Numerische Verfahren

(für Studierende des Studienganges Bauingieurwesen und Umwelttechnik)

Jens-Peter M. Zemke[1]

Technische Universität Hamburg–Harburg
Arbeitsbereich Mathematik
2005

Inhaltsverzeichnis

Kapitel 1

Einleitung

Aufgabe der Numerischen Mathematik ist, Algorithmen (d.h. Rechenvorschriften) für die näherungsweise numerische Lösung mathematischer Probleme der Naturwissenschaften, Technik, Ökonomie u.s.w. bereitzustellen und zu diskutieren.

Gesichtspunkte bei der Bewertung eines Algorithmus (und beim Vergleich von Algorithmen) sind der Aufwand (z.B. Zahl der Operationen), Speicherplatzbedarf und eine Fehleranalyse.

Man unterscheidet drei Typen von Fehlern nach ihren Quellen:

Datenfehler: Die Eingangsdaten einer Aufgabe können fehlerhaft sein, wenn sie etwa aus vorhergehenden Rechnungen, physikalischen Messungen oder empirischen Untersuchungen stammen.

Verfahrensfehler: Dies sind Fehler, die dadurch entstehen, dass man ein Problem diskretisiert (z.B. eine Differentialgleichung durch eine Differenzengleichung ersetzt) oder ein Iterationsverfahren nach endlich vielen Schritten abbricht.

Rundungsfehler: Bei der Ausführung der Rechenoperationen auf einer Rechenanlage entstehen Fehler, da das Ergebnis (aber auch schon alle Zwischenergebnisse) nur im Rahmen eines begrenzten Zahlbereichs (sog. Maschinenzahlen) dargestellt werden kann, also gerundet werden muss.

Die Frage, wie sich Datenfehler auf die Lösung einer Aufgabe auswirken, nennt man das Konditionsproblem der Aufgabe.

Bewirken kleine Eingangsfehler auch nur kleine Ergebnisfehler, so nennt man das Problem gut konditioniert, anderenfalls schlecht konditioniert. Die Kondition eines Problems hängt nicht nur von der Aufgabenstellung (z.B. „Lösung eines linearen Gleichungssystems“), sondern auch von den Eingangsdaten (z.B. den Koeffizienten der Matrix) ab.

Beispiel 1.1 Das lineare Gleichungssystem

$$\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad a \in \mathbb{R} \tag{1.1}$$

besitzt die Lösung

$$x_0 = 1, \quad y_0 = 0.$$

Das gestörte Gleichungssystem

$$\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ \delta \end{pmatrix}$$

n	vorwärts	rückwärts
0	$9.53101798043249E-0002$	$9.53101798043249E-0002$
1	$4.68982019567514E-0002$	$4.68982019567514E-0002$
2	$3.10179804324860E-0002$	$3.10179804324860E-0002$
3	$2.31535290084733E-0002$	$2.31535290084733E-0002$
4	$1.84647099152673E-0002$	$1.84647099152671E-0002$
5	$1.53529008473272E-0002$	$1.53529008473289E-0002$
6	$1.31376581933944E-0002$	$1.31376581933773E-0002$
7	$1.14805609231986E-0002$	$1.14805609233700E-0002$
8	$1.01943907680135E-0002$	$1.01943907663000E-0002$
9	$9.16720343097581E-0003$	$9.16720344811137E-0003$
10	$8.32796569024192E-0003$	$8.32796551888631E-0003$
11	$7.62943400667172E-0003$	$7.62943572022779E-0003$
12	$7.03899326661614E-0003$	$7.03897613105546E-0003$
13	$6.53314425691553E-0003$	$6.53331561252233E-0003$
14	$6.09712885941609E-0003$	$6.09541530334817E-0003$
15	$5.69537807250574E-0003$	$5.71251363318499E-0003$
16	$5.54621927494255E-0003$	$5.37486366815006E-0003$
17	$3.36133666233920E-0003$	$5.07489273026414E-0003$
18	$2.19421889321635E-0002$	$4.80662825291418E-0003$
19	$-1.66790310374267E-0001$	$4.56529641822660E-0003$
20	$1.71790310374267E+0000$	$4.34703581773402E-0003$
21		$4.14868944170746E-0003$
22		$3.96765103747089E-0003$
23		$3.80175049485636E-0003$
24		$3.64916171810310E-0003$
25		$3.50838281896903E-0003$
26		$3.37771027184820E-0003$
27		$3.25993431855501E-0003$
28		$3.11494252873563E-0003$
29		$3.33333333333333E-0003$
30		$0.00000000000000E+0000$

Tabelle 1.1: Vorwärtige und rückwärtige Rekursion

hat die Lösung

$$x_\delta = 1 - \delta a, \quad y_\delta = \delta.$$

Damit gilt

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \begin{pmatrix} x_\delta \\ y_\delta \end{pmatrix} = \delta \begin{pmatrix} -a \\ 1 \end{pmatrix}.$$

Änderungen der rechten Seite $\begin{pmatrix} 1 & 0 \end{pmatrix}^T$ in der zweiten Komponente werden also mit dem Faktor $\sqrt{1+a^2}$ (bzgl. der Euklidischen Norm) verstärkt. Damit ist das Problem für kleine $|a|$ gut und für große $|a|$ schlecht konditioniert. □

Ein numerisches Verfahren heißt gut konditioniert (numerisch stabil), wenn die gelieferte Lösung eines gegebenen Problems die exakte Lösung eines Problems ist, das aus dem ursprünglichen Problem durch geringe Änderung der Eingangsdaten hervorgeht. Anderenfalls heißt das numerische Verfahren schlecht konditioniert (oder numerisch instabil).

Beispiel 1.2 Das Integral

$$\int_0^1 \frac{x^{20}}{x+10}\,dx$$

kann auf folgende Weise berechnet werden: Für

$$y_n := \int_0^1 \frac{x^n}{x+10}\,dx$$

gilt

$$y_n + 10y_{n-1} = \int_0^1 \frac{x^n + 10x^{n-1}}{x+10}\,dx = \int_0^1 x^{n-1}\,dx = \frac{1}{n}, \tag{1.2}$$

$$y_0 = \int_0^1 \frac{dx}{x+10} = \ln(1.1).$$

Wertet man die Differenzenformel

$$y_n = \frac{1}{n} - 10y_{n-1}$$

für $n = 1, \ldots, 20$ aus, so erhält man die mittlere Spalte der Tabelle 1.1. Obwohl das Problem, das Integral zu berechnen, gut konditioniert ist, erhält man ein unbrauchbares Resultat. Das Verfahren ist also instabil.

Löst man (1.2) nach y_{n-1} auf,

$$y_{n-1} = 0.1\left(\frac{1}{n} - y_n\right),$$

und startet man mit der groben Näherung $y_{30} = 0$, so erhält man $y_{20}, \ldots, y_0$ mit einer Genauigkeit von wenigstens 10 gültigen Stellen, siehe hierzu die letzte Spalte von Tabelle 1.1. □

1.1 Zahlendarstellung

Üblicherweise stellt man Zahlen im Dezimalsystem dar, d.h. eine reelle Zahl x wird durch die Koeffizienten α_i der Dezimaldarstellung von x festgelegt:

$$x = \pm\left(\alpha_n \cdot 10^n + \alpha_{n-1} \cdot 10^{n-1} + \cdots + \alpha_0 \cdot 10^0 + \alpha_{-1} \cdot 10^{-1} + \cdots\right)$$
$$\text{mit} \quad \alpha_i \in \{0, 1, \ldots, 9\}.$$

Abgekürzt schreibt man dafür auch

$$\pm\alpha_n\alpha_{n-1}\ldots\alpha_0.\alpha_{-1}\alpha_{-2}\ldots.$$

Aus technischen Gründen arbeiten digitale Rechenanlagen im Dualsystem (zur Basis 2) oder im Hexadezimalsystem (zur Basis 16). Wir bleiben der Anschauung halber bei der Basis 10.

Für die interne Darstellung einer Zahl in einem Rechner steht nur eine feste Anzahl t (=Wortlänge) von Dezimalstellen zur Verfügung. Diese Wortlänge wird auf zwei Arten zur Darstellung einer Zahl benutzt:

1.1.1 Festpunktdarstellung

Bei der **Festpunktdarstellung** sind n_1 und n_2, die Zahl der Stellen vor und nach dem Dezimalpunkt, festgelegt:

Beispiel 1.3 ($t = 8$, $n_1 = 3$, $n_2 = 5$)

$$\begin{array}{lll|l} 30.411 & \to & 030 & 41100 \\ 0.0023 & \to & 000 & 00230. \end{array}$$

□

Wegen des verhältnismäßig kleinen Bereichs darstellbarer Zahlen wird mit Festpunktzahlen nur dann gearbeitet, wenn keine zu großen Unterschiede in der Größenordnung der auftretenden Zahlen bestehen (vor allem im kaufmännisch-organisatorischen Bereich: Stückzahlen: $n_2 = 0$, Preise: $n_2 = 2$.)

1.1.2 Gleitpunktdarstellung

Schreibt man x in der **Gleitpunktdarstellung**, so liegt die Mantissenlänge $t = n_1 + n_2$ fest; die Lage des Dezimalpunktes wird durch einen Exponenten markiert:

$$\begin{aligned} x &= \pm \left(\alpha_{n_1-1} \cdot 10^{n_1-1} + \alpha_{n_1-2} \cdot 10^{n_1-2} + \cdots + \alpha_0 \cdot 10^0 + \alpha_{-1} \cdot 10^{-1} + \right. \\ &\quad \left. \cdots + \alpha_{-n_2} \cdot 10^{-n_2}\right). \\ &= \pm \left(\alpha_{n_1-1} \cdot 10^{-1} + \alpha_{n_1-2} \cdot 10^{-2} + \cdots + \alpha_{-n_2} \cdot 10^{-(n_1+n_2)}\right) \cdot 10^{n_1} \\ &= \pm 0.\underbrace{\alpha_{n_1-1}\alpha_{n_1-2}\ldots\alpha_{-n_2}}_{\text{Mantisse}} \cdot 10^{n_1}, \quad n_1 : \text{ Exponent} \end{aligned}$$

Beispiel 1.4 ($t = 4$)

$$0.0023 \to 0.0023_{10}0 \quad \text{oder} \quad 0.2300_{10} - 2.$$

□

Die Gleitpunktdarstellung einer Zahl ist i.A. nicht eindeutig. Sie heißt normalisiert, falls $x = 0$ oder für die erste Ziffer $\alpha_1 \neq 0$ gilt. Normalisierte Gleitpunktzahlen ungleich Null sind eindeutig. Daher betrachten wir von nun an nur noch normalisierte Gleitpunktzahlen.

1.2 Rundungsfehler und Gleitpunktrechnung

Die Menge $\mathbb{F}$ (engl. *floating point numbers*) der in einer Maschine darstellbaren Zahlen ist endlich (die Mantissenlänge t ist endlich, und für die Darstellung des Exponenten stehen auch nur $e < \infty$ viele Stellen zur Verfügung).

Für ein gegebenes $x \in \mathbb{R}$ bezeichnen wir mit $\mathrm{fl}(x) \in \mathbb{F}$ eine Maschinenzahl, durch die x am besten approximiert wird, d.h.

$$|\,\mathrm{fl}(x) - x| \leqslant |a - x| \quad \text{für alle} \quad a \in \mathbb{F}.$$

Diese Vorschrift ist noch nicht eindeutig (wird 0.5 auf- oder abgerundet?). Wir setzen fest: Sei $x \in \mathbb{R}$ gegeben mit der normalisierten Gleitpunktdarstellung

$$x = \pm 0.\alpha_1\alpha_2\ldots\alpha_t\alpha_{t+1} \cdot \cdots \cdot 10^n,$$

dann wird x durch die folgende Vorschrift gerundet:

$$\mathrm{fl}(x) = \begin{cases} \pm\ 0.\alpha_1\alpha_2\ldots\alpha_t \cdot 10^n & , \quad \text{falls} \quad 0 \leqslant \alpha_{t+1} \leqslant 4 \\ \pm(0.\alpha_1\alpha_2\ldots\alpha_t + 10^{-t}) \cdot 10^n & , \quad \text{falls} \quad 5 \leqslant \alpha_{t+1} \leqslant 9. \end{cases}$$

Für den absoluten Fehler gilt

$$|x - \mathrm{fl}(x)| \leqslant \frac{1}{2} \cdot 10^{n-t},$$

und für den relativen Fehler

$$\left| \frac{x - \mathrm{fl}(x)}{x} \right| \leqslant \frac{1}{2} \cdot 10^{n-t} 10^{-n+1} = 5 \cdot 10^{-t} \qquad (\alpha_1 \neq 0)\,.$$

Mit der Abkürzung $\mathbf{u} = 5 \cdot 10^{-t}$ (Maschinengenauigkeit) gilt also

$$\mathrm{fl}(x) = (1 + \varepsilon)x, \quad |\varepsilon| \leqslant \mathbf{u}. \tag{1.3}$$

$\mathrm{fl}(x)$ ist nicht stets eine Maschinenzahl, da nicht beliebig große oder kleine Zahlen dargestellt werden können:

Beispiel 1.5 ($t = 4$, $e = 2$)

Exponentenüberlauf:

$$\mathrm{fl}(0.99997_{10}99) = 0.1000_{10}100 \notin \mathbb{F},$$

Exponentenunterlauf:

$$\mathrm{fl}(0.01234_{10} - 99) = 0.1234_{10} - 100 \notin \mathbb{F}. \qquad \square$$

Setzt man im zweiten Fall $\mathrm{fl}(0.01234_{10} - 99) = 0$ oder $0.0123_{10} - 99$, so gilt zwar $\mathrm{fl}(\,\cdot\,) \in \mathbb{F}$, aber es ist nicht mehr (1.3) erfüllt. Da bei den heutigen Anlagen e genügend groß ist, tritt Exponentenüberlauf oder -unterlauf nur sehr selten auf. Wir nehmen daher für das Weitere $e = \infty$ an, so dass bei der Rundung (1.3) gilt.

Offensichtlich gilt

$$x, y \in \mathbb{F} \qquad \not\Rightarrow \qquad x \pm y, x \cdot y, x/y \in \mathbb{F}.$$

Statt der Operationen $+, -, \cdot, /$ sind daher auf dem Rechner als Ersatz die **Gleitpunktoperationen** $\boxplus, \boxminus, \boxdot, \boxslash$ realisiert, die man mit Hilfe von fl so beschreiben kann ($x, y \in \mathbb{F}$):

$$x \circledcirc y := \mathrm{fl}(x \circ y) \quad \forall \circ \in \{+, -, \cdot, /\}.$$

(In der Maschine wird die Operation exakt ausgeführt, danach wird gerundet). Wegen (1.3) gilt

$$x \circledcirc y = (x \circ y)(1 + \varepsilon), \quad |\varepsilon| \leqslant \mathbf{u},$$

für jede der Operationen $\circ \in \{+, -, \cdot, /\}$. (Der relative Fehler hat also die Größenordnung der Maschinengenauigkeit).

Waren aber x und y keine Maschinenzahlen, so wird zunächst gerundet und dann $\mathrm{fl}(x) \circledcirc \mathrm{fl}(y)$ berechnet.

Hierfür gilt wegen $\mathrm{fl}(x) = (1 + \varepsilon_x)x$, $\mathrm{fl}(y) = (1 + \varepsilon_y)y$

$$\frac{\mathrm{fl}(x) + \mathrm{fl}(y) - (x + y)}{x + y} = \frac{x}{x + y}\varepsilon_x + \frac{y}{x + y}\varepsilon_y.$$

Haben also bei der Addition die Summanden entgegengesetztes Vorzeichen, den gleichen Exponenten und gleiche führende Ziffern der Mantisse, so ist $x + y$ klein gegen x und gegen y und der relative Fehler wird verstärkt. (Man spricht dann von Auslöschung).

Beispiel 1.6 ($t = 6$, $x = 1234.567$, $y = -1234.60$) Es gilt

$$\left|\frac{(\mathrm{fl}(x) + \mathrm{fl}(y)) - (x + y)}{x + y}\right| = \left|\frac{-0.03 - (-0.033)}{-0.033}\right| = \frac{1}{11},$$

aber

$$\varepsilon_x = \left|\frac{\mathrm{fl}(x) - x}{x}\right| = \frac{0.003}{1234.567} \approx 2.5 \cdot 10^{-6}, \quad \varepsilon_y = 0. \qquad \square$$

Die Operationen $\cdot$ (und $/$) sind wegen

$$\frac{\mathrm{fl}(x) \cdot \mathrm{fl}(y) - x \cdot y}{x \cdot y} = \varepsilon_x + \varepsilon_y + \varepsilon_x \cdot \varepsilon_y \approx \varepsilon_x + \varepsilon_y$$

für die Fehlerfortpflanzung in einer Rechnung unkritisch.

Kapitel 2

Interpolation

2.1 Problemstellung

Wir betrachten in diesem Kapitel das

Interpolationsproblem: Gegeben seien eine Funktion

$$\Phi\left(x; a_1, \ldots, a_n\right) : \mathbb{R} \supset I \to \mathbb{R},$$

die auf einem Intervall I erklärt ist und von n Parametern $a_1, \ldots, a_n$ abhängt, paarweise verschiedene **Knoten** (oder **Stützstellen**) $x_1, \ldots, x_m \in I$, Vielfachheiten $r_1, \ldots, r_m \in \mathbb{N}$ mit $\sum_{i=1}^{m} r_i = n$ und Werte $y_{ij} \in \mathbb{R}$, $i = 1, \ldots, m$, $j = 0, \ldots, r_i - 1$.

Bestimme die Parameter $a_1, \ldots, a_n$ so, dass die Interpolationsbedingungen

$$\Phi^{(j)}\left(x_i; a_1, \ldots, a_n\right) = y_{ij}, \quad i = 1, \ldots, m, \quad j = 0, \ldots, r_i - 1,$$

erfüllt sind.

Ist Φ linear in den Parametern a_i, d.h.

$$\Phi\left(x; a_1, \ldots, a_n\right) = \sum_{i=1}^{n} a_i \phi_i(x),$$

so heißt das Interpolationsproblem **linear**.

Gilt $r_j = 1$ für alle j, so spricht man von **Lagrange Interpolation**, anderenfalls von **Hermite Interpolation**.

Bemerkung 2.1 Bei der Lagrange Interpolation werden nur Funktionswerte, aber keine Ableitungen vorgeschrieben. Man beachte, dass bei der Hermite Interpolation alle Ableitungen der Ordnung $0, \ldots, r_i - 1$ vorgeschrieben werden. Lässt man Lücken bei den vorgeschriebenen Ableitungen zu, so spricht man von einem **Hermite Birkhoff Interpolationsproblem**. □

Beispiel 2.2 Wir geben eine Liste von prominenten Instanzen des Interpolationsproblemes:

1. Polynominterpolation:

$$\Phi\left(x; a_0, \ldots, a_n\right) = \sum_{j=0}^{n} a_j x^j$$

2. trigonometrische Interpolation:

$$\Phi\left(x; a_0, \ldots, a_{2n}\right) = a_0 + \sum_{j=1}^{n} \left(a_{2j-1} \sin(jx) + a_{2j} \cos(jx)\right)$$

3. rationale Interpolation:

$$\Phi\left(x; a_0, \ldots, a_n, b_0, \ldots b_p\right) = \frac{\sum_{i=0}^{n} a_i x^i}{\sum_{j=0}^{p} b_j x^j}.$$

4. Spline Interpolation:

$$\Phi\left(x; a_1, \ldots, a_n\right) = \sum_{i=1}^{n} a_i \phi_i(x),$$

wobei die ϕ_i auf Teilintervallen von I mit Polynomen übereinstimmen.

Wir werden uns nur mit der Polynominterpolation und der Interpolation mit Splines beschäftigen. Die trigonometrische und die rationale Interpolation werden ausführlich in Braess [10], Stoer [70] und Schwarz [62] behandelt.

Man benötigt die Interpolation zur

1. Bestimmung von Zwischenwerten aus Funktionstafeln (was seit der Verbreitung von elektronischen Taschenrechnern an Bedeutung verloren hat),
2. Herleitung von Formeln zur numerischen Integration,
3. Konvergenzbeschleunigung durch Extrapolation,
4. Numerische Behandlung von gewöhnlichen Differentialgleichungen.

2.2 Polynominterpolation

Wir betrachten in diesem Abschnitt die Interpolation mit Polynomen. Dabei behandeln wir vor allem das Lagrangesche Interpolationsproblem:

> Gegeben seien $n+1$ verschiedene Knoten $x_j \in \mathbb{R}$, $j = 0, \ldots, n$, und $n+1$ nicht notwendig verschiedene Werte $y_j \in \mathbb{R}$.
>
> Gesucht ist ein Polynom p vom Höchstgrade n, so dass gilt
>
> $$p(x_j) = y_j \quad \text{für} \quad j = 0, \ldots, n.$$

Bemerkung 2.3 Die Beweise werden zeigen, dass die Existenz- und Eindeutigkeitsresultate und die Algorithmen zur Berechnung des Interpolationspolynoms ohne Änderungen für *komplexe* Knoten x_j und *komplexe* Daten y_j richtig bleiben. Nur die Fehlerabschätzungen beziehen sich ausschließlich auf *reelle* Probleme. □

Wir bezeichnen mit Π_n die Menge der Polynome von Höchstgrad n (mit reellen oder komplexen Koeffizienten).

2.2.1 Lagrangesche Interpolationsformel

Satz 2.4 (Existenz und Eindeutigkeit) *Zu beliebigen* $n+1$ *Daten* $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, *mit* $x_j \neq x_k$ *für* $j \neq k$ *gibt es* genau ein *Polynom* $p \in \Pi_n$ *mit*

$$p(x_j) = y_j, \quad j = 0, \ldots, n. \tag{2.1}$$

Beweis: Eindeutigkeit: Angenommen es gibt zwei Polynome $p_1, p_2 \in \Pi_n$ mit $p_k(x_j) = y_j$, $j = 0, \ldots, n$, $k = 1, 2$. Dann besitzt das Polynom $p := p_1 - p_2 \in \Pi_n$ die $n+1$ Nullstellen $x_0, \ldots, x_n$, und daher folgt aus dem Fundamentalsatz der Algebra $p \equiv 0$.

Existenz: Die Existenz zeigen wir konstruktiv. Es sei

$$\ell_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \Bigg/ \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i), \quad j = 0, \ldots, n. \tag{2.2}$$

Dann gilt $\ell_j \in \Pi_n$ und $\ell_j(x_k) = \delta_{jk}$ für $j, k = 0, \ldots, n$, und daher erfüllt

$$p(x) := \sum_{j=0}^{n} y_j \ell_j(x) \in \Pi_n \tag{2.3}$$

die Interpolationsbedingungen (2.1). ■

Definition 2.5 Die Darstellung (2.2), (2.3) des Interpolationspolynomes heißt **Lagrangesche Interpolationsformel**.

Bemerkung 2.6 Prinzipiell kann man bei dem Ansatz

$$p(x) = \sum_{j=0}^{n} a_j x^j$$

die Koeffizienten a_j aus dem linearen Gleichungssystem

$$\sum_{j=0}^{n} a_j x_k^j = y_k, \quad k = 0, \ldots, n \tag{2.4}$$

berechnen. Für Gleichungssysteme, deren Koeffizientenmatrix $(x_k^j)_{j,k=0,\ldots,n}$ eine Vandermonde Matrix ist, gibt es spezielle Algorithmen (vgl. Abschnitt 4.5.1 des Skriptes „Grundlagen der Numerischen Mathematik“), die $2.5n^2$ flops erfordern. Der Aufwand der folgenden Algorithmen ist geringer. □

Bemerkung 2.7 MATLAB stellt die Funktionen `p=polyfit(x,y,n)` zur Verfügung, durch die die Koeffizienten p des Ausgleichspolynomes vom Grad n zu den Daten (x, y) bestimmt werden. Ist $\dim x = n+1$, so erhält man das Interpolationspolynom. Mit `y=polyval(p,x)` kann man das Polynom dann an der Stelle x (oder den Stellen $x(j)$, falls x ein Vektor ist) auswerten. □

Bemerkung 2.8 Verwendet man die Lagrangesche Interpolationsformel (2.3) naiv, so benötigt man zur Auswertung von p an einer festen Stelle $\hat{x}$ zur Bereitstellung der $\ell_j(\hat{x})$ aus (2.2) jeweils $4n$ flops (berechne $(\hat{x} - x_i)/(x_j - x_i)$ für $i = 0, \ldots, n$, $i \neq j$, und das Produkt dieser Quotienten) und zur Auswertung von (2.3) weitere $2n$ flops, zusammen also $4n^2 + O(n)$ flops. Dieses kann (vgl. Abschnitt 2.2.1 des Skriptes „Grundlagen der Numerischen Mathematik“) reduziert werden auf $1.5n^2 + O(n)$ flops. □

2.2.2 Die Newtonsche Interpolationsformel

Wir werden nun eine Form des Interpolationspolynomes herleiten, bei der $1.5n(n+1)$ flops zur Vorbereitung benötigt werden und die Auswertung an einer festen Stelle zusätzlich $3n$ flops erfordert. Wir behandeln dazu zunächst die folgende Frage:

Das Polynom $p_n(x) \in \Pi_n$ interpoliere bereits die Daten $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$. Wir nehmen ein weiteres Zahlenpaar $(x_{n+1}, y_{n+1}) \in \mathbb{R}^2$, $x_{n+1} \neq x_j$, $j = 0, \ldots, n$, hinzu. Kann man dann das Interpolationspolynom $p_{n+1}(x) \in \Pi_{n+1}$ zu den Daten (x_j, y_j), $j = 0, \ldots, n+1$, schreiben als

$$p_{n+1}(x) = p_n(x) + f(x),$$

mit einer leicht berechenbaren Funktion $f(x)$?

Wegen $p_n(x) \in \Pi_n$, $p_{n+1}(x) \in \Pi_{n+1}$ gilt $f(x) = p_{n+1}(x) - p_n(x) \in \Pi_{n+1}$, und wegen

$$y_j = p_{n+1}(x_j) = p_n(x_j) + f(x_j) = y_j + f(x_j), \quad j = 0, \ldots, n,$$

gilt $f(x_j) = 0$, $j = 0, \ldots, n$. Daher hat $f(x)$ mit einem $a \in \mathbb{R}$ die Gestalt

$$f(x) = a \prod_{j=0}^{n} (x - x_j).$$

a kann man aus der Interpolationsbedingung

$$y_{n+1} = p_{n+1}(x_{n+1}) = p_n(x_{n+1}) + a \prod_{j=0}^{n} (x_{n+1} - x_j)$$

ermitteln:

$$a = \frac{y_{n+1} - p_n(x_{n+1})}{(x_{n+1} - x_0) \cdot \cdots \cdot (x_{n+1} - x_n)}.$$

Wir wollen nun ein Verfahren zur Berechnung der Zahl a, des führenden Koeffizienten des Interpolationspolynomes, herleiten. Grundlage dafür ist

Satz 2.9 (Aitken Lemma) *Es sei zu $(x_j, y_j) \in \mathbb{R}^2$, $j = 0, \ldots, n$, $x_i \neq x_j$ für $i \neq j$, das Interpolationspolynom gesucht.*

Seien $p_{[0]} \in \Pi_{n-1}$ durch die Interpolationsbedingungen $p_{[0]}(x_j) = y_j$, $j = 0, \ldots, n-1$, festgelegt, und sei $p_{[n]} \in \Pi_{n-1}$ definiert durch $p_{[n]}(x_j) = y_j$, $j = 1, \ldots, n$. Dann gilt

$$p(x) = \frac{p_{[0]}(x)(x - x_n) - p_{[n]}(x)(x - x_0)}{x_0 - x_n}.$$

Beweis: Das angegebene Polynom erfüllt offenbar die Interpolationsbedingungen $p(x_j) = y_j$, $j = 0, \ldots, n$. ■

Für $0 \leqslant i \leqslant j \leqslant n$ sei nun $p_{ij} \in \Pi_{j-i}$ das Interpolationspolynom, das $p_{ij}(x_k) = y_k$, $k = i, \ldots, j$, erfüllt. Dann folgt aus dem Aitken Lemma, dass die p_{ij} rekursiv berechnet werden können durch

$$\begin{aligned} p_{ij}(x) &= \frac{p_{i,j-1}(x)(x - x_j) - p_{i+1,j}(x)(x - x_i)}{x_i - x_j} \\ &= p_{i+1,j}(x) + (p_{i+1,j}(x) - p_{i,j-1}(x)) \frac{x - x_j}{x_j - x_i} \end{aligned} \tag{2.5}$$

Diese Rekursionsformel kann verwendet werden, um den Wert des Interpolationspolynomes an einer festen Stelle x (nicht das Polynom selbst) zu berechnen:

Algorithmus 2.10 (Algorithmus von Neville und Aitken)

```
for j = 0 : n
  t(j) = y(j);
  xj = x - x(j);
  if j > 0
     for i = j-1 : -1 : 0
        t(i) = t(i+1) + (t(i+1) - t(i))*xj /  (x(j) - x(i));
        end
     end
  end
p = t(0);
```

Bemerkung 2.11 Mit dem Algorithmus von Neville und Aitken wird das linke Tableau aufgestellt, das die Werte P_{ij} der interpolierenden Polynome p_{ij} an der festen Stelle x enthält. Das rechte Tableau enthält die Reihenfolge, in der die P_{ij} berechnet werden.

$$\begin{array}{lllllll} x_0 & y_0 = P_{00} & & & & & \\ x_1 & y_1 = P_{11} & P_{01} & & & & \\ x_2 & y_2 = P_{22} & P_{12} & P_{02} & & & \\ x_3 & y_3 = P_{33} & P_{23} & P_{13} & P_{03} & P_{04} = p(x) & \\ x_4 & y_4 = P_{44} & P_{34} & P_{24} & P_{14} & & \end{array} \qquad \begin{array}{llll} 1 & & & \\ 2 & 3 & & \\ 4 & 5 & 6 & \\ 7 & 8 & 9 & 10 \end{array}$$

□

Bemerkung 2.12 Zur Auswertung des Interpolationspolynomes p an einer festen Stelle benötigt man mit dem Algorithmus von Neville und Aitken offenbar $\frac{5}{2}n^2 + O(n)$ flops. □

Man erhält aus der Rekursionsformel (2.5) eine weitere Darstellung des Interpolationspolynomes, die Newtonsche Interpolationsformel. Da a in

$$p_n(x) = p_{n-1}(x) + a \prod_{j=0}^{n-1} (x - x_j)$$

der Koeffizient bei der höchsten Potenz x^n in $p_n(x)$ ist, liest man aus (2.5) sofort ab:

Satz 2.13 (Newtonsche Interpolationsformel) *Das Interpolationspolynom $p \in \Pi_n$ aus (2.1) hat die Gestalt*

$$p(x) = \sum_{j=0}^{n} [x_0, \ldots, x_j] \prod_{k=0}^{j-1} (x - x_k), \tag{2.6}$$

wobei die ***dividierten Differenzen*** $[x_0, \ldots, x_j]$ *rekursiv definiert sind durch*

$$\begin{aligned} [x_j] &:= y_j, \\ [x_k, \ldots, x_j] &:= \frac{[x_{k+1}, \ldots, x_j] - [x_k, \ldots, x_{j-1}]}{x_j - x_k}, \quad j > k \geqslant 0. \end{aligned} \tag{2.7}$$

Die dividierten Differenzen $c_j := [x_0, \ldots, x_j]$ kann man mit folgendem Algorithmus aus den Wertepaaren (x_j, y_j) berechnen:

Algorithmus 2.14 (Dividierte Differenzen)

```
for k = 0 : n
   t(k) = y(k);
   if k > 0
      for i = k-1 : -1 : 0
         t(i) = (t(i+1) - t(i)) / (x(k) - x(i));
         end
      end
   c(k) = t(0);
   end
```

Danach kann man das Interpolationspolynom an jeder gewünschten Stelle $x0$ mit einem Horner-ähnlichen Schema auswerten:

Algorithmus 2.15

```
p = c(n);
for i = n-1 : -1 : 0
   p = p * (x0 - x(i)) + c(i);
   end
```

Bemerkung 2.16 Die $t(k)$ in dem Algorithmus enthalten die folgenden dividierten Differenzen, die in derselben Reihenfolge wie im Algorithmus von Neville und Aitken berechnet werden:

$$\begin{array}{llll} t(0) = y_0 & t(0) = [x_0, x_1] & t(0) = [x_0, x_1, x_2] & t(0) = [x_0, x_1, x_2, x_3] \\ t(1) = y_1 & t(1) = [x_1, x_2] & t(1) = [x_1, x_2, x_3] & \\ t(2) = y_2 & t(2) = [x_2, x_3] & & \\ t(3) = y_3 & & & \end{array}$$

□

Bemerkung 2.17 Man benötigt $\frac{3}{2}n\,(n+1)$ flops zur Berechnung aller c_j, zur Auswertung von p an einer festen Stelle $\hat{x}$ zusätzlich $3n$ flops.

Die Newtonsche Interpolationsformel liefert also die effizienteste Methode zur Auswertung des Interpolationspolynomes. Selbst wenn man nur an einem Funktionswert interessiert ist, ist der Aufwand geringer als mit dem Algorithmus von Neville und Aitken. Wegen seiner einfachen Gestalt wird der Algorithmus von Neville und Aitken dennoch in der Praxis verwendet. □

Bemerkung 2.18 Natürlich erhält man für jede Anordnung der Knoten x_i (bei rundungsfehlerfreier Rechnung) dasselbe Interpolationspolynom $p_n(x)$. Will man $p_n(x)$ nur an einer Stelle x (oder in deren Nähe) auswerten, so kann man den Rundungsfehlereinfluss klein halten, indem man die Knoten so numeriert, dass gilt

$$|x - x_i| \leqslant |x - x_{i+1}|, \qquad i = 0, \ldots, n-1.$$ □

2.2.3 Fehlerbetrachtungen

Wir behandeln nun die Frage, wie gut eine gegebene, auf einem reellen Intervall definierte Funktion f durch das Interpolationspolynom zu vorgegebenen Knoten x_i in I approximiert wird (es sei also $y_i = f(x_i)$).

Beispiel 2.19 Gegeben sei auf dem Intervall $I = [-1, 1]$ die Funktion

$$f(x) = \sin \pi x,$$

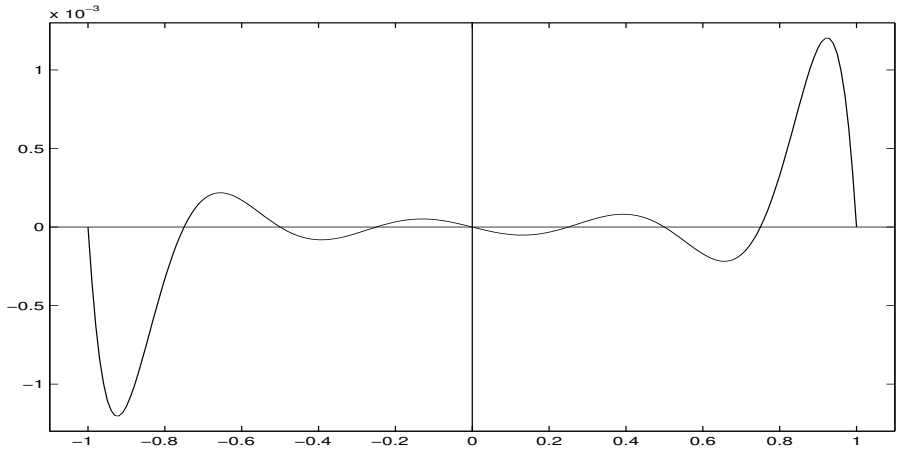

Abbildung 2.1: Fehlerkurve (äquidistante Knoten)

und $p_8(x) \in \Pi_8$ sei das Interpolationspolynom zu den Knoten

$$x_i = -1 + i \cdot 0.25, \quad i = 0, \ldots, 8.$$

Dann erhält man die Fehlerkurve aus Abbildung 2.1. Das Interpolationspolynom liefert also am Rande des Intervalls eine ziemlich schlechte Approximation für f, obwohl f sehr gutartig ist (beliebig oft differenzierbar). □

Das gezeichnete Verhalten ist typisch für die Polynominterpolation mit äquidistanten Knoten bei größerem n. Eine gewisse Erklärung hierfür gibt

Satz 2.20 *Sei $f \in C^{n+1}[a,b]$, seien $x_j \in [a,b]$, $j = 0, \ldots, n$, paarweise verschiedene Knoten, und sei $p \in \Pi_n$ das durch $p(x_j) = f(x_j)$, $j = 0, \ldots, n$, bestimmte Interpolationspolynom. Dann gilt:*

Zu jedem $x \in [a,b]$ existiert $\xi = \xi(x)$ aus dem kleinsten Intervall $I(x, x_0, \ldots, x_n)$, das alle x_j und x enthält, so dass

$$f(x) - p(x) = \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\xi) \tag{2.8}$$

gilt mit

$$\omega(x) = \prod_{i=0}^{n} (x - x_i). \tag{2.9}$$

Bemerkung 2.21 ω hat für äquidistante Knoten x_j die Gestalt von $f - p_8$ der Skizze aus Abbildung 2.1. □

Beweis: Für $x = x_j$, $j = 0, \ldots, n$, ist die Aussage trivial.
Für festes $x \neq x_j$ betrachten wir die Funktion

$$F(z) := f(z) - p(z) - \alpha\omega(z) \tag{2.10}$$

und bestimmen $\alpha \in \mathbb{R}$ so, dass $F(x) = 0$ gilt.

