, & \text{falls } n \text{ ungerade.} \end{cases} \end{aligned} \quad (3.245)$$

Für größere Werte von n ist (3.245) für numerische Zwecke hoffnungslos instabil, da sich ein Anfangsfehler in I_0 mit einem Verstärkungsfaktor $n!$ auf den Wert von I_n auswirkt! Die Rekursionsformel (3.245) wird numerisch stabil, falls man sie in der umgekehrten Form

$$I_{n-1} = \begin{cases} \left\{ \left(e - \frac{1}{e} \right) - I_n \right\} / n, & \text{falls } n \text{ gerade,} \\ \left\{ \left(e + \frac{1}{e} \right) - I_n \right\} / n, & \text{falls } n \text{ ungerade,} \end{cases} \quad (3.246)$$

anwendet und für ein hinreichend großes N mit $I_N = 0$ startet. Um die gewünschten Integrale mit vierzehnstelliger Genauigkeit zu erhalten, genügt es $N = 24$ zu wählen. In Tab. 3.6 sind die Werte der Integrale I_0 bis I_6 zusammengestellt mit den daraus resultierenden Koeffizienten c_0 bis c_6 .

Tab. 3.6 Integrale und Entwicklungskoeffizienten.

k	I_k	c_k
0	2.35040238729	1.1752011936
1	0.73575888234	1.1036383235
2	0.87888462260	0.3578143506
3	0.44950740182	0.0704556337
4	0.55237277999	0.0099651281
5	0.32429736969	0.0010995861
6	0.40461816913	0.0000994543

Der Verlauf der Fehlerfunktion $\varepsilon(x) := e^x - g_6(x)$ ist in Abb. 3.37 dargestellt. Im Vergleich zur entsprechenden Approximation mit T-Polynomen ist der Betrag des Fehlers an den Enden des Intervalls gut zweimal größer, während er im Innern des Intervalls vergleichbar ist. Das unterschiedliche

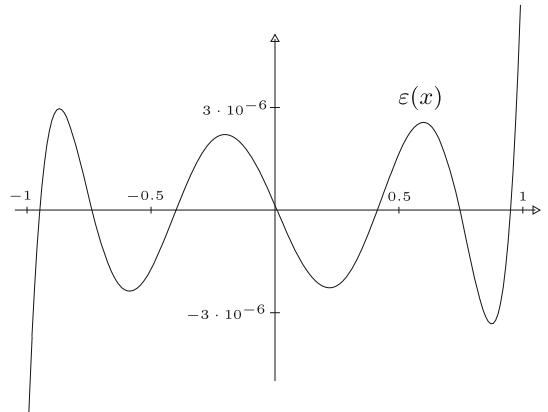


Abb. 3.37 Gauß-Approximation mit Legendre-Polynomen, Approximationsfehler für e^x .

Verhalten der Fehlerfunktion ist auf die Gewichtsfunktionen zurückzuführen. \triangle

3.9 Software

Unterprogramme zu allen geschilderten Methoden finden sich in den großen numerischen Software-Bibliotheken, bei NAG [NAGa, NAGb] in den Kapiteln C06 (Tschebyscheff-Reihen, FFT, auch zwei- und dreidimensional), E01 (ein- und mehrdimensionale Interpolation mit Polynomen und Splines) und E02 (die entsprechenden Approximations-Verfahren).

Ein bekanntes Fortran-Paket zur ein- und mehrdimensionalen Interpolation und Approximation ist FITPACK von Dierckx⁴ [Die 89, Die 06].

Auch in MATLAB sind die Standardverfahren zur ein- und mehrdimensionalen Interpolation und Approximation enthalten. MATLAB hat darüber hinaus eine Werkzeugkiste für spezielle Splinemethoden (Spline Toolbox).

Unsere Problemlöseumgebung PAN (<http://www.upb.de/SchwarzKoeckler/>) verfügt über vier Programme zur ein- und zweidimensionalen Interpolation, zur Kurveninterpolation und zur Approximation mit trigonometrischen Funktionen und mit Tschebyscheff-Polynomen.

3.10 Aufgaben

3.1. Auf Grund des kubischen Interpolationspolynoms $P_3(x)$ für die äquidistanten Stützstellen $x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, x_3 = x_0 + 3h$ leite man die Formeln (3.30) zur angeneherten Berechnung der ersten und zweiten Ableitung an den Stellen x_0, x_1 und $x_M = (x_0 + x_3)/2$ her. Mit den

⁴<http://gams.nist.gov/serve.cgi/PackageModules/DIERCKX>

Differenzierungsformeln berechne man an der Stelle $x_M = 0.4$ die Näherungen für die erste und zweite Ableitung der Funktion $f(x) = e^{-x}$ für die Folge der Schrittweiten $h = 0.1, 0.01, 0.001, \dots, 10^{-8}$. Die berechneten Werte sind mit den exakten Werten der Ableitungen zu vergleichen.

3.2. Für die Funktion $f(x) = \ln(x) - 2(x - 1)/x$ sind zu den nicht äquidistanten Stützstellen $x_0 = 1, x_1 = 2, x_2 = 4, x_3 = 8, x_4 = 10$ mit dem Interpolationspolynom $P_4(x)$ an den beiden Stellen $x = 2.9$ und $x = 5.25$ die interpolierten Werte zu berechnen. Wie groß sind der Interpolationsfehler und die Fehlerschranke?

3.3. Mit einem Computerprogramm bestimme man die Fehlerfunktion $r(x) = f(x) - P_{10}(x)$ für die Funktion $f(x) = e^{-3x}$ im Intervall $[0, 5]$ unter Verwendung der elf äquidistanten Stützstellen $x_k = 0.5k, k = 0, 1, \dots, 10$.

3.4. Mit der Methode der Extrapolation ist der Wert der ersten Ableitung der Funktion $f(x) = e^{-x}$ an der Stelle $x = 0.4$ möglichst genau zu ermitteln auf Grund des ersten Differenzenquotienten und des zentralen Differenzenquotienten unter Verwendung der Schrittweiten $h_0 = 0.2, h_1 = 0.15, h_2 = 0.10, h_3 = 0.05$ und $h_4 = 0.02$. Welches sind die extrapolierten Werte für lineare, quadratische und kubische Interpolation?

3.5. Man berechne die Hermiteschen Interpolationspolynome dritten und fünften Grades, die die Funktion $f(x) = e^{-x}$ und seine erste Ableitung an den Stützstellen $x_0 = 0.3, x_1 = 0.4$ bzw. $x_0 = 0.2, x_1 = 0.3, x_2 = 0.4$ interpolitieren. Man berechne die interpolierten Werte an der Stelle $x = 0.34$ und bestimme den Interpolationsfehler.

3.6. Man versuche mit der Methode der inversen Interpolation eine Näherung der kleinsten positiven Nullstelle der Funktion $f(x) = \cos(x) \cosh(x) + 1$ zu bestimmen. Die gesuchte Nullstelle liegt im Intervall $[1.8, 1.9]$.

3.7. Zur Funktion $f(x) = 1/(1+x^2)$ bestimme man die natürliche Spline-Interpolierende und die B-Spline-Interpolierende zu den sechs äquidistanten Stützstellen $x_k = k, k = 0, 1, \dots, 5$. Man stelle die beiden kubischen Spline-Interpolierenden graphisch dar, bestimme die Abweichungen in den Punkten $x_k = k + 0.5, k = 0, 1, \dots, 4$, und versuche diese Abweichungen zu erklären.

3.8. Zu einer Datentabelle $(x_i, y_i), i = 0, 1, \dots, n$, mit $x_0 < x_1 < \dots < x_n$ sei $s(x) : [x_0, x_n] \rightarrow \mathbb{R}$ die stückweise lineare Funktion, die die Werte y_i interpoliert. $s(x)$ ist also eine Linearkombination von Hutfunktionen.

Man zeige, dass $s(x)$ die einzige Funktion ist, die folgende Bedingungen erfüllt:

- a) $s(x)$ ist stetig im Intervall $[x_0, x_n]$ und stetig differenzierbar in jedem Teilintervall $(x_i, x_{i+1}), i = 0, 1, \dots, n - 1$.
- b) $s(x_i) = y_i, i = 0, 1, \dots, n$.
- c) Unter allen Funktionen $r(x)$, die die beiden ersten Bedingungen erfüllen minimiert $s(x)$ als einzige das Funktional

$$F(r) = \int_{x_0}^{x_n} (r'(t))^2 dt.$$

3.9. Das vollständige Profil eines Körpers ist durch die zwölf Punkte $P_0, P_1, \dots, P_{10}, P_{11} = P_0$ in der Ebene beschrieben, wobei das Profil im Punkt P_0 eine Spitze aufweist. Die Koordinaten der Punkte $P_k(x_k, y_k)$ sind

k	0	1	2	3	4	5	6	7	8	9	10	11
x_k	25	19	13	9	5	2.2	1	3	8	13	18	25
y_k	5	7.5	9.1	9.4	9	7.5	5	2.1	2	3.5	4.5	5

Gesucht ist die Parameterdarstellung der Kurve auf Grund von zwei Spline-Interpolationen. Das resultierende Profil soll graphisch dargestellt werden.

3.10. Man ermittle die fünf Bézier-Punkte zur Approximation eines Viertelkreises vom Radius $r = 1$ durch eine Bézier-Kurve vierten Grades so, dass die Tangenten der Bézier-Kurve in den Endpunkten mit denjenigen des Kreises übereinstimmen, und dass die Punkte $\mathbf{x}(\lambda)$ für $\lambda = 0.25$ und $\lambda = 0.5$ auf dem Kreis liegen. Es ist zu beachten, dass die drei zu bestimmenden Bézier-Punkte symmetrisch verteilt sind. Welche maximale Abweichung der Bézier-Kurve vom Kreis resultiert?

3.11. Ein Halbkreis vom Radius $r = 1$ soll durch eine Bézier-Kurve vierten Grades so approximiert werden, dass die Tangenten der Bézier-Kurve in den Endpunkten mit denjenigen des Kreises übereinstimmen. Bei der Festlegung der Bézier-Punkte achte man auf die Symmetrie der Kurve und versuche zusätzlich, diese so festzulegen, dass die Punkte $\mathbf{P}(t)$ für $t = 0.25$ und $t = 0.5$ auf dem Kreis liegen. Wie groß ist die maximale Distanz der Bézier-Kurve vom Halbkreis?

3.12. Man bestimme geeignete Bézier-Punkte, um den Umriss eines Großbuchstabens \mathbf{C} zu definieren. Wie ist vorzugehen, um den Umriss eines Großbuchstabens \mathbf{D} mittels einer einzigen Bézier-Kurve festzulegen?

3.13. Wie lauten die Fourier-Reihen von folgenden (2π) -periodischen Funktionen, die in der Elektrotechnik bedeutungsvoll sind?

a) $f_1(x) = |\sin(x)|$, (kommutierte Sinusfunktion)

b) $f_2(x) = \begin{cases} \sin(x), & 0 \leq x \leq \pi, \\ 0, & \pi \leq x \leq 2\pi, \end{cases}$ (gleichgerichtete Sinusfunktion)

c) $f_3(x) = \begin{cases} 1, & 0 < x < \pi, \\ -1, & \pi < x < 2\pi, \end{cases}$ (Rechteckfunktion)

d) $f_4(x) = x - \pi, \quad 0 < x < 2\pi,$ (Sägezahmfunktion)

Aus den Fourier-Reihen leite man auf Grund von Satz 3.26 für geeignete Wahl von x die Werte von speziellen Reihen her. Um das Konvergenzverhalten der Fourier-Reihen zu studieren, stelle man den Verlauf von Fourier-Polynomen $g_n(x)$ für $3 \leq n \leq 8$ graphisch dar.

3.14. Zu den Funktionen $f_i(x)$, $i = 1, 2, 3, 4$, von Aufgabe 3.13 berechne man die Näherungswerte (3.167) der Fourier-Koeffizienten für $N = 8, 16, 32, 64$ mit der schnellen Fourier-Transformation. An den erhaltenen Näherungswerten verifizierte man die Relationen (3.179) und (3.180). Wie groß muss N gewählt werden, damit die Näherungswerte a_k^* und b_k^* die Fourier-Koeffizienten a_k und b_k höchstens mit einem Fehler von 10^{-6} approximieren?

3.15. Mit den Näherungen a_k^* und b_k^* der Fourier-Koeffizienten der Funktionen $f_i(x)$ von Aufgabe 3.13 für $N = 16$ stelle man die zugehörigen Fourier-Polynome $g_4^*(x)$ und $g_8^*(x)$ zusammen

mit den Funktionen $f_i(x)$ graphisch dar. In welchem Sinn approximieren die Fourier-Polynome die gegebenen Funktionen?

3.16. Es sei $N = 2n + 1, n \in \mathbb{N}^*$. Man zeige, dass

$$g_n^*(x) := \frac{1}{2}a_0^* + \sum_{k=1}^n \{a_k^* \cos(kx) + b_k^* \sin(kx)\}$$

mit den Koeffizienten (3.167) das eindeutige, interpolierende Fourier-Polynom zu den Stützstellen x_j (3.166) mit den Stützwerten $f(x_j), j = 0, 1, \dots, N$, ist.

3.17. Die Funktion $f(x) = \sin\left(\frac{\pi}{2}x\right)$ ist im Intervall $[-1, 1]$ durch eine Entwicklung nach Tschebyscheff-Polynomen $g_n(x)$ (3.209) für $n = 3, 5, 7, 9, 11$ zu approximieren. Man berechne die Entwicklungskoeffizienten c_k näherungsweise als Fourier-Koeffizienten mit $N = 32$ und schätze auf Grund der erhaltenen Resultate den Fehler in den Koeffizienten c_k^* ab. Wie groß sind die Schranken der Approximationsfehler $|g_n(x) - f(x)|$? Welches ist der qualitative Verlauf der Fehlerfunktionen $\varepsilon_n(x) = g_n(x) - f(x)$?

3.18. Die Funktion $f(x) = \sin\left(\frac{\pi}{2}x\right)$ ist für $n = 3, 5, 7, 9$ im Intervall $[-1, 1]$ durch ein Polynom n -ten Grades in der Form einer Entwicklung nach Legendre-Polynomen im quadratischen Mittel zu approximieren. Die Entwicklungskoeffizienten c_k können unter Verwendung von Rekursionsformeln für Integrale explizit angegeben werden. Sind die resultierenden Formeln für die numerische Auswertung besonders gut geeignet? Man vergleiche die Koeffizienten der approximierenden Polynome $g_n(x)$ mit den entsprechenden der abgebrochenen Taylor-Reihen $t_n(x)$ von $f(x)$. Schließlich stelle man die Fehlerfunktionen $\varepsilon_n(x) = g_n(x) - f(x)$ und $\delta_n(x) = t_n(x) - f(x)$ dar und bestimme daraus das Verhältnis der maximalen Fehlerbeträge.

3.19. Die Laguerre-Polynome $L_n(x)$ können definiert werden durch

$$L_n(x) := e^x \frac{d^n}{dx^n} (x^n e^{-x}), \quad n = 0, 1, 2, \dots$$

a) Wie lauten die Laguerre-Polynome $L_0(x), L_1(x), \dots, L_6(x)$? Man zeige, dass $L_n(x)$ auf Grund der Definition ein Polynom vom echten Grad n ist, und dass die allgemeine Darstellung gilt

$$L_n(x) = n! \sum_{j=0}^n \binom{n}{j} \frac{(-x)^j}{j!}, \quad n = 0, 1, 2, \dots$$

b) Man zeige die Orthogonalitätseigenschaft der $L_n(x)$

$$\int_0^\infty e^{-x} L_i(x) L_j(x) dx = 0 \quad \text{für alle } i \neq j, \quad i, j \in \mathbb{N}.$$

c) Man beweise, dass das Laguerre-Polynom $L_n(x)$ im Intervall $(0, \infty)$ n einfache Nullstellen besitzt.

d) Man zeige, dass die Rekursionsformel gilt

$$L_{n+1}(x) = (2n + 1 - x)L_n(x) - n^2 L_{n-1}(x), \quad n = 1, 2, 3, \dots$$

4 Nichtlineare Gleichungen

Eines der Grundprobleme der Mathematik, dem wir schon in der Schule begegnen, ist die Bestimmung von Nullstellen einer gegebenen Funktion. Es entspricht der Lösung einer nichtlinearen Gleichung.

In Anwendungsproblemen ist die Lösung einer solchen Gleichung oder eines Systems von nichtlinearen Gleichungen oft Teilaufgabe einer komplexeren Problemstellung. Eine exakte analytische Lösung dieser Aufgabe wird selten möglich sein. Deshalb werden iterative Verfahren verwendet, welche eine gesuchte Lösung als Grenzwert einer Folge von Näherungen liefern. Als theoretische Grundlage für das Studium der Konvergenzeigenschaften werden wir zuerst den Banachschen Fixpunktsatz in allgemeiner Formulierung bereitstellen. Für die Brauchbarkeit und Effizienz eines Verfahrens ist das Konvergenzverhalten der Näherungsfolge gegen die Lösung entscheidend. Unter diesem Gesichtspunkt werden einige Methoden zur Bestimmung einer Lösung einer nichtlinearen Gleichung in einer Unbekannten entwickelt und betrachtet. Dabei werden wir neben Standard-Verfahren auch solche kennen lernen, die sich als Black-box-Verfahren eignen und deshalb gern in Softwarepaketen Verwendung finden. Anschließend werden die Überlegungen auf Systeme übertragen. Als wichtigen Spezialfall und teilweise als Anwendung der Methoden behandeln wir abschließend die Berechnung von Polynom-Nullstellen.

4.1 Theoretische Grundlagen

4.1.1 Problemstellung

Gegeben sei eine stetige Vektorfunktion $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

Gesucht ist eine Nullstelle von $\mathbf{f}(\mathbf{x})$, d.h. ein fester Vektor $\mathbf{s} \in \mathbb{R}^n$ mit

$$\boxed{\mathbf{f}(\mathbf{s}) = \mathbf{0}.} \quad (4.1)$$

(4.1) ist ein System von n nichtlinearen Gleichungen mit n Unbekannten:

$$\begin{aligned} f_1(s_1, s_2, \dots, s_n) &= 0, \\ &\dots \\ f_n(s_1, s_2, \dots, s_n) &= 0. \end{aligned} \quad (4.2)$$

Um dieses Problem der iterativen Lösung zugänglich zu machen, wird es zunächst in ein

äquivalentes *Fixpunktproblem*

$$\boxed{\mathbf{x} = \mathbf{F}(\mathbf{x})} \quad (4.3)$$

umgeformt, für das gilt

$$\mathbf{f}(\mathbf{s}) = \mathbf{0} \iff \mathbf{s} = \mathbf{F}(\mathbf{s}). \quad (4.4)$$

\mathbf{s} heißt dann *Fixpunkt* von \mathbf{F} .

Eine große Klasse von Iterationsverfahren zur Lösung solcher nichtlinearen Gleichungssysteme hat die Form

$$\boxed{\mathbf{x}^{(k+1)} = \mathbf{F}(\mathbf{x}^{(k)}), \quad k = 0, 1, 2, \dots,} \quad (4.5)$$

wo $\mathbf{x}^{(k)}$ eine reelle Zahl, ein Vektor oder auch eine Funktion mit bestimmten Eigenschaften sein kann und $\mathbf{F}(\mathbf{x})$ eine Abbildung der betreffenden Menge in sich darstellt. Mit (4.5) wird zu einem gegebenen Startwert $\mathbf{x}^{(0)}$ eine Folge von Iterierten $\mathbf{x}^{(k)}$ definiert mit dem Ziel, die Gleichung zu lösen. Die Iterationsvorschrift (4.5) nennt man *Fixpunktiteration*, oder man spricht auch von der *Methode der sukzessiven Approximation*. Da in (4.5) zur Definition des Folgeelementes $\mathbf{x}^{(k+1)}$ nur das Element $\mathbf{x}^{(k)}$ benötigt wird, und da die angewandte Rechenvorschrift von k unabhängig sein soll, findet man in der Literatur auch die präzisierende Bezeichnung als einstufiges, stationäres Iterationsverfahren.

Beispiel 4.1. Gesucht ist die Lösung des Gleichungssystems

$$\begin{aligned} x_1^2 + x_2 - 4 &= 0, \\ x_2 e^{x_1} - 2 &= 0. \end{aligned}$$

Hier sind also offenbar

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 + x_2 - 4, \\ f_2(x_1, x_2) &= x_2 e^{x_1} - 2. \end{aligned}$$

Beide Gleichungen können nach x_2 aufgelöst werden:

$$\begin{aligned} x_2 &= 4 - x_1^2 =: g_1(x_1), \\ x_2 &= 2 e^{-x_1} =: g_2(x_1). \end{aligned}$$

Gesucht sind also Schnittpunkte dieser beiden Funktionen $g_k(x_1)$, $k = 1, 2$. Deshalb kann man sich eine Näherung durch eine Zeichnung verschaffen, siehe Abb. 4.1. Man erkennt an der Zeichnung, dass es zwei Lösungen des Problems gibt, da die Funktionen zwei Schnittpunkte haben. Für beide bekommt man auch recht gute Näherungslösungen durch Ablesen der Werte aus der Zeichnung. Eine Fixpunktgleichung bekommt man, wenn man die erste Gleichung nach x_1 auflöst:

$$\begin{aligned} x_1 &= \sqrt{4 - x_2} =: F_1(x_1, x_2), \\ x_2 &= 2 e^{-x_1} =: F_2(x_1, x_2). \end{aligned}$$

Geht man jetzt mit irgendwelchen Startwerten in das Iterationsverfahren (4.5), so stellt sich schnell Konvergenz gegen den rechten Schnittpunkt ein. Das liegt daran, dass der linke Schnittpunkt ein *abstoßender* Fixpunkt ist. Für seine Bestimmung ist die Iteration also nicht geeignet.

Das Beispiel zeigt, dass die Eindeutigkeit von Lösungen bei nichtlinearen Problemen selbst bei einfachen Aufgaben oft nicht gegeben ist. Das gilt auch für die Existenz und für die Konvergenz einfacher Iterationsverfahren.

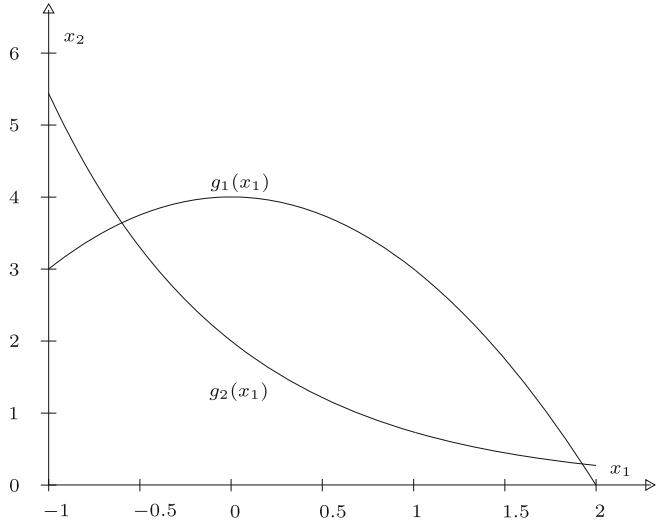


Abb. 4.1 Schnittpunkte zweier Funktionen.

△

4.1.2 Konvergenztheorie und Banachscher Fixpunktsatz

Zur einheitlichen theoretischen Behandlung der Iterationsverfahren (4.5) wird den Betrachtungen ein Banach-Raum zu Grunde gelegt.

Definition 4.1. Ein *Banach-Raum* B ist ein normierter, linearer Vektorraum über einem Zahlenkörper \mathbb{K} (\mathbb{R} oder \mathbb{C}), in dem jede Cauchy-Folge $\mathbf{x}^{(k)}$ von Elementen aus B gegen ein Element in B konvergiert. Dabei hat eine Norm in B die Eigenschaften

$$\begin{aligned}\|\mathbf{x}\| &\geq 0 && \text{für alle } \mathbf{x} \in B, \\ \|\mathbf{x}\| = 0 &\Leftrightarrow \mathbf{x} = \mathbf{0}, \\ \|\gamma \mathbf{x}\| &= |\gamma| \cdot \|\mathbf{x}\| && \text{für alle } \gamma \in \mathbb{K}, \mathbf{x} \in B, \\ \|\mathbf{x} + \mathbf{y}\| &\leq \|\mathbf{x}\| + \|\mathbf{y}\| && \text{für alle } \mathbf{x}, \mathbf{y} \in B.\end{aligned}$$

Der Körper der reellen Zahlen \mathbb{R} mit dem Betrag als Norm ist ein Banach-Raum. Allgemeiner ist der n -dimensionale reelle Vektorraum \mathbb{R}^n mit irgendeiner Vektornorm $\|\mathbf{x}\|$ ein Banach-Raum. Schließlich bildet auch die Menge der stetigen Funktionen über einem abgeschlossenen Intervall $I = [a, b]$, d.h. $C(I)$ mit der Norm

$$\|f\| := \max_{x \in I} |f(x)|, \quad f(x) \in C(I) \tag{4.6}$$

einen Banach-Raum.

Definition 4.2. Die Folge $\{\mathbf{x}^{(k)}\}$ mit dem Grenzwert s hat mindestens die *Konvergenzordnung* $p \geq 1$, wenn es eine von k unabhängige Konstante $K > 0$ gibt, so dass

$$\|\mathbf{x}^{(k+1)} - s\| \leq K \|\mathbf{x}^{(k)} - s\|^p \quad \forall k \geq 0 \quad (4.7)$$

mit $K < 1$, falls $p = 1$. K wird *Konvergenzrate* oder *Fehlerkonstante* genannt.

Die Konvergenz heißt *linear*, falls $p = 1$, sie heißt *quadratisch*, falls $p = 2$, und *kubisch*, falls $p = 3$ ist. Eine höhere als kubische Konvergenz kommt nur sehr selten vor. K und p bestimmen die *Konvergenzgeschwindigkeit* der Folge. Je kleiner K und je größer p sind, desto schneller ist die Konvergenz. Dabei bestimmt die Ordnung das Verhalten wesentlich stärker als die Konstante K . Diese ist wichtig beim Vergleich linear konvergenter Verfahren.

Definition 4.3. Für eine abgeschlossene Teilmenge $A \subset B$ eines Banach-Raumes B sei eine Abbildung $\mathbf{F} : A \rightarrow A$ von A in A gegeben. Die Abbildung \mathbf{F} heißt *Lipschitz-stetig* auf A mit der Lipschitz-Konstanten $0 < L < \infty$, wenn gilt

$$\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\| \quad \text{für alle } \mathbf{x}, \mathbf{y} \in A. \quad (4.8)$$

Die Abbildung \mathbf{F} nennt man *kontrahierend* auf A , falls die Lipschitz-Konstante in (4.8) $L < 1$ ist.

Satz 4.4. (*Banachscher Fixpunktsatz*) Sei A eine abgeschlossene Teilmenge eines Banach-Raumes B und sei $\mathbf{F} : A \rightarrow A$ eine kontrahierende Abbildung. Dann gilt:

1. Die Abbildung \mathbf{F} besitzt genau einen Fixpunkt $s \in A$.

2. Für jeden Startwert $\mathbf{x}^{(0)} \in A$ konvergiert die durch

$$\mathbf{x}^{(k+1)} = \mathbf{F}(\mathbf{x}^{(k)}), \quad k = 0, 1, 2, \dots, \quad (4.9)$$

definierte Folge gegen den Fixpunkt $s \in A$.

3.

$$\|\mathbf{x}^{(k)} - s\| \leq L^k \|\mathbf{x}^{(0)} - s\|. \quad (4.10)$$

4. *A priori Fehlerabschätzung*:

$$\|\mathbf{x}^{(k)} - s\| \leq \frac{L^k}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|. \quad (4.11)$$

5. *A posteriori Fehlerabschätzung*:

$$\|\mathbf{x}^{(k)} - s\| \leq \frac{L}{1-L} \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|. \quad (4.12)$$

Beweis. Als erstes wird die Differenz zweier aufeinander folgender Iterierter abgeschätzt:

$$\begin{aligned} \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| &= \|\mathbf{F}(\mathbf{x}^{(k)}) - \mathbf{F}(\mathbf{x}^{(k-1)})\| \\ &\leq L \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| = L \|\mathbf{F}(\mathbf{x}^{(k-1)}) - \mathbf{F}(\mathbf{x}^{(k-2)})\| \\ &\leq L^2 \|\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)}\| \leq \dots \end{aligned}$$

oder allgemein

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq L^{k-l} \|\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}\| \quad \text{für } 0 \leq l \leq k. \quad (4.13)$$

Wegen der Voraussetzung $\mathbf{F} : A \rightarrow A$ liegt mit $\mathbf{x}^{(0)} \in A$ auch jedes $\mathbf{x}^{(k)} \in A$. Jetzt zeigen wir, dass die durch (4.9) definierte Folge $\mathbf{x}^{(k)}$ für jedes $\mathbf{x}^{(0)} \in A$ konvergiert. Wegen (4.13) mit $l = 0$ bilden die $\mathbf{x}^{(k)}$ eine Cauchy-Folge, denn es gilt für beliebige $m \geq 1$ und $k \geq 1$

$$\begin{aligned}\|\mathbf{x}^{(k+m)} - \mathbf{x}^{(k)}\| &= \|\mathbf{x}^{(k+m)} - \mathbf{x}^{(k+m-1)} + \mathbf{x}^{(k+m-1)} - \dots - \mathbf{x}^{(k)}\| \\ &\leq \sum_{\mu=k}^{k+m-1} \|\mathbf{x}^{(\mu+1)} - \mathbf{x}^{(\mu)}\| \\ &\leq L^k (L^{m-1} + L^{m-2} + \dots + L + 1) \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \\ &= L^k \frac{1-L^m}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|. \end{aligned} \quad (4.14)$$

Wegen $L < 1$ existiert somit zu jedem $\varepsilon > 0$ ein $N \in \mathbb{N}^*$, so dass $\|\mathbf{x}^{(k+m)} - \mathbf{x}^{(k)}\| < \varepsilon$ für $k \geq N$ und $m \geq 1$ ist. Deshalb besitzt die Folge $\mathbf{x}^{(k)}$ in der abgeschlossenen Teilmenge A einen Grenzwert

$$\mathbf{s} := \lim_{k \rightarrow \infty} \mathbf{x}^{(k)}, \quad \mathbf{s} \in A.$$

Wegen der Stetigkeit der Abbildung \mathbf{F} gilt weiter

$$\mathbf{F}(\mathbf{s}) = \mathbf{F}(\lim_{k \rightarrow \infty} \mathbf{x}^{(k)}) = \lim_{k \rightarrow \infty} \mathbf{F}(\mathbf{x}^{(k)}) = \lim_{k \rightarrow \infty} \mathbf{x}^{(k+1)} = \mathbf{s}.$$

Damit ist nicht allein die Existenz eines Fixpunktes $\mathbf{s} \in A$ nachgewiesen, sondern gleichzeitig auch die Konvergenz der Folge $\mathbf{x}^{(k)}$ gegen einen Fixpunkt.

Die Eindeutigkeit des Fixpunktes zeigen wir indirekt. Es seien $\mathbf{s}_1 \in A$ und $\mathbf{s}_2 \in A$ zwei Fixpunkte der Abbildung \mathbf{F} mit $\|\mathbf{s}_1 - \mathbf{s}_2\| > 0$. Da $\mathbf{s}_1 = \mathbf{F}(\mathbf{s}_1)$ und $\mathbf{s}_2 = \mathbf{F}(\mathbf{s}_2)$ gelten, führt dies wegen $\|\mathbf{s}_1 - \mathbf{s}_2\| = \|\mathbf{F}(\mathbf{s}_1) - \mathbf{F}(\mathbf{s}_2)\| \leq L\|\mathbf{s}_1 - \mathbf{s}_2\|$ auf den Widerspruch $L \geq 1$.

In (4.14) halten wir k fest und lassen $m \rightarrow \infty$ streben. Wegen $\lim_{m \rightarrow \infty} \mathbf{x}^{(k+m)} = \mathbf{s}$ und $L < 1$ ergibt sich dann die Fehlerabschätzung (4.11).

(4.10) ergibt sich aus

$$\begin{aligned}\|\mathbf{x}^{(k)} - \mathbf{s}\| &= \|\mathbf{F}(\mathbf{x}^{(k-1)}) - \mathbf{F}(\mathbf{s})\| \leq L\|\mathbf{x}^{(k-1)} - \mathbf{s}\| \\ &= L\|\mathbf{F}(\mathbf{x}^{(k-2)}) - \mathbf{F}(\mathbf{s})\| \leq L^2\|\mathbf{x}^{(k-2)} - \mathbf{s}\| = \dots \\ &\leq L^k \|\mathbf{x}^{(0)} - \mathbf{s}\|. \end{aligned}$$

(4.12) folgt schließlich aus (4.11) durch Umnummerierung $k \rightarrow 1$, d.h. es wird $\mathbf{x}^{(k-1)}$ als Startwert $\mathbf{x}^{(0)}$ interpretiert, dann wird $\mathbf{x}^{(k)}$ zu $\mathbf{x}^{(1)}$ und die beiden Abschätzungen werden identisch. \square

Die a priori Fehlerabschätzung gestattet, nach Berechnung von $\mathbf{x}^{(1)}$ aus dem Startwert $\mathbf{x}^{(0)}$ den Fehler von $\mathbf{x}^{(k)}$ gegenüber \mathbf{s} mit einer absoluten Schranke vorherzusagen. Gleichzeitig besagt (4.11), dass die Norm des Fehlers $\mathbf{x}^{(k)} - \mathbf{s}$ mindestens wie eine geometrische Folge mit dem Quotienten L abnimmt. Je kleiner die Lipschitz-Konstante ist, desto besser ist die

Konvergenz der Folge $\mathbf{x}^{(k)}$ gegen s . Mit der a posteriori Fehlerabschätzung kann nach k ausgeführten Iterationsschritten die Abweichung von $\mathbf{x}^{(k)}$ gegenüber s abgeschätzt werden.

Die Aussage von Satz 4.4 hat in vielen praktischen Anwendungen nur lokalen Charakter, denn die abgeschlossene Teilmenge A kann sehr klein sein. Auch wird es häufig schwierig sein, im konkreten Fall die Menge A quantitativ zu beschreiben, und man wird sich mit der Existenz zufrieden geben müssen.

Ist für $n = 1$ die Funktion F mindestens einmal stetig differenzierbar, dann kann nach dem Mittelwertsatz die Lipschitz-Konstante durch das Betragsmaximum der ersten Ableitung in A ersetzt werden. Diese Aussage wird im folgenden Satz, dessen Beweis wir als Übungsaufgabe stellen wollen, verallgemeinert.

Satz 4.5. *Es sei $n = 1$. In der abgeschlossenen Umgebung A des Fixpunktes s sei die Iterationsfunktion F $p + 1$ mal stetig differenzierbar.*

Dann ist das Verfahren mindestens linear konvergent, falls

$$|F'(s)| < 1. \quad (4.15)$$

Das Verfahren ist von mindestens p -ter Ordnung, $p \geq 2$, falls

$$F^{(k)}(s) = 0 \quad \text{für } k = 1, 2, \dots, p - 1. \quad (4.16)$$

Beispiel 4.2. Gesucht sei die Lösung der nichtlinearen Gleichung

$$x = e^{-x} =: F(x), \quad x \in \mathbb{R}, \quad (4.17)$$

d.h. der Fixpunkt der Abbildung $F : \mathbb{R} \rightarrow \mathbb{R}_{>0}$. Um den Fixpunktsatz anwenden zu können, benötigen wir im Banach-Raum $B = \mathbb{R}$ ein abgeschlossenes Intervall A , für welches die Voraussetzungen zutreffen. Beispielsweise wird $A := [0.5, 0.69]$ durch die Abbildung F (4.17) in sich abgebildet. Nach Satz 4.5 ist die Lipschitz-Konstante L für die stetig differenzierbare Funktion F gegeben durch

$$L = \max_{x \in A} |F'(x)| = \max_{x \in A} |-e^{-x}| = e^{-0.5} \doteq 0.606531 < 1.$$

Folglich ist F in A eine kontrahierende Abbildung, und es existiert in A ein eindeutiger Fixpunkt s . Das Ergebnis der Fixpunktiteration ist in Tab. 4.1 für den Startwert $x^{(0)} = 0.55 \in A$ auszugsweise bei achtstelliger Dezimalrechnung wiedergegeben.

Tab. 4.1 Fixpunktiteration.

k	$x^{(k)}$	k	$x^{(k)}$	k	$x^{(k)}$
0	0.55000000	10	0.56708394	20	0.56714309
1	0.57694981	11	0.56717695	21	0.56714340
2	0.56160877	12	0.56712420	22	0.56714323
3	0.57029086	13	0.56715412	23	0.56714332
4	0.56536097	14	0.56713715	24	0.56714327

Auf Grund der beiden ersten Werte $x^{(0)}$ und $x^{(1)}$ kann mit der a priori Fehlerabschätzung (4.11) die Zahl k der Iterationen geschätzt werden, die nötig sind, damit die Abweichung $|x^{(k)} - s| \leq \varepsilon = 10^{-6}$ ist. Man erhält aus (4.11)

$$k \geq \ln \left(\frac{\varepsilon(1 - L)}{|x^{(1)} - x^{(0)}|} \right) / \ln(L) \doteq 22.3 \quad (4.18)$$

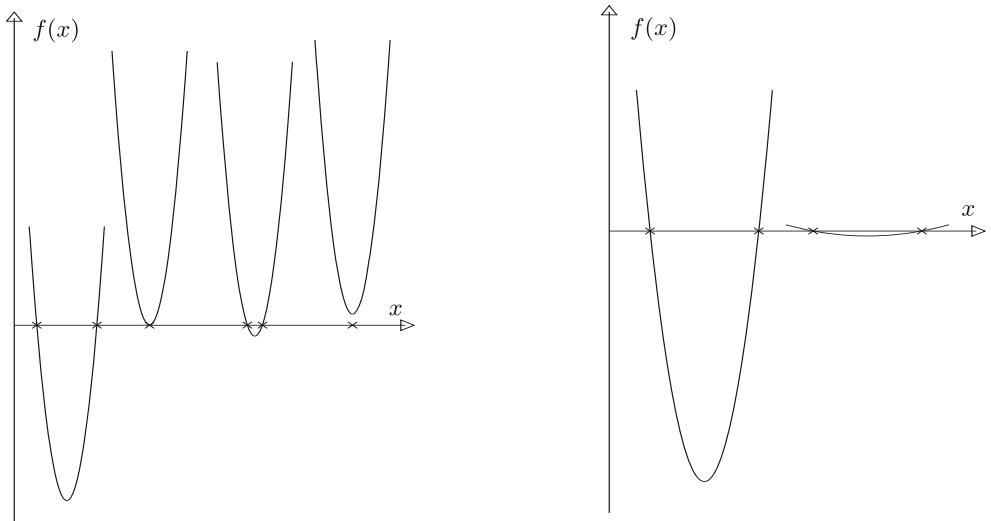


Abb. 4.2 Unterschiedlich konditionierte Nullstellen.

eine leichte, zu erwartende Überschätzung, wie aus Tab. 4.1 ersichtlich ist.

Für den iterierten Wert $x^{(12)}$ liefert (4.11) die a priori Fehlerschranke $|x^{(12)} - s| \leq 1.70 \cdot 10^{-4}$, während (4.12) die bessere a posteriori Fehlerschranke $|x^{(12)} - s| \leq 8.13 \cdot 10^{-5}$ ergibt. Da der Fixpunkt $s \doteq 0.56714329$ ist, beträgt die Abweichung tatsächlich $|x^{(12)} - s| \doteq 1.91 \cdot 10^{-5}$. Für $x^{(23)}$ erhalten wir nach (4.12) die sehr realistische Abschätzung $|x^{(23)} - s| \leq 1.4 \cdot 10^{-7}$, die nur etwa fünfmal so groß ist. Sie zeigt, dass nicht 23 Iterationen nötig sind, um die oben geforderte absolute Genauigkeit zu erreichen. \triangle

4.1.3 Stabilität und Kondition

Wir wollen an verschiedenen eindimensionalen Beispielen auf typische Stabilitätsunterschiede beim Nullstellenproblem hinweisen. Neben der Schwierigkeit, im allgemeinen Fall Konvergenz zu erzielen, kann es aus Stabilitätsgründen auch schwierig sein die gewünschte Genauigkeit bei der Lokalisierung von Nullstellen zu erreichen.

In Abb. 4.2 sehen wir links vier verschiedene Parabeln. Wir wollen die unterschiedliche Situation für die Nullstellenbestimmung dieser Funktionen von links nach rechts beschreiben:

1. Stabile Situation mit zwei gut getrennten und leicht zu berechnenden Nullstellen.
2. Doppelte Nullstelle, hier gilt also

$$f(s) = f'(s) = 0. \quad (4.19)$$

Die numerische Berechnung einer doppelten Nullstelle ist immer schlecht konditioniert, weil eine leichte Verschiebung nach oben oder unten – z.B. durch Rundungsfehler – zu zwei oder keiner reellen Nullstelle führt, siehe 3. und 4.

3. Eng beieinander liegende Nullstellen führen leicht zu numerischen Schwierigkeiten, besonders, wenn man nicht irgendeine Nullstelle, sondern mehrere oder alle Nullstellen sucht.

4. Diese Funktion hat nur komplexe Nullstellen. Um sie zu finden, muss komplex gerechnet werden, oder es muss ein spezieller Algorithmus angewendet werden, der Realteil und Imaginärteil eines solchen Nullstellenpaars im Reellen berechnet, siehe Abschnitt 4.4. Auch dann kann die Aufgabe der Berechnung noch schlecht konditioniert sein, weil die beiden komplexen Nullstellen nahe bei einer doppelten reellen Nullstelle liegen.

Wilkinson untersucht in [Wil 94] die unterschiedlichsten Situationen bei der Bestimmung von Polynomnullstellen. Er definiert Konditionszahlen für die Nullstellen und bestimmt sie für verschiedene Probleme. Dabei sieht man, dass gewisse Nullstellen selbst bei einer scheinbar harmlosen Verteilung schlecht konditioniert sein können.

Ein solches Beispiel sehen wir in Abb. 4.2 rechts. Die Nullstellen der beiden Funktionen haben den gleichen Abstand voneinander. Betrachtet man aber ein Intervall $[x_-, x_+]$ um jede dieser Nullstellen, so dass

$$|f(x)| < \varepsilon \quad \forall x \in [x_-, x_+] , \quad (4.20)$$

so bekommt man für die linke Funktion bei kleinem ε zwei kleine und deutlich getrennte Intervalle für die Nullstellen. Die entsprechenden Intervalle bei der rechten Funktion sind viel größer und gehen schon für nicht gar zu kleines ε ineinander über.

4.2 Gleichungen in einer Unbekannten

Wir betrachten die Aufgabe, zu einer stetigen, nichtlinearen Funktion $f(x)$ mit $f : \mathbb{R} \rightarrow \mathbb{R}$ Lösungen der Gleichung

$$\boxed{f(x) = 0} \quad (4.21)$$

zu berechnen. Wir behandeln hauptsächlich den Fall von reellen Lösungen von (4.21), werden aber an einigen Stellen darauf hinweisen, wie auch komplexe Lösungen gefunden werden können.

4.2.1 Das Verfahren der Bisektion

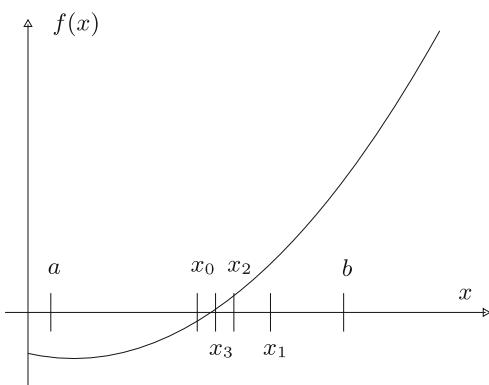


Abb. 4.3
Das Verfahren der Bisektion.