Dann besitzt F in $I(x, x_0, \ldots, x_n)$ wenigstens $n+2$ Nullstellen. Nach dem Satz von Rolle besitzt F' dort wenigstens $n+1$ Nullstellen, F'' wenigstens n Nullstellen, F''' wenigstens $n-1$ Nullstellen, ..., schließlich hat $F^{(n+1)}$ mindestens eine Nullstelle $\xi \in I(x, x_0, \ldots, x_n)$.

Wegen $p \in \Pi_n$ erhält man aus (2.10)

$$F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - 0 - \alpha \cdot (n+1)! = 0.$$

Hiermit folgt wegen $F(x) = 0$ aus (2.10) die Behauptung. ■

Bemerkung 2.22 Aus Satz 2.20 erhält man die folgende Abschätzung für die Güte der Approximation einer Funktion durch das Interpolationspolynom

$$\|f - p\|_\infty := \max_{x \in [a,b]} |f(x) - p(x)| \leqslant \frac{K(f)}{(n+1)!} \|\omega\|_\infty, \tag{2.11}$$

wobei

$$K(f) = \|f^{(n+1)}\|_\infty.$$ □

Bemerkung 2.23 Man erhält ferner aus Satz 2.20 eine Darstellung der dividierten Differenzen.

Nimmt man im Newtonschen Interpolationspolynom in Satz 2.13 den Knoten x als weitere Stützstelle mit dem Wert $f(x)$ hinzu, so erhält man für das in Satz 2.13 definierte Polynom p

$$f(x) - p(x) = [x_n, x_{n-1}, \ldots, x_0, x] \prod_{i=0}^{n} (x - x_i). \tag{2.12}$$

Durch Vergleich mit der Abschätzung (2.8) folgt

$$[x_n, \ldots, x_0, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

und damit allgemein für die $n-$te dividierte Differenz

$$[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!} \quad \text{für ein } \xi \in I\,(x_0, \ldots x_n)\,.$$ □

2.3 Spline Interpolation

Die in den vorhergehenden Abschnitten behandelte Polynominterpolation ist zwar leicht durchführbar, hat aber den Nachteil, dass bei Verfeinerung der Zerlegung keine Konvergenz zu erwarten ist. Bessere Konvergenzeigenschaften haben die nun zu besprechenden Spline-Funktionen.

Definition 2.24 Sei

$$\Delta \;:\; a := x_0 < x_1 < \ldots < x_n \;=:\; b \tag{2.13}$$

eine Zerlegung des Intervalls $[a, b]$. Dann bezeichnen wir mit $S(\Delta, p, q)$, $p, q \in \mathbb{N}_0$, $0 \leqslant q < p$, die Menge aller Funktionen $s \in C^q[a, b]$, die auf jedem Teilintervall $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, mit einem Polynom vom Höchstgrad p übereinstimmen.

Jedes $s \in S(\Delta, p, q)$ heißt (Polynom-)Spline vom Grade p der Differenzierbarkeitsklasse q zur Zerlegung Δ.

Am häufigsten treten in den Anwendungen (auf Randwertaufgaben) die Räume $S(\Delta, 3, 2)$ (**kubische Splines**) und $S(\Delta, 3, 1)$ (**kubische Hermite Splines**) auf.

Daneben untersucht man für spezielle Aufgabenstellungen, bei denen Pole oder verschiedenes Wachstum in verschiedenen Teilen des Intervalls erwartet wird (Grenzschichtprobleme), nichtlineare Splines (rationale oder exponentielle Splines).

Wir behandeln nur $S(\Delta, 3, 2)$, sowie zur Motivation $S(\Delta, 1, 0)$.

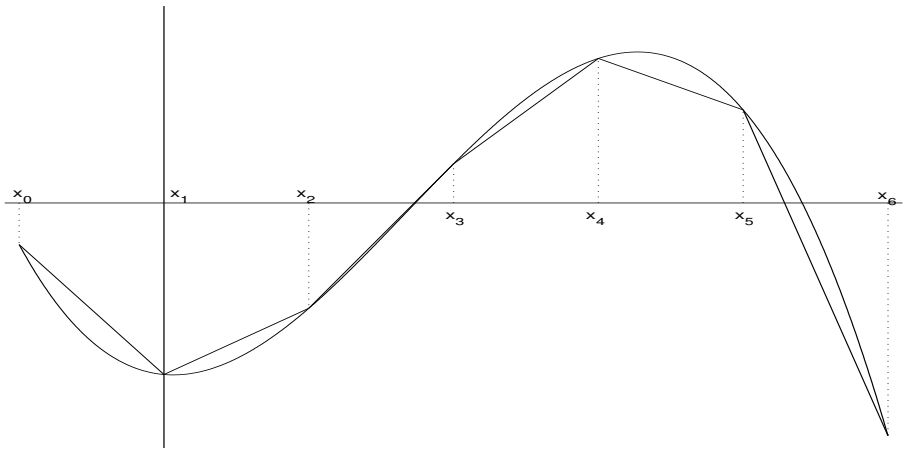

Abbildung 2.2: Stückweise lineare Interpolation

2.3.1 Stückweise lineare Funktionen

Es sei $\Delta : a = x_0 < x_1 < \cdots < x_n = b$ eine gegebene Zerlegung des Intervalls $[a, b]$ und

$$S(\Delta, 1, 0) = \left\{ s \in C[a, b] \,:\, s\,\big|_{[x_{i-1}, x_i]} \in \Pi_1,\ i = 1, \ldots, n \right\}$$

die Menge der stückweise linearen Funktionen auf $[a, b]$ zu dieser Zerlegung.

Für gegebene Interpolationsdaten $y_0, \ldots, y_n \in \mathbb{R}$ gibt es offenbar genau einen interpolierenden Spline $s \in S(\Delta, 1, 0)$ mit $s(x_i) = y_i$, $i = 0, \ldots, n$, und dieses s erhält man, indem man in jedem Teilintervall $[x_{i-1}, x_i]$ die Daten (x_{i-1}, y_{i-1}) und (x_i, y_i) nach Abschnitt 2.2 durch eine lineare Funktion interpoliert.

Man erhält in den Teilintervallen

$$s(x) = \frac{1}{x_i - x_{i-1}} \big(y_i (x - x_{i-1}) + y_{i-1} (x_i - x) \big), \quad x \in [x_{i-1}, x_i]. \tag{2.14}$$

Gilt $y_i = f(x_i)$ für eine Funktion $f \in C^2[a, b]$, so erhalten wir aus Satz 2.20 sofort für $x \in [x_{i-1}, x_i]$ den Fehler

$$f(x) - s(x) = \frac{1}{2} (x - x_{i-1})(x - x_i) f''(\xi_i), \quad \xi_i \in [x_{i-1}, x_i],$$

und daher für $x \in [x_{i-1}, x_i]$

$$|f(x) - s(x)| \leqslant \frac{1}{8} (x_i - x_{i-1})^2 |f''(\xi_i)|.$$

Mit $|\Delta| := \max\limits_{i=1,\ldots,n} (x_i - x_{i-1})$ folgt

$$\|f - s\|_\infty \leqslant \frac{1}{8} |\Delta|^2 \, \|f''\|_\infty. \tag{2.15}$$

Ist also Δ_n irgendeine Zerlegungsfolge von $[a, b]$ mit

$$\lim_{n \to \infty} |\Delta_n| = 0$$

und $f \in C^2[a, b]$, so konvergiert die zugehörige Folge der interpolierenden Splines $s_n \in S(\Delta_n, 1, 0)$ gleichmäßig gegen f, und die Konvergenzgeschwindigkeit wird durch (2.15) beschrieben.

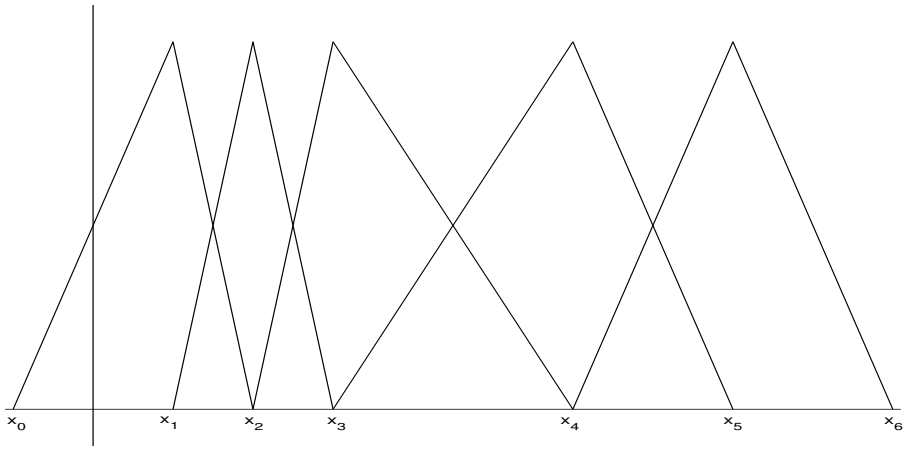

Abbildung 2.3: Dachfunktionen

Für $f \in C^0[a,b]$ erhält man für $x \in [x_{i-1}, x_i]$ aus (2.14)

$$\begin{aligned} |f(x) - s(x)| &= \frac{1}{x_i - x_{i-1}} \Big| (x - x_{i-1})(f(x) - f(x_i)) + (x_i - x)(f(x) - f(x_{i-1})) \Big| \\ &\leqslant \frac{1}{x_i - x_{i-1}} \Big((x - x_{i-1}) |f(x) - f(x_i)| + (x_i - x) |f(x) - f(x_{i-1})| \Big) \\ &\leqslant \max \left(|f(x) - f(x_i)|, |f(x) - f(x_{i-1})| \right), \end{aligned}$$

und daher folgt

$$\|f - s\|_\infty \leqslant \omega\,(f, |\Delta|),$$

wobei

$$\omega\,(f,h) \;:= \sup \{ |f(x) - f(y)| \;:\; x, y \in [a,b],\; |x - y| \leqslant h \}$$

den **Stetigkeitsmodul** von f zur Schrittweite h bezeichnet.

Ist Δ_n eine Zerlegungsfolge von $[a,b]$ mit $\lim_{n\to\infty} |\Delta_n| = 0$, so konvergieren auch für nur stetiges f die interpolierenden linearen Splines gleichmäßig gegen f.

Ähnlich wie bei der Lagrangeschen Interpolation können wir s darstellen als

$$s(x) = \sum_{i=0}^{n} f_i \phi_i(x),\; x \in [a,b]$$

mit den Basisfunktionen

$$\phi_i(x) := \begin{cases} (x - x_{i-1}) \,/\, (x_i - x_{i-1}) & , \quad x \in [x_{i-1}, x_i] \\ (x_{i+1} - x) \,/\, (x_{i+1} - x_i) & , \quad x \in [x_i, x_{i+1}] \\ 0 & , \quad x \notin [x_{i-1}, x_{i+1}] \end{cases} , \quad i = 0, \ldots, n,$$

wobei $x_{-1} < a$ und $x_{n+1} > b$ beliebig gewählt sind.

Die angegebenen ϕ_i heißen **Dachfunktionen** (engl.: hat functions). Sie besitzen einen lokalen Träger $[x_{i-1}, x_{i+1}]$. Will man also s an einer Stelle $x \in (x_{i-1}, x_i)$ auswerten, so hat man wegen $\phi_j(x) = 0$ für $j \notin \{i, i-1\}$ nur $\phi_i(x)$ und $\phi_{i-1}(x)$ zu berechnen (anders als bei den $\ell_j(x)$ in Satz 2.4).

2.3.2 Kubische Splines

Bei den kubischen Splines $s \in S(\Delta, 3, 2)$ sind die Verhältnisse nicht ganz so einfach wie in den Fällen $S(\Delta, 3, 1)$ und $S(\Delta, 1, 0)$, da man die Interpolationsaufgabe nicht in jedem Teilintervall getrennt ausführen kann.

Ein Spline $s \in S(\Delta, 3, 2)$ ist in jedem Teilintervall $[x_{i-1}, x_i]$ ein Polynom dritten Grades, also ist s durch $4n$ Parameter bestimmt. Durch die Forderung $s \in C^2[a, b]$ werden in jedem inneren Knoten x_i, $i = 1, \ldots, n-1$, drei Bedingungen gegeben:

$$s^{(j)}(x_i - 0) = s^{(j)}(x_i + 0), \ j = 0, 1, 2.$$

Daher besitzt ein kubischer Spline noch (wenigstens) $n + 3$ Freiheitsgrade, und wir können nicht erwarten, dass s durch die $n + 1$ Interpolationsbedingungen

$$s(x_i) = y_i, \ i = 0, \ldots, n,$$

festgelegt ist, sondern es müssen noch zwei „Randbedingungen“ hinzugenommen werden. Dieses kann (je nach Aufgabenstellung) auf verschiedene Weise geschehen.

Satz 2.25 *Es sei Δ eine Zerlegung von [a,b], und es seien $y_0, \ldots, y_n \in \mathbb{R}$ gegeben. Dann gibt es für jede der vier folgenden Randbedingungen*

$$\begin{aligned}
&(i) && s'(x_0) = y'_0, \ \ s'(x_n) = y'_n, && (y'_0, y'_n \in \mathbb{R} \text{ gegeben})\\
&(ii) && s'(x_0) = s'(x_n), \ \ s''(x_0) = s''(x_n), &&\\
&(iii) && s''(x_0) = s''(x_n) = 0, &&\\
&(iv) && s''(x_0) = y''_0, \ \ s''(x_n) = y''_n, && (y''_0, y''_n \in \mathbb{R} \text{ gegeben})
\end{aligned}$$

genau einen interpolierenden kubischen Spline $s \in S(\Delta, 3, 2)$ mit

$$s(x_j) = y_j, \quad j = 0, \ldots, n. \tag{2.16}$$

Bemerkung 2.26 Wird die Interpolation mit kubischen Splines verwendet, um eine Funktion $f : [a, b] \to \mathbb{R}$ zu approximieren, so wird man die Randbedingungen (i) verwenden, wenn die Ableitung von f in den Randpunkten a und b des Intervalls bekannt sind, die Randbedingung (ii), wenn die Funktion f periodisch mit der Periode $b-a$ ist, und die Randbedingung (iv), wenn zusätzlich zu den Lagrangeschen Interpolationsdaten am Rand des Intervalls die zweiten Ableitungen bekannt sind. □

Bemerkung 2.27 Erfüllt $s \in S(\Delta, 3, 2)$ die Randbedingung (iii), so kann man s auf $\{x : x < x_0\}$ bzw. $\{x : x > x_n\}$ zweimal stetig differenzierbar als lineare Funktion fortsetzen. Kubische Splines, die man auf diese Weise erhält, heißen **natürliche Splines**. □

Bemerkung 2.28 In de Boor [16] werden vier weitere Randbedingungen diskutiert, unter denen die Existenz und Eindeutigkeit des interpolierenden Splines gesichert ist. Unter ihnen besonders wichtig ist die sog. **not-a-knot Bedingung**. Diese wird verwendet, wenn nichts über das Verhalten der interpolierenden Funktion an den Rändern des Intervalls (außer den Funktionswerten) bekannt ist. Man wählt dann als Knoten für den Spline die Punkte $x_0 < x_2 < \cdots < x_{n-2} < x_n$. Ein Spline zu diesen $n - 2$ Intervallen hat $n + 1$ Freiheitsgrade und ist durch die $n + 1$ Interpolationsbedingungen $s(x_j) = y_j$, $j = 0, \ldots, n$ eindeutig bestimmt. □

Beweis: (von Satz 2.25) Der Beweis ist konstruktiv, er liefert also ein Verfahren zur Berechnung der jeweiligen interpolierenden Splines.

Sei $h_{j+1} := x_{j+1} - x_j$, $j = 0, \ldots, n-1$, und $m_j := s''(x_j)$, $j = 0, \ldots, n$, die zweite Ableitung der gesuchten Splinefunktion an der Stelle x_j.

Wir wollen zeigen, dass der Vektor $\boldsymbol{m} := (m_0, \ldots, m_n)^T$ für jede der vier Randbedingungen eindeutige Lösung eines linearen Gleichungssystems ist und dass man aus den m_j den Spline $s(x)$ an jeder Stelle $x \in [a, b]$ leicht errechnen kann.

s ist in jedem Teilintervall $[x_j, x_{j+1}]$ ein kubisches Polynom. Also ist s'' stückweise linear, und man kann s'' mit Hilfe der m_j so beschreiben (vgl. die Überlegungen in Abschnitt 2.3.1 für $S(\Delta, 1, 0)$).

$$s''(x) = \frac{1}{h_{j+1}} \left(m_j(x_{j+1} - x) + m_{j+1}(x - x_j) \right), \quad x \in [x_j, x_{j+1}].$$

Durch Integration erhält man für $x \in [x_j, x_{j+1}]$, $j = 0, \ldots, n-1$,

$$s'(x) = -m_j \frac{(x_{j+1} - x)^2}{2h_{j+1}} + m_{j+1} \frac{(x - x_j)^2}{2h_{j+1}} + \alpha_j \tag{2.17}$$

und

$$s(x) = m_j \frac{(x_{j+1} - x)^3}{6h_{j+1}} + m_{j+1} \frac{(x - x_j)^3}{6h_{j+1}} + \alpha_j(x - x_j) + \beta_j. \tag{2.18}$$

Die Integrationskonstanten α_j und β_j erhält man aus den Interpolationsbedingungen

$$\begin{aligned} s(x_j) &= m_j \frac{h_{j+1}^2}{6} + \beta_j = y_j \\ s(x_{j+1}) &= m_{j+1} \frac{h_{j+1}^2}{6} + \alpha_j h_{j+1} + \beta_j = y_{j+1}, \end{aligned}$$

d.h.

$$\begin{aligned} \beta_j &= y_j - m_j \frac{h_{j+1}^2}{6} \\ \alpha_j &= \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1} - m_j). \end{aligned} \tag{2.19}$$

Setzt man α_j und β_j aus (2.19) in (2.18) ein, so erhält man die gewünschte Darstellung von s in Abhängigkeit von den m_j.

$n-1$ Bestimmungsgleichungen für die m_j erhält man, indem man die Stetigkeit von s' ausnutzt: Setzt man α_j aus (2.19) in (2.17) ein, so erhält man

$$\begin{aligned} s'(x) = -m_j \frac{(x_{j+1} - x)^2}{2h_{j+1}} + m_{j+1} \frac{(x - x_j)^2}{2h_{j+1}} \\ + \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{h_{j+1}}{6}(m_{j+1} - m_j), \quad x \in [x_j, x_{j+1}]. \end{aligned} \tag{2.20}$$

Also liefert die Forderung $s'(x_j - 0) = s'(x_j + 0)$

$$\frac{y_j - y_{j-1}}{h_j} + \frac{h_j}{3} m_j + \frac{h_j}{6} m_{j-1} = \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{h_{j+1}}{3} m_j - \frac{h_{j+1}}{6} m_{j+1} \ ,$$

d.h. für $j = 1, \ldots, n-1$

$$\frac{h_j}{6} m_{j-1} + \frac{h_j + h_{j+1}}{3} m_j + \frac{h_{j+1}}{6} m_{j+1} = \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{y_j - y_{j-1}}{h_j}. \tag{2.21}$$

Die restlichen beiden Bedingungen für die m_j erhält man aus den Randbedingungen (i), (ii), (iii) oder (iv).

Für die natürlichen Splines hat man in (2.20)

$$s''(a) = m_0 = 0 = m_n = s''(b)$$

zu setzen, im Fall (iv) die inhomogenen Gleichungen

$$m_0 = y_0'' \quad \text{und} \quad m_n = y_n''.$$

Im Fall (i) hat man wegen (2.20) das System (2.21) zu ergänzen durch

$$\begin{aligned} \frac{h_1}{3}m_0 + \frac{h_1}{6}m_1 &= \frac{y_1 - y_0}{h_1} - y_0', \\ \frac{h_n}{6}m_{n-1} + \frac{h_n}{3}m_n &= y_n' - \frac{y_n - y_{n-1}}{h_n}. \end{aligned} \tag{2.22}$$

Im Falle periodischer Randbedingungen gilt $m_0 = m_n$, und wegen $s'(a) = s'(b)$ erhält man aus (2.20)

$$\frac{h_1}{6}m_1 + \frac{h_n}{6}m_{n-1} + \frac{h_n + h_1}{3}m_0 = \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}. \tag{2.23}$$

In jedem Fall ergeben sich also die m_j als Lösung eines linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.24}$$

wobei für alle Zeilen der Matrix $\boldsymbol{A}$ gilt:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| .$$

Die Koeffizientenmatrix ist also strikt diagonaldominant. Nach dem Satz von Gerschgorin ist sie dann regulär, und daher ist das Gleichungssystem (2.24) für jede rechte Seite $\boldsymbol{b}$ eindeutig lösbar. ■

Satz 2.29 *Sei $f \in C^3[a,b]$ und sei $s \in S(\Delta, 3, 2)$ der interpolierende Spline, der eine der Randbedingungen (i) oder (ii) oder $s''(a) = f''(a)$, $s''(b) = f''(b)$ erfüllt.*

Dann gilt mit den Konstanten $C_0 = \frac{9}{8}$ und $C_1 = C_2 = \frac{9}{4}$

$$\|s^{(\nu)} - f^{(\nu)}\|_\infty \leqslant C_\nu |\Delta|^{3-\nu}\ \omega(f''', |\Delta|),\ \ \nu = 0, 1, 2, \tag{2.25}$$

wobei

$$\omega(g, h) := \sup\{|g(x) - g(y)| \ :\ x, y \in [a,b],\ \ |x - y| \leqslant h\}$$

den Stetigkeitsmodul von g bezeichnet.

Genügt f''' einer Lipschitzbedingung

$$|f'''(x) - f'''(y)| \leqslant L|x - y| \quad \text{für alle } x, y \in [a,b],$$

so gilt

$$\|s^{(\nu)} - f^{(\nu)}\|_\infty \leqslant L \cdot C_\nu |\Delta|^{4-\nu}, \quad \nu = 0, 1, 2. \tag{2.26}$$

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 2.3.3. ■

Bemerkung 2.30 Satz 2.29 zeigt, dass Spline Interpolationen geeignet sind, um Ableitungen von Funktionen, von denen nur diskrete Werte bekannt sind, zu approximieren. □

Auch der Raum der kubischen Splines $S(\Delta, 3, 2)$ besitzt eine lokale Basis.

Seien $x_{-3} < x_{-2} < x_{-1} < a$ und $b < x_{n+1} < x_{n+2} < x_{n+3}$ zusätzliche Knoten. Dann überzeugt man sich leicht (aber mit Hilfe einer längeren Rechnung), dass die

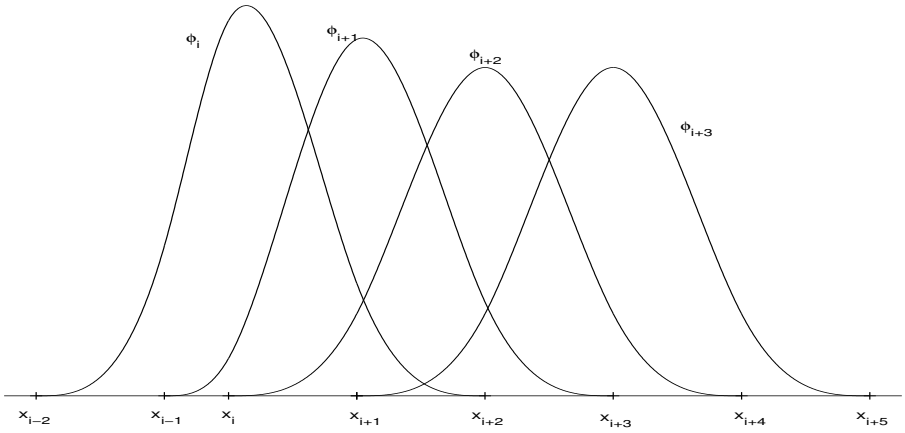

Abbildung 2.4: B-Splines

Funktionen

$$\phi_i(x) = \begin{cases} (x-x_{i-2})_+^3 \Big/ \displaystyle\prod_{j=i-1}^{i+2} (x_j - x_{i-2}) \\ \qquad + (x-x_{i-1})_+^3 \Big/ \displaystyle\prod_{\substack{j=i-2 \\ j\neq i-1}}^{i+2} (x_j - x_{i-1}) \,, & x \leqslant x_i \\ (x_{i+2}-x)_+^3 \Big/ \displaystyle\prod_{j=i-2}^{i+1} (x_{i+2} - x_j) \\ \qquad + (x_{i+1}-x)_+^3 \Big/ \displaystyle\prod_{\substack{j=i-2 \\ j\neq i+1}}^{i+2} (x_{i+1} - x_j) \,, & x \geqslant x_i \end{cases}$$

für $i = -1, \ldots, n+1$, eine Basis von $S(\Delta, 3, 2)$ bilden. Dabei bezeichnet $(x)_+$ die Funktion

$$(x)_+ := \begin{cases} x & \text{für} \quad x \geqslant 0, \\ 0 & \text{für} \quad x \leqslant 0. \end{cases}$$

Die Basisfunktionen ϕ_j heißen **B-Splines**. Abbildung 2.4 enthält die Graphen einiger B-Splines.

Jeder B-Spline ist auf 4 Teilintervallen nichttrivial. Man kann zeigen, dass es keine Basis von $S(\Delta, 3, 2)$ mit kleinerem Träger gibt.

Man kann mit diesen ϕ_i den Ansatz

$$s(x) = \sum_{i=-1}^{n+1} c_i \phi_i(x)$$

machen, und zur Lösung des Interpolationsproblems das lineare Gleichungssystem

$$s(x_i) = y_i + \text{Randbedingungen}$$

behandeln. Dieses besitzt ähnlich wie das für die M_i wieder eine tridiagonale, diagonaldominante Koeffizientenmatrix, ist also problemlos lösbar.

Nicht benutzen sollte man jedoch für numerische Zwecke die folgende Basis von $S(\Delta, 3, 2)$, die bisweilen in Büchern angegeben ist:

$$1, x, x^2, x^3, (x-x_i)_+^3, \ i = 1, \ldots, n-1,$$

wegen der schlechten Kondition des entstehenden linearen Gleichungssystems.

MATLAB stellt die Funktion `yy=spline(x,y,xx)` zur Verfügung. Besitzen die Vektoren x und y gleiche Länge, so wird der interpolierende kubische Spline mit not-a-knot Randbedingungen bestimmt. Ist die Länge von y um 2 größer als die Länge n von x, so werden die erste und letzte Komponente von y als Steigungen von s in $x(1)$ und $x(n)$ gedeutet. Der Vektor yy enthält in der j-ten Komponente $yy(j) = s(xx(j))$.

Zusätzlich wird von MATLAB die Toolbox SPLINES angeboten. Hierin werden auch andere Randbedingungen für $S(\Delta, p, p-1)$, Glättung von Daten durch Ausgleich und Tensorprodukte von Splines zur Darstellung von Flächen angeboten.

Kapitel 3

Numerische Integration

In vielen Fällen ist es nicht möglich, ein gegebenes Integral

$$\int_a^b f(x)\,dx$$

in geschlossener Form auszuwerten; z.B. ist für das in der Statistik häufig auftretende Integral

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2}\,dt$$

keine elementare Stammfunktion angebbar. In diesem Fall ist man auf numerische Verfahren zur Integration, sog. Quadraturverfahren, angewiesen. Wir wollen in diesem Abschnitt einige wichtige Verfahren zur näherungsweisen Berechnung bestimmter Integrale besprechen.

3.1 Konstruktion von Quadraturformeln

Wir betrachten das bestimmte Integral

$$\int_a^b f(x)\,dx$$

mit einer gegebenen integrierbaren Funktion $f : [a,b] \to \mathbb{R}$. Mit der Variablentransformation $x = a + t(b-a)$ erhält man

$$\int_a^b f(x)\,dx = (b-a) \int_0^1 f(a + t(b-a))\,dt,$$

so dass man sich ohne Beschränkung der Allgemeinheit auf das Intervall $[a,b] = [0,1]$ beschränken kann.

Eine naheliegende Idee zur Konstruktion von Quadraturformeln ist, $n+1$ verschiedene Knoten $x_0, x_1, \ldots, x_n \in [0,1]$ zu wählen, das Interpolationspolynom p von f zu diesen Knoten zu bestimmen und als Näherung für das Integral von f das Integral über das Interpolationspolynom p zu wählen,

$$\int_0^1 f(x)\,dx \approx \int_0^1 p(x)\,dx =: Q(f).$$

Mit

$$\ell_j(x) := \prod_{\substack{i=0 \\ i\neq j}}^{n} (x - x_i) \Bigg/ \prod_{\substack{i=0 \\ i\neq j}}^{n} (x_j - x_i)\ , \qquad j = 0, \ldots, n,$$

gilt nach der Lagrangeschen Interpolationsformel

$$p(x) = \sum_{j=0}^{n} f(x_j)\ell_j(x),$$

und daher erhält man

$$Q(f) = \int_0^1 \sum_{j=0}^{n} f(x_j)\ell_j(x)\,dx = \sum_{j=0}^{n} f(x_j) \int_0^1 \ell_j(x)\,dx =: \sum_{j=0}^{n} \alpha_j f(x_j).$$

Dabei hängen die Gewichte

$$\alpha_j := \int_0^1 \ell_j(x)\,dx$$

nur von den gewählten Knoten $x_0, \ldots, x_n$ ab und sind unabhängig vom aktuellen Integranden f. Sie können also ein für alle mal berechnet werden und in Tafeln oder Dateien bereitgestellt werden.

Beispiel 3.1 Für $n = 0$ und $x_0 = 0.5$ gilt $\ell_0(x) \equiv 1$ und

$$\alpha_0 = \int_0^1 \ell_0(x)\,dx = 1.$$

Die entstehende Quadraturformel

$$\int_0^1 f(x)\,dx \approx f(0.5) =: R(f),$$

bzw. die Quadraturformel für das allgemeine Intervall,

$$\int_a^b f(x)\,dx \approx (b-a) f(\frac{a+b}{2}) =: R(f),$$

heißt **Rechteckregel** oder auch **Mittelpunktregel**. □

Beispiel 3.2 Für $n = 1$, $x_0 = 0$ und $x_1 = 1$ ist $\ell_0(x) = 1 - x$ und $\ell_1(x) = x$. Durch Integration erhält man $\alpha_0 = \alpha_1 = 0.5$ und damit die **Trapezregel**

$$\int_0^1 f(x)\,dx \approx \frac{f(0) + f(1)}{2} =: T(f),$$

bzw. die **Trapezregel** für das allgemeine Intervall

$$\int_a^b f(x)\,dx \approx (b-a)\frac{f(a) + f(b)}{2} =: T(f).$$ □

Beispiel 3.3 Für $n = 2$, $x_0 = 0$, $x_1 = 0.5$ und $x_2 = 1$ erhält man wie in den beiden vorhergehenden Beispielen die **Simpson Regel** (in der deutschsprachigen Literatur auch als **Keplersche Fassregel** bekannt)

$$\int_0^1 f(x)\,dx \approx \frac{1}{6} \cdot (f(0) + 4f(0.5) + f(1)) =: S(f)$$

bzw.

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6} \cdot \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) =: S(f).$$ □

n	$\alpha_j^{(n)}$					Name
1	1/2	1/2				Trapezregel
2	1/6	4/6	1/6			Simpson Regel
3	1/8	3/8	3/8	1/8		3/8 Regel
4	7/90	32/90	12/90	32/90	7/90	Milne Regel

Tabelle 3.1: abgeschlossene Newton-Cotes Formeln

Beispiel 3.4 Für $n = 4$ und $x_j = a + j(b-a)/4$, $j = 0, 1, \ldots, 4$, erhält man die **Milne Regel**

$$\int_a^b f(x)\,dx \approx \frac{b-a}{90}(7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)) =: M(f). \quad \square$$

Es ist naheliegend (und dies ist auch die historisch älteste Wahl), die Knoten äquidistant im Intervall $[a, b]$ zu wählen. Berücksichtigt man dabei die Intervallenden, wählt man also

$$x_j = a + j \cdot \frac{b-a}{n}, \qquad j = 0, \ldots, n,$$

so erhält man die **abgeschlossenen Newton-Cotes Formeln**

$$\int_0^1 f(x)\,dx \approx \sum_{j=0}^{n} \alpha_j^{(n)} f\left(\frac{j}{n}\right)$$

bzw.

$$\int_a^b f(x)\,dx \approx (b-a)\sum_{j=0}^{n} \alpha_j^{(n)} f\left(a + j\frac{b-a}{n}\right) =: ANC_n(f).$$

Berücksichtigt man die Intervallenden nicht, wählt man also

$$x_j = a + (j+1) \cdot \frac{b-a}{n+2}, \qquad j = 0, \ldots, n,$$

so erhält man die **offenen Newton-Cotes Formeln**

$$\int_0^1 f(x)\,dx \approx \sum_{j=0}^{n} \beta_j^{(n)} f\left(\frac{j+1}{n+2}\right)$$

bzw.

$$\int_a^b f(x)\,dx \approx (b-a)\sum_{j=0}^{n} \beta_j^{(n)} f\left(a + (j+1)\frac{b-a}{n+2}\right) =: ONC_n(f).$$

Bemerkung 3.5 Die Mittelpunktregel ist die offene Newton-Cotes Regel für den Fall $n = 0$. Die Trapezregel, die Simpson und die Milne Regel sind die abgeschlossenen Newton-Cotes Formeln für die Fälle $n = 1$, $n = 2$ und $n = 4$. $\square$

Tabelle 3.1 enthält die wichtigsten abgeschlossenen Newton-Cotes Formeln, Tabelle 3.2 die ersten offenen Newton-Cotes Formeln.

n	$\beta_j^{(n)}$				
0	1				
1	1/2	1/2			
2	2/3	−1/3	2/3		
3	11/24	1/24	1/24	11/24	
4	11/20	−14/20	26/20	−14/20	11/20

Tabelle 3.2: offene Newton-Cotes Formeln

Bemerkung 3.6 Offene Formeln (d.h. solche Formeln, bei denen die Intervallenden keine Knoten sind,) verwendet man, wenn die Auswertung des Integranden an den Randpunkten schwierig ist (z.B. der Grenzwert eines Quotienten, für den Zähler und Nenner gegen 0 gehen, oder gar eine Singularität von f am Rand vorliegt).

Offene Newton-Cotes Formeln werden heute (mit Ausnahme der Mittelpunktregel) kaum noch verwendet, da sie wesentlich schlechtere Fehlerordnungen haben als die Gauß Formeln (vgl. Abschnitt 3.3), die ebenfalls offen sind. □

Die Gewichte der Newton-Cotes Formeln wachsen rasch an. Für die abgeschlossenen Formeln treten für $n \geqslant 8$ wechselnde Vorzeichen auf (für die offenen Formeln sogar schon für $n \geqslant 2$). Diese Formeln sind also anfällig für Rundungsfehler. Man benutzt die Newton-Cotes Formeln daher nur für kleine n auf Teilintervallen von $[a, b]$ und summiert auf. Man erhält dann die **summierten Newton-Cotes Formeln** oder **zusammengesetzten Newton-Cotes Formeln**.