Die *Bisektion* ist durch Überlegungen der Analysis motiviert, die reelle Lösung einer Gleichung (4.21) durch systematisch kleiner werdende Intervalle einzuschließen. Dazu geht man von der Annahme aus, es sei ein Intervall $I = [a, b]$ bekannt, so dass $f(a) \cdot f(b) < 0$ gilt. Aus Stetigkeitsgründen existiert eine Lösung s im Innern von I mit $f(s) = 0$. Für den Mittelpunkt $\mu = (a+b)/2$ wird der Funktionswert $f(\mu)$ berechnet. Ist $f(\mu) \neq 0$, entscheidet sein Vorzeichen, in welchem der beiden Teilintervalle $[a, \mu]$ und $[\mu, b]$ die gesuchte Lösung s ist. Die Lage der Lösung s ist auf das Innere eines gegenüber I halb so langen Intervalls eingeschränkt, und das Verfahren kann fortgesetzt werden, bis s im Innern eines hinreichend kleinen Intervalls liegt. Bezeichnen wir mit $L_0 := b - a$ die Länge des Startintervalls, so bildet die Folge der Längen $L_k := (b - a)/2^k$, $k = 0, 1, 2, \dots$ eine Nullfolge. Der Mittelpunkt $x^{(k)}$ des Intervalls nach k Intervallhalbierungen stellt für s eine Näherung dar mit der a priori Fehlerabschätzung

$$|x^{(k)} - s| \leq \frac{b - a}{2^{k+1}}, \quad k = 0, 1, 2, \dots \quad (4.22)$$

Da diese Fehlerschranke wie eine geometrische Folge mit dem Quotienten $q = \frac{1}{2}$ abnimmt, ist die Konvergenzordnung $p = 1$.

Beispiel 4.3. Die Berechnung der kleinsten positiven Lösung der Gleichung

$$f(x) := \cos(x) \cosh(x) + 1 = 0 \quad (4.23)$$

werden wir mit verschiedenen Verfahren durchführen, die wir dadurch vergleichen können. Wir beginnen mit der Bisektion. Als Startintervall verwenden wir $I = [0, 3]$, für welches $f(0) = 2$, $f(3) \doteq -9$ und damit $f(0) f(3) < 0$ gilt. Die Ergebnisse sind auszugweise in Tab. 4.2 zusammengestellt. Wir haben mit 16 wesentlichen Dezimalstellen gerechnet und die Iteration abgebrochen, wenn die Intervallbreite kleiner als $2.0 \cdot 10^{-9}$ oder der Funktionswert im Mittelpunkt dem Betrage nach kleiner als $1.0 \cdot 10^{-9}$ war. Das erste Kriterium ist nach 31 Schritten erfüllt, da $3 \cdot 2^{-31} \approx 1.4 \cdot 10^{-9}$ ist. \triangle

Tab. 4.2 Bisektion.

k	a	b	μ	$f(\mu)$
0	0.0	3.0	1.5	> 0
1	1.5	3.0	2.25	< 0
2	1.5	2.25	1.875	> 0
3	1.875	2.25	2.0625	< 0
4	1.875	2.0625	1.96875	< 0
5	1.875	1.96875	1.921875	< 0
\vdots	\vdots	\vdots	\vdots	\vdots
30	1.8751040669	1.8751040697	1.8751040683	> 0
31	1.8751040683	1.8751040697	1.8751040690	$-1.2 \cdot 10^{-9}$

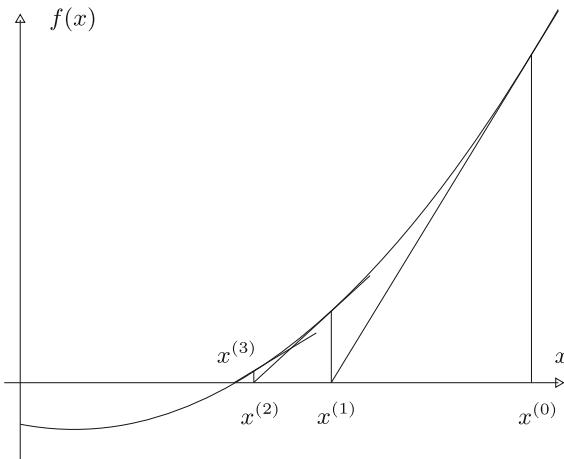


Abb. 4.4 Konvergente Folge des Newton-Verfahrens.

4.2.2 Das Verfahren von Newton

Ist die Funktion $f(x)$ der zu lösenden Gleichung $f(x) = 0$ stetig differenzierbar, und ist überdies die erste Ableitung ohne großen Rechenaufwand berechenbar, so wird im *Verfahren von Newton* die Funktion $f(x)$ im allgemeinen Näherungswert $x^{(k)}$ linearisiert und der iterierte Wert $x^{(k+1)}$ als Abszisse des Schnittpunktes der Tangente mit der x -Achse definiert, es wird sozusagen für die Bestimmung einer Näherung $x^{(k+1)}$ der Nullstelle die Funktion durch ihre Tangente in $(x^{(k)}, f(x^{(k)}))$ ersetzt, siehe Abb. 4.4.

Aus der Tangentengleichung

$$t(x) = f(x^{(k)}) + (x - x^{(k)})f'(x^{(k)})$$

und der Forderung $t(x^{(k+1)}) = 0$ ergibt sich so die Iterationsvorschrift

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k = 0, 1, 2, \dots$$

(4.24)

Die Methode von Newton gehört zur Klasse der Fixpunktiterationen mit der Funktion

$$F(x) := x - \frac{f(x)}{f'(x)}, \quad (4.25)$$

und für den Fixpunkt $s = F(s)$ gilt offenbar $f(s) = 0$. Über die Konvergenzbedingung und die Konvergenzordnung der Folge (4.24) gibt der folgende Satz Auskunft.

Satz 4.6. *Die Funktion $f(x)$ sei dreimal stetig differenzierbar in einem Intervall $I_1 = [a, b]$ mit $a < s < b$, und es sei $f'(s) \neq 0$, d.h. s sei eine einfache Nullstelle von $f(x)$. Dann existiert ein Intervall $I = [s - \delta, s + \delta]$ mit $\delta > 0$, für welches $F : I \rightarrow I$ eine kontrahierende Abbildung ist. Für jeden Startwert $x^{(0)} \in I$ ist die Folge $x^{(k)}$ des Newtonschen Verfahrens (4.24) mindestens quadratisch konvergent.*

Beweis. Um die kontrahierende Eigenschaft der Abbildung F (4.25) in einer nicht leeren Umgebung von s nachzuweisen, zeigen wir, dass der Betrag der ersten Ableitung von F in einer Umgebung von s kleiner Eins ist. Für die erste Ableitung erhalten wir

$$F'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}, \quad (4.26)$$

und deshalb ist wegen $f(s) = 0$ unter Berücksichtigung der vorausgesetzten Differenzierbarkeitseigenschaften von $f(x)$ auch $F'(s) = 0$. Aus Stetigkeitsgründen existiert dann in der Tat ein $\delta > 0$, so dass

$$-1 < F'(x) < 1 \quad \text{für alle } x \in [s - \delta, s + \delta] =: I$$

gilt. Für das Intervall I sind die Voraussetzungen des Banachschen Fixpunktsatzes erfüllt, und die Konvergenz der Folge $x^{(k)}$ gegen s ist damit gezeigt.

Nach Satz 4.5 ist die Konvergenzordnung des Verfahrens von Newton mindestens $p = 2$. \square

Das Newton-Verfahren kann auch noch bei einer doppelten Nullstelle, an der ja $f'(s) = 0$ gilt, konvergieren, aber nicht mehr quadratisch, sondern nur noch linear bzw. superlinear, d.h. mit einem Wert $1 < p < 2$ in (4.7). Die Länge des Konvergenz-Intervalls I , dessen Existenz im Satz 4.6 gezeigt worden ist, kann in manchen Fällen sehr klein sein. Die praktische Bestimmung von I aus der Darstellung (4.26) ist entsprechend kompliziert. Das Newton-Verfahren ist damit weit entfernt davon, zu den Black-box-Methoden gezählt werden zu dürfen. Eine solche Methode werden wir im Abschnitt 4.2.4 darstellen.

Die beiden in Abb. 4.5 dargestellten Fälle zeigen zwei unterschiedliche Möglichkeiten von divergenten Folgen des Newton-Verfahrens. Die linke Abbildung zeigt eine gegen $-\infty$ divergente Folge, weil der Startwert $x^{(0)}$ nicht im Konvergenzbereich liegt. Die rechte Abbildung zeigt den Fall alternierender Divergenz: Auf Grund der Tangentensteigung im Punkt $(x^{(0)}, f(x^{(0)}))$ wird ein Punkt $x^{(1)}$ berechnet, der im nächsten Schritt wieder zu $x^{(0)}$ zurückführt. Dieser Fall ist sicher sehr selten, der Grund für diese Art der Divergenz ist aber auch der, dass $x^{(0)}$ nicht im Konvergenzbereich liegt.

Beispiel 4.4. Die m -te Wurzel aus einer positiven Zahl a ist die Lösung der Gleichung

$$f(x) = x^m - a = 0, \quad a > 0. \quad (4.27)$$

Das Verfahren von Newton ergibt mit $f'(x) = mx^{m-1}$ die Vorschrift

$$x^{(k+1)} = x^{(k)} - \frac{(x^{(k)})^m - a}{m(x^{(k)})^{m-1}} = \frac{a + (m-1)(x^{(k)})^m}{m(x^{(k)})^{m-1}} \quad \text{oder}$$

$$x^{(k+1)} = \frac{1}{m} \left\{ \frac{a}{(x^{(k)})^{m-1}} + (m-1)x^{(k)} \right\}, \quad k = 0, 1, 2, \dots \quad (4.28)$$

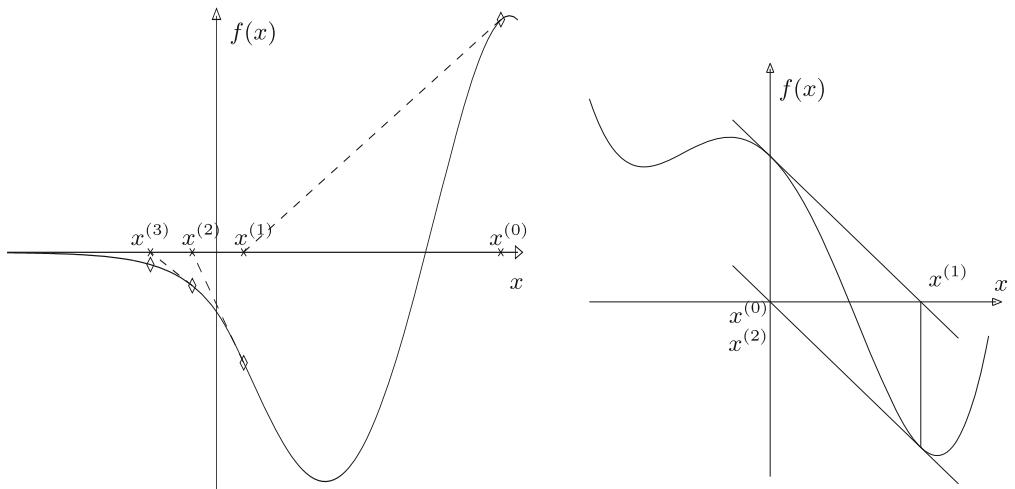


Abb. 4.5 Divergierende Folgen des Newton-Verfahrens.

Daraus leiten sich die folgenden Spezialfälle ab:

$$x^{(k+1)} = \frac{1}{2} \left[\frac{a}{x^{(k)}} + x^{(k)} \right] \quad \text{Quadratwurzel} \quad (4.29)$$

$$x^{(k+1)} = \frac{1}{3} \left[\frac{a}{(x^{(k)})^2} + 2x^{(k)} \right] \quad \text{Kubikwurzel} \quad (4.30)$$

$$x^{(k+1)} = (2 - ax^{(k)})x^{(k)} \quad m = -1, \text{ Kehrwert} \quad (4.31)$$

Da $f''(s) = m(m-1)s^{m-2} \neq 0$ ist, ist die Konvergenzordnung für das Iterationsverfahren (4.28) genau $p = 2$. (4.29) wird in Rechnern zur Bestimmung der Quadratwurzel verwendet. Um die Fehlerkonstante $K = |f''(s)/(2f'(s))| = 1/(2\sqrt{a})$ in Grenzen zu halten, wird aus a durch Multiplikation mit einer geraden Potenz der Zahlenbasis des Rechners ein Hilfswert a' gebildet, der in einem bestimmten Intervall liegt. Dann wird ein von a' abhängiger Startwert $x^{(0)}$ so gewählt, dass eine bestimmte Zahl von Iterationsschritten die Quadratwurzel mit voller Rechnergenauigkeit liefert.

Die Iteration (4.31) ermöglichte für erste Computer ohne eingebaute Division die Zurückführung dieser Operation auf Multiplikationen und Subtraktionen. Die Division von zwei Zahlen wurde so ausgeführt, dass der Kehrwert des Divisors bestimmt und dieser dann mit dem Dividenden multipliziert wurde. \triangle

Beispiel 4.5. Wir wollen die Berechnung der kleinsten positiven Lösung der Gleichung

$$\begin{aligned} f(x) &= \cos(x) \cosh(x) + 1 = 0 \quad \text{mit} \\ f'(x) &= \cos(x) \sinh(x) - \sin(x) \cosh(x) \end{aligned}$$

jetzt mit der Methode von Newton durchführen. Als Startwert nehmen wir den Mittelpunkt des Startintervalls aus Beispiel 4.3. Die Iteration wurde auf Grund des Kriteriums eines betragsmäßig hinreichend kleinen Funktionswertes $|f(x^{(5)})| < 1 \cdot 10^{-15}$ nach fünf Schritten abgebrochen. Es wurden je fünf Auswertungen von $f(x)$ und $f'(x)$ benötigt. Die Korrekturen bilden offensichtlich eine quadratisch konvergente Nullfolge. \triangle

Tab. 4.3 Methode von Newton.

k	$x^{(k)}$	$f(x^{(k)})$	$f'(x^{(k)})$	$f(x^{(k)})/f'(x^{(k)})$
0	1.5	1.16640287	-2.1958975	-0.5311736
1	2.031173635	-0.72254393	-5.1377014	0.14063564
2	1.890537993	-0.06459399	-4.2324326	0.01526163
3	1.875276323	-0.00071290	-4.1391805	0.00017225
4	1.875104090	-0.00000009	-4.1381341	0.00000002
5	1.875104069	0.00000000	—	—

Es gibt verschiedene Varianten des Newton-Verfahrens, die hier nur kurz beschrieben werden sollen.

- Das vereinfachte Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k = 0, 1, 2, \dots \quad (4.32)$$

Häufig ändert sich der Wert der ersten Ableitung im Verfahren von Newton nicht mehr stark in den letzten Iterationsschritten. Um dieser Feststellung Rechnung zu tragen, und um gleichzeitig die Zahl der oft aufwändigen Berechnungen der ersten Ableitung zu reduzieren, wird die erste Ableitung einmal für den (guten!) Startwert $x^{(0)}$ berechnet und für die weiteren Iterationen beibehalten. Die Konvergenzordnung für das vereinfachte Verfahren (4.32) ist zwar nur $p = 1$, doch ist die Fehlerkonstante $K := |f'(s)| = |1 - f'(s)/f'(x^{(0)})|$ meistens sehr klein.

- Das modifizierte Newton-Verfahren

$$x^{(k+1)} = x^{(k)} - \lambda_k \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k = 0, 1, 2, \dots \quad (4.33)$$

Um Schwierigkeiten mit der Konvergenz des Newton-Verfahrens zu vermeiden, kann die Korrektur mit einem Faktor λ_k versehen werden, der solange halbiert wird, bis der Absolutbetrag des Funktionswertes verkleinert wird $|f(x^{(k+1)})| < |f(x^{(k)})|$. Dieses Verfahren eignet sich in leichter Abwandlung auch besonders für Systeme von nichtlinearen Gleichungen, vgl. Abschnitt 4.3.2.

4.2.3 Die Sekantenmethode

Soll die Berechnung der Ableitung vermieden werden, so kann statt der Nullstelle der Tangente, die das Newton-Verfahren verwendet, die Nullstelle einer Sekante berechnet werden. Dazu werden zwei Startwerte $x^{(0)}$ und $x^{(1)}$ benötigt, die aber nicht wie bei der Bisektion die gesuchte Nullstelle einschließen müssen. Der allgemeine iterierte Wert $x^{(k+1)}$ wird als Nullstelle der Sekanten des Graphen der Funktion berechnet, die durch die Punkte $(x^{(k)}, f(x^{(k)}))$ und $(x^{(k-1)}, f(x^{(k-1)}))$ verläuft. Dies führt zur *Sekantenmethode*

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}. \quad (4.34)$$

Die Sekantenmethode ist ein *zweistufiges Iterationsverfahren*. Um jeden Schritt durchführen zu können, muss offensichtlich $f(x^{(k)}) \neq f(x^{(k-1)})$ sein.

Satz 4.7. Falls $f'(s) \neq 0$ und $f''(s) \neq 0$ gelten, ist die Konvergenzordnung der Sekantenmethode $p = \frac{1}{2}(1 + \sqrt{5}) \doteq 1.618$.

Beweis. Siehe [Sch 97]. □

Leider gibt es bei der Sekantenmethode Fälle sehr langsamer Konvergenz, bei denen also die superlineare Konvergenz, die ja nur asymptotisch gilt, erst sehr spät eintritt, siehe Beispiel 4.6 und Abschnitt 4.2.4.

Beispiel 4.6. Zur Berechnung der kleinsten positiven Lösung der Gleichung

$$f(x) = \cos(x) \cosh(x) + 1 = 0$$

wird die Sekantenmethode angewandt. Wir verwenden jetzt das Startintervall $I = [1.5, 3.0]$ bzw. $x^{(0)} = 3.0$ und $x^{(1)} = 1.5$. In diesem Intervall hat f einen Vorzeichenwechsel. Auch das größere Intervall $I = [0, 3.0]$ erzeugt Konvergenz, die aber auf Grund großer Sprünge in den ersten Werten von $x^{(k)}$ sehr langsam ist, vgl. Tab. 4.6. Es wurde wieder mit sechzehnstelliger Dezimalrechnung gerechnet. Die Iteration wurde abgebrochen, sobald $|f(\mu)| \leq 10^{-9}$ galt. Das Verfahren liefert die Lösung in acht Iterationsschritten gemäß Tab. 4.4.

Tab. 4.4 Sekantenmethode.

k	$x^{(k)}$	$f(x^{(k)})$	$ x^{(k)} - s $
0	3.0	-8.967	1.125
1	1.5	1.166	0.375
2	1.6726587	0.7196	0.202
3	1.7712715	0.3978	0.104
4	1.8931365	-0.07561	0.0180
5	1.8736712	0.005923	0.00143
6	1.8750852	$7.785 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
7	1.87510409	$-8.194 \cdot 10^{-8}$	$2 \cdot 10^{-8}$
8	1.8751040687	$1.131 \cdot 10^{-12}$	$1 \cdot 10^{-13}$

Die superlineare Konvergenz mit $p \doteq 1.618$ und $K \approx 1$ ist etwa ab dem 3. oder 4. Schritt gut zu erkennen. △

4.2.4 Brents Black-box-Methode

Methoden, die *garantiert* konvergieren, werden auch Black-box-Verfahren genannt. Zu ihnen gehört die Bisektion, die aber zu langsam konvergiert. Es gibt eine Reihe von Methoden, die versuchen, die Sicherheit der Bisektion mit der größeren Konvergenzgeschwindigkeit des Newton-Verfahrens zu verbinden. Sie entstehen durch Kombination verschiedener Verfahrenskomponenten und sind deshalb in ihrer Definition komplexer als die Verfahren, die wir bisher kennen gelernt haben. Wir wollen hier kurz das Verfahren von Brent vorstellen; wegen weiterer Einzelheiten verweisen wir auf die Originalarbeit [Bre 71].

Brents Algorithmus

Brent versucht, die Einschließungseigenschaft und sichere Konvergenz der Bisektion zu erhalten, aber schneller zu sein, wenn die Verhältnisse es erlauben. Dazu benutzt er inverse quadratische Interpolation, siehe Abschnitt 3.1.5. Für diese werden drei Vorgängerpunkte $a := x_{i-2}$, $b := x_{i-1}$ und $c := x_i$ benutzt, durch die das quadratische Interpolationspolynom als Funktion von y gelegt wird. Dann wird dessen Wert bei $y = 0$ als neuer Wert $x := x_{i+1}$ genommen. Das ergibt folgenden Algorithmus:

$$\begin{aligned} \text{Mit } R &:= \frac{f(c)}{f(b)}, & S &:= \frac{f(c)}{f(a)}, & T &:= \frac{f(a)}{f(b)} \quad \text{und} \\ P &:= S\{T(R-T)(b-c) - (1-R)(c-a)\}, \\ Q &:= (T-1)(R-1)(S-1) \\ \text{wird } x &:= c + \frac{P}{Q}. \end{aligned} \tag{4.35}$$

Die Übereinstimmung dieses Algorithmus mit der inversen Interpolation für drei Wertepaare wollen wir nicht zeigen, da er die Interpolationsmethode von Neville, siehe etwa [Sch 97], benutzt, die wir nicht behandelt haben. Es lässt sich aber leicht nachrechnen, dass dieser Algorithmus für das Beispiel 3.3 denselben Wert $x = -1/3$ liefert.

In gutartigen Fällen ist c eine Näherung der gesuchten Nullstelle, die durch die Korrektur P/Q verbessert wird. Dann ist die Konvergenz des Verfahrens superlinear. Jetzt wird der neue Wert x mit den alten Werten a , b und c so angeordnet, dass drei neue Werte für a , b und c entstehen, die die Einschließungseigenschaft erfüllen.

Wenn Division durch null auftritt, oder wenn der Korrekturterm P/Q zu einem Wert außerhalb des alten Intervalls führt, so wird statt des dargestellten Schritts ein Schritt des Sekantenverfahrens durchgeführt. Liefert dieses kein Einschließungsintervall, so wird stattdessen das Bisektionsverfahren durchgeführt. Auf diese Weise wird sichere Konvergenz erreicht, die in den allermeisten Fällen wesentlich schneller als Bisektion oder Sekantenverfahren und nur etwas langsamer als das Newton-Verfahren ist, wie wir am folgenden Beispiel sehen werden.

Beispiel 4.7. Die kleinste positive Lösung der Gleichung

$$f(x) = \cos(x) \cosh(x) + 1 = 0$$

soll jetzt mit der Methode von Brent bestimmt werden. Wir nehmen wieder das Startintervall aus Beispiel 4.3 und seinen Mittelpunkt als dritten Wert c . Im ersten Schritt liegt die Brent-Näherung $x \doteq 3.25$ außerhalb des Startintervalls, und es wird auf das Sekantenverfahren für das Halb-Intervall mit dem Vorzeichenwechsel $[1.5, 3]$ gewechselt. Es liefert einen neuen Näherungswert $x \doteq 1.67$ innerhalb dieses Intervalls. Damit konvergiert jetzt die Iterationsfolge der Methode von Brent und die Iteration wird nach weiteren fünf Iterationsschritten auf Grund eines betragsmäßig hinreichend kleinen Funktionswertes $|f(x^{(6)})| \doteq 3 \cdot 10^{-14}$ abgebrochen. Insgesamt werden also neun Funktionsauswertungen benötigt, vgl. Tab. 4.5. \triangle

Das Verfahren von Brent benötigt ein Einschließungsintervall. Für die Bestimmung eines solchen gibt es auch Verfahren, die man *Finde-Routinen* nennt. Erst mit ihrer Hilfe wird

Tab. 4.5 Methode von Brent.

k	a	b	c	$x^{(k+1)}$
0	0.0	3.0	1.5	3.25 (Brent versagt)
0	1.5	3.0	—	1.67 (Sekante)
1	1.5	3.0	1.67265867	1.93012033
2	1.67265867	3.0	1.93012033	1.86892123
3	1.67265867	1.93012033	1.86892123	1.87517286
4	1.86892123	1.93012033	1.87517286	1.87510405
5	1.86892123	1.87517286	1.87510405	1.87510407

die gesamte Methode zu einer wirklichen Black-box-Routine. Auf ihre Schilderung wollen wir verzichten; wir verweisen hierzu auf [Swi 78].

Verfahrensvergleich

Beispiele können nichts beweisen. Sie stellen nur mathematische Experimente dar, die Erfahrungen im Umgang mit numerischen Verfahren belegen oder widerlegen. In Tab. 4.6 wird deshalb ohne jeden Anspruch auf Allgemeingültigkeit die Anwendung der verschiedenen Verfahren auf das Beispiel $f(x) := \cos(x) \cosh(x) + 1 = 0$ noch einmal zusammengefasst. Alle Verfahren liefern eine Lösung hoher Genauigkeit, ausgehend von dem relativ großen Startintervall $[0, 3]$ bzw. dessen Mittelpunkt. Sie benötigen dazu unterschiedlichen Aufwand. Der Aufwand wird in diesem Zusammenhang gemessen als Zahl der benötigten Funktionsauswertungen. Wir fügen dem Vergleich noch eine Fixpunktiteration an. Sie entspricht der Vorgehensweise von Beispiel 4.2. Als Fixpunktfunktion haben wir

$$F(x) := \pi - \arccos\left(\frac{1}{\cosh(x)}\right) \quad (4.36)$$

gewählt. Die Fixpunktiteration liefert eine linear konvergente Folge für jeden Startwert aus dem Intervall $[1.5, 3]$ mit einer Konvergenzrate $K = 0.425$.

Tab. 4.6 Aufwandsvergleich für die eindimensionalen Verfahren.

Verfahren	Start	Anz. Schritte	Anz. f -Auswertungen
Fixpunktiteration	$x^{(0)} = 2$	17	17
Bisektion	$[0, 3]$	32	34
Sekantenverfahren	$[1.5, 3]$	8	9
Sekantenverfahren	$[0, 3]$	15	16
Newton	$x^{(0)} = 1.5$	5	11
Brent	$[0, 1.5, 3]$	6	9

4.3 Gleichungen in mehreren Unbekannten

Gegeben sei ein System von n nichtlinearen Gleichungen in n Unbekannten, und gesucht seien seine Lösungen. Die Aufgabe lässt sich mit einer stetigen, nichtlinearen Funktion $\mathbf{f}(\mathbf{x})$ mit $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ allgemein so formulieren, dass Lösungen $\mathbf{x} \in \mathbb{R}^n$ der Gleichung

$$\boxed{\mathbf{f}(\mathbf{x}) = \mathbf{0}} \quad (4.37)$$

zu bestimmen sind. Um die Darstellung zu vereinfachen, werden wir die grundlegenden Ideen und Prinzipien an einem System von zwei nichtlinearen Gleichungen in zwei Unbekannten darlegen. Sie lassen sich in offenkundiger Weise verallgemeinern.

4.3.1 Fixpunktiteration und Konvergenz

Als Grundlage für die Diskussion des Konvergenzverhaltens der nachfolgend dargestellten Iterationsverfahren zur Lösung von (4.37) betrachten wir zwei nichtlineare Gleichungen in zwei Unbekannten in der *Fixpunktform*

$$x = F(x, y), \quad y = G(x, y). \quad (4.38)$$

Jedes Wertepaar (s, t) mit der Eigenschaft

$$s = F(s, t), \quad t = G(s, t) \quad (4.39)$$

heißt *Fixpunkt* der Abbildung $\mathbf{F} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, welche definiert ist durch

$$\mathbf{F}(\mathbf{x}) := \begin{bmatrix} F(x, y) \\ G(x, y) \end{bmatrix}, \quad \mathbf{x} := \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2. \quad (4.40)$$

Um den Banachschen Fixpunktsatz anwenden zu können, muss für die Abbildung $\mathbf{F} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ ein abgeschlossener Bereich $A \subset \mathbb{R}^2$ (beispielsweise ein Rechteck oder ein Kreis) angegeben werden können, für den mit einer beliebigen Vektornorm

$$\mathbf{F} : A \rightarrow A \text{ mit } \|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}^*)\| \leq L\|\mathbf{x} - \mathbf{x}^*\|, \quad L < 1; \quad \mathbf{x}, \mathbf{x}^* \in A \quad (4.41)$$

gilt. Unter der zusätzlichen Voraussetzung, dass die Funktionen $F(x, y)$ und $G(x, y)$ in A stetige partielle Ableitungen nach x und y besitzen, lassen sich hinreichende Kriterien für die Lipschitz-Bedingung (4.41) angeben. Die Jacobi-Matrix oder *Funktionalmatrix* der Abbildung $\mathbf{F} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ ist gegeben durch

$$\mathbf{J}(x, y) := \begin{bmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} \end{bmatrix}_{(x,y)}. \quad (4.42)$$

Falls für irgendeine Matrixnorm $\|\mathbf{J}(x, y)\| \leq L < 1$ für alle $\mathbf{x} \in A$ gilt und A konvex ist, dann ist für eine mit ihr verträglichen Vektornorm (4.41) erfüllt. Mit feineren Hilfsmitteln der linearen Algebra kann gezeigt werden, dass folgende Bedingung notwendig und hinreichend ist (vgl. Abschnitt 11.2.2). Es seien $\lambda_i(\mathbf{J})$ die Eigenwerte einer Matrix $\mathbf{J} \in \mathbb{R}^{n,n}$. Man bezeichnet mit

$$\varrho(\mathbf{J}) := \max_i |\lambda_i(\mathbf{J})| \quad (4.43)$$

den *Spektralradius* der Matrix \mathbf{J} . Dann ist $\mathbf{F} : A \rightarrow A$ genau dann kontrahierend, falls für alle $\mathbf{x} \in A$ der Spektralradius $\varrho(\mathbf{J}) < 1$ ist.

Es ist oft schwierig, den Bereich A für eine Abbildung \mathbf{F} konkret anzugeben, für den eine der genannten Bedingungen für die Anwendung des Fixpunktsatzes von Banach auf die Iteration $\mathbf{x}^{(k+1)} = \mathbf{F}(\mathbf{x}^{(k)})$, $k = 0, 1, 2, \dots$, erfüllt ist. Wir wollen annehmen, dass die Voraussetzungen des Banachschen Fixpunktsatzes erfüllt sind. Dann kann in Verallgemeinerung von Satz 4.5 gezeigt werden, dass die Fixpunktiteration linear konvergiert, die Konvergenzordnung also $p = 1$ ist mit der Konvergenzrate $K = \varrho(\mathbf{J})$. Die Folge $\mathbf{x}^{(k)}$ konvergiert deshalb *linear* gegen den Fixpunkt \mathbf{s} . Eine höhere Konvergenzordnung kann nur vorliegen, wenn die Funktionalmatrix $\mathbf{J}(s, t)$ im Fixpunkt gleich der Nullmatrix ist.

4.3.2 Das Verfahren von Newton

Zur iterativen Bestimmung einer Lösung (s, t) eines Systems von zwei nichtlinearen Gleichungen

$$\boxed{f(x, y) = 0, \quad g(x, y) = 0} \quad (4.44)$$

ist das *Verfahren von Newton* oder eine seiner Varianten oft recht geeignet. Die Funktionen $f(x, y)$ und $g(x, y)$ seien im Folgenden als mindestens einmal stetig differenzierbar vorausgesetzt. Ausgehend von einer geeigneten Startnäherung $\mathbf{x}^{(0)} = (x^{(0)}, y^{(0)})^T$ für eine gesuchte Lösung $\mathbf{s} = (s, t)^T$ von (4.44) arbeitet man im allgemeinen k -ten Schritt mit dem *Korrekturansatz*

$$s = x^{(k)} + \xi^{(k)}, \quad t = y^{(k)} + \eta^{(k)}, \quad (4.45)$$

mit welchem das gegebene nichtlineare System *linearisiert* wird.

$$\begin{aligned} f(s, t) &= f(x^{(k)} + \xi^{(k)}, y^{(k)} + \eta^{(k)}) \\ &\approx f(x^{(k)}, y^{(k)}) + \xi^{(k)} f_x(x^{(k)}, y^{(k)}) + \eta^{(k)} f_y(x^{(k)}, y^{(k)}) = 0 \\ g(s, t) &= g(x^{(k)} + \xi^{(k)}, y^{(k)} + \eta^{(k)}) \\ &\approx g(x^{(k)}, y^{(k)}) + \xi^{(k)} g_x(x^{(k)}, y^{(k)}) + \eta^{(k)} g_y(x^{(k)}, y^{(k)}) = 0 \end{aligned} \quad (4.46)$$

Auf diese Weise ist für die Korrekturen $\xi^{(k)}, \eta^{(k)}$ ein lineares Gleichungssystem entstanden. (4.46) lautet mit den Vektoren

$$\begin{aligned} \mathbf{x}^{(k)} &:= \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix}, \quad \boldsymbol{\xi}^{(k)} := \begin{bmatrix} \xi^{(k)} \\ \eta^{(k)} \end{bmatrix}, \quad \mathbf{f}^{(k)} := \begin{bmatrix} f(x^{(k)}, y^{(k)}) \\ g(x^{(k)}, y^{(k)}) \end{bmatrix} \\ \boxed{\Phi(x^{(k)}, y^{(k)}) \boldsymbol{\xi}^{(k)} = -\mathbf{f}^{(k)}}, \end{aligned} \quad (4.47)$$

wo wir im Unterschied zu (4.42) die Funktionalmatrix

$$\Phi(x, y) := \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix}_{(x, y)} \quad (4.48)$$

zugehörig zu den Funktionen $f(x, y)$ und $g(x, y)$ des Systems (4.44) eingeführt haben. Unter der Voraussetzung, dass die Jacobi-Matrix Φ (4.48) für die Näherung $\mathbf{x}^{(k)}$ regulär ist, besitzt das lineare Gleichungssystem (4.47) eine eindeutige Lösung $\boldsymbol{\xi}^{(k)}$. Dieser Korrekturvektor $\boldsymbol{\xi}^{(k)}$

liefert natürlich nicht die Lösung \mathbf{s} , sondern eine neue Näherung $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\xi}^{(k)}$, welche die formale Darstellung

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \Phi^{-1}(\mathbf{x}^{(k)}) \mathbf{f}^{(k)} =: \mathbf{F}(\mathbf{x}^{(k)}), \quad k = 0, 1, 2, \dots, \quad (4.49)$$

bekommt. (4.49) stellt eine Fixpunktiteration dar. Sie kann als direkte Verallgemeinerung der Iterationsvorschrift (4.24) angesehen werden. Sie darf aber nicht dazu verleiten, die Inverse von Φ zu berechnen, vielmehr soll die Korrektur $\boldsymbol{\xi}^{(k)}$ als Lösung des linearen Gleichungssystems (4.47) bestimmt werden, z.B. mit dem Gauß-Algorithmus, siehe Kapitel 2.

Im Fall eines Systems von zwei nichtlinearen Gleichungen (4.44) kann das lineare Gleichungssystem (4.48) explizit mit der Cramerschen Regel gelöst werden:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \xi^{(k)} = x^{(k)} + \frac{gf_y - fg_y}{f_x g_y - f_y g_x} =: F(x^{(k)}, y^{(k)}) \\ y^{(k+1)} &= y^{(k)} + \eta^{(k)} = y^{(k)} + \frac{fg_x - gf_x}{f_x g_y - f_y g_x} =: G(x^{(k)}, y^{(k)}) \end{aligned} \quad (4.50)$$

Satz 4.8. Die Funktionen $f(x, y)$ und $g(x, y)$ von (4.44) seien in einem Bereich A , welcher die Lösung \mathbf{s} im Innern enthält, dreimal stetig differenzierbar. Die Funktionalmatrix $\Phi(s, t)$ (4.48) sei regulär. Dann besitzt die Methode von Newton mindestens die Konvergenzordnung $p = 2$.

Beweis. Die Jacobi-Matrix der Fixpunktiteration (4.49) ist gegeben durch

$$\mathbf{J}(x, y) = \begin{bmatrix} 1 + \xi_x & \xi_y \\ \eta_x & 1 + \eta_y \end{bmatrix}_{(x, y)}. \quad (4.51)$$

In [Sch 97] wird gezeigt, dass $\mathbf{J}(s, t)$ im Lösungspunkt gleich der Nullmatrix ist. \square

Beispiel 4.8. Gesucht sei die Lösung (s, t) der beiden quadratischen Gleichungen

$$\begin{aligned} x^2 + y^2 + 0.6y &= 0.16 \\ x^2 - y^2 + x - 1.6y &= 0.14 \end{aligned} \quad (4.52)$$

für die $s > 0$ und $t > 0$ gilt. Im Verfahren von Newton sind somit $f(x, y) = x^2 + y^2 + 0.6y - 0.16$ und $g(x, y) = x^2 - y^2 + x - 1.6y - 0.14$, und die benötigten ersten partiellen Ableitungen sind

$$f_x = 2x, \quad f_y = 2y + 0.6, \quad g_x = 2x + 1, \quad g_y = -2y - 1.6.$$

Für die Startnäherungen $x^{(0)} = 0.6$, $y^{(0)} = 0.25$ lautet (4.47)

$$\begin{aligned} 1.2 \xi^{(0)} + 1.1 \eta^{(0)} &= -0.4125 \\ 2.2 \xi^{(0)} - 2.1 \eta^{(0)} &= -0.3575 \end{aligned} \quad (4.53)$$

mit den resultierenden Korrekturen $\xi^{(0)} \doteq -0.254960$, $\eta^{(0)} \doteq -0.096862$ und den neuen Näherungen $x^{(1)} \doteq 0.345040$, $y^{(1)} \doteq 0.153138$. Das Gleichungssystem für die neuen Korrekturen lautet mit gerundeten Zahlenwerten

$$\begin{aligned} 0.690081 \xi^{(1)} + 0.906275 \eta^{(1)} &= -0.0743867 \\ 1.690081 \xi^{(1)} - 1.906275 \eta^{(1)} &= -0.0556220. \end{aligned} \quad (4.54)$$

Der weitere Rechenablauf der Methode von Newton ist bei zehnstelliger Rechnung in Tab. 4.7 zusammengestellt. Die euklidische Norm des nachträglich berechneten Fehlers $\|\boldsymbol{x}^{(k)} - \boldsymbol{s}\|$ nimmt quadratisch gegen null ab ($p = 2$, $K \approx 1$). \triangle

Tab. 4.7 Methode von Newton für ein System nichtlinearer Gleichungen.

k	$x^{(k)}$	$y^{(k)}$	$\xi^{(k)}$	$\eta^{(k)}$	$\ \boldsymbol{x}^{(k)} - \boldsymbol{s}\ $
0	0.6000000000	0.2500000000	$-2.54960 \cdot 10^{-1}$	$-9.68623 \cdot 10^{-2}$	$3.531 \cdot 10^{-1}$
1	0.3450404858	0.1531376518	$-6.75094 \cdot 10^{-2}$	$-3.06747 \cdot 10^{-2}$	$8.050 \cdot 10^{-2}$
2	0.2775310555	0.1224629827	$-5.64594 \cdot 10^{-3}$	$-2.79860 \cdot 10^{-3}$	$6.347 \cdot 10^{-3}$
3	0.2718851108	0.1196643843	$-4.06023 \cdot 10^{-5}$	$-2.10056 \cdot 10^{-5}$	$4.572 \cdot 10^{-5}$
4	0.2718445085	0.1196433787	$-2.1579 \cdot 10^{-9}$	$-1.1043 \cdot 10^{-9}$	$2.460 \cdot 10^{-9}$
5	0.2718445063	0.1196433776			

Die Berechnung der Funktionalmatrix $\Phi(\boldsymbol{x}^{(k)})$ kann in komplizierteren Fällen aufwändig sein, da für ein System von n nichtlinearen Gleichungen n^2 partielle Ableitungen auszuwerten sind. Aus diesem Grund existieren eine Reihe von vereinfachenden Varianten, die skizziert werden sollen.

Das vereinfachte Newton-Verfahren

In diesem Verfahren, das dem eindimensionalen Verfahren (4.32) entspricht, wird die Funktionalmatrix $\Phi(\boldsymbol{x}^{(0)})$ nur einmal für einen möglichst guten Startvektor $\boldsymbol{x}^{(0)}$ berechnet, und die Korrekturen $\xi^{(k)}$ werden an Stelle von (4.47) aus dem Gleichungssystem

$$\boxed{\Phi(\boldsymbol{x}^{(0)}) \boldsymbol{\xi}^{(k)} = -\boldsymbol{f}^{(k)}, \quad k = 0, 1, 2, \dots}, \quad (4.55)$$

mit der gleich bleibenden Matrix $\Phi(\boldsymbol{x}^{(0)})$ bestimmt. Dies verringert den Aufwand auch dadurch, dass bei Anwendung des Gauß-Algorithmus die *LR*-Zerlegung der Matrix nur einmal erfolgen muss und für alle Iterationsschritte nur die Vorwärts- und Rücksubstitution auszuführen sind. Die vereinfachte Methode (4.55) hat die Konvergenzordnung $p = 1$, da die Jacobi-Matrix $\boldsymbol{J}(s, t)$ (4.51) nicht die Nullmatrix ist. Eine zum Beweis von Satz 4.8 analoge Analyse zeigt, dass für eine gute Startnäherung $\boldsymbol{x}^{(0)}$ die Matrixelemente von $\boldsymbol{J}(s, t)$ betragsmäßig klein sind, so dass wegen einer entsprechend kleinen Lipschitz-Konstante L die Folge $\boldsymbol{x}^{(k)}$ doch relativ rasch konvergiert.

Beispiel 4.9. Das Gleichungssystem (4.52) wird mit dem vereinfachten Verfahren von Newton mit den guten Näherungen $x^{(0)} = 0.3$ und $y^{(0)} = 0.1$ behandelt. Mit der Funktionalmatrix

$$\Phi(x^{(0)}, y^{(0)}) = \begin{pmatrix} 0.6 & 0.8 \\ 1.6 & -1.8 \end{pmatrix} \quad (4.56)$$

konvergiert die Folge der Näherungen $\boldsymbol{x}^{(k)}$ linear mit einer Konvergenzrate $K \approx 0.06$, so dass die euklidische Norm des Fehlers $\boldsymbol{\varepsilon}^{(k)} := \boldsymbol{x}^{(k)} - \boldsymbol{s}$ in jedem Schritt etwa auf den fünfzehnten Teil

Tab. 4.8 Vereinfachte Methode von Newton für ein System.

k	$x^{(k)}$	$y^{(y)}$	$\xi^{(k)}$	$\eta^{(k)}$	$\ \varepsilon^{(k)}\ $
0	0.3000000000	0.1000000000	$-2.71186 \cdot 10^{-2}$	$2.03390 \cdot 10^{-2}$	$3.433 \cdot 10^{-2}$
1	0.2728813559	0.1203389831	$-9.85495 \cdot 10^{-4}$	$-6.97248 \cdot 10^{-4}$	$1.249 \cdot 10^{-3}$
2	0.2718958608	0.1196417355	$-4.81441 \cdot 10^{-5}$	$2.92648 \cdot 10^{-6}$	$5.138 \cdot 10^{-5}$
3	0.2718477167	0.1196446620	$-3.03256 \cdot 10^{-6}$	$-1.25483 \cdot 10^{-6}$	$3.458 \cdot 10^{-6}$
4	0.2718446841	0.1196434072	$-1.67246 \cdot 10^{-7}$	$-2.64407 \cdot 10^{-8}$	$1.802 \cdot 10^{-7}$
5	0.2718445169	0.1196433808	$-9.9322 \cdot 10^{-9}$	$-3.0508 \cdot 10^{-9}$	$1.107 \cdot 10^{-8}$
6	0.2718445070	0.1196433777	$-6.1017 \cdot 10^{-10}$	$-4.2373 \cdot 10^{-11}$	$7.07 \cdot 10^{-10}$
7	0.2718445064	0.1196433777			

verkleinert wird. Deshalb sind nur sieben Iterationsschritte nötig, um die Lösung mit zehnstelliger Genauigkeit zu erhalten. Die Ergebnisse sind in Tab. 4.8 zusammengefasst.

Benutzt man im vereinfachten Verfahren von Newton die gleichen Startwerte wie im Beispiel 4.8, so stellt sich eine langsame lineare Konvergenz ein mit einer Konvergenzrate $K \approx 0.45$. Es sind 27 Iterationsschritte erforderlich, um die Lösung mit zehnstelliger Genauigkeit zu berechnen. \triangle

Einzelschritt- und SOR-Verfahren

Eine andere Variante, in welcher die Berechnung der Funktionalmatrix $\Phi(x^{(k)})$ teilweise vermieden wird, besteht darin, die Lösung eines Systems von n nichtlinearen Gleichungen in n Unbekannten auf die sukzessive Lösung von nichtlinearen Gleichungen in einer Unbekannten zurückzuführen. Wir betrachten das System von n Gleichungen

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad i = 1, 2, \dots, n. \quad (4.57)$$

Wir setzen voraus, dass sie so angeordnet seien, dass

$$\frac{\partial f_i(x_1, x_2, \dots, x_n)}{\partial x_i} \neq 0 \quad \text{für } i = 1, 2, \dots, n \quad (4.58)$$

gilt, was nur bedeutet, dass die i -te Gleichung die i -te Unbekannte x_i enthält.

Es sei $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})^T$ der k -te Näherungsvektor. In Anlehnung an das *Einzelschrittverfahren* zur iterativen Lösung von linearen Gleichungssystemen (vgl. Abschnitt 11.2) soll die i -te Komponente des $(k+1)$ -ten Näherungsvektors als Lösung der i -ten Gleichung wie folgt bestimmt werden.

$$f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k+1)}, x_{i+1}^{(k)}, \dots, x_n^{(k)}) = 0, \quad i = 1, 2, \dots, n. \quad (4.59)$$

Für die ersten $(i-1)$ Variablen werden in (4.59) die bereits bekannten Komponenten von $\mathbf{x}^{(k+1)}$ eingesetzt, für die letzten $(n-i)$ Variablen die entsprechenden Komponenten von $\mathbf{x}^{(k)}$. Für jeden Index i stellt somit (4.59) eine Gleichung für die Unbekannte $x_i^{(k+1)}$ dar, welche mit dem Verfahren von Newton bestimmt werden kann. Zur Durchführung dieser *inneren Iteration* werden nur die partiellen Ableitungen $\partial f_i / \partial x_i$ benötigt. Insgesamt sind dies nur die n Diagonalelemente der Funktionalmatrix Φ , aber natürlich mit wechselnden Argumenten. Der Wert $x_i^{(k)}$ stellt einen geeigneten Startwert für die Iteration dar.

Das *nichtlineare Einzelschrittverfahren* für ein System (4.57) zu gegebenem Startvektor $\mathbf{x}^{(0)}$ ist in (4.60) algorithmisch beschrieben.

<p>Für $k = 0, 1, 2, \dots$: $s = 0$ für $i = 1, 2, \dots, n$:</p> $\xi_i = x_i^{(k)}$ $t := \frac{\partial f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, \xi_i, x_{i+1}^{(k)}, \dots, x_n^{(k)})}{\partial x_i}$ <p style="margin-left: 40px;">A</p> $\Delta \xi_i = f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, \xi_i, x_{i+1}^{(k)}, \dots, x_n^{(k)})/t$ $\xi_i = \xi_i - \Delta \xi_i$ <p style="margin-left: 40px;">falls $\Delta \xi_i > \text{tol}_1$: gehe zu A</p> $s = s + \xi_i - x_i^{(k)} ; x_i^{(k+1)} = \xi_i$ <p style="margin-left: 40px;">falls $s < \text{tol}_2$: STOP</p>	<p>(4.60)</p>
--	---------------

Das nichtlineare Einzelschrittverfahren auf der Basis des Verfahrens von Newton zur Lösung der i -ten Gleichung (4.59) besitzt mehrere Modifikationsmöglichkeiten. So braucht die i -te Gleichung nicht exakt nach $x_i^{(k+1)}$ aufgelöst zu werden, da dieser Wert ohnehin nur eine neue Näherung darstellt. Deshalb verzichtet man auf die innere Iteration und führt nur einen einzigen Schritt der Newton-Korrektur zur Bestimmung von $x_i^{(k+1)}$ mit Hilfe der i -ten Gleichung aus. So entsteht das *Newtonsche Einzelschrittverfahren* mit

<p>Für $i = 1, 2, \dots, n$:</p> $t := \frac{\partial f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k)}, x_{i+1}^{(k)}, \dots, x_n^{(k)})}{\partial x_i}$ $x_i^{(k+1)} = x_i^{(k)} - f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k)}, x_{i+1}^{(k)}, \dots, x_n^{(k)})/t$	<p>(4.61)</p>
---	---------------

Die lineare Konvergenz des Newtonschen Einzelschrittverfahrens lässt sich in der Regel dadurch verbessern, dass in Analogie zur *Methode der sukzessiven Überrelaxation* (vgl. Abschnitt 11.2) die Korrektur der i -ten Komponente $x_i^{(k)}$ mit einem konstanten, geeignet gewählten *Relaxationsfaktor* $\omega \in (0, 2)$ multipliziert wird. Das so entstehende *SOR-Newton-Verfahren* lautet somit

<p>Für $i = 1, 2, \dots, n$:</p> $t := \frac{\partial f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k)}, \dots, x_n^{(k)})}{\partial x_i}$ $x_i^{(k+1)} = x_i^{(k)} - \omega f_i(x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}, x_i^{(k)}, \dots, x_n^{(k)})/t$	<p>(4.62)</p>
--	---------------

Das SOR-Newton-Verfahren wird sehr erfolgreich zur Lösung von großen Systemen von nichtlinearen Gleichungen eingesetzt, welche die Eigenschaft besitzen, dass die i -te Gleichung nur wenige der Unbekannten miteinander verknüpft (schwach gekoppelte Systeme).

Diese Situation tritt bei der numerischen Behandlung von nichtlinearen Randwertaufgaben mit Differenzenmethoden oder mit der Methode der finiten Elemente auf. In diesen wichtigen Spezialfällen existieren Aussagen über die günstige Wahl des Relaxationsfaktors ω zur Erzielung optimaler Konvergenz [Ort 00].

Powell beschreibt in [Rab 88] eine Methode zur Lösung eines nichtlinearen Gleichungssystems, die ein typisches Beispiel für ein auf Softwarezwecke zugeschnittenes Black-box-Verfahren darstellt. Es werden nämlich wie beim Algorithmus von Brent (vgl. Abschnitt 4.2.4) verschiedene Verfahren so miteinander kombiniert, dass für möglichst alle Fälle Konvergenz gezeigt werden kann. Dabei wird zusätzlich der Aufwand bei den aufwändigsten Teilen des Algorithmus minimiert. Wir wollen die Schritte, die zum Entstehen eines solchen Verfahrens führen, hier nachzeichnen, verweisen aber zum Nachlesen aller Details – von den mathematischen Beweisen bis zur Genauigkeitssteuerung im Flußdiagramm – auf die beiden Beiträge von Powell in [Rab 88].

Ausgangspunkt ist das Gleichungssystem (4.47) für das Newton-Verfahren

$$\begin{aligned}\Phi(x^{(k)}, y^{(k)}) \xi^{(k)} &= -\mathbf{f}^{(k)} \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \xi^{(k)}.\end{aligned}$$

Garantierte Konvergenz kann mit dem Newton-Verfahren aber nicht erzielt werden. Eine Modifizierung des Newton-Verfahrens besteht nun darin, dass man zwar in Richtung $\xi^{(k)}$ verbessert, aber die Länge der Verbesserung steuert:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda \xi^{(k)}. \quad (4.63)$$

Die Werte $\lambda := \lambda_k$ werden in jedem Schritt so festgelegt, dass

$$F(\mathbf{x}^{(k+1)}) < F(\mathbf{x}^{(k)}) \quad (4.64)$$

mit der Funktion

$$F(\mathbf{x}) = \sum_{i=1}^n \left(f_i(\mathbf{x}^{(k)}) \right)^2. \quad (4.65)$$

F ist genau dann gleich null, wenn alle f_i gleich null sind, und jedes lokale Minimum von F mit dem Wert null ist auch globales Minimum. Wenn die Jacobi-Matrix \mathbf{J} nicht singulär ist, erhält man mit diesem Verfahren eine monoton fallende Folge von Werten $F(\mathbf{x}^{(k)})$ mit dem Grenzwert null. Ist aber \mathbf{J} in der Nullstelle singulär und besitzt F ein lokales Minimum mit einem Wert ungleich null, dann kann auch dieses Verfahren gegen einen falschen Grenzwert konvergieren; ein Beispiel findet man in [Rab 88].

Ein anderer Ansatz zur Modifikation des Newton-Verfahrens ist der folgende: Die Minimierung der Funktion $F(\mathbf{x})$ wird auf die Aufgabe zurückgeführt, eine Folge von Lösungen kleinster Quadrate (kurz: lss) der linearen Gleichungssysteme $\Phi(x^{(k)}, y^{(k)}) \xi^{(k)} = -\mathbf{f}^{(k)}$ zu bestimmen. Dies wollen wir für das Folgende abgekürzt schreiben als $\Phi \xi = -\mathbf{f}$. Die lss ist aber Lösung des Normalgleichungssystems (vgl. Kapitel 6)

$$\Phi^T \Phi \xi = -\Phi^T \mathbf{f}. \quad (4.66)$$

Dieses wird mit einem Parameter $\mu := \mu_k$ versehen¹:

$$\begin{aligned} (\Phi^T \Phi + \mu I) \boldsymbol{\eta} &= -\Phi^T \mathbf{f}, \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \boldsymbol{\eta}. \end{aligned} \quad (4.67)$$

$\boldsymbol{\eta}$ ist von μ abhängig: $\boldsymbol{\eta} = \boldsymbol{\eta}(\mu)$, und es gilt

$$\boldsymbol{\eta}(0) = \boldsymbol{\xi}.$$

Läßt man $\mu \rightarrow \infty$ wachsen, so gilt

$$\mu \boldsymbol{\eta} \rightarrow -\Phi^T \mathbf{f}, \quad (4.68)$$

d.h. das Verfahren wird zur klassischen Methode des steilsten Abstiegs mit einem gegen null gehenden Längenfaktor $1/\mu$. Mit Hilfe dieser Überlegungen kann nun gezeigt werden, dass man immer einen Wert μ bestimmen kann mit

$$F(\mathbf{x}^{(k+1)}) < F(\mathbf{x}^{(k)}). \quad (4.69)$$

Das Verfahren kann nur versagen, wenn die Vektorfunktion \mathbf{f} keine beschränkten ersten Ableitungen besitzt, oder in Extremfällen durch den Einfluß von Rundungsfehlern. Für die Implementierung in einem Softwaresystem müssen ohnehin in ein Verfahren Abfragen (z.B. gegen unendliche Schleifen oder ein Übermaß an Funktionsauswertungen) eingebaut werden, die den Algorithmus in gewissen Extremfällen abbrechen lassen, obwohl theoretisch Konvergenz vorliegt. Hier sollte für den Fall, dass die Funktionswerte $F(\mathbf{x}^{(k)})$ wesentlich größer sind als die Werte der Ableitungen von F , eine Abfrage mit einer Konstanten M vorgesehen werden:

$$\begin{aligned} F(\mathbf{x}^{(k)}) &> M \| \mathbf{g}^{(k)} \| \\ \text{mit} \quad g_j^{(k)} &:= \left. \frac{\partial}{\partial x_j} F(\mathbf{x}) \right|_{\mathbf{x}=\mathbf{x}^{(k)}}. \end{aligned} \quad (4.70)$$

Wird bei genügend großem M diese Abfrage bejaht, so ist die Wahrscheinlichkeit groß, dass das Verfahren nicht gegen eine Nullstelle von \mathbf{f} , sondern gegen ein lokales Minimum von F konvergiert. Bei der Konstruktion von Verfahren zur Lösung nichtlinearer Gleichungssysteme muß man auch an den Fall denken, dass es gar keine Lösung gibt. Auch für diesen Fall sind eine solche Abfrage und das Festsetzen einer Maximalzahl von Funktionsauswertungen wichtig.

Der vollständige Algorithmus von Powell enthält noch weitere Anteile wie z.B. die Unterscheidung zwischen funktionaler und diskretisierter Jacobi-Matrix. Auf ihre Darstellung wollen wir ebenso verzichten wie auf ein Beispiel. Es müsste sehr schlecht konditioniert sein, um den Unterschied zum normalen oder vereinfachten Newton-Verfahren zu verdeutlichen, so wie etwa das ‘Trog-Beispiel’, das in Abschnitt 4.4.2 von [Köc 94] zu finden ist.

¹ Diese Idee verwendet man auch bei Regularisierungsmethoden für inkorrekt gestellte Probleme oder bei der Lösung schlecht konditionierter Gleichungssysteme. Marquardt wendet eine fast identische Idee bei der Entwicklung einer effizienten Methode zur Lösung nichtlinearer Ausgleichsprobleme an, siehe Abschnitt 6.4.2.

4.4 Nullstellen von Polynomen

4.4.1 Reelle Nullstellen: Das Verfahren von Newton-Maehly

Zur Berechnung der Nullstellen eines Polynoms n -ten Grades

$$P_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-1}x + a_n, \quad a_0 \neq 0 \quad (4.71)$$

stellt die Methode von Newton ein zweckmäßiges, effizientes und einfach zu programmierendes Verfahren dar. Wir betrachten hauptsächlich den Fall von Polynomen mit reellen Koeffizienten a_j . Die meisten Überlegungen bleiben ohne Einschränkung auch für Polynome mit komplexen Koeffizienten gültig.

Als Grundlage für die Anwendung der Methode von Newton wenden wir uns der Aufgabe zu, den Wert des Polynoms $P_n(x)$ und denjenigen seiner ersten Ableitung für einen gegebenen Wert des Argumentes x effizient und auf einfache Art zu berechnen. Der zu entwickelnde Algorithmus beruht auf der Division mit Rest eines Polynoms durch einen linearen Faktor

$$P_n(x) = (x - p)P_{n-1}(x) + R, \quad p \text{ gegeben.} \quad (4.72)$$

Das Quotientenpolynom $P_{n-1}(x)$ habe die Darstellung

$$P_{n-1}(x) = b_0x^{n-1} + b_1x^{n-2} + b_2x^{n-3} + \dots + b_{n-2}x + b_{n-1}, \quad (4.73)$$

und der Rest wird aus bald ersichtlichen Gründen $R = b_n$ gesetzt. Dann folgt aus (4.72)

$$\begin{aligned} & a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_{n-2}x^2 + a_{n-1}x + a_n \\ &= (x - p)(b_0x^{n-1} + b_1x^{n-2} + b_2x^{n-3} + \dots + b_{n-2}x + b_{n-1}) + b_n \end{aligned}$$

durch Koeffizientenvergleich

$$a_0 = b_0, a_1 = b_1 - pb_0, \dots, a_{n-1} = b_{n-1} - pb_{n-2}, a_n = b_n - pb_{n-1}.$$

Daraus resultiert der folgende Algorithmus zur rekursiven Berechnung der Koeffizienten b_j

$$b_0 = a_0; \quad b_j = a_j + pb_{j-1}, \quad j = 1, 2, \dots, n; \quad R = b_n. \quad (4.74)$$

Die praktische Bedeutung des Divisionsalgorithmus (4.74) wird deutlich durch

Lemma 4.9. *Der Wert des Polynoms $P_n(x)$ für $x = p$ ist gleich dem Rest R bei der Division von $P_n(x)$ durch $(x - p)$.*

Beweis. Setzen wir in (4.72) $x = p$ ein, so folgt in der Tat $P_n(p) = R$. □

Der einfach durchführbare Algorithmus (4.74) liefert somit den Wert des Polynoms $P_n(x)$ für einen gegebenen Wert x mit einem Rechenaufwand von je n Multiplikationen und Additionen. Um auch für die Berechnung der ersten Ableitung eine ähnliche Rechenvorschrift zu erhalten, differenzieren wir (4.72) nach x und erhalten die Beziehung

$$P'_n(x) = P_{n-1}(x) + (x - p)P'_{n-1}(x). \quad (4.75)$$

Für $x = p$ folgt daraus

$$P'_n(p) = P_{n-1}(p). \quad (4.76)$$

Der Wert der ersten Ableitung von $P_n(x)$ für einen bestimmten Argumentwert $x = p$ ist gleich dem Wert des Quotientenpolynoms $P_{n-1}(x)$ für $x = p$. Nach Satz 4.9 kann sein Wert mit Hilfe des Divisionsalgorithmus berechnet werden. Setzen wir

$$\begin{aligned} P_{n-1}(x) &= (x - p)P_{n-2}(x) + R_1 \quad \text{und} \\ P_{n-2}(x) &= c_0x^{n-2} + c_1x^{n-3} + \dots + c_{n-3}x + c_{n-2}, \quad c_{n-1} = R_1, \end{aligned}$$

dann sind die Koeffizienten c_j rekursiv gegeben durch

$$c_0 = b_0; \quad c_j = b_j + pc_{j-1}, \quad j = 1, 2, \dots, n-1; \quad R_1 = c_{n-1}. \quad (4.77)$$

Die erste Ableitung $P'_n(p)$ ist wegen (4.76) und (4.77) mit weiteren $(n-1)$ Multiplikationen berechenbar.

Obwohl im Zusammenhang mit der Methode von Newton die höheren Ableitungen nicht benötigt werden, sei doch darauf hingewiesen, dass sie durch die analoge Fortsetzung des Divisionsalgorithmus berechnet werden können.

Die im Divisionsalgorithmus anfallenden Größen werden zweckmäßigerweise im *Horner-Schema* zusammengestellt, das sich insbesondere für die Handrechnung gut eignet. Das Horner-Schema ist in (4.78) im Fall $n = 6$ für die ersten beiden Divisionsschritte mit ergänzenden Erklärungen wiedergegeben.

$$\begin{array}{l} P_6(x) : \quad a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \\ p : \quad pb_0 \quad pb_1 \quad pb_2 \quad pb_3 \quad pb_4 \quad pb_5 \\ \hline P_5(x) : \quad b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad | \quad \underline{\underline{b_6 = P_6(p)}} \\ p : \quad pc_0 \quad pc_1 \quad pc_2 \quad pc_3 \quad pc_4 \\ \hline P_4(x) : \quad c_0 \quad c_1 \quad c_2 \quad c_3 \quad c_4 \quad | \quad \underline{\underline{c_5 = P'_6(p)}} \end{array} \quad (4.78)$$

Jede weitere Zeile verkürzt sich um einen Wert entsprechend der Abnahme des Grades des Quotientenpolynoms.

Ist man nur am Wert eines Polynoms n -ten Grades ($n > 2$) und seiner ersten Ableitung für gegebenes p interessiert, kann dies in einem Rechner ohne indizierte Variablen b und c realisiert werden. Es genügt dazu die Feststellung, dass der Wert von b_{j-1} nicht mehr benötigt wird, sobald zuerst c_{j-1} und dann b_j berechnet sind. Der entsprechende Algorithmus lautet deshalb

$$\begin{array}{l} b = a_0; \quad c = b; \quad b = a_1 + p \cdot b \\ \text{für } j = 2, 3, \dots, n : \\ \quad c = b + p \cdot c; \quad b = a_j + p \cdot b \\ P_n(p) = b; \quad P'_n(p) = c \end{array} \quad (4.79)$$

Beispiel 4.10. Für das Polynom fünften Grades

$$P_5(x) = x^5 - 5x^3 + 4x + 1$$

soll eine Nullstelle berechnet werden, für welche die Näherung $x^{(0)} = 2$ bekannt ist. Das Horner-Schema lautet

$$\begin{array}{cccccc} 1 & 0 & -5 & 0 & 4 & 1 \\ 2 : & 2 & 4 & -2 & -4 & 0 \\ \hline 1 & 2 & -1 & -2 & 0 & | \underline{\underline{1 = P_5(2)}} \\ 2 : & 2 & 8 & 14 & 24 \\ \hline 1 & 4 & 7 & 12 & | \underline{\underline{24 = P'_5(2)}} \end{array}$$

Mit diesen Werten erhält man mit dem Verfahren von Newton $x^{(1)} = 2 - \frac{1}{24} \doteq 1.95833$. Das nächste Horner-Schema für $x^{(1)}$ lautet bei Rechnung mit sechs wesentlichen Dezimalstellen

$$\begin{array}{cccccc} 1 & 0 & -5 & 0 & 4 & 1 \\ 1.95833 : & 1.95833 & 3.83506 & -2.28134 & -4.46762 & -0.915754 \\ \hline 1 & 1.95833 & -1.16494 & -2.28134 & -0.467620 & | \underline{\underline{0.084246}} \\ 1.95833 : & 1.95833 & 7.67011 & 12.7393 & 20.4802 \\ \hline 1 & 3.91666 & 6.50517 & 10.4580 & | \underline{\underline{20.0126}} \end{array}$$

Daraus ergibt sich der zweite Näherungswert $x^{(2)} \doteq 1.95833 - 0.00420965 \doteq 1.95412$. Ein weiterer Iterationsschritt ergibt $x^{(3)} \doteq 1.95408$, und dies ist der auf sechs Stellen gerundete Wert einer ersten Nullstelle. \triangle

Für eine Nullstelle z_1 des Polynoms $P_n(x)$ ist $(x - z_1)$ ein Teiler von $P_n(x)$, so dass diese Nullstelle abgespalten werden kann. Division von $P_n(x)$ durch den Linearfaktor $(x - z_1)$ liefert das Quotientenpolynom $P_{n-1}(x)$, dessen Nullstellen die restlichen Nullstellen von $P_n(x)$ sind. Die Koeffizienten von $P_{n-1}(x)$ berechnen sich nach dem Divisionsalgorithmus (4.74), wobei der resultierende Rest R verschwinden muss und somit als Rechenkontrolle dienen kann. Man fährt mit $P_{n-1}(x)$ analog weiter, dividiert einen weiteren Linearfaktor $(x - z_2)$ ab und reduziert auf diese Weise sukzessive den Grad des Polynoms, für welches eine nächste Nullstelle zu berechnen ist. Diese Vorgehensweise nennt man Deflation von Nullstellen.

Die sukzessive *Deflation* von Nullstellen kann sich auf die nachfolgend zu bestimmenden Nullstellen der Polynome von kleinerem Grad aus zwei Gründen ungünstig auswirken. Einerseits ist die berechnete Nullstelle im allgemeinen nur eine Näherung der exakten Nullstelle, und andererseits entstehen bei der Berechnung der Koeffizienten b_j von $P_{n-1}(x)$ unvermeidliche Rundungsfehler. Deshalb berechnet man eine Nullstelle z_2^* eines gefälschten Polynoms $P_{n-1}^*(x)$, die bei entsprechender Empfindlichkeit auf kleine Änderungen in den Koeffizienten eine große Abweichung gegenüber der exakten Nullstelle z_2 von $P_n(x)$ aufweisen kann.

Um die erwähnten Schwierigkeiten zu vermeiden, existiert nach einer Idee von Maehly [Mae 54] eine einfache Modifikation des Verfahrens von Newton. Die bekannten Nullstellen von $P_n(x)$ werden *implizit* abgespalten, so dass mit den unveränderten Koeffizienten von $P_n(x)$ gearbeitet werden kann. Wir bezeichnen mit z_1, z_2, \dots, z_n die Nullstellen von $P_n(x)$. Dann gelten

$$P_n(x) = \sum_{j=0}^n a_j x^{n-j} = a_0 \prod_{j=1}^n (x - z_j), \quad a_0 \neq 0, \quad (4.80)$$

$$P'_n(x) = a_0 \sum_{i=1}^n \prod_{\substack{j=1 \\ j \neq i}}^n (x - z_j), \quad \frac{P'_n(x)}{P_n(x)} = \sum_{i=1}^n \frac{1}{x - z_i}. \quad (4.81)$$

Die Korrektur eines Näherungswertes $x^{(k)}$ im Verfahren von Newton ist gegeben durch den Kehrwert der formalen Partialbruchzerlegung (4.81) des Quotienten $P'_n(x^{(k)})/P_n(x^{(k)})$. Nun nehmen wir an, es seien bereits die m Nullstellen z_1, z_2, \dots, z_m , ($1 \leq m < n$) berechnet worden, so dass nach ihrer Deflation das Polynom

$$P_{n-m}(x) := P_n(x) \left/ \prod_{i=1}^m (x - z_i) \right. = a_0 \prod_{j=m+1}^n (x - z_j) \quad (4.82)$$

im Prinzip für die Bestimmung der nächsten Nullstelle zu verwenden ist. Für $P_{n-m}(x)$ gilt wegen (4.81)

$$\frac{P'_{n-m}(x)}{P_{n-m}(x)} = \sum_{j=m+1}^n \frac{1}{x - z_j} = \frac{P'_n(x)}{P_n(x)} - \sum_{i=1}^m \frac{1}{x - z_i}, \quad (4.83)$$

und folglich lautet die modifizierte Rechenvorschrift des Verfahrens von Newton mit *impliziter Deflation* der bereits berechneten Nullstellen z_1, z_2, \dots, z_m

$$x^{(k+1)} = x^{(k)} - \frac{1}{\frac{P'_n(x^{(k)})}{P_n(x^{(k)})} - \sum_{i=1}^m \frac{1}{(x^{(k)} - z_i)}}, \quad k = 0, 1, 2, \dots \quad (4.84)$$

Der Rechenaufwand für einen Iterationsschritt (4.84) beträgt für die $(m + 1)$ -te Nullstelle $Z_N = 2n + m + 1$ multiplikative Operationen. Er nimmt mit jeder berechneten Nullstelle zu.

Tab. 4.9 Methode von Newton mit impliziter Deflation.

k	$x^{(k)}$	$P_5(x^{(k)})$	$P'_5(x^{(k)})$	$(x^{(k)} - z_1)^{-1}$	$\Delta x^{(k)} := x^{(k+1)} - x^{(k)}$
0	2.00000	1.00000	24.000	21.7770	$-4.49843 \cdot 10^{-1}$
1	1.55016	-2.47327	-3.17298	-2.47574	$-2.66053 \cdot 10^{-1}$
2	1.28411	-0.95915	-7.13907	-1.49260	$-1.11910 \cdot 10^{-1}$
3	1.17220	-0.15139	-7.17069	-1.27897	$-2.05572 \cdot 10^{-2}$
4	1.15164	-0.00466	-7.09910	-1.24620	$-6.55884 \cdot 10^{-4}$
5	<u>1.15098</u> $\doteq z_2$				

Beispiel 4.11. Für das Polynom von Beispiel 4.10 und der näherungsweise bekannten Nullstelle $z_1 = 1.95408$ wird mit der Technik der impliziten Deflation eine zweite Nullstelle bestimmt. Bei sechsstelliger Rechnung erhalten wir Werte, die in Tab. 4.9 wiedergegeben sind. Es wurde absichtlich die gleiche Startnäherung $x^{(0)} = 2$ wie im Beispiel 4.10 gewählt, um zu illustrieren, dass die Folge der iterierten Werte $x^{(k)}$ tatsächlich gegen eine andere Nullstelle z_2 konvergiert. \triangle

4.4.2 Komplexe Nullstellen: Das Verfahren von Bairstow

Das Verfahren von Newton zur Berechnung der Nullstellen von Polynomen ist ohne Änderung auch dann durchführbar, wenn entweder die Koeffizienten des Polynoms komplex sind oder bei reellen Koeffizienten paarweise konjugiert komplexe Nullstellen auftreten. Die Rechenschritte sind mit komplexen Zahlen und Operationen durchzuführen, wobei im zweiten genannten Fall der Startwert komplex zu wählen ist, da andernfalls die Folge der Iterierten $x^{(k)}$ reell bleibt.

Das Rechnen mit komplexen Zahlen kann aber im Fall von Polynomen $P_n(x)$ mit reellen Koeffizienten vollständig vermieden werden. Ist nämlich $z_1 = u + iv, v \neq 0$, eine komplexe Nullstelle von $P_n(x)$, dann ist bekanntlich auch der dazu konjugiert komplexe Wert $z_2 = u - iv$ Nullstelle von $P_n(x)$. Somit ist das Produkt der zugehörigen Linearfaktoren

$$(x - z_1)(x - z_2) = (x - u - iv)(x - u + iv) = x^2 - 2ux + (u^2 + v^2) \quad (4.85)$$

ein *quadratischer Teiler* von $P_n(x)$, dessen Koeffizienten nach (4.85) reell sind. Auf Grund dieser Feststellung formulieren wir als neue Zielsetzung die Bestimmung eines quadratischen Teilers mit reellen Koeffizienten von $P_n(x)$. Ist ein solcher quadratischer Teiler gefunden, folgen daraus entweder ein Paar von konjugiert komplexen Nullstellen oder zwei reelle Nullstellen des Polynoms $P_n(x)$. Die übrigen $(n - 2)$ Nullstellen werden anschließend als Nullstellen des Quotientenpolynoms $P_{n-2}(x)$ ermittelt.

Als Vorbereitung ist der Divisionsalgorithmus mit Rest für einen quadratischen Faktor zu formulieren. Das gegebene Polynom sei wieder

$$P_n(x) = a_0x^n + a_1x^{n-1} + \dots + a_{n-2}x^2 + a_{n-1}x + a_n, \quad a_j \in \mathbb{R}, \quad a_0 \neq 0.$$

Den quadratischen Faktor setzen wir wie folgt fest

$$x^2 - px - q, \quad p, q \in \mathbb{R}. \quad (4.86)$$

Gesucht sind die Koeffizienten des Quotientenpolynoms

$$P_{n-2}(x) = b_0x^{n-2} + b_1x^{n-3} + \dots + b_{n-4}x^2 + b_{n-3}x + b_{n-2}, \quad (4.87)$$

so dass gilt

$$P_n(x) = (x^2 - px - q)P_{n-2}(x) + b_{n-1}(x - p) + b_n. \quad (4.88)$$

Das lineare Restpolynom $R_1(x) = b_{n-1}(x - p) + b_n$ in (4.88) wird in dieser unkonventionellen Form angesetzt, damit die resultierende Rechenvorschrift zur Berechnung der Koeffizienten b_j systematisch für alle Indizes gilt. Der Koeffizientenvergleich liefert die Beziehungen

$$\begin{aligned} x^n : \quad a_0 &= b_0 \\ x^{n-1} : \quad a_1 &= b_1 - pb_0 \\ x^{n-j} : \quad a_j &= b_j - pb_{j-1} - qb_{j-2}, \quad j = 2, 3, \dots, n. \end{aligned}$$

Die gesuchten Koeffizienten von $P_{n-2}(x)$ und des Restes $R_1(x)$ lassen sich somit rekursiv berechnen mit dem einfachen Divisionsalgorithmus

$$\begin{aligned} b_0 &= a_0; & b_1 &= a_1 + pb_0 \\ b_j &= a_j + pb_{j-1} + qb_{j-2}, & j &= 2, 3, \dots, n. \end{aligned} \quad (4.89)$$

Ein Faktor $x^2 - px - q$ ist genau dann Teiler von $P_n(x)$, falls der Rest $R_1(x)$ identisch verschwindet, d.h. falls in (4.88) die beiden Koeffizienten $b_{n-1} = b_n = 0$ sind. Zu gegebenen Werten p und q sind sowohl die Koeffizienten von $P_{n-2}(x)$ als auch die Koeffizienten des Restes $R_1(x)$ eindeutig bestimmt. Sie können somit als Funktionen der beiden Variablen p und q aufgefasst werden. Die oben formulierte Aufgabe, einen quadratischen Teiler von $P_n(x)$ zu bestimmen, ist deshalb äquivalent damit, das nichtlineare Gleichungssystem

$$\boxed{b_{n-1}(p, q) = 0, \quad b_n(p, q) = 0} \quad (4.90)$$

nach den Unbekannten p und q zu lösen. Die Funktionen $b_{n-1}(p, q)$ und $b_n(p, q)$ sind durch den Algorithmus (4.89) definiert. Entsprechend unserer Zielsetzung interessieren allein die reellen Lösungen von (4.90), und diese sollen mit der Methode von Newton iterativ berechnet werden. Dazu werden die partiellen Ableitungen der beiden Funktionen $b_{n-1}(p, q)$ und $b_n(p, q)$ nach p und q benötigt. Diese können mit Hilfe einer einzigen Rekursionsformel berechnet werden, die zu (4.89) analog gebaut ist. Aus (4.89) folgt

$$\frac{\partial b_j}{\partial p} = b_{j-1} + p \frac{\partial b_{j-1}}{\partial p} + q \frac{\partial b_{j-2}}{\partial p}, \quad j = 2, 3, \dots, n, \quad (4.91)$$

und zusätzlich

$$\frac{\partial b_0}{\partial p} = 0, \quad \frac{\partial b_1}{\partial p} = b_0. \quad (4.92)$$

Die Struktur von (4.91) legt es nahe, die Hilfsgrößen

$$\boxed{c_{j-1} := \frac{\partial b_j}{\partial p}, \quad j = 1, 2, \dots, n,} \quad (4.93)$$

zu definieren. Denn so wird (4.91) unter Berücksichtigung von (4.92) zur Rechenvorschrift

$$\boxed{c_0 = b_0; \quad c_1 = b_1 + pc_0 \\ c_j = b_j + pc_{j-1} + qc_{j-2}, \quad j = 2, 3, \dots, n-1.} \quad (4.94)$$

Partielle Differenziationen von (4.89) nach q gibt die Identität

$$\frac{\partial b_j}{\partial q} = b_{j-2} + p \frac{\partial b_{j-1}}{\partial q} + q \frac{\partial b_{j-2}}{\partial q}, \quad j = 2, 3, \dots, n, \quad (4.95)$$

wobei jetzt zu beachten ist, dass zusätzlich

$$\frac{\partial b_0}{\partial q} = 0, \quad \frac{\partial b_1}{\partial q} = 0, \quad \frac{\partial b_2}{\partial q} = b_0, \quad \frac{\partial b_3}{\partial q} = b_1 + p \frac{\partial b_2}{\partial q}$$

gelten. Damit wird offensichtlich, dass mit der Identifikation

$$\boxed{\frac{\partial b_j}{\partial q} = c_{j-2}, \quad j = 2, 3, \dots, n,} \quad (4.96)$$

diese partiellen Ableitungen dieselbe Rekursionsformel (4.94) erfüllen; damit gilt

$$\frac{\partial b_j}{\partial p} = \frac{\partial b_{j+1}}{\partial q} = c_{j-1}, \quad j = 1, 2, \dots, n-1.$$

Im Speziellen erhalten wir nach (4.93) und (4.96) für die im Verfahren von Newton wichtigen Ableitungen

$$\boxed{\frac{\partial b_{n-1}}{\partial p} = c_{n-2}; \quad \frac{\partial b_{n-1}}{\partial q} = c_{n-3}; \quad \frac{\partial b_n}{\partial p} = c_{n-1}; \quad \frac{\partial b_n}{\partial q} = c_{n-2}.} \quad (4.97)$$

Damit sind alle Elemente bereitgestellt, die für einen Iterationsschritt zur Lösung des Systems (4.90) nötig sind. Durch (4.97) sind die vier Matrixelemente der Funktionalmatrix Φ gegeben. Aus der allgemeinen Rechenvorschrift (4.50) erhalten wir durch entsprechende Substitution

$$\boxed{\begin{aligned} p^{(k+1)} &= p^{(k)} + \frac{b_n c_{n-3} - b_{n-1} c_{n-2}}{c_{n-2}^2 - c_{n-1} c_{n-3}}, \\ q^{(k+1)} &= q^{(k)} + \frac{b_{n-1} c_{n-1} - b_n c_{n-2}}{c_{n-2}^2 - c_{n-1} c_{n-3}}. \end{aligned}} \quad (4.98)$$

In (4.98) sind die auftretenden b - und c -Werte für $p^{(k)}$ und $q^{(k)}$ zu berechnen. Der Nenner $c_{n-2}^2 - c_{n-1} c_{n-3}$ ist gleich der Determinante der Funktionalmatrix Φ . Ist die Determinante gleich null, so ist (4.98) nicht anwendbar. In diesem Fall sind die Näherungen $p^{(k)}$ und $q^{(k)}$ beispielsweise durch Addition von Zufallszahlen zu ändern. Weiter ist es möglich, dass die Determinante zwar nicht verschwindet, aber betragsmäßig klein ist. Das hat zur Folge, dass dem Betrag nach sehr große Werte $p^{(k+1)}$ und $q^{(k+1)}$ entstehen können, welche unmöglich brauchbare Näherungen für die Koeffizienten eines quadratischen Teilers des Polynoms darstellen können. Denn es ist zu beachten, dass die Beträge der Nullstellen eines Polynoms $P_n(x)$ (4.71) beschränkt sind beispielsweise durch

$$|z_i| \leq R := \max \left\{ \left| \frac{a_n}{a_0} \right|, 1 + \left| \frac{a_1}{a_0} \right|, 1 + \left| \frac{a_2}{a_0} \right|, \dots, 1 + \left| \frac{a_{n-1}}{a_0} \right| \right\}. \quad (4.99)$$

Die Schranke (4.99) lässt sich mit der so genannten Begleitmatrix, [Zur 65], deren charakteristisches Polynom $a_0^{-1} P_n(x)$ ist, herleiten. Diese Matrix enthält in der letzten Spalte im Wesentlichen die Koeffizienten von $a_0^{-1} P_n(x)$, in der Subdiagonalen Elemente gleich Eins und sonst lauter Nullen. Die Zeilensummennorm dieser Matrix stellt eine obere Schranke für die Beträge der Eigenwerte und somit für die Nullstellen von $P_n(x)$ dar. Nach (4.85) und (4.86) ist insbesondere $q = -(u^2 + v^2) = -|z_1|^2$ im Fall eines konjugiert komplexen Nullstellenpaars oder aber es ist $|q| = |z_1| \cdot |z_2|$ für zwei reelle Nullstellen. Deshalb ist es angezeigt zu prüfen, ob $|q^{(k+1)}| \leq R^2$ gilt. Andernfalls muss für $p^{(k+1)}$ und $q^{(k+1)}$ eine entsprechende Reduktion vorgenommen werden.

Die *Methode von Bairstow* zur Bestimmung eines quadratischen Teilers $x^2 - px - q$ eines Polynoms $P_n(x)$ mit reellen Koeffizienten vom Grad $n > 2$ besteht zusammengefasst aus den folgenden Teilschritten einer einzelnen Iteration:

1. Zu gegebenen Näherungen $p^{(k)}$ und $q^{(k)}$ berechne man die Werte b_j nach (4.89) und die Werte c_j nach (4.94).
2. Falls die Determinante der Funktionalmatrix Φ von null verschieden ist, können die iterierten Werte $p^{(k+1)}$ und $q^{(k+1)}$ gemäß (4.98) bestimmt werden.
3. Die Iteration soll abgebrochen werden, sobald die Werte b_{n-1} und b_n dem Betrag nach

in Relation zu den a_{n-1} und a_n , aus denen sie gemäß (4.89) entstehen, genügend klein sind. Wir verwenden deshalb die Bedingung

$$|b_{n-1}| + |b_n| \leq \varepsilon(|a_{n-1}| + |a_n|). \quad (4.100)$$

Darin ist $\varepsilon > 0$ eine Toleranz, die größer als die Maschinengenauigkeit τ ist.

4. Es sind die beiden Nullstellen z_1 und z_2 des gefundenen quadratischen Teilers zu bestimmen. Erst an dieser Stelle tritt im Fall eines konjugiert komplexen Nullstellenpaars eine komplexe Zahl auf.

5. Die Deflation der beiden Nullstellen erfolgt durch Division von $P_n(x)$ durch den zuletzt erhaltenen Teiler. Die Koeffizienten des Quotientenpolynoms sind durch (4.89) gegeben.

Das Verfahren von Bairstow besitzt die Konvergenzordnung $p = 2$, denn das zu lösende System von nichtlinearen Gleichungen (4.90) wird mit der Methode von Newton behandelt. Die im Verfahren zu berechnenden Größen des Divisionsalgorithmus werden zweckmäßigerweise in einem *doppelzeiligen Horner-Schema* zusammengestellt, welches sich für eine übersichtliche Handrechnung gut eignet. Im Fall $n = 6$ ist das Schema in (4.101) angegeben.

$$\begin{array}{ccccccc}
 & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\
 q) & & & qb_0 & qb_1 & qb_2 & qb_3 & qb_4 \\
 p) & & pb_0 & pb_1 & pb_2 & pb_3 & pb_4 & pb_5 \\
 \hline
 & b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\
 q) & & & qc_0 & qc_1 & qc_2 & qc_3 & \\
 p) & & pc_0 & pc_1 & pc_2 & pc_3 & pc_4 & \\
 \hline
 & c_0 & c_1 & c_2 & c_3 & c_4 & c_5 &
 \end{array} \quad (4.101)$$

Beispiel 4.12. Für das Polynom fünften Grades

$$P_5(x) = x^5 - 2x^4 + 3x^3 - 12x^2 + 18x - 12$$

lautet das doppelzeilige Horner-Schema (4.101) für die Startwerte $p^{(0)} = -2, q^{(0)} = -5$

$$\begin{array}{ccccccc}
 & 1 & -2 & 3 & -12 & 18 & -12 \\
 -5) & & & -5 & 20 & -30 & 20 \\
 -2) & & -2 & 8 & -12 & 8 & 8 \\
 \hline
 & 1 & -4 & 6 & -4 & -4 & 16 \\
 -5) & & -5 & 30 & -65 & & \\
 -2) & & -2 & 12 & -26 & 0 & \\
 \hline
 & 1 & -6 & 13 & 0 & -69 &
 \end{array}$$

Der Verlauf des Verfahrens von Bairstow ist bei Rechnung mit acht wesentlichen Dezimalstellen in Tab. 4.10 dargestellt. Der quadratische Teiler $x^2 + 1.7320508x + 4.7320507$ ergibt das konjugiert komplexe Nullstellenpaar $z_{1,2} = -0.86602540 \pm 1.9955076i$. Mit dem Quotientenpolynom $P_3(x) = x^3 - 3.7320508x^2 + 4.7320509x - 2.5358990$ kann mit der Methode von Bairstow ein weiterer quadratischer Teiler $x^2 - 1.7320506x + 1.2679494$ ermittelt werden mit dem zweiten konjugiert komplexen Nullstellenpaar $z_{3,4} = 0.86602530 \pm 0.71968714i$. Die fünfte Nullstelle ist $z_5 = 2$. \triangle

Tab. 4.10 Methode von Bairstow.

k	0	1	2	3	4
$p^{(k)}$	-2	-1.7681159	-1.7307716	-1.7320524	-1.7320508
$q^{(k)}$	-5	-4.6923077	-4.7281528	-4.7320527	-4.7320508
b_4	-4	0.17141800	0.0457260	-0.000029	$-1.0 \cdot 10^{-6}$
b_5	16	2.2742990	-0.0455520	0.000077	0.0
c_2	13	10.066549	9.4534922	9.4641203	9.4641012
c_3	0	5.0722220	6.9160820	6.9281770	6.9282040
c_4	-69	-56.032203	-56.621988	-56.784711	-56.784610
$\Delta p^{(k)}$	0.232	0.037344	-0.0012808	$1.5880 \cdot 10^{-6}$	$1.1835 \cdot 10^{-8}$
$\Delta q^{(k)}$	0.308	-0.035845	-0.0038999	$1.9017 \cdot 10^{-6}$	$9.6999 \cdot 10^{-8}$

4.5 Software

Die NAG-Bibliotheken enthalten im Kapitel C02 Routinen zur Bestimmung sämtlicher Nullstellen eines Polynoms und im Kapitel C05 zur Lösung nichtlinearer Gleichungen. Kann der Benutzer bei einer einzelnen Gleichung ein Intervall mit Vorzeichenwechsel angeben, so findet er Routinen mit garantierter Konvergenz gegen eine einzelne Nullstelle. Die anderen Routinen können auch versagen bzw. verlangen dem Benutzer etwas Experimentierfreude ab. Für Systeme von nichtlinearen Gleichungen gibt es Routinen, welche die Jacobi-Matrix verwenden, und solche, die die Ableitungen nur angenähert berechnen. Wann immer es geht, sollte man erstere benutzen, weil sie zuverlässiger Nullstellen finden können. Da ein nichtlineares Gleichungssystem als Spezialfall eines Problems kleinster Quadrate betrachtet werden kann, kommen auch die Routinen des entsprechenden Kapitels E04 in Betracht.