Beispiele hierfür sind mit

$$h := \frac{b-a}{m} \qquad \text{und} \qquad x_j := a + j \cdot h, \qquad j = 0, \ldots, m,$$

die **summierte Rechteckregel**

$$\int_a^b f(x)\,dx \approx h \sum_{j=1}^{m} f(x_j - h/2) =: R_h(f), \tag{3.1}$$

die **summierte Trapezregel**

$$\int_a^b f(x)\,dx \approx h \left(\frac{1}{2} f(a) + \sum_{i=1}^{m-1} f(x_i) + \frac{1}{2} f(b) \right) =: T_h(f), \tag{3.2}$$

und für $m = 2k$ die **summierte Simpson Regel** (beachten Sie, dass die Simpson Regel jeweils auf zwei benachbarte Teilintervalle der Gesamtlänge $2h$ angewandt wird)

$$\begin{aligned} &\int_a^b f(x)\,dx \\ &\quad \approx \frac{2h}{6} \left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{2k-1}) + f(x_{2k}) \right) \\ &\quad = \frac{2h}{6} \left(f(a) + 4 \sum_{i=1}^{k} f(x_{2i-1}) + 2 \sum_{i=1}^{k-1} f(x_{2i}) + f(b) \right) =: S_h(f). \end{aligned}$$

Abbildung 3.1 enthält links eine graphische Darstellung der summierten Mittelpunktregel und rechts der summierten Trapezregel. In Abbildung 3.2 ist die summierte Simpson Regel dargestellt.

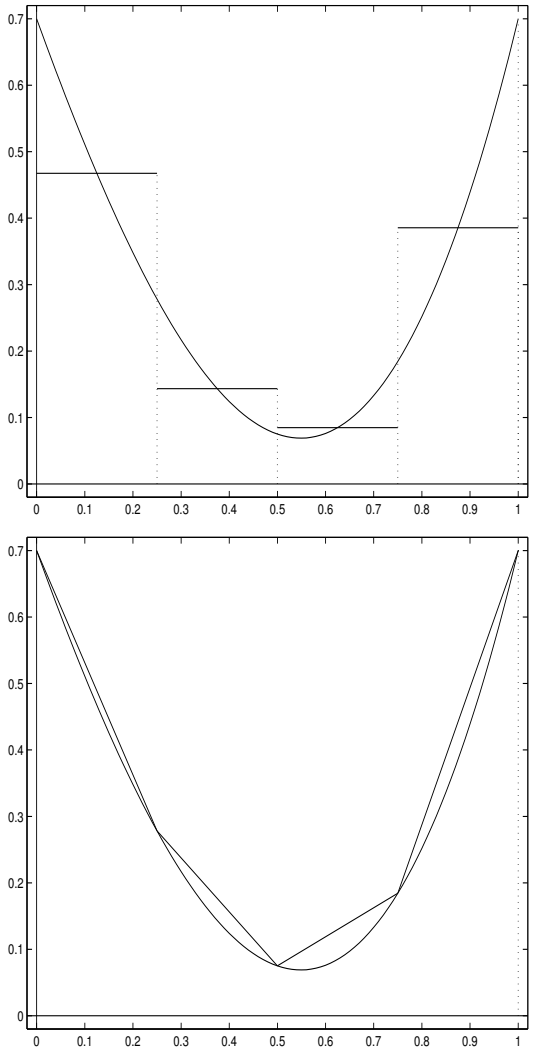

Abbildung 3.1: Summierte Mittelpunkt- / Trapezregel

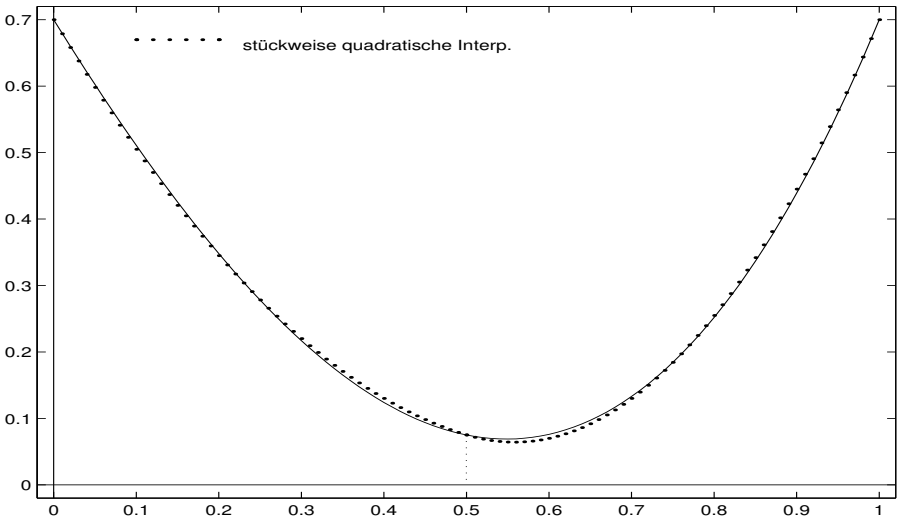

Abbildung 3.2: Summierte Simpson Regel

3.2 Fehler von Quadraturformeln

Wir untersuchen nun den Fehler von Quadraturformeln. Wir betrachten das Integral

$$I := \int_0^1 f(x)\,dx$$

und eine zugehörige Quadraturformel mit den Knoten $x_0, \ldots, x_n \in [0,1]$ und den Gewichten $w_0, \ldots, w_n \in \mathbb{R}$

$$Q(f) := \sum_{i=0}^{n} w_i f(x_i)$$

für das Referenzintervall $[0,1]$.

Definition 3.7 Die Quadraturformel $Q(f)$ hat die **Fehlerordnung** m, wenn für den Fehler

$$E(f) := \int_0^1 f(x)\,dx - \sum_{i=0}^{n} w_i f(x_i)$$

gilt

1. $E(p) = 0$ für alle Polynome $p \in \Pi_{m-1}$,
2. $E(p) \neq 0$ für ein $p \in \Pi_m$.

Bemerkung 3.8 Wegen der Linearität des Fehlers ist klar, dass Q genau dann die Fehlerordnung m hat, wenn $E(x^j) = 0$ für $j = 0, \ldots, m-1$ und $E(x^m) \neq 0$ gilt. □

Bemerkung 3.9 Die Konstruktion liefert, dass die Newton-Cotes Formeln *wenigstens* die Fehlerordnung $n+1$ haben. Für die Trapezregel ist dies die genaue Fehlerordnung, denn

$$T(x^2) = \frac{1}{2} \neq \int_0^1 x^2\,dx = \frac{1}{3}.$$

Für die Simpson Regel gilt

$$S(x^3) = \frac{1}{6}\left(0 + 4 \cdot \frac{1}{8} + 1\right) = \frac{1}{4} = \int_0^1 x^3\,dx, \quad S(x^4) = \frac{5}{24} \neq \int_0^1 x^4\,dx = \frac{1}{5},$$

so dass die Simpson Regel sogar die Ordnung 4 hat. □

Satz 3.10 *Es sei Q eine Quadraturformel der Fehlerordnung $m \geqslant 1$. Dann hat für $f \in C^m[0,1]$ der Fehler von Q die Darstellung*

$$E(f) = \int_0^1 f(x)\,dx - Q(f) = \int_0^1 K(x) f^{(m)}(x)\,dx \tag{3.3}$$

mit

$$K(x) = \frac{1}{(m-1)!}\left(\frac{1}{m}(1-x)^m - \sum_{i=0}^{n} w_i (x_i - x)_+^{m-1}\right).$$

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 3.2. ■

Definition 3.11 Die Funktion K in der Fehlerdarstellung (3.3) heißt der **Peano Kern** der Quadraturformel Q.

Beispiel 3.12 Für die Trapezregel gilt $x_0 = 0$, $x_1 = 1$, $w_0 = w_1 = 0.5$, $m = 2$, und daher

$$K_T(x) = \frac{1}{2}(1-x)^2 - \frac{1}{2}(1-x) = -\frac{1}{2}x(1-x).$$ □

Beispiel 3.13 Für die Simpson Regel gilt $x_0 = 0$, $x_1 = 0.5$, $x_2 = 1$, $w_0 = w_2 = 1/6$, $w_1 = 2/3$ und $m = 4$, und daher

$$K_S(x) = \frac{1}{3!}\left(\frac{1}{4}(1-x)^4 - \frac{1}{6}(1-x)^3 - \frac{2}{3}\left(\frac{1}{2} - x\right)_+^3\right).$$ □

Aus Satz 3.10 erhält man die folgende Abschätzung für den Fehler

$$|E(f)| \leqslant \|f^{(m)}\|_\infty \int_0^1 |K(x)|\,dx =: \tilde{c}_m \,\|f^{(m)}\|_\infty. \tag{3.4}$$

In vielen Fällen wechselt der Peano Kern $K(x)$ das Vorzeichen auf dem Intervall $[0,1]$ nicht. Dann folgt aus (3.3) mit dem Mittelwertsatz der Integralrechnung mit einem $\xi \in (0,1)$

$$E(f) = f^{(m)}(\xi) \int_0^1 K(x)\,dx =: c_m\, f^{(m)}(\xi) \tag{3.5}$$

für eine Quadraturformel der Ordnung m.

Definition 3.14 Die Konstante c_m heißt **Fehlerkonstante** des Verfahrens Q.

Für die Trapezregel gilt

$$K_T(x) = -\frac{1}{2}\,x(1-x) \quad \text{für alle} \quad x \in [0,1].$$

Da T die Ordnung 2 hat, gilt

$$E_T(f) = -\frac{1}{2}\,f''(\xi) \int_0^1 x(1-x)\,dx = -\frac{1}{12}\,f''(\xi),$$

und die Fehlerkonstante der Trapezregel ist somit $c_2 = -1/12$.

Eine elementare Rechnung zeigt, dass auch für die Simpson Regel

$$K_S(x) = \frac{1}{3!}\left(\frac{1}{4}(1-x)^4 - \frac{1}{6}(1-x)^3 - \frac{2}{3}\left(\frac{1}{2}-x\right)_+^3\right) \leqslant 0$$

für alle $x \in [0,1]$ gilt. Durch Integration von K von 0 bis 1 erhält man für den Fehler wegen $m = 4$

$$E_S(f) = -\frac{f^{(4)}(\xi)}{2880}$$

für ein $\xi \in (0,1)$, d.h. die Fehlerkonstante ist $c_4 = -1/2880$.

Bemerkung 3.15 Man kann die Fehlerkonstante der Simpson Regel und auch anderer Formeln ohne Integration (sogar ohne Kenntnis) des Peano Kerns bestimmen. Ist bekannt, dass der Peano Kern einer Formel der Ordnung m sein Vorzeichen in $[0,1]$ nicht wechselt, der Fehler also eine Darstellung (3.5) hat, so hat man nur $E(x^m)$ zu berechnen. Es ist nämlich

$$\frac{d^m}{dx^m}(x^m) = m!$$

und daher

$$E(x^m) = m! \cdot c_m,$$

und hieraus erhält man c_m. □

Beispiel 3.16 Im Fall der Simpson Regel ist

$$E(x^4) = \int_0^1 x^4\,dx - \frac{1}{6}\left((0)^4 + 4\cdot(1/2)^4 + 1^4\right) = \frac{1}{5} - \frac{1}{6}\cdot\frac{5}{4} = -\frac{1}{120},$$

und daher gilt

$$c_4 = \frac{1}{4!}\cdot\left(-\frac{1}{120}\right) = -\frac{1}{2880}.$$ □

Wir betrachten nun das Integral von f über ein Intervall der Länge h:

$$\int_\alpha^{\alpha+h} f(x)\,dx.$$

Mit der Variablentransformation $x =: \alpha + ht$ geht dieses über in

$$\int_\alpha^{\alpha+h} f(x)\,dx = h\int_0^1 g(t)\,dt, \quad g(t) := f(\alpha+ht) = f(x),$$

das wir mit der Quadraturformel

$$Q(g) = \sum_{i=0}^n w_i g(x_i)$$

behandeln, d.h.

$$Q_{[\alpha,\alpha+h]}(f) := h\sum_{i=0}^n w_i\, f(\alpha + hx_i).$$

Für den Fehler gilt

$$\begin{aligned} E_{[\alpha,\alpha+h]}(f) &:= \int_\alpha^{\alpha+h} f(x)\,dx - Q_{[\alpha,\alpha+h]}(f) \\ &= h\left(\int_0^1 g(t)\,dt - Q(g)\right) = h\,E(g). \end{aligned}$$

Besitzt der Fehler $E(g)$ eine Darstellung (3.4), so folgt wegen

$$\frac{d^m g}{dt^m} = \frac{d^m f}{dx^m} \cdot \left(\frac{dx}{dt}\right)^m = h^m \, \frac{d^m f}{dx^m}$$

$$|E_{[\alpha,\alpha+h]}(f)| \leqslant h^{m+1} \cdot \tilde{c}_m \cdot \max_{\alpha \leqslant x \leqslant \alpha+h} |f^{(m)}(x)|. \tag{3.6}$$

Gilt eine Darstellung (3.5), so erhält man genauso

$$E_{[\alpha,\alpha+h]}(f) = h^{m+1} \cdot c_m \cdot f^{(m)}(\eta), \quad \eta \in [\alpha, \alpha+h]. \tag{3.7}$$

Wir haben bereits erwähnt, dass die Genauigkeit einer Näherung für ein Integral nicht durch die Erhöhung der Ordnung der benutzten Quadraturformel verbessert wird, sondern dass das Intervall $[a,b]$ in n Teilintervalle der Länge h zerlegt wird, und dass in jedem Teilintervall eine Quadraturformel kleiner Ordnung verwendet wird. Für die summierte Quadraturformel

$$Q_h(f) := \sum_{i=1}^{n} Q_{[a+(i-1)\,h, a+i\,h]}(f)$$

erhält man aus (3.6) (und genauso aus (3.7))

$$\begin{aligned}
\left|\int_a^b f(x)\,dx - Q_h(f)\right| &= \left|\sum_{i=1}^{n}\Big(\int_{a+(i-1)\,h}^{a+i\,h} f(x)\,dx - Q_{[a+(i-1)\,h, a+i\,h]}(f)\Big)\right| \\
&\leqslant \sum_{i=1}^{n} |E_{[a+(i-1)\,h, a+i\,h]}(f)| \\
&\leqslant \sum_{i=1}^{n} h^{m+1} \cdot \tilde{c}_m \cdot \max\{|f^{(m)}(x)| \,:\, a+(i-1)\,h \leqslant x \leqslant a+i\,h\} \\
&\leqslant n\,h \cdot h^m \cdot \tilde{c}_m \cdot \|f^{(m)}\|_\infty = h^m\,(b-a)\,\tilde{c}_m \cdot \|f^{(m)}\|_\infty.
\end{aligned}$$

Man verliert also für die summierte Formel eine Potenz in h. Insbesondere erhält für die summierte Trapezregel den Fehler

$$\left|\int_a^b f(x)\,dx - T_h(f)\right| \leqslant \frac{h^2}{12}\,(b-a) \cdot \|f''\|_\infty \tag{3.8}$$

und für die summierte Simpson Regel

$$\left|\int_a^b f(x)\,dx - S_h(f)\right| \leqslant \frac{h^4}{2880}\,(b-a) \cdot \|f^{(4)}\|_\infty.$$

3.3 Quadraturformeln von Gauß

Wir haben bisher in Abschnitt 3.1 die Knoten der Quadraturformeln (äquidistant) vorgegeben. Die Fehlerordnung war dann (wenigstens) gleich der Anzahl der Knoten (im Falle der Simpson Regel bei 3 Knoten 4). Wir fragen nun, wie weit wir durch Wahl der Knoten und der Gewichte die Fehlerordnung erhöhen können.

Beispiel 3.17 Wir betrachten die Quadraturformel $G_1(f) := w_1\, f(x_1)$ mit einem Knoten x_1 für das Integral $\int_0^1 f(x)\,dx$ und bestimmen $x_1 \in [0,1]$ und $w_1 \in \mathbb{R}$ so, dass Polynome möglichst hohen Grades exakt integriert werden:

$$\int_0^1 x^0\,dx = 1 = w_1\, x_1^0 = w_1 \quad \Rightarrow \quad w_1 = 1,$$
$$\int_0^1 x^1\,dx = 0.5 = w_1\, x_1^1 = x_1 \quad \Rightarrow \quad x_1 = 0.5.$$

Durch diese beiden Gleichungen ist also die Quadraturformel

$$G_1(f) = f(0.5)$$

bereits festgelegt. Man erhält die Mittelpunktregel. Wegen

$$\int_0^1 x^2\,dx = \frac{1}{3} \neq 1 \cdot (0.5)^2$$

hat sie die Fehlerordnung 2. □

Beispiel 3.18 Für die Quadraturformel

$$G_2(f) = w_1 f(x_1) + w_2 f(x_2)$$

mit zwei Knoten $x_1, x_2 \in [0,1]$ erhält man die Bestimmungsgleichungen

$$\int_0^1 x^0\,dx = 1 = w_1 + w_2$$
$$\int_0^1 x^1\,dx = \frac{1}{2} = w_1 x_1 + w_2 x_2$$
$$\int_0^1 x^2\,dx = \frac{1}{3} = w_1 x_1^2 + w_2 x_2^2$$
$$\int_0^1 x^3\,dx = \frac{1}{4} = w_1 x_1^3 + w_2 x_2^3$$

mit der (bis auf Vertauschung von x_1 und x_2) eindeutigen Lösung

$$w_1 = w_2 = \frac{1}{2},\ x_1 = \frac{1}{2}\left(1 - \frac{1}{\sqrt{3}}\right),\ x_2 = \frac{1}{2}\left(1 + \frac{1}{\sqrt{3}}\right).$$

Wegen

$$\int_0^1 x^4\,dx = \frac{1}{5} \neq w_1 x_1^4 + w_2 x_2^4 = \frac{7}{36}$$

hat die gefundene Formel G_2 die Fehlerordnung 4. □

Prinzipiell kann man so fortfahren und Quadraturformeln immer höherer Ordnung konstruieren. Man erhält dann nichtlineare Gleichungssysteme, die immer unübersichtlicher werden. Wir gehen einen anderen Weg.

In Verallgemeinerung unserer bisherigen Überlegungen betrachten wir gleich

$$I(f) = \int_a^b w(x)\, f(x)\,dx$$

mit einer positiven Gewichtsfunktion $w \in C[a,b]$.

Man kann zeigen, dass es zu jedem $n \in \mathbb{N}$ und jeder positiven Gewichtsfunktion $w \in C[a,b]$ eindeutig bestimmte Gewichte $w_i > 0$ und Knoten $x_i \in (a,b)$, $i = 1,\ldots,n$, gibt, so dass die Quadraturformel

$$G_n(f) := \sum_{i=1}^{n} w_i f(x_i) \tag{3.9}$$

die Fehlerordnung $2n$ besitzt (also für alle Polynome vom Höchstgrad $2n-1$ exakt ist, nicht aber für x^{2n}) und dass durch keine Wahl von Knoten und Gewichten eine höhere Fehlerordnung erreichbar ist.

Dass mit n Knoten nicht die Fehlerordnung $2n+1$ erreicht werden kann, sieht man so ein. Besitzt (3.9) die Fehlerordnung $2n+1$, so wird insbesondere das Polynom

$$p(x) := \prod_{j=1}^{n} (x - x_j)^2 \in \Pi_{2n}$$

exakt integriert. Wegen $p \geqslant 0$ und $p \not\equiv 0$ gilt aber

$$\int_a^b w(x)p(x)\,dx > 0, \quad \text{während} \quad G_n(p) = 0 \quad \text{ist.}$$

Wir geben einige Gewichte und Knoten für den Spezialfall $w(x) \equiv 1$ und ohne Beschränkung der Allgemeinheit $[a,b] = [-1,1]$ an.

n	w_i	x_i
1	$w_1 = 2$	$x_1 = 0$
2	$w_1 = w_2 = 1$	$x_2 = -x_1 = \frac{1}{\sqrt{3}}$
3	$w_1 = w_3 = \frac{5}{9}, w_2 = \frac{8}{9}$	$x_3 = -x_1 = \sqrt{\frac{3}{5}}\ ,\ x_2 = 0$

Weitere Werte findet man in Abramowitz und Stegun [1], pp. 916 ff, und in Piessens et al. [56], pp. 19 ff.

Bemerkung 3.19 Auch für unbeschränkte Integrationsintervalle kann man Gaußsche Quadraturformeln entwickeln. So erhält man für Integrale der Gestalt

$$\int_0^\infty e^{-x} f(x)\,dx$$

(mit den Nullstellen der **Laguerre Polynome** als Knoten) die **Gauß-Laguerre Quadraturformeln** , und für Integrale der Gestalt

$$\int_{-\infty}^\infty e^{-x^2} f(x)\,dx$$

(mit den Nullstellen der **Hermite Polynome** als Knoten) die **Gauß-Hermite Quadraturformeln**.

Knoten und Gewichte der Gauß-Laguerre Formeln (für $n \leqslant 15$) findet man in Abramowitz und Stegun [1], p. 923, und für die Gauß-Hermite Formeln (bis $n = 20$) auf Seite 924. □

Beispiel 3.20 Für das Integral

$$\int_0^\infty \frac{x}{1+e^x}\,dx = \int_0^\infty e^{-x}\frac{x}{1+e^{-x}}\,dx = \frac{\pi^2}{12} = 0.8224670334241132$$

enthält Tabelle 3.3 die Näherungen mit den Gauß-Laguerre Quadraturformeln und die Fehler. Man sieht, dass man auch mit wenigen Knoten zu sehr guten Näherungen gelangt. □

n	Q_n	Fehler
1	0.7310585786300049	$9.14e-2$
2	0.8052717896130982	$1.72e-2$
3	0.8238172597250991	$-1.35e-3$
4	0.8236994602380588	$-1.23e-3$
5	0.8226695411616926	$-2.03e-4$
6	0.8224050273750929	$6.20e-5$

Tabelle 3.3: Quadraturformeln von Gauß-Laguerre

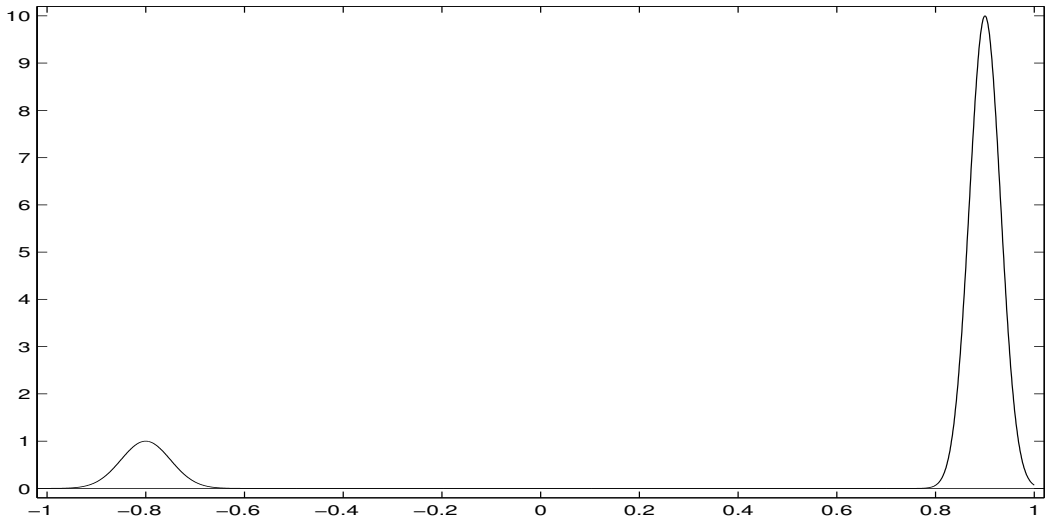

Abbildung 3.3: Zu Beispiel 3.21

3.4 Adaptive Quadratur

Wir haben summierte Quadraturformeln nur mit konstanter Schrittweite $h > 0$ betrachtet. Es ist klar, dass man dabei für Funktionen mit unterschiedlichem Verhalten in verschiedenen Teilen des Integrationsintervalls entweder bei zu groß gewähltem h ein ungenaues Ergebnis erhält oder bei zu kleinem h Arbeit in den Bereichen verschenkt, in denen die Funktion „gutartig“ ist.

Beispiel 3.21 (siehe dazu Abbildung 3.3)

$$f(x) = \exp\Big(-200\,(x+0.8)^2\Big) + 10 \cdot \exp\Big(-500\,(x-0.9)^2\Big), \quad -1 \leqslant x \leqslant 1. \qquad \square$$

Wir entwickeln nun eine Vorgehensweise, mit der bei gegebenem Integranden, gegebenen Grenzen a und b, und gegebener Genauigkeitsschranke $\varepsilon > 0$ eine Quadraturformel

$$I(f) = \sum_{j=0}^{k} w_j\, f(x_j)$$

erzeugt wird, für die gilt

$$\left| \int_a^b f(x)\,dx - I(f) \right| \dot{<} \varepsilon,$$

wobei der Punkt über dem Ungleichungszeichen besagt, dass die Ungleichung nur asymptotisch für feine Zerlegungen $a \leqslant x_0 < x_1 < \cdots < x_k \leqslant b$ von $[a,b]$ gilt.

Die Knoten x_j und Gewichte w_j werden adaptiv durch das Verfahren in Abhängigkeit von f und ε erzeugt.

Es sei

$$Q(f) = \sum_{i=1}^{n} w_i \, f(t_i)$$

eine Quadraturformel der Ordnung m für das Referenzintervall $[-1,1]$.

Es sei das Integral bis zum Punkt $x_j \in [a,b)$ schon näherungsweise bestimmt, und es sei $x_{j+1} \in (x_j, b]$ gegeben. Dann berechnen wir die zwei Näherungen

$$Q_{[x_j,x_{j+1}]}(f) = \frac{x_{j+1}-x_j}{2} \sum_{i=1}^{n} w_i \, f\left(\frac{x_{j+1}+x_j}{2} + t_i \, \frac{x_{j+1}-x_j}{2}\right)$$

und

$$\begin{aligned}\tilde{Q}_{[x_j,x_{j+1}]}(f) := {} & \frac{x_{j+\frac{1}{2}}-x_j}{2} \sum_{i=1}^{n} w_i \, f\left(\frac{x_{j+\frac{1}{2}}+x_j}{2} + t_i \, \frac{x_{j+\frac{1}{2}}-x_j}{2}\right) \\ & + \frac{x_{j+1}-x_{j+\frac{1}{2}}}{2} \sum_{i=1}^{n} w_i \, f\left(\frac{x_{j+1}+x_{j+\frac{1}{2}}}{2} + t_i \, \frac{x_{j+1}-x_{j+\frac{1}{2}}}{2}\right)\end{aligned}$$

für $\displaystyle\int_{x_j}^{x_{j+1}} f(x)\,dx$, wobei $x_{j+\frac{1}{2}} := 0.5(x_j + x_{j+1})$ gesetzt ist, und hiermit

$$\hat{Q}_{[x_j,x_{j+1}]} := \frac{1}{2^m - 1}\left(2^m \cdot \tilde{Q}_{[x_j,x_{j+1}]}(f) - Q_{[x_j,x_{j+1}]}(f)\right).$$

Dann ist mit $\tilde{Q}$ und Q auch $\hat{Q}$ eine Quadraturformel von mindestens der Ordnung m, und man kann leicht mit Hilfe der Fehlerdarstellung aus Satz 3.10 zeigen, dass durch $\hat{Q}$ sogar Polynome vom Grade m exakt integriert werden, $\hat{Q}$ also wenigstens die Ordnung $m+1$ hat.

Wir benutzen nun $\hat{Q}$, um den Fehler von $\tilde{Q}$ (der genaueren der beiden Ausgangsformeln) zu schätzen.

Es gilt mit $h := x_{j+1} - x_j$

$$\begin{aligned}\tilde{E}_{[x_j,x_{j+1}]}(f) &:= \int_{x_j}^{x_{j+1}} f(x)\,dx - \tilde{Q}_{[x_j,x_{j+1}]}(f) \\ &= \hat{Q}_{[x_j,x_{j+1}]}(f) - \tilde{Q}_{[x_j,x_{j+1}]}(f) + O(h^{m+2}) \\ &= \frac{1}{2^m-1}\left(2^m \, \tilde{Q}_{[x_j,x_{j+1}]}(f) - Q_{[x_j,x_{j+1}]}(f)\right) - \tilde{Q}_{[x_j,x_{j+1}]}(f) + O(h^{m+2}) \\ &= \frac{1}{2^m-1}\left(\tilde{Q}_{[x_j,x_{j+1}]}(f) - Q_{[x_j,x_{j+1}]}(f)\right) + O(h^{m+2}).\end{aligned}$$

Da andererseits $\tilde{E}_{[x_j,x_{j+1}]}(f) = O(h^{m+1})$ gilt, können wir für kleine h den Summanden $O(h^{m+2})$ vernachlässigen und erhalten die Fehlerschätzung

$$\tilde{E}_{[x_j,x_{j+1}]}(f) \approx \frac{1}{2^m-1}\left(\tilde{Q}_{[x_j,x_{j+1}]}(f) - Q_{[x_j,x_{j+1}]}(f)\right). \tag{3.10}$$

Wir benutzen (3.10), um das Intervall $[a,b]$ durch Bisektion zu zerlegen in $a = x_0 < x_1 < \cdots < x_k = b$, so dass für $j = 1, \ldots, k$ gilt

$$\left|\tilde{Q}_{[x_{j-1},x_j]}(f) - Q_{[x_{j-1},x_j]}(f)\right| \leqslant \frac{2^m-1}{b-a}\,(x_j - x_{j-1})\,\varepsilon. \tag{3.11}$$

Dann folgt für die summierte Quadraturformel

$$Q(f) := \sum_{j=1}^{k} \tilde{Q}_{[x_{j-1},x_j]}(f)$$

aus (3.10)

$$\begin{aligned} E(f) = \int_a^b f(x)\,dx - Q(f) &= \sum_{j=1}^{k} \tilde{E}_{[x_{j-1},x_j]}(f) \\ &\dot{\leqslant} \frac{1}{2^m-1} \sum_{j=1}^{k} \left| \tilde{Q}_{[x_{j-1},x_j]}(f) - Q_{[x_{j-1},x_j]}(f) \right| \\ &\leqslant \frac{1}{b-a} \sum_{j=1}^{k} (x_j - x_{j-1})\,\varepsilon = \varepsilon. \end{aligned}$$

Mit der Fehlerschätzung kann man nun auf folgende Weise ein Integral mit einer gewünschten (asymptotischen) Genauigkeit adaptiv berechnen: Ist also das Integral

$$\int_a^{x_j} f(x)\,dx$$

schon mit der gewünschten Genauigkeit bestimmt und wurden mit der Schrittweite h die Näherungen $Q_{[x_j,x_j+h]}(f)$ und $\tilde{Q}_{[x_j,x_j+h]}(f)$ berechnet, so kann man hiermit das Erfülltsein der lokalen Fehlerschranke (3.11) prüfen. Ist dies der Fall, so geht man zu dem neuen Intervall $[x_{j+1}, x_{j+1} + h_{\text{neu}}]$, $x_{j+1} := x_j + h$, über. Sonst wiederholt man den Schritt mit einer verkleinerten Schrittweite h_{neu}.

Die neue Schrittweite kann man so bestimmen: Es ist

$$E_{[x_j,x_j+h]} \approx \frac{1}{2^m-1} |Q_{[x_j,x_j+h]}(f) - \tilde{Q}_{[x_j,x_j+h]}(f)| \approx C h^{m+1},$$

d.h.

$$C \approx \frac{h^{-m-1}}{2^m-1} |Q_{[x_j,x_j+h]}(f) - \tilde{Q}_{[x_j,x_j+h]}(f)|,$$

und daher kann man erwarten, dass mit

$$\varepsilon \frac{h_{\text{neu}}}{b-a} = C h_{\text{neu}}^{m+1} = \frac{h^{-m-1}}{2^m-1} |Q_{[x_j,x_j+h]}(f) - \tilde{Q}_{[x_j,x_j+h]}(f)| h_{\text{neu}}^{m+1},$$

d.h.

$$h_{\text{neu}} = h \left(\frac{(2^m-1)h\varepsilon}{(b-a)|Q_{[x_j,x_j+h]}(f) - \tilde{Q}_{[x_j,x_j+h]}(f)|} \right)^{1/m}$$

die lokale Fehlerschranke (3.11) eingehalten wird. Hiermit erhält man das folgende Verfahren:

Algorithmus 3.22 (Adaptive Gauß Quadratur)

```
function int=adap_gauss3(f,a,b,tol);
factor=63*tol/(b-a);  h=0.1*(b-a);   int=0;
while a < b
  q=gauss3(f,a,a+h);
  qs=gauss3(f,a,a+0.5*h)+gauss3(f,a+0.5*h,a+h);
  h_neu=0.9*h*(h*factor/abs(qs-q))^(1/6);
  if abs(q-qs) > h*factor;
```

ε	Fehler	Funktionsauswertungen
$1E-3$	$8.78E-02$	54
$1E-4$	$2.99E-05$	153
$1E-5$	$2.63E-07$	270
$1E-6$	$8.22E-09$	351
$1E-7$	$2.10E-09$	531
$1E-8$	$8.02E-12$	747

Tabelle 3.4: Adaptive Gauß Quadratur

```
  h=h_neu;
else
  int=int+qs+(qs-q)/63;
  a=a+h;
  h=min(h_neu,b-a);
  end
end
```

Beispiel 3.23 Für das Beispiel 3.21 erhält man hiermit die Ergebnisse aus Tabelle 3.4. Wie erwartet wird der tatsächliche Fehler (jedenfalls für $\varepsilon \leqslant 1E-04$) deutlich kleiner als die vorgegebene Toleranz ε. □

Verwendet man wie oben in einem adaptiven Verfahren dieselbe Gauß Formel für Q und $\tilde{Q}$, so kann man die an den Knoten von Q berechneten Funktionswerte bei der Auswertung von $\tilde{Q}$ nicht wiederverwenden. Von Kronrod [45] wurde 1965 die folgende Vorgehensweise vorgeschlagen, die diesen Nachteil nicht hat:

Wir gehen aus von einer Gauß Formel

$$G_n(f) = \sum_{i=1}^{n} w_i\, f(x_i)$$

der Ordnung $2n$ mit den Knoten $x_1, \ldots, x_n \in (-1,1)$, und bestimmen $n+1$ weitere Knoten $y_0, \ldots, y_n \in (-1,1)$ und Gewichte α_i, β_i, so dass die Quadraturformel

$$K_n(f) := \sum_{i=1}^{n} \alpha_i\, f(x_i) + \sum_{i=0}^{n} \beta_i\, f(y_i)$$

möglichst hohe Ordnung besitzt. K_n heißt die zu G_n gehörige ***Kronrod Formel****.*

Beispiel 3.24 Ausgehend von

$$G_2(f) = f\Big(-\frac{1}{\sqrt{3}}\Big) + f\Big(\frac{1}{\sqrt{3}}\Big)$$

machen wir für K_2 den Ansatz

$$K_2(f) = \alpha_1\, f\Big(-\frac{1}{\sqrt{3}}\Big) + \alpha_2\, f\Big(\frac{1}{\sqrt{3}}\Big) + \beta_0\, f(y_0) + \beta_1\, f(y_1) + \beta_2\, f(y_2)$$

und bestimmen die 8 Unbekannten α_1, α_2, β_0, β_1, β_2, y_0, y_1, y_2 so, dass die Funktionen x^j, $j = 0, 1, 2, \ldots, m$, für möglichst großes m durch K_2 exakt integriert werden.