Die zuletzt genannte Möglichkeit muss in MATLAB gewählt werden zur Lösung eines Systems von nichtlinearen Gleichungen. Für eine einzelne Gleichung gibt es einen einfachen Befehl. Das Ergebnis der Nullstellensuche hängt vom Startwert oder -intervall ab, welches der Benutzers eingibt.

Unsere Problemlöseumgebung PAN (<http://www.upb.de/SchwarzKoeckler/>) verfügt über ein Programm zur Lösung einer nichtlinearen Gleichung und eines zur Lösung eines nichtlinearen Gleichungssystems.

4.6 Aufgaben

4.1. Eine Fixpunktiteration $x^{(n+1)} = F(x^{(n)})$ ist definiert durch $F(x) = 1 + \frac{1}{x} + \left(\frac{1}{x}\right)^2$.

- a) Man verifiziere, dass $F(x)$ für das Intervall $[1.75, 2.0]$ die Voraussetzungen des Fixpunktsatzes von Banach erfüllt. Wie groß ist die Lipschitz-Konstante L ?

b) Mit $x^{(0)} = 1.825$ bestimme man die Iterierten bis $x^{(20)}$. Welche a priori und a posteriori Fehlerschranken ergeben sich? Wie groß ist die Anzahl der erforderlichen Iterationsschritte zur Gewinnung einer weiteren richtigen Dezimalstelle?

4.2. Die nichtlineare Gleichung $2x - \sin(x) = 0.5$ kann in die Fixpunktform $x = 0.5 \sin(x) + 0.25 =: F(x)$ übergeführt werden. Man bestimme ein Intervall, für welches $F(x)$ die Voraussetzungen des Fixpunktsatzes von Banach erfüllt. Zur Bestimmung des Fixpunktes s verfahren man wie in Aufgabe 4.1.

4.3. Man berechne die positive Nullstelle von $f(x) = e^{2x} - \sin(x) - 2$

- a) mit der Sekantenmethode, $x^{(0)} = 0.25$, $x^{(1)} = 0.35$;
- b) mit der Methode von Newton, $x^{(0)} = 0.25$;
- c) mit der vereinfachten Methode von Newton, $x^{(0)} = 0.25$;
- d) mit der Methode von Brent, $x^{(0)} = 0$, $x^{(1)} = 1$, $x^{(2)} = 0.5$.

4.4. Wie in Aufgabe 4.3 bestimme man die kleinsten positiven Lösungen der Gleichungen

- a) $\tan(x) - \tanh(x) = 0$;
- b) $\tan(x) - x - 4x^3 = 0$;
- c) $(1+x^2)\tan(x) - x(1-x^2) = 0$.

4.5. Zur Bestimmung einer Lösung der Gleichung $f(x) = 0$ kann die Iterationsvorschchrift

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})f'(x^{(k)})}{f'(x^{(k)})^2 - 0.5f(x^{(k)})f''(x^{(k)})}, \quad k = 0, 1, 2, \dots$$

verwendet werden unter der Voraussetzung, dass $f(x)$ mindestens zweimal stetig differenzierbar ist. Man zeige, dass die Konvergenzordnung (mindestens) $p = 3$ ist. Wie lauten die daraus resultierenden Iterationsformeln zur Berechnung von \sqrt{a} als Lösung von $f(x) = x^2 - a = 0$ und von $\sqrt[3]{a}$ zur Lösung von $f(x) = x^3 - a = 0$? Man berechne mit diesen Iterationsvorschriften $\sqrt{2}$ und $\sqrt[3]{2}$ mit den Startwerten $x^{(0)} = 1$.

4.6. Das lineare Gleichungssystem

$$\begin{array}{rclclclclclcl} 8x_1 & + & x_2 & - & 2x_3 & - & 8 & = & 0 \\ x_1 & + & 18x_2 & - & 6x_3 & + & 10 & = & 0 \\ 2x_1 & + & x_2 & + & 16x_3 & + & 2 & = & 0 \end{array}$$

mit diagonal dominanter Matrix kann in folgende Fixpunktform übergeführt werden:

$$\begin{array}{lclclclclclcl} x_1 & = & 0.2x_1 & - & 0.1x_2 & + & 0.2x_3 & + & 0.8 \\ x_2 & = & -0.05x_1 & + & 0.1x_2 & + & 0.3x_3 & - & 0.5 \\ x_3 & = & -0.1x_1 & - & 0.05x_2 & + & 0.2x_3 & - & 0.1 \end{array}$$

Man verifizierte, dass die beiden Systeme äquivalent sind. Dann zeige man, dass die so definierte Abbildung in ganz \mathbb{R}^3 kontrahierend ist und bestimme die Lipschitz-Konstante L und den Spektralradius der Jacobi-Matrix. Mit dem Startwert $x^{(0)} = (1, -0.5, -0.3)^T$ führe man (mindestens) zehn Iterationsschritte durch und gebe die a priori und a posteriori Fehlerschranken an. An der Folge der iterierten Vektoren verifizierte man die Konvergenzrate.

4.7. Das nichtlineare Gleichungssystem

$$\begin{array}{rclclclclclcl} e^{xy} & + & x^2 & + & y & - & 1.2 & = & 0 \\ x^2 & + & y^2 & + & x & - & 0.55 & = & 0 \end{array}$$

ist zu lösen. Mit den Startwerten $x^{(0)} = 0.6$, $y^{(0)} = 0.5$ wende man die Methode von Newton an. Für die besseren Startwerte $x^{(0)} = 0.4$, $y^{(0)} = 0.25$ löse man das Gleichungssystem mit Hilfe des vereinfachten Newton-Verfahrens. Wie groß ist die Konvergenzrate der linear konvergenten Iterationsfolge auf Grund des Spektralradius der zugehörigen Jacobi-Matrix?

4.8. Zur Lösung des nichtlinearen Gleichungssystems

$$\begin{array}{lclcl} 4.72 \sin(2x) & - & 3.14e^y & - & 0.495 = 0 \\ 3.61 \cos(3x) & + & \sin(y) & - & 0.402 = 0 \end{array}$$

verwende man mit den Startwerten $x^{(0)} = 1.5$, $y^{(0)} = -4.7$ das Verfahren von Newton, das vereinfachte Verfahren von Newton, das Newtonsche Einzelschrittverfahren und das SOR-Newton-Verfahren für verschiedene Werte von ω .

4.9. Die Nullstellen der Polynome

$$P_5(x) = x^5 - 2x^4 - 13x^3 + 14x^2 + 24x - 1,$$

$$P_8(x) = x^8 - x^7 - 50x^6 + 62x^5 + 650x^4 - 528x^3 - 2760x^2 + 470x + 2185$$

sollen mit der Methode von Newton-Maehly bestimmt werden.

4.10. Mit der Methode von Bairstow berechne man die (komplexen) Nullstellen der Polynome

$$P_6(x) = 3x^6 + 9x^5 + 9x^4 + 5x^3 + 3x^2 + 8x + 5,$$

$$P_8(x) = 3x^8 + 2x^7 + 6x^6 + 4x^5 + 7x^4 + 3x^3 + 5x^2 + x + 8.$$

4.11. *Effektiver Jahreszins:* Gegeben sei folgender Kreditvertrag: Am Anfang erhalte der Kunde $K \in \mathbb{E}$. Nach einem, zwei, ..., n Jahren muss der Kunde eine gleich bleibende Rate von $R \in \mathbb{E}$ zahlen, welche sich aus der Tilgung und den Zinsen zusammensetzt:

$$R = \text{Tilgung} + \text{Zinsen} = K/n + Kq/100;$$

der Nominalzinssatz des Kreditvertrages ist also q (Prozent).

Gesucht ist nun der effektive Jahreszins \bar{p} , welcher wie folgt definiert ist:

Am Anfang wird ein (fiktives) Konto eingerichtet und der Auszahlungsbetrag von $K \in \mathbb{E}$ eingezahlt. Nach einem, zwei, ..., n Jahren wird der Kontostand mit $p\%$ verzinst und jeweils unmittelbar danach die Rate von $R \in \mathbb{E}$ abgebucht.

Der effektive Jahreszins ist nun der Zinssatz p , bei dem nach n Jahren der Kontostand gleich null ist, also: Sei $f(p)$ der Kontostand nach n Jahren. Dann ist $\bar{p} > 0$ definiert durch $f(\bar{p}) = 0$.

a) Man zeige:

$$\begin{aligned} f(p) &= (1 + p^*)^n K - \left(\sum_{j=0}^{n-1} (1 + p^*)^j \right) R \\ &= (1 + p^*)^n K - \frac{(1 + p^*)^n - 1}{p^*} R, \end{aligned}$$

wobei $p^* = p/100$.

b) Man formuliere das Newton-Verfahren für das Problem $f(p) = 0$.

c) Man berechne \bar{p} für $K = 100000$, $n = 10$ und $q = 8$.

5 Eigenwertprobleme

Es wird zunächst das *spezielle Eigenwertproblem* betrachtet, d.h. die Aufgabe die Eigenwerte und Eigenvektoren einer reellen, quadratischen Matrix \mathbf{A} zu berechnen:

$$\mathbf{Ax} = \lambda \mathbf{x}. \quad (5.1)$$

Daneben wird kurz auf das *allgemeine Eigenwertproblem*

$$\mathbf{Ax} = \lambda \mathbf{Bx}, \quad (5.2)$$

eingegangen, das in der Regel auf das spezielle Eigenwertproblem zurückgeführt werden kann.

Beide Aufgabenstellungen treten insbesondere in der Physik und den Ingenieurwissenschaften bei der Behandlung von Schwingungsproblemen auf. Auch ist etwa in der Statistik im Zusammenhang mit der Varianzanalyse eine Eigenwertaufgabe zu lösen. Dem Eigenwertproblem sind wir auch schon im Kapitel 2 begegnet, wo zur Berechnung der Spektralnorm (2.77) Eigenwerte benötigt werden. Die Bestimmung der Konditionszahl einer Matrix mit der Spektralnorm (Abschnitt 2.12) erfordert den größten und den kleinsten Eigenwert der Matrix. Auch bei der Lösung von gewöhnlichen Differenzialgleichungssystemen und von partiellen Differenzialgleichungen ist die Bestimmung der Eigenwerte von Matrizen für die Untersuchung des stabilen Verhaltens der Verfahren maßgebend (vgl. dazu die Kapitel 8 und 10).

Die Wahl des numerischen Verfahrens zur Behandlung des Eigenwertproblems hängt davon ab, ob alle oder nur wenige Eigenwerte bestimmt werden sollen und ob die Eigenvektoren zusätzlich berechnet werden müssen. Darüber hinaus sollte die Verfahrenswahl vom Matrixtyp abhängig gemacht werden (symmetrisch oder unsymmetrisch, voll oder schwach besetzt). Wir werden uns auf einige geeignete Methoden zur Eigenwertbestimmung beschränken, die Grundlage der in Bibliotheken verwendeten Algorithmen sind. Für ausführlichere und umfassendere Darstellungen von Eigenwertmethoden verweisen wir auf [Bun 95, Gol 96b, Jen 77, Par 98, Ste 83, Tör 88, Wil 88, Zur 97, Zur 86].

5.1 Theoretische Grundlagen

5.1.1 Das charakteristische Polynom

Gegeben sei eine reelle, quadratische Matrix \mathbf{A} der Ordnung n . Gesucht sind ihre (i.a. komplexen) Eigenwerte λ_k und die zugehörigen Eigenvektoren $\mathbf{x}_k \neq 0$, so dass

$$\mathbf{A}\mathbf{x}_k = \lambda_k \mathbf{x}_k \quad \text{oder} \quad (\mathbf{A} - \lambda_k \mathbf{I})\mathbf{x}_k = \mathbf{0} \quad (5.3)$$

gilt. Da der Eigenvektor \mathbf{x}_k nicht der Nullvektor sein darf, muss notwendigerweise die Determinante der Matrix $\mathbf{A} - \lambda_k \mathbf{I}$ verschwinden. Folglich muss der Eigenwert λ_k Nullstelle des *charakteristischen Polynoms*

$$P(\lambda) := (-1)^n |\mathbf{A} - \lambda \mathbf{I}| = \lambda^n + p_{n-1} \lambda^{n-1} + p_{n-2} \lambda^{n-2} + \dots + p_1 \lambda + p_0 \quad (5.4)$$

sein. Da das charakteristische Polynom den echten Grad n besitzt, hat es n Nullstellen, falls sie mit der entsprechenden Vielfachheit gezählt werden.

Auf Grund dieser Betrachtung scheint die Aufgabe der Bestimmung der Eigenwerte bereits gelöst, denn man braucht dazu nur die Koeffizienten des charakteristischen Polynoms zu berechnen, um anschließend seine Nullstellen nach einer der Methoden aus Kapitel 4 zu ermitteln. Die Eigenvektoren \mathbf{x}_k zu den Eigenwerten λ_k ergeben sich dann als nichttriviale Lösungen der homogenen Gleichungssysteme (5.3).

Das durch die Theorie vorgezeichnete Lösungsverfahren ist aber aus numerischen Gründen für größere Ordnungen n ungeeignet, da die Bestimmung der Nullstellen dieses Polynoms instabil sein kann. Deshalb werden im Folgenden nur solche Verfahren zur Behandlung der Eigenwertaufgabe betrachtet, welche die Berechnung der Koeffizienten des charakteristischen Polynoms nicht erfordern.

5.1.2 Ähnlichkeitstransformationen

Definition 5.1. 1. Ist $\mathbf{T} \in \mathbb{C}^{n,n}$ eine reguläre Matrix, dann heißt

$$\mathbf{T}^{-1} \mathbf{A} \mathbf{T} \quad (5.5)$$

Ähnlichkeitstransformation.

2. Eine Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ heißt *diagonalähnlich*, wenn es eine reguläre Matrix $\mathbf{T} \in \mathbb{C}^{n,n}$ gibt, so dass

$$\mathbf{\Lambda} := \mathbf{T}^{-1} \mathbf{A} \mathbf{T} \quad (5.6)$$

eine Diagonalmatrix ist: $\mathbf{\Lambda} := \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$, $\mathbf{\Lambda} \in \mathbb{C}^{n,n}$.

Die Zahlen λ_i sind dann gerade die Eigenwerte von \mathbf{A} .

Lemma 5.2. Ist $\mathbf{T} \in \mathbb{C}^{n,n}$ eine reguläre Matrix und

$$\mathbf{C} := \mathbf{T}^{-1} \mathbf{A} \mathbf{T}, \quad (5.7)$$

dann haben \mathbf{A} und \mathbf{C} dieselben Eigenwerte und zu einem Eigenvektor $\mathbf{y} \in \mathbb{C}^n$ von \mathbf{C} gehört der Eigenvektor $\mathbf{x} := \mathbf{T}\mathbf{y}$ von \mathbf{A} :

$$\mathbf{C}\mathbf{y} = \lambda\mathbf{y} \Rightarrow \mathbf{A}\mathbf{x} = \lambda\mathbf{x} \quad \text{mit } \mathbf{x} := \mathbf{T}\mathbf{y}. \quad (5.8)$$

Beweis.

$$\begin{aligned} P_C(\lambda) &= |\lambda \mathbf{I} - \mathbf{C}| = |\lambda \mathbf{I} - \mathbf{T}^{-1} \mathbf{A} \mathbf{T}| = |\mathbf{T}^{-1} (\lambda \mathbf{I} - \mathbf{A}) \mathbf{T}| \\ &= |\mathbf{T}^{-1}| |\lambda \mathbf{I} - \mathbf{A}| |\mathbf{T}| = |\lambda \mathbf{I} - \mathbf{A}| = P_A(\lambda). \end{aligned}$$

Dabei haben wir die Determinantenrelationen $|\mathbf{XY}| = |\mathbf{X}| |\mathbf{Y}|$ und $|\mathbf{T}^{-1}| = |\mathbf{T}|^{-1}$ verwendet. Aus $\mathbf{T}^{-1} \mathbf{ATy} = \lambda \mathbf{y}$ folgt $\mathbf{ATy} = \lambda \mathbf{Ty}$ und daraus $\mathbf{x} = \mathbf{Ty}$. \square

5.1.3 Symmetrische Eigenwertprobleme

Oft sind die Matrizen des Eigenwertproblems symmetrisch. Dies ist ein großer Vorteil, da dann alle Eigenwerte reell sind und ihre numerische Berechnung gut konditioniert ist, siehe Abschnitt 5.7.

Satz 5.3. *Sei \mathbf{A} eine symmetrische Matrix der Ordnung n . Dann besitzt \mathbf{A} nur reelle Eigenwerte, und die Eigenvektoren sind orthogonal. Ist also \mathbf{X} die Matrix mit den Eigenvektoren als Spalten und sind diese normiert auf $\|\mathbf{x}_i\|_2 = 1$, so ist*

$$\mathbf{X}^T \mathbf{X} = \mathbf{I} \implies \mathbf{X}^T = \mathbf{X}^{-1}. \quad (5.9)$$

Dies bedeutet auch, dass \mathbf{A} sich mit der Matrix der Eigenvektoren vermittels

$$\mathbf{X}^{-1} \mathbf{AX} = \mathbf{D}. \quad (5.10)$$

auf Diagonalgestalt ähnlich transformieren lässt, wobei die Diagonalmatrix \mathbf{D} gerade die Eigenwerte als Diagonalelemente enthält.

Dieses so genannte *Hauptachsensatz* bildet die Motivation für die folgenden Verfahren, in denen die Matrix \mathbf{A} mit orthogonalen Transformationen *ähnlich* auf Diagonalgestalt transformiert wird.

5.1.4 Elementare Rotationsmatrizen

Zur Transformation einer Matrix auf eine bestimmte Form benutzt man oft möglichst einfache Transformationsmatrizen, deren Effekt übersichtlich ist. Zu dieser Klasse gehören die orthogonalen Matrizen $\mathbf{U}(p, q; \varphi)$ (5.11), für die gilt:

$$\begin{aligned} u_{ii} &= 1, \quad i \neq p, q, & u_{pp} &= u_{qq} = \cos \varphi, \\ u_{pq} &= \sin \varphi, & u_{qp} &= -\sin \varphi, \\ u_{ij} &= 0 \quad \text{sonst.} \end{aligned}$$

Die Orthogonalität der Matrix $\mathbf{U}(p, q; \varphi)$ ist offenkundig, so dass aus $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ sofort $\mathbf{U}^{-1} = \mathbf{U}^T$ folgt. Wenn wir $\mathbf{U}(p, q; \varphi)$ als Darstellungsmatrix einer linearen Transformation im \mathbb{R}^n auffassen, entspricht sie einer Drehung um den Winkel $-\varphi$ in der zweidimensionalen Ebene, die durch die p -te und q -te Koordinatenrichtung aufgespannt wird. Das Indexpaar (p, q) mit $1 \leq p < q \leq n$ heißt das *Rotationsindexpaar* und die Matrix $\mathbf{U}(p, q; \varphi)$ eine (p, q) -*Rotationsmatrix*.

$$U(p, q; \varphi) := \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \cos \varphi & \sin \varphi \\ & & & & 1 \\ & & & & \ddots \\ & & & & & 1 \\ & & & -\sin \varphi & \cos \varphi & \\ & & & & & 1 \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix} \quad \begin{array}{l} \leftarrow p \\ \leftarrow q \end{array} \quad (5.11)$$

Die Wirkung einer (p, q) -Rotationsmatrix (5.11) auf eine Matrix \mathbf{A} im Fall einer Ähnlichkeitstransformation $\mathbf{A}'' = \mathbf{U}^{-1}\mathbf{AU} = \mathbf{U}^T\mathbf{AU}$ stellen wir in zwei Schritten fest. Die Multiplikation der Matrix \mathbf{A} von links mit \mathbf{U}^T zur Matrix $\mathbf{A}' = \mathbf{U}^T\mathbf{A}$ bewirkt nur eine Linearkombination der p -ten und q -ten Zeilen von \mathbf{A} , während die übrigen Matrixelemente unverändert bleiben. Die Elemente von $\mathbf{A}' = \mathbf{U}^T\mathbf{A}$ sind gegeben durch

$$\boxed{\begin{aligned} a'_{pj} &= a_{pj} \cos \varphi - a_{qj} \sin \varphi \\ a'_{qj} &= a_{pj} \sin \varphi + a_{qj} \cos \varphi \\ a'_{ij} &= a_{ij} \quad \text{für } i \neq p, q \end{aligned}} \quad j = 1, 2, \dots, n. \quad (5.12)$$

Die anschließende Multiplikation der Matrix \mathbf{A}' von rechts mit \mathbf{U} zur Bildung von $\mathbf{A}'' = \mathbf{A}'\mathbf{U}$ bewirkt jetzt nur eine Linearkombination der p -ten und q -ten Spalten von \mathbf{A}' , während alle anderen Spalten unverändert bleiben. Die Elemente von $\mathbf{A}'' = \mathbf{A}'\mathbf{U} = \mathbf{U}^T\mathbf{AU}$ sind gegeben durch

$$\boxed{\begin{aligned} a''_{ip} &= a'_{ip} \cos \varphi - a'_{iq} \sin \varphi \\ a''_{iq} &= a'_{ip} \sin \varphi + a'_{iq} \cos \varphi \\ a''_{ij} &= a'_{ij} \quad \text{für } j \neq p, q \end{aligned}} \quad i = 1, 2, \dots, n. \quad (5.13)$$

Zusammenfassend können wir festhalten, dass eine Ähnlichkeitstransformation der Matrix \mathbf{A} mit einer (p, q) -Rotationsmatrix \mathbf{U} nur die Elemente der p -ten und q -ten Zeilen und Spalten verändert, wie dies in Abb. 5.1 anschaulich dargestellt ist.

Die Matrixelemente in den vier Kreuzungspunkten werden sowohl nach (5.12) als auch nach (5.13) transformiert. Bevor wir die entsprechenden Formeln angeben, halten wir noch fest, dass bei einer orthogonalen Ähnlichkeitstransformation die Symmetrie erhalten bleibt. In der Tat gilt mit $\mathbf{A}^T = \mathbf{A}$ und $\mathbf{U}^{-1} = \mathbf{U}^T$ allgemein

$$\mathbf{A}''^T = (\mathbf{U}^{-1}\mathbf{AU})^T = (\mathbf{U}^T\mathbf{AU})^T = \mathbf{U}^T\mathbf{A}^T\mathbf{U} = \mathbf{U}^T\mathbf{AU} = \mathbf{A}''.$$

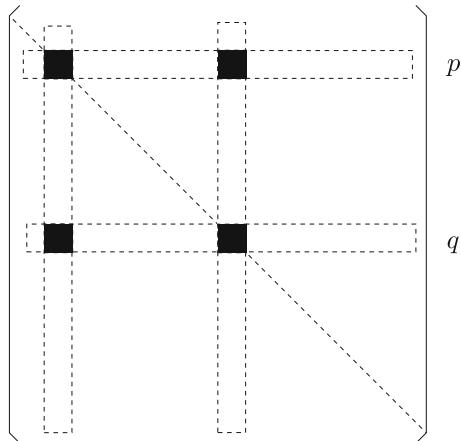


Abb. 5.1
Wirkung der Ähnlichkeitstransformation.

Nach Substitution von (5.12) in (5.13) sind die Elemente an den Kreuzungsstellen definiert durch

$$a''_{pp} = a_{pp} \cos^2 \varphi - 2a_{pq} \cos \varphi \sin \varphi + a_{qq} \sin^2 \varphi \quad (5.14)$$

$$a''_{qq} = a_{pp} \sin^2 \varphi + 2a_{pq} \cos \varphi \sin \varphi + a_{qq} \cos^2 \varphi \quad (5.15)$$

$$a''_{pq} = a''_{qp} = (a_{pp} - a_{qq}) \cos \varphi \sin \varphi + a_{pq} (\cos^2 \varphi - \sin^2 \varphi) \quad (5.16)$$

Solche elementaren orthogonalen Ähnlichkeitstransformationen wurden von *Jacobi* 1846 [Jac 46] zur sukzessiven Diagonalisierung einer symmetrischen Matrix verwendet. Sie werden deshalb *Jacobi-Rotationen* genannt oder, um das Rotationsindexpaar hervorzuheben, (p, q) -Drehungen.

Die tatsächliche Ausführung einer Jacobi-Rotation kann unter Berücksichtigung der Symmetrie mit den Matrixelementen in und unterhalb der Diagonalen erfolgen. Deshalb sind nach Abb. 5.1 insgesamt $2(n-2)$ Elemente gemäß (5.12) und (5.13) und dazu die drei Elemente an den Kreuzungsstellen umzurechnen. Beachtet man noch die Relation $a''_{pp} + a''_{qq} = a_{pp} + a_{qq}$, so sind nur etwas mehr als $4n$ Multiplikationen erforderlich.

5.2 Das klassische Jacobi-Verfahren

In diesem Abschnitt betrachten wir das Problem, für eine reelle *symmetrische Matrix A* alle Eigenwerte und die zugehörigen Eigenvektoren zu berechnen.

Auf Grund des Hauptachsentheorems lässt sich jede symmetrische Matrix A durch eine orthogonale Ähnlichkeitstransformation diagonalisieren. Die Idee von *Jacobi* besteht darin, diese Transformation durch eine Folge von elementaren Rotationen durchzuführen.

Eine naheliegende Strategie besteht darin, in einem einzelnen Transformationsschritt das momentan absolut größte Paar von Außendiagonalelementen $a_{pq} = a_{qp}$ zum Verschwinden zu bringen. Dies soll durch eine (p, q) -Drehung erfolgen, so dass also a_{pq} und a_{qp} in den

aufendiagonalen Kreuzungsstellen liegen (vgl. Abb. 5.1). Um unnötige Indizes zu vermeiden, betrachten wir einen Schritt der Transformationsfolge. Nach (5.16) ergibt sich folgende Bedingungsgleichung für den Drehwinkel φ

$$(a_{pp} - a_{qq}) \cos \varphi \sin \varphi + a_{qp}(\cos^2 \varphi - \sin^2 \varphi) = 0. \quad (5.17)$$

In (5.17) ist mit a_{qp} berücksichtigt, dass nur mit der unteren Hälfte der Matrix \mathbf{A} gearbeitet werden soll. Mit den trigonometrischen Identitäten

$$\sin(2\varphi) = 2 \cos \varphi \sin \varphi \quad \text{und} \quad \cos(2\varphi) = \cos^2 \varphi - \sin^2 \varphi$$

lautet (5.17)

$$\cot(2\varphi) = \frac{\cos^2 \varphi - \sin^2 \varphi}{2 \cos \varphi \sin \varphi} = \frac{a_{qq} - a_{pp}}{2a_{qp}} =: \Theta. \quad (5.18)$$

Da $a_{qp} \neq 0$ ist, hat der Quotient Θ stets einen endlichen Wert. Für die Jacobi-Rotation werden aber nur die Werte von $\cos \varphi$ und $\sin \varphi$ benötigt, nicht der des Winkels φ . Diese Werte können aus (5.18) numerisch sicher wie folgt berechnet werden. Mit $t := \tan \varphi$ ergibt sich aus (5.18)

$$\frac{1 - t^2}{2t} = \Theta \quad \text{oder} \quad t^2 + 2\Theta t - 1 = 0 \quad (5.19)$$

eine quadratische Gleichung für t mit den beiden Lösungen

$$t_{1,2} = -\Theta \pm \sqrt{\Theta^2 + 1} = \frac{1}{\Theta \pm \sqrt{\Theta^2 + 1}}. \quad (5.20)$$

Wir wählen die betragskleinere der beiden Lösungen und setzen fest

$$t = \tan \varphi = \begin{cases} \frac{1}{\Theta + \operatorname{sgn}(\Theta)\sqrt{\Theta^2 + 1}} & \text{für } \Theta \neq 0 \\ 1 & \text{für } \Theta = 0 \end{cases} \quad (5.21)$$

Damit wird erreicht, dass einerseits im Nenner von (5.21) keine Auslöschung stattfindet und dass andererseits $-1 < \tan \varphi \leq 1$ gilt, so dass der Drehwinkel φ auf das Intervall $-\pi/4 < \varphi \leq \pi/4$ beschränkt ist. Aus dem jetzt bekannten Wert für $\tan \varphi$ ergeben sich schließlich

$$\cos \varphi = \frac{1}{\sqrt{1+t^2}}, \quad \sin \varphi = t \cos \varphi \quad (5.22)$$

Damit sind die Elemente der Jacobi-Rotation festgelegt, und die Transformation der Matrix \mathbf{A} in $\mathbf{A}'' = \mathbf{U}^T \mathbf{A} \mathbf{U}$ kann nach den Formeln (5.12) und (5.13) erfolgen. Für die Diagonalelemente der transformierten Matrix \mathbf{A}'' ergeben sich infolge des aus der Gleichung (5.17) bestimmten Drehwinkels φ bedeutend einfachere Darstellungen. Aus (5.14) folgt nämlich mit (5.18)

$$\begin{aligned} a''_{pp} &= a_{pp} - 2a_{qp} \cos \varphi \sin \varphi + (a_{qq} - a_{pp}) \sin^2 \varphi \\ &= a_{pp} - a_{qp} \left\{ 2 \cos \varphi \sin \varphi - \frac{\cos^2 \varphi - \sin^2 \varphi}{\cos \varphi \sin \varphi} \sin^2 \varphi \right\} \\ &= a_{pp} - a_{qp} \tan \varphi. \end{aligned}$$

Also gelten die beiden Darstellungen

$$\boxed{a''_{pp} = a_{pp} - a_{qp} \tan \varphi, \quad a''_{qq} = a_{qq} + a_{qp} \tan \varphi} \quad (5.23)$$

Mit den Formeln (5.23) wird nicht nur die Zahl der Multiplikationen verkleinert, sondern es wird vor allem auch der Rundungsfehler in den Diagonalelementen verringert. Desgleichen lassen sich die Formeln (5.12) und (5.13) so umformen, dass sie bessere Eigenschaften hinsichtlich Rundungsfehlern aufweisen [Rut 66]. Dies gilt insbesondere für betragskleine Drehwinkel φ . Um die modifizierten Darstellungen der transformierten Matrixelemente zu erhalten, verwenden wir die Identität für

$$\cos \varphi = \frac{\cos \varphi + \cos^2 \varphi}{1 + \cos \varphi} = \frac{1 + \cos \varphi - \sin^2 \varphi}{1 + \cos \varphi} = 1 - \frac{\sin^2 \varphi}{1 + \cos \varphi}.$$

Mit dem Wert

$$\boxed{r := \frac{\sin \varphi}{1 + \cos \varphi} \left(= \tan \left(\frac{\varphi}{2} \right) \right)} \quad (5.24)$$

ergeben sich aus (5.12) und (5.13)

$$\boxed{\begin{aligned} a'_{pj} &= a_{pj} - \sin \varphi [a_{qj} + r a_{pj}] \\ a'_{qj} &= a_{qj} + \sin \varphi [a_{pj} - r a_{qj}] \end{aligned} \quad \left. \right\} j = 1, 2, \dots, n,} \quad (5.25)$$

$$\boxed{\begin{aligned} a''_{ip} &= a'_{ip} - \sin \varphi [a'_{iq} + r a'_{ip}] \\ a''_{iq} &= a'_{iq} + \sin \varphi [a'_{ip} - r a'_{iq}] \end{aligned} \quad \left. \right\} i = 1, 2, \dots, n.} \quad (5.26)$$

Man beachte, dass die Zahl der Multiplikationen gegenüber (5.12)/(5.13) unverändert bleibt.

Im *klassischen Jacobi-Verfahren* wird mit $\mathbf{A}^{(0)} = \mathbf{A}$ eine Folge von orthogonalähnlichen Matrizen

$$\mathbf{A}^{(k)} = \mathbf{U}_k^T \mathbf{A}^{(k-1)} \mathbf{U}_k, \quad k = 1, 2, \dots \quad (5.27)$$

gebildet, so dass im k -ten Schritt das absolut größte Nicht-Diagonalelement $a_{qp}^{(k-1)}$ der Matrix $\mathbf{A}^{(k-1)}$

$$|a_{qp}^{(k-1)}| = \max_{i>j} |a_{ij}^{(k-1)}| \quad (5.28)$$

durch eine Jacobi-Rotation mit $\mathbf{U}_k = \mathbf{U}(p, q; \varphi)$ zu null gemacht wird. Obwohl die Nicht-Diagonalelemente, die mit einer bestimmten Jacobi-Rotation zum Verschwinden gebracht werden, im Allgemeinen in einem nachfolgenden Transformationsschritt wieder ungleich null werden, gilt der

Satz 5.4. *Die Folge der zueinander ähnlichen Matrizen $\mathbf{A}^{(k)}$ (5.27) des klassischen Jacobi-Verfahrens konvergiert gegen eine Diagonalmatrix \mathbf{D} .*

Beweis. Für die Summe der Quadrate der Nicht-Diagonalelemente der Matrix $\mathbf{A}^{(k)}$

$$S(\mathbf{A}^{(k)}) = \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \{a_{ij}^{(k)}\}^2, \quad k = 1, 2, \dots, \quad (5.29)$$

zeigen wir, dass $S(\mathbf{A}^{(k)})$ mit wachsendem k eine monotone Nullfolge bildet. Dazu untersuchen wir vorbereitend die Änderung von $S(\mathbf{A})$ im Fall einer allgemeinen Jacobi-Rotation mit dem Rotationsindexpaar (p, q) . Die Summe $S(\mathbf{A}'') = S(\mathbf{U}^T \mathbf{A} \mathbf{U})$ wird dazu in Teilsummen aufgeteilt.

$$\begin{aligned} S(\mathbf{A}'') &= \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}''^2 && (5.30) \\ &= \sum_{\substack{i=1 \\ i \neq p, q}}^n \sum_{\substack{j=1 \\ j \neq i, p, q}}^n a_{ij}''^2 + \sum_{\substack{i=1 \\ i \neq p, q}}^n (a_{ip}''^2 + a_{iq}''^2) + \sum_{\substack{j=1 \\ j \neq p, q}}^n (a_{pj}''^2 + a_{qj}''^2) + 2a_{qp}''^2 \end{aligned}$$

Einerseits ändern sich bei einer (p, q) -Drehung nur die Elemente in den p -ten und q -ten Zeilen und Spalten, so dass $a_{ij}'' = a_{ij}$ für alle $i \neq p, q$ und alle $j \neq i, p, q$ gilt, und andererseits gelten wegen der Orthogonalität der Transformation

$$a_{ip}''^2 + a_{iq}''^2 = a_{ip}^2 + a_{iq}^2 \quad (i \neq p, q),$$

$$a_{pj}''^2 + a_{qj}''^2 = a_{pj}^2 + a_{qj}^2 \quad (j \neq p, q).$$

Deshalb folgt aus (5.30) für eine allgemeine Jacobi-Rotation

$$S(\mathbf{A}'') = S(\mathbf{U}^T \mathbf{A} \mathbf{U}) = \{S(\mathbf{A}) - 2a_{qp}^2\} + 2a_{qp}''^2. \quad (5.31)$$

Im klassischen Jacobi-Verfahren gilt nach dem k -ten Schritt $a_{qp}^{(k)} = 0$, so dass aus (5.31) folgt

$$S(\mathbf{A}^{(k)}) = S(\mathbf{A}^{(k-1)}) - 2a_{qp}^{(k-1)^2}, \quad k = 1, 2, \dots. \quad (5.32)$$

Die Werte $S(\mathbf{A}^{(k)})$ bilden mit zunehmendem k eine streng monoton abnehmende Folge, solange $\max_{i \neq j} |a_{ij}^{(k-1)}| \neq 0$ ist. Die Abnahme des Wertes $S(\mathbf{A}^{(k-1)})$ in der k -ten Jacobi-Rotation ist sogar maximal. Es bleibt noch zu zeigen, dass $S(\mathbf{A}^{(k)})$ eine Nullfolge bildet. Wegen (5.28) gilt

$$S(\mathbf{A}^{(k-1)}) \leq (n^2 - n)a_{qp}^{(k-1)^2},$$

und deshalb folgt für $S(\mathbf{A}^{(k)})$ aus (5.32) die Abschätzung

$$S(\mathbf{A}^{(k)}) = S(\mathbf{A}^{(k-1)}) - 2a_{qp}^{(k-1)^2} \leq \left\{ 1 - \frac{2}{n^2 - n} \right\} S(\mathbf{A}^{(k-1)}). \quad (5.33)$$

Da die Abschätzung (5.33) unabhängig von $a_{qp}^{(k-1)^2}$ ist, ergibt ihre rekursive Anwendung

$$S(\mathbf{A}^{(k)}) \leq \left\{ 1 - \frac{2}{n^2 - n} \right\}^k S(\mathbf{A}^{(0)}). \quad (5.34)$$

Für $n = 2$ ist $1 - 2/(n^2 - n) = 0$ im Einklang mit der Tatsache, dass mit $S(\mathbf{A}^{(1)}) = 0$ eine einzige Jacobi-Rotation in diesem Fall zur Diagonalisierung der Matrix der Ordnung zwei genügt. Für $n > 2$ ist $1 - 2/(n^2 - n) < 1$ und somit gilt

$$\lim_{k \rightarrow \infty} S(\mathbf{A}^{(k)}) = 0.$$

□

Das Produkt der Rotationsmatrizen

$$\mathbf{V}_k := \mathbf{U}_1 \mathbf{U}_2 \cdots \mathbf{U}_k, \quad k = 1, 2, \dots, \quad (5.35)$$

ist eine orthogonale Matrix, für die

$$\mathbf{A}^{(k)} = \mathbf{U}_k^T \mathbf{U}_{k-1}^T \cdots \mathbf{U}_2^T \mathbf{U}_1^T \mathbf{A}^{(0)} \mathbf{U}_1 \mathbf{U}_2 \cdots \mathbf{U}_{k-1} \mathbf{U}_k = \mathbf{V}_k^T \mathbf{A} \mathbf{V}_k \quad (5.36)$$

gilt. Nach Satz 5.4 stellt die Matrix $\mathbf{A}^{(k)}$ für hinreichend großes k mit beliebiger Genauigkeit eine Diagonalmatrix \mathbf{D} dar, d.h. es gilt

$$\mathbf{A}^{(k)} = \mathbf{V}_k^T \mathbf{A} \mathbf{V}_k \approx \mathbf{D}.$$

Die Diagonalelemente von $\mathbf{A}^{(k)}$ stellen Approximationen der Eigenwerte λ_j von \mathbf{A} dar. Die Spalten von \mathbf{V}_k sind Näherungen der zugehörigen orthonormierten Eigenvektoren \mathbf{x}_j . Insbesondere erhält man auch im Fall von mehrfachen Eigenwerten ein vollständiges System von orthonormierten Näherungen der Eigenvektoren. Auf Grund der Abschätzung (5.34) nimmt die Wertefolge $S(\mathbf{A}^{(k)})$ mindestens wie eine geometrische Folge mit dem Quotienten $q = 1 - 2/(n^2 - n)$ ab, d.h. die Konvergenz ist mindestens linear. Damit lässt sich die Anzahl der erforderlichen Jacobi-Rotationen zumindest abschätzen, um etwa die Bedingung

$$S(\mathbf{A}^{(k)}) / S(\mathbf{A}^{(0)}) \leq \varepsilon^2 \quad (5.37)$$

zu erfüllen. Bezeichnen wir mit $N = (n^2 - n)/2$ die Anzahl der Nicht-Diagonalelemente der unteren Hälfte der Matrix \mathbf{A} , so ist (5.37) sicher erfüllt, falls gilt

$$\left[1 - \frac{1}{N} \right]^k \leq \varepsilon^2.$$

Aufgelöst nach k erhalten wir daraus mit dem natürlichen Logarithmus für größere Werte von N

$$k \geq \frac{2 \ln(\varepsilon)}{\ln\left(1 - \frac{1}{N}\right)} \approx 2N \ln\left(\frac{1}{\varepsilon}\right) = (n^2 - n) \ln\left(\frac{1}{\varepsilon}\right). \quad (5.38)$$

Da eine Jacobi-Rotation rund $4n$ Multiplikationen erfordert, liefert (5.38) mit $\varepsilon = 10^{-\alpha}$ eine Schätzung des Rechenaufwandes Z_{Jacobi} zur Diagonalisierung einer symmetrischen Matrix \mathbf{A} der Ordnung n mit dem klassischen Jacobi-Verfahren von

$$Z_{\text{Jacobi}} = 4nk \approx 4n(n^2 - n)\alpha \ln(10) \approx 9.21\alpha(n^3 - n^2) \quad (5.39)$$

Multiplikationen. Der Aufwand steigt mit der dritten Potenz der Ordnung n , wobei der Faktor 9.21α von der geforderten Genauigkeit abhängt. Die Schätzung des Rechenaufwandes (5.39) ist zu pessimistisch, denn die Wertefolge der $S(\mathbf{A}^{(k)})$ konvergiert sogar *quadratisch* gegen null, sobald die Nicht-Diagonalelemente betragsmäßig genügend klein bezüglich der minimalen Differenz von zwei Eigenwerten geworden sind [Hen 58, Sch 61, Sch 64]. Falls für die Eigenwerte

$$\min_{i \neq j} |\lambda_i - \lambda_j| =: 2\delta > 0 \quad (5.40)$$

gilt, und im klassischen Jacobi-Verfahren der Zustand erreicht worden ist, dass

$$S(\mathbf{A}^{(k)}) < \frac{1}{4} \delta^2 \quad (5.41)$$

zutrifft, dann ist nach $N = (n^2 - n)/2$ weiteren Jacobi-Rotationen

$$S(\mathbf{A}^{(k+N)}) \leq \left(\frac{1}{2}n - 1\right) S(\mathbf{A}^{(k)})^2 / \delta^2. \quad (5.42)$$

Eine zu (5.42) analoge Aussage über die quadratische Konvergenz kann auch für den Fall von doppelten Eigenwerten gezeigt werden [Sch 61, Sch 64]. Die asymptotisch quadratische Konvergenz reduziert selbstverständlich den totalen Rechenaufwand wesentlich. Er bleibt aber doch proportional zu n^3 , wobei aber der Proportionalitätsfaktor im Vergleich zu (5.39) kleiner ist.

Eine Aussage über den absoluten Fehler der Approximation der Eigenwerte λ_j von \mathbf{A} durch die Diagonalelemente $a_{ii}^{(k)}$ der Matrix $\mathbf{A}^{(k)}$ liefert der in [Hen 58] bewiesene

Satz 5.5. *Die Eigenwerte λ_j der symmetrischen Matrix $\mathbf{A} = \mathbf{A}^{(0)}$ seien in aufsteigender Reihenfolge $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ angeordnet. Die der Größe nach geordneten Diagonalelemente $a_{ii}^{(k)}$ bezeichnen wir mit $d_j^{(k)}$, so dass $d_1^{(k)} \leq d_2^{(k)} \leq \dots \leq d_n^{(k)}$ gilt. Dann erfüllen die Eigenwerte λ_j die Fehlerabschätzung*

$$|d_j^{(k)} - \lambda_j| \leq \sqrt{S(\mathbf{A}^{(k)})}, \quad j = 1, 2, \dots, n; k = 0, 1, 2, \dots \quad (5.43)$$

Um die aufwändige Berechnung von $S(\mathbf{A}^{(k)})$ zu vermeiden, schätzen wir die Summe mit Hilfe des absolut größten Nicht-Diagonalelementes $a_{qp}^{(k)}$ von $\mathbf{A}^{(k)}$ durch $S(\mathbf{A}^{(k)}) \leq (n^2 - n)\{a_{qp}^{(k)}\}^2 < n^2\{a_{qp}^{(k)}\}^2$ ab. Dies liefert die gegenüber (5.43) im Allgemeinen bedeutend schlechtere Abschätzung

$$|d_j^{(k)} - \lambda_j| < n|a_{qp}^{(k)}|, \quad j = 1, 2, \dots, n; k = 0, 1, 2, \dots \quad (5.44)$$

Sie ist als einfaches Abbruchkriterium anwendbar.