Man kann zeigen, dass die Kronrod Formeln symmetrisch sind (hier: $\alpha_1 = \alpha_2$, $\beta_0 = \beta_2$, $y_0 = -y_2$, $y_1 = 0$). Unter Ausnutzung dieser Symmetrie folgt

$$K_2(x^{2j+1}) = 0 = \int_{-1}^{1} x^{2j+1}\,dx \qquad \text{für alle } j = 0, 1, 2, \ldots$$

Für die geraden Potenzen ergibt sich das nichtlineare Gleichungssystem

$$\begin{array}{lrcrcrcl} x^0\colon & 2\,\alpha_1 & + & 2\,\beta_0 & + & \beta_1 & = & 2, \\ x^2\colon & \frac{2}{3}\,\alpha_1 & + & 2\,\beta_0\,y_0^2 & & & = & \frac{2}{3}, \\ x^4\colon & \frac{2}{5}\,\alpha_1 & + & 2\,\beta_0\,y_0^4 & & & = & \frac{2}{5}, \\ x^6\colon & \frac{2}{27}\,\alpha_1 & + & 2\,\beta_0\,y_0^6 & & & = & \frac{2}{7}, \end{array}$$

mit der eindeutigen Lösung

$$y_0 = \sqrt{\frac{6}{7}}, \quad \alpha_1 = \frac{243}{495}, \quad \beta_0 = \frac{98}{495}, \quad \beta_1 = \frac{308}{495},$$

d.h.

$$K_2(f) = \frac{243}{495}\,G_2(f) + \frac{1}{495}\left(98\left(f\Big(-\sqrt{\frac{6}{7}}\Big) + f\Big(\sqrt{\frac{6}{7}}\Big)\right) + 308\,f(0)\right).$$

Nach Konstruktion hat diese Formel mindestens die Ordnung 8, und durch Berechnung von $E(x^8)$ sieht man, dass die Ordnung genau 8 ist. □

Man kann zeigen, dass zu einer n-Punkt Gauß Formel G_n stets die $(2n+1)$-Punkt Kronrod Formel konstruiert werden kann, und dass ihre Ordnung $3n+2$ ist, falls n gerade ist, und $3n+3$, falls n ungerade ist.

Die Kronrod Formel K_n kann man nun auf folgende Weise nutzen, um den Fehler E_n der zugehörigen Gauß Formel zu schätzen. Es gilt für $[x_j, x_{j+1}] \subset [-1, 1]$

$$\int_{x_j}^{x_{j+1}} f(x)\,dx = K_n(f) + h^{3n+2} \cdot c_{3n+2} \cdot f^{(3n+2)}(\eta),$$

$h := x_{j+1} - x_j$, und daher folgt

$$\begin{aligned} E_n &:= \int_{x_j}^{x_{j+1}} f(x)\,dx - G_n(f) \\ &= K_n(f) - G_n(f) + h^{3n+2} \cdot c_{3n+2} \cdot f^{(3n+2)}(\eta). \end{aligned}$$

Da E_n proportional zu h^{2n+1} ist, können wir für kleine h den letzten Summanden vernachlässigen und erhalten

$$E_n \approx K_n(f) - G_n(f).$$

In dem folgenden Algorithmus schätzen wir hiermit den Fehler der Gauß Formel, verwenden aber als Näherung für das Integral $\int_{x_j}^{x_{j+1}} f(x)\,dx$ den mit der Kronrod Formel ermittelten Wert. Dies führt dazu, dass der Fehler wesentlich unterhalb der geforderten Toleranz liegt.

Der folgende Algorithmus realisiert ähnlich wie Algorithmus 3.22 das schrittweise Vorgehen mit der Gauß-Kronrod Formel.

Algorithmus 3.25 (Adaptive Kronrod Quadratur)

ε	Fehler	Funktionsauswertungen
$1E-1$	$4.79E-05$	70
$1E-2$	$3.51E-07$	150
$1E-3$	$1.44E-08$	255
$1E-4$	$6.67E-11$	435

Tabelle 3.5: Adaptive Kronrod Quadratur

```
function int=adap_kronrod(f,a,b,tol);
h=0.1*(b-a);
int=0;
eps=tol/(b-a);
while a < b
  [int1,int2]=kronrod(f,a,a+h);
  h_neu=0.9*h*(h*eps/abs(int1-int2))^(1/4);
  if abs(int1-int2) > h*eps;
    h=h_neu;
  else
    int=int+int2;
    a=a+h;
    h=min(h_neu,b-a);
    end
  end
```

Beispiel 3.26 Für den Algorithmus 3.25 erhält man hiermit die Ergebnisse aus Tabelle 3.5. Wie erwartet wird der tatsächliche Fehler wiederum deutlich kleiner als die vorgegebene Toleranz ε. □

Bemerkung 3.27 In dem Handbuch Piessens et al. [56] des Software Pakets QUADPACK finden sich die Knoten und Gewichte für Gauß Formeln und die zugehörigen Kronrod Formeln (bis hin zur 30-Punkt Gauß und 61-Punkt Kronrod Formel) und weiteres Material zu adaptiven Quadratur Methoden. Die FORTRAN 77 Subroutines von QUADPACK können als Public Domain Software bezogen werden von

http://www.netlib.org/quadpack

3.5 Numerische Differentiation

Wir betrachten eine Funktion $f : [a,b] \to \mathbb{R}$, von der nur die Funktionswerte $y_j := f(x_j)$ an diskreten Punkten $a \leqslant x_1 < x_2 < \cdots < x_n \leqslant b$ bekannt sind. Aufgabe ist es, aus diesen diskreten Daten eine Näherung für den Wert einer Ableitung $f^{(m)}(x)$, $m \geqslant 1$, an einer Stelle x zu ermitteln.

Ähnlich wie bei der numerischen Integration interpolieren wir hierzu einige der gegebenen Daten (x_j, y_j) in der Nähe des Punktes x und wählen die m-te Ableitung der interpolierenden Funktion an der Stelle x als Näherung für $f^{(m)}(x)$. Gebräuchlich ist die Interpolation mit Polynomen und mit Splines.

Wir beschränken uns hier auf die Interpolation mit Polynomen und betrachten nur den Fall, dass die Stelle x, an der die Ableitung approximiert werden soll, ein Knoten ist.

Beispiel 3.28 Wir leiten Formeln für die Approximation der ersten Ableitung her.

Interpoliert man f linear mit den Daten (x_j, y_j) und (x_{j+1}, y_{j+1}), so erhält man

$$p(x) = y_j + \frac{y_{j+1} - y_j}{x_{j+1} - x_j}(x - x_j),$$

und als Näherung für die Ableitung

$$f'(x_j) \approx p'(x_j) = \frac{y_{j+1} - y_j}{x_{j+1} - x_j}. \tag{3.12}$$

Dieser Ausdruck heißt der **vorwärtsgenommene Differenzenquotient** . Interpoliert man die Daten (x_{j-1}, y_{j-1}) und (x_j, y_j) linear, so erhält man genauso die Approximation

$$f'(x_j) \approx \frac{y_j - y_{j-1}}{x_j - x_{j-1}} \tag{3.13}$$

durch den **rückwärtsgenommenen Differenzenquotienten** .

Interpoliert man f quadratisch mit den Knoten (x_{j+k}, y_{j+k}), $k = -1, 0, 1$, so erhält man

$$p(x) = y_j + [x_{j-1}, x_j](x - x_j) + [x_{j+1}, x_{j-1}, x_j](x - x_{j-1})(x - x_j)$$

mit der Ableitung

$$p'(x) = [x_{j-1}, x_j] + [x_{j+1}, x_{j-1}, x_j](2x - x_{j-1} - x_j).$$

Einsetzen von $x = x_j$ liefert nach kurzer Rechnung

$$f'(x_j) \approx \frac{x_{j+1} - x_j}{x_{j+1} - x_{j-1}}[x_{j-1}, x_j] + \frac{x_j - x_{j-1}}{x_{j+1} - x_{j-1}}[x_j, x_{j+1}]. \tag{3.14}$$

Ist speziell $x_{j+1} - x_j = x_j - x_{j-1} =: h$, so ist (3.14) der **zentrale Differenzenquotient**

$$f'(x_j) \approx \frac{1}{2}[x_{j-1}, x_j] + \frac{1}{2}[x_j, x_{j+1}] = \frac{y_{j+1} - y_{j-1}}{2h}. \tag{3.15}$$

Sind nur Funktionswerte von f auf einer Seite von x_j bekannt, so verwendet man einseitige Differenzenapproximationen. Z.B. erhält man mit $y_{j+k} = f(x_{j+k})$, $k = 0, 1, 2$, die Näherung

$$f'(x_j) \approx \frac{x_{j+2} + x_{j+1} - 2x_j}{x_{j+2} - x_j}[x_j, x_{j+1}] - \frac{x_{j+1} - x_j}{x_{j+2} - x_j}[x_{j+1}, x_{j+2}], \tag{3.16}$$

und im äquidistanten Fall

$$f'(x_j) \approx \frac{3}{2}[x_j, x_{j+1}] - \frac{1}{2}[x_{j+1}, x_{j+2}] = \frac{-y_{j+2} + 4y_{j+1} - 3y_j}{2h}. \tag{3.17}$$

Genauso erhält man mit 5 bzw. 7 äquidistanten Knoten die zentralen Differenzenapproximationen

$$f'(x_j) \approx \frac{1}{12h}(y_{j-2} - 8y_{j-1} + 8y_{j+1} - y_{j+2}) \tag{3.18}$$

und

$$f'(x_j) \approx \frac{1}{60h}(-y_{j-3} + 9y_{j-2} - 45y_{j-1} + 45y_{j+1} - 9y_{j+2} + y_{j+3}). \tag{3.19}$$

□

j	Fehler	j	Fehler	j	Fehler
0	$1.15e-1$	6	$2.70e-7$	12	$7.81e-5$
1	$2.56e-2$	7	$2.81e-8$	13	$7.81e-5$
2	$2.69e-3$	8	$3.03e-9$	14	$2.29e-3$
3	$2.70e-4$	9	$1.30e-7$	15	$1.58e-1$
4	$2.70e-5$	10	$3.52e-7$	16	$8.41e-1$
5	$2.70e-6$	11	$3.52e-7$		

Tabelle 3.6: Vorwärtsgenommene Differenzenquotienten

Den Fehler einer Differenzenformel kann man mit Hilfe des Taylorschen Satzes bestimmen. Für $f \in C^2$ gilt mit einem $\xi \in (x_j, x_j + h)$

$$f(x_j + h) = f(x_j) + f'(x_j)h + \frac{h^2}{2} f''(\xi),$$

d.h.

$$f'(x_j) = \frac{y_{j+1} - y_j}{h} - \frac{h}{2} f''(\xi),$$

und genauso mit einem $\eta \in (x_j - h, x_j)$

$$f'(x_j) = \frac{y_j - y_{j-1}}{h} - \frac{h}{2} f''(\eta).$$

Es gilt also für den Fehler sowohl des vorwärtsgenommenen als auch des rückwärtsgenommenen Differnzenquotienten die Asymptotik $O(h)$.

Definition 3.29 Eine Differenzenapproximation $D_r f(x; h)$ für die Ableitung $f^{(r)}(x)$ mit der Schrittweite h besitzt die **Fehlerordnung** p, falls gilt

$$D_r f(x; h) - f^{(r)}(x) = O(h^p).$$

Vorwärts- und rückwärtsgenommene Differenzenquotienten zur Approximation von $f'(x)$ besitzen also die Fehlerordnung 1.

Für $f \in C^4$ gilt

$$f(x_j \pm h) = f(x_j) \pm h f'(x_j) + \frac{h^2}{2} f''(x_j) \pm \frac{h^3}{6} f'''(x_j) + O(h^4),$$

und daher

$$\frac{y_{j+1} - y_{j-1}}{2h} - f'(x_j) = \frac{h^2}{6} f'''(x_j) + O(h^3).$$

Der zentrale Differenzenquotient besitzt also die Fehlerordnung 2. Genauso erhält man für die Approximationen in (3.18) und (3.19) die Fehlerordnungen 4 und 6.

Bei Rechnung in exakter Arithmetik beschreibt die Fehlerordnung das Verhalten des Fehlers für $h \to 0$. Die Differenzenformeln enthalten alle Differenzen von Funktionswerten von f, die bei kleinem h nahe beieinander liegen. Dies führt beim Rechnen mit endlicher Stellenzahl zu Auslöschungen.

Beispiel 3.30 Wir bestimmen für $f(x) := \cos x$ Näherungen für $f'(1)$ mit dem vorwärtsgenommenen Differenzenquotienten. Tabelle 3.6 enthält die Fehler der Approximationen für $h_j = 10^{-j}$, $j = 0, 1, \ldots, 16$.

Der Fehler fällt also (wie durch die Fehlerordnung 1 vorausgesagt) zunächst bis $h = 10^{-8}$ linear, steigt aber danach durch Auslöschung in der Differenzenformel wieder an. Abbildung 3.4 enthält auf der linken Seite den Graphen des Fehlers in Abhängigkeit von h in doppeltlogarithmischer Darstellung.

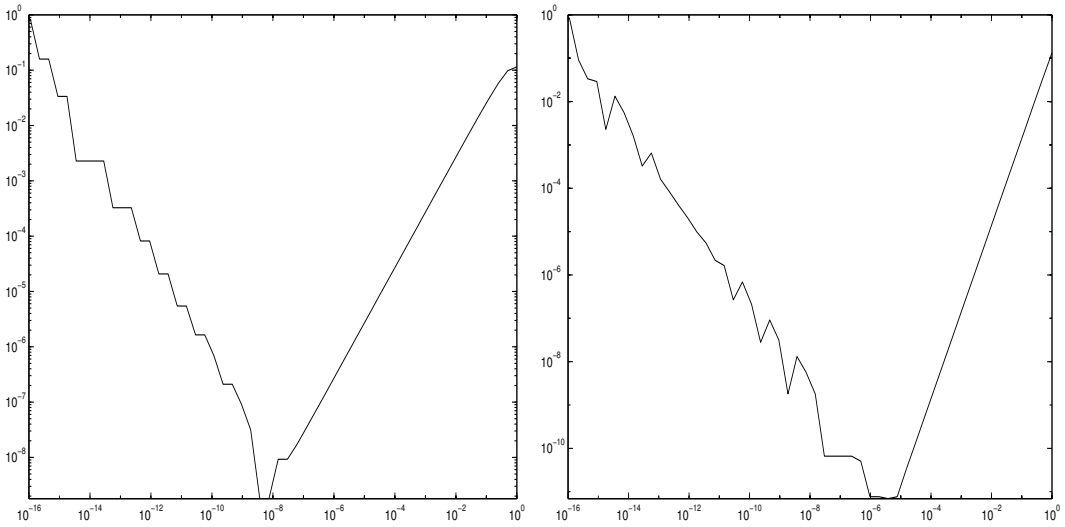

Abbildung 3.4: Differenzenformel Ordnung 1 / Ordnung 2

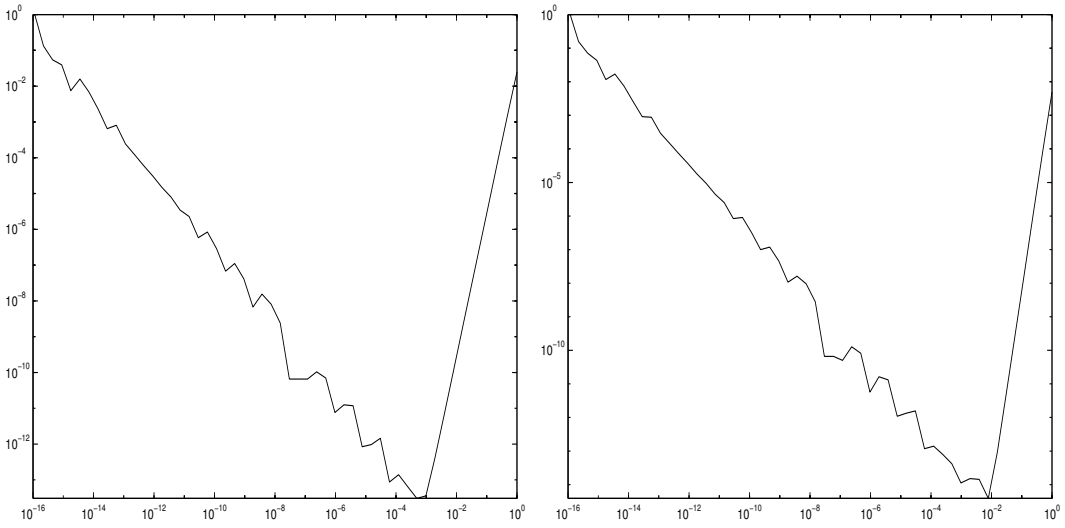

Abbildung 3.5: Differenzenformel Ordnung 4 / Ordnung 6

Für andere Differenzenformeln beobachtet man dasselbe Verhalten. In Abbildung 3.4 rechts ist der Fehler für den zentralen Differenzenquotienten der Ordnung 2 dargestellt und in Abbildung 3.5 für die Formeln (3.18) und (3.19). □

Das Verhalten im letzten Beispiel kann man auf folgende Weise erklären. Wir nehmen an, dass der errechnete Funktionswert $\tilde{y}_j \approx y_j$ die Größe

$$\tilde{y}_j = y_j(1+\delta_j), \quad |\delta_j| \leqslant K\mathbf{u} \tag{3.20}$$

hat, wobei $\mathbf{u}$ die Maschinengenauigkeit bezeichnet und K eine „kleine“ Konstante ist. Dann gilt für den errechneten vorwärtsgenommenen Differenzenquotienten

$$\tilde{D}_1(h) := \mathrm{fl}\left(\frac{\tilde{y}_{j+1}-\tilde{y}_j}{h}\right) = \frac{\tilde{y}_{j+1}-\tilde{y}_j}{h}(1+\varepsilon_1)(1+\varepsilon_2), \quad |\varepsilon_1|, |\varepsilon_2| \leqslant \mathbf{u}.$$

Unter Verwendung von (3.20) erhält man, wenn man Terme der Größenordnung $\mathbf{u}^2$ vernachlässigt,

$$\tilde{D}_1(h) = \frac{y_{j+1}-y_j}{h} + \frac{y_{j+1}\delta_{j+1}-y_j\delta_j}{h} + \frac{y_{j+1}-y_j}{h}(\varepsilon_1+\varepsilon_2),$$

und daher ist der Rundungsfehler bei der Auswertung der Differenzenformel $D_1(h) := (y_{j+1}-y_j)/h$ mit Konstanten C_1, C_2

$$|\tilde{D}_1(h) - D_1(h)| = \left|\frac{y_{j+1}\delta_{j+1}-y_j\delta_j}{h} + \frac{y_{j+1}-y_j}{h}(\varepsilon_1+\varepsilon_2)\right| \leqslant \frac{C_1}{h}\mathbf{u} + C_2\mathbf{u}. \tag{3.21}$$

Da die vorwärtsgenommene Differenzenformel die Ordnung 1 hat, gibt es eine Konstante C_3 mit

$$|D_1(h) - f'(x_j)| \leqslant C_3 h,$$

und daher folgt für den Gesamtfehler

$$|\tilde{D}_1(h) - f'(x_j)| \leqslant \frac{C_1}{h}\mathbf{u} + C_2\mathbf{u} + C_3 h =: \Delta(h). \tag{3.22}$$

Der Graph dieser Funktion hat die Gestalt der Fehlerfunktion in Abbildung 3.4, links: Mit fallendem h fällt die Funktion $\Delta(h)$ bis zum Minimum, das durch

$$\Delta'(h) = -\frac{C_1\mathbf{u}}{h^2} + C_3 = 0$$

charakterisiert ist, d.h. bis

$$h_{\mathrm{opt}} = \sqrt{\frac{C_1}{C_3}} \cdot \sqrt{\mathbf{u}}, \tag{3.23}$$

und steigt danach wieder. In MATLAB ist $\mathbf{u} = 2^{-53} \approx 10^{-16}$, das Minimum des Fehlers muss also in der Größenordnung von 10^{-8} liegen. Abbildung 3.4, links, zeigt, dass dies tatsächlich bei Beispiel 3.30 der Fall ist.

Besitzt die Differenzenformel D_1 die Fehlerordnung m, so bleibt (3.21) richtig, und man erhält entsprechend (3.22) den Gesamtfehler

$$|\tilde{D}_1(h) - f'(x_0)| \leqslant \frac{C_1}{h}\mathbf{u} + C_2\mathbf{u} + C_3 h^m =: \Delta(h). \tag{3.24}$$

In diesem Fall erhält man als Größenordnung der optimalen Schrittweite

$$h_{\mathrm{opt}} = C\mathbf{u}^{1/(m+1)},$$

die ebenfalls durch die Beispiele in Abbildung 3.4, rechts, und Abbildung 3.5 bestätigt werden.

Kapitel 4

Lineare Systeme

4.1 Zerlegung regulärer Matrizen

Wir betrachten das lineare Gleichungssystem

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{A} \in \mathbb{R}^{(n\times n)}, \quad \boldsymbol{b} \in \mathbb{R}^n. \tag{4.1}$$

Es sei $\boldsymbol{A} \in \mathbb{R}^{(n\times n)}$ regulär. Dann existiert (vgl. LA Satz 4.42) eine Permutationsmatrix $\boldsymbol{P}$, eine untere Dreiecksmatrix $\boldsymbol{L}$, deren Diagonalelemente zu 1 normiert sind, und eine obere Dreiecksmatrix $\boldsymbol{R}$ mit

$$\boldsymbol{P}\boldsymbol{A} = \boldsymbol{L}\boldsymbol{R}. \tag{4.2}$$

Diese Zerlegung heißt die **LR Zerlegung** der Matrix $\boldsymbol{A}$. Es ist ferner bekannt (vgl. LA Satz 4.44), dass die Permutationsmatrix $\boldsymbol{P}$ genau dann nicht benötigt wird, wenn alle Hauptuntermatrizen (engl.: principal submatrices)

$$\boldsymbol{A}(1:k, 1:k) := (a_{ij})_{i,j=1,\ldots,k}, \ k = 1, \ldots, n,$$

von $\boldsymbol{A}$ regulär sind. Wir haben hierbei die in MATLAB übliche Notation $\boldsymbol{A}(1:k, 1:k)$ für die Untermatrix von $\boldsymbol{A}$ verwendet, die die erste bis k-te Zeile und erste bis k-te Spalte von $\boldsymbol{A}$ enthält. Ist dies der Fall, so kann man die LR Zerlegung von $\boldsymbol{A}$ mit dem **Gaußschen Eliminationsverfahren** bestimmen. Dabei speichern wir die von Null verschiedenen Elemente von $\boldsymbol{R}$ in dem oberen Dreieck von $\boldsymbol{A}$ und die Elemente unterhalb der Diagonale von $\boldsymbol{L}$ in dem strikten unteren Dreieck von $\boldsymbol{A}$ ab.

Algorithmus 4.1 (LR Zerlegung ohne Pivotsuche)

```
for i=1 : n-1
   for j=i+1 : n
      a(j,i)=a(j,i)/a(i,i);   \% Bestimme i-te Spalte von L
      for k=i+1 : n
         a(j,k)=a(j,k)-a(j,i)*a(i,k);   \% Datiere j-te Zeile auf
         end
      end
   end
```

Ist die LR Zerlegung von $\boldsymbol{A}$ bekannt, so kann man die Lösung des Gleichungssystems (4.1) schreiben als

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}\boldsymbol{R}\boldsymbol{x} =: \boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{R}\boldsymbol{x} = \boldsymbol{y},$$

und durch **Vorwärtseinsetzen**

Algorithmus 4.2 (Vorwärtseinsetzen)

```
for j=1 : n
   y(j)=b(j);
   for k=1 : j-1
      y(j)=y(j)-a(j,k)*y(k);
      end
   end
```

die Lösung $\boldsymbol{y}$ von $\boldsymbol{Ly} = \boldsymbol{b}$ bestimmen, und durch **Rückwärtseinsetzen**

Algorithmus 4.3 (Rückwärtseinsetzen)

```
for j=n : -1 : 1
   x(j)=y(j);
   for k=j+1 : n
      x(j)=x(j)-a(j,k)*x(k);
      end
   x(j)=x(j)/a(j,j);
   end
```

die Lösung $\boldsymbol{x}$ von $\boldsymbol{Rx} = \boldsymbol{y}$, d.h. von $\boldsymbol{Ax} = \boldsymbol{b}$.

Als Aufwand der LR Zerlegung erhält man

$$\sum_{i=1}^{n-1}\Big(\sum_{j=i+1}^{n}\big(1+\sum_{k=i+1}^{n}2\big)\Big)=\sum_{i=1}^{n-1}\Big((n-i)+2(n-i)^2\Big)=\frac{2}{3}n^3+O(n^2).$$

flops (floating point operations). Das Vorwärts- und Rückwärtseinsetzen erfordert offenbar jeweils $O(n^2)$ flops.

Existiert eine singuläre Hauptuntermatrix $\boldsymbol{A}(1:i,1:i)$ der regulären Matrix $\boldsymbol{A}$, so bricht Algorithmus 4.1 ab, da das Pivotelement $a(i,i)$ bei der Elimination der i-ten Variable Null wird. In diesem Fall gibt es ein $a_{ij} \neq 0$, $j > i$, (das im Verlaufe des Algorithmus erzeugt worden ist), und man kann die aktuelle Zeile i mit der Zeile j vertauschen und den Algorithmus fortsetzen. Sammelt man diese Vertauschungen in der Permutationsmatrix $\boldsymbol{P}$, so erhält man am Ende des so modifizierten Algorithmus 4.1 eine LR Zerlegung der Matrix $\boldsymbol{PA}$:

$$\boldsymbol{PA} = \boldsymbol{LR}.$$

Aus Stabilitätsgründen empfiehlt es sich, auch dann eine Vertauschung vorzunehmen, wenn a_{ii} zwar von Null verschieden ist, in der Restmatrix $\boldsymbol{A}(i:n,i:n)$ aber Elemente vorhanden sind, deren Betrag wesentlich größer ist als der Betrag von a_{ii}.

Beispiel 4.4 Es sei

$$\boldsymbol{A} = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}$$

Wir bezeichnen mit $\mathrm{fl}(a)$ die Gleitpunktdarstellung von a. Dann liefert Algorithmus 4.1 bei dreistelliger Rechnung

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 \\ \mathrm{fl}(1/10^{-4}) & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 10^4 & 1 \end{pmatrix}$$

und

$$\boldsymbol{R} = \begin{pmatrix} 10^{-4} & 1 \\ 0 & \mathrm{fl}(1-10^4) \end{pmatrix} = \begin{pmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{pmatrix},$$

und damit

$$\boldsymbol{LR} = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 0 \end{pmatrix}.$$

□

In vielen Fällen reicht es aus, vor dem Annullieren der Elemente der i-ten Spalte $j \in \{i, \ldots, n\}$ zu bestimmen mit $|a_{ji}| \geqslant |a_{ki}|$ für alle $k = i, \ldots, n$ und dann die i-te Zeile mit der j-ten Zeile zu vertauschen. Dieses Vorgehen heißt **Spaltenpivotsuche** Man erhält folgende Modifikation von Algorithmus 4.1:

Algorithmus 4.5 (LR Zerlegung mit Spaltenpivotsuche)

```
for i=1: n-1
   Waehle j >= i mit |a(j,i)| >= |a(k,i)| fuer alle k >= i
   und vertausche die i-te mit der j-ten Zeile
   for j=i+1 : n
      a(j,i)=a(j,i)/a(i,i);
      for k=i+1 : n
         a(j,k)=a(j,k)-a(j,i)*a(i,k);
         end
      end
   end
```

Beispiel 4.6 Für Beispiel 4.4 erhält man mit Algorithmus 4.5

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 \\ 10^{-4} & 1 \end{pmatrix}, \quad \boldsymbol{R} = \begin{pmatrix} 1 & 1 \\ 0 & \mathrm{fl}(1 - 10^{-4}) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

und

$$\boldsymbol{LR} = \begin{pmatrix} 1 & 1 \\ 10^{-4} & 1 + 10^{-4} \end{pmatrix}$$

ist eine gute Approximation von

$$\boldsymbol{PA} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 10^{-4} & 1 \end{pmatrix}.$$

Nicht in allen Fällen führt die Spaltenpivotsuche zum Erfolg. Wendet man Algorithmus 4.5 bei dreistelliger Rechnung auf

$$\tilde{\mathbf{A}} = \begin{pmatrix} 1 & 10^4 \\ 1 & 1 \end{pmatrix}$$

an, so erhält man

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad \boldsymbol{R} = \begin{pmatrix} 1 & 10^4 \\ 0 & \mathrm{fl}(1 - 10^4) \end{pmatrix} = \begin{pmatrix} 1 & 10^4 \\ 0 & -10^4 \end{pmatrix},$$

und hiermit

$$\boldsymbol{LR} = \begin{pmatrix} 1 & 10^4 \\ 1 & 0 \end{pmatrix}.$$ □

Treten, wie in unserem Beispiel, in der Matrix Elemente sehr unterschiedlicher Größenordnung auf, so empfiehlt es sich, eine **vollständige Pivotsuche** auszuführen. In diesem Fall werden im i-ten Eliminationsschritt ein Zeilenindex $j \geqslant i$ und ein Spaltenindex $k \geqslant i$ bestimmt mit $|a_{jk}| \geqslant |a_{\ell m}|$ für alle $\ell, m \geqslant i$, und es wird vor der Elimination die i-te mit der j-ten Zeile und die i-te Spalte mit der k-ten Spalte getauscht. Man erhält dann eine Zerlegung

$$\boldsymbol{PAQ} = \boldsymbol{LR},$$

wobei die Permutationsmatrix $\boldsymbol{P}$ die Zeilenvertauschungen in $\boldsymbol{A}$ vornimmt und die Permutationsmatrix $\boldsymbol{Q}$ die Spaltenvertauschungen.

Ist die reguläre Matrix $\boldsymbol{A}$ symmetrisch und existiert eine LR Zerlegung, so kann man $\boldsymbol{R}$ als Produkt einer Diagonalmatrix $\boldsymbol{D}$ und einer normierten oberen Dreiecksmatrix $\tilde{\mathbf{R}}$ schreiben. Hiermit gilt dann

$$\boldsymbol{A}^T = \tilde{\mathbf{R}}^T \boldsymbol{D}\boldsymbol{L}^T$$

mit einer normierten unteren Dreiecksmatrix $\tilde{\mathbf{R}}^T$ und einer oberen Dreiecksmatrix $\boldsymbol{D}\boldsymbol{L}^T$. Da die LR Zerlegung einer regulären Matrix eindeutig bestimmt ist, folgt $\tilde{\mathbf{R}}^T = \boldsymbol{L}$. Damit kann man $\boldsymbol{A}$ auch schreiben als

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$$

mit einer Diagonalmatrix $\boldsymbol{D}$ und $\boldsymbol{L}$ wie oben. Diese Zerlegung heißt die LDL^T **Zerlegung** von $\boldsymbol{A}$. Ist die symmetrische Matrix $\boldsymbol{A}$ zusätzlich positiv definit, so sind alle Hauptuntermatrizen von $\boldsymbol{A}$ ebenfalls positiv definit, also regulär, und daher existiert in diesem Fall die LDL^T Zerlegung. Ferner sind die Diagonalelemente von $\boldsymbol{D} = \operatorname{diag}\{d_1, \ldots, d_n\}$ positiv, und man kann mit

$$\boldsymbol{C} = \boldsymbol{L} \operatorname{diag}\{\sqrt{d_1}, \ldots, \sqrt{d_n}\}$$

die LDL^T Zerlegung von $\boldsymbol{A}$ schreiben als

$$\boldsymbol{A} = \boldsymbol{C}\boldsymbol{C}^T. \tag{4.3}$$

Diese Zerlegung heißt die **Cholesky Zerlegung** von $\boldsymbol{A}$.

Prinzipiell kann man die Cholesky Zerlegung mit Hilfe des Gaußschen Eliminationsverfahrens bestimmen. Man benötigt dann $\frac{2}{3}n^3 + O(n^2)$ Operationen und n^2 Speicherplätze. Durch direkten Vergleich der Elemente in (4.3) erhält man einen Algorithmus, der mit der Hälfte des Aufwandes auskommt. Da $\boldsymbol{C}$ eine untere Dreiecksmatrix ist, gilt $c_{ij} = 0$ für $1 \leqslant i < j \leqslant n$, und daher ist

$$a_{ij} = \sum_{k=1}^{n} c_{ik}c_{jk} = \sum_{k=1}^{i} c_{ik}c_{jk}.$$

Speziell für $i = j = 1$ bedeutet dies

$$a_{11} = c_{11}^2, \quad \text{d.h. } c_{11} = \sqrt{a_{11}},$$

und für $i = 1$ und $j = 2, \ldots, n$

$$a_{1j} = c_{11}c_{j1}, \quad \text{d.h. } c_{j1} = a_{j1}/c_{11}.$$

Damit ist die erste Spalte von $\boldsymbol{C}$ bestimmt. Sind schon $c_{\mu\nu}$ für $\nu = 1, \ldots, i-1$ und $\mu = \nu, \nu+1, \ldots, n$ bestimmt, so erhält man aus

$$a_{ii} = c_{ii}^2 + \sum_{k=1}^{i-1} c_{ik}^2$$

das i-te Diagonalelement

$$c_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} c_{ik}^2}$$

von $\boldsymbol{C}$, und

$$a_{ij} = c_{ii}c_{ji} + \sum_{k=1}^{i-1} c_{ik}c_{jk}$$

liefert

$$c_{ji} = \frac{1}{c_{ii}}\Big(a_{ij} - \sum_{k=1}^{i-1} c_{ik}c_{jk}\Big), \quad j = i+1, \ldots, n.$$

Damit erhält man das folgende Verfahren zur Bestimmung der Cholesky Zerlegung. Dabei überschreiben wir das untere Dreieck von $\boldsymbol{A}$ durch die wesentlichen Elemente von $\boldsymbol{C}$.

Algorithmus 4.7 (Cholesky Zerlegung)

```
for i=1 : n
   for k=1 : i-1
      a(i,i)=a(i,i)-a(i,k)*a(i,k);
      end
   a(i,i)=sqrt(a(i,i));
   for j=i+1 : n
      for k=1 : i-1
         a(j,i)=a(j,i)-a(i,k)*a(j,k);
         end
      a(j,i)=a(j,i)/a(i,i);
      end
   end
```

Der Aufwand des Cholesky Verfahrens ist

$$\sum_{i=1}^{n}\Big(\sum_{k=1}^{i-1} 2 + \sum_{j=i+1}^{n}\big(1 + \sum_{k=1}^{i-1} 2\big)\Big) = \frac{1}{3}n^3 + O(n^2)$$

flops und n Quadratwurzeln. Besitzt die symmetrische Matrix $\boldsymbol{A}$ eine LDL^T Zerlegung, so kann man diese mit einem ähnlichen Verfahren wie Algorithmus 4.7 bestimmen. Man beachte aber, dass eine Pivotsuche wie beim Gaußschen Eliminationsverfahren die Symmetrie der Matrix zerstört. Man muss also gleichlautende Zeilen- und Spaltenvertauschungen vornehmen. Auf diese Weise kann man nicht immer für eine reguläre Matrix ein von Null verschiedenes Pivotelement erzeugen, wie das Beispiel

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

zeigt. Auch wenn die LDL^T Zerlegung existiert, kann die Übertragung von Algorithmus 4.7 instabil sein.