Die gleichzeitige Berechnung der Eigenvektoren der Matrix \mathbf{A} ist bereits durch (5.35) und (5.36) vorgezeichnet. Die j -te Spalte von \mathbf{V}_k enthält eine Näherung des normierten Eigenvektors zur Eigenwertnäherung $a_{jj}^{(k)}$. Die so erhaltenen Approximationen der Eigenvektoren bilden innerhalb der numerischen Genauigkeit stets ein System von paarweise orthogonalen und normierten Vektoren. Dies trifft auf Grund der Konstruktion auch im Fall von mehrfachen Eigenwerten zu. Da die Konvergenz der Matrixfolge \mathbf{V}_k komplizierteren Gesetzen

gehorcht als die Konvergenz der Diagonalelemente $a_{jj}^{(k)}$ gegen die Eigenwerte, werden die Eigenvektoren in der Regel weniger gut approximiert [Wil 88]. Die Matrixfolge \mathbf{V}_k berechnet sich rekursiv durch sukzessive Multiplikation der Rotationsmatrizen gemäß der Vorschrift

$$\mathbf{V}_0 = \mathbf{I}; \quad \mathbf{V}_k = \mathbf{V}_{k-1} \mathbf{U}_k \quad k = 1, 2, \dots \quad (5.45)$$

Für die Matrizen \mathbf{V}_k ist eine volle $(n \times n)$ -Matrix vorzusehen, weil sie ja nicht symmetrisch sind. Ist \mathbf{U}_k eine (p, q) -Rotationsmatrix, bewirkt die Multiplikation $\mathbf{V}_{k-1} \mathbf{U}_k$ nur eine Linearkombination der p -ten und q -ten Spalten von \mathbf{V}_{k-1} gemäß den Formeln (5.13) oder (5.26). Der Rechenaufwand für diese Operation beträgt $4n$ wesentliche Operationen. Werden also die Eigenvektoren mitberechnet, so verdoppelt sich der Gesamtaufwand des klassischen Jacobi-Verfahrens.

Beispiel 5.1. Die Eigenwerte und Eigenvektoren der Matrix

$$\mathbf{A}^{(0)} = \begin{pmatrix} 20 & -7 & 3 & -2 \\ -7 & 5 & 1 & 4 \\ 3 & 1 & 3 & 1 \\ -2 & 4 & 1 & 2 \end{pmatrix} \quad \text{mit } S(\mathbf{A}^{(0)}) = 160$$

werden mit dem klassischen Jacobi-Verfahren berechnet. Da die Matrix sechs Elemente unterhalb der Diagonale enthält, sollen immer sechs Schritte ohne Zwischenergebnis ausgeführt werden. Nach sechs Rotationen lautet die resultierende Matrix

$$\mathbf{A}^{(6)} \doteq \begin{pmatrix} 23.524454 & 0.000000 & -0.005235 & 0.268672 \\ 0.000000 & 6.454132 & -0.090526 & -0.196317 \\ -0.005235 & -0.090526 & 1.137646 & -0.288230 \\ 0.268672 & -0.196317 & -0.288230 & -1.116232 \end{pmatrix}$$

Die Summe der Quadrate der Nicht-Diagonalelemente ist $S(\mathbf{A}^{(6)}) \doteq 0.404$. In den nachfolgenden drei \times sechs Rotationen werden $S(\mathbf{A}^{(12)}) \doteq 3.648 \cdot 10^{-6}$, $S(\mathbf{A}^{(18)}) \doteq 9.0 \cdot 10^{-23}$ und $S(\mathbf{A}^{(24)}) \doteq 1.5 \cdot 10^{-32}$. Die sehr rasche Konvergenz ist deutlich erkennbar und setzt in diesem Beispiel sehr früh ein, weil die Eigenwerte, entnommen aus der Diagonale von $\mathbf{A}^{(24)}$, $\lambda_1 \doteq 23.527386$, $\lambda_2 \doteq 6.460515$, $\lambda_3 \doteq 1.173049$ und $\lambda_4 \doteq -1.160950$ mit $\min_{i \neq j} |\lambda_i - \lambda_j| \doteq 2.334$ gut getrennt sind. Die Matrix

$\mathbf{V}_{24} = \mathbf{I} \mathbf{U}_1 \mathbf{U}_2 \dots \mathbf{U}_{24}$ mit den Näherungen der Eigenvektoren lautet

$$\mathbf{V}_{24} \doteq \begin{pmatrix} 0.910633 & 0.260705 & -0.269948 & -0.172942 \\ -0.370273 & 0.587564 & -0.249212 & -0.674951 \\ 0.107818 & 0.549910 & 0.819957 & 0.116811 \\ -0.148394 & 0.533292 & -0.438967 & 0.707733 \end{pmatrix}.$$

Eigenwerte und Eigenvektoren ändern sich in den letzten sechs Schritten innerhalb der gezeigten Stellen nicht mehr. \triangle

Das *zyklische Jacobi-Verfahren* erspart sich den aufwändigen Suchprozess zur Ermittlung des absolut größten Nicht-Diagonalelementes, der $N = (n^2 - n)/2$ Vergleichsoperationen erfordert.

Da eine Jacobi-Rotation nur $4n$ Multiplikationen benötigt, steht der Aufwand des Suchprozesses, zumindest für größere Ordnungen n , in einem ungünstigen Verhältnis zu demjenigen der eigentlichen Transformation. Deswegen werden beim zyklischen Jacobi-Verfahren die N Nicht-Diagonalelemente unterhalb der Diagonale in systematischer und immer gleich

bleibender Reihenfolge in einem Zyklus von N Rotationen je einmal zum Verschwinden gebracht.

Auch das zyklische Jacobi-Verfahren erzeugt eine Folge von Matrizen $\mathbf{A}^{(k)}$, die gegen die Diagonalmatrix der Eigenwerte konvergiert. Der Nachweis der Konvergenz ist aber bei weitem nicht mehr so elementar wie im Fall des klassischen Jacobi-Verfahrens.

5.3 Die Vektoriteration

5.3.1 Die einfache Vektoriteration nach von Mises

Es soll nur der einfachste Fall beschrieben werden: \mathbf{A} sei eine reguläre Matrix der Ordnung n mit nur reellen Eigenwerten und einem System von n linear unabhängigen Eigenvektoren \mathbf{x}_i , $i = 1, \dots, n$. λ_1 sei ein einfacher Eigenwert mit

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|. \quad (5.46)$$

Im Algorithmus (5.47) wird die Bestimmung des betragsgrößten Eigenwertes λ_1 und des zugehörigen Eigenvektors \mathbf{x}_1 beschrieben.

- | | |
|--|--|
| <ol style="list-style-type: none"> (1) Wähle ein $\mathbf{z}^{(0)} \in \mathbb{R}^n$ mit $\ \mathbf{z}^{(0)}\ _2 = 1$, setze $i := 0$. (2) Berechne $\mathbf{u}^{(i)} := \mathbf{A}\mathbf{z}^{(i)}$, den Index k mit
 $u_k^{(i)} = \max_j u_j^{(i)} ,$
 $s_i := \text{sign} \left(\frac{z_k^{(i)}}{u_k^{(i)}} \right)$
 $\mathbf{z}^{(i+1)} := s_i \frac{\mathbf{u}^{(i)}}{\ \mathbf{u}^{(i)}\ _2}$ und
 $\lambda_1^{(i)} := s_i \ \mathbf{u}^{(i)}\ _2$ (3) Wenn $\ \mathbf{z}^{(i+1)} - \mathbf{z}^{(i)}\ _2 < \varepsilon$, dann gehe nach (5). (4) Setze $i := i + 1$ und gehe nach (2). (5) Setze
 $\tilde{\lambda}_1 := \lambda_1^{(i)}$
 $\tilde{\mathbf{x}}_1 = \mathbf{z}^{(i+1)}.$ | $\tilde{\lambda}_1 \text{ und } \tilde{\mathbf{x}}_1 \text{ sind Näherungen für den gesuchten Eigenwert und zugehörigen Eigenvektor.}$ |
|--|--|

Wann, warum und wie dieses Verfahren konvergiert, sagt

Satz 5.6. Für den Startvektor $\mathbf{z}^{(0)}$ der Vektoriteration gelte:

$$\mathbf{z}^{(0)} = \sum_{j=1}^n c_j \mathbf{x}_j \quad \text{mit } c_1 \neq 0, \quad (5.48)$$

wo $\{\mathbf{x}_j\}$ das System der linear unabhängigen Eigenvektoren ist.

Dann konvergieren nach Algorithmus (5.47)

$$\text{und } s_i \|\mathbf{u}^{(i)}\|_2 \rightarrow \lambda_1 \text{ mit } i \rightarrow \infty \\ \mathbf{z}^{(i)} \rightarrow \mathbf{x}_1 \text{ mit } i \rightarrow \infty. \quad (5.49)$$

Die Konvergenz ist linear mit der Konvergenzrate $|\lambda_2|/|\lambda_1|$.

Im Algorithmus verursachen die Matrix-Vektor-Multiplikationen in (2) den Hauptaufwand. Dieser Aufwand kann bei schwach besetztem \mathbf{A} stark verringert werden, wenn die Elemente von \mathbf{A} entsprechend gespeichert sind. Deswegen ist dieser Algorithmus für den Fall großer, schwach besetzter Matrizen geeignet, einen oder mehrere Eigenwerte zu berechnen. Im Fall mehrerer Eigenwerte wird der Algorithmus nach einer Idee von F. L. Bauer verallgemeinert, siehe [Rut 69] oder [Köc 94].

Beispiel 5.2. Sei

$$\mathbf{A} = \begin{pmatrix} + & \times & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \times & + & \times & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \times & + & \times & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \times & + & \times & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \times & + & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \times & + & \times & \cdot & \cdot & \cdot & \times & \cdot & \cdot \\ \times & \cdot & \cdot & \cdot & \cdot & + & \times & \cdot & \cdot & \cdot & \times & \cdot & \cdot \\ \cdot & \times & \cdot & \cdot & \cdot & \times & + & \times & \cdot & \cdot & \cdot & \times & \cdot \\ \cdot & \cdot & \times & \cdot & \cdot & \cdot & \times & + & \times & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \times & + & \times & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & + & \times & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \times & + \end{pmatrix}$$

mit den folgenden Abkürzungen: $+$ $\equiv 0.1111$, \times $\equiv -0.0278$, \cdot $\equiv 0$.

Von einer solchen Matrix würde man normalerweise nur die Elemente ungleich null in der Form $(i, j, a(i, j))$ speichern. Wir suchen den absolut größten Eigenwert und den zugehörigen Eigenvektor. Wir erwarten auf Grund von Problem-Vorkenntnissen, dass aufeinander folgende Komponenten dieses Eigenvektors entgegengesetzte Vorzeichen aufweisen. Deshalb wählen wir als Startvektor

$$\mathbf{z}^{(0)} := \frac{1}{\sqrt{12}}(-1, 1, -1, 1, -1, 1, 1, -1, 1, -1, 1)^T, \quad \|\mathbf{z}^{(0)}\|_2 = 1.$$

Er liefert die Folge von iterierten Vektoren

$$\mathbf{u}^{(0)} := \mathbf{A}\mathbf{z}^{(0)} \rightarrow \mathbf{z}^{(1)} = \frac{\mathbf{u}^{(0)}}{\|\mathbf{u}^{(0)}\|_2} \rightarrow \mathbf{u}^{(1)} := \mathbf{A}\mathbf{z}^{(1)} \rightarrow \mathbf{z}^{(2)} = \frac{\mathbf{u}^{(1)}}{\|\mathbf{u}^{(1)}\|_2} \rightarrow \dots,$$

deren Zahlenwerte in Tab. 5.1 stehen.

Die Folge $\lambda_1^{(i)}$ der Approximationen für den größten Eigenwert λ_1 nach Schritt (2) des Algorithmus (5.47) konvergiert nach Satz 5.6 linear mit der asymptotischen Konvergenzrate $|\lambda_2|/|\lambda_1| = 0.9065$, also sehr langsam:

Tab. 5.1 Vektorsequenz zur Potenzmethode.

$\mathbf{u}^{(0)}$	$\mathbf{z}^{(1)}$	$\mathbf{u}^{(1)}$	$\mathbf{z}^{(2)}$...	$\mathbf{z}^{(12)}$	\mathbf{x}_1
-0.0481	-0.2587	-0.0455	-0.2418	...	-0.2235	-0.2290
0.0561	0.3018	0.0587	0.3119	...	0.3482	0.3569
-0.0561	-0.3018	-0.0587	-0.3119	...	-0.3510	-0.3527
0.0561	0.3018	0.0575	0.3055	...	0.2858	0.2742
-0.0481	-0.2587	-0.0443	-0.2354	...	-0.1632	-0.1489
0.0561	0.3018	0.0575	0.3055	...	0.2916	0.2961
-0.0642	-0.3449	-0.0707	-0.3755	...	-0.4277	-0.4361
0.0561	0.3018	0.0599	0.3182	...	0.3734	0.3749
-0.0561	-0.3018	-0.0575	-0.3055	...	-0.2919	-0.2804
0.0481	0.2587	0.0443	0.2354	...	0.1647	0.1505
-0.0481	-0.2587	-0.0443	-0.2354	...	-0.1790	-0.1795
0.0481	0.2587	0.0455	0.2418	...	0.2133	0.2158

Schritt Nr.	1	2	3	4	5	6
$\lambda_1^{(i)}$	0.1860	0.1882	0.1892	0.1897	0.1900	0.1901
	7	8	9	10	11	12
	0.1902	0.1903	0.1903	0.1903	0.1904	0.1904

Nach zwölf Schritten bekommen wir eine auf 4 wesentliche Stellen genaue Approximation des Eigenwerts $\lambda_1 \approx \lambda_1^{(12)} = 0.1904$. Die zugehörige Eigenvektorapproximation $\mathbf{z}^{(12)}$ und der korrekte Eigenvektor \mathbf{x}_1 sind auch in Tab. 5.1 zu finden. Die Konvergenz wäre noch langsamer gewesen, wenn wir einen anderen Startvektor gewählt hätten wie z.B.

$$\mathbf{z}^{(0)} := (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T / \sqrt{12}. \quad \triangle$$

5.3.2 Die inverse Vektoriteration

Soll nicht der größte Eigenwert, sondern irgendein Eigenwert λ_k und der zugehörige Eigenvektor \mathbf{x}_k bestimmt werden, so können diese mit Hilfe der *inversen Vektoriteration* berechnet werden. Ist $\bar{\lambda}$ ein Näherungswert für den Eigenwert λ_k mit

$$0 < |\lambda_k - \bar{\lambda}| =: \varepsilon \ll \delta := \min_{i \neq k} |\lambda_i - \bar{\lambda}|, \quad (5.50)$$

dann wird ausgehend von einem geeigneten Startvektor $\mathbf{z}^{(0)}$ die Folge von Vektoren $\mathbf{z}^{(\nu)}$ auf Grund der Vorschrift

$$(\mathbf{A} - \bar{\lambda}\mathbf{I})\mathbf{z}^{(\nu)} = \mathbf{z}^{(\nu-1)}, \quad \nu = 1, 2, \dots, \quad (5.51)$$

gebildet. Wenn wir wieder voraussetzen, dass die Matrix \mathbf{A} ein System von n linear unabhängigen Eigenvektoren $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ besitzt, hat $\mathbf{z}^{(0)}$ die eindeutige Darstellung

$$\mathbf{z}^{(0)} = \sum_{i=1}^n c_i \mathbf{x}_i. \quad (5.52)$$

Für die iterierten Vektoren $\mathbf{z}^{(\nu)}$ gilt dann wegen $(\mathbf{A} - \bar{\lambda}\mathbf{I})\mathbf{x}_i = (\lambda_i - \bar{\lambda})\mathbf{x}_i$

$$\mathbf{z}^{(\nu)} = \sum_{i=1}^n \frac{c_i}{(\lambda_i - \bar{\lambda})^\nu} \mathbf{x}_i. \quad (5.53)$$

Ist weiter in der Entwicklung (5.52) der Koeffizient $c_k \neq 0$, dann folgt für $\mathbf{z}^{(\nu)}$ aus (5.53)

$$\mathbf{z}^{(\nu)} = \frac{1}{(\lambda_k - \bar{\lambda})^\nu} \left[c_k \mathbf{x}_k + \sum_{\substack{i=1 \\ i \neq k}}^n c_i \left(\frac{\lambda_k - \bar{\lambda}}{\lambda_i - \bar{\lambda}} \right)^\nu \mathbf{x}_i \right] \quad (5.54)$$

eine rasche lineare Konvergenz mit der Konvergenzrate $K = \varepsilon/\delta$ gegen den Eigenvektor \mathbf{x}_k , falls wie im Algorithmus (5.47) nach jedem Iterationsschritt der resultierende Vektor $\mathbf{z}^{(\nu)}$ normiert wird. Die Konvergenz wird gemäß (5.50) und (5.54) durch die gegenseitige Lage der Eigenwerte und die Güte der Näherung $\bar{\lambda}$ bestimmt. Außerdem hängt sie vom Startvektor $\mathbf{z}^{(0)}$ ab.

Die Lösung des linearen Gleichungssystems (5.51) nach $\mathbf{z}^{(\nu)}$ erfolgt mit dem Gauß-Algorithmus unter Verwendung der relativen Spaltenmaximumstrategie. Wenn es sich bei $\mathbf{A} - \bar{\lambda}\mathbf{I}$ um eine Hessenberg-Matrix handelt wie in Abschnitt 5.4.1, wird mit Vorteil die im Abschnitt 2.3.3 entwickelte Rechentechnik für tridiagonale Gleichungssysteme angewandt. Werden die Zeilenvertauschungen wie dort vor Ausführung des betreffenden Eliminationsschrittes vorgenommen, dann entsteht in Analogie zu (2.122) ein Schlusschema mit Hessenberg-Struktur. Die resultierende Rechtsdreiecksmatrix ist selbstverständlich ausgefüllt mit von null verschiedenen Elementen. Das Vorgehen vereinfacht sowohl die Eliminationsschritte, d.h. im Wesentlichen die Zerlegung, als auch die Vorrwärtssubstitution.

Die Matrix $\mathbf{A} - \bar{\lambda}\mathbf{I}$ ist fast singulär, weil $\bar{\lambda}$ eine gute Näherung eines Eigenwertes darstellt. Folglich ist das Gleichungssystem (5.51) schlecht konditioniert. Deshalb ist zu erwarten, dass der berechnete Vektor $\tilde{\mathbf{z}}^{(\nu)}$ einen großen relativen Fehler gegenüber dem exakten Vektor $\mathbf{z}^{(\nu)}$ aufweist. Es zeigt sich, dass der Fehler $\tilde{\mathbf{z}}^{(\nu)} - \mathbf{z}^{(\nu)}$ eine dominante Komponente in Richtung des betragskleinsten Eigenwertes von $\mathbf{A} - \bar{\lambda}\mathbf{I}$ besitzt. Nun ist aber $\lambda_k - \bar{\lambda}$ der betragskleinste Eigenwert von $\mathbf{A} - \bar{\lambda}\mathbf{I}$; deshalb ist der Fehler in der Lösung im Wesentlichen proportional zu \mathbf{x}_k , d.h. in der gewünschten Richtung. Diese Feststellung macht die inverse Vektoriteration überhaupt erst zu einem brauchbaren Verfahren zur Eigenvektorberechnung.

5.4 Transformationsmethoden

Die numerische Behandlung der Eigenwertaufgabe wird stark vereinfacht, falls die gegebene Matrix \mathbf{A} durch eine orthogonale Ähnlichkeitstransformation zuerst auf eine geeignete Form gebracht wird. Im Folgenden behandeln wir diesen vorbereitenden Schritt, wobei wir in erster Linie die Transformation einer unsymmetrischen Matrix \mathbf{A} auf Hessenberg-Form betrachten; das ist eine um eine Nebendiagonale erweiterte Dreiecksmatrix. Die Anwendung derselben Transformation auf eine symmetrische Matrix liefert eine Tridiagonalmatrix. Auf die Matrizen in der einfacheren Form werden dann andere Verfahren zur Berechnung von Eigenwerten und Eigenvektoren angewendet.

5.4.1 Transformation auf Hessenberg-Form

Eine gegebene *unsymmetrische* Matrix \mathbf{A} der Ordnung n soll durch eine Ähnlichkeitstransformation auf eine obere *Hessenberg-Matrix*

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & \dots & h_{1n} \\ h_{21} & h_{22} & h_{23} & h_{24} & \dots & h_{2n} \\ 0 & h_{32} & h_{33} & h_{34} & \dots & h_{3n} \\ 0 & 0 & h_{43} & h_{44} & \dots & h_{4n} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & h_{n,n-1} & h_{nn} \end{pmatrix} \quad (5.55)$$

gebracht werden, für deren Elemente gilt

$$h_{ij} = 0 \quad \text{für alle } i > j + 1. \quad (5.56)$$

Die gewünschte Transformation wird erreicht durch eine geeignete Folge von Jacobi-Rotationen, wobei in jedem Schritt ein Matrixelement zu null gemacht wird. Dabei wird darauf geachtet, dass ein einmal annuliertes Matrixelement durch nachfolgende Drehungen den Wert null beibehält. Im Gegensatz zu den Jacobi-Verfahren müssen die Rotationsindexpaare so gewählt werden, dass das zu annullierende Element nicht im außendiagonalen Kreuzungspunkt der veränderten Zeilen und Spalten liegt. Zur Unterscheidung spricht man deshalb auch von *Givens-Rotationen* [Par 98]. Zur Durchführung der Transformation existieren zahlreiche Möglichkeiten. Es ist naheliegend, die Matrixelemente unterhalb der Nebendiagonale spaltenweise in der Reihenfolge

$$a_{31}, a_{41}, \dots, a_{n1}, a_{42}, a_{52}, \dots, a_{n2}, \dots, a_{n,n-2} \quad (5.57)$$

zu eliminieren. Mit den korrespondierenden Rotationsindexpaaren

$$(2, 3), (2, 4), \dots, (2, n), (3, 4), (3, 5), \dots, (3, n), \dots, (n-1, n) \quad (5.58)$$

und noch geeignet festzulegenden Drehwinkel φ wird das Ziel erreicht werden. Zur Elimination des aktuellen Elementes $a_{ij} \neq 0$ mit $i \geq j + 2$ wird gemäß (5.57) und (5.58) eine $(j+1, i)$ -Drehung angewandt. Das Element a_{ij} wird durch diese Rotation nur von der Zeilenoperation betroffen. Die Forderung, dass das transformierte Element a'_{ij} verschwinden soll, liefert nach (5.12) wegen $p = j + 1 < i = q$ die Bedingung

$$a'_{ij} = a_{j+1,j} \sin \varphi + a_{ij} \cos \varphi = 0. \quad (5.59)$$

Zusammen mit der Identität $\cos^2 \varphi + \sin^2 \varphi = 1$ ist das Wertepaar $\cos \varphi$ und $\sin \varphi$, abgesehen von einem frei wählbaren Vorzeichen, bestimmt. Aus einem bald ersichtlichen Grund soll der Drehwinkel φ auf das Intervall $[-\pi/2, \pi/2]$ beschränkt werden. Deshalb werden die Werte

wie folgt festgelegt.

Falls	$a_{j+1,j} \neq 0 :$	
	$\cos \varphi = \frac{ a_{j+1,j} }{\sqrt{a_{j+1,j}^2 + a_{ij}^2}}, \quad \sin \varphi = \frac{-\operatorname{sgn}(a_{j+1,j}) a_{ij}}{\sqrt{a_{j+1,j}^2 + a_{ij}^2}}$	(5.60)
Falls	$a_{j+1,j} = 0 :$	
	$\cos \varphi = 0, \quad \sin \varphi = 1$	

Nun bleibt noch zu verifizieren, dass mit der Rotationsreihenfolge (5.58) die Transformation einer Matrix \mathbf{A} auf Hessenberg-Form (5.55) erreicht wird. Dazu ist zu zeigen, dass die bereits erzeugten Nullelemente durch spätere Drehungen unverändert bleiben. Für die erste Spalte ist dies offensichtlich, denn jede der $(2, i)$ -Rotationen betrifft nur die zweiten und i -ten Zeilen und Spalten, wobei genau das Element a_{i1} eliminiert wird. Für die Transformation der weiteren Spalten zeigen wir dies durch einen Induktionsschluss. Dazu nehmen wir an, dass die ersten r Spalten bereits auf die gewünschte Form gebracht worden seien. Zur Elimination der Elemente $a_{i,r+1}$ der $(r+1)$ -ten Spalte mit $i \geq r+3$ werden sukzessive $(r+2, i)$ -Drehungen angewandt. Unter der getroffenen Voraussetzung werden in den ersten r Spalten durch die Zeilenoperation nur Nullelemente miteinander linear kombiniert, so dass diese unverändert bleiben. Da $i > r+2$ ist, werden in der Spaltenoperation nur Spalten mit Indizes größer als $(r+1)$ verändert. Somit wird mit insgesamt $N^* = (n-1)(n-2)/2$ Givens-Rotationen die Ähnlichkeitstransformation von \mathbf{A} auf eine obere Hessenberg-Matrix \mathbf{H} erzielt. Um den dazu erforderlichen Rechenaufwand zu bestimmen, sehen wir uns noch die Transformation des Elementes $a_{j+1,j}$ näher an. Nach (5.12) und mit (5.60) ergibt sich

$$a'_{j+1,j} = \frac{a_{j+1,j} |a_{j+1,j}| + \operatorname{sgn}(a_{j+1,j}) a_{ij}^2}{\sqrt{a_{j+1,j}^2 + a_{ij}^2}} = \operatorname{sgn}(a_{j+1,j}) \sqrt{a_{j+1,j}^2 + a_{ij}^2}.$$

Werden die Formeln (5.60) modifiziert zu

Falls	$a_{j+1,j} \neq 0 :$	$w := \operatorname{sgn}(a_{j+1,j}) \sqrt{a_{j+1,j}^2 + a_{ij}^2},$	
	$\cos \varphi = \frac{a_{j+1,j}}{w}, \quad \sin \varphi = \frac{-a_{ij}}{w}$		(5.61)
Falls	$a_{j+1,j} = 0 :$	$w := -a_{ij},$	
	$\cos \varphi = 0, \quad \sin \varphi = 1,$		

dann ist $a'_{j+1,j} = w$, und die Berechnung dieses neuen Elementes erfordert keine Operation. Eine Givens-Rotation zur Elimination des Elementes a_{ij} der j -ten Spalte benötigt vier multiplikative Operationen und eine Quadratwurzel gemäß (5.61), dann $4(n-j)$ Multiplikationen für die Zeilenoperation (5.12) und schließlich $4n$ Multiplikationen für die Spaltenoperation (5.13). Zur Behandlung der j -ten Spalte sind somit $4(n-j-1)(2n-j+1)$ Multiplikationen und $(n-j-1)$ Quadratwurzeln erforderlich. Die Summation über j von 1 bis $(n-2)$ ergibt einen Rechenaufwand von

$$Z_{\text{HessG}} = (n-2)(n-1) \left(\frac{10}{3}n + 2 \right) = \frac{10}{3}n^3 - 8n^2 + \frac{2}{3}n + 4 \quad (5.62)$$

Multiplikationen und $N^* = (n - 1)(n - 2)/2$ Quadratwurzeln.

Da wir im Folgenden die Eigenwertaufgabe für die Hessenberg-Matrix \mathbf{H} lösen werden, müssen wir uns dem Problem zuwenden, wie die Eigenvektoren von \mathbf{A} aus denjenigen von \mathbf{H} erhalten werden können. Für \mathbf{H} gilt die Darstellung

$$\begin{aligned}\mathbf{H} &= \mathbf{U}_{N^*}^T \dots \mathbf{U}_2^T \mathbf{U}_1^T \mathbf{A} \mathbf{U}_1 \mathbf{U}_2 \dots \mathbf{U}_{N^*} = \mathbf{Q}^T \mathbf{A} \mathbf{Q}, \\ \mathbf{Q} &= \mathbf{U}_1 \mathbf{U}_2 \dots \mathbf{U}_{N^*}.\end{aligned}$$

Darin ist \mathbf{U}_k die k -te Jacobi-Rotationsmatrix, und \mathbf{Q} ist als Produkt der N^* orthogonalen Matrizen \mathbf{U}_k selbst orthogonal. Die Eigenwertaufgabe $\mathbf{Ax} = \lambda x$ geht mit der Matrix \mathbf{Q} über in

$$\mathbf{Q}^T \mathbf{A} \mathbf{Q} \mathbf{Q}^T x = \mathbf{H}(\mathbf{Q}^T x) = \lambda(\mathbf{Q}^T x),$$

d.h. zwischen den Eigenvektoren x_j von \mathbf{A} und y_j von \mathbf{H} besteht die Beziehung

$$y_j = \mathbf{Q}^T x_j, \quad \text{oder} \quad x_j = \mathbf{Q} y_j = \mathbf{U}_1 \mathbf{U}_2 \dots \mathbf{U}_{N^*} y_j. \quad (5.63)$$

Ein Eigenvektor x_j berechnet sich somit aus y_j durch sukzessive Multiplikation des letzteren mit den Jacobi-Rotationsmatrizen \mathbf{U}_k , jedoch in *umgekehrter* Reihenfolge, wie sie bei der Transformation von \mathbf{A} auf Hessenberg-Form zur Anwendung gelangten. Dazu muss die Information über die einzelnen Jacobi-Matrizen zur Verfügung stehen. Es ist naheliegend, die Werte $\cos \varphi$ und $\sin \varphi$ abzuspeichern, doch sind dazu $(n - 1)(n - 2) \approx n^2$ Speicherplätze erforderlich. Nach einem geschickten Vorschlag [Ste 76] genügt aber ein einziger Zahlenwert, aus dem sich $\cos \varphi$ und $\sin \varphi$ mit geringem Rechenaufwand und vor allem numerisch genau wieder berechnen lassen. Man definiert den Zahlenwert

$$\varrho := \begin{cases} 1, & \text{falls } \sin \varphi = 1, \\ \sin \varphi, & \text{falls } |\sin \varphi| < \cos \varphi, \\ \operatorname{sgn}(\sin \varphi) / \cos \varphi, & \text{falls } |\sin \varphi| \geq \cos \varphi \text{ und } \sin \varphi \neq 1. \end{cases} \quad (5.64)$$

Hier wird wesentlich von der oben getroffenen Begrenzung des Drehwinkels Gebrauch gemacht, denn dadurch ist sichergestellt, dass $\cos \varphi \geq 0$ gilt. Die Berechnung von $\cos \varphi$ und $\sin \varphi$ aus ϱ erfolgt durch entsprechende Fallunterscheidungen.

Der die Rotation vollkommen charakterisierende Wert ϱ kann an die Stelle des eliminierten Matrixelementes a_{ij} gespeichert werden. Nach ausgeführter Transformation von \mathbf{A} auf Hessenbergform \mathbf{H} ist die Information für die Rücktransformation der Eigenvektoren unterhalb der Nebendiagonale vorhanden.

Die Transformation einer unsymmetrischen Matrix \mathbf{A} der Ordnung n auf Hessenberg-Form besitzt die einfache algorithmische Beschreibung (5.65). Die Bereitstellung der Werte ϱ ist

allerdings nicht berücksichtigt. Dabei stellt τ die Maschinengenauigkeit dar, siehe (1.7).

für $j = 1, 2, \dots, n-2$:

für $i = j+2, j+3, \dots, n$:

falls $a_{ij} \neq 0$:

falls $|a_{j+1,j}| < \tau \times |a_{ij}|$:

$$w = -a_{ij}; c = 0; s = 1$$

sonst

$$w = \operatorname{sgn}(a_{j+1,j}) \sqrt{a_{j+1,j}^2 + a_{ij}^2}$$

$$c = a_{j+1,j}/w; s = -a_{ij}/w$$

$$a_{j+1,j} = w; a_{ij} = 0$$

für $k = j+1, j+2, \dots, n$:

$$h = c \times a_{j+1,k} - s \times a_{ik}$$

$$a_{ik} = s \times a_{j+1,k} + c \times a_{ik}; a_{j+1,k} = h$$

für $k = 1, 2, \dots, n$:

$$h = c \times a_{k,j+1} - s \times a_{ki}$$

$$a_{ki} = s \times a_{k,j+1} + c \times a_{ki}; a_{k,j+1} = h$$

(5.65)

Beispiel 5.3. Die Matrix

$$\mathbf{A} = \begin{pmatrix} 7 & 3 & 4 & -11 & -9 & -2 \\ -6 & 4 & -5 & 7 & 1 & 12 \\ -1 & -9 & 2 & 2 & 9 & 1 \\ -8 & 0 & -1 & 5 & 0 & 8 \\ -4 & 3 & -5 & 7 & 2 & 10 \\ 6 & 1 & 4 & -11 & -7 & -1 \end{pmatrix} \quad (5.66)$$

wird mittels einer orthogonalen Ähnlichkeitstransformation auf Hessenberg-Form transformiert. Nach vier Givens-Transformationen zur Elimination der Elemente der ersten Spalte lautet die zu \mathbf{A} ähnliche Matrix $\mathbf{A}^{(4)}$, falls die Matrixelemente auf sechs Dezimalstellen nach dem Komma gerundet werden.

$$\mathbf{A}^{(4)} \doteq \begin{pmatrix} 7.000000 & -7.276069 & 3.452379 & -9.536895 & -5.933434 & -6.323124 \\ -12.369317 & 4.130719 & -6.658726 & 8.223249 & 0.509438 & 19.209667 \\ 0 & -0.571507 & 4.324324 & 7.618758 & 9.855831 & -1.445000 \\ 0 & -0.821267 & 3.340098 & -0.512443 & -0.698839 & -5.843013 \\ 0 & 1.384035 & -3.643194 & 0.745037 & -0.162309 & 5.627779 \\ 0 & -6.953693 & 0.630344 & -4.548600 & -3.872656 & 4.219708 \end{pmatrix}$$

Die Fortsetzung der Transformation liefert die Hessenberg-Matrix \mathbf{H} , in welcher unterhalb der

Subdiagonale die Werte ϱ (5.64) anstelle der Nullen eingesetzt sind.

$$\mathbf{H} \doteq \begin{pmatrix} 7.000000 & -7.276069 & -5.812049 & 0.139701 & 9.015201 & -7.936343 \\ -12.369317 & 4.130719 & 18.968509 & -1.207073 & -10.683309 & 2.415951 \\ -0.164399 & -7.160342 & 2.447765 & -0.565594 & 4.181396 & -3.250955 \\ -1.652189 & -1.750721 & -8.598771 & 2.915100 & 3.416858 & 5.722969 \\ -0.369800 & 1.706882 & -1.459149 & -1.046436 & -2.835101 & 10.979178 \\ 0.485071 & -4.192677 & 4.429714 & 1.459546 & -1.414293 & 5.341517 \end{pmatrix} \quad (5.67)$$

△

5.4.2 Transformation auf tridiagonale Form

Wird die Folge der im letzten Abschnitt beschriebenen Ähnlichkeitstransformationen auf eine *symmetrische* Matrix \mathbf{A} angewendet, so ist die resultierende Matrix $\mathbf{J} = \mathbf{Q}^T \mathbf{A} \mathbf{Q}$ infolge der Bewahrung der Symmetrie *tridiagonal*.

$$\mathbf{J} = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \beta_2 & \alpha_3 & \beta_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix} \quad (5.68)$$

Den Prozess bezeichnet man als *Methode von Givens* [Giv 54, Giv 58]. Unter Berücksichtigung der Symmetrie vereinfacht sich der oben beschriebene Algorithmus, da allein mit den Matrixelementen in und unterhalb der Diagonale gearbeitet werden kann. Die typische Situation ist in Abb. 5.2 zur Elimination eines Elementes a_{ij} (\bullet), ($i \geq j + 2$) dargestellt.

Aus Abb. 5.2 ergibt sich folgender Ablauf für die Durchführung der Givens-Rotation:

1. In der j -ten Spalte ist nur das Element $a_{j+1,j}$ (\times) zu ersetzen.
2. Es werden die drei Matrixelemente in den Kreuzungspunkten umgerechnet.
3. Behandlung der Matrixelemente zwischen den Kreuzungspunkten in der $(j+1)$ -ten Spalte und der i -ten Zeile.
4. Transformation der Elemente unterhalb der i -ten Zeile, die in der i -ten und in der $(j+1)$ -ten Spalte liegen.

Zur Reduktion des Rechenaufwandes schreiben wir die Formeln (5.14), (5.15) und (5.16) mit $z := (a_{pp} - a_{qq}) \sin \varphi + 2a_{pq} \cos \varphi$ in der Form

$$\begin{aligned} a''_{pp} &= a_{pp} - (z \sin \varphi), & a''_{qq} &= a_{qq} + (z \sin \varphi), \\ a''_{qp} &= -a_{qp} + z \cos \varphi. \end{aligned}$$

Damit beträgt der Rechenaufwand zur Elimination von a_{ij} nur $4 + 5 + 4(n - j - 2) = 4(n - j) + 1$ Multiplikationen und eine Quadratwurzel. Zur Behandlung der j -ten Spalte sind somit $(n - j - 1)(4n - 4j + 1)$ Multiplikationen und $(n - j - 1)$ Quadratwurzeln nötig. Nach Summation über j von 1 bis $(n - 2)$ ergibt sich der Rechenaufwand der Methode von

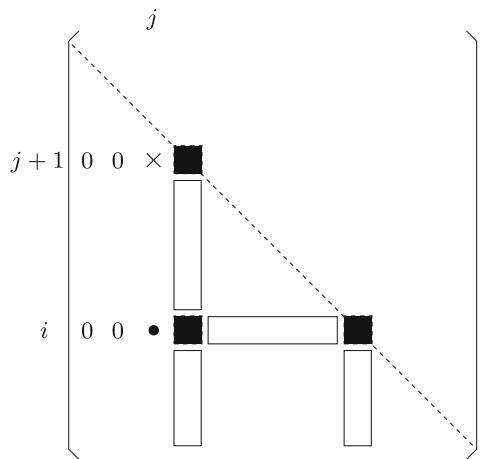


Abb. 5.2
Givens-Rotation für eine symmetrische Matrix.

Givens zu

$$Z_{\text{Givens}} = \frac{4}{3}n^3 - \frac{7}{2}n^2 + \frac{7}{6}n + 1 \quad (5.69)$$

Multiplikationen und $N^* = (n-1)(n-2)/2$ Quadratwurzeln.

Die algorithmische Beschreibung der Transformation auf tridiagonale Gestalt geht aus (5.65) dadurch hervor, dass dort die beiden letzten Schleifenanweisungen ersetzt werden durch die Anweisungen in (5.70).

$$\begin{aligned} d &= a_{j+1,j+1} - a_{ii}; \quad z = d \times s + 2 \times c \times a_{i,j+1}; \\ h &= z \times s; \quad a_{j+1,j+1} = a_{j+1,j+1} - h; \quad a_{ii} = a_{ii} + h; \\ a_{i,j+1} &= z \times c - a_{i,j+1} \\ \text{für } k &= j+2, j+3, \dots, i-1: \\ h &= c \times a_{k,j+1} - s \times a_{ik} \quad (5.70) \\ a_{ik} &= s \times a_{k,j+1} + c \times a_{ik}; \quad a_{k,j+1} = h \\ \text{für } k &= i+1, i+2, \dots, n: \\ h &= c \times a_{k,j+1} - s \times a_{ki} \\ a_{ki} &= s \times a_{k,j+1} + c \times a_{ki}; \quad a_{k,j+1} = h \end{aligned}$$

Beispiel 5.4. Die Transformation der symmetrischen Matrix

$$\mathbf{A} = \begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 6 & 0 & 4 & 3 \\ 3 & 0 & 7 & 6 & 5 \\ 2 & 4 & 6 & 8 & 7 \\ 1 & 3 & 5 & 7 & 9 \end{pmatrix}$$

auf tridiagonale Form ergibt

$$\mathbf{J} \doteq \begin{pmatrix} 5.000000 & 5.477226 & 0 & 0 & 0 \\ 5.477226 & 13.933333 & 9.298506 & 0 & 0 \\ -0.600000 & 9.298506 & 9.202474 & -2.664957 & 0 \\ -0.371391 & -3.758508 & -2.664957 & 4.207706 & 2.154826 \\ -0.182574 & -1.441553 & 0.312935 & 2.154826 & 2.656486 \end{pmatrix}. \quad (5.71)$$

Unterhalb der Subdiagonale sind wieder die Werte ϱ (5.64) der Givens-Rotationen eingesetzt, die allenfalls für eine Rücktransformation der Eigenvektoren von \mathbf{J} in diejenigen von \mathbf{A} gebraucht werden. \triangle

5.4.3 Schnelle Givens-Transformation

Wir betrachten die Multiplikation einer Matrix \mathbf{A} mit einer (p, q) -Rotationsmatrix $\mathbf{U}(p, q; \varphi)$ (5.11) $\mathbf{A}' = \mathbf{U}^T \mathbf{A}$. Da nur die Matrixelemente der p -ten und q -ten Zeilen geändert werden, richten wir unser Augenmerk darauf und schreiben zur Vereinfachung $c = \cos \varphi$, $s = \sin \varphi$. Nach den Formeln (5.12) sind zur Berechnung eines Paares von geänderten Matrixelementen

$$\begin{aligned} a'_{pj} &= ca_{pj} - sa_{qj} \\ a'_{qj} &= sa_{pj} + ca_{qj} \end{aligned} \quad (5.72)$$

vier Multiplikationen erforderlich. Es gelingt aber nach einer Idee von *Gentleman* [Gen 73, Ham 74, Rat 82] die Zahl der Operationen im Wesentlichen auf die Hälfte zu reduzieren, indem man sowohl die Matrix \mathbf{A} als auch die transformierte Matrix \mathbf{A}' mit geeignet zu wählenden regulären Diagonalmatrizen \mathbf{D} und \mathbf{D}' in der faktorisierten Form ansetzt

$$\mathbf{A} = \mathbf{D}\tilde{\mathbf{A}}, \quad \mathbf{A}' = \mathbf{D}'\tilde{\mathbf{A}}', \quad \mathbf{D} = \text{diag}(d_1, d_2, \dots, d_n). \quad (5.73)$$

Dabei wird für den Iterationsprozess $\mathbf{A}^{(0)} \rightarrow \mathbf{A}^{(1)} \rightarrow \mathbf{A}^{(2)} \dots$ die Diagonalmatrix $\mathbf{D} = \mathbf{D}_0 = \mathbf{I}$ gesetzt. Aus (5.72) erhalten wir so die neue Darstellung

$$\begin{aligned} d'_p \tilde{a}'_{pj} &= c d_p \tilde{a}_{pj} - s d_q \tilde{a}_{qj} \\ d'_q \tilde{a}'_{qj} &= s d_p \tilde{a}_{pj} + c d_q \tilde{a}_{qj} \end{aligned} \quad (5.74)$$

Um die Elemente \tilde{a}'_{pj} und \tilde{a}'_{qj} mit nur zwei Multiplikationen berechnen zu können, existieren im Wesentlichen die folgenden vier Möglichkeiten zur Festlegung der Diagonalelemente d'_p und d'_q .

- a) $d'_p = c d_p$ und $d'_q = c d_q$
- b) $d'_p = s d_q$ und $d'_q = s d_p$
- c) $d'_p = c d_p$ und $d'_q = s d_p$
- d) $d'_p = s d_q$ und $d'_q = c d_q$

Im Folgenden verwenden wir den Fall a) von (5.75), falls $|c| \geq |s|$ ist, sonst den Fall b). Dies geschieht im Hinblick darauf, dass stets eine Folge von Multiplikationen mit Rotationsmatrizen anzuwenden sein wird, so dass entsprechende Diagonalelemente in \mathbf{D} mit c oder s zu multiplizieren sein werden. Wir erhalten somit

$$\begin{aligned} \text{Fall a)} \quad \tilde{a}'_{pj} &= \tilde{a}_{pj} - \left(\frac{sd_q}{cd_p} \right) \tilde{a}_{qj} \\ \tilde{a}'_{qj} &= \left(\frac{sd_p}{cd_q} \right) \tilde{a}_{pj} + \tilde{a}_{qj} \end{aligned} \tag{5.76}$$

$$\begin{aligned} \text{Fall b)} \quad \tilde{a}'_{pj} &= \left(\frac{cd_p}{sd_q} \right) \tilde{a}_{pj} - \tilde{a}_{qj} \\ \tilde{a}'_{qj} &= \tilde{a}_{pj} + \left(\frac{cd_q}{sd_p} \right) \tilde{a}_{qj} \end{aligned} \tag{5.77}$$

Mit (5.76) oder (5.77) ist bereits der entscheidende Schritt vollzogen worden, die neuen Elemente \tilde{a}'_{pj} und \tilde{a}'_{qj} der diagonalskalierten Matrix $\tilde{\mathbf{A}}'$ nach Vorbereitung der einschlägigen Multiplikatoren mit zwei Multiplikationen zu berechnen. Wir gehen noch einen Schritt weiter und beachten, dass eine Multiplikation von \mathbf{A} mit \mathbf{U}^T das Ziel hat, das Matrixelement $a_{qk} \neq 0$ mit einem bestimmten Index k zum Verschwinden zu bringen. Nach (5.74) soll dann

$$d'_q \tilde{a}'_{qk} = s d_p \tilde{a}_{pk} + c d_q \tilde{a}_{qk} = 0$$

sein, also

$$T := \cot \varphi = \frac{c}{s} = -\frac{d_p \tilde{a}_{pk}}{d_q \tilde{a}_{qk}}. \tag{5.78}$$

Mit (5.78) gilt dann aber für die beiden Multiplikatoren in (5.77)

$$-\frac{cd_p}{sd_q} = \frac{d_p^2 \tilde{a}_{pk}}{d_q^2 \tilde{a}_{qk}} =: f_1, \quad -\frac{cd_q}{sd_p} = \frac{\tilde{a}_{pk}}{\tilde{a}_{qk}} =: f_2, \tag{5.79}$$

während die beiden Multiplikatoren in (5.76) die Kehrwerte von (5.79) sind. Aus (5.79) erkennt man aber, dass zur Bestimmung der Multiplikatoren f_1 und f_2 gar nicht die Diagonalelemente d_p und d_q benötigt werden, sondern ihre Quadrate d_p^2 und d_q^2 . Deshalb führt man diese Werte mit und ersetzt (5.75) durch

$$\begin{aligned} \text{a)} \quad (d'_p)^2 &= c^2(d_p)^2 \quad \text{und} \quad (d'_q)^2 = c^2(d_q)^2 \\ \text{b)} \quad (d'_p)^2 &= s^2(d_q)^2 \quad \text{und} \quad (d'_q)^2 = s^2(d_p)^2 \end{aligned} \tag{5.80}$$

Die in (5.80) benötigten Werte c^2 oder s^2 lassen sich entweder aus $T^2 = \cot^2 \varphi$ oder aus seinem Kehrwert $t^2 = \tan^2 \varphi$ auf Grund von trigonometrischen Identitäten berechnen. Die Entscheidung, ob Fall a) oder b) vorliegt, erfolgt auf Grund des nach (5.79) berechneten Wertes von $T^2 = f_1 f_2$, welcher unter der getroffenen Voraussetzung $\tilde{a}_{qk} \neq 0$ problemlos gebildet werden kann. Mit $T^2 \geq 1$ liegt der Fall a) vor, und es sind für die jetzt gültigen

Multiplikatoren die Kehrwerte zu bilden. Mit dieser Fallunterscheidung erhalten wir

$$\boxed{\begin{aligned} \text{a)} \quad t^2 &= 1/T^2; & c^2 &= \cos^2 \varphi = \frac{1}{1+t^2} \\ \text{b)} \quad T^2 &= f_1 f_2; & s^2 &= \sin^2 \varphi = \frac{1}{1+T^2} \end{aligned}} \quad (5.81)$$

Mit dieser Modifikation ist gleichzeitig die Berechnung der Quadratwurzel in (5.61) eliminiert worden. Die *schnelle Givens-Transformation* beruht also darauf, die *Quadrat*e der Diagonalelemente von \mathbf{D} der faktorisierten Matrix $\mathbf{A} = \mathbf{D}\tilde{\mathbf{A}}$ nachzuführen und die Matrix $\tilde{\mathbf{A}}$ umzurechnen, so dass $\mathbf{D}'\tilde{\mathbf{A}}'$ gleich der transformierten Matrix \mathbf{A}' ist. Wird die transformierte Matrix nach einer Folge von Givens-Transformationen benötigt, ist die faktorisierte Darstellung noch auszumultiplizieren. Dazu sind n Quadratwurzeln und eine der Form der transformierten Matrix entsprechende Zahl von Multiplikationen nötig.

Nun wenden wir uns den Givens-Rotationen als Ähnlichkeitstransformationen zu. Da sowohl eine Multiplikation von links als auch eine Multiplikation von rechts erfolgt, sind die Matrizen in der folgenden faktorisierten Form anzusetzen

$$\mathbf{A} = \mathbf{D}\tilde{\mathbf{A}}, \quad \mathbf{A}'' = \mathbf{D}'\tilde{\mathbf{A}}''\mathbf{D}', \quad \mathbf{D} = \text{diag}(d_1, d_2, \dots, d_n). \quad (5.82)$$

Den ersten Teilschritt der Transformation $\mathbf{A}' = \mathbf{U}^T \mathbf{A}$ mit der Darstellung $\mathbf{A}' = \mathbf{D}'\tilde{\mathbf{A}}'\mathbf{D}$, wo die rechts stehende Diagonalmatrix \mathbf{D} diejenige aus der Faktorisierung von \mathbf{A} ist, können wir von oben unverändert übernehmen. Denn die Diagonalelemente von \mathbf{D} treten in den zu (5.74) analogen Formeln beidseitig auf und kürzen sich deshalb weg. Für den zweiten Teilschritt erhalten wir aus (5.13) und (5.82)

$$a''_{ip} = d'_i \tilde{a}_{ip}'' d'_p = c a'_{ip} - s a'_{iq} = c d'_i \tilde{a}_{ip}' d_p - s d'_i \tilde{a}_{iq}' d_q$$

$$a''_{iq} = d'_i \tilde{a}_{iq}'' d'_q = s a'_{ip} + c a'_{iq} = s d'_i \tilde{a}_{ip}' d_p + c d'_i \tilde{a}_{iq}' d_q$$

Hier kürzen sich für jeden Index i die Diagonalelemente d'_i weg, so dass sich daraus für die beiden Fälle a) und b) die zu (5.76) und (5.77) analogen Formeln ergeben.

$$\boxed{\begin{aligned} \text{Fall a)} \quad \tilde{a}_{ip}'' &= \tilde{a}_{ip}' - \left(\frac{sd_q}{cd_p} \right) \tilde{a}_{iq}' = \tilde{a}_{ip}' + f_1 \tilde{a}_{iq}' \\ \tilde{a}_{iq}'' &= \left(\frac{sd_p}{cd_q} \right) \tilde{a}_{ip}' + \tilde{a}_{iq}' = -f_2 \tilde{a}_{ip}' + \tilde{a}_{iq}' \\ \text{Fall b)} \quad \tilde{a}_{ip}'' &= \left(\frac{cd_p}{sd_q} \right) \tilde{a}_{ip}' - \tilde{a}_{iq}' = -f_1 \tilde{a}_{ip}' - \tilde{a}_{iq}' \\ \tilde{a}_{iq}'' &= \tilde{a}_{ip}' + \left(\frac{cd_q}{sd_p} \right) \tilde{a}_{iq}' = \tilde{a}_{ip}' - f_2 \tilde{a}_{iq}' \end{aligned}} \quad (5.83)$$

Die schnelle Version der Givens-Rotationen wenden wir an, um eine unsymmetrische Matrix \mathbf{A} auf Hessenberg-Form zu transformieren. Die Behandlung des Elementes a_{ij} mit $i \geq j+2$ der j -ten Spalte benötigt die aufeinander folgende Berechnung der Werte f_1 und f_2 nach (5.79) und von $T^2 = f_1 f_2$ (vier wesentliche Operationen). Nach einer Fallunterscheidung sind allenfalls die Kehrwerte dieser drei Werte zu bilden (drei wesentliche Operationen).

Die Berechnung von c^2 oder s^2 und die Nachführung der beiden Diagonalelemente nach (5.80) erfordert drei Operationen. Bis zu dieser Stelle beträgt die Zahl der multiplikativen Rechenoperationen entweder 7 oder 10. Unter der Annahme, die beiden Fälle seien etwa gleich häufig, setzen wir die Zahl der Operationen für das Folgende mit 9 fest. Eine weitere Multiplikation tritt auf zur Berechnung von $\tilde{a}'_{j+1,j}$ nach (5.76) oder (5.77), und die Zeilen- und Spaltenoperationen benötigen zusammen $2(n - j + n)$ Multiplikationen. Für die Elimination der Elemente a_{ij} der j -ten Spalte ist der Aufwand $(n - j - 1)(4n - 2j + 10)$, und nach Summation über j von 1 bis $(n - 2)$ ergibt sich so die Zahl von $\frac{5}{3}n^3 - \frac{35}{3}n + 10$ Operationen.

Aus der faktorisierten Darstellung erhalten wir die gesuchte Hessenberg-Matrix \mathbf{H} bei $(n^2 + 3n - 2)/2$ von null verschiedenen Matrixelementen mit weiteren $(n^2 + 3n - 2)$ Multiplikationen und n Quadratwurzeln. Der totale Rechenaufwand zur Transformation von \mathbf{A} auf Hessenberg-Form mit den schnellen Givens-Rotationen beträgt damit

$$Z_{\text{HessSG}} = \frac{5}{3}n^3 + n^2 - \frac{26}{3}n + 8. \quad (5.84)$$

Im Vergleich zu (5.62) reduziert sich die Zahl der Multiplikationen für große Ordnungen n tatsächlich auf die Hälfte, und die Anzahl der Quadratwurzeln sinkt von $N^* = (n - 1)(n - 2)/2$ auf n .

Die schnelle Givens-Transformation hat den kleinen Nachteil, dass die Information über die ausgeführten Rotationen nicht mehr in je einer einzigen Zahl ϱ (5.64) zusammengefasst werden kann, vielmehr sind jetzt zwei Werte nötig.

Obwohl die auf die skalierten Matrizen $\tilde{\mathbf{A}}$ angewandten Transformationen nicht mehr orthogonal sind, zeigt eine Analyse der Fehlerfortpflanzung, dass die schnelle Version der Givens-Rotationen im Wesentlichen die gleichen guten Eigenschaften hat wie die normale Givens-Transformation [Par 98, Rat 82].

Die algorithmische Beschreibung der schnellen Givens-Transformation ist weitgehend analog zu (5.65). Die Bestimmung der Faktoren f_1 und f_2 und die problemgerechte Wahl und Ausführung der Fälle a) oder b) erfordern entsprechende Erweiterungen.

Beispiel 5.5. Die Matrix \mathbf{A} (5.66) der Ordnung $n = 6$ soll mit Hilfe der schnellen Givens-Transformation auf Hessenberg-Form transformiert werden. Zur Elimination von a_{31} ist wegen $\mathbf{D} = \mathbf{I}$ gemäß (5.79) $f_1 = f_2 = 6$, und da $T^2 = f_1 f_2 = 36 > 1$ ist, liegt Fall a) vor, d.h. es gelten $f_1 = f_2 = 1/6$ und $c^2 \doteq 0.97297297$. Nach der betreffenden Transformation ist

$$\tilde{\mathbf{A}}_1 \doteq \begin{pmatrix} 7.00000 & 3.66667 & 3.50000 & -11.00000 & -9.00000 & -2.00000 \\ -6.16667 & 1.72222 & -5.08333 & 7.33333 & 2.50000 & 12.16667 \\ 0 & -9.19444 & 4.44444 & 0.83333 & 8.83333 & -1.00000 \\ -8.00000 & -0.16667 & -1.00000 & 5.00000 & 0 & 8.00000 \\ -4.00000 & 2.16667 & -5.50000 & 7.00000 & 2.00000 & 10.00000 \\ 6.00000 & 1.66667 & 3.83333 & -11.00000 & -7.00000 & -1.00000 \end{pmatrix}$$

und $\mathbf{D}_1^2 \doteq \text{diag}(1, 0.97297297, 0.97297297, 1, 1, 1)$. Die Elimination von $\tilde{a}_{41}^{(1)}$ erfolgt mit Multiplikatoren von Fall b), nämlich $f_1 = 0.75$ und $f_2 \doteq 0.770833$, und somit sind $T^2 = 0.578125$ und

$s^2 \doteq 0.63366337$. Die transformierte Matrix ist

$$\tilde{\mathbf{A}}_2 \doteq \begin{pmatrix} 7.00000 & 8.25000 & 3.50000 & 12.14583 & -9.00000 & -2.00000 \\ 12.62500 & 11.34375 & 4.81250 & 6.96875 & -1.87500 & -17.12500 \\ 0 & 6.06250 & 4.44444 & -9.83681 & 8.83333 & -1.00000 \\ 0 & -4.86719 & -4.31250 & -0.83116 & 2.50000 & 6.00000 \\ -4.00000 & -8.62500 & -5.50000 & -3.22917 & 2.00000 & 10.00000 \\ 6.00000 & 9.75000 & 3.83333 & 10.14583 & -7.00000 & -1.00000 \end{pmatrix}$$

mit $\mathbf{D}_2^2 \doteq \text{diag}(1, 0.633663, 0.972973, 0.616537, 1, 1)$. Nach zehn Transformationsschritten erhält man die gesuchte Matrix in faktorisierter Form mit

$$\tilde{\mathbf{A}}_{10} \doteq \begin{pmatrix} 7.00000 & 11.25000 & 6.84385 & -0.19611 & -33.40105 & -18.47911 \\ 19.12500 & 9.87500 & 34.53504 & -2.61995 & -61.19928 & -8.69770 \\ 0 & -13.03649 & 3.39400 & -0.93493 & 18.24220 & 8.91338 \\ 0 & 0 & -14.21384 & 5.74463 & 17.77116 & -18.70622 \\ 0 & 0 & 0 & -5.44254 & -38.91688 & -94.71419 \\ 0 & 0 & 0 & 0 & 12.20070 & 28.95913 \end{pmatrix}$$

und $\mathbf{D}_{10}^2 \doteq \text{diag}(1, 0.418301, 0.721204, 0.507448, 0.072850, 0.184450)$. Daraus ergibt sich die Hessenberg-Matrix $\mathbf{H} = \mathbf{D}_{10} \tilde{\mathbf{A}}_{10} \mathbf{D}_{10}$, welche im Wesentlichen mit (5.67) übereinstimmt. Die beiden resultierenden Matrizen unterscheiden sich nur dadurch, dass die Vorzeichen der Elemente einer Zeile und der entsprechenden Spalte verschieden sein können. \triangle

5.5 QR-Algorithmus

Die numerisch zuverlässigste Methode zur Berechnung der Eigenwerte einer Hessenberg-Matrix \mathbf{H} oder einer symmetrischen, tridiagonalen Matrix \mathbf{J} besteht darin, eine Folge von orthogonal-ähnlichen Matrizen zu bilden, die gegen eine Grenzmatrix konvergieren, welche die gesuchten Eigenwerte liefert.

5.5.1 Grundlagen zur QR-Transformation

Zur Begründung und anschließenden praktischen Durchführung des Verfahrens stellen wir einige Tatsachen zusammen.

Satz 5.7. Jede quadratische Matrix \mathbf{A} lässt sich als Produkt einer orthogonalen Matrix \mathbf{Q} und einer Rechtsdreiecksmatrix \mathbf{R} in der Form

$$\boxed{\mathbf{A} = \mathbf{QR}} \tag{5.85}$$

darstellen. Man bezeichnet die Faktorisierung (5.85) als QR-Zerlegung der Matrix \mathbf{A} .

Beweis. Die Existenz der QR-Zerlegung zeigen wir auf konstruktive Art. Zu diesem Zweck wird die Matrix \mathbf{A} sukzessive mit geeigneten zu wählenden Rotationsmatrizen $\mathbf{U}^T(p, q; \varphi)$

von links so multipliziert, dass die Matrixelemente unterhalb der Diagonale fortlaufend eliminiert werden. Die Elimination erfolgt etwa spaltenweise in der Reihenfolge

$$a_{21}, a_{31}, \dots, a_{n1}, a_{32}, a_{42}, \dots, a_{n2}, a_{43}, \dots, a_{n,n-1} \quad (5.86)$$

mit (p, q) -Rotationsmatrizen und den entsprechenden Rotationsindexpaaren

$$(1, 2), (1, 3), \dots, (1, n), (2, 3), (2, 4), \dots, (2, n), (3, 4), \dots, (n-1, n). \quad (5.87)$$

Zur Elimination des Matrixelementes $a_{ij}^{(k-1)}$ in $\mathbf{A}^{(k-1)}$ wird eine (j, i) -Rotationsmatrix $\mathbf{U}_k^T = \mathbf{U}^T(j, i; \varphi_k)$ angewandt zur Bildung von

$$\mathbf{A}^{(k)} = \mathbf{U}_k^T \mathbf{A}^{(k-1)}, \quad \mathbf{A}^{(0)} = \mathbf{A}, \quad k = 1, 2, \dots, \frac{1}{2}n(n-1). \quad (5.88)$$

Bei dieser Multiplikation werden nach Abschnitt 5.1.4 nur die j -te und i -te Zeile linear kombiniert. Insbesondere soll nach (5.12)

$$a_{ij}^{(k)} = a_{jj}^{(k-1)} \sin \varphi_k + a_{ij}^{(k-1)} \cos \varphi_k = 0 \quad (5.89)$$

sein, woraus sich die Werte $\cos \varphi_k$ und $\sin \varphi_k$ analog zu (5.61) bestimmen lassen. Ist $a_{ij}^{(k-1)} = 0$, so erfolgt selbstverständlich keine Multiplikation, und es ist $\mathbf{U}_k = \mathbf{I}$ zu setzen.

Nun bleibt zu verifizieren, dass die in (5.88) erzeugte Folge von Matrizen $\mathbf{A}^{(k)}$ für $N = n(n-1)/2$ eine Rechtsdreiecksmatrix $\mathbf{A}^{(N)} = \mathbf{R}$ liefert. Die ersten $(n-1)$ Multiplikationen eliminieren in offensichtlicher Weise die Elemente a_{i1} mit $2 \leq i \leq n$. Wir nehmen jetzt an, dass die ersten $(j-1)$ Spalten bereits die gewünschte Form aufweisen. Die Elimination irgendeines Elementes a_{ij} der j -ten Spalte mit $i > j$ geschieht durch Linksmultiplikation mit einer (j, i) -Rotationsmatrix. In den ersten $(j-1)$ Spalten werden nur Matrixelemente linear kombiniert, die nach Annahme verschwinden. Die bereits erzeugten Nullelemente bleiben tatsächlich erhalten, und es gilt

$$\mathbf{U}_N^T \mathbf{U}_{N-1}^T \dots \mathbf{U}_2^T \mathbf{U}_1^T \mathbf{A}^{(0)} = \mathbf{R}.$$

Die Matrix $\mathbf{Q}^T := \mathbf{U}_N^T \mathbf{U}_{N-1}^T \dots \mathbf{U}_2^T \mathbf{U}_1^T$ ist als Produkt von orthogonalen Matrizen orthogonal, und somit ist die Existenz der QR-Zerlegung $\mathbf{A}^{(0)} = \mathbf{A} = \mathbf{QR}$ nachgewiesen. \square

Eine unmittelbare Folge des Beweises von Satz 5.7 ist der

Satz 5.8. *Die QR-Zerlegung einer Hessenberg-Matrix \mathbf{H} oder einer tridiagonalen Matrix \mathbf{J} der Ordnung n ist mit $(n-1)$ Rotationsmatrizen durchführbar.*

Beweis. Ist die gegebene Matrix \mathbf{A} entweder von Hessenberg-Form oder tridiagonal, erfolgt die Elimination des einzigen, eventuell von null verschiedenen Elementes a_{21} der ersten Spalte unterhalb der Diagonale durch Linksmultiplikation mit der Matrix $\mathbf{U}^T(1, 2; \varphi_1) = \mathbf{U}_1^T$. Dadurch bleibt die Struktur der Matrix \mathbf{A} für die $(n-1)$ -reihige Untermatrix erhalten, welche durch Streichen der ersten Zeile und Spalte von $\mathbf{A}^{(1)} = \mathbf{U}_1^T \mathbf{A}$ entsteht. Diese

Feststellung gilt dann zwangsläufig auch für die folgenden Eliminationsschritte. Die Linksmultiplikation von \mathbf{A} mit der Folge von $(n - 1)$ Rotationsmatrizen mit den Indexpaaren $(1, 2), (2, 3), \dots, (n - 1, n)$ leistet die QR -Zerlegung $\mathbf{A} = \mathbf{Q}\mathbf{R}$ mit $\mathbf{Q} := \mathbf{U}_1\mathbf{U}_2 \dots \mathbf{U}_{n-1}$. Im Fall einer tridiagonalen Matrix \mathbf{J} enthält die Rechtsdreiecksmatrix \mathbf{R} in den beiden der Diagonale benachbarten oberen Nebendiagonalen im Allgemeinen von null verschiedene Elemente, denn jede Linksmultiplikation mit $\mathbf{U}^T(i, i + 1; \varphi_i)$ erzeugt einen in der Regel nichtverschwindenden Wert an der Stelle $(i, i + 2)$ für $i = 1, 2, \dots, n - 2$. \square

Die Aussage des Satzes 5.8 ist für den nachfolgend beschriebenen Algorithmus von entscheidender Bedeutung, da die Berechnung der QR -Zerlegung einer Hessenberg-Matrix wesentlich weniger aufwändig ist als diejenige einer voll besetzten Matrix. Eine QR -Transformation entsteht, wenn der QR -Zerlegung von \mathbf{A} eine Multiplikation der Matrizen \mathbf{R} und \mathbf{Q} folgt:

$$\boxed{\text{QR-Transformation: } \mathbf{A} = \mathbf{Q}\mathbf{R} \rightarrow \mathbf{A}' = \mathbf{R}\mathbf{Q}} \quad (5.90)$$

Satz 5.9. *Die Matrix $\mathbf{A}' = \mathbf{R}\mathbf{Q}$ nach (5.90) ist orthogonal-ähnlich zur Matrix $\mathbf{A} = \mathbf{Q}\mathbf{R}$.*

Beweis. Da die Matrix \mathbf{Q} der QR -Zerlegung von \mathbf{A} orthogonal und somit regulär ist, existiert ihre Inverse $\mathbf{Q}^{-1} = \mathbf{Q}^T$. Aus (5.90) folgt die orthogonale Ähnlichkeit der Matrizen \mathbf{A}' und \mathbf{A} gemäß

$$\mathbf{R} = \mathbf{Q}^{-1}\mathbf{A} \quad \text{und} \quad \mathbf{A}' = \mathbf{R}\mathbf{Q} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q}. \quad \square$$

Satz 5.10. *Die Hessenberg-Form einer Matrix \mathbf{H} der Ordnung n bleibt bei einer QR -Transformation erhalten.*

Beweis. Nach Satz 5.8 ist die Matrix \mathbf{Q} der QR -Zerlegung einer Hessenberg-Matrix \mathbf{H} der Ordnung n gegeben als Produkt der $(n - 1)$ Rotationsmatrizen

$$\mathbf{Q} = \mathbf{U}(1, 2; \varphi_1)\mathbf{U}(2, 3; \varphi_2) \dots \mathbf{U}(n - 1, n; \varphi_{n-1}). \quad (5.91)$$

Die QR -Transformierte $\mathbf{H}' = \mathbf{R}\mathbf{Q}$ ergibt sich mit (5.91) durch fortgesetzte Multiplikation der Rechtsdreiecksmatrix \mathbf{R} von rechts mit den Rotationsmatrizen. Die Matrix $\mathbf{R}\mathbf{U}(1, 2; \varphi_1)$ ist keine Rechtsdreiecksmatrix mehr, denn die Linearkombinationen der ersten und zweiten Spalten erzeugt genau an der Stelle $(2, 1)$ ein von null verschiedenes Matrixelement unterhalb der Diagonale. Die darauffolgende Multiplikation mit $\mathbf{U}(2, 3; \varphi_2)$ liefert durch die Spaltenoperation genau ein weiteres von null verschiedenes Element an der Position $(3, 2)$. Allgemein erzeugt die Rechtsmultiplikation mit $\mathbf{U}(k, k + 1; \varphi_k)$ an der Stelle $(k + 1, k)$ ein Matrixelement ungleich null, ohne dabei die vorhergehenden Spalten zu verändern. \mathbf{H}' ist somit eine Hessenberg-Matrix. \square

Der Rechenaufwand einer QR -Transformation für eine Hessenberg-Matrix \mathbf{H} der Ordnung n setzt sich zusammen aus demjenigen für die Bildung der Rechtsdreiecksmatrix \mathbf{R} und demjenigen für die Berechnung von \mathbf{H}' . Der allgemeine j -te Schritt des ersten Teils benötigt unter Verwendung von (5.61) und (5.12) insgesamt $4 + 4(n - j)$ Multiplikationen und eine Quadratwurzel. Die Matrixmultiplikation mit $\mathbf{U}(j, j + 1; \varphi_j)$ im zweiten Teil erfordert

$4j + 2$ Multiplikationen, falls man berücksichtigt, dass das Diagonalelement und das neu entstehende Nebendiagonalelement nur je eine Operation benötigen. Nach Summation über j von 1 bis $(n - 1)$ beträgt der Aufwand

$$Z_{\text{QR-Hess}} = (4n + 6)(n - 1) = 4n^2 + 2n - 6 \quad (5.92)$$

Multiplikationen und $(n - 1)$ Quadratwurzeln.

Satz 5.11. *Die QR-Transformierte einer symmetrischen tridiagonalen Matrix ist wieder symmetrisch und tridiagonal.*

Beweis. Da nach Satz 5.7 die Matrix \mathbf{A}' orthogonal-ähnlich zu \mathbf{A} ist, bleibt die Symmetrie wegen $\mathbf{A}'^T = (\mathbf{Q}^{-1}\mathbf{A}\mathbf{Q})^T = \mathbf{Q}^T\mathbf{A}^T\mathbf{Q}^{-1T} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q} = \mathbf{A}'$ erhalten. Weiter ist eine tridiagonale Matrix eine spezielle Hessenberg-Matrix, und folglich ist \mathbf{A}' nach Satz 5.10 von Hessenberg-Form. Unter Berücksichtigung der Symmetrie muss \mathbf{A}' demzufolge tridiagonal sein. \square

Als motivierende Grundlage des QR-Algorithmus dient der folgende Satz von Schur [Sch 09]. Da wir die Eigenwerte von reellen Matrizen berechnen wollen, wird der Satz nur in seiner reellen Form formuliert.

Satz 5.12. *Zu jeder reellen Matrix \mathbf{A} der Ordnung n existiert eine orthogonale Matrix \mathbf{U} der Ordnung n , so dass die zu \mathbf{A} ähnliche Matrix $\mathbf{R} := \mathbf{U}^{-1}\mathbf{A}\mathbf{U}$ die Quasidreiecksgestalt*

$$\mathbf{R} := \mathbf{U}^{-1}\mathbf{A}\mathbf{U} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \mathbf{R}_{13} & \cdots & \mathbf{R}_{1m} \\ \mathbf{0} & \mathbf{R}_{22} & \mathbf{R}_{23} & \cdots & \mathbf{R}_{2m} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{33} & \cdots & \mathbf{R}_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{mm} \end{pmatrix} \quad (5.93)$$

hat. Die Matrizen \mathbf{R}_{ii} , ($i = 1, 2, \dots, m$) besitzen entweder die Ordnung eins oder die Ordnung zwei und haben im letzten Fall ein Paar von konjugiert komplexen Eigenwerten.

Beweis. Der Nachweis der Existenz einer orthogonalen Matrix \mathbf{U} erfolgt in drei Teilen.

1) Es sei λ ein reeller Eigenwert von \mathbf{A} und \mathbf{x} ein zugehöriger, normierter Eigenvektor mit $\|\mathbf{x}\|_2 = 1$. Zu \mathbf{x} gibt es im \mathbb{R}^n weitere $(n - 1)$ normierte, paarweise und zu \mathbf{x} orthogonale Vektoren, welche die Spalten einer orthogonalen Matrix \mathbf{U}_1 bilden sollen. Der Eigenvektor \mathbf{x} stehe in der ersten Spalte von $\mathbf{U}_1 = (\mathbf{x}, \tilde{\mathbf{U}}_1)$, wo $\tilde{\mathbf{U}}_1 \in \mathbb{R}^{n, (n-1)}$ eine Matrix mit $(n - 1)$ orthonormierten Vektoren darstellt. Dann gilt $\mathbf{AU}_1 = (\lambda\mathbf{x}, \mathbf{A}\tilde{\mathbf{U}}_1)$ und weiter

$$\begin{aligned} \mathbf{U}_1^{-1}\mathbf{AU}_1 &= \mathbf{U}_1^T\mathbf{AU}_1 = \begin{pmatrix} \mathbf{x}^T \\ \tilde{\mathbf{U}}_1^T \end{pmatrix} (\lambda\mathbf{x}, \mathbf{A}\tilde{\mathbf{U}}_1) \\ &= \begin{pmatrix} \lambda & \cdots & \mathbf{x}^T \mathbf{A} \tilde{\mathbf{U}}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \vdots & \tilde{\mathbf{U}}_1^T \mathbf{A} \tilde{\mathbf{U}}_1 \end{pmatrix} =: \mathbf{A}_1. \end{aligned} \quad (5.94)$$

In der zu \mathbf{A} orthogonal-ähnlichen Matrix \mathbf{A}_1 (5.94) steht der Eigenwert λ in der linken oberen Ecke. In der ersten Spalte steht darunter der Nullvektor auf Grund der Orthogonalität von \mathbf{x} zu den Spalten von $\tilde{\mathbf{U}}_1$, und $\tilde{\mathbf{U}}_1^T \mathbf{A} \tilde{\mathbf{U}}_1$ ist eine reelle Matrix der Ordnung $(n - 1)$.

2) Ist $\lambda = \alpha + i\beta$ mit $\beta \neq 0$ ein komplexer Eigenwert von \mathbf{A} mit dem zugehörigen Eigenvektor $\mathbf{x} = \mathbf{u} + i\mathbf{v}$, dann ist notwendigerweise $\bar{\lambda} = \alpha - i\beta$ der dazu konjugiert komplexe Eigenwert mit dem zugehörigen konjugiert komplexen Eigenvektor $\bar{\mathbf{x}} = \mathbf{u} - i\mathbf{v}$. Es gilt also

$$\mathbf{A}(\mathbf{u} \pm i\mathbf{v}) = (\alpha \pm i\beta)(\mathbf{u} \pm i\mathbf{v}),$$

und damit für Real- und Imaginärteil

$$\mathbf{A}\mathbf{u} = \alpha\mathbf{u} - \beta\mathbf{v}, \quad \mathbf{A}\mathbf{v} = \beta\mathbf{u} + \alpha\mathbf{v}. \quad (5.95)$$

Weil Eigenvektoren zu verschiedenen Eigenwerten linear unabhängig sind, trifft dies für \mathbf{x} und $\bar{\mathbf{x}}$ zu, so dass auch die beiden Vektoren \mathbf{u} und \mathbf{v} linear unabhängig sind und folglich einen zweidimensionalen Unterraum im \mathbb{R}^n aufspannen. Bildet man mit \mathbf{u} und \mathbf{v} die Matrix $\mathbf{Y} := (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^{n,2}$, so gilt nach (5.95)

$$\mathbf{AY} = \mathbf{Y} \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} =: \mathbf{Y}\Gamma. \quad (5.96)$$

In dem von \mathbf{u} und \mathbf{v} aufgespannten Unterraum gibt es zwei orthonormierte Vektoren \mathbf{x}_1 und \mathbf{x}_2 , welche wir zur Matrix $\mathbf{X} \in \mathbb{R}^{n,2}$ zusammenfassen, und es besteht eine Beziehung

$$\mathbf{Y} = \mathbf{XC} \quad \text{mit } \mathbf{C} \in \mathbb{R}^{2,2} \text{ regulär.}$$

Die Matrizengleichung (5.96) geht damit über in

$$\mathbf{AXC} = \mathbf{X}\mathbf{C}\Gamma \quad \text{oder} \quad \mathbf{AX} = \mathbf{X}\mathbf{C}\Gamma\mathbf{C}^{-1} =: \mathbf{XS}. \quad (5.97)$$

Die Matrix $\mathbf{S} = \mathbf{C}\Gamma\mathbf{C}^{-1} \in \mathbb{R}^{2,2}$ ist ähnlich zu Γ und besitzt folglich das Paar von konjugiert komplexen Eigenwerten $\lambda = \alpha + i\beta$ und $\bar{\lambda} = \alpha - i\beta$.

Im \mathbb{R}^n gibt es $(n-2)$ weitere orthonormierte Vektoren, die zu \mathbf{x}_1 und \mathbf{x}_2 orthogonal sind. Mit diesen n orthonormierten Vektoren bilden wir die orthogonale Matrix $\mathbf{U}_2 := (\mathbf{x}_1, \mathbf{x}_2, \tilde{\mathbf{U}}_2) = (\mathbf{X}, \tilde{\mathbf{U}}_2)$, wo $\tilde{\mathbf{U}}_2 \in \mathbb{R}^{n,n-2}$ eine Matrix mit orthonormierten Spaltenvektoren ist. Wegen (5.97) gilt dann

$$\begin{aligned} \mathbf{U}_2^{-1} \mathbf{AU}_2 &= \mathbf{U}_2^T \mathbf{AU}_2 = \begin{pmatrix} \mathbf{X}^T \\ \tilde{\mathbf{U}}_2^T \end{pmatrix} \mathbf{A}(\mathbf{X}, \tilde{\mathbf{U}}_2) \\ &= \begin{pmatrix} \mathbf{S} & \vdots & \mathbf{X}^T \mathbf{A} \tilde{\mathbf{U}}_2 \\ \mathbf{0} & \vdots & \tilde{\mathbf{U}}_2^T \mathbf{A} \tilde{\mathbf{U}}_2 \end{pmatrix} =: \mathbf{A}_2. \end{aligned} \quad (5.98)$$

In der zu \mathbf{A} orthogonal-ähnlichen Matrix \mathbf{A}_2 (5.98) steht links oben die Matrix \mathbf{S} . Darunter ist $\tilde{\mathbf{U}}_2^T \mathbf{A} \mathbf{X} = \tilde{\mathbf{U}}_2^T \mathbf{X} \mathbf{S} = \mathbf{0} \in \mathbb{R}^{(n-2),2}$ eine Nullmatrix wegen der Orthogonalität der Vektoren \mathbf{x}_1 und \mathbf{x}_2 zu den Spaltenvektoren von $\tilde{\mathbf{U}}_2$. Schließlich ist $\tilde{\mathbf{U}}_2^T \mathbf{A} \tilde{\mathbf{U}}_2$ eine reelle Matrix der Ordnung $(n-2)$.

3) Auf Grund der Gestalt der Matrizen \mathbf{A}_1 (5.94) und \mathbf{A}_2 (5.98) besteht die Gesamtheit der Eigenwerte von \mathbf{A} im ersten Fall aus λ und den Eigenwerten von $\tilde{\mathbf{U}}_1^T \mathbf{A} \tilde{\mathbf{U}}_1$ und im zweiten Fall aus dem konjugiert komplexen Eigenwertpaar von \mathbf{S} und den Eigenwerten von $\tilde{\mathbf{U}}_2^T \mathbf{A} \tilde{\mathbf{U}}_2$. Die orthogonal-ähnliche Transformation kann analog auf die Untermatrix $\tilde{\mathbf{A}}_i := \tilde{\mathbf{U}}_i^T \mathbf{A} \tilde{\mathbf{U}}_i$ der Ordnung $(n-i)$ mit $i=1$ oder $i=2$ mit einem weiteren Eigenwert oder Eigenwertpaar angewandt werden. Ist $\tilde{\mathbf{V}}_i \in \mathbb{R}^{(n-i),(n-i)}$ die orthogonale Matrix, welche $\tilde{\mathbf{A}}_i$ in die Form von (5.94) oder (5.98) transformiert, so ist

$$\mathbf{V} := \begin{pmatrix} \mathbf{I}_i & & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & & \tilde{\mathbf{V}}_i \end{pmatrix} \in \mathbb{R}^{n,n}$$

diejenige orthogonale Matrix, welche mit $\mathbf{V}^T \mathbf{A}_i \mathbf{V}$ die gegebene Matrix \mathbf{A} einen Schritt weiter auf Quasidreiecksgestalt transformiert. Die konsequente Fortsetzung der Transformation liefert die Aussage des Satzes, wobei \mathbf{U} als Produkt der orthogonalen Matrizen gegeben ist. \square

Der Satz von Schur gilt unabhängig von der Vielfachheit der Eigenwerte der Matrix \mathbf{A} . Die Eigenwerte von \mathbf{A} sind aus den Untermatrizen \mathbf{R}_{ii} der Quasidreiecksmatrix \mathbf{R} entweder direkt ablesbar oder leicht berechenbar. Satz 5.12 enthält eine reine Existenzaussage, und der Beweis ist leider nicht konstruktiv, da er von der Existenz eines Eigenwertes und eines zugehörigen Eigenvektors Gebrauch macht, die ja erst zu bestimmen sind.

Um Fehlinterpretationen vorzubeugen, sei darauf hingewiesen, dass die Spalten der orthogonalen Matrix \mathbf{U} in (5.93) im Allgemeinen nichts mit den Eigenvektoren von \mathbf{A} zu tun haben. Für einen komplexen Eigenwert von \mathbf{A} ist dies ganz offensichtlich, da die aus (5.93) hervorgehende Matrizengleichung $\mathbf{AU} = \mathbf{UR}$ reelle Matrizen enthält.

5.5.2 Praktische Durchführung, reelle Eigenwerte

Durch sukzessive Anwendung der QR -Transformation (5.90) bilden wir nach Satz 5.9 eine Folge von orthogonal-ähnlichen Matrizen mit dem Ziel, die Aussage des Satzes von Schur konstruktiv zu realisieren. Um den Rechenaufwand zu reduzieren und wegen Satz 5.10 wird die QR -Transformation nur auf eine Hessenberg-Matrix $\mathbf{H} = \mathbf{H}_1$ angewandt und die Folge der ähnlichen Matrizen gemäß folgender Rechenvorschrift konstruiert.

$$\boxed{\mathbf{H}_k = \mathbf{Q}_k \mathbf{R}_k, \quad \mathbf{H}_{k+1} = \mathbf{R}_k \mathbf{Q}_k, \quad k = 1, 2, \dots} \quad (5.99)$$

Mit (5.99) ist der einfache QR -Algorithmus von Francis [Fra 62] erklärt. Für die Folge der Matrizen \mathbf{H}_k kann folgende Konvergenzeigenschaft gezeigt werden [Fra 62, Par 98, Wil 88].

Satz 5.13. Es seien λ_i die Eigenwerte von \mathbf{H} mit der Eigenschaft $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$. Weiter seien \mathbf{x}_i die Eigenvektoren von \mathbf{H} , die als Spalten in der regulären Matrix $\mathbf{X} \in \mathbb{R}^{n,n}$ zusammengefasst seien. Falls für \mathbf{X}^{-1} die LR-Zerlegung existiert, dann konvergieren die Matrizen \mathbf{H}_k für $k \rightarrow \infty$ gegen eine Rechtsdreiecksmatrix, und es gilt $\lim_{k \rightarrow \infty} h_{ii}^{(k)} = \lambda_i$, $i = 1, 2, \dots, n$.

Hat die Matrix \mathbf{H} Paare von konjugiert komplexen Eigenwerten, derart dass ihre Beträge und die Beträge der reellen Eigenwerte paarweise verschieden sind, und existiert die (komplexe) LR-Zerlegung von \mathbf{X}^{-1} , dann konvergieren die Matrizen \mathbf{H}_k gegen eine Quasidreiecksmatrix (5.93).

Satz 5.13 garantiert zwar unter den einschränkenden Voraussetzungen die Konvergenz der Matrizenfolge \mathbf{H}_k (5.99) gegen eine Quasidreiecksmatrix, doch kann die Konvergenz derjenigen Matrixelemente $h_{i+1,i}^{(k)}$, die überhaupt gegen null konvergieren, sehr langsam sein. Somit kann der einfache QR-Algorithmus wegen der großen Zahl von Iterationsschritten zu aufwändig sein. Der QR-Algorithmus steht in enger Beziehung zur inversen Vektoriteration, siehe Abschnitt 5.3.2, und eine darauf beruhende Analyse des Konvergenzverhaltens der Subdiagonalelemente zeigt, dass asymptotisch für hinreichend großes k gilt

$$|h_{i+1,i}^{(k)}| \approx \left| \frac{\lambda_{i+1}}{\lambda_i} \right|^k, \quad i = 1, 2, \dots, n-1. \quad (5.100)$$

Die Beträge der Subdiagonalelemente $h_{i+1,i}^{(k)}$ konvergieren für $|\lambda_{i+1}/\lambda_i| < 1$ wie geometrische Folgen gegen null. Die lineare Konvergenz wird bestimmt durch den Quotienten der Beträge aufeinanderfolgender Eigenwerte, welcher natürlich beliebig nahe bei Eins sein kann. Für ein konjugiert komplexes Paar $\lambda_i, \lambda_{i+1} = \bar{\lambda}_i$ kann $h_{i+1,i}^{(k)}$ wegen (5.100) nicht gegen null konvergieren.

Die Konvergenzaussage (5.100) liefert aber den Hinweis, dass die lineare Konvergenz für bestimmte Subdiagonalelemente durch eine geeignete Spektralverschiebung wesentlich verbessert werden kann. Wir wollen vorerst den Fall betrachten, dass \mathbf{H} nur reelle Eigenwerte hat und das Vorgehen im Fall von komplexen Eigenwerten später behandeln. Mit $\sigma \in \mathbb{R}$ hat die Matrix $\mathbf{H} - \sigma \mathbf{I}$ die um σ verschobenen Eigenwerte $\lambda_i - \sigma$, $i = 1, 2, \dots, n$. Sie seien jetzt so indiziert, dass $|\lambda_1 - \sigma| > |\lambda_2 - \sigma| > \dots > |\lambda_n - \sigma|$ gilt. Für die Matrizenfolge $\tilde{\mathbf{H}}_k$ (5.99) mit $\tilde{\mathbf{H}}_1 = \mathbf{H} - \sigma \mathbf{I}$ folgt aus (5.100)

$$|\tilde{h}_{i+1,i}^{(k)}| \approx \left| \frac{\lambda_{i+1} - \sigma}{\lambda_i - \sigma} \right|^k, \quad i = 1, 2, \dots, n-1. \quad (5.101)$$

Ist σ eine gute Näherung für λ_n mit $|\lambda_n - \sigma| \ll |\lambda_i - \sigma|$, $i = 1, 2, \dots, n-1$, dann konvergiert $\tilde{h}_{n,n-1}^{(k)}$ sehr rasch gegen null, und die Matrix $\tilde{\mathbf{H}}_k$ zerfällt nach wenigen Iterationsschritten.