4.2 Modifikationen des Gaußschen Verfahrens

In Algorithmus 4.1 haben wir die Elimination durchgeführt, indem wir im i-ten Schritt ein geeignetes Vielfaches der i-ten Zeile von den nachfolgenden Zeilen abgezogen haben. In Vektorschreibweise haben wir also

Algorithmus 4.8 (LR Zerlegung; zeilenorientiert)

```
for i=1: n-1
   for j=i+1 : n
      a(j,i)=a(j,i)/a(i,i);
      a(j,i+1:n)=a(j,i+1:n)-a(j,i)*a(i,i+1:n);
      end
   end
```

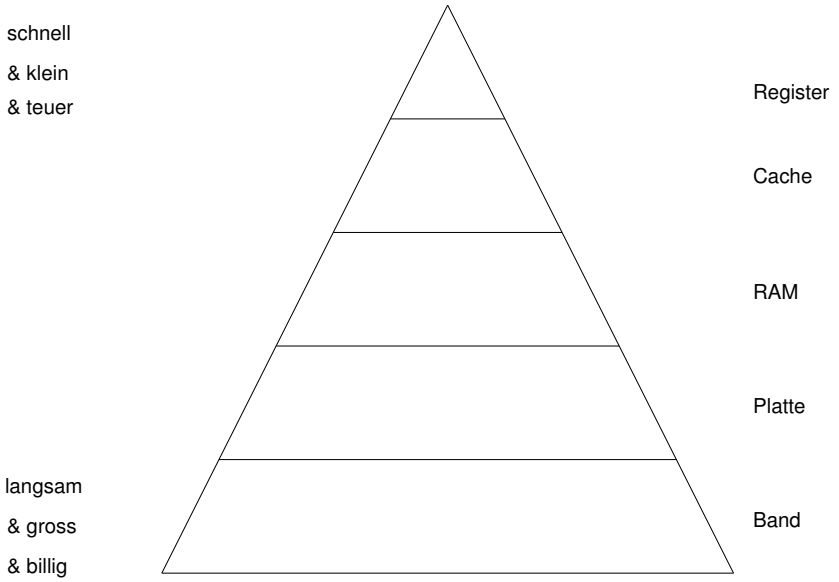

Abbildung 4.1: Hierarchischer Speicher

Vertauscht man die beiden inneren Schleifen, so erhält man eine spaltenorientierte Version des Gaußschen Eliminationsverfahren

Algorithmus 4.9 (LR Zerlegung; spaltenorientiert)

```
for i=1: n-1
   a(i+1:n,i)=a(i+1:n,i)/a(i,i);
   for k=i+1 : n
      a(i+1:n,k)=a(i+1:n,k)-a(i,k)*a(i+1:n,i);
      end
   end
```

Auch die i-Schleife kann man mit der j-Schleife und/oder der k-Schleife vertauschen. Man erhält insgesamt 6 Varianten der Gauß Elimination, die alle von der Zahl der Rechenoperationen her denselben Aufwand besitzen. Sie können sich aber auf verschiedenen Plattformen unter verschiedenen Programmiersprachen sehr unterschiedlich verhalten.

Der Grund hierfür ist, dass Speicher von Rechnern hierarchisch aufgebaut sind, beginnend mit sehr langsamen, sehr großen, sehr billigen Magnetband Speichern, über schnellere, kleinere, teurere Plattenspeicher, den noch schnelleren, noch kleineren, noch teureren Hauptspeicher, den schnellen, kleinen und teuren Cache und die sehr schnellen, sehr kleinen und sehr teuren Register der CPU. Arithmetische und logische Operationen können nur in den Registern ausgeführt werden. Daten, die sich nicht in den Registern befinden und mit denen Operationen ausgeführt werden sollen, können nur in den benachbarten Speicher bewegt werden, wobei der Datentransport zwischen den billigen Speicherarten (also z.B. zwischen der Platte und dem Hauptspeicher) sehr langsam geschieht, während der Transport zwischen den teuren Speichern an der Spitze der Hierarchie sehr schnell geschieht. Insbesondere ist die Geschwindigkeit, mit der arithmetische Operationen ausgeführt werden, sehr viel höher als die Geschwindigkeit, mit der Daten transportiert werden.

Faktoren zwischen 10 und 10000 (je nach Speicherebene) sind nicht ungewöhnlich. Algorithmen müssen also so gestaltet werden, dass der Datentransport zwischen verschiedenen Speicherebenen möglichst klein ist.

Dass die Reihenfolge, in der die Schleifen im Gaußschen Eliminationsverfahren durchlaufen werden, auf die Laufzeit eines Programms einen Einfluss hat, sieht man so ein: Eine Matrix wird in einem langen (eindimensionalen) Array gespeichert. In C geschieht das zeilenweise:

$$\begin{array}{cccccccccc} & a(1,1) & \to & a(1,2) & \to & a(1,3) & \to & \dots & \to & a(1,n) \\ \to & a(2,1) & \to & a(2,2) & \to & a(2,3) & \to & \dots & \to & a(2,n) \\ \to & a(3,1) & \to & a(3,2) & \to & a(3,3) & \to & \dots & \to & a(3,n) \\ & \vdots & & \vdots & & \vdots & & & & \vdots \\ \to & a(n,1) & \to & a(n,2) & \to & a(n,3) & \to & \dots & \to & a(n,n) \end{array}$$

In der zeilenorientierten Version in Algorithmus 4.8 werden in der innersten Schleife Daten verwendet, die im Speicher nahe beieinander liegen. Es sind daher nur wenige Datentransporte zwischen den verschiedenen Speicherebenen erforderlich. Für die spaltenorientierte Version liegen die nacheinander benutzten Daten (wenigstens für große Dimensionen n) sehr weit auseinander, und daher ist ein "cache miss" (das benötigte Wort liegt nicht im Cache, und es müssen zwei Blöcke zwischen dem Cache und dem Hauptspeicher ausgetauscht werden) oder sogar ein "page fault" (das benötigte Wort liegt nicht einmal im Hauptspeicher, und es müssen zwei Seiten zwischen dem Hauptspeicher und der Platte ausgetauscht werden) wahrscheinlicher. In FORTRAN ist die Situation umgekehrt. Matrizen werden spaltenweise gespeichert, und für Algorithmus 4.9 ist die Datenlokalität hoch, während sie in Algorithmus 4.8 gering ist. Um nicht für jeden neuen Rechner neue Software schreiben zu müssen, um die Modularität und Effizienz von Programmen zu erhöhen und um ihre Pflege zu erleichtern, wurden Standardoperationen der (numerischen) linearen Algebra, die basic linear algebra subprograms (BLAS), definiert, und es wurden Standardschnittstellen für ihren Aufruf festgelegt. Diese werden (jedenfalls für Hochleistungsrechner) von den Herstellern auf der Hardwareebene realisiert (nicht zuletzt, da die Hersteller wissen, dass die üblichen Benchmarktests Programme enthalten, die aus den BLAS Routinen aufgebaut sind!). Die Benutzung der BLAS in eigenen Programmen bietet eine Reihe von Vorteilen:

- Die Robustheit von Berechnungen der linearen Algebra wird durch die BLAS erhöht, denn in Ihnen werden Details der Algorithmen und der Implementierung berücksichtigt, die bei der Programmierung eines Anwendungsproblems leicht übersehen werden, wie z.B. die Berücksichtigung von Overflow Problemen.

- Die Portabilität von Programmen wird erhöht, ohne auf Effizienz zu verzichten, denn es werden optimierte Versionen der BLAS auf Rechnern verwendet, für die diese existieren. Für alle anderen Plattformen existieren kompatible Standard Fortran oder C Implementierungen. Diese können als Public Domain Software bezogen werden von

  http://www.netlib.org/blas/

  bzw.

  http://www.netlib.org/clapack/cblas/

  Im ATLAS Projekt (Automatically Tuned Linear Algebra Software) wurden und werden empirische Methoden entwickelt, um Bibliotheken hoher Performance zu erzeugen und zu pflegen, und so in der durch die Software diktierten

Geschwindigkeit Schritt mit der Hardware Entwicklung zu halten. Es werden z.B. für den benutzten Rechner die Größen des Registers und Caches ermittelt und für die Level 2 und Level 3 BLAS die Blockgrößen angepasst. Die im ATLAS Projekt entwickelte BLAS, für die es Fortran und C Interfaces gibt, kann herunter geladen werden von

http://www.netlib.org/atlas/

– Die Lesbarkeit von Programmen wird dadurch erhöht, dass mnemonische Namen für Standardoperationen verwendet werden und der Programmablauf nicht durch Codierungsdetails unterbrochen wird. Dies erleichtert auch die Dokumentation von Programmen.

Die erste Stufe der BLAS Definitionen (**Level 1 BLAS** oder **BLAS1**) wurde 1979 durchgeführt und enthielt Vektor-Vektor Operationen wie das Skalarprodukt oder die Summe eines Vektors und des Vielfachen eines weiteren Vektors. Es wurde eine Namenskonvention eingeführt, die einen drei- bis fünfbuchstabigen, einprägsamen Namen verwendet, dem ein Buchstabe zur Kennzeichnung des Datentyps vorangestellt wird (S, D, C oder Z). Als Beispiele nennen wir nur die Function `_DOT`, für die durch "ALPHA=DDOT(N,X,1,Y,1)" das innere Produkt der doppeltgenauen Vektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ der Dimension n berechnet werden, oder die Subroutine `_AXPY` ("a x plus y") mit dem Aufruf "CAXPY(N,ALPHA,X,1,Y,1)" durch die für die komplexen Vektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ der Dimension n der komplexe Vektor $\alpha\boldsymbol{x}+\boldsymbol{y}$ berechnet wird und im Speicher von $\boldsymbol{y}$ abgelegt wird. Statt der Einsen in den Aufrufen kann man Inkremente angeben, so dass auch innere Produkte der Art

$$\sum_{j=0}^{n-1} a_{1+mj}a_{2+mj}$$

berechnet werden können, also bei spaltenweiser Speicherung einer (m,n)-Matrix $\boldsymbol{A}$ das innere Produkt der ersten und zweiten Zeile.

Beispiel 4.10 Als Beispiel betrachten wir die Bestimmung des inneren Produktes zweier Vektoren der Dimension $n = 10^7$. Wir verwenden (wie auch bei den folgenden Beispielen zur Performance der BLAS Routinen) eine HP 9000/785/C3000 Workstation mit einer CPU 2.0.PA8500 mit der Taktfrequenz 400 MHz und den Fortran90 Compiler f90-HP. Die folgende Tabelle enthält die Laufzeiten eines naiven Codes mit einer Laufanweisung, einer BLAS1 Implementierung in FORTRAN77, die aus der netlib heruntergeladen werden kann, einer BLAS Implementierung, die mit dem Fortran90 Compiler von HP geliefert wird, der BLAS1 Routine aus dem Atlas Projekt und einer optimierten BLAS Routine der veclib von HP.

Implementierung	CPU Zeit	Relation
naiv	0.92	7.9
BLAS (netlib)	0.52	4.5
BLAS (f90-HP)	0.29	2.4
BLAS (ATLAS)	0.23	2.0
veclib	0.12	1.0

□

Die BLAS1 haben zu effizienten Implementierungen von Algorithmen auf skalaren Maschinen geführt, auf Vektorrechnern oder Parallelrechnern werden weitere Standardoperationen benötigt. Man kann z.B. das Produkt einer Matrix $\boldsymbol{A}$ mit einem Vektor $\boldsymbol{x}$ codieren als

Algorithmus 4.11 (Matrix-Vektor Produkt mit DAXPY)

```
Y=zeros(n,1);
for j=1 : n
   DAXPY(n,X(j),A(:,j),1,Y,1)
   end
```

Dabei wird aber nicht berücksichtigt, dass der Ergebnisvektor $\boldsymbol{y}$ im Register gehalten werden könnte. BLAS1 hat also den Nachteil, dass zu wenige (nützliche) flops ausgeführt werden im Verhältnis zu (nutzlosen) Datenbewegungen. Für die Ausführung eines `_AXPY`s sind z.B. $3n+1$ Speicherzugriffe erforderlich (die Vektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ und der Skalar α müssen gelesen werden, und der Ergebnisvektor $\boldsymbol{y}$ muss geschrieben werden) und es werden $2n$ flops ausgeführt. Das Verhältnis ist also 2/3. Dies wurde 1988 verbessert mit der Definition der **Level 2 BLAS** oder **BLAS2**, die Matrix-Vektor Operationen enthalten, wie z.B. subroutine `_GEMV`, durch die $\boldsymbol{y} \leftarrow \alpha \boldsymbol{A}\boldsymbol{x} + \beta \boldsymbol{y}$ berechnet wird, das Produkt einer Dreiecksmatrix mit einem Vektor, Rang-1- oder Rang-2-Aufdatierungen von Matrizen wie $\boldsymbol{A} \leftarrow \alpha \boldsymbol{x}\boldsymbol{y}^T + \boldsymbol{A}$, oder Lösungen von Gleichungssystemen mit Dreiecksmatrizen. Für die Subroutine `_GEMV` sind im n-dimensionalen Fall n^2+3n+2 Speicherzugriffe erforderlich, und es werden $2n^2$ flops ausgeführt. Das Verhältnis ist also 2.

Algorithmus 4.9 ist schon fast eine BLAS2 Implementierung der LR Zerlegung. Man muss nur noch die Modifikationen der Spalten zu einer Rang-1-Modifikation des unteren $(n-i, n-i)$-Blocks umschreiben.

Algorithmus 4.12 (LR Zerlegung; BLAS2)

```
for i=1: n-1
   a(i+1:n,i)=a(i+1:n,i)/a(i,i);
   a(i+1:n,i+1:n)=a(i+1:n,i+1:n)-a(i+1:m,i)*a(i,i+1:n);
   end
```

Beispiel 4.13 Als Beispiel zur Performance der BLAS2 betrachten wir die Bestimmung des Produktes einer $(10^4, 10^3)$-Matrix mit einem Vektor. Hierfür erhält man die Laufzeiten

Implementierung	CPU Zeit	Relation
naiv	4.32	61.7
BLAS2 (netlib)	1.39	19.9
BLAS2 (f90-HP)	0.62	8.9
BLAS2 (ATLAS)	0.17	2.4
veclib	0.07	1.0

□

In **Level 3 BLAS** oder **BLAS3** wurden schließlich Matrix-Matrix Operationen wie $\boldsymbol{C} \leftarrow \alpha \boldsymbol{A}\boldsymbol{B} + \beta \boldsymbol{C}$ aufgenommen. Um die Wiederverwendung von Daten in den Registern oder im Cache in möglichst hohem Maße zu erreichen, werden die beteiligten Matrizen in Blöcke unterteilt und die Operationen blockweise ausgeführt. Auf diese Weise erreicht man, dass für die obige Operation bei $4n^2 + 2$ Speicherzugriffen $2n^3 + O(n^2)$ flops ausgeführt werden, das Verhältnis von nützlicher Arbeit zu Speicherzugriffen also auf $n/2$ steigt.

Beispiel 4.14 Als Beispiel zur Performance der BLAS3 betrachten wir die Bestimmung des Produktes zweier $(10^3, 10^3)$ Matrizen. Hierfür erhält man die Laufzeiten

Implementierung	CPU Zeit	Relation
naiv	462.42	247.34
BLAS3 (netlib)	320.00	171.1
BLAS3 (f90-HP)	7.25	3.9
BLAS3 (ATLAS)	4.26	2.3
veclib	1.87	1.0

□

4.3 Störungen linearer Systeme

Wir wollen nun den Begriff der **Kondition** einer Matrix einführen, deren Wert—wie oben angedeutet—verantwortlich ist für die numerische Behandelbarkeit eines linearen Gleichungssystems. Wir betrachten dazu neben dem linearen Gleichungssystem

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{4.4}$$

mit der regulären Matrix $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ ein gestörtes System

$$(\boldsymbol{A} + \Delta\boldsymbol{A})(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b} + \Delta\boldsymbol{b} \tag{4.5}$$

und fragen uns, wie die Lösung des ursprünglichen Systems auf diese Störungen reagiert.

Bemerkung 4.15 Kleine Störungen kann man bei der praktischen Lösung linearer Gleichungssysteme grundsätzlich nicht ausschließen. Einerseits können die Eingangsdaten des Systems aus Messungen herrühren und somit a priori fehlerbehaftet sein. Andererseits wird man bei der Benutzung eines elektronischen Rechners durch die endliche Genauigkeit der Zahlendarstellungen auf dem Rechner Eingabefehler machen müssen.

Man muss also grundsätzlich davon ausgehen, dass man mit gestörten Systemen anstelle der wirklich zu lösenden Systeme rechnen muss. Allerdings kann man meistens annehmen, dass die Störungen klein sind. □

Bevor wir die Wirkung von Störungen untersuchen können, benötigen wir noch einige Aussagen über sogenannte Matrixnormen, durch die wir die Größe einer Matrix messen.

Definition 4.16 Es sei $\boldsymbol{A} \in \mathbb{C}^{(m\times n)}$ eine Matrix, $\|\cdot\|_n$ eine Vektornorm auf $\mathbb{C}^n$ und $\|\cdot\|_m$ eine Vektornorm auf $\mathbb{C}^m$. Dann heißt

$$\|\boldsymbol{A}\|_{m,n} := \max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_m}{\|\boldsymbol{x}\|_n}$$

die **Matrixnorm** von $\boldsymbol{A}$, die den Normen $\|\cdot\|_n$ und $\|\cdot\|_m$ zugeordnet ist.

Bemerkung 4.17 Wegen $\dfrac{\|\boldsymbol{A}\boldsymbol{x}\|_m}{\|\boldsymbol{x}\|_n} = \left\|\boldsymbol{A}\left(\dfrac{\boldsymbol{x}}{\|\boldsymbol{x}\|_n}\right)\right\|_m$ genügt es, das Maximum über Vektoren der Länge 1 zu erstrecken:

$$\|\boldsymbol{A}\|_{m,n} = \max\{\|\boldsymbol{A}\boldsymbol{y}\|_m \,:\, \|\boldsymbol{y}\|_n = 1\}.$$

Die Existenz des Maximums (daß es also einen Vektor $\boldsymbol{x}$ mit $\|\boldsymbol{x}\|_n = 1$ und $\|\boldsymbol{A}\boldsymbol{x}\|_m = \|\boldsymbol{A}\|_{m,n}$ gibt) folgt aus der Stetigkeit der Abbildung $\boldsymbol{x} \mapsto \|\boldsymbol{A}\boldsymbol{x}\|$. □

Bemerkung 4.18 $\|\cdot\|_{m,n}$ ist eine Norm auf $\mathbb{C}^{(m\times n)}$, denn

$$\begin{aligned}
\|\boldsymbol{A}\|_{m,n} = 0 \Leftrightarrow \|\boldsymbol{A}\boldsymbol{x}\|_m = 0 \quad \forall \boldsymbol{x} \in \mathbb{C}^n \Leftrightarrow \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \quad \forall \boldsymbol{x} \in \mathbb{C}^n \Leftrightarrow \boldsymbol{A} = \boldsymbol{O},\\
\|\lambda\boldsymbol{A}\|_{m,n} &= \max\{\|\lambda\boldsymbol{A}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_n = 1\} = \max\{|\lambda|\cdot\|\boldsymbol{A}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_n = 1\}\\
&= |\lambda|\cdot\|\boldsymbol{A}\|_{m,n},\\
\|\boldsymbol{A}+\boldsymbol{B}\|_{m,n} &= \max\{\|\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_n = 1\}\\
&\leqslant \max\{\|\boldsymbol{A}\boldsymbol{x}\|_m + \|\boldsymbol{B}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_n = 1\}\\
&\leqslant \max\{\|\boldsymbol{A}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_n = 1\} + \max\{\|\boldsymbol{B}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_n = 1\}\\
&= \|\boldsymbol{A}\|_{m,n} + \|\boldsymbol{B}\|_{m,n}.
\end{aligned}$$

Diese ist zusätzlich **submultiplikativ** im folgenden Sinne:

$$\|\boldsymbol{A}\boldsymbol{x}\|_m \leqslant \|\boldsymbol{A}\|_{m,n}\cdot\|\boldsymbol{x}\|_n \text{für alle } \boldsymbol{x}\in\mathbb{C}^n, \boldsymbol{A}\in\mathbb{C}^{(m\times n)}, \tag{4.6}$$

$$\|\boldsymbol{A}\boldsymbol{B}\|_{m,p} \leqslant \|\boldsymbol{A}\|_{m,n}\cdot\|\boldsymbol{B}\|_{n,p} \text{für alle } \boldsymbol{A}\in\mathbb{C}^{(m\times n)}, \boldsymbol{B}\in\mathbb{C}^{(n,p)}. \tag{4.7}$$

Die Ungleichung (4.6) folgt sofort aus der Definition 4.16; denn es ist $\|\boldsymbol{A}\|_{m,n} \geqslant \dfrac{\|\boldsymbol{A}\boldsymbol{x}\|_m}{\|\boldsymbol{x}\|_n}$ für alle $\boldsymbol{x}\in\mathbb{C}^n$.
Die Ungleichung (4.7) erschließt man mit (4.6) wie folgt: Für alle $\boldsymbol{x}\in\mathbb{C}^p$ ist

$$\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|_m = \|\boldsymbol{A}(\boldsymbol{B}\boldsymbol{x})\|_m \leqslant \|\boldsymbol{A}\|_{m,n}\cdot\|\boldsymbol{B}\boldsymbol{x}\|_n \leqslant \|\boldsymbol{A}\|_{m,n}\cdot\|\boldsymbol{B}\|_{n,p}\cdot\|\boldsymbol{x}\|_p,$$

Also ist $\|\boldsymbol{A}\boldsymbol{B}\|_{m,p} = \max\{\|\boldsymbol{A}\boldsymbol{B}\boldsymbol{x}\|_m \,:\, \|\boldsymbol{x}\|_p = 1\} \leqslant \|\boldsymbol{A}\|_{m,n}\cdot\|\boldsymbol{B}\|_{n,p}$. □

Bemerkung 4.19 Die Matrixnorm $\|\boldsymbol{A}\|_{m,n}$ ist die kleinste nichtnegative reelle Zahl μ, mit der

$$\|\boldsymbol{A}\boldsymbol{x}\|_m \leqslant \mu\cdot\|\boldsymbol{x}\|_n \quad \forall \boldsymbol{x}\in\mathbb{C}^n$$

ist. $\|\boldsymbol{A}\|_{m,n}$ ist demnach die maximale Verlängerung, die ein $\boldsymbol{x}$ durch Abbildung mit $\boldsymbol{A}$ erfahren kann, wobei $\boldsymbol{x}$ selbst in der $\|\cdot\|_n$-Norm und $\boldsymbol{A}\boldsymbol{x}$ in der Norm $\|\cdot\|_m$ des Bildraumes gemessen wird. □

Wir betrachten nun nur noch den Fall, daß im Urbildraum und im Bildraum dieselbe Norm verwendet wird, auch wenn beide Räume verschiedene Dimension haben, und verwenden für die Matrixnorm dasselbe Symbol wie für die Vektornorm. Für $\boldsymbol{A}\in\mathbb{R}^{(5,9)}$ bezeichnet also $\|\boldsymbol{A}\|_\infty$ die Matrixnorm von $\boldsymbol{A}$ gemäß Definition 4.16, wenn sowohl im Urbildraum $\mathbb{R}^9$ als auch im Bildraum $\mathbb{R}^5$ die Maximumnorm verwendet wird.

Der folgende Satz 4.20 enthält die den wichtigsten Vektornormen zugeordneten Matrixnormen:

Satz 4.20 *Es sei* $\boldsymbol{A}\in\mathbb{C}^{(m\times n)}$ *gegeben.*

1. *Die* ***Zeilensummennorm***

$$\|\boldsymbol{A}\|_\infty := \max_{i=1,\ldots,m}\sum_{j=1}^{n}|a_{ij}|$$

ist der Maximumnorm $\|\boldsymbol{x}\|_\infty := \max\limits_{i=1,\ldots,n}|x_i|$ *zugeordnet.*

2. *Die* ***Spektralnorm***

$$\|\boldsymbol{A}\|_2 := \max\{\sqrt{\lambda} \,:\, \lambda \text{ ist Eigenwert von } \boldsymbol{A}^H\boldsymbol{A}\}$$

ist der Euklidischen Norm zugeordnet.

3. *Die* ***Spaltensummennorm***

$$\|\boldsymbol{A}\|_1 := \max_{j=1,\dots,n} \sum_{i=1}^{m} |a_{ij}|$$

ist der Summennorm $\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$ *zugeordnet.*

Beweis: (1): Für alle $\boldsymbol{x} \in \mathbb{C}^n$ gilt

$$\|\boldsymbol{A}\boldsymbol{x}\|_\infty = \max_{i=1,\dots,m} |\sum_{j=1}^{n} a_{ij}x_j| \leqslant \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}| \cdot |x_j| \leqslant \|\boldsymbol{x}\|_\infty \cdot \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|,$$

und daher

$$\|\boldsymbol{A}\|_\infty \leqslant \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|. \tag{4.8}$$

Sei $k \in \{1,\dots,m\}$ mit $\sum_{j=1}^{n} |a_{ij}| \leqslant \sum_{j=1}^{n} |a_{kj}|$ für alle i. Wir definieren $\boldsymbol{x} \in \mathbb{C}^n$ durch $x_j := 1$, falls $a_{kj} = 0$, und $x_j := \overline{a_{kj}}/|a_{kj}|$, sonst. Dann gilt $\|\boldsymbol{x}\|_\infty = 1$ und

$$\|\boldsymbol{A}\boldsymbol{x}\|_\infty = \max_{i=1,\dots,m} |\sum_{j=1}^{n} a_{ij}x_j| \geqslant |\sum_{j=1}^{n} a_{kj}x_j| = \Big|\sum_{j=1}^{n} |a_{kj}|\Big| = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|,$$

und daher

$$\|\boldsymbol{A}\|_\infty = \max\{\|\boldsymbol{A}\boldsymbol{y}\|_\infty : \|\boldsymbol{y}\|_\infty = 1\} \geqslant \|\boldsymbol{A}\boldsymbol{x}\|_\infty \geqslant \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|,$$

zusammen mit (4.8) also die Behauptung.

(2): Es ist $\boldsymbol{A}^H\boldsymbol{A} \in \mathbb{C}^{(n\times n)}$ eine Hermitesche Matrix, und daher ist nach dem Rayleighschen Prinzip

$$\max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_2^2} = \max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\boldsymbol{x}^H\boldsymbol{A}^H\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H\boldsymbol{x}}$$

der maximale Eigenwert von $\boldsymbol{A}^H\boldsymbol{A}$.

(3): Zeigt man ähnlich wie (1). ■

Beispiel 4.21

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0.1 & -0.1 \\ 0.1 & 2 & -0.4 \\ 0.2 & 0.4 & 3 \end{pmatrix}.$$

Es ist

$$\|\boldsymbol{A}\|_\infty = \max\{1.2, 2.5, 3.6\} = 3.6$$
$$\|\boldsymbol{A}\|_1 = \max\{1.3, 2.5, 3.5\} = 3.5$$

Die Spektralnorm von $\boldsymbol{A}$ ist die Quadratwurzel aus dem maximalen Eigenwert von

$$\boldsymbol{A}^T\boldsymbol{A} = \begin{pmatrix} 1.05 & 0.38 & 0.46 \\ 0.38 & 4.17 & 0.39 \\ 0.46 & 0.39 & 9.17 \end{pmatrix}.$$

Nach einiger Rechnung ergibt dies

$$\|\boldsymbol{A}\|_2 \approx \sqrt{9.2294} \approx 3.04.$$ □

Die Spektralnorm einer Matrix ist nur schwer zu berechnen. Für viele Zwecke genügt jedoch die folgende Abschätzung:

Satz 4.22 *Für alle* $\boldsymbol{A} \in \mathbb{C}^{(m\times n)}$ *gilt*

$$\|\boldsymbol{A}\|_2 \leqslant \|\boldsymbol{A}\|_S := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}.$$

Bemerkung 4.23 $\|\boldsymbol{A}\|_S$ heißt die **Schur Norm** oder **Erhard Schmidt Norm** (oder—speziell in der anglo-amerikanischen Literatur—**Frobenius Norm** der Matrix $\boldsymbol{A}$). $\|\cdot\|_S$ ist zwar eine Vektornorm auf $\mathbb{C}^{(m\times n)}$ (= Euklidische Norm auf $\mathbb{C}^{m\cdot n}$), aber keine einer Vektornorm zugeordnete Matrixnorm, denn für eine solche gilt im Fall $n = m$ stets

$$\|\boldsymbol{E}\| = \max\{\|\boldsymbol{E}\boldsymbol{x}\| \,:\, \|\boldsymbol{x}\| = 1\} = 1.$$

Es ist aber $\|\boldsymbol{E}\|_S = \sqrt{n}$. □

Beweis:(von Satz 4.22) Wegen der Cauchy-Schwarzschen Ungleichung gilt für alle $\boldsymbol{x} \in \mathbb{C}^n$

$$\|\boldsymbol{A}\boldsymbol{x}\|_2^2 = \sum_{i=1}^{m}\Big|\sum_{j=1}^{n}a_{ij}x_j\Big|^2 \leqslant \Big(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\Big)\sum_{j=1}^{n}|x_j|^2 = \|\boldsymbol{A}\|_S^2\cdot\|\boldsymbol{x}\|_2^2,$$

und daher gilt $\|\boldsymbol{A}\|_2 \leqslant \|\boldsymbol{A}\|_S$. ■

Beispiel 4.24 Für die Matrix $\boldsymbol{A}$ aus Beispiel 4.21 erhält man

$$\|\boldsymbol{A}\|_2 \leqslant \|\boldsymbol{A}\|_S = \sqrt{14.39} \leqslant 3.80.$$ □

Wir betrachten nun das gestörte System (4.5) und nehmen an, dass die Störungen der Matrixelemente so klein sind, dass auch die Matrix $\boldsymbol{A} + \Delta\boldsymbol{A}$ regulär ist (dass dies für hinreichend kleine Störungen bei regulärer Matrix $\boldsymbol{A}$ überhaupt stets möglich ist, wird aus dem Satz 4.25 unten folgen). Löst man mit dieser Annahme (4.5) nach $\Delta\boldsymbol{x}$ auf, so erhält man für den durch die Störungen $\Delta\boldsymbol{A}$ und $\Delta\boldsymbol{b}$ hervorgerufenen absoluten Fehler wegen $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$

$$\begin{aligned}\Delta\boldsymbol{x} &= (\boldsymbol{A}+\Delta\boldsymbol{A})^{-1}(\Delta\boldsymbol{b} - \Delta\boldsymbol{A}\boldsymbol{x})\\ &= (\boldsymbol{I}+\boldsymbol{A}^{-1}\Delta\boldsymbol{A})^{-1}\boldsymbol{A}^{-1}(\Delta\boldsymbol{b} - \Delta\boldsymbol{A}\boldsymbol{x}).\end{aligned}$$

Mit einer beliebigen Vektornorm $\|\cdot\|$ auf dem $\mathbb{R}^n$ und der gleich bezeichneten zugeordneten Matrixnorm kann man also abschätzen

$$\|\Delta\boldsymbol{x}\| \leqslant \|(\boldsymbol{I}+\boldsymbol{A}^{-1}\Delta\boldsymbol{A})^{-1}\|\cdot\|\boldsymbol{A}^{-1}\|\left(\|\Delta\boldsymbol{b}\| + \|\Delta\boldsymbol{A}\|\cdot\|\boldsymbol{x}\|\right).$$

Für $\boldsymbol{b} \neq \boldsymbol{0}$ und folglich $\boldsymbol{x} \neq \boldsymbol{0}$ erhält man daraus für den relativen Fehler $\|\Delta\boldsymbol{x}\|/\|\boldsymbol{x}\|$ die Abschätzung

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \|(\boldsymbol{I}+\boldsymbol{A}^{-1}\Delta\boldsymbol{A})^{-1}\|\cdot\|\boldsymbol{A}^{-1}\|\left(\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{x}\|} + \|\Delta\boldsymbol{A}\|\right). \tag{4.9}$$

Um in dieser Ungleichung $\|(\boldsymbol{I}+\boldsymbol{A}^{-1}\Delta\boldsymbol{A})^{-1}\|$ weiter abzuschätzen, benötigen wir das folgende Störungslemma, das gleichzeitig darüber Auskunft gibt, unter wie großen Störungen der Matrixelemente die Existenz der Inversen für die gestörte Matrix noch gesichert ist.

Satz 4.25 (Störungslemma) *Es sei $\boldsymbol{B} \in \mathbb{R}^{(n,n)}$ und es gelte für eine beliebige einer Vektornorm zugeordneten Matrixnorm die Ungleichung $\|\boldsymbol{B}\| < 1$. Dann ist die Matrix $\boldsymbol{I} - \boldsymbol{B}$ regulär, und es gilt*

$$\|(\boldsymbol{I} - \boldsymbol{B})^{-1}\| \leqslant \frac{1}{1 - \|\boldsymbol{B}\|}.$$

Beweis: Für alle $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq \boldsymbol{0}$, gilt

$$\|(\boldsymbol{I} - \boldsymbol{B})\boldsymbol{x}\| \geqslant \|\boldsymbol{x}\| - \|\boldsymbol{B}\boldsymbol{x}\| \geqslant \|\boldsymbol{x}\| - \|\boldsymbol{B}\|\|\boldsymbol{x}\| = (1 - \|\boldsymbol{B}\|)\|\boldsymbol{x}\| > 0,$$

d.h. $(\boldsymbol{I} - \boldsymbol{B})\boldsymbol{x} = \boldsymbol{0}$ ist nur für $\boldsymbol{x} = \boldsymbol{0}$ lösbar, und daher ist $\boldsymbol{I} - \boldsymbol{B}$ regulär. Die Abschätzung der Norm der Inversen von $\boldsymbol{I} - \boldsymbol{B}$ erhält man so:

$$\begin{aligned} 1 &= \|(\boldsymbol{I} - \boldsymbol{B})^{-1}(\boldsymbol{I} - \boldsymbol{B})\| = \|(\boldsymbol{I} - \boldsymbol{B})^{-1} - (\boldsymbol{I} - \boldsymbol{B})^{-1}\boldsymbol{B}\| \\ &\geqslant \|(\boldsymbol{I} - \boldsymbol{B})^{-1}\| - \|(\boldsymbol{I} - \boldsymbol{B})^{-1}\boldsymbol{B}\| \\ &\geqslant \|(\boldsymbol{I} - \boldsymbol{B})^{-1}\| - \|(\boldsymbol{I} - \boldsymbol{B})^{-1}\| \cdot \|\boldsymbol{B}\| = (1 - \|\boldsymbol{B}\|) \cdot \|(\boldsymbol{I} - \boldsymbol{B})^{-1}\|. \end{aligned}$$

■

Mit dem Störungslemma (mit $\boldsymbol{B} = -\boldsymbol{A}^{-1}\Delta\boldsymbol{A}$) können wir nun den relativen Fehler des gestörten Problems aus (4.9) weiter abschätzen, wobei wir $\|\boldsymbol{A}^{-1}\Delta\boldsymbol{A}\| \leqslant \|\boldsymbol{A}^{-1}\| \cdot \|\Delta\boldsymbol{A}\|$ und $\|\boldsymbol{b}\| = \|\boldsymbol{A}\boldsymbol{x}\| \leqslant \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|$ beachten.

$$\begin{aligned} \frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} &\leqslant \frac{\|\boldsymbol{A}^{-1}\|}{1 - \|\boldsymbol{A}^{-1}\| \cdot \|\Delta\boldsymbol{A}\|} \left(\|\boldsymbol{A}\| \frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} + \|\Delta\boldsymbol{A}\| \right) \\ &\leqslant \frac{\|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\|}{1 - \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\| \frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}} \left(\frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|} + \frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \right). \end{aligned} \tag{4.10}$$

Aus der Abschätzung (4.10) liest man ab, dass für kleine Störungen der Matrixelemente (so dass der Nenner nicht wesentlich von 1 abweicht) der relative Fehler der rechten Seite und der Matrixelemente um den Faktor $\|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\|$ verstärkt wird. Diesen Verstärkungsfaktor nennen wir die Kondition der Matrix $\boldsymbol{A}$.

Definition 4.26 Es sei $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ regulär und $\|\cdot\|_p$ eine einer (gleichbezeichneten) Vektornorm zugeordnete Matrixnorm auf $\mathbb{R}^{(n,n)}$. Dann heißt

$$\kappa_p(\boldsymbol{A}) := \|\boldsymbol{A}^{-1}\|_p \cdot \|\boldsymbol{A}\|_p$$

die **Kondition** der Matrix $\boldsymbol{A}$ (oder des linearen Gleichungssystems (4.4)) bezüglich der Norm $\|\cdot\|_p$.

Wir fassen unsere Überlegungen zusammen:

Satz 4.27 *Es seien $\boldsymbol{A}, \Delta\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ und $\boldsymbol{b}, \Delta\boldsymbol{b} \in \mathbb{R}^n$, $\boldsymbol{b} \neq \boldsymbol{0}$, gegeben, so dass $\boldsymbol{A}$ regulär ist und $\|\boldsymbol{A}^{-1}\| \cdot \|\Delta\boldsymbol{A}\| < 1$ mit einer Vektornorm zugeordneten Matrixnorm $\|\cdot\|$ gilt. Dann existiert neben der Lösung des linearen Gleichungssystems (4.4) auch die Lösung $\boldsymbol{x} + \Delta\boldsymbol{x}$ des gestörten Systems (4.5), und es gilt mit der Kondition $\kappa(\boldsymbol{A}) := \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\|$ die Abschätzung*

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{\kappa(\boldsymbol{A})}{1 - \kappa(\boldsymbol{A}) \cdot \frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}} \left(\frac{\|\Delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|} + \frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \right).$$

Bemerkung 4.28 Für jede reguläre Matrix $\boldsymbol{A}$ und jede Norm $\|\cdot\|$ gilt $\kappa(\boldsymbol{A}) \geqslant 1$, denn

$$1 = \|\boldsymbol{I}\| = \|\boldsymbol{A}\boldsymbol{A}^{-1}\| \leqslant \|\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\| = \kappa(\boldsymbol{A}).$$ □

Bemerkung 4.29 Werden die Rechnungen bei der Lösung eines linearen Gleichungssystems mit der Mantissenlänge ℓ ausgeführt, so haben die Daten von $\boldsymbol{A}$ und $\boldsymbol{b}$ bereits einen relativen Eingabefehler der Größe $5 \cdot 10^{-\ell}$. Gilt $\kappa(\boldsymbol{A}) = 10^{\gamma}$, so ist (abgesehen von Rundungsfehlern, die sich im numerischen Lösungsalgorithmus ergeben) für die numerische Lösung mit einem relativen Fehler der Größe $5 \cdot 10^{\gamma-\ell}$ zu rechnen. Grob gesprochen verliert man also beim Lösen eines linearen Gleichungssystems γ Stellen, wenn die Koeffizientenmatrix eine Kondition der Größenordnung 10^{γ} besitzt. Dieser Verlust von Stellen ist nicht dem jeweilig verwendeten Algorithmus zuzuschreiben. Er ist problemimmanent. □

Beispiel 4.30 Wir betrachten das lineare Gleichungssystem

$$\begin{pmatrix} 1 & 1 \\ 1 & 0.999 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 2 \\ 1.999 \end{pmatrix},$$

das offensichtlich die Lösung $\boldsymbol{x} = (1,1)^T$ besitzt. Für den Vektor $\boldsymbol{x} + \Delta\boldsymbol{x} := (5, -3.002)^T$ gilt

$$\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \begin{pmatrix} 1.998 \\ 2.001002 \end{pmatrix} =: \boldsymbol{b} + \Delta\boldsymbol{b}.$$

Es ist also

$$\frac{\|\Delta\boldsymbol{b}\|_\infty}{\|\boldsymbol{b}\|_\infty} = 1.001 \cdot 10^{-3} \quad \text{und} \quad \frac{\|\Delta\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty} = 4.002,$$

und daher gilt für die Kondition

$$\kappa_\infty(\boldsymbol{A}) \geqslant \frac{4.002}{1.001}\, 10^3 \approx 3998.$$

Tatsächlich gilt $\boldsymbol{A}^{-1} = \begin{pmatrix} -999 & 1000 \\ 1000 & -1000 \end{pmatrix}$ und daher $\kappa_\infty(\boldsymbol{A}) = 4000$. Man sieht an diesem Beispiel, dass die Abschätzung in (4.10) scharf ist. □

Der nächste Satz gibt eine geometrische Charakterisierung der Konditionszahl. Er sagt, dass der relative Abstand einer regulären Matrix zur nächsten singulären Matrix in der Euklidischen Norm gleich dem Reziproken der Kondition ist.

Satz 4.31 *Es sei $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ regulär. Dann gilt*

$$\min\left\{ \frac{\|\Delta\boldsymbol{A}\|_2}{\|\boldsymbol{A}\|_2} \,:\, \boldsymbol{A} + \Delta\boldsymbol{A} \text{ singulär} \right\} = \frac{1}{\kappa_2(\boldsymbol{A})}.$$

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 4.4. ■

4.4 Software für lineare Gleichungssysteme

Sehr hochwertige Public Domain Software ist in den Bibliotheken **LAPACK** und **ScaLAPACK** erhältlich unter der Adresse

http://www.netlib.org/lapack/

bzw.

http://www.netlib.org/scalapack/

Die Fortran 77 Bibliothek LAPACK (und die Übertragungen in andere Sprachen: die C-Version CLAPACK , die C++ Version LAPACK++ und die Fortran 90 Version LAPACK90, die ebenfalls in der Netlib frei erhältlich sind,) ist für PCs, Workstations, Vektorrechner oder Parallelrechner mit gemeinsamen Speicher geeignet, ScaLAPACK für Parallelrechner mit verteiltem Speicher oder vernetzte Workstations. Beide Bibliotheken wurden unter Benutzung von BLAS3 geschrieben. Der Quellcode ist frei zugänglich und sehr gut dokumentiert. Die kommerziellen Bibliotheken ISML oder NAG verwenden (zum Teil geringfügig modifizierte) LAPACK Routinen. MATLAB 6.1 verwendet LAPACK Routinen.

Kapitel 5

Lineare Ausgleichsprobleme

5.1 Normalgleichungen

Wir betrachten in diesem Abschnitt das **lineare Ausgleichsproblem**

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min \tag{5.1}$$

mit gegebenem $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$, $m \geqslant n$, und $\boldsymbol{b} \in \mathbb{R}^m$.

Notwendig für eine Lösung des Ausgleichsproblems (5.1) ist, dass der Gradient des Funktionals

$$\phi(\boldsymbol{x}) = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

verschwindet, d.h. $2\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{0}$ oder

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}. \tag{5.2}$$

Die Gleichungen (5.2) heißen die **Normalgleichungen** des Ausgleichsproblems (5.1). Dass jede Lösung von (5.2) auch das Ausgleichsproblem (5.1) löst, sieht man so: Es ist für alle $\boldsymbol{h} \in \mathbb{R}^n$

$$\begin{aligned}\|\boldsymbol{A}(\boldsymbol{x}+\boldsymbol{h}) - \boldsymbol{b}\|_2^2 &= (\boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}\boldsymbol{h} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}\boldsymbol{h} - \boldsymbol{b}) \\ &= \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \|\boldsymbol{A}\boldsymbol{h}\|_2^2 + 2\boldsymbol{h}^T(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{b}) \\ &= \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \|\boldsymbol{A}\boldsymbol{h}\|_2^2 \geqslant \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2.\end{aligned}$$

Schließlich ist die Matrix $\boldsymbol{A}^T\boldsymbol{A}$ genau dann regulär, wenn die Matrix $\boldsymbol{A}$ den Rang n besitzt. Damit ist das Ausgleichsproblem genau dann eindeutig lösbar, wenn $\text{Rang}\boldsymbol{A} = n$ gilt. Wir haben also

Satz 5.1 *Die Lösungen des Ausgleichsproblems*

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min$$

sind genau die Lösungen der Normalgleichungen

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}.$$

(5.1) *ist genau dann eindeutig lösbar, wenn $\boldsymbol{A}$ linear unabhängige Spalten besitzt.*

Besitzt $\boldsymbol{A}$ unabhängige Spalten, so ist die Koeffizientenmatrix $\boldsymbol{A}^T\boldsymbol{A}$ positiv definit, und man kann daher die Normalgleichungen unter Benutzung der Cholesky Zerlegung lösen. Diese Methode erfordert zur Aufstellung der Normalgleichungen (unter Beachtung der Symmetrie von $\boldsymbol{A}^T\boldsymbol{A}$) $\frac{1}{2}n(n+1) + n$ innere Produkte von Vektoren der Länge m, also $n^2m + 3nm$ flops und zur Lösung der Normalgleichungen $\frac{1}{3}n^3 + O(n^2)$ flops. Das Verfahren ist geeignet bei Problemen mit kleinem n. Bei größerem n können Stabilitätsprobleme auftreten und eines der folgenden Verfahren ist vorzuziehen.

5.2 Orthogonale Zerlegung von Matrizen

Ist $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$, $m \geqslant n$, eine gegebene Matrix mit linear unabhängigen Spalten, so gibt es eine Matrix $\boldsymbol{Q} \in \mathbb{R}^{(m,n)}$ mit orthogonalen Spalten und eine obere Dreiecksmatrix $\boldsymbol{R} \in \mathbb{R}^{(n,n)}$, so dass gilt

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}. \tag{5.3}$$

(5.3) heißt eine **QR Zerlegung** der Matrix $\boldsymbol{A}$.

Wir haben bereits in der Vorlesung Lineare Algebra gesehen, dass man die QR Zerlegung einer Matrix mit Hilfe der Orthogonalisierung nach Gram-Schmidt oder durch Multiplikation mit geeigneten Householder Transformationen bestimmen kann.

Beim Gram-Schmidt Verfahren werden im j-ten Schritt, d.h. nachdem die ersten $j-1$ Spalten $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^{j-1}$ von $\boldsymbol{Q}$ bereits bestimmt wurden, von der j-ten Spalte $\boldsymbol{a}^j$ von $\boldsymbol{A}$ die Anteile in Richtung von $\boldsymbol{q}^i$, $i = 1, \ldots, j-1$, abgezogen und der so bestimmte, zu den $\boldsymbol{q}^i$ orthogonale Vektor normiert. Dabei werden zugleich die Elemente $r_{ij} := (\boldsymbol{q}^i)^T \boldsymbol{a}^j$ bestimmt.

Algorithmus 5.2 (Klassische Gram-Schmidt Orthogonalisierung)

```
for i=1 : n
   q(:,i)=a(:,i);
   for j=1 : i-1
      r(j,i)=q(:,j)'*a(:,i);
      q(:,i)=q(:,i)-r(j,i)*q(:,j);
      end
   r(i,i)=‖q(:,i)‖_2;
   q(:,i)=q(:,i)/r(i,i);
   end
```

Wir haben vorausgesetzt, dass die Spalten von $\boldsymbol{A}$ linear unabhängig sind. Ist dies nicht der Fall, so wird Algorithmus 5.2 mit $r_{ii} = 0$ für ein i abbrechen. Diese Abfrage wird man natürlich noch in einen Algorithmus einbauen.

Sind die Spalten von $\boldsymbol{A}$ nahezu linear abhängig, so ist das oben angegebene **klassische Gram-Schmidt Verfahren** numerisch instabil. Die berechneten Vektoren $\boldsymbol{q}^j$ sind also nicht orthogonal. Etwas besser wird die Stabilität, wenn man die Berechnung der r_{ij} ersetzt durch $r_{ij} = (\boldsymbol{q}^j)^T \boldsymbol{q}^i$. Diese dem klassischen Gram-Schmidt Verfahren äquivalente Methode heißt **modifiziertes Gram-Schmidt Verfahren**.

Algorithmus 5.3 (Modifizierte Gram-Schmidt Orthogonalisierung)

```
for i=1 : n
   q(:,i)=a(:,i);
   for j=1 : i-1
      r(j,i)=q(:,j)'*q(:,i);
      q(:,i)=q(:,i)-r(j,i)*q(:,j);
      end
   r(i,i)=‖q(:,i)‖_2;
   q(:,i)=q(:,i)/r(i,i);
   end
```

Beispiel 5.4 Das folgende Beispiel von Björck zeigt die Überlegenheit des modifizierten Gram-Schmidt Verfahrens. Sei

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix}$$

wobei ε so klein ist, dass $\mathrm{fl}(1+\varepsilon^2) = 1$ gilt. Dann erhält man mit dem klassischen und dem modifizierten Gram-Schmidt Verfahren (bis auf Normierung der Spalten) die Matrizen

$$Q_{kGS} = \begin{pmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix} \quad \text{und} \quad Q_{mGS} = \begin{pmatrix} 1 & 0 & 0 \\ \varepsilon & -\varepsilon & -\varepsilon/2 \\ 0 & \varepsilon & -\varepsilon/2 \\ 0 & 0 & \varepsilon \end{pmatrix}.$$

Für die minimalen Winkel φ_{kGS} und φ_{mGS} zwischen zwei Spalten gilt

$$\cos\varphi_{kGS} = \frac{1}{2} \quad \text{und} \quad \cos\varphi_{mGS} = -\frac{\varepsilon}{\sqrt{2}}. \qquad \square$$

Stabiler als das Gram-Schmidt Verfahren sind Methoden zur Bestimmung der QR Zerlegung einer Matrix, die auf der Multiplikation von $\boldsymbol{A}$ mit geeigneten orthogonalen Matrizen beruhen. Wir betrachten zunächst die Orthogonalisierung mit Hilfe von Householder Matrizen.

Definition 5.5 Es sei $\boldsymbol{w} \in \mathbb{R}^n$ mit $\|\boldsymbol{w}\|_2 = 1$. Dann heißt die Matrix

$$\boldsymbol{H} := \boldsymbol{I} - 2\boldsymbol{w}\boldsymbol{w}^T$$

Householder Matrix.

Die Householder Matrix $\boldsymbol{H}$ beschreibt eine Spiegelung an der Hyperebene $\boldsymbol{w}^\perp$.

Offensichtlich ist $\boldsymbol{H}$ ist eine symmetrische und orthogonale Matrix, d.h. $\boldsymbol{H}^T = \boldsymbol{H}$ und $\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{H}^2 = \boldsymbol{I}$.

Die explizite Matrixgestalt von $\boldsymbol{H}$ wird außerordentlich selten benötigt. $\boldsymbol{H} = \boldsymbol{I} - \beta\boldsymbol{u}\boldsymbol{u}^T$ ist vollständig charakterisiert durch einen Normalenvektor $\boldsymbol{u}$ der Hyperebene, an der gespiegelt wird, und durch den Skalierungsfaktor $\beta = 2/\|\boldsymbol{u}\|_2^2$. Sind diese beiden bekannt, so kann man eine Multiplikation eines Vektors $\boldsymbol{x}$ mit $\boldsymbol{H}$ ausführen gemäß

$$\boldsymbol{H}\boldsymbol{x} = \boldsymbol{x} - \beta(\boldsymbol{u}^T\boldsymbol{x})\boldsymbol{u}.$$

Es sind also ein inneres Produkt und ein _axpy, d.h. $4n$ flops, erforderlich. Wegen $\boldsymbol{H}^{-1} = \boldsymbol{H}^T = \boldsymbol{H}$ gilt dasselbe für das Lösen eines Gleichungssystems mit der Koeffizientenmatrix $\boldsymbol{H}$.

Um die Orthogonalisierung einer Matrix mit Householder Transformationen auszuführen, benötigen wir zu gegebenem Vektor $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \neq \boldsymbol{0}$, eine Householder Matrix $\boldsymbol{H}$ mit

$$\boldsymbol{H}\boldsymbol{x} = \pm\|\boldsymbol{x}\|_2\boldsymbol{e}^1, \ \boldsymbol{e}^1 = (1, 0, \ldots, 0)^T.$$

In der Vorlesung Lineare Algebra wurde gezeigt (und dies rechnet man auch leicht nach), dass für den Fall, dass $\boldsymbol{x}$ kein Vielfaches von $\boldsymbol{e}^1$ ist, die Householder Matrizen

$$\boldsymbol{H}_\pm = \boldsymbol{I} - \beta\boldsymbol{w}_\pm\boldsymbol{w}_\pm^T,$$

die durch die Normalenvektoren

$$\boldsymbol{w}_\pm = \boldsymbol{x} \pm \|\boldsymbol{x}\|_2\boldsymbol{e}^1,$$

bestimmt sind, das Gewünschte leisten. Ist $\boldsymbol{x}$ ein Vielfaches des ersten Einheitsvektors $\boldsymbol{e}^1$, so ist einer der beiden Vektoren der Nullvektor, und zwar gilt $\tilde{w}_- = \boldsymbol{0}$ im Fall $x_1 > 0$ und $\tilde{w}_+ = \boldsymbol{0}$ im Fall $x_1 < 0$. Ist $\boldsymbol{x} \approx c\boldsymbol{e}^1$, so erhält man für $\tilde{w}_-$ im Fall $c > 0$ und für $\tilde{w}_+$ im Fall $c < 0$ Auslöschung. Um diese zu vermeiden, wählen wir daher stets

$$\boldsymbol{H} = \boldsymbol{I} - \frac{2}{\|\tilde{w}\|_2^2}\tilde{w}\tilde{w}^T, \quad \tilde{w} = \boldsymbol{x} + \mathrm{sign}(x_1)\|\boldsymbol{x}\|_2\boldsymbol{e}^1.$$

Damit erhalten wir den folgenden Algorithmus zur Berechnung der Matrix $\boldsymbol{H} = \boldsymbol{I} - \beta\boldsymbol{u}\boldsymbol{u}^T$, die $\boldsymbol{x}$ in $\mathrm{span}(\boldsymbol{e}^1)$ abbildet:

Algorithmus 5.6 (Householder Vektor)

```
function [u,β]=householder(x)
   μ=x(2:n)'*x(2:n);
   u=[1; x(2:n)];
   if μ == 0
      β=0;
   else
      u(1)=x(1)+sign(x(1))sqrt(x(1)^2 + μ);
      β=2/(u(1)^2+μ);
   end
```

Wir verwenden nun Householder Matrizen, um die Matrix $\boldsymbol{A}$ orthogonal auf obere Dreiecksgestalt zu transformieren. Sei dazu $\boldsymbol{H}_1$ wie oben beschrieben gewählt, so dass

$$\boldsymbol{H}_1 \boldsymbol{a}^1 = \pm \|\boldsymbol{a}^1\|_2 \boldsymbol{e}^1 =: r_{11}\boldsymbol{e}^1.$$

Dann gilt mit $\boldsymbol{P}_1 = \boldsymbol{H}_1$ (und mit einer Matrix $\boldsymbol{A}_1 \in \mathbb{R}^{(m-1,n-1)}$)

$$\boldsymbol{P}_1\boldsymbol{A} = \begin{pmatrix} r_{11} & r(1,2:n) \\ \boldsymbol{0} & \boldsymbol{A}_1 \end{pmatrix}.$$

Ist nach j Schritten, $1 \leqslant j < n$, die Gestalt

$$\left(\begin{array}{cccc|cccc} r_{11} & r_{12} & \dots & r_{1j} & r_{1,j+1} & r_{1,j+2} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2j} & r_{2,j+1} & r_{2,j+2} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & r_{jj} & r_{j,j+1} & r_{j,j+2} & \dots & r_{j,n} \\ \hline 0 & 0 & \dots & 0 & r_{j+1,j+1} & r_{j+1,j+2} & \dots & r_{j+1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & r_{m,j+1} & r_{m,j+2} & \dots & r_{m,n} \end{array}\right)$$

erreicht, so wählen wir eine Householder Matrix $\boldsymbol{H}_{j+1} \in \mathbb{R}^{(m-j,n-j)}$, so dass $\boldsymbol{H}_{j+1} r(j+1:m, j+1)$ eine Vielfaches des ersten Einheitsvektors ist, und setzen hiermit

$$\boldsymbol{P}_{j+1} = \begin{pmatrix} \boldsymbol{I}_j & \boldsymbol{O}_{j,n-j} \\ \boldsymbol{O}_{m-j,j} & \boldsymbol{H}_{j+1} \end{pmatrix}.$$

Dann gilt

$$\boldsymbol{P}_{j+1}\boldsymbol{P}_j \dots \boldsymbol{P}_1\boldsymbol{A} = \left(\begin{array}{ccccc|ccc} r_{11} & r_{12} & \dots & r_{1j} & r_{1,j+1} & r_{1,j+2} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2j} & r_{2,j+1} & r_{2,j+2} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & r_{jj} & r_{j,j+1} & r_{j,j+2} & \dots & r_{j,n} \\ 0 & 0 & \dots & 0 & r_{j+1,j+1} & r_{j+1,j+2} & \dots & r_{j+1,n} \\ \hline 0 & 0 & \dots & 0 & 0 & r_{j+2,j+2} & \dots & r_{j+2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & 0 & r_{m,j+2} & \dots & r_{m,n} \end{array}\right).$$

Nach n Schritten ist damit die obere Dreiecksgestalt erreicht (wobei wir vorausgesetzt haben, dass die Matrix $\boldsymbol{A}$ linear unabhängige Spalten besitzt, da sonst das Verfahren vorher abgebrochen wäre).

Setzt man

$$\boldsymbol{Q}^T := \boldsymbol{P}_n\boldsymbol{P}_{n-1}\dots\boldsymbol{P}_1,$$

so ist $\boldsymbol{Q}$ eine orthogonale Matrix, und es gilt

$$\boldsymbol{A} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O}_{m-n,n} \end{pmatrix} = \boldsymbol{P}_1 \boldsymbol{P}_2 \dots \boldsymbol{P}_n \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O}_{m-n,n} \end{pmatrix}.$$

Die Householder Matrizen $\boldsymbol{H}_j$ (und damit die Faktoren $\boldsymbol{P}_j$ in $\boldsymbol{Q}$) sind jeweils durch Vektoren $\boldsymbol{u}^j$ und Skalare β_j vollständig bestimmt. Die $\boldsymbol{u}^j$ können (bis auf die erste Komponente) im Verlauf eines Algorithmus unterhalb der Diagonalen von $\boldsymbol{A}$ gespeichert werden. Tatsächlich müssen diese Elemente wegen der Gestalt der $\boldsymbol{u}^j$ nicht einmal geändert werden. Die ersten Komponenten von $\boldsymbol{u}^j$ und die β_j müssen in einem zusätzlichen Array gespeichert werden. Die Elemente von $\boldsymbol{R}$ können das obere Dreieck von $\boldsymbol{A}$ einschließlich der Diagonale überschreiben.

Algorithmus 5.7 (QR Zerlegung mit Householder Matrizen)

```
for i=1 : min(m-1,n)
   [u,β]=householder(a(i:m,i));
   a(i:m,i+1:n)=a(i:m,i+1:n)-βu*(u'*a(i:m,i+1:n));
   a(i,i)=a(i,i)-βu(1)u'*a(i:m,i);
   uu(i)=u(1);
   be(i)=β;
   end
```

Man zählt leicht ab, dass dieser Algorithmus im wesentlichen $2n^2m - \frac{2}{3}n^3$ flops benötigt für die QR Zerlegung von $\boldsymbol{A}$.

Eine weitere Möglichkeit zur Bestimmung der QR Zerlegung bieten die Givens Rotationen. Es ist bekannt, dass eine Rotation im $\mathbb{R}^2$ um den Winkel θ gegeben ist durch

$$\boldsymbol{x} \mapsto \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \boldsymbol{x}.$$

Entsprechend kann man eine Multiplikation eines Vektors $\boldsymbol{x} \in \mathbb{R}^n$ mit der Matrix

$$\boldsymbol{R}(i,j,\theta) = \begin{pmatrix} \boldsymbol{I}_{i-1} & \boldsymbol{0} & \boldsymbol{O} & \boldsymbol{0} & \boldsymbol{O} \\ \boldsymbol{0}^T & \cos\theta & \boldsymbol{0}^T & -\sin\theta & \boldsymbol{0}^T \\ \boldsymbol{O} & \boldsymbol{0} & \boldsymbol{I}_{j-i-1} & \boldsymbol{0} & \boldsymbol{O} \\ \boldsymbol{0}^T & \sin\theta & \boldsymbol{0}^T & \cos\theta & \boldsymbol{0}^T \\ \boldsymbol{O} & \boldsymbol{0} & \boldsymbol{O} & \boldsymbol{0} & \boldsymbol{I}_{n-j} \end{pmatrix}$$

als Rotation in der von $\boldsymbol{e}^i$ und $\boldsymbol{e}^j$ aufgespannten Ebene auffassen. $\boldsymbol{R}(i,j,\theta)$ heißt eine **Givens Rotation** oder seltener auch **Jacobi Rotation** . Diese Abbildungen wurden 1958 von Givens [30] benutzt, um eine Matrix orthogonal auf obere Dreiecksgestalt zu transformieren. Jacobi [40] verwandte diese Abbildung schon 1846 zur Lösung des vollständigen symmetrischen Eigenwertproblems.

Es sei nun für $\boldsymbol{x} \in \mathbb{R}^n$ die j-te Komponente x_j von Null verschieden. Wir wählen θ so, dass

$$\cos\theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}} \quad \text{und} \quad \sin\theta = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}.$$

Dann gilt

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix} = \begin{pmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{pmatrix},$$

und daher wird in $\boldsymbol{R}(i,j,\theta)\boldsymbol{x}$ die j-te Komponente annulliert. Damit ist klar, wie man mit einer Folge von Multiplikationen mit Givens Rotationen eine QR Zerlegung einer Matrix $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ bestimmen kann.

Man annulliert nacheinander die Elemente der ersten Spalte unterhalb der Diagonale mit geeigneten Rotationen $\boldsymbol{R}(1,2,\theta_{12})$, $\boldsymbol{R}(1,3,\theta_{13})$, $\ldots$, $\boldsymbol{R}(1,m,\theta_{1m})$, und dann mit Rotationen $\boldsymbol{R}(i,j,\theta_{ij})$, $i=2,\ldots,n$, $j=i+1,\ldots,m$ die Elemente der weiteren Spalten von links nach rechts und von oben nach unten.

Analog kann man auch die sog. **Givens Reflexionen** verwenden, die ausgehend von der Spiegelung

$$\boldsymbol{x} \mapsto \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \boldsymbol{x}$$

im $\mathbb{R}^2$ analog definiert werden.

Ein Vorteil der Verwendung von Givens Rotationen oder Reflexionen gegenüber den Householder Transformationen zur QR Zerlegung einer Matrix $\boldsymbol{A}$ besteht darin, dass man auf übersichtliche Weise gewisse Nullen in $\boldsymbol{A}$ berücksichtigen und damit die Arbeit vermindern kann.

Ein Nachteil ist, dass die direkte Implementierung des beschriebenen Verfahrens doppelt so teuer ist wie die QR Zerlegung mit Householder Matrizen. Man beachte aber, dass Gentleman [28] und Hammarling [35] eine Methode, die **schnelle Givens Transformation**, angegeben haben, mit der der Aufwand genauso hoch ist, wie mit Householder Matrizen. Diese wurde in der BLAS1 verwendet.

Ist eine QR Zerlegung

$$\boldsymbol{A} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O}_{m-n,n} \end{pmatrix}$$

der Matrix $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ bekannt, so ist es leicht, das lineare Ausgleichsproblem

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min!$$

zu lösen. Da die Multiplikation eines Vektors mit einer orthogonalen Matrix seine Euklidische Länge nicht verändert, gilt

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \|\boldsymbol{Q}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2 = \left\| \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O}_{m-n,n} \end{pmatrix} \boldsymbol{x} - \boldsymbol{Q}^T\boldsymbol{b} \right\|_2$$

Mit

$$\boldsymbol{Q}^T\boldsymbol{b} =: \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix}, \quad \boldsymbol{b}_1 \in \mathbb{R}^n,\ \boldsymbol{b}_2 \in \mathbb{R}^{m-n}$$

folgt

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \|\boldsymbol{R}\boldsymbol{x} - \boldsymbol{b}_1\|_2^2 + \|\boldsymbol{b}_2\|_2^2.$$

Den zweiten Summanden kann man durch die Wahl von $\boldsymbol{x}$ nicht beeinflussen. Daher ist die Lösung des Ausgleichsproblems

$$\boldsymbol{x} = \boldsymbol{R}^{-1}\boldsymbol{b}_1,$$

und der Defekt ist $\|\boldsymbol{b}_2\|$.

5.3 Singulärwertzerlegung

Eine für viele Anwendungen wichtige Zerlegung einer Matrix ist die Singulärwertzerlegung.

Definition 5.8 Sei $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$. Gilt für orthogonale Matrizen $\boldsymbol{U} \in \mathbb{R}^{(m,m)}$ und $\boldsymbol{V} \in \mathbb{R}^{(n,n)}$ und eine Diagonalmatrix $\boldsymbol{\Sigma} = (\sigma_i \delta_{ij})_{i,j} \in \mathbb{R}^{(m,n)}$

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{5.4}$$

so heißt (5.4) eine **Singulärwertzerlegung** (**SVD**[1]) von $\boldsymbol{A}$.

Beispiel 5.9 Ist $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ symmetrisch mit den Eigenwerten $\mu_1, \ldots, \mu_n$ und ist $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n$ ein zugehöriges System von orthonormalen Eigenvektoren, so gilt mit der Diagonalmatrix $\boldsymbol{\Sigma} := \operatorname{diag}\{\mu_1, \ldots, \mu_n\}$ und mit $\boldsymbol{V} := (\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n)$

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

und dies ist eine Singulärwertzerlegung von $\boldsymbol{A}$. □

Setzt man $\boldsymbol{U} = (\boldsymbol{u}^1, \ldots, \boldsymbol{u}^m)$ und $\boldsymbol{V} = (\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n)$, so ist (5.4) äquivalent zu

$$\boldsymbol{A}\boldsymbol{v}^i = \begin{cases} \sigma_i \boldsymbol{u}^i, & i = 1, \ldots, \min(m,n), \\ \boldsymbol{0}, & i = \min(m,n)+1, \ldots, n. \end{cases} \tag{5.5}$$

Ohne Einschränkung können die σ_i nichtnegativ gewählt werden. Ist etwa $\sigma_j < 0$, so ersetze man die j-te Spalte $\boldsymbol{v}^j$ von $\boldsymbol{V}$ durch $-\boldsymbol{v}^j$. Ferner können wir durch Umordnung von Zeilen von $\boldsymbol{V}^T$ bzw. Spalten von $\boldsymbol{U}$ erreichen, dass $\sigma_1 \geqslant \sigma_2 \geqslant \ldots$ gilt. Wir werden nun nur noch Singulärwertzerlegungen betrachten, in denen die Diagonalelmente $\sigma_1, \ldots, \sigma_{\min(m,n)}$ nichtnegativ und der Größe nach geordnet sind.

Definition 5.10 Es seien in einer Singulärwertzerlegung (5.4) von $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ die Diagonalelemente $\sigma_1, \ldots, \sigma_{\min(m,n)}$ von $\boldsymbol{\Sigma}$ nicht negativ. Dann heißen die nichtnegativen[2] σ_i die **Singulärwerte** oder die **singulären Werte** von $\boldsymbol{A}$.

Beispiel 5.11 (Fortsetzung von Beispiel 5.9)
Es sei $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ symmetrisch und μ_i und $\boldsymbol{v}^i$ wie in Beispiel 5.9, wobei die Reihenfolge so gewählt ist, dass $|\mu_1| \geqslant |\mu_2| \geqslant \cdots \geqslant |\mu_r| > \mu_{r+1} = \cdots = \mu_n = 0$ gilt.

Es sei $\boldsymbol{u}^i := \boldsymbol{v}^i$, falls $\mu_i \geqslant 0$, und $\boldsymbol{u}^i := -\boldsymbol{v}^i$, falls $\mu_i < 0$, und hiermit $\boldsymbol{U} := (\boldsymbol{u}^1, \ldots, \boldsymbol{u}^n)$. Dann gilt

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \quad \boldsymbol{\Sigma} = (|\mu_i|\delta_{ij}),$$

und $|\mu_1|, \ldots, |\mu_r|$ und 0 (mehrfach) sind die singulären Werte von $\boldsymbol{A}$. □

Aus (5.4) folgt

$$\boldsymbol{A}^T = \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T; \tag{5.6}$$

besitzt also $\boldsymbol{A}$ eine Singulärwertzerlegung, so auch $\boldsymbol{A}^T$, und die singulären Werte stimmen überein. (5.6) kann man (entsprechend (5.5)) schreiben als

$$\boldsymbol{A}^T\boldsymbol{u}^i = \begin{cases} \sigma_i \boldsymbol{v}^i, & i = 1, \ldots, \min(m,n), \\ \boldsymbol{0}, & i = \min(m,n)+1, \ldots, m. \end{cases}$$

Aus (5.4) und (5.6) folgt

$$\begin{aligned} \boldsymbol{A}^T\boldsymbol{A} &= \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T, \\ \boldsymbol{A}\boldsymbol{A}^T &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\boldsymbol{U}^T, \end{aligned} \tag{5.7}$$

d.h. die Quadrate der singulären Werte von $\boldsymbol{A}$ (und $\boldsymbol{A}^T$) sind Eigenwerte von $\boldsymbol{A}^T\boldsymbol{A}$ und $\boldsymbol{A}\boldsymbol{A}^T$, und $\{\boldsymbol{v}^1, \ldots, \boldsymbol{v}^n\}$ bzw. $\{\boldsymbol{u}^1, \ldots, \boldsymbol{u}^m\}$ ist ein Orthonormalsystem von Eigenvektoren von $\boldsymbol{A}^T\boldsymbol{A}$ bzw. $\boldsymbol{A}\boldsymbol{A}^T$.

[1] **SVD** ist die Abkürzung der englischen Bezeichnung *Singular Value Decomposition*. Da diese Abkürzung meistens auch in der deutschen Literatur verwendet wird, schließen wir uns dieser Sprachregelung an.

[2] Diese Definition weicht von der Definition in alten Skriptversionen und derjenigen im Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß ab. Dort werden nur die *positiven* σ_i als Singulärwerte definiert.

Dies zeigt, dass die singulären Werte σ_i eindeutig bestimmt sind, nicht aber die Matrizen $\boldsymbol{U}$ und $\boldsymbol{V}$, da mehrfache Eigenwerte von $\boldsymbol{A}\boldsymbol{A}^T$ und $\boldsymbol{A}^T\boldsymbol{A}$ auftreten können.

Satz 5.12 *Jedes $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ besitzt eine Singulärwertzerlegung.*

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 5.3. ■

Der folgende Satz 5.13 sagt, welche Bedeutungen die in der Singulärwertzerlegung von $\boldsymbol{A}$ auftretenden Größen für die Matrix $\boldsymbol{A}$ haben:

Satz 5.13 *Ist die Singulärwertzerlegung von $\boldsymbol{A}$ durch (5.4) gegeben und gilt $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \cdots = \sigma_{\min(m,n)} = 0$, so ist*

(i) *r der Rang von $\boldsymbol{A}$,*

(ii) *$Kern(\boldsymbol{A}) := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\} = span\{\boldsymbol{v}^{r+1}, \ldots, v^n\}$,*

(iii) *$Bild(\boldsymbol{A}) := \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^n\} = span\{\boldsymbol{u}^1, \ldots, \boldsymbol{u}^r\}$,*

(iv)

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}^i (\boldsymbol{v}^i)^T = \boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^T$$

mit $\boldsymbol{U}_r = (\boldsymbol{u}^1, \ldots, \boldsymbol{u}^r)$, $\boldsymbol{V}_r = (\boldsymbol{v}^1, \ldots, \boldsymbol{v}^r)$, $\boldsymbol{\Sigma}_r = diag(\sigma_1, \ldots, \sigma_r)$,

(v)

$$\|\boldsymbol{A}\|_S^2 := \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 = \sum_{i=1}^{r} \sigma_i^2,$$

(vi)

$$\|\boldsymbol{A}\|_2 := \sigma_1.$$

Beweis: *(i)*: Da die Multiplikation mit den regulären Matrizen $\boldsymbol{U}^T$ und $\boldsymbol{V}$ den Rang nicht verändert, gilt $\text{Rang}\boldsymbol{A} = \text{Rang}\boldsymbol{\Sigma} = r$.

(ii): Es ist $\boldsymbol{V}^T\boldsymbol{v}^i = \boldsymbol{e}^i$. Somit ist

$$\boldsymbol{A}\boldsymbol{v}^i = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{v}^i = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{e}^i = \boldsymbol{0} \text{ für } i = r+1, \ldots, n.$$

Also gilt

$$\boldsymbol{v}^{r+1}, \ldots, \boldsymbol{v}^n \in \text{Kern}(\boldsymbol{A}).$$

Da $\dim \text{Kern}(\boldsymbol{A}) = n - r$, bilden diese Vektoren eine Basis von $\text{Kern}(\boldsymbol{A})$.