Die Technik der Spektralverschiebung wird nicht nur einmal angewandt, sondern vor Ausführung eines jeden Schrittes des QR-Algorithmus, wobei die Verschiebung σ_k auf Grund der vorliegenden Information in \mathbf{H}_k geeignet gewählt wird. Anstelle von (5.99) definiert man mit

$$\boxed{\mathbf{H}_k - \sigma_k \mathbf{I} = \mathbf{Q}_k \mathbf{R}_k, \quad \mathbf{H}_{k+1} = \mathbf{R}_k \mathbf{Q}_k + \sigma_k \mathbf{I}, \quad k = 1, 2, \dots} \quad (5.102)$$

den *QR-Algorithmus* mit *expliziter Spektralverschiebung*. Da die Spektralverschiebung in \mathbf{H}_k bei der Berechnung von \mathbf{H}_{k+1} wieder rückgängig gemacht wird, sind die Matrizen \mathbf{H}_{k+1} und \mathbf{H}_k orthogonal-ähnlich zueinander. Auf Grund der oben erwähnten Verwandtschaft des *QR*-Algorithmus mit der Vektoriteration ist die Wahl der Verschiebung σ_k gemäß

$$\boxed{\sigma_k = h_{nn}^{(k)}, \quad k = 1, 2, \dots} \quad (5.103)$$

als letztes Diagonalelement von \mathbf{H}_k angezeigt [Ste 83]. Mit dieser Festsetzung von σ_k wird erreicht, dass die Folge der Subdiagonalelemente $h_{n,n-1}^{(k)}$ schließlich *quadratisch* gegen null konvergiert [Ste 83].

Die praktische Durchführung des *QR*-Algorithmus (5.102) ist durch die Überlegungen in den Beweisen der Sätze 5.8 und 5.10 vollkommen vorgezeichnet, falls man zuerst die *QR*-Zerlegung von $\mathbf{H}_k - \sigma_k \mathbf{I}$ mit Hilfe von $(n-1)$ Rotationsmatrizen durchführt und anschließend die Matrix \mathbf{H}_{k+1} bildet. Bei diesem Vorgehen sind die $(n-1)$ Wertepaare $c_i := \cos \varphi_i$ und $s_i := \sin \varphi_i$ der Matrizen $\mathbf{U}(i, i+1; \varphi_i) =: \mathbf{U}_i$ abzuspeichern. Das ist aber nicht nötig, wie die folgende Betrachtung zeigt. Für eine *QR*-Transformation gelten ja

$$\begin{aligned} \mathbf{U}_{n-1}^T & \cdots \mathbf{U}_3^T \mathbf{U}_2^T \mathbf{U}_1^T (\mathbf{H}_k - \sigma_k \mathbf{I}) = \mathbf{Q}_k^T (\mathbf{H}_k - \sigma_k \mathbf{I}) = \mathbf{R}_k \\ \mathbf{H}_{k+1} &= \mathbf{R}_k \mathbf{Q}_k + \sigma_k \mathbf{I} = \mathbf{R}_k \mathbf{U}_1 \mathbf{U}_2 \mathbf{U}_3 \cdots \mathbf{U}_{n-1} + \sigma_k \mathbf{I} \\ &= \mathbf{U}_{n-1}^T \cdots \mathbf{U}_3^T \mathbf{U}_2^T \mathbf{U}_1^T (\mathbf{H}_k - \sigma_k \mathbf{I}) \mathbf{U}_1 \mathbf{U}_2 \mathbf{U}_3 \cdots \mathbf{U}_{n-1} + \sigma_k \mathbf{I} \end{aligned} \quad (5.104)$$

Infolge der Assoziativität der Matrizenmultiplikation können wir die Reihenfolge der Operationen zur Bildung von \mathbf{H}_{k+1} gemäß (5.104) geeignet wählen. Nach Ausführung der beiden Matrizenmultiplikationen $\mathbf{U}_2^T \mathbf{U}_1^T (\mathbf{H}_k - \sigma_k \mathbf{I})$ im Verlauf der *QR*-Zerlegung hat die resultierende Matrix für $n = 6$ die folgende Struktur:

$$\left(\begin{array}{cccccc} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{array} \right)$$

Nun ist zu beachten, dass die weiteren Multiplikationen von links mit $\mathbf{U}_3^T, \mathbf{U}_4^T, \dots$ nur noch die dritte und die folgenden Zeilen betreffen, d.h. dass die ersten beiden Zeilen und damit auch die ersten beiden Spalten in der momentanen Matrix durch die *QR*-Zerlegung nicht mehr verändert werden. Die Multiplikation mit \mathbf{U}_1 von rechts, die ja Linearkombinationen der ersten und zweiten Spalten bewirkt, operiert folglich mit den hier vorhandenen, bereits endgültigen Werten der Matrix \mathbf{R}_k der *QR*-Zerlegung. Sobald die Multiplikation mit \mathbf{U}_3^T erfolgt ist, ist aus dem analogen Grund die Multiplikation mit \mathbf{U}_2 von rechts ausführbar.

Bei dieser gestaffelten Ausführung der *QR*-Transformation (5.104) wird im i -ten Schritt ($2 \leq i \leq n-1$) in der i -ten und $(i+1)$ -ten Zeile die *QR*-Zerlegung weitergeführt, während in der $(i-1)$ -ten und i -ten Spalte bereits im Wesentlichen die Hessenberg-Matrix \mathbf{H}_{k+1} aufgebaut wird. Die Subtraktion von σ_k in der Diagonale von \mathbf{H}_k und die spätere Addition von σ_k zu den Diagonalelementen kann fortlaufend in den Prozess einbezogen werden. Falls wir die Matrix \mathbf{H}_{k+1} am Ort von \mathbf{H}_k aufbauen und die sich laufend verändernde Matrix mit

H bezeichnen, können wir den QR-Schritt (5.104) zu gegebenem Wert von σ algorithmisch in (5.106) formulieren. Es ist τ wieder die Maschinengenauigkeit, siehe (1.7).

Da die Subdiagonalelemente von \mathbf{H}_k gegen null konvergieren, ist ein Kriterium notwendig, um zu entscheiden, wann ein Element $|h_{i+1,i}^{(k)}|$ als genügend klein betrachtet werden darf, um es gleich null zu setzen. Ein sicheres, auch betragskleinen Eigenwerten von \mathbf{H}_1 Rechnung tragendes Kriterium ist

$$|h_{i+1,i}^{(k)}| < \tau \max\{|h_{ii}^{(k)}|, |h_{i+1,i+1}^{(k)}|\}. \quad (5.105)$$

Sobald ein Subdiagonalelement $h_{i+1,i}^{(k)}$ die Bedingung (5.105) erfüllt, zerfällt die Hessenberg-Matrix \mathbf{H}_k . Die Berechnung der Eigenwerte von \mathbf{H}_k ist zurückgeführt auf die Aufgabe, die Eigenwerte der beiden Untermatrizen von Hessenberg-Form zu bestimmen.

QR-Transformation

mit Shift σ und Überspeicherung der \mathbf{H}_k .

$$h_{11} = h_{11} - \sigma$$

für $i = 1, 2, \dots, n$:

falls $i < n$:

$$\text{falls } |h_{ii}| < \tau \times |h_{i+1,i}|:$$

$$w = |h_{i+1,i}|; c = 0; s = \operatorname{sgn}(h_{i+1,i})$$

sonst

$$w = \sqrt{h_{ii}^2 + h_{i+1,i}^2}; c = h_{ii}/w; s = -h_{i+1,i}/w$$

$$h_{ii} = w; h_{i+1,i} = 0; h_{i+1,i+1} = h_{i+1,i+1} - \sigma$$

für $j = i+1, i+2, \dots, n$:

$$g = c \times h_{ij} - s \times h_{i+1,j}$$

$$h_{i+1,j} = s \times h_{ij} + c \times h_{i+1,j}; h_{ij} = g$$

falls $i > 1$:

für $j = 1, 2, \dots, i$:

$$g = \tilde{c} \times h_{j,i-1} - \tilde{s} \times h_{ji}$$

$$h_{ji} = \tilde{s} \times h_{j,i-1} + \tilde{c} \times h_{ji}; h_{j,i-1} = g$$

$$h_{i-1,i-1} = h_{i-1,i-1} + \sigma$$

$$\tilde{c} = c; \tilde{s} = s$$

$$h_{nn} = h_{nn} + \sigma$$

(5.106)

Im betrachteten Fall von reellen Eigenwerten von \mathbf{H}_1 und unter der angewendeten Strategie der Spektralverschiebungen ist es am wahrscheinlichsten, dass das Subdiagonalelement $h_{n,n-1}^{(k)}$ zuerst die Bedingung (5.105) erfüllt. Die Matrix \mathbf{H}_k zerfällt dann in eine Hessenberg-Matrix $\hat{\mathbf{H}}$ der Ordnung $(n-1)$ und eine Matrix der Ordnung Eins, welche notwendigerweise

als Element einen Eigenwert λ enthalten muss; sie hat also die Form

$$\mathbf{H}_k = \begin{pmatrix} \times & \times & \times & \times & \times & \vdots & \times \\ \times & \times & \times & \times & \times & \vdots & \times \\ 0 & \times & \times & \times & \times & \vdots & \times \\ 0 & 0 & \times & \times & \times & \vdots & \times \\ 0 & 0 & 0 & \times & \times & \vdots & \times \\ \dots & \dots & \dots & \dots & \dots & \vdots & \dots \\ 0 & 0 & 0 & 0 & 0 & \vdots & \lambda \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{H}} & \vdots & \hat{\mathbf{h}} \\ \mathbf{0}^T & \vdots & \lambda \end{pmatrix}. \quad (5.107)$$

Sobald die Situation (5.107) eingetreten ist, kann der QR -Algorithmus mit der Untermatrix $\hat{\mathbf{H}} \in \mathbb{R}^{(n-1),(n-1)}$ fortgesetzt werden, die ja die übrigen Eigenwerte besitzt. Darin zeigt sich ein wesentlicher Vorteil des QR -Algorithmus: Mit jedem berechneten Eigenwert reduziert sich die Ordnung der noch weiter zu bearbeitenden Matrix.

Beispiel 5.6. Zur Berechnung von Eigenwerten der Hessenberg-Matrix \mathbf{H} (5.67) aus Beispiel 5.3 wird der QR -Algorithmus (5.106) angewandt. Die Spektralverschiebungen σ_k für die QR -Schritte werden dabei nach der Vorschrift (5.109) des folgenden Abschnittes festgelegt. Die ersten sechs QR -Schritte ergeben folgende Werte.

k	σ_k	$h_{66}^{(k+1)}$	$h_{65}^{(k+1)}$	$h_{55}^{(k+1)}$
1	2.342469183	2.358475293	$-1.65619 \cdot 10^{-1}$	5.351616732
2	2.357470399	2.894439187	$-4.69253 \cdot 10^{-2}$	0.304289831
3	2.780738412	2.984399174	$-7.64577 \cdot 10^{-3}$	1.764372360
4	2.969415975	2.999399624	$-1.04138 \cdot 10^{-4}$	5.056039373
5	2.999618708	2.999999797	$-3.90852 \cdot 10^{-8}$	3.998363523
6	3.000000026	3.000000000	$1.15298 \cdot 10^{-15}$	3.871195470

Man erkennt an den Werten σ_k und $h_{66}^{(k+1)}$ die Konvergenz gegen einen Eigenwert $\lambda_1 = 3$ und das quadratische Konvergenzverhalten von $h_{65}^{(k+1)}$ gegen null. Nach dem sechsten QR -Schritt ist die Bedingung (5.105) mit $\tau = 10^{-12}$ erfüllt. Die Untermatrix $\hat{\mathbf{H}}$ der Ordnung fünf in \mathbf{H}_7 lautet näherungsweise

$$\hat{\mathbf{H}} \doteq \begin{pmatrix} 4.991623 & 5.988209 & -3.490458 & -5.233181 & 1.236387 \\ -5.982847 & 5.021476 & 6.488094 & 7.053759 & -16.317615 \\ 0 & -0.033423 & 0.160555 & 5.243584 & 11.907163 \\ 0 & 0 & -0.870774 & 1.955150 & -1.175364 \\ 0 & 0 & 0 & -0.107481 & 3.871195 \end{pmatrix}.$$

Wir stellen fest, dass die Subdiagonalelemente von $\hat{\mathbf{H}}$ im Vergleich zur Matrix (5.67) betragsmäßig abgenommen haben. Deshalb stellt die Verschiebung σ_7 für den nächsten QR -Schritt bereits eine gute Näherung für einen weiteren reellen Eigenwert dar. Die wichtigsten Werte der drei folgenden QR -Schritte für $\hat{\mathbf{H}}$ sind

k	σ_k	$h_{55}^{(k+1)}$	$h_{54}^{(k+1)}$	$h_{44}^{(k+1)}$
7	3.935002781	4.003282759	$-2.36760 \cdot 10^{-3}$	0.782074
8	4.000557588	4.000000470	$2.91108 \cdot 10^{-7}$	-0.391982
9	4.000000053	4.000000000	$-2.29920 \cdot 10^{-15}$	-1.506178

Die Bedingung (5.105) ist bereits erfüllt, und wir erhalten den Eigenwert $\lambda_2 = 4$ und die Untermatrix der Ordnung vier

$$\hat{\mathbf{H}} \doteq \begin{pmatrix} 4.993897 & 5.997724 & -10.776453 & 2.629846 \\ -6.000388 & 4.999678 & -1.126468 & -0.830809 \\ 0 & -0.003201 & 3.512604 & 2.722802 \\ 0 & 0 & -3.777837 & -1.506178 \end{pmatrix}.$$

Die Eigenwerte von $\hat{\mathbf{H}}$ sind paarweise konjugiert komplex. Ihre Berechnung wird im folgenden Abschnitt im Beispiel 5.7 erfolgen. \triangle

5.5.3 QR-Doppelschritt, komplexe Eigenwerte

Die reelle Hessenberg-Matrix $\mathbf{H} = \mathbf{H}_1$ besitze jetzt auch Paare von konjugiert komplexen Eigenwerten. Da mit den Spektralverschiebungen (5.103) $\sigma_k = h_{nn}^{(k)}$ keine brauchbaren Näherungen für komplexe Eigenwerte zur Verfügung stehen, muss ihre Wahl angepasst werden. Man betrachtet deshalb in \mathbf{H}_k die Untermatrix der Ordnung zwei in der rechten unteren Ecke

$$\mathbf{C}_k := \begin{pmatrix} h_{n-1,n-1}^{(k)} & h_{n-1,n}^{(k)} \\ h_{n,n-1}^{(k)} & h_{n,n}^{(k)} \end{pmatrix}. \quad (5.108)$$

Sind die Eigenwerte $\mu_1^{(k)}$ und $\mu_2^{(k)}$ von \mathbf{C}_k reell, so wird die Verschiebung σ_k anstelle von (5.103) häufig durch denjenigen Eigenwert $\mu_1^{(k)}$ festgelegt, welcher näher bei $h_{nn}^{(k)}$ liegt, d.h.

$$\boxed{\sigma_k = \mu_1^{(k)} \in \mathbb{R} \text{ mit } |\mu_1^{(k)} - h_{nn}^{(k)}| \leq |\mu_2^{(k)} - h_{nn}^{(k)}|, k = 1, 2, \dots} \quad (5.109)$$

Falls aber die Eigenwerte von \mathbf{C}_k konjugiert komplex sind, sollen die Spektralverschiebungen für die beiden folgenden QR-Transformationen durch die konjugiert komplexen Werte festgelegt werden.

$$\boxed{\sigma_k = \mu_1^{(k)}, \quad \sigma_{k+1} = \mu_2^{(k)} = \bar{\sigma}_k, \quad \mu_1^{(k)} \in \mathbb{C}} \quad (5.110)$$

Die Matrix $\mathbf{H}_k - \sigma_k \mathbf{I}$ ist mit (5.110) eine Matrix mit komplexen Diagonalelementen. Nun sind aber wegen (5.110) die beiden Werte

$$\boxed{\begin{aligned} s &:= \sigma_k + \sigma_{k+1} = h_{n-1,n-1}^{(k)} + h_{n,n}^{(k)} \\ t &:= \sigma_k \sigma_{k+1} = h_{n-1,n-1}^{(k)} h_{n,n}^{(k)} - h_{n-1,n}^{(k)} h_{n,n-1}^{(k)} \end{aligned}} \quad (5.111)$$

reell, und deshalb ist auch die Matrix

$$\mathbf{X} := \mathbf{H}_k^2 - s\mathbf{H}_k + t\mathbf{I} \quad (5.112)$$

reell. Damit gelingt es, einen Doppelschritt $\mathbf{H}_k \rightarrow \mathbf{H}_{k+2}$ mit reeller Rechnung und vertretbarem Aufwand durchzuführen. Auf Details wollen wir verzichten, sondern nur die algorithmischen Schritte angeben. \mathbf{H}_{k+2} ist orthogonal-ähnlich zu \mathbf{H}_k . Für die Berechnung von $\mathbf{H}_{k+2} = \mathbf{Q}^T \mathbf{H}_k \mathbf{Q}$ sind die folgenden Tatsachen wesentlich. Erstens ist die orthogonale

Matrix \mathbf{Q} durch die QR -Zerlegung von \mathbf{X} definiert. Die erste Spalte von \mathbf{Q} ist im Wesentlichen durch die erste Spalte von \mathbf{X} festgelegt. Dann sei \mathbf{Q}_0 diejenige orthogonale Matrix, welche in der QR -Zerlegung von \mathbf{X} nur die erste Spalte auf die gewünschte Form bringt, so dass gilt

$$\mathbf{Q}_0^T \mathbf{X} = \begin{pmatrix} r_{11} & \dots & \mathbf{g}^T \\ \mathbf{0} & \vdots & \hat{\mathbf{X}} \end{pmatrix}. \quad (5.113)$$

Im weiteren Verlauf der QR -Zerlegung mit der Folge von Givens-Rotationen (5.87) ändert sich die erste Spalte von \mathbf{Q}_0 nicht mehr. Zweitens hat die orthogonale Matrix $\tilde{\mathbf{Q}}$, welche eine beliebige Matrix \mathbf{A} auf Hessenberg-Form transformiert, die Eigenschaft, dass ihre erste Spalte $\tilde{\mathbf{q}}_1$ gleich dem ersten Einheitsvektor \mathbf{e}_1 ist. In der Tat folgt dies auf Grund der Darstellung von $\tilde{\mathbf{Q}} = \mathbf{U}_1 \mathbf{U}_2 \dots \mathbf{U}_{N^*}$ als Produkt der Rotationsmatrizen \mathbf{U}_k unter Berücksichtigung der Rotationsindexpaare (5.58), unter denen der Index Eins nie vorkommt. Dann ist aber in der Matrix

$$\mathbf{Q} := \mathbf{Q}_0 \tilde{\mathbf{Q}} \quad (5.114)$$

die erste Spalte \mathbf{q}_1 gleich der ersten Spalte von \mathbf{Q}_0 . Wenn wir also die gegebene Hessenberg-Matrix \mathbf{H}_k in einem ersten Teilschritt mit der Matrix \mathbf{Q}_0 orthogonal-ähnlich in

$$\mathbf{B} = \mathbf{Q}_0^T \mathbf{H}_k \mathbf{Q}_0 \quad (5.115)$$

transformieren, und anschließend \mathbf{B} mittels $\tilde{\mathbf{Q}}$ wieder auf Hessenberg-Form gemäß

$$\mathbf{H}_{k+2} = \tilde{\mathbf{Q}}^T \mathbf{B} \tilde{\mathbf{Q}} = \tilde{\mathbf{Q}}^T \mathbf{Q}_0^T \mathbf{H}_k \mathbf{Q}_0 \tilde{\mathbf{Q}} = \mathbf{Q}^T \mathbf{H}_k \mathbf{Q} \quad (5.116)$$

bringen, dann ist auf Grund der wesentlichen Eindeutigkeit die gewünschte Transformation erreicht.

Zur Durchführung des skizzierten Vorgehens ist zuerst die Matrix \mathbf{Q}_0 durch die Elemente der ersten Spalte von \mathbf{X} (5.112) festzulegen. Da \mathbf{H}_k eine Hessenberg-Matrix ist, sind in der ersten Spalte von \mathbf{X} nur die ersten drei Elemente, die wir mit x_1, x_2 und x_3 bezeichnen, ungleich null. Sie sind wegen (5.112) wie folgt definiert.

$$\begin{aligned} x_1 &= h_{11}^2 + h_{12}h_{21} - sh_{11} + t \\ x_2 &= h_{21}[h_{11} + h_{22} - s] \\ x_3 &= h_{21}h_{32} \end{aligned} \quad (5.117)$$

Die Matrix \mathbf{Q}_0 ist infolge dieser speziellen Struktur als Produkt von zwei Rotationsmatrizen darstellbar

$$\mathbf{Q}_0 = \mathbf{U}(1, 2; \varphi_1) \mathbf{U}(1, 3; \varphi_2) \quad (5.118)$$

mit Winkeln φ_1 und φ_2 , die sich nacheinander aus den Wertepaaren (x_1, x_2) und (x'_1, x_3) nach (5.89) bestimmen, wobei x'_1 der durch die erste Transformation geänderte Wert von x_1 ist.

Die Ähnlichkeitstransformation (5.115) von \mathbf{H}_k mit \mathbf{Q}_0 betrifft in \mathbf{H}_k nur die ersten drei Zeilen und Spalten, so dass die Matrix \mathbf{B} eine spezielle Struktur erhält. Sie ist beispielsweise

für $n = 7$:

$$\mathbf{B} = \mathbf{Q}_0^T \mathbf{H}_k \mathbf{Q}_0 = \begin{pmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ \boxed{\times} & \times & \times & \times & \times & \times & \times \\ \times & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times \end{pmatrix} \quad (5.119)$$

Die Hessenberg-Form ist nur in der ersten Spalte von \mathbf{B} verloren gegangen, da dort zwei im Allgemeinen von null verschiedene Matrixelemente unterhalb der Subdiagonalen entstanden sind. Es gilt nun, die Matrix \mathbf{B} ähnlich auf Hessenberg-Form zu transformieren, was mit geeigneten Givens-Rotationen mit Matrizen $\mathbf{U}_{23} = \mathbf{U}(2, 3; \varphi_3)$ und $\mathbf{U}_{24} = \mathbf{U}(2, 4; \varphi_4)$ erfolgt. Da dabei zuerst nur die zweiten und dritten Zeilen und Spalten und dann nur die zweiten und vierten Zeilen und Spalten linear kombiniert werden, erhält die so transformierte Matrix für $n = 7$ folgende Struktur.

$$\mathbf{B}_1 := \mathbf{U}_{24}^T \mathbf{U}_{23}^T \mathbf{B} \mathbf{U}_{23} \mathbf{U}_{24} = \begin{pmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & \boxed{\times} & \times & \times & \times & \times & \times \\ 0 & \boxed{\times} & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times \end{pmatrix} \quad (5.120)$$

Die beiden von null verschiedenen Matrixelemente in der ersten Spalte unterhalb der Subdiagonalen von \mathbf{B} sind durch die Transformation (5.120) in zwei andere, von null verschiedenen Matrixelemente in der zweiten Spalte unterhalb der Subdiagonalen von \mathbf{B}_1 übergegangen. Mit Ausnahme der zweiten Spalte hat \mathbf{B}_1 Hessenberg-Gestalt. Mit Paaren von analogen Givens-Transformationen werden die zwei die Hessenberg-Form störenden Elemente sukzessive nach rechts unten verschoben, bis die Hessenberg-Gestalt wieder hergestellt ist. Die Behandlung der $(n-2)$ -ten Spalte erfordert selbstverständlich nur eine Givens-Transformation.

Damit sind die rechentechnischen Details zur Berechnung von \mathbf{H}_{k+2} aus \mathbf{H}_k für einen QR-Doppelschritt mit den zueinander konjugiert komplexen Spektralverschiebungen σ_k und $\sigma_{k+1} = \bar{\sigma}_k$ nach (5.110) vollständig beschrieben, mehr Einzelheiten finden sich in [Sch 97]. Die beiden Verschiebungen mit σ_k und σ_{k+1} werden bei diesem Vorgehen gar nicht explizit vorgenommen, vielmehr sind sie nur durch die Werte s und t (5.111) und dann durch die Werte x_1, x_2 und x_3 (5.117) zur Festlegung der ersten Spalte von \mathbf{Q} verwendet worden, d.h. sie sind implizit in der orthogonalen Matrix \mathbf{Q}_0 enthalten. Deshalb nennt man den Übergang von \mathbf{H}_k nach \mathbf{H}_{k+2} einen *QR-Doppelschritt mit impliziter Spektralverschiebung* [Fra 62].

Der Rechenaufwand für einen solchen QR-Doppelschritt setzt sich im Wesentlichen aus demjenigen zur Berechnung von \mathbf{B} (5.119) und demjenigen zur Transformation von \mathbf{B} in \mathbf{H}_{k+2} zusammen. Die Bereitstellung der Werte $c = \cos \varphi$ und $s = \sin \varphi$ der beiden Rotationsmatrizen für \mathbf{Q}_0 benötigt acht multiplikative Operationen und zwei Quadratwurzeln. Die Ausführung der beiden Givens-Transformationen erfordert $8n + 24$ Multiplikationen. Dasselbe gilt für die beiden Transformationen, die zur Behandlung der j -ten Spalte nötig sind, unabhängig von j , falls die Struktur beachtet wird. Da der letzte Schritt nur halb so

aufwändig ist, beträgt der Rechenaufwand für den QR -Doppelschritt etwa

$$Z_{\text{QR-Doppel}} \cong 8n^2 + 20n - 44 \quad (5.121)$$

multiplikative Operationen und $(2n - 3)$ Quadratwurzeln. Er verdoppelt sich gegenüber dem Aufwand (5.92) für einen einzelnen QR -Schritt mit expliziter Spektralverschiebung für größere n . Der Rechenaufwand für den Doppelschritt reduziert sich, falls entweder die schnelle Givens-Transformation [Rat 82] oder die *Householder-Transformation* [Gol 96b, Ste 83, Wil 86] angewandt wird. Im besten Fall wird der Aufwand $Z_{\text{QR-Doppel}} \approx 5n^2$.

Beispiel 5.7. Die beiden Eigenwerte der Untermatrix \mathbf{C}_k (5.108) der Hessenberg-Matrix $\hat{\mathbf{H}}$ von Beispiel 5.6 sind konjugiert komplex. Mit $s_1 \doteq 2.006425$ und $t_1 \doteq 4.995697$ liefert der QR -Doppelschritt mit impliziter Spektralverschiebung die Matrix

$$\hat{\mathbf{H}}_2 \doteq \begin{pmatrix} 4.999999 & -5.999999 & -3.368568 & 2.979044 \\ 5.999997 & 5.000002 & -9.159566 & 4.566268 \\ 0 & 2.292 \cdot 10^{-6} & 1.775362 & -5.690920 \\ 0 & 0 & 0.808514 & 0.224637 \end{pmatrix}.$$

Nach einem weiteren QR -Doppelschritt mit $s_2 \doteq 1.999999$ und $t_2 \doteq 5.000001$ zerfällt die Hessenberg-Matrix

$$\hat{\mathbf{H}}_3 \doteq \begin{pmatrix} 5.000000 & -6.000000 & -6.320708 & 5.447002 \\ 6.000000 & 5.000000 & 4.599706 & -5.847384 \\ 0 & 0 & 0.296328 & -5.712541 \\ 0 & 0 & 0.786892 & 1.703672 \end{pmatrix}.$$

Daraus berechnen sich die beiden Paare von konjugiert komplexen Eigenwerten $\lambda_{3,4} = 1 \pm 2i$ und $\lambda_{5,6} = 5 \pm 6i$. \triangle

5.5.4 QR -Algorithmus für tridiagonale Matrizen

Im Abschnitt 5.4.1 wurde gezeigt, wie eine symmetrische Matrix \mathbf{A} orthogonal-ähnlich auf eine symmetrische, tridiagonale Matrix

$$\mathbf{J} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix} \quad (5.122)$$

transformiert werden kann. Wir setzen voraus, dass die Matrix \mathbf{J} nicht zerfällt, so dass also $\beta_i \neq 0$ für $i = 1, 2, \dots, (n - 1)$ gilt. Da nach Satz 5.11 der QR -Algorithmus eine Folge von orthogonal-ähnlichen tridiagonalen Matrizen $\mathbf{J}_{k+1} = \mathbf{Q}_k^T \mathbf{J}_k \mathbf{Q}_k$, $k = 1, 2, \dots$, mit $\mathbf{J}_1 = \mathbf{J}$ erzeugt, ergibt sich ein sehr effizienter Algorithmus zur Berechnung aller Eigenwerte von \mathbf{J} . Als Spektralverschiebung σ_k könnte nach (5.103) $\sigma_k = \alpha_n^{(k)}$ gewählt werden. Doch bietet die Festsetzung der Verschiebung σ_k durch denjenigen der beiden reellen Eigenwerte der

Untermatrix

$$C_k := \begin{pmatrix} \alpha_{n-1}^{(k)} & \beta_{n-1}^{(k)} \\ \beta_{n-1}^{(k)} & \alpha_n^{(k)} \end{pmatrix}, \quad (5.123)$$

welcher näher bei $\alpha_n^{(k)}$ liegt, Vorteile bezüglich der Konvergenzeigenschaften des Algorithmus. Die Eigenwerte von C_k sind gegeben durch

$$\begin{aligned} \mu_{1,2}^{(k)} &= \frac{\alpha_{n-1}^{(k)} + \alpha_n^{(k)}}{2} \pm \sqrt{d^2 + \beta_{n-1}^{(k)2}} \\ &= \alpha_n^{(k)} + d \pm \sqrt{d^2 + \beta_{n-1}^{(k)2}} \quad \text{mit } d = \frac{\alpha_{n-1}^{(k)} - \alpha_n^{(k)}}{2}. \end{aligned}$$

Man erhält den zu $\alpha_n^{(k)}$ näher gelegenen Eigenwert, wenn man das Vorzeichen der Wurzel gleich dem entgegengesetzten von d setzt. Ist zufällig $d = 0$, dann sind beide Eigenwerte $\mu_{1,2}^{(k)}$ gleich weit von $\alpha_n^{(k)}$ entfernt, und man wählt $\sigma_k = \alpha_n^{(k)}$, falls $\alpha_n^{(k)} \neq 0$ ist. Damit erhält die Spektralverschiebung im Normalfall die Darstellung

$$\sigma_k = \alpha_n^{(k)} + d - \operatorname{sgn}(d) \sqrt{d^2 + \beta_{n-1}^{(k)2}}, \quad d = \frac{1}{2}(\alpha_{n-1}^{(k)} - \alpha_n^{(k)}). \quad (5.124)$$

Zur praktischen Durchführung des QR-Schrittes $\mathbf{J}_k - \sigma_k \mathbf{I} = \mathbf{Q}_k \mathbf{R}_k$, $\mathbf{J}_{k+1} = \mathbf{R}_k \mathbf{Q}_k + \sigma_k \mathbf{I}$ mit $\mathbf{J}_{k+1} = \mathbf{Q}_k^T \mathbf{J}_k \mathbf{Q}_k$ soll die Technik der *impliziten* Spektralverschiebung angewandt werden. Die Überlegungen von Abschnitt 5.5.3 werden sinngemäß und in vereinfachter Form übernommen.

Die Matrix \mathbf{Q}_0 , welche in der QR-Zerlegung von $\mathbf{J}_k - \sigma_k \mathbf{I}$ die erste Spalte transformiert, ist im vorliegenden Fall eine Jacobi-Matrix $\mathbf{U}(1, 2; \varphi_0)$, deren Werte $c = \cos \varphi_0$ und $s = \sin \varphi_0$ durch die beiden Elemente $\alpha_1 - \sigma_k$ und β_1 bestimmt sind. Mit \mathbf{Q}_0 bilden wir die orthogonal-ähnliche Matrix $\mathbf{B} = \mathbf{Q}_0^T \mathbf{J}_k \mathbf{Q}_0$, die anschließend durch weitere orthogonale Ähnlichkeitstransformationen auf tridiagonale Form gebracht werden muss. Die Matrix \mathbf{B} besitzt eine besonders einfache Struktur, da nur die ersten beiden Zeilen und Spalten von \mathbf{J}_k verändert werden. Ohne den oberen Indexwert k lautet \mathbf{B} für $n = 6$

$$\mathbf{B} = \mathbf{Q}_0^T \mathbf{J}_k \mathbf{Q}_0 = \begin{pmatrix} \alpha'_1 & \beta'_1 & y & & & \\ \beta'_1 & a'_2 & \beta'_2 & & & \\ y & \beta'_2 & \alpha_3 & \beta_3 & & \\ & \beta_3 & \alpha_4 & \beta_4 & & \\ & & \beta_4 & \alpha_5 & \beta_5 & \\ & & & \beta_5 & \alpha_6 & \end{pmatrix}. \quad (5.125)$$

Die tridiagonale Gestalt ist nur in der ersten Spalte und ersten Zeile zerstört worden. Eine erste Givens-Rotation mit einer Jacobi-Matrix $\mathbf{U}_1 = \mathbf{U}(2, 3; \varphi_1)$ eliminiert mit geeignetem Winkel φ_1 das Element y in (5.125), erzeugt aber ein neues, von null verschiedenes Paar

von Matrixelementen außerhalb der drei Diagonalen. Das Resultat ist

$$\mathbf{U}_1^T \mathbf{B} \mathbf{U}_1 = \begin{pmatrix} \alpha'_1 & \beta''_1 & & & & \\ \beta''_1 & \alpha''_2 & \beta''_2 & y' & & \\ & \beta''_2 & \alpha'_3 & \beta'_3 & & \\ & y' & \beta'_3 & \alpha_4 & \beta_4 & \\ & & & \beta_4 & \alpha_5 & \beta_5 \\ & & & & \beta_5 & \alpha_6 \end{pmatrix}. \quad (5.126)$$

Durch jede weitere Givens-Rotation wird das störende Element nach rechts unten verschoben. Nach $(n-2)$ Transformationsschritten ist \mathbf{B} auf tridiagonale Form gebracht und stellt damit die Matrix \mathbf{J}_{k+1} dar.

Für die praktische Realisierung eines QR -Schrittes mit impliziter Spektralverschiebung σ für eine tridiagonale Matrix \mathbf{J} werden die Elemente der beiden Diagonalen zweckmäßigerweise als zwei Vektoren $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n)^T$ und $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_{n-1})^T$ vorgegeben, mit denen die Transformation von \mathbf{J}_k in \mathbf{J}_{k+1} vorgenommen wird. Weiter ist zu beachten, dass alle auszuführenden $(n-1)$ Givens-Transformationen zwei aufeinanderfolgende Zeilen und Spalten betreffen, so dass die Diagonalelemente und die zugehörigen Nebendiagonalelemente nach den Formeln (5.14), (5.15) und (5.16) umzurechnen sind. Die Darstellung (5.23) ist nicht anwendbar, da der Drehwinkel φ anders bestimmt wird. Die erwähnten Formeln werden zur Reduktion der Rechenoperationen mit $c = \cos \varphi, s = \sin \varphi$ und $q = p + 1$ wie folgt umgeformt.

$$\begin{aligned} \alpha''_p &= \alpha_p - 2\beta_p c s - (\alpha_p - \alpha_{p+1})s^2 = \alpha_p - z, \\ \alpha''_{p+1} &= \alpha_{p+1} + 2\beta_p c s + (\alpha_p - \alpha_{p+1})s^2 = \alpha_{p+1} + z, \\ \beta''_p &= (\alpha_p - \alpha_{p+1})c s + \beta_p(c^2 - s^2), \\ \text{mit } z &= [2\beta_p c + (\alpha_p - \alpha_{p+1})s]s. \end{aligned}$$

$$\begin{aligned} x &= \alpha_1 - \sigma; \quad y = \beta_1 \\ \text{für } p &= 1, 2, \dots, n-1: \\ \text{falls } |x| &\leq \tau \times |y|: \\ w &= -y; \quad c = 0; \quad s = 1 \\ \text{sonst} \quad w &= \sqrt{x^2 + y^2}; \quad c = x/w; \quad s = -y/w \end{aligned}$$

$$d = \alpha_p - \alpha_{p+1}; \quad z = (2 \times c \times \beta_p + d \times s) \times s$$

$$\alpha_p = \alpha_p - z; \quad \alpha_{p+1} = \alpha_{p+1} + z$$

$$\beta_p = d \times c \times s + (c^2 - s^2) \times \beta_p; \quad x = \beta_p$$

$$\text{falls } p > 1: \quad \beta_{p-1} = w$$

$$\text{falls } p < n-1: \quad y = -s \times \beta_{p+1}; \quad \beta_{p+1} = c \times \beta_{p+1}$$

(5.127)

Schließlich unterscheidet sich die Bestimmung des Drehwinkels für die Transformation mit \mathbf{Q}_0 von derjenigen für die nachfolgenden Givens-Rotationen mit \mathbf{U}_k . Dies wird durch eine geeignete Definition von Variablen x und y im Algorithmus (5.127) erreicht. Die Spektralverschiebung σ sei nach (5.124) vorgegeben. Ein QR -Schritt für eine symmetrische, tridiagonale

Matrix \mathbf{J}_k erfordert mit dem Algorithmus (5.127) etwa

$$Z_{\text{QR,trid}} \cong 15(n - 1) \quad (5.128)$$

multiplikative Operationen und $(n - 1)$ Quadratwurzeln. Die hohe Effizienz des Verfahrens beruht darauf, dass das letzte Außendiagonalelement $\beta_{n-1}^{(k)}$ *kubisch* gegen null konvergiert [Gou 79, Wil 68]. Das hat zur Folge, dass für größere Ordnung n im Durchschnitt zur Berechnung aller Eigenwerte nur zwei bis drei QR-Schritte nötig sind, da mit fortschreitender Rechnung die Spektralverschiebungen (5.124) hervorragende Näherungen für den nächsten Eigenwert liefern. Zudem nimmt die Ordnung der zu bearbeitenden tridiagonalen Matrizen ab.

Beispiel 5.8. Der QR-Algorithmus zur Berechnung der Eigenwerte der tridiagonalen Matrix \mathbf{J} (5.71) aus Beispiel 5.4 ist sehr effizient. Mit den Spektralverschiebungen σ_k (5.124) liefern drei QR-Schritte Werte gemäß folgender Zusammenstellung.

k	σ_k	$\alpha_5^{(k+1)}$	$\beta_4^{(k+1)}$
1	1.141933723	1.330722500	-0.148259790
2	1.323643137	1.327045601	$1.46939 \cdot 10^{-4}$
3	1.327045590	1.327045600	$-4.388 \cdot 10^{-13}$

Die kubische Konvergenz von $\beta_4^{(k)}$ gegen null ist deutlich erkennbar. Nach diesen drei QR-Schritten zerfällt die tridiagonale Matrix. Mit $\alpha_5^{(4)}$ ist der erste Eigenwert $\lambda_1 \doteq 1.327045600$ gefunden. Die reduzierte Matrix der Ordnung vier lautet

$$\hat{\mathbf{J}} \doteq \begin{pmatrix} 22.354976 & 0.881561 & & \\ 0.881561 & 7.399454 & 0.819413 & \\ & 0.819413 & 1.807160 & 3.010085 \\ & & 3.010085 & 2.111365 \end{pmatrix}.$$

Die wichtigsten Werte der weiteren QR-Schritte für $\hat{\mathbf{J}}$ sind in der folgenden Tabelle zusammengestellt.

k	σ_k	$\alpha_4^{(k+1)}$	$\beta_3^{(k+1)}$
4	4.973188459	4.846875984	0.116631770
5	4.849265094	4.848950120	$6.52773 \cdot 10^{-6}$
6	4.848950120	4.848950120	$-1.2 \cdot 10^{-16}$

Der zweite berechnete Eigenwert ist $\lambda_2 \doteq 4.848950120$, und die verbleibende Untermatrix der Ordnung drei ist

$$\hat{\hat{\mathbf{J}}} \doteq \begin{pmatrix} 22.406874 & 0.003815 & \\ 0.003815 & 3.850210 & 4.257076 \\ & 4.257076 & 2.566920 \end{pmatrix},$$

aus der sich die weiteren Eigenwerte mit zwei, bzw. einem QR-Schritt in der Reihenfolge $\lambda_3 \doteq -1.096595182$, $\lambda_4 \doteq 7.513724154$ und $\lambda_5 \doteq 22.406875308$ bestimmen. \triangle

5.5.5 Zur Berechnung der Eigenvektoren

Der QR -Algorithmus, wie er oben dargestellt worden ist, liefert die Eigenwerte einer Hessenberg-Matrix \mathbf{H} oder einer tridiagonalen, symmetrischen Matrix \mathbf{J} . Wir behandeln jetzt noch die Aufgabe, die zugehörigen Eigenvektoren zu bestimmen.

Das Produkt von allen orthogonalen Matrizen $\mathbf{Q}_k, k = 1, 2, \dots, M$, die im Verlauf des QR -Algorithmus auftreten, ist eine orthogonale Matrix

$$\mathbf{Q} := \mathbf{Q}_1 \mathbf{Q}_2 \dots \mathbf{Q}_M, \quad (5.129)$$

welche \mathbf{H} auf eine Quasidreiecksmatrix $\mathbf{R} = \mathbf{Q}^T \mathbf{H} \mathbf{Q}$ (5.93), bzw. \mathbf{J} auf eine Diagonalmatrix $\mathbf{D} = \mathbf{Q}^T \mathbf{J} \mathbf{Q}$ transformiert. Mit dieser Matrix \mathbf{Q} wird die Eigenwertaufgabe $\mathbf{H}\mathbf{x} = \lambda\mathbf{x}$, bzw. $\mathbf{J}\mathbf{x} = \lambda\mathbf{x}$ übergeführt in

$$\mathbf{R}\mathbf{y} = \lambda\mathbf{y}, \quad \text{bzw.} \quad \mathbf{D}\mathbf{y} = \lambda\mathbf{y} \quad \text{mit} \quad \mathbf{y} = \mathbf{Q}^T \mathbf{x}. \quad (5.130)$$

Bei bekanntem Eigenwert λ_k ist der zugehörige Eigenvektor \mathbf{y}_k von \mathbf{R} einfach berechenbar. Im Fall der Matrix \mathbf{D} ist $\mathbf{y}_k = \mathbf{e}_k$, falls λ_k das k -te Diagonalelement von \mathbf{D} ist. Der Eigenvektor \mathbf{x}_k von \mathbf{H} ist wegen (5.130) gegeben durch $\mathbf{x}_k = \mathbf{Q}\mathbf{y}_k$, derjenige von \mathbf{J} ist gleich der k -ten Spalte von \mathbf{Q} .

Eine erste Methode zur Berechnung der Eigenvektoren besteht darin, die Matrix \mathbf{Q} (5.129) explizit als Produkt von sämtlichen \mathbf{Q}_k zu bilden. Jede der Matrizen \mathbf{Q}_k ist selbst wieder das Produkt von einfachen Rotationsmatrizen, so dass sich \mathbf{Q} analog zu (5.45) rekursiv aufbauen lässt. Der Rechenaufwand für einen einzelnen QR -Schritt erhöht sich sehr stark, denn die Multiplikation von rechts mit einer Rotationsmatrix ist stets für die ganzen Spalten von \mathbf{Q} auszuführen, auch dann, wenn nach Berechnung von einigen Eigenwerten die Ordnung der zu behandelnden Matrix \mathbf{H} oder \mathbf{J} kleiner als n ist. Das skizzierte Vorgehen hat den Vorteil, dass in \mathbf{Q} für die tridiagonale, symmetrische Matrix \mathbf{J} stets n orthonormierte Eigenvektoren geliefert werden.

Um die aufwändige Berechnung der Matrix \mathbf{Q} zu vermeiden, können die Eigenvektoren \mathbf{x}_k auch mit der *inversen Vektoriteration* nach Abschnitt 5.3.2 bestimmt werden. Der Startvektor $\mathbf{z}^{(0)}$ kann nach einem Vorschlag von *Wilkinson* als Lösung des Gleichungssystems

$$\tilde{\mathbf{R}}\mathbf{z}^{(0)} = \mathbf{e}, \quad \mathbf{e} = (1, 1, \dots, 1)^T \quad (5.131)$$

gefunden werden mit der vom Näherungswert $\bar{\lambda}$ für λ_k abhängigen Rechtsdreiecksmatrix $\tilde{\mathbf{R}}$, welche im Gauß-Algorithmus bei der Zerlegung von $(\mathbf{H} - \bar{\lambda}\mathbf{I})$, bzw. von $(\mathbf{J} - \bar{\lambda}\mathbf{I})$ unter Verwendung der relativen Spaltenmaximumstrategie entsteht. Der Vektor $\mathbf{z}^{(0)}$ wird aus (5.131) durch Rücksubstitution gewonnen. Dies entspricht einem halben Iterationsschritt der inversen Vektoriteration.