(iii): Wegen $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ ist

$$\text{Bild}(\boldsymbol{A}) = \boldsymbol{U} \cdot \text{Bild}(\boldsymbol{\Sigma}) = \boldsymbol{U} \cdot \text{span}(\boldsymbol{e}^1, \ldots, \boldsymbol{e}^r) = \text{span}(\boldsymbol{u}^1, \ldots, \boldsymbol{u}^r).$$

Punkt *(iv)*: Durch Block-Matrix-Multiplikation erhält man

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{pmatrix} \boldsymbol{u}^1 & \ldots & \boldsymbol{u}^m \end{pmatrix} \boldsymbol{\Sigma} \begin{pmatrix} (\boldsymbol{v}^1)^T \\ \vdots \\ (\boldsymbol{v}^n)^T \end{pmatrix} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}^i (\boldsymbol{v}^i)^T.$$

(v): Es sei $\boldsymbol{A} = (\boldsymbol{a}^1, \ldots, a^n)$. Da die orthogonale Matrix $\boldsymbol{U}^T$ die Euklidische Länge nicht verändert, gilt

$$\|\boldsymbol{A}\|_S^2 = \sum_{i=1}^{n} \|\boldsymbol{a}^i\|_2^2 = \sum_{i=1}^{n} \|\boldsymbol{U}^T\boldsymbol{a}^i\|_2^2 = \|\boldsymbol{U}^T\boldsymbol{A}\|_S^2.$$

Durch entsprechende Argumentation mit den Zeilen von $\boldsymbol{U}^T\boldsymbol{A}$ erhält man

$$\|\boldsymbol{A}\|_S^2 = \|\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V}\|_S^2 = \|\boldsymbol{\Sigma}\|_S^2 = \sum_{i=1}^{r} \sigma_i^2.$$

(*vi*): $\|\boldsymbol{A}\|_2$ ist ein singulärer Wert von $\boldsymbol{A}$, d.h. $\|\boldsymbol{A}\|_2 \leqslant \sigma_1$, und

$$\|\boldsymbol{A}\|_2 = \max\{\|\boldsymbol{A}\boldsymbol{x}\|_2 \,:\, \|\boldsymbol{x}\|_2 = 1\} \geqslant \|\boldsymbol{A}\boldsymbol{v}^1\|_2 = \sigma_1.$$

■

Bemerkung 5.14 Besitzt die reguläre Matrix $\boldsymbol{A}$ die Singulärwertzerlegung $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, so gilt $\|\boldsymbol{A}\|_2 = \sigma_1$, und wegen $\boldsymbol{A}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T$ ist $\|\boldsymbol{A}^{-1}\|_2 = \dfrac{1}{\sigma_n}$. Daher ist die Kondition von $\boldsymbol{A}$ bzgl. der Euklidischen Norm

$$\kappa_2(\boldsymbol{A}) := \frac{\sigma_1}{\sigma_n}.$$ □

Bemerkung 5.15 Im Prinzip kann man mit Hilfe von (5.7) die Singulärwertzerlegung von $\boldsymbol{A}$ berechnen (wenn man Eigenwertaufgaben numerisch lösen kann). Dazu hat man $\boldsymbol{A}^T\boldsymbol{A}$ und $\boldsymbol{A}\boldsymbol{A}^T$ zu berechnen. Dies ist zum einen sehr aufwendig, zum anderen kann — wie wir später sehen werden — die Kondition drastisch verschlechtert und damit die numerische Behandlung erheblich erschwert werden.

In der Praxis verwendet man einen Algorithmus von Golub und Reinsch (1971), der auf dem sogenannten QR Algorithmus zur Bestimmung der Eigenwerte von $\boldsymbol{A}^T\boldsymbol{A}$ beruht, aber die Berechnung von $\boldsymbol{A}^T\boldsymbol{A}$ bzw. $\boldsymbol{A}\boldsymbol{A}^T$ vermeidet. □

5.4 Pseudoinverse

Wir betrachten erneut das lineare Ausgleichsproblem

> *Es seien* $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ *und* $\boldsymbol{b} \in \mathbb{R}^m$ *gegeben mit* $m \geqslant n$. *Man bestimme* $\boldsymbol{x} \in \mathbb{R}^n$ *mit*
>
> $$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min \tag{5.8}$$

und untersuchen dieses nun mit Hilfe der Singulärwertzerlegung.

Wir bezeichnen in diesem ganzen Abschnitt mit $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$ die singulären Werte von $\boldsymbol{A}$, mit $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ eine Singulärwertzerlegung von $\boldsymbol{A}$ und mit $\boldsymbol{u}^j$ bzw. $\boldsymbol{v}^k$ die Spalten von $\boldsymbol{U}$ bzw. $\boldsymbol{V}$.

Satz 5.16 *Es sei* $\boldsymbol{c} := \boldsymbol{U}^T\boldsymbol{b} \in \mathbb{R}^m$. *Dann ist die Lösungsmenge des linearen Ausgleichsproblems* (5.8) *gegeben durch*

$$L = \bar{\boldsymbol{x}} + \mathit{Kern}(\boldsymbol{A}), \tag{5.9}$$

wobei $\bar{\boldsymbol{x}}$ *die folgende spezielle Lösung von* (5.8) *bezeichnet:*

$$\bar{\boldsymbol{x}} := \sum_{i=1}^{r} \frac{c_i}{\sigma_i}\boldsymbol{v}^i. \tag{5.10}$$

Beweis: Da die Multiplikation mit einer orthogonalen Matrix die Euklidische Länge nicht ändert, gilt mit $\boldsymbol{z} := \boldsymbol{V}^T\boldsymbol{x}$

$$\begin{aligned}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 &= \|\boldsymbol{U}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2^2 = \|\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} - \boldsymbol{U}^T\boldsymbol{b}\|_2^2 \\ &= \|\boldsymbol{\Sigma}\boldsymbol{z} - \boldsymbol{c}\|_2^2 = \|(\sigma_1 z_1 - c_1, \ldots, \sigma_r z_r - c_r, -c_{r+1}, \ldots, -c_m)^T\|_2^2.\end{aligned}$$

Wie in Abschnitt 5.1 liest man hieraus die Lösung von (5.8) sofort ab: $z_i := \frac{c_i}{\sigma_i}$, $i = 1, \ldots, r$, und $z_i \in \mathbb{R}$ beliebig für $i = r+1, \ldots, n$, d.h.

$$\boldsymbol{x} = \sum_{i=1}^{r} \frac{c_i}{\sigma_i} \boldsymbol{v}^i + \sum_{i=r+1}^{n} z_i \boldsymbol{v}^i, z_i \in \mathbb{R}, i = r+1, \ldots, n. \tag{5.11}$$

Da die letzten $n - r$ Spalten von $\boldsymbol{V}$ nach Satz 5.13 den Kern von $\boldsymbol{A}$ aufspannen, kann man die Lösungsmenge L von (5.8) schreiben als (5.9), (5.10). ■

In Satz 5.1 wurde gezeigt (und das liest man auch aus Satz 5.16 ab), dass die Lösung des Ausgleichsproblems (5.8) genau dann eindeutig ist, wenn $r = \text{Rang}\boldsymbol{A} = n$ gilt. Wir erzwingen nun auch im Fall $r < n$ die Eindeutigkeit, indem wir zusätzlich fordern, dass die Euklidische Norm der Lösung möglichst klein werden soll.

Definition 5.17 Es sei L die Lösungsmenge des Ausgleichsproblems (5.8). $\tilde{\boldsymbol{x}} \in L$ heißt **Pseudonormallösung** von (5.8), falls

$$\|\tilde{\boldsymbol{x}}\|_2 \leqslant \|\boldsymbol{x}\|_2 \quad \text{für alle } \boldsymbol{x} \in L.$$

Aus der Darstellung (5.11) der allgemeinen Lösung von (5.8) liest man ab, dass $\bar{\boldsymbol{x}}$ aus (5.10) Pseudonormallösung von (5.8) ist, denn

$$\Big\|\bar{\boldsymbol{x}} + \sum_{i=r+1}^{n} z_i \boldsymbol{v}^i\Big\|_2^2 = \|\bar{\boldsymbol{x}}\|_2^2 + \sum_{i=r+1}^{n} |z_i|^2 \cdot \|\boldsymbol{v}^i\|_2^2 \geqslant \|\bar{\boldsymbol{x}}\|_2^2.$$

Ferner ist die Pseudonormallösung eindeutig bestimmt, und $\bar{\boldsymbol{x}}$ ist offensichtlich die einzige Lösung von (5.8) mit $\boldsymbol{x} \in \text{Kern}(\boldsymbol{A})^\perp \cap L$. Daher gilt

Satz 5.18 *Es gibt genau eine Pseudonormallösung $\bar{\boldsymbol{x}}$ von (5.8). Diese ist charakterisiert durch $\bar{\boldsymbol{x}} \in \text{Kern}(\boldsymbol{A})^\perp \cap L$.*

Für jedes $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ ist durch

$$\mathbb{R}^m \ni \boldsymbol{b} \mapsto \bar{\boldsymbol{x}} \in \mathbb{R}^n \ : \ \|\boldsymbol{A}\bar{\boldsymbol{x}} - \boldsymbol{b}\|_2 \leqslant \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \forall \boldsymbol{x} \in \mathbb{R}^n, \ \|\bar{\boldsymbol{x}}\|_2 \text{ minimal}$$

eine Abbildung erklärt. Diese ist offensichtlich linear (vgl. die Darstellung von $\bar{\boldsymbol{x}}$ in (5.10)), kann also durch eine Matrix $\boldsymbol{A}^\dagger \in \mathbb{R}^{(n,m)}$ dargestellt werden.

Definition 5.19 Es sei $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$. Dann heißt die Matrix $\boldsymbol{A}^\dagger \in \mathbb{R}^{(n,m)}$, für die durch $\bar{\boldsymbol{x}} := \boldsymbol{A}^\dagger \boldsymbol{b}$ für alle $\boldsymbol{b} \in \mathbb{R}^m$ die Pseudonormallösung des Ausgleichsproblems (5.8) gegeben ist, die **Pseudoinverse** (oder **Moore-Penrose Inverse**) von $\boldsymbol{A}$.

Bemerkung 5.20 Ist $\text{Rang}\boldsymbol{A} = n$, so ist das Ausgleichsproblem (5.8) eindeutig lösbar, und aus den Normalgleichungen folgt, dass die Lösung $\bar{\boldsymbol{x}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$ ist. In diesem Fall ist also $\boldsymbol{A}^\dagger = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$. Ist noch spezieller $n = m$ und $\boldsymbol{A}$ regulär, so ist $\boldsymbol{A}^\dagger = \boldsymbol{A}^{-1}$. Die Pseudo-Inverse wird also zur „normalen Inversen“, wenn diese existiert, und ist somit eine konsistente Erweiterung dieses Begriffes. □

Aus unserer Konstruktion ergibt sich sofort im allgemeinen Fall

Satz 5.21 *Sei $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ mit der Singulärwertzerlegung*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \quad \boldsymbol{\Sigma} = (\sigma_i \delta_{ij})_{i,j}.$$

Dann gilt

1. $\boldsymbol{\Sigma}^\dagger = (\tau_i \delta_{ij})_{j,i},\ \tau_i = \begin{cases} \sigma_i^{-1}, & \text{falls } \sigma_i \neq 0 \\ 0, & \text{falls } \sigma_i = 0 \end{cases}$,
2. $\boldsymbol{A}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger \boldsymbol{U}^T$.

Bemerkung 5.22 Die explizite Matrix-Darstellung der Pseudoinverse braucht man genauso häufig wie die der inversen Matrix, nämlich fast nie. □

Aus der Darstellung der Pseudoinversen in Satz 5.212 folgt unmittelbar

Korollar 5.23 *Für jede Matrix* $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ *gilt*

$$\boldsymbol{A}^{\dagger\dagger} = \boldsymbol{A}$$

und

$$(\boldsymbol{A}^\dagger)^T = (\boldsymbol{A}^T)^\dagger.$$

$\boldsymbol{A}^\dagger$ besitzt also die üblichen Eigenschaften der Inversen $\boldsymbol{A}^{-1}$ für reguläres $\boldsymbol{A}$. Es gilt jedoch i.a.

$$(\boldsymbol{A}\boldsymbol{B})^\dagger \neq \boldsymbol{B}^\dagger \boldsymbol{A}^\dagger.$$

Beispiel 5.24 Es gilt

$$\boldsymbol{A} = \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix} = \boldsymbol{I} \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix},$$

und daher besitzt $\boldsymbol{A}$ die Pseudoinverse

$$\boldsymbol{A}^\dagger = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 0 \end{pmatrix} \boldsymbol{I} = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}.$$

Es ist $\boldsymbol{A}^2 = \boldsymbol{A}$ und $(\boldsymbol{A}^\dagger)^2 = \frac{1}{2}\boldsymbol{A}^\dagger$, d.h. $(\boldsymbol{A}^2)^\dagger \neq (\boldsymbol{A}^\dagger)^2$. □

5.5 Störung von Ausgleichsproblemen

Wir übertragen nun den Begriff der Kondition einer Matrix auf singuläre und allgemeiner auf nicht quadratische Matrizen. Es ist klar, dass für den quadratischen, nicht regulären Fall dieser verallgemeinerte Konditionsbegriff nicht mehr als Verstärkungsfaktor für die Störung linearer Systeme gedeutet werden kann. Er spielt aber eine ähnliche Rolle für lineare Ausgleichsprobleme.

Wir beschränken uns auf die Euklidische Norm. Zur Motivation betrachten wir das lineare Ausgleichsproblem

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min \tag{5.12}$$

mit $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$, $\mathrm{Rang}(\boldsymbol{A}) = r$, und eine Störung hiervon

$$\|\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) - (\boldsymbol{b} + \Delta\boldsymbol{b})\|_2 = \min, \tag{5.13}$$

wobei wir zunächst nur Störungen von $\boldsymbol{b}$, nicht aber der Koeffizientenmatrix $\boldsymbol{A}$ zulassen.

Es sei $\boldsymbol{x} = \boldsymbol{A}^\dagger \boldsymbol{b}$ bzw. $\boldsymbol{x} + \Delta\boldsymbol{x} = \boldsymbol{A}^\dagger(\boldsymbol{b} + \Delta\boldsymbol{b})$ die Pseudonormallösung von (5.12) bzw. (5.13). Dann gilt $\Delta\boldsymbol{x} = \boldsymbol{A}^\dagger \Delta\boldsymbol{b}$, und aus $\|\boldsymbol{A}^\dagger\|_2 = \dfrac{1}{\sigma_r}$ folgt

$$\|\Delta\boldsymbol{x}\|_2 \leqslant \|\boldsymbol{A}^\dagger\|_2 \cdot \|\Delta\boldsymbol{b}\|_2 = \frac{1}{\sigma_r}\|\Delta\boldsymbol{b}\|_2.$$

Ferner gilt (vgl. (5.11)):

$$\|\boldsymbol{x}\|_2^2 = \sum_{i=1}^{r} \frac{c_i^2}{\sigma_i^2} \geqslant \frac{1}{\sigma_1^2} \sum_{i=1}^{r} c_i^2 = \frac{1}{\sigma_1^2} \| \sum_{i=1}^{r} (\boldsymbol{u}^i)^T \boldsymbol{b} \boldsymbol{u}^i \|_2^2.$$

Da offenbar $\sum_{i=1}^{r} (\boldsymbol{u}^i)^T \boldsymbol{b} \boldsymbol{u}^i$ die Projektion von $\boldsymbol{b}$ auf den Bildbereich Bild($\boldsymbol{A}$) von $\boldsymbol{A}$ ist, folgt also für den relativen Fehler

$$\frac{\|\Delta \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \leqslant \frac{\sigma_1}{\sigma_r} \cdot \frac{\|\Delta \boldsymbol{b}\|_2}{\|\boldsymbol{P}_{\text{Bild}(\mathbf{A})} \boldsymbol{b}\|_2}. \tag{5.14}$$

Diese Ungleichung beschreibt wieder, wie sich der relative Fehler der rechten Seite eines Ausgleichsproblems auf die Lösung auswirken kann. Wir definieren daher

Definition 5.25 $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$ besitze die Singulärwertzerlegung $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Dann nennen wir $\kappa_2(\boldsymbol{A}) := \dfrac{\sigma_1}{\sigma_r}$ die **Kondition** der Pseudonormallösung des Ausgleichproblems bzgl. Störungen der rechten Seite.

Bemerkung 5.26 Ist $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ regulär, so stimmt diese Definition der Kondition mit der vorher gegebenen (für die Euklidische Norm) überein. □

Bemerkung 5.27 Wegen $\kappa_2(\boldsymbol{A}^T\boldsymbol{A}) = \kappa_2(\boldsymbol{A})^2$ und $\kappa_2(\boldsymbol{A}) \geqslant 1$ sind die Normalgleichungen eines linearen Ausgleichsproblems im Allgemeinen schlechter konditioniert als die Koeffizientenmatrix des Ausgleichsproblems. □

Lässt man auch Störungen der Koeffizientenmatrix $\boldsymbol{A}$ zu, so erhält man (vgl. Demmel [17] p. 117)

Satz 5.28 *Die Matrix $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$, $m \geqslant n$, besitze den vollen Rang n. Es sei $\boldsymbol{x}$ die Lösung des Ausgleichsproblems (5.12) und $\tilde{\boldsymbol{x}}$ die Lösung des gestörten Problems*

$$\|(\boldsymbol{A} + \Delta\boldsymbol{A})\boldsymbol{x} - (\boldsymbol{b} + \Delta\boldsymbol{b})\|_2 = \min, \tag{5.15}$$

wobei

$$\varepsilon := \max\left(\frac{\|\Delta\boldsymbol{A}\|_2}{\|\boldsymbol{A}\|_2}, \frac{\|\Delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}\right) < \frac{1}{\kappa_2(\boldsymbol{A})} = \frac{\sigma_n(\boldsymbol{A})}{\sigma_1(\boldsymbol{A})}. \tag{5.16}$$

Dann gilt

$$\frac{\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2}{\|\boldsymbol{x}\|_2} \leqslant \varepsilon \left(\frac{2\kappa_2(\boldsymbol{A})}{\cos\theta} + \tan\theta \cdot \kappa_2^2(\boldsymbol{A})\right) + O(\varepsilon^2), \tag{5.17}$$

wobei θ den Winkel zwischen $\boldsymbol{b}$ und seiner Projektion auf den Raum Bild($\boldsymbol{A}$) bezeichnet.

5.6 Regularisierung schlecht konditionierter Probleme

Einige Probleme der Anwendungen (z.B. in der Tomographie oder bei der Ausbreitung von Rissen in den Materialwissenschaften) führen auf lineare Gleichungssysteme oder Ausgleichsprobleme mit schlecht konditionierten Koeffizientenmatrizen. In diesen Fällen führen die bisher betrachteten Verfahren zu schlechten oder gar unbrauchbaren Ergebnissen.

Beispiel 5.29 Das Problem, die orthogonale Projektion einer gegebenen Funktion $f : [0,1] \to \mathbb{R}$ auf den Raum Π_{n-1} der Polynome vom Höchstgrad $n-1$ bzgl. des inneren Produkts

$$\langle f, g\rangle := \int_0^1 f(x)g(x)\,dx$$

zu berechnen, führt bei der Wahl der Basis $\{1, x, \ldots, x^{n-1}\}$ auf das lineare Gleichungssystem

$$\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b} \tag{5.18}$$

mit der Matrix

$$\boldsymbol{A} = (a_{ij})_{i,j=1,\ldots,n},\ a_{ij} := \frac{1}{i+j-1}, \tag{5.19}$$

der sogenannten **Hilbert Matrix**, und $\boldsymbol{b} \in \mathbb{R}^n$, $b_i := \langle f, x^{i-1}\rangle$.

Wir wählen für die Dimensionen $n = 10$, $n = 20$ und $n = 40$ die rechte Seite von (5.18) so, dass $\boldsymbol{y} = (1, \ldots, 1)^T$ die eindeutige Lösung ist, und behandeln (5.18) mit den bekannten numerischen Verfahren. Unter MATLAB erhält man mit der LR-Zerlegung mit Spaltenpivotsuche (in MATLAB $\boldsymbol{A}\backslash\boldsymbol{b}$), mit dem Cholesky Verfahren, der QR Zerlegung der Matrix $\boldsymbol{A}$ und der Singulärwertzerlegung von $\boldsymbol{A}$ die folgenden Fehler in der Euklidischen Norm:

	$n = 10$	$n = 20$	$n = 40$
LR Zerlegung	5.24 E-4	8.25 E+1	3.78 E+2
Cholesky	7.15 E-4	numer. nicht pos. def.	
QR Zerlegung	1.41 E-3	1.67 E+2	1.46 E+3
SVD	8.24 E-4	3.26 E+2	8.35 E+2

Die Ergebnisse sind also (wenigstens für die Dimensionen $n = 20$ oder $n = 40$) unbrauchbar.

Ein ähnliches Verhalten zeigt sich bei dem Ausgleichsproblem. Wir betrachten für $n = 10$, $n = 20$ und $n = 40$ und $m = n + 10$ die Ausgleichsprobleme

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \min$$

mit der Hilbert Matrix $\boldsymbol{A} \in \mathbb{R}^{(m,n)}$, wobei $\boldsymbol{b}$ so gewählt ist, dass $\boldsymbol{x} = (1, \ldots, 1)^T$ die Lösung ist mit dem Residuum $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$. Die folgende Tabelle enthält wieder die Fehler in der Euklidischen Norm für die Lösung der Normalgleichungen mit der LR-Zerlegung (das Cholesky Verfahren führte schon bei $n = 10$ zu der Meldung, dass die Koeffizientenmatrix nicht positiv definit ist), mit der QR Zerlegung und der Singulärwertzerlegung. Die Lösungen sind ebenfalls unbrauchbar.

	$n = 10$	$n = 20$	$n = 40$
Normalgleichungen	2.91 E+2	2.40 E+2	8.21 E+2
QR Zerlegung	1.93 E-5	5.04 E+0	1.08 E+1
SVD	4.67 E-5	6.41 E+1	3.72 E+2

□

Bei schlecht konditionierten Ausgleichsproblemen oder Gleichungssystemen ($n = m$) ist das folgende Vorgehen zu empfehlen:

Bestimme die Singulärwertzerlegung $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ von $\boldsymbol{A}$; setze

$$\boldsymbol{\Sigma}_\tau^\dagger = diag(\eta_i \delta_{ji}), \quad \eta_i := \begin{cases} \sigma_i^{-1} & \text{falls } \sigma_i \geqslant \tau, \\ 0 & \text{sonst,} \end{cases}$$

wobei $\tau > 0$ eine gegebene Zahl ist,

$$\boldsymbol{A}_\tau^\dagger := \boldsymbol{V}\boldsymbol{\Sigma}_\tau^\dagger \boldsymbol{U}^T, \quad \boldsymbol{x} := \boldsymbol{A}_\tau^\dagger \boldsymbol{b}.$$

$\boldsymbol{A}_\tau^\dagger$ heißt **effektive Pseudoinverse** von $\boldsymbol{A}$.

Dieses Voergehen nennt man eine **Regularisierung**. Zu kleine singuläre Werte werden unschädlich gemacht, um die Kondition zu verbessern. Dafür nimmt man einen Verfahrensfehler in Kauf. Man löst an Stelle des Gleichungssystems $\boldsymbol{Ax} = \boldsymbol{b}$ das System $\boldsymbol{Ax} = \boldsymbol{Pb}$, wobei $\boldsymbol{P}$ die orthogonale Projektion auf den Teilraum $\operatorname{span}\{\boldsymbol{u}^i : \sigma_i \geqslant \tau\}$ bezeichnet.

Die bekannteste Regularisierung wurde unabhängig von Philips [57] und Tichonov [74] eingeführt und wird als **Tichonov Regularisierung** bezeichnet. Sie entspricht einer Dämpfung des Einflusses kleiner singulärer Werte bei der Lösung. Man löst an Stelle des Systems $\boldsymbol{Ax} = \boldsymbol{b}$ das Gleichungssystem

$$(\boldsymbol{A}^T\boldsymbol{A} + \alpha\boldsymbol{I}_n)\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{5.20}$$

mit einem Regularisierungsparameter $\alpha > 0$.

Offenbar ist (5.20) äquivalent zu

$$\|\boldsymbol{Ax} - \boldsymbol{b}\|^2 + \alpha\|\boldsymbol{x}\|^2 = \min \tag{5.21}$$

(dies ist die übliche Formulierung der Tichonov Regularisierung) oder zu

$$\|\tilde{\boldsymbol{A}}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|^2 = \min, \quad \tilde{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{A} \\ \sqrt{\alpha}\boldsymbol{I}_n \end{pmatrix}, \tilde{\boldsymbol{b}} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix}. \tag{5.22}$$

Diese letzte Formulierung wurde zusammen mit der QR Zerlegung der Matrix $\tilde{\boldsymbol{A}}$ von Golub [31] erstmals verwendet, um die Regularisierung stabil auszuführen.

Wegen $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} = \boldsymbol{A}^T\boldsymbol{A} + \alpha\boldsymbol{I}_n$ besitzt $\tilde{\boldsymbol{A}}$ die singulären Werte $\sqrt{\sigma_i^2 + \alpha}$, wenn die σ_i die singulären Werte von $\boldsymbol{A}$ sind, und die Kondition von (5.22) wird verkleinert zu $\sqrt{\dfrac{\sigma_1^2 + \alpha}{\sigma_n^2 + \alpha}}$.

Ist $\boldsymbol{\beta} := \boldsymbol{U}^T\boldsymbol{b}$, so ist (5.22) wegen (5.20) äquivalent zu

$$\boldsymbol{V}(\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma} + \alpha\boldsymbol{I}_n)\boldsymbol{V}^T\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\beta},$$

d.h.

$$\boldsymbol{x} = \boldsymbol{V}(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma} + \alpha\boldsymbol{I}_n)^{-1}\boldsymbol{\Sigma}^T\boldsymbol{\beta} = \sum_{i=1}^{n} \frac{\beta_i\sigma_i}{\sigma_i^2 + \alpha}\boldsymbol{v}^i.$$

Man benötigt also nur die Singulärwertzerlegung von $\boldsymbol{A}$, um das regularisierte Problem für verschiedene Regularisierungsparameter α zu lösen.

Wir wenden auf das Gleichungssystem (5.18) die Regularisierungen durch Abschneiden der kleinen singulären Werte an und die verschiedenen Formen der Tichonov Regularisierungen. Dabei wird der Regularisierungsparameter α jeweils so gewählt, dass der Fehler minimal wird. Dies ist bei praktischen Problemen natürlich nicht möglich (die Lösung des Problems wird ja erst noch gesucht). Strategien zur Wahl des Parameters findet man in Engl [24] oder Louis [48]. Als grobe Richtlinie kann man sagen, dass man eine numerische Lösung, die bedingt durch die schlechte Kondition eines Problems verfälscht ist, häufig daran erkennt, das sie stark oszilliert. Man variiert dann interaktiv den Parameter solange, bis man die Lösung glaubt (d.h. bis die gefundene Lösung die physikalischen Eigenschaften des modellierten Problems richtig wiedergibt).

Für das lineare Gleichungssystem (5.18), (5.19) erhält man die Fehler der folgenden Tabelle. Dabei wurden die Normalgleichungen der Tichonov Regularisierung (5.20) mit der Cholesky Zerlegung gelöst (die LR-Zerlegung mit dem MATLAB Befehl \ lieferte ähnliche Ergebnisse) und das regularisierte Ausgleichsproblem (5.22) wurde mit der QR Zerlegung von $\tilde{\boldsymbol{A}}$ und der Singulärwertzerlegung von $\boldsymbol{A}$ gelöst.

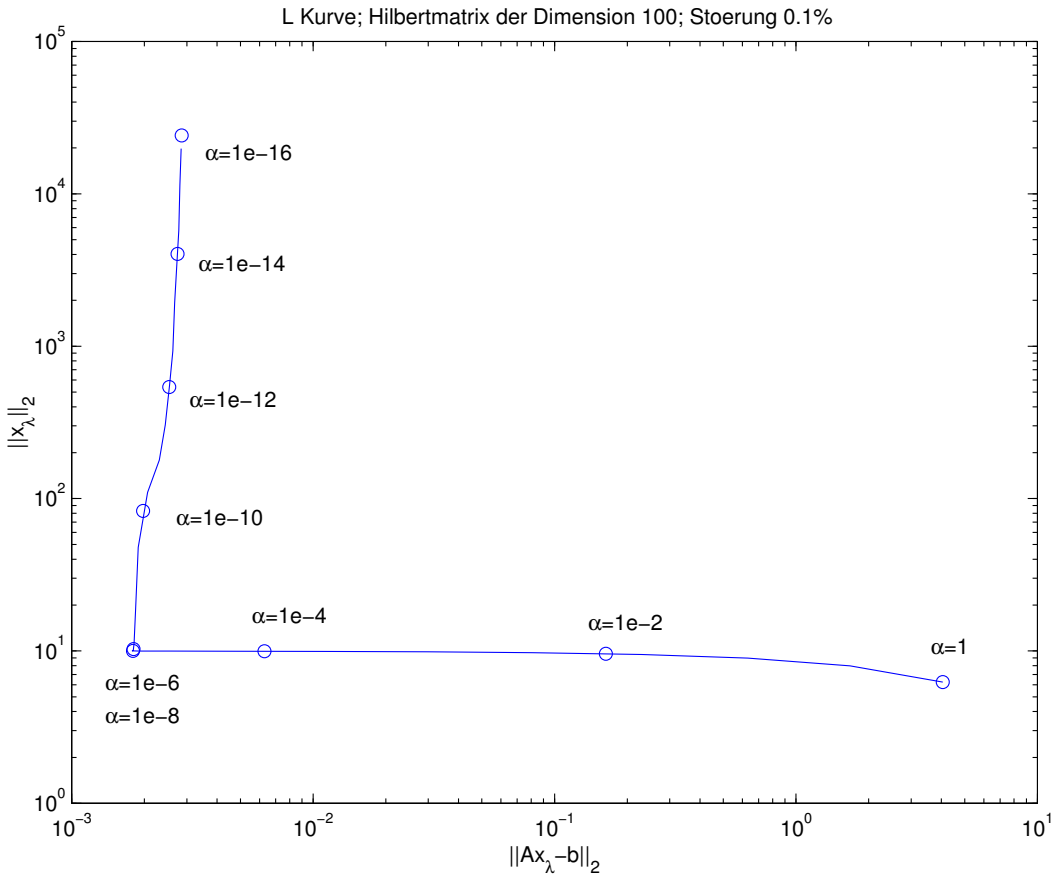

Abbildung 5.1: L-Kurve

	$n = 10$	$n = 20$	$n = 40$
Tichonov Cholesky	1.41 E-3	2.03 E-3	3.51 E-3
Tichonov QR	3.50 E-6	5.99 E-6	7.54 E-6
Tichonov SVD	3.43 E-6	6.33 E-6	9.66 E-6
Abgeschnittene SVD	2.77 E-6	3.92 E-6	7.35 E-6

Für das Ausgleichsproblem erhält man

	$n = 10$	$n = 20$	$n = 40$
Tichonov Cholesky	3.85 E-4	1.19 E-3	2.27 E-3
Tichonov QR	2.24 E-7	1.79 E-6	6.24 E-6
Tichonov SVD	8.51 E-7	1.61 E-6	3.45 E-6
Abgeschnittene SVD	7.21 E-7	1.94 E-6	7.70 E-6

Ein häufig verwendetes Verfahren zur Schätzung des optimalen Regularisierungsparameters ist die **L-Kurven Methode** [36], [37]. In ihr betrachtet man die Kurve

$$\alpha \mapsto (\|\boldsymbol{A}\boldsymbol{x}_\alpha - \boldsymbol{b}\|_2, \|\boldsymbol{x}_\alpha\|_2).$$

Diese sog. **L-Kurve** hat häufig die Gestalt des Buchstaben L wie in Abbildung 5.1, wobei der Parameter α, der zu dem „Knick“ gehört, optimal ist.

Für Systeme, für die Koeffizienten $\boldsymbol{b}^T\boldsymbol{u}^j$ der (ungestörten) rechten Seite bzgl. der singulären Vektoren $\boldsymbol{u}^j$ von $\boldsymbol{A}$ schneller abfallen als die singulären Werte σ_j von

$\boldsymbol{A}$, kann man die L-Kurven Methode begründen (vgl. Hansen [36]). Im allgemeinen Fall kann diese Methode aber versagen.

Abbildung 5.1 enthält die L-Kurve für das Gleichungssystem mit der Hilbertmatrix der Dimension 100 und der Lösung $\bar{\boldsymbol{x}} = (1, \ldots, 1)^T$, wobei zusätzlich die rechte Seite mit einem zufälligen Fehler von höchstens 0.1% verfälscht worden ist (Für dieses Beispiel ist die im letzten Absatz genannte Bedingung übrigens nicht erfüllt). Die folgende Tabelle enthält $\|\boldsymbol{A}\boldsymbol{x}_\alpha - \boldsymbol{b}\|_2$, $\|\boldsymbol{x}_\alpha\|_2$ und den Fehler $\|\boldsymbol{x}_\alpha - \bar{\boldsymbol{x}}\|_2$ für verschiedene Parameter α. Tatsächlich ist der Fehler minimal für $\alpha = 1e - 06$, und bei diesem Parameter liegt auch der Knick der L-Kurve.

α	$\|\boldsymbol{A}\boldsymbol{x}_\alpha - \boldsymbol{b}\|_2$	$\|\boldsymbol{x}_\alpha\|_2$	$\|\boldsymbol{x}_\alpha - \bar{\boldsymbol{x}}\|_2$
1e+00	4.0508e+00	6.2357e+00	5.3199e+00
1e-01	8.7802e-01	8.6987e+00	2.9868e+00
1e-02	1.6321e-01	9.5915e+00	1.6413e+00
1e-03	3.0354e-02	9.8697e+00	9.0701e-01
1e-04	6.2919e-03	9.9553e+00	5.2049e-01
1e-05	2.2522e-03	9.9819e+00	2.5714e-01
1e-06	1.7941e-03	9.9944e+00	1.0069e-01
1e-07	1.7862e-03	1.0017e+01	4.9295e-01
1e-08	1.8052e-03	1.0287e+01	2.3157e+00
1e-09	1.8355e-03	1.7458e+01	1.4260e+01
1e-10	1.9748e-03	8.3028e+01	8.2396e+01
1e-11	2.3529e-03	2.0179e+02	2.0153e+02
1e-12	2.5359e-03	5.3920e+02	5.3911e+02
1e-13	2.6525e-03	1.4070e+03	1.4070e+03
1e-14	2.7449e-03	4.0380e+03	4.0380e+03
1e-15	2.8050e-03	1.0709e+04	1.0709e+04
1e-16	2.8535e-03	2.4176e+04	2.4176e+04

Kapitel 6

Eigenwertaufgaben

6.1 Vorbetrachtungen

Da Eigenwerte und Eigenvektoren reeller Matrizen komplex sein können, betrachten wir gleich für $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ das **spezielle Eigenwertproblem**

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}. \tag{6.1}$$

Besitzt (6.1) eine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}$, so heißt λ ein **Eigenwert** von $\boldsymbol{A}$ und $\boldsymbol{x}$ ein zugehöriger **Eigenvektor**. Die Menge aller Eigenwerte einer Matrix $\boldsymbol{A}$ heißt das **Spektrum** von $\boldsymbol{A}$ und wird mit $\sigma(\boldsymbol{A})$ bezeichnet.

Genauer heißt $\boldsymbol{x} \neq \boldsymbol{0}$ mit (6.1) ein **Rechtseigenvektor** von $\boldsymbol{A}$ zu λ. Ist λ ein Eigenwert von $\boldsymbol{A}$, so besitzt auch die Gleichung

$$\boldsymbol{y}^H(\boldsymbol{A} - \lambda \boldsymbol{I}) = \boldsymbol{0} \tag{6.2}$$

nichttriviale Lösungen. Jedes $\boldsymbol{y} \in \mathbb{C}^n$, das die Gleichung (6.2) erfüllt, heißt ein **Linkseigenvektor** von $\boldsymbol{A}$. Die Notation ist hier nicht einheitlich. In einigen Büchern werden die nichttrivialen Lösungen von $\boldsymbol{y}^T(\boldsymbol{A} - \lambda \boldsymbol{I}) = \boldsymbol{0}$ die Linkseigenvektoren von $\boldsymbol{A}$ genannt. Wenn wir nur von Eigenvektoren sprechen, so sind stets Rechtseigenvektoren gemeint.

Die Eigenwerte von $\boldsymbol{A}$ sind die Nullstellen des **charakteristischen Polynoms**

$$\chi(\lambda) := \det(\boldsymbol{A} - \lambda \boldsymbol{I}).$$

Ist $\tilde{\lambda}$ ein k-fache Nullstelle von χ (ist also das Polynom $\chi(\lambda)$ durch $(\lambda - \tilde{\lambda})^k$ aber nicht durch $(\lambda - \tilde{\lambda})^{k+1}$ teilbar), so heißt k die **algebraische Vielfachheit** von $\tilde{\lambda}$. Da χ den Grad n besitzt, besitzt also $\boldsymbol{A}$ genau n Eigenwerte, wenn man jeden Eigenwert entsprechend seiner algebraischen Vielfachheit zählt.