Die Eigenvektorbestimmung nach der zweiten Methode erfordert im Wesentlichen eine Zerlegung von $(\mathbf{H} - \bar{\lambda}\mathbf{I})$ für jeden Eigenwert. Eine Zerlegung für eine Hessenberg-Matrix \mathbf{H} der Ordnung n erfordert etwa $n^2/2$, die Vorwärtssubstitution etwa n und die Rücksubstitution etwa $n^2/2$ multiplikative Operationen. Unter der meistens zutreffenden Annahme, dass zwei Iterationsschritte der inversen Vektoriteration genügen, beträgt der Rechenaufwand zur Bestimmung von allen n Eigenvektoren etwa $2n^3$ Operationen. Für eine tridiagonale Matrix \mathbf{J} ist der Rechenaufwand sogar nur proportional zu n^2 .

Ist die Matrix \mathbf{H} bzw. \mathbf{J} durch eine Ähnlichkeitstransformation aus einer Matrix \mathbf{A} entstanden, sind die Eigenvektoren gemäß Abschnitt 5.4.1 in diejenigen von \mathbf{A} zurückzutransformieren.

5.6 Das allgemeine Eigenwertproblem

Wir betrachten jetzt für zwei Matrizen \mathbf{A}, \mathbf{B} der Ordnung n das Problem

$$\mathbf{Ax} = \lambda \mathbf{Bx}. \quad (5.132)$$

Dieses Problem kann auf das spezielle Eigenwertproblem zurückgeführt werden, wenn \mathbf{B} regulär, also invertierbar ist:

$$\mathbf{Ax} = \lambda \mathbf{Bx} \iff \mathbf{B}^{-1} \mathbf{Ax} = \lambda \mathbf{x}. \quad (5.133)$$

Diese Vorgehensweise ist aber bei schlecht konditioniertem \mathbf{B} nicht empfehlenswert. Sind \mathbf{A} und \mathbf{B} symmetrisch, so kann außerdem die Symmetrie bei $\mathbf{B}^{-1} \mathbf{A}$ verlorengehen. Eine symmetrie- und stabilitätserhaltende Transformation ist möglich, wenn \mathbf{A} oder \mathbf{B} symmetrisch und positiv definit ist, siehe Abschnitt 5.6.1. Sind beide Matrizen symmetrisch, aber nicht notwendig positiv definit, so hilft der folgende Satz, dessen konstruktiver Beweis einen Algorithmus liefert, der mehrere Singulärwertzerlegungen benutzt, [Rao 71]:

Satz 5.14. *Sind \mathbf{A}, \mathbf{B} zwei symmetrische, positiv semidefinite Matrizen der Ordnung n , dann gibt es eine reguläre Matrix \mathbf{T} derart, dass $\mathbf{T}^{-1} \mathbf{AT}$ und $\mathbf{T}^{-1} \mathbf{BT}$ diagonal sind.*

Für den unsymmetrischen Fall ohne Regularitätsvoraussetzung an \mathbf{A} oder \mathbf{B} ist der QZ-Algorithmus zu empfehlen, siehe etwa [Mol 73].

5.6.1 Der symmetrisch positiv definite Fall

Bei vielen Anwendungen sind beide Matrizen \mathbf{A} und \mathbf{B} symmetrisch und mindestens eine ist positiv definit. Dann lässt sich das allgemeine Eigenwertproblem numerisch stabil auf das spezielle Eigenwertproblem zurückführen. Hier soll o.B.d.A. diese Eigenschaften für \mathbf{B} vorausgesetzt werden. Dann existiert die Cholesky-Zerlegung (siehe Abschnitt 2.3.1) von \mathbf{B}

$$\mathbf{B} = \mathbf{L} \mathbf{L}^T. \quad (5.134)$$

Die Zerlegung (5.134) von \mathbf{B} wird in (5.132) eingesetzt, die Gleichung von links mit \mathbf{L}^{-1} multipliziert und noch die Identität $\mathbf{I} = \mathbf{L}^{-T} \mathbf{L}^T$ eingefügt:

$$(\mathbf{L}^{-1} \mathbf{A} \mathbf{L}^{-T})(\mathbf{L}^T \mathbf{x}) = \lambda (\mathbf{L}^{-1} \mathbf{L})(\mathbf{L}^T \mathbf{x}) \quad (5.135)$$

Mit den Substitutionen

$$\mathbf{C} = \mathbf{L}^{-1} \mathbf{A} \mathbf{L}^{-T}, \quad \mathbf{y} = \mathbf{L}^T \mathbf{x} \quad (5.136)$$

wird (5.135) in der Tat ein spezielles Eigenwertproblem

$$\mathbf{Cy} = \lambda \mathbf{y} \quad (5.137)$$

mit der symmetrischen Matrix \mathbf{C} , deren Eigenvektoren \mathbf{y}_j wegen (5.136) mit den Eigenvektoren \mathbf{x}_j von (5.132) zusammenhängen. Die Symmetrie von \mathbf{C} bestätigt man unter Benutzung der Symmetrie von \mathbf{A} durch

$$\mathbf{C}^T = (\mathbf{L}^{-1} \mathbf{A} \mathbf{L}^{-T})^T = \mathbf{L}^{-1} \mathbf{A}^T \mathbf{L}^{-T} = \mathbf{L}^{-1} \mathbf{A} \mathbf{L}^{-T} = \mathbf{C}. \quad (5.138)$$

Die Eigenvektoren \mathbf{y}_j einer symmetrischen Matrix \mathbf{C} bilden bei entsprechender Normierung ein Orthonormalsystem, d.h.

$$\mathbf{y}_j^T \mathbf{y}_k = \delta_{jk}, \quad j, k = 1, 2, \dots, n. \quad (5.139)$$

Wegen der Relation (5.136) bilden die Eigenvektoren \mathbf{x}_j der allgemeinen Eigenwertaufgabe (5.132) ein System von orthonormierten Vektoren im verallgemeinerten Sinn bezüglich der positiv definiten Matrix \mathbf{B} . Sie erfüllen die Beziehungen

$$\mathbf{y}_j^T \mathbf{y}_k = \mathbf{x}_j^T \mathbf{L} \mathbf{L}^T \mathbf{x}_k = \mathbf{x}_j^T \mathbf{B} \mathbf{x}_k = \delta_{jk}, \quad j, k = 1, 2, \dots, n. \quad (5.140)$$

Mit dieser Normierung der Eigenvektoren \mathbf{x}_j folgt aus der Eigenwertbeziehung $\mathbf{A} \mathbf{x}_k = \lambda_k \mathbf{B} \mathbf{x}_k$ nach Multiplikation mit \mathbf{x}_j^T von links unter Benutzung von (5.140) überdies

$$\mathbf{x}_j^T \mathbf{A} \mathbf{x}_k = \lambda_k \delta_{jk}, \quad j, k = 1, 2, \dots, n. \quad (5.141)$$

Das bedeutet, dass die Eigenvektoren zu verschiedenen indizierten Eigenwerten - die Eigenwerte dürfen dabei gleich sein - sowohl bezüglich \mathbf{B} als auch bezüglich \mathbf{A} im verallgemeinerten Sinn orthogonal sind.

Die tatsächliche Reduktion von (5.132) in (5.137) erfordert nach erfolgter Cholesky-Zerlegung von \mathbf{B} die Berechnung der Matrix \mathbf{C} nach (5.136). Dazu sind die Inversion der Linksdreiecksmatrix \mathbf{L} und die beiden Matrizenmultiplikationen gar nicht nötig. Vielmehr lässt sich die Matrix \mathbf{C} am effizientesten und mit kleinstem Speicherbedarf wie folgt berechnen. Die Hilfsmatrix $\mathbf{H} = \mathbf{A} \mathbf{L}^{-T}$ wird aus $\mathbf{H} \mathbf{L}^T = \mathbf{A}$ spaltenweise bestimmt. Für $n = 4$ lautet diese Matrizengleichung in reduzierter Form

$$\begin{pmatrix} h_{11} & & & \\ h_{21} & h_{22} & & \\ h_{31} & h_{32} & h_{33} & \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ & l_{22} & l_{32} & l_{42} \\ & & l_{33} & l_{43} \\ & & & l_{44} \end{pmatrix} = \begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}. \quad (5.142)$$

In (5.142) wurde in der Matrix \mathbf{A} angedeutet, dass aus Symmetriegründen nur die übliche untere Hälfte von \mathbf{A} gespeichert und verwendet wird. Obwohl die Matrix \mathbf{H} nicht symmetrisch ist, genügt es, wie im zweiten Teilschritt der Reduktion ersichtlich sein wird, nur die Elemente in und unterhalb der Diagonale wirklich zu berechnen. Für diese Elemente ergibt sich wie bei der Vorwärtssubstitution

$$h_{ik} = \left\{ \frac{a_{ik} - \sum_{j=1}^{k-1} h_{ij} l_{kj}}{l_{kk}} \right\}$$

$$k = 1, 2, \dots, n; \quad i = k, k+1, \dots, n.$$

(5.143)

Für $k = 1$ ist selbstverständlich die Summe leer.

Die gesuchte Matrix $\mathbf{C} = \mathbf{L}^{-1} \mathbf{H}$ ist symmetrisch, weshalb nur die eine Hälfte aus der Matriengleichung $\mathbf{LC} = \mathbf{H}$ berechnet werden muss. Für $n = 4$ lautet die Bestimmungsgleichung

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ l_{31} & l_{32} & l_{33} & \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} c_{11} & & & \\ c_{21} & c_{22} & & \\ c_{31} & c_{32} & c_{33} & \\ c_{41} & c_{42} & c_{43} & c_{44} \end{pmatrix} = \begin{pmatrix} h_{11} & & & \\ h_{21} & h_{22} & & \\ h_{31} & h_{32} & h_{33} & \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix}. \quad (5.144)$$

In (5.144) ist in \mathbf{H} angedeutet, dass nur die Elemente in und unterhalb der Diagonale bekannt sind. Diese genügen aber vollkommen, um die wesentlichen Elemente von \mathbf{C} in und unterhalb der Diagonale zeilenweise sukzessive zu berechnen, wobei im Prozess der Vorrwärtssubstitution zu berücksichtigen ist, dass die Elemente von \mathbf{C} oberhalb der Diagonale nicht verfügbar sind und durch die dazu symmetrischen Elemente zu ersetzen sind.

$$c_{ik} = \left\{ \frac{h_{ik} - \sum_{j=1}^{k-1} l_{ij} c_{kj} - \sum_{j=k}^{i-1} l_{ij} c_{jk}}{l_{ii}} \right\} \quad i = 1, 2, \dots, n; \quad k = 1, 2, \dots, i. \quad (5.145)$$

Hier sind wieder einige Summen leer.

Was den Speicherbedarf anbetrifft, kann die Cholesky-Matrix \mathbf{L} am Platz von \mathbf{B} aufgebaut und gespeichert werden. Die Hilfsmatrix \mathbf{H} kann nach (5.143) genau so an die Stelle von \mathbf{A} gespeichert werden, da das Element a_{ik} nur zur Berechnung des betreffenden Elementes h_{ik} mit den gleichen Indizes gebraucht wird. Dieselbe Bemerkung gilt für (5.145), so dass die Matrix \mathbf{C} , genauer gesagt ihre untere Hälfte, am Platz von \mathbf{H} und damit von \mathbf{A} aufgebaut werden kann. Damit benötigt der ganze Reduktionsalgorithmus überhaupt keinen zusätzlichen Speicherplatz.

Beispiel 5.9.

$$\mathbf{A} = \begin{pmatrix} 192 & -56 & -80 \\ -56 & 75 & -34 \\ -80 & -34 & 165 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 64 & -8 & -8 \\ -8 & 17 & -3 \\ -8 & -3 & 66 \end{pmatrix}.$$

\mathbf{B} ist symmetrisch positiv definit und die Cholesky-Zerlegung $\mathbf{B} = \mathbf{LL}^T$ ergibt

$$\mathbf{L} = \begin{pmatrix} 8 & 0 & 0 \\ -1 & 4 & 0 \\ -1 & -1 & 8 \end{pmatrix}$$

Die Lösung der Gleichung (5.142)/(5.143) ergibt die untere Hälfte von \mathbf{H}

$$\mathbf{H} = \begin{pmatrix} 24 & & \\ -7 & 17 & \\ -10 & -11 & 18 \end{pmatrix}.$$

Jetzt liefert die Lösung von (5.144)/(5.145) für das spezielle Eigenwertproblem $\mathbf{Cx} = \lambda x$ die Matrix

$$\mathbf{C} = \mathbf{L}^{-1} \mathbf{A} (\mathbf{L}^T)^{-1} = \begin{pmatrix} 3 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 2 \end{pmatrix}.$$

△

5.7 Eigenwertschranken, Kondition, Stabilität

Definition 5.15. 1. Als *Spektralradius* $\rho(\mathbf{A})$ bezeichnet man den betragsmäßig größten Eigenwert von \mathbf{A} .

2. Als *Spektrum* $\sigma(\mathbf{A})$ bezeichnet man die Menge aller Eigenwerte von \mathbf{A}

$$\sigma(\mathbf{A}) := \{\lambda_1, \lambda_2, \dots, \lambda_n\} \quad (5.146)$$

3. Als *Modalmatrix* wird eine Matrix mit Eigenvektoren als Spalten bezeichnet.

Lemma 5.16. Ist die Matrixnorm $\|\cdot\|$ submultiplikativ, so gilt für jeden Eigenwert λ von \mathbf{A}

$$|\lambda| \leq \|\mathbf{A}\| \quad \text{und damit} \quad \rho(\mathbf{A}) \leq \|\mathbf{A}\|. \quad (5.147)$$

Satz 5.17. Satz von Gershgorin: Sei

$$K_i := \{z \in \mathbb{C} \mid |z - a_{ii}| \leq r_i := \sum_{\substack{k=1 \\ k \neq i}}^n |a_{ik}|\}. \quad (5.148)$$

Dann gilt:

- (i) Jeder Eigenwert von \mathbf{A} liegt in $\bigcup_{i=1}^n K_i$.
- (ii) Sei $J \subset \{1, \dots, n\}$ eine Indexmenge, für die gilt: $|J| = m < n$, sei $K = \bigcup_{i \in J} K_i$ und es gelte $K \cap \bigcup_{i \notin J} K_i = \emptyset$.

Dann enthält K genau m Eigenwerte von \mathbf{A} unter Berücksichtigung ihrer Vielfachheiten.

In (5.148) können Zeilensummen durch Spaltensummen ersetzt werden, da

$$\sigma(\mathbf{A}) = \sigma(\mathbf{A}^T).$$

Beweis. (i)

Sei $\lambda \in \sigma(\mathbf{A})$, \mathbf{x} zugehöriger Eigenvektor: $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. Das lautet komponentenweise

$$(\lambda - a_{ii})x_i = \sum_{\substack{k=1 \\ i \neq k}}^n a_{ik}x_k, \quad i = 1, \dots, n,$$

$$\Rightarrow |\lambda - a_{ii}| |x_i| \leq r_i \max_{i \neq k} |x_k|, \quad i = 1, \dots, n.$$

Wähle jetzt ein i , für das gilt

$$|x_i| := \max_{k=1, \dots, n} |x_k| \Rightarrow |x_i| \geq \max_{k \neq i} |x_k|$$

$$\Rightarrow |\lambda - a_{ii}| \leq r_i, \Rightarrow \lambda \in K_i \Rightarrow \lambda \in \bigcup K_i.$$

D.h. für jedes Eigenpaar (λ, \mathbf{x}) gibt es mindestens ein i mit $|\lambda - a_{ii}| \leq r_i$. Der Index i hängt von der Betrags-maximalen Komponente x_i des Eigenvektors \mathbf{x} ab.

(ii)

Sei $\mathbf{D} = \text{diag}(a_{11}, a_{22}, \dots, a_{nn})$, $\mathbf{E} := \mathbf{A} - \mathbf{D}$. Sei weiter

$$\mathbf{A}(\varepsilon) := \mathbf{D} + \varepsilon \mathbf{E} \quad (\mathbf{A}(1) = \mathbf{A}),$$

und $p(\lambda, \varepsilon) := \det(\mathbf{A}(\varepsilon) - \lambda \mathbf{I})$ das charakteristische Polynom von $\mathbf{A}(\varepsilon)$. Die Koeffizienten von p sind offensichtlich stetig von ε abhängig, damit auch die Nullstellen $\lambda_i(\varepsilon)$. Also bilden die Werte $\lambda_i(\varepsilon)$, $1 \leq i \leq n$, eine zusammenhängende Kurve in \mathbb{C} . Für $\varepsilon = 0$ ist $\mathbf{A}(0) = \mathbf{D}$, d.h. $\lambda = a_{ii}$ und der Kreis K_i schrumpft zum Punkt a_{ii} zusammen. Die n Eigenwerte λ_i zu der Indexmenge, die K bildet, bleiben in den entsprechenden Kreisen $K_i, i \in J$, mit wachsendem $\varepsilon > 0$ und können K nicht verlassen, da $K(\varepsilon)$ mit $\bigcup_{i \notin J} K_i(\varepsilon)$ disjunkt bleibt

$\forall \varepsilon \leq 1$. Also gilt $\lambda_i(1) \in K$ für $i \in J$. \square

Beispiel 5.10. Gegeben sei die symmetrische Tridiagonalmatrix

$$\mathbf{A} = \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{pmatrix}.$$

\mathbf{A} besitzt nur reelle Eigenwerte und diese liegen nach dem Satz von Gershgorin alle im Intervall $[2, 6]$, das ja die reelle Teilmenge von $\bigcup_{i=1}^n K_i$ ist.

Nun sind aber $(\lambda_i)^{-1}$ die Eigenwerte von \mathbf{A}^{-1} , also gilt für deren Spektrum

$$\sigma(\mathbf{A}^{-1}) \subset \left[\frac{1}{6}, \frac{1}{2} \right].$$

Mit diesen beiden Aussagen kann jetzt die Kondition der Matrix \mathbf{A} zur Lösung des linearen Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ abgeschätzt werden:

$$\text{cond}_2(\mathbf{A}) \leq 3$$

Also ist das Problem gut konditioniert. \triangle

Satz 5.18. Die Matrix \mathbf{A} sei diagonalähnlich, $\|\cdot\|$ sei eine zugeordnete Norm und \mathbf{T} die Modalmatrix von \mathbf{A} :

$$\mathbf{T}^{-1} \mathbf{AT} = \mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n). \quad (5.149)$$

Sei $\lambda \in \sigma(\mathbf{A} + \mathbf{E})$ ein Eigenwert der gestörten Matrix $\mathbf{A} + \mathbf{E}$.

Dann gilt

$$\min_{\lambda_i \in \sigma(\mathbf{A})} |\lambda_i - \lambda| \leq \|\mathbf{T}\| \|\mathbf{T}^{-1}\| \|\mathbf{E}\| \quad (5.150)$$

Beweis. Sei $\lambda \neq \lambda_i$ für alle i , sonst ist (5.150) trivial.

Sei \mathbf{x} Eigenvektor zu λ von $\mathbf{A} + \mathbf{E}$:

$$\begin{aligned} (\mathbf{A} + \mathbf{E})\mathbf{x} &= \lambda\mathbf{x} \Rightarrow (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = -\mathbf{E}\mathbf{x} \\ \Rightarrow (\mathbf{TDT}^{-1} - \lambda\mathbf{I})\mathbf{x} &= -\mathbf{E}\mathbf{x} \\ \Rightarrow \mathbf{T}(\mathbf{D} - \lambda\mathbf{I})\mathbf{T}^{-1}\mathbf{x} &= -\mathbf{E}\mathbf{x} \\ \Rightarrow (\mathbf{D} - \lambda\mathbf{I})\mathbf{T}^{-1}\mathbf{x} &= -\mathbf{T}^{-1}\mathbf{E}\underbrace{\mathbf{T}\mathbf{T}^{-1}}_{\mathbf{I}}\mathbf{x}. \end{aligned}$$

Wegen $\lambda \neq \lambda_i$ ist $(\mathbf{D} - \lambda\mathbf{I})$ regulär:

$$\begin{aligned} (\mathbf{D} - \lambda\mathbf{I})^{-1} &= \text{diag}\{(\lambda_1 - \lambda)^{-1}, \dots, (\lambda_n - \lambda)^{-1}\} \\ \Rightarrow \mathbf{T}^{-1}\mathbf{x} &= -(\mathbf{D} - \lambda\mathbf{I})^{-1}(\mathbf{T}^{-1}\mathbf{E}\mathbf{T})\mathbf{T}^{-1}\mathbf{x} \\ \Rightarrow \|\mathbf{T}^{-1}\mathbf{x}\| &\leq \|(\mathbf{D} - \lambda\mathbf{I})^{-1}\| \|\mathbf{T}^{-1}\mathbf{E}\mathbf{T}\| \|\mathbf{T}^{-1}\mathbf{x}\| \\ \Rightarrow 1 &\leq \max_i |(\lambda_i - \lambda)^{-1}| \|\mathbf{T}^{-1}\| \|\mathbf{E}\| \|\mathbf{T}\| \\ \Rightarrow \min_i |\lambda_i - \lambda| &\leq \|\mathbf{T}^{-1}\| \|\mathbf{E}\| \|\mathbf{T}\|. \end{aligned}$$

□

Es ist also folgende Definition sinnvoll:

Definition 5.19. Sei \mathbf{A} diagonalähnlich, \mathbf{T} Modalmatrix von \mathbf{A} . Dann heißt

$$\text{EW-cond}(\mathbf{A}) := \|\mathbf{T}^{-1}\| \|\mathbf{T}\| \quad (= \text{cond}(\mathbf{T})) \quad (5.151)$$

Konditionszahl des Eigenwertproblems von \mathbf{A} .

Lemma 5.20. Wenn \mathbf{A} eine symmetrische Matrix ist, dann ist ihre Modalmatrix orthogonal, und es folgt mit (5.151) aus (5.150)

$$\text{EW-cond}_2(\mathbf{A}) = 1, \implies \min_{\lambda_i \in \sigma(\mathbf{A})} |\lambda_i - \lambda| \leq \|\mathbf{E}\|_2 \quad (5.152)$$

Für eine symmetrische Matrix ist daher das Eigenwertproblem immer gut konditioniert.

Die Abschätzungen (5.150) und (5.152) sind Aussagen über den absoluten Fehler der Eigenwerte auf Grund von Datenfehlern in der Matrix \mathbf{A} . Bei stark unterschiedlich großen Eigenwerten bewirken diese Datenfehler einen größeren relativen Fehler bei den kleineren Eigenwerten. Hinzu kommen Rundungsfehlereinfüsse, deshalb sind numerisch stabile Verfahren äußerst wichtig.

Beispiel 5.11. Hilbert-Matrix, $n = 3$

$$\mathbf{A} = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}$$

$$\mathbf{A} + \mathbf{E} = \begin{pmatrix} 1.000 & 0.5000 & 0.3333 \\ 0.5000 & 0.3333 & 0.2500 \\ 0.3333 & 0.2500 & 0.2000 \end{pmatrix}$$

Also

$$\mathbf{E} = \begin{pmatrix} 0 & 0 & -0.0000\bar{3} \\ 0 & -0.0000\bar{3} & 0 \\ -0.0000\bar{3} & 0 & 0 \end{pmatrix}, \|\mathbf{E}\|_2 = 0.0000\bar{3}$$

$$\begin{aligned} \sigma(\mathbf{A}) &\doteq \{1.408319, 0.1223271, 0.002687340\} \\ \sigma(\mathbf{A} + \mathbf{E}) &\doteq \{1.408294, 0.1223415, 0.002664489\} \\ \lambda_i - \tilde{\lambda}_i &\doteq 0.000025, -0.000014, 0.000023 \end{aligned}$$

d.h. die absoluten Fehler liegen alle in derselben Größenordnung wie $\|\mathbf{E}\|_2$.

Für die relativen Fehler gilt das natürlich nicht:

$$\frac{\lambda_i - \tilde{\lambda}_i}{\lambda_i} = 0.000018, -0.00012, 0.0085.$$

△

Lemma 5.21. Für eine orthogonale Matrix \mathbf{U} gilt

$$\|\mathbf{U}\mathbf{x}\|_2 = \|\mathbf{x}\|_2. \quad (5.153)$$

Satz 5.22. Sei \mathbf{A} symmetrisch, (λ, \mathbf{x}) eine Eigenpaar-Näherung mit $\|\mathbf{x}\|_2 = 1$ und dem Residuum $\boldsymbol{\eta} := \mathbf{A}\mathbf{x} - \lambda\mathbf{x}$.

Dann gilt:

$$\min_{\lambda_i \in \sigma(\mathbf{A})} |\lambda - \lambda_i| \leq \|\boldsymbol{\eta}\|_2 \quad (5.154)$$

Beweis. Sei $\lambda \neq \lambda_i$ für $i = 1, \dots, n$, sonst ist der Beweis trivial. Da \mathbf{A} symmetrisch ist, gibt es nach Satz 5.3 eine orthogonale Matrix \mathbf{U} derart, dass

$$\mathbf{U}^T \mathbf{A} \mathbf{U} = \text{diag}(\lambda_1, \dots, \lambda_n) =: \mathbf{D}.$$

Damit folgt

$$\mathbf{U}^T \boldsymbol{\eta} = \mathbf{U}^T \mathbf{A} \mathbf{x} - \lambda \mathbf{U}^T \mathbf{x} = \mathbf{U}^T \mathbf{U} \mathbf{D} \mathbf{U}^T \mathbf{x} - \lambda \mathbf{U}^T \mathbf{x} = (\mathbf{D} - \lambda \mathbf{I}) \mathbf{U}^T \mathbf{x}.$$

Wegen $\lambda \neq \lambda_i$ ist also $(\mathbf{D} - \lambda \mathbf{I})^{-1} \mathbf{U}^T \boldsymbol{\eta} = \mathbf{U}^T \mathbf{x}$. Mit $\|\mathbf{x}\|_2 = 1$ und Lemma 5.21 folgt

$$\begin{aligned} 1 &= \|\mathbf{x}\|_2 = \|\mathbf{U}^T \mathbf{x}\|_2 = \|(\mathbf{D} - \lambda \mathbf{I})^{-1} \mathbf{U}^T \boldsymbol{\eta}\|_2 \leq \|(\mathbf{D} - \lambda \mathbf{I})^{-1}\|_2 \|\boldsymbol{\eta}\|_2 \\ &= \max_{\lambda_i \in \sigma(\mathbf{A})} \frac{1}{|\lambda - \lambda_i|} \|\boldsymbol{\eta}\|_2 = \frac{1}{\min_{\lambda_i \in \sigma(\mathbf{A})} |\lambda - \lambda_i|} \|\boldsymbol{\eta}\|_2 \end{aligned}$$

Daraus folgt

$$\min_{\lambda_i \in \sigma(\mathbf{A})} |\lambda - \lambda_i| \leq \|\boldsymbol{\eta}\|_2.$$

□

Viele numerische Verfahren zur Eigenwertbestimmung arbeiten mit orthogonalen Ähnlichkeitstransformationen. Deshalb ist es wichtig, dass dadurch die Kondition des Eigenwertproblems nicht verschlechtert wird. Das zeigt das folgende Lemma.

Lemma 5.23. Sei \mathbf{A} diagonalähnlich, seien weiter \mathbf{U} eine orthogonale Matrix, λ ein einfacher Eigenwert von \mathbf{A} und \mathbf{x} der zugehörige Eigenvektor mit $\|\mathbf{x}\|_2 = 1$. Dann gilt:

1. λ ist auch einfacher Eigenwert von $\mathbf{U}^T \mathbf{A} \mathbf{U}$ und $\mathbf{y} = \mathbf{U}^T \mathbf{x}$ Eigenvektor zu λ von $\mathbf{U}^T \mathbf{A} \mathbf{U}$ und es gilt $\|\mathbf{y}\|_2 = 1$.

2.

$$EW\text{-}cond_2(\mathbf{A}) = EW\text{-}cond_2(\mathbf{U}^T \mathbf{A} \mathbf{U})$$

Beweis. Übung. □

5.8 Anwendung: Membranschwingungen

Wir betrachten ein Gebiet G in der Ebene \mathbb{R}^2 , in welchem eine ideal elastische Membran liegen soll, die am Rand dieses Gebietes fest eingespannt ist. Wir wollen die beiden kleinsten Eigenfrequenzen dieser Membran angenähert berechnen. Wenn wir uns die Membran als ein über eine Trommel gespanntes Fell vorstellen, dann entsprechen diese Eigenfrequenzen dem Grund- und dem ersten Oberton. Zur Berechnung diskretisieren wir die zugehörige partielle Differenzialgleichung

$$\begin{aligned} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} &= \lambda u \quad \text{für } (x, y) \in G, \\ u(x, y) &= 0 \quad \text{für } (x, y) \in \Gamma. \end{aligned} \tag{5.155}$$

Dabei ist G das Gebiet der Membran, Γ sein Rand. Als Kochrezept für die Diskretisierung kann folgende Vorgehensweise gewählt werden (Einzelheiten findet man in Kapitel 10):

Die inneren Punkte der Diskretisierung werden durchnummieriert, z.B. von links nach rechts und von oben nach unten wie in unserem Beispiel, siehe Abb. 5.3. Jeder innere Punkt bestimmt die Elemente einer Zeile von \mathbf{A} . Wenn seine Nummer k ist mit den inneren Nachbarknoten p, q, r und s , dann bekommen wir den so genannten *Differenzenstern*:

$a_{kk} = 4$	q
$a_{kp} = a_{kq} = a_{kr} = a_{ks} = -1$	r
$a_{kj} = 0$, sonst.	k
	p
	s

Nachbarknoten auf dem Rand werden wegen der Null-Randwerte in (5.155) nicht berücksichtigt. Sie bekommen auch keine Nummer zugeteilt. Die Matrix wird dann noch durch h^2 , das Quadrat der Gitterweite, dividiert.

Als Beispiel wollen wir das Gebiet aus Abb. 5.3 nehmen. Die dort dargestellte Diskretisierung (Gitterweite $h = 1$) führt auf das Eigenwertproblem $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ mit der in (5.156) symbolisch dargestellten Matrix \mathbf{A} (mit $\times \equiv -1$, $+$ $\equiv 4$ und $\cdot \equiv 0$).

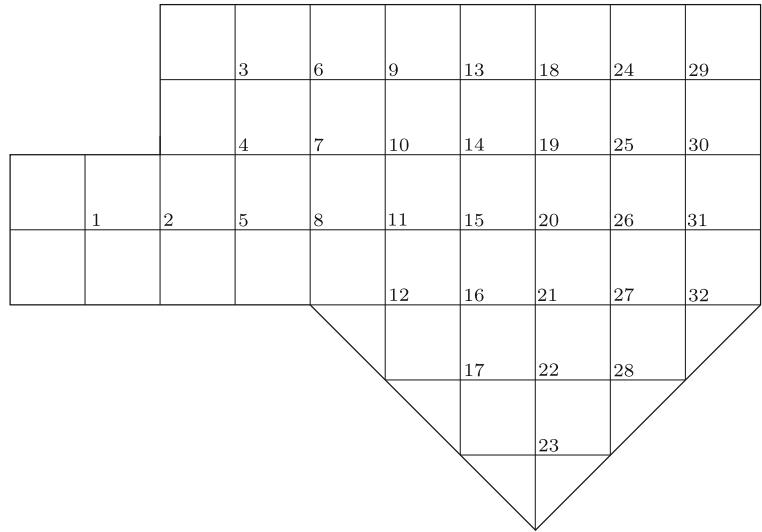


Abb. 5.3 Eine grob diskretisierte Membran.

Wir haben das Beispiel für drei Gitter mit $h = 1$, $h = 1/2$ und $h = 1/4$ durchgerechnet. Das ergibt schwach besetzte Matrizen mit den Ordnungen $n = 32$ für $h = 1$ wie in (5.156), $n = 153$ für $h = 1/2$ und $n = 665$ für $h = 1/4$.

Die folgende Tabelle gibt eine Übersicht über die drei gerechneten Beispiele:

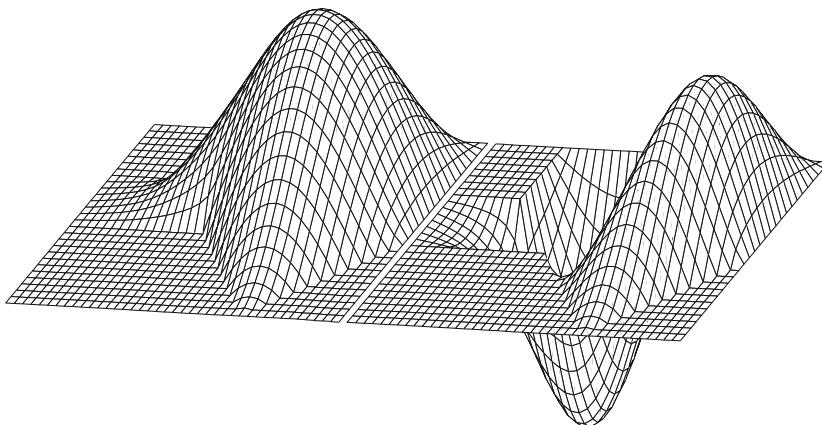


Abb. 5.4 Eigenschwingungsformen einer Membran.

h	Ordnung n	Anzahl Matrixelemente $\neq 0$	λ_1	λ_2
1	32	130	0.5119	1.0001
1/2	153	701	0.5220	1.0344
1/4	665	3193	0.5243	1.0406

Die Eigenwerte ändern sich noch stark mit kleiner werdender Gitterweite; also ist die Genauigkeit der Eigenwerte noch nicht zufriedenstellend. Für eine höhere Genauigkeit wäre eine weitere Verfeinerung des Gitters notwendig gewesen. Die Eigenfunktionen zu diesen zwei Eigenwerten sehen wir in Abb. 5.4. Dargestellt sind die linear verbundenen diskreten Werte zu $h = 1/4$.

5.9 Software

Über die historische Quelle aller Software zur numerischen linearen Algebra und ihre Weiterentwicklungen haben wir schon in Abschnitt 2.6 berichtet. Über diese wichtigsten Software-Quellen hinaus gibt es spezielle Software im Wesentlichen im Zusammenhang mit der Lösung von Eigenwertproblemen bei gewöhnlichen und partiellen Differentialgleichungen, siehe Kapitel 8 und 10.

Die NAG-Bibliotheken enthalten Routinen für alle wichtigen Fälle im Kapitel F02. Dazu gehört auch die Berechnung von Singulärwerten und -vektoren. Das gilt entsprechend für MATLAB. Ein spezielles FORTRAN-Paket für Eigenwertprobleme von großen, schwach besetzten Matrizen ist ARPACK. Es benutzt ein iteratives Verfahren von Arnoldi, [Leh 96, Leh 98]. Der MATLAB-Befehl `eigs` verwendet dieses Paket. Darüber hinaus verfügt MATLAB über einige Befehle zur Berechnung von Eigen- und Singulärwerten und -vektoren von schwach besetzten Matrizen, siehe auch Abschnitt 11.7.

Eine Gruppe von Autoren, darunter einige der LAPACK-Autoren, hat so genannte Templates für Eigenwertprobleme [Bai 00] entwickelt. *Templates*, deutsch auch *Masken* genannt, sind makrosprachlich beschriebene Algorithmen. Vorbildliche Templates beschreiben neben dem Algorithmus die Fälle für seinen effizienten Einsatz, die notwendigen Eingabeparameter, eine Schätzung der Rechenzeit und des Speicherplatzbedarfs sowie alternative Verfahren. Hinzu sollten Beispiele für einfache und schwierige Fälle kommen, Hinweise auf Implementierungen in verschiedenen Programmiersprachen und Literaturhinweise.

Unsere Problemlösungsumgebung PAN (<http://www.upb.de/SchwarzKoeckler/>) verfügt über vier Programme zur Lösung von nicht notwendig symmetrischen, von symmetrischen, von symmetrischen, schwach besetzten und von allgemeinen Eigenwertproblemen.

5.10 Aufgaben

5.1. Die Matrix

$$\mathbf{A} = \begin{pmatrix} 1 & -6 \\ -6 & -4 \end{pmatrix}$$

soll durch eine Jacobi-Rotation auf Diagonalgestalt transformiert werden. Welches sind die Eigenwerte und Eigenvektoren?

5.2. An der symmetrischen Matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 3 & -4 \\ 3 & 6 & 2 \\ -4 & 2 & 10 \end{pmatrix}$$

führt man einen Schritt des klassischen Jacobi-Verfahrens aus. Um welchen Wert nimmt $S(\mathbf{A})$ ab? Die Matrix \mathbf{A} ist sodann mit dem Verfahren von Givens auf tridiagonale Form zu transformieren. Warum entsteht im Vergleich zum vorhergehenden Ergebnis eine andere tridiagonale Matrix? Wie groß ist in diesem Fall die Abnahme von $S(\mathbf{A})$? Zur Transformation von \mathbf{A} auf tridiagonale Gestalt wende man auch noch die schnelle Givens-Transformation an.

5.3. Programmieren Sie die wichtigsten Algorithmen dieses Kapitels. Bestimmen Sie mit dem klassischen Jacobi-Verfahren, dann mit Hilfe der Transformation auf tridiagonale Form und mit dem QR-Algorithmus sowie inverser Vektoriteration die Eigenwerte und Eigenvektoren von folgenden symmetrischen Matrizen.

$$\mathbf{A}_1 = \begin{pmatrix} 3 & -2 & 4 & 5 \\ -2 & 7 & 3 & 8 \\ 4 & 3 & 10 & 1 \\ 5 & 8 & 1 & 6 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 5 & -5 & 5 & 0 \\ -5 & 16 & -8 & 7 \\ 5 & -8 & 16 & 7 \\ 0 & 7 & 7 & 21 \end{pmatrix}$$

$$\mathbf{A}_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 262 \end{pmatrix}, \quad \mathbf{A}_4 = \begin{pmatrix} 19 & 5 & -12 & 6 & 7 & 16 \\ 5 & 13 & 9 & -18 & 12 & 4 \\ -12 & 9 & 18 & 4 & 6 & 14 \\ 6 & -18 & 4 & 19 & 2 & -16 \\ 7 & 12 & 6 & 2 & 5 & 15 \\ 16 & 4 & 14 & -16 & 15 & 13 \end{pmatrix}$$

5.4. Für die Hessenberg-Matrix

$$\mathbf{H} = \begin{pmatrix} 2 & -5 & -13 & -25 \\ 4 & 13 & 29 & 60 \\ 0 & -2 & -15 & -48 \\ 0 & 0 & 5 & 18 \end{pmatrix}$$

berechne man mit der inversen Vektoriteration zwei reelle Eigenwerte und die zugehörigen Eigenvektoren. Dazu verwende man einmal den Startwert $\lambda^{(0)} = 2.5$, im anderen Fall den Startwert $\lambda^{(0)} = 7.5$.

5.5. Man bestimme mit der Transformation auf Hessenberg-Form, der QR -Transformation und der inversen Vektoriteration die Eigenwerte und Eigenvektoren von folgenden unsymmetrischen Matrizen.

$$\mathbf{A}_1 = \begin{pmatrix} -3 & 9 & 0 & 1 \\ 1 & 6 & 0 & 0 \\ -23 & 23 & 4 & 3 \\ -12 & 15 & 1 & 3 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix},$$

$$\mathbf{A}_3 = \begin{pmatrix} 28 & 17 & -16 & 11 & 9 & -2 & -27 \\ -1 & 29 & 7 & -6 & -2 & 28 & 1 \\ -11 & -1 & 12 & 3 & -8 & 10 & 11 \\ -6 & -11 & 12 & 8 & -12 & -5 & 6 \\ -3 & 1 & -4 & 3 & 8 & 4 & 3 \\ 14 & 16 & -7 & 6 & 2 & 4 & -14 \\ -37 & -18 & -9 & 5 & 7 & 26 & 38 \end{pmatrix},$$

$$\mathbf{A}_4 = \begin{pmatrix} 3 & -5 & 4 & -2 & 0 & 8 & 1 \\ 4 & 2 & -1 & 7 & 6 & 2 & 9 \\ -5 & 8 & -2 & 3 & 1 & 4 & 2 \\ -6 & -4 & 2 & 5 & -8 & 1 & -3 \\ 1 & -2 & 7 & 5 & 2 & 8 & 4 \\ 8 & 1 & -7 & 6 & 4 & 0 & -1 \\ -1 & -8 & -9 & -1 & 3 & -3 & 2 \end{pmatrix}.$$

Die Eigenwerte der Matrizen \mathbf{A}_1 und \mathbf{A}_3 sind sämtlich reell.

5.6. Es sei \mathbf{H} eine nichtreduzierte Hessenberg-Matrix der Ordnung n . Man zeige mit Hilfe der Transformationsformeln der einzelnen Schritte, dass in der QR -Zerlegung $\mathbf{H} = \mathbf{Q}\mathbf{R}$ für die Diagonalelemente von \mathbf{R}

$$|r_{ii}| \geq |h_{i+1,i}| \quad \text{für } i = 1, 2, \dots, n-1$$

gilt. Falls $\bar{\lambda}$ eine gute Näherung eines Eigenwertes von \mathbf{H} ist, folgere man daraus, dass in der QR -Zerlegung von $(\mathbf{H} - \bar{\lambda}\mathbf{I}) = \mathbf{Q}\mathbf{R}$ höchstens r_{nn} betragsmäßig sehr klein sein kann.

5.7. Für die Matrizen der Aufgaben 5.3 und 5.5 untersuche man experimentell den Einfluss der reellen Spektralverschiebungen σ_k nach (5.103) oder (5.109) auf die Zahl der Iterationsschritte des QR -Algorithmus.

5.8. Man entwickle den Algorithmus für einen QR -Schritt mit impliziter, reeller Spektralverschiebung σ für eine Hessenberg-Matrix und schreibe dazu ein Programm.

5.9. Man beweise Lemma 5.23.

5.10. Man benutze Satz 5.22, um Abschätzungen $|\lambda - \lambda_i|$ für alle drei Eigenwerte der Matrix

$$\mathbf{A} = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 3.5 & 2 \\ 2 & 2 & 4 \end{pmatrix}$$

zu finden. Für die Modalmatrix \mathbf{T} von \mathbf{A} liegt eine Näherung vor:

$$\tilde{\mathbf{T}} = \begin{pmatrix} -0.58 & -0.62 & 0.48 \\ -0.36 & 0.77 & 0.55 \\ 0.73 & -0.15 & 0.68 \end{pmatrix}.$$

5.11. Der Klang der Trommel

Man bestimme wie in der Anwendung 5.8 die Matrix (der Ordnung $n = 21$) zu der in Abb. 5.5 gezeichneten Trommel.

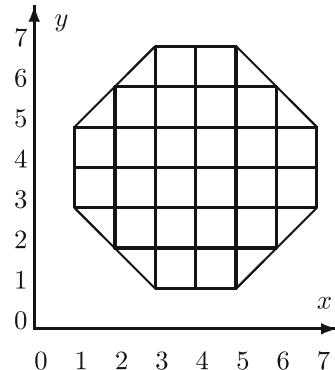


Abb. 5.5

Eine diskretisierte Trommel.

Der Grundton der Trommel ist näherungsweise proportional zum kleinsten Eigenwert von \mathbf{A} . Bei der Lösung des Problems können alle Symmetrien des Problems ausgenutzt werden. Das führt schließlich zu einem Eigenwertproblem der Ordnung $n = 5$.

Schreiben Sie ein Programm, das das Eigenwertproblem für die diskretisierte Trommel für verschiedene Gitterweiten $h_i := 2^{-i}$, $i = 0, 1, 2, \dots$, löst. Wie klein muss die Gitterweite sein, damit der kleinste Eigenwert sich in drei wesentlichen Stellen nicht mehr ändert?