Neben der algebraischen Vielfachheit betrachtet man die **geometrische Vielfachheit** eines Eigenwerts $\tilde{\lambda}$. Dies ist die Dimension des Lösungsraums des linearen Gleichungssystems

$$(\boldsymbol{A} - \tilde{\lambda}\boldsymbol{I})\boldsymbol{x} = \boldsymbol{0}.$$

Nach LA Satz 8.18 ist für jeden Eigenwert die geometrische Vielfachheit kleiner oder gleich der algebraischen Vielfachheit.

Gilt $\boldsymbol{B} = \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}$ mit einer regulären Matrix $\boldsymbol{X} \in \mathbb{C}^{(n,n)}$, so heißen die Matrizen $\boldsymbol{A}$ und $\boldsymbol{B}$ **ähnlich**, den Übergang von $\boldsymbol{A}$ zu $\boldsymbol{B}$ nennt man eine **Ähnlichkeitstransformation**. Ähnliche Matrizen besitzen (vgl. LA Satz 8.15) dieselben Eigenwerte einschließlich ihrer algebraischen und geometrischen Vielfachheit.

Naheliegend ist es nun, die Matrix $\boldsymbol{A}$, deren Eigenwerte bestimmt werden sollen, durch geeignete Ähnlichkeitstransformationen auf eine Gestalt zu bringen, aus der

man die Eigenwerte (oder jedenfalls Näherungen hierfür) von $\boldsymbol{A}$ leicht ablesen kann oder für die das Eigenwertproblem leichter gelöst werden kann.

Da folgende Lemma ermöglicht es, die Dimension des Eigenwertproblems zu reduzieren.

Lemma 6.1 *Besitzt* $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ *eine Blockzerlegung*

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{O} & \boldsymbol{A}_{22} \end{pmatrix}$$

mit $\boldsymbol{A}_{11} \in \mathbb{C}^{(m,m)}$, $\boldsymbol{A}_{22} \in \mathbb{C}^{(n-m,n-m)}$, *so gilt*

$$\sigma(\boldsymbol{A}) = \sigma(\boldsymbol{A}_{11}) \cup \sigma(\boldsymbol{A}_{22}). \tag{6.3}$$

Beweis: Es gelte

$$\boldsymbol{A}\boldsymbol{x} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{O} & \boldsymbol{A}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = \lambda \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix}$$

mit $\boldsymbol{x}_1 \in \mathbb{C}^m$ und $\boldsymbol{x}_2 \in \mathbb{C}^{n-m}$. Gilt $\boldsymbol{x}_2 \neq \boldsymbol{0}$, so ist $\boldsymbol{A}_{22}\boldsymbol{x}_2 = \lambda \boldsymbol{x}_2$ und $\lambda \in \sigma(\boldsymbol{A}_{22})$. Ist $\boldsymbol{x}_2 = \boldsymbol{0}$, so gilt $\boldsymbol{A}_{11}\boldsymbol{x}_1 = \lambda \boldsymbol{x}_1$ und $\lambda \in \sigma(\boldsymbol{A}_{11})$. Es ist also

$$\sigma(\boldsymbol{A}) \subset \sigma(\boldsymbol{A}_{11}) \cup \sigma(\boldsymbol{A}_{22}).$$

Zählt man alle Eigenwerte entsprechend ihrer algebraischen Vielfachheit, so besitzt $\boldsymbol{A}$ n Eigenwerte, $\boldsymbol{A}_{11}$ m Eigenwerte und $\boldsymbol{A}_{22}$ $n-m$ Eigenwerte. Damit kann die Inklusion nicht echt sein, und es ist (6.3) bewiesen. ■

Ist $\boldsymbol{J}$ die **Jordansche Normalform** der Matrix $\boldsymbol{A}$ und $\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{J}$, so kann man (vgl. LA Abschnitt 8.5) alle Eigenwerte und Eigenvektoren (und auch Hauptvektoren) von $\boldsymbol{A}$ aus $\boldsymbol{J}$ und der Transformationsmatrix $\boldsymbol{X}$ sofort ablesen. Trotzdem wird die Jordansche Normalform in der numerischen Mathematik nicht verwendet. Den Grund zeigt

Beispiel 6.2 Die Störung

$$\boldsymbol{J}_\alpha := \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \alpha & 0 & 0 & \dots & 0 \end{pmatrix} \in \mathbb{R}^{(n,n)}$$

der Jordanschen Normalform $\boldsymbol{J}_0$ mit dem n-fachen Eigenwert 0 besitzt das charakteristische Polynom

$$\chi_\alpha(\lambda) = (-1)^n(\lambda^n - \alpha).$$

$\boldsymbol{J}_\alpha$ besitzt also n verschiedene Nullstellen, und ist daher auf Diagonalgestalt transformierbar. Dies zeigt, dass die Struktur der Jordanschen Normalform nicht stabil unter kleinen Störungen ist. □

Wir betrachten ferner nicht beliebige Ähnlichkeitstransformationen, denn die nichtsinguläre Transformationsmatrix $\boldsymbol{X}$ kann schlecht konditioniert sein. In diesem Fall ist die Transformation $\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}$ anfällig für Rundungsfehler. Um dies auszuschließen, beschränken wir uns auf unitäre Transformationsmatrizen $\boldsymbol{U}$, für die also gilt $\boldsymbol{U}^H\boldsymbol{U} = \boldsymbol{I}$. Für diese ist

$$\|\boldsymbol{U}\boldsymbol{x}\|_2^2 = (\boldsymbol{U}\boldsymbol{x})^H(\boldsymbol{U}\boldsymbol{x}) = \boldsymbol{x}^H\boldsymbol{U}^H\boldsymbol{U}\boldsymbol{x} = \boldsymbol{x}^H\boldsymbol{x} = \|\boldsymbol{x}\|_2^2$$

für alle $\boldsymbol{x} \in \mathbb{C}^n$, daher gilt $\|\boldsymbol{U}\|_2 = 1$, und da mit $\boldsymbol{U}$ auch $\boldsymbol{U}^{-1} = \boldsymbol{U}^H$ unitär ist, erhält man $\kappa_2(\boldsymbol{U}) = 1$.

Wir wissen (vgl. LA Satz 8.31), dass man genau die **normalen** Matrizen mit unitären Matrizen auf Diagonalgestalt transformieren kann. Allgemeine Matrizen kann man auf obere Dreiecksgestalt transformieren, aus der man dann wieder die Eigenwerte unmittelbar ablesen kann.

Satz 6.3 (Schursche Normalform) *Es sei $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$. Dann existiert eine unitäre Matrix $\boldsymbol{U}$, so dass*

$$\boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U} = \boldsymbol{T} \tag{6.4}$$

*obere Dreiecksgestalt besitzt. $\boldsymbol{T}$ heißt **Schursche Normalform** der Matrix $\boldsymbol{A}$.*

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 6.1. ■

Bemerkung 6.4 Besitzt $\boldsymbol{A}$ die Schur Zerlegung

$$\boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U} = \operatorname{diag}\{\lambda_1, \ldots, \lambda_n\} + \boldsymbol{N}$$

mit einer strikten oberen Dreiecksmatrix $\boldsymbol{N}$, so gilt unabhängig von der Wahl von $\boldsymbol{U}$ für die Schur Norm

$$\|\boldsymbol{N}\|_S^2 = \|\boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U}\|_S^2 - \sum_{j=1}^{n} |\lambda_j|^2 = \|\boldsymbol{A}\|_S^2 - \sum_{j=1}^{n} |\lambda_j|^2.$$

Es gilt $\|\boldsymbol{N}\|_S^2 = 0$ genau dann, wenn $\boldsymbol{A}$ eine normale Matrix ist. $\|\boldsymbol{N}\|_S =: \Delta(\boldsymbol{A})$ heißt die **Abweichung** von $\boldsymbol{A}$ **von der Normalität**. □

Bei der bisherigen Behandlung der Ähnlichkeitstransformationen haben wir einen wichtigen Aspekt nicht berücksichtigt. Ist die Ausgangsmatrix $\boldsymbol{A}$ reell, so wird man nur orthogonale Matrizen $\boldsymbol{U}$ (d.h. reelle unitäre Matrizen) zur Transformation benutzen, da sonst $\boldsymbol{U}^H \boldsymbol{A} \boldsymbol{U}$ komplexe Elemente enthält. Besitzt $\boldsymbol{A}$ komplexe Eigenwerte, so kann $\boldsymbol{A}$ hiermit offenbar nicht auf obere Dreiecksgestalt transformiert werden (die Diagonale enthält ja die Eigenwerte). Da mit $\lambda \in \mathbb{C} \setminus \mathbb{R}$ auch $\bar{\lambda}$ Eigenwert von $\boldsymbol{A}$ ist (das charakteristische Polynom hat reelle Koeffizienten!), kann man $\boldsymbol{A}$ auf **Quasidreiecksgestalt** transformieren. Dabei heißt eine Matrix $\boldsymbol{R}$ Quasidreiecksmatrix, falls

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} & \boldsymbol{R}_{13} & \ldots & \boldsymbol{R}_{1k} \\ \boldsymbol{O} & \boldsymbol{R}_{22} & \boldsymbol{R}_{23} & \ldots & \boldsymbol{R}_{2k} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \ldots & \boldsymbol{R}_{kk} \end{pmatrix}$$

obere Blockdreiecksgestalt besitzt und die Diagonalblöcke $\boldsymbol{R}_{jj}$ alle die Dimension 1 oder 2 besitzen.

Satz 6.5 (Reelle Schursche Normalform) *Sei $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$. Dann existiert eine orthogonale Matrix $\boldsymbol{U}$, so dass $\boldsymbol{U}^T \boldsymbol{A} \boldsymbol{U}$ quasidreieckig ist. $\boldsymbol{U}$ kann so gewählt werden, dass jeder 2×2–Diagonalblock $\boldsymbol{R}_{jj}$ nur komplexe Eigenwerte besitzt.*

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 6.1. ■

Viele Verfahren zur Bestimmung von Eigenwerten bestehen darin, Matrizen $\boldsymbol{X}_k$ zu bestimmen, so dass $\boldsymbol{X}_k^{-1} \boldsymbol{A} \boldsymbol{X}_k$ mehr und mehr einer Diagonalmatrix gleicht. Wir fragen daher, wie gut die Eigenwerte einer Matrix durch ihre Diagonalelemente approximiert werden.

Satz 6.6 (Gerschgorin) *Sei*

$$\boldsymbol{A} := (a_{ij}) \in \mathbb{C}^{(n,n)}, \quad z_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \quad s_j := \sum_{\substack{i=1 \\ i\neq j}}^{n} |a_{ij}|.$$

Dann gilt

1. *Alle Eigenwerte von $\boldsymbol{A}$ liegen in der Vereinigung der Kreise*

$$Z_i := \{z \in \mathbb{C} : |z - a_{ii}| \leqslant z_i\}$$

2. *Alle Eigenwerte von $\boldsymbol{A}$ liegen in der Vereinigung der Kreise*

$$S_j := \{z \in \mathbb{C} : |z - a_{jj}| \leqslant s_j\}$$

3. *Jede Zusammenhangskomponente von $\bigcup_{i=1}^{n} Z_i$ bzw. $\bigcup_{j=1}^{n} S_j$ enthält genau so viele Eigenwerte von $\boldsymbol{A}$ wie Kreise an der Komponente beteiligt sind, wobei Kreise und Eigenwerte entsprechend ihrer (algebraischen) Vielfachheit gezählt sind.*

Beweis: Siehe dazu LA, Satz 8.84. ■

Bemerkung 6.7 Die Aussage (iii) lässt sich nicht verschärfen zu: In jedem Kreis S_i bzw. Z_j liegt mindestens ein Eigenwert von $\boldsymbol{A}$, denn sei $\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$. Dann sind die Eigenwerte $\lambda_{1/2} = \pm\sqrt{2}$, und es liegt kein Eigenwert in $Z_1 = S_2 = \{z : |z| \leqslant 1\}$. □

Für viele Eigenwert Routinen kann man zeigen, dass die berechneten Eigenwerte $\lambda_1, \ldots, \lambda_n$ die exakten Eigenwerte einer gestörten Matrix $\boldsymbol{A} + \boldsymbol{E}$ sind, wobei $\boldsymbol{E}$ eine kleine Norm besitzt. Wir fragen daher, wie Störungen die Eigenwerte einer Matrix beeinflussen. Beispielhaft hierfür ist:

Satz 6.8 (Bauer, Fike) *Ist μ Eigenwert von $\boldsymbol{A} + \boldsymbol{E} \in \mathbb{C}^{(n,n)}$ und*

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{\Lambda} = \mathit{diag}\{\lambda_1, \ldots, \lambda_n\},$$

so gilt

$$\min_{\lambda\in\sigma(A)} |\lambda - \mu| \leqslant \kappa_p(\boldsymbol{X})\|E\|_p.$$

Dabei bezeichnet $\|\cdot\|_p$ die von der Höldernorm

$$\|\boldsymbol{x}\|_p := \left(\sum_{j=1}^{n} |x_j|^p\right)^{1/p}, \quad p \geqslant 1,$$

induzierte Matrixnorm und κ_p die Kondition bzgl. $\|\cdot\|_p$.

Beweis: Für $\mu \in \sigma(\boldsymbol{A})$ ist die Aussage trivial. Sei also $\mu \notin \sigma(\boldsymbol{A})$.
Da $\boldsymbol{X}^{-1}(\boldsymbol{A} + \boldsymbol{E} - \mu\boldsymbol{I})\boldsymbol{X}$ singulär ist, ist auch

$$\boldsymbol{I} + (\boldsymbol{\Lambda} - \mu\boldsymbol{I})^{-1}(\boldsymbol{X}^{-1}\boldsymbol{E}\boldsymbol{X}) = (\boldsymbol{\Lambda} - \mu\boldsymbol{I})^{-1}\boldsymbol{X}^{-1}(\boldsymbol{A} + \boldsymbol{E} - \mu\boldsymbol{I})\boldsymbol{X}$$

singulär.

Also existiert $\boldsymbol{y} \in \mathbb{C}^n$, $\|\boldsymbol{y}\|_p = 1$ mit

$$\boldsymbol{y} + (\boldsymbol{\Lambda} - \mu \boldsymbol{I})^{-1}(\boldsymbol{X}^{-1}\boldsymbol{E}\boldsymbol{X})\boldsymbol{y} = \boldsymbol{0}$$

Hieraus folgt

$$\begin{aligned} 1 = \|\boldsymbol{y}\|_p &= \|(\boldsymbol{\Lambda} - \mu \boldsymbol{I})^{-1}(\boldsymbol{X}^{-1}\boldsymbol{E}\boldsymbol{X})\boldsymbol{y}\|_p \\ &\leqslant \max_{\|\boldsymbol{z}\|_p=1} \|(\boldsymbol{\Lambda} - \mu \boldsymbol{I})^{-1}(\boldsymbol{X}^{-1}\boldsymbol{E}\boldsymbol{X})\boldsymbol{z}\|_p \\ &= \|(\boldsymbol{\Lambda} - \mu \boldsymbol{I})^{-1}\boldsymbol{X}^{-1}\boldsymbol{E}\boldsymbol{X}\|_p \leqslant \frac{1}{\min\limits_{\lambda\in\sigma(A)} |\lambda - \mu|} \|\boldsymbol{X}^{-1}\|_p \|\boldsymbol{E}\|_p \|\boldsymbol{X}\|_p, \end{aligned}$$

wobei ausgenutzt wurde, dass die p–Norm einer Diagonalmatrix gleich dem Betrag des betragsmaximalen Elements der Matrix ist.

■

Für normale Matrizen erhält man insbesondere

Korollar 6.9 *Es sei $\boldsymbol{A}$ eine normale Matrix und μ ein Eigenwert von $\boldsymbol{A} + \boldsymbol{E} \in \mathbb{C}^{(n,n)}$. Dann gilt*

$$\min_{\lambda\in\sigma(A)} |\lambda - \mu| \leqslant \|\boldsymbol{E}\|_2$$

Beweis: Da $\boldsymbol{A}$ normal, kann man in Satz 6.8 $\boldsymbol{X}$ unitär wählen, und die Behauptung folgt dann aus $\kappa_2(\boldsymbol{X}) = 1$. ■

6.2 Potenzmethode

Wir betrachten in diesem Abschnitt ein Verfahren zur Bestimmung des betragsgrößten Eigenwerts und zugehörigen Eigenvektors und eine Modifikation hiervon, um auch andere Eigenwerte zu berechnen. Wir besprechen die Methoden vor allem zum besseren Verständnis des dann im nächsten Abschnitt folgenden Arbeitspferdes zur Berechnung von Eigenwerten und Eigenvektoren, des QR Algorithmus.

Wir betrachten für $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ die spezielle Eigenwertaufgabe

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}. \tag{6.5}$$

Es sei $\boldsymbol{A}$ diagonalisierbar, $\boldsymbol{A}\boldsymbol{x}^i = \lambda_i \boldsymbol{x}^i$, $i = 1, \ldots, n$, und es besitze $\boldsymbol{A}$ einen dominanten Eigenwert λ_1, d.h.

$$|\lambda_1| > |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|. \tag{6.6}$$

Es sei $\boldsymbol{u}^0 \in \mathbb{C}^n$, $\boldsymbol{u}^0 \neq \boldsymbol{0}$, gegeben. Multipliziert man

$$\boldsymbol{u}^0 =: \sum_{i=1}^{n} \alpha_i \boldsymbol{x}^i$$

m mal mit der Matrix $\boldsymbol{A}$, so folgt

$$\boldsymbol{A}^m \boldsymbol{u}^0 = \sum_{i=1}^{n} \alpha_i \lambda_i^m \boldsymbol{x}^i = \lambda_1^m \left\{ \alpha_1 \boldsymbol{x}^1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^m \boldsymbol{x}^i \right\}. \tag{6.7}$$

Wegen (6.6) konvergiert daher die Folge $\{\lambda_1^{-m} \boldsymbol{A}^m \boldsymbol{u}^0\}$ gegen ein Vielfaches von $\boldsymbol{x}^1$, falls $\alpha_1 \neq 0$ gilt, d.h. $\{\boldsymbol{A}^m \boldsymbol{u}^0\}$ konvergiert gegen die Eigenrichtung $\operatorname{span}\{\boldsymbol{x}^1\}$ zum

dominanten Eigenwert λ_1, wenn der Startvektor $\boldsymbol{u}^0$ eine Komponente in Richtung von $\boldsymbol{x}^1$ besitzt.

Gilt $|\lambda_1| \neq 1$, so erhält man für große m Exponentenüberlauf oder -unterlauf. Man wird also die Folge $\{\boldsymbol{A}^m\boldsymbol{u}^0\}$ noch geeignet normieren und erhält die **Potenzmethode** oder das **von Mises Verfahren**.

Sei $\|\cdot\|$ irgendeine Norm auf $\mathbb{C}^n$ und $\boldsymbol{u}^0 \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}$.

Algorithmus 6.10 (Potenzmethode)

```
for m=0,1,... until convergence do
   v_{m+1}=Au_m;
   k_{m+1}=‖v_{m+1}‖;
   u_{m+1}=v_{m+1}/k_{m+1};
end
```

Häufig wird eine andere Skalierung der erzeugten Folge gewählt, die billiger ist als die Normierung. Sei dazu $\boldsymbol{\ell} \in \mathbb{C}^n$ ein gegebener Vektor. Hiermit iterieren wir gemäß

Algorithmus 6.11 (Potenzmethode)

```
for m=0,1,... until convergence do
   v_{m+1}=Au_m;
   k_{m+1}=l'*v_{m+1};
   u_{m+1}=v_{m+1}/k_{m+1};
end
```

Satz 6.12 *Es sei λ_1 dominanter Eigenwert von $\boldsymbol{A}$ und*

$$\boldsymbol{u}^0 := \sum_{i=1}^{n} \alpha_i \boldsymbol{x}^i$$

mit $\alpha_1 \neq 0$ und $\boldsymbol{\ell} \in \mathbb{C}^n$.

Es seien $\boldsymbol{u}^m$ und k_m nach Algorithmus 6.11 berechnet. Gilt dann $\boldsymbol{\ell}^H\boldsymbol{u}^m \neq 0$ für alle $m \in \mathbb{N}$ und $\boldsymbol{\ell}^H\boldsymbol{x}^1 \neq 0$, so ist

$$\lim_{m\to\infty} k_m = \lambda_1, \quad \lim_{m\to\infty} \boldsymbol{u}^m = \frac{1}{\boldsymbol{\ell}^H\boldsymbol{x}^1}\boldsymbol{x}^1.$$

Beweis: Offensichtlich gilt

$$\boldsymbol{u}^m = \boldsymbol{A}^m\boldsymbol{u}^0 / \boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0.$$

Daher ist

$$k_m = \boldsymbol{\ell}^H\boldsymbol{v}^m = \boldsymbol{\ell}^H\boldsymbol{A}\boldsymbol{u}^{m-1} = \boldsymbol{\ell}^H\boldsymbol{A}\frac{\boldsymbol{A}^{m-1}\boldsymbol{u}^0}{\boldsymbol{\ell}^H\boldsymbol{A}^{m-1}\boldsymbol{u}^0} = \frac{\boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0}{\boldsymbol{\ell}^H\boldsymbol{A}^{m-1}\boldsymbol{u}^0},$$

und aus

$$\lim_{m\to\infty} \lambda_1^{-m}\boldsymbol{A}^m\boldsymbol{u}^0 = \alpha_1\boldsymbol{x}^1$$

folgt

$$\lim_{m\to\infty} \lambda_1^{-1}k_m = \frac{\lim\limits_{m\to\infty} \boldsymbol{\ell}^H(\lambda_1^{-m}\boldsymbol{A}^m\boldsymbol{u}^0)}{\lim\limits_{m\to\infty} \boldsymbol{\ell}^H(\lambda_1^{-(m-1)}\boldsymbol{A}^{m-1}\boldsymbol{u}^0)} = 1,$$

d.h.

$$\lim_{m\to\infty} k_m = \lambda_1$$

und

$$\lim_{m\to\infty} \boldsymbol{u}^m = \lim_{m\to\infty} \frac{\boldsymbol{A}^m\boldsymbol{u}^0}{\boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0} = \frac{\alpha_1\boldsymbol{x}^1 + \sum_{i=2}^{n}\alpha_i(\lambda_i/\lambda_1)^m\boldsymbol{x}^i}{\alpha_1\boldsymbol{\ell}^H\boldsymbol{x}^1 + \sum_{i=2}^{n}\alpha_1(\lambda_i/\lambda_1)^m\boldsymbol{\ell}^H\boldsymbol{x}^i} = \frac{\boldsymbol{x}^1}{\boldsymbol{\ell}^H\boldsymbol{x}^1}.$$

■

Bemerkung 6.13

1. Eine häufige Wahl von $\boldsymbol{\ell}$ ist $\boldsymbol{\ell} = \boldsymbol{e}^k$ für ein $k \in \{1,\ldots,n\}$, d.h. man normiert im Eigenvektor die $k-$te Komponente zu 1, wobei man nicht vorhersagen kann, welches k geeignet ist.

2. Ist $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ nichtnegativ und irreduzibel, so ist der zum Spektralradius gehörende Eigenvektor positiv, und man kann $\boldsymbol{\ell} = (1,1,\ldots,1)^T$ wählen, wenn $\boldsymbol{u}^0$ positive Komponenten hat.

3. Wir haben $\alpha_1 \neq 0$ vorausgesetzt. Diese Voraussetzung ist nicht einschneidend, denn durch Rundungsfehler wird (fast immer) ein $\boldsymbol{u}^m$ erzeugt, das eine Komponente in Richtung von $\boldsymbol{x}^1$ besitzt.

4. Ist λ_1 mehrfacher Eigenwert von $\boldsymbol{A}$ und $\boldsymbol{A}$ diagonalähnlich, so bleiben alle Überlegungen richtig, wenn nur $\boldsymbol{u}^0$ eine Komponente in Richtung des Eigenraumes von $\boldsymbol{A}$ zu λ_1 besitzt oder eine solche durch Rundungsfehler erzeugt wird.

5. Ist $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ mit $|\lambda_1| = |\lambda_2| > |\lambda_3| \geqslant \ldots \geqslant |\lambda_n|$ und $\lambda_1 \neq \lambda_2$ (also $\lambda_1 = \bar{\lambda}_2$ oder $\lambda_1 = -\lambda_2$), so erhält man keine Konvergenz, sondern es treten in der Folge $\{\boldsymbol{u}^m\}$ Oszillationen auf. Auch in diesem Fall kann man aus (drei aufeinanderfolgenden Elementen) der Folge $\{\boldsymbol{u}^m\}$ Näherungen für die zu λ_1 und λ_2 gehörenden Eigenvektoren ermitteln. Eine ausführliche Diskussion findet man in Zurmühl [83], pp. 283 ff.

6. Schließlich haben wir vorausgesetzt, dass $\boldsymbol{A}$ diagonalisierbar" ist. Ist dies nicht erfüllt, so muss man $\boldsymbol{u}^0$ als Linearkombination von Hauptvektoren darstellen und erhält ebenfalls Konvergenz des Verfahrens (vgl. Collatz [13], pp. 325 ff.).

7. Für die Konvergenzgeschwindigkeit gilt mit

$$q := \max_{i=2,\ldots,m}\left|\frac{\lambda_i}{\lambda_1}\right|$$

$$\begin{aligned}
|k_{m+1} - \lambda_1| &= \left|\frac{\boldsymbol{\ell}^H\boldsymbol{A}^{m+1}\boldsymbol{u}^0}{\boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0} - \lambda_1\right| = \left|\frac{1}{\boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0}\boldsymbol{\ell}^H(\boldsymbol{A}^{m+1}\boldsymbol{u}^0 - \lambda_1\boldsymbol{A}^m\boldsymbol{u}^0)\right| \\
&= \left|\frac{\lambda_1^{m+1}}{\boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0}\boldsymbol{\ell}^H\left(\sum_{i=2}^{n}\alpha_i\left(\left(\frac{\lambda_i}{\lambda_1}\right)^{m+1} - \left(\frac{\lambda_i}{\lambda_1}\right)^m\right)\boldsymbol{x}^i\right)\right| \\
&\leqslant \left|\frac{\lambda_1^{m+1}}{\boldsymbol{\ell}^H\boldsymbol{A}^m\boldsymbol{u}^0}\right| \|\boldsymbol{\ell}\|_2 \sum_{i=2}^{n}|\alpha_i|\left|\frac{\lambda_i}{\lambda_1}\right|^m\left|\frac{\lambda_i}{\lambda_1} - 1\right| \|\boldsymbol{x}^i\|_2 \leqslant Cq^m \qquad (6.8)
\end{aligned}$$

d.h. wir haben lineare Konvergenz mit dem Konvergenzfaktor q.

Ist also $|\lambda_1|$ von den Beträgen der übrigen Eigenwerte gut getrennt, so erhält man rasche Konvergenz, i.a. ist das Verfahren aber sehr langsam.

8. Bisweilen kann die Konvergenzgeschwindigkeit durch eine Spektralverschiebung verbessert werden, d.h. betrachte $\boldsymbol{A}+\alpha\boldsymbol{I}$, $\alpha\in\mathbb{C}$. Dann gehen die Eigenwerte über in $\lambda_i+\alpha$, und die Konvergenzgeschwindigkeit wird bestimmt durch
$$q := \max_{i=2,\dots,n}\left|\frac{\lambda_i+\alpha}{\lambda_1+\alpha}\right|.$$
Die Verbesserung hierdurch ist meistens unwesentlich.

Eine erhebliche Beschleunigung der von Mises Iteration erhält man durch die inverse Iteration, die 1944 von Wielandt [81] eingeführt wurde bei der Stabilitätsanalyse von Strukturen, die kleine Störungen bekannter Systeme sind. In diesem Fall sind gute Approximationen für die relevanten Eigenwerte bekannt, und man erhält (wie wir noch sehen werden) rasche Konvergenz. Heute wird die inverse Iteration vor allem verwendet zur Bestimmung von Eigenvektoren, wenn wenige Eigenwerte und Eigenvektoren gesucht sind.

Sei $\lambda\neq\lambda_i$ für alle i. Dann besitzt die Matrix $(\lambda\boldsymbol{I}-\boldsymbol{A})^{-1}$ die Eigenwerte $1/(\lambda-\lambda_i)$ und die Eigenvektoren stimmen mit denen von $\boldsymbol{A}$ überein.

Führt man die von Mises Iteration für $(\lambda\boldsymbol{I}-\boldsymbol{A})^{-1}$ aus und gilt $|\lambda-\lambda_i|<|\lambda-\lambda_j|$, $j=1,\dots,n$, $j\neq i$, mit einem einfachen Eigenwert λ_i von $\boldsymbol{A}$, so ist $\dfrac{1}{\lambda-\lambda_i}$ dominanter Eigenwert von $(\lambda\boldsymbol{I}-\boldsymbol{A})^{-1}$ und die Konvergenzgeschwindigkeit wird bestimmt durch
$$q := \max_{j\neq i}\left|\frac{\lambda-\lambda_i}{\lambda-\lambda_j}\right|.$$
Ist also λ eine gute Näherung für λ_i, so erhält man sehr schnelle Konvergenz.

Algorithmus 6.14 (Inverse Iteration; feste Shifts)
```
for m=0,1,... until convergence do
   L"ose (λI-A)v_{m+1}=u_m;
   k_{m+1}=l'*v_{m+1};
   u_{m+1}=v_{m+1}/k_{m+1};
end
```

Wie in Satz 6.12 erhält man im Fall $|\lambda-\lambda_i|<|\lambda-\lambda_j|$ für alle $j\neq i$
$$\boldsymbol{u}^m\to\frac{\boldsymbol{x}^i}{\boldsymbol{\ell}^H\boldsymbol{x}^i},\quad k_m\to\frac{1}{\lambda-\lambda_i}$$
unter den entsprechenden Voraussetzungen $\alpha_i\neq 0$, $\boldsymbol{\ell}^H\boldsymbol{v}^m\neq 0$, $\boldsymbol{\ell}^H\boldsymbol{x}^i\neq 0$.

Die Konvergenz ist natürlich linear (vgl. (6.8)). Dies wird verbessert, wenn man λ gleichzeitig iteriert:

Algorithmus 6.15 (Inverse Iteration; variable Shifts)
```
for m=0,1,... until convergence do
   L"ose (λ_m I-A)v_{m+1}=u_m;
   k_{m+1}=l'*v_{m+1};
   u_{m+1}=v_{m+1}/k_{m+1};
   λ_{m+1}=λ_m-1/k_{m+1};
end
```

Es gilt ja $k_{m+1}\to\dfrac{1}{\lambda-\lambda_i}$. Hieraus erhält man eine Schätzung $\lambda_{(m+1)}$ für λ_i durch
$$k_{m+1}\approx 1/(\lambda_{(m)}-\lambda_{(m+1)}),$$
und damit die Aufdatierung des Shift Parameters in Algorithmus 6.15.

Satz 6.16 *Sei $\tilde{\lambda}$ ein algebraisch einfacher Eigenwert von $\boldsymbol{A}$ mit zugehörigem Eigenvektor $\tilde{u}$, und es gelte $\boldsymbol{\ell}^H\tilde{u} = \boldsymbol{\ell}^H\boldsymbol{u}^0 = 1$. Dann konvergiert das Verfahren in Algorithmus 6.15 lokal quadratisch gegen $\tilde{\lambda}$ bzw. $\tilde{u}$.*

Beweis: Wir zeigen, dass das Verfahren genau die Iterationsvorschrift des Newton-Verfahrens für das Gleichungssystem

$$F(\boldsymbol{u},\lambda) := \begin{pmatrix} \lambda\boldsymbol{u} - \boldsymbol{A}\boldsymbol{u} \\ \boldsymbol{\ell}^H\boldsymbol{u} - 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ 0 \end{pmatrix}$$

ist und dass $F'(\tilde{u},\tilde{\lambda})$ nichtsingulär ist. Dann folgt die quadratische Konvergenz sowohl für den Eigenvektor als auch den Eigenwert aus einem Satz in Mathematik III. Wir werden auf das Newton Verfahren in Kapitel 7 eingehen.

Es gilt

$$F'(\boldsymbol{u},\lambda) = \begin{pmatrix} \lambda\boldsymbol{I} - \boldsymbol{A} & \boldsymbol{u} \\ \boldsymbol{\ell}^H & 0 \end{pmatrix}.$$

Das Newton-Verfahren ist also gegeben durch

$$\begin{pmatrix} \lambda_{(m)}\boldsymbol{I} - \boldsymbol{A} & \boldsymbol{u}^m \\ \boldsymbol{\ell}^H & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{m+1} - \boldsymbol{u}^m \\ \lambda_{(m+1)} - \lambda_{(m)} \end{pmatrix} = - \begin{pmatrix} \lambda_{(m)}\boldsymbol{u}^m - \boldsymbol{A}\boldsymbol{u}^m \\ \boldsymbol{\ell}^H\boldsymbol{u}^m - 1 \end{pmatrix}$$

d.h.

$$\begin{aligned} (\lambda_{(m)}\boldsymbol{I} - \boldsymbol{A})\boldsymbol{u}^{m+1} + \left(\lambda_{(m+1)} - \lambda_{(m)}\right)\boldsymbol{u}^m &= \boldsymbol{0} \\ \boldsymbol{\ell}^H\boldsymbol{u}^{m+1} &= 1, \end{aligned}$$

und dies ist äquivalent Algorithmus 6.15.

Es sei

$$F'(\tilde{u},\tilde{\lambda}) \begin{pmatrix} \boldsymbol{v} \\ \mu \end{pmatrix} = \boldsymbol{0}$$

d.h.

$$(\tilde{\lambda}\boldsymbol{I} - \boldsymbol{A})\boldsymbol{v} + \mu\tilde{u} = \boldsymbol{0}, \quad \boldsymbol{\ell}^H\boldsymbol{v} = 0.$$

Im Falle $\mu = 0$ gilt $(\tilde{\lambda}\boldsymbol{I} - \boldsymbol{A})\boldsymbol{v} = \boldsymbol{0}$, d.h., da $\tilde{\lambda}$ ein einfacher Eigenwert von $\boldsymbol{A}$ ist, $\boldsymbol{v} = \alpha\tilde{u}$ für ein $\alpha \in \mathbb{C}$, und es folgt $0 = \boldsymbol{\ell}^H\tilde{u} = \alpha$.

Im Falle $\mu \neq 0$ gilt

$$(\tilde{\lambda}\boldsymbol{I} - \boldsymbol{A})\boldsymbol{v} = -\mu\tilde{u} \neq \boldsymbol{0},$$

und daher

$$(\tilde{\lambda}\boldsymbol{I} - \boldsymbol{A})^2\boldsymbol{v} = -\mu(\tilde{\lambda}\boldsymbol{I} - \boldsymbol{A})\tilde{u} = \boldsymbol{0}.$$

$\boldsymbol{v}$ ist also ein Hauptvektor 2. Stufe, und daher besitzt $\tilde{\lambda}$ mindestens die algebraische Vielfachheit 2.

■

Bemerkung 6.17 Man entnimmt dem Beweis sofort, dass das Verfahren in Algorithmus 6.14 auch dann lokal konvergiert, wenn $\boldsymbol{A}$ nicht diagonalisierbar ist. □

Die inverse Iteration konvergiert sogar kubisch, wenn die Matrix $\boldsymbol{A}$ symmetrisch ist und wenn die Näherung für den Eigenwert aufdatiert wird mit dem Rayleighschen Quotienten

$$\lambda_{(m)} = \frac{(\boldsymbol{u}^m)^H\boldsymbol{A}\boldsymbol{u}^m}{\|\boldsymbol{u}^m\|_2^2}.$$

Diese Aufdatierung ist auch bei nicht symmetrischen Matrizen sinnvoll wegen des folgenden Lemmas. Man erhält dann wie in Satz 6.16 quadratische Konvergenz (s. Stewart [67], pp. 346 ff.).

Lemma 6.18 *Sei* $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ *beliebig,* $\boldsymbol{x} \in \mathbb{C}^n$*, und für* $\mu \in \mathbb{C}$

$$\boldsymbol{r}(\mu) := \boldsymbol{A}\boldsymbol{x} - \mu\boldsymbol{x}$$

der zugehörige Defekt. Dann gilt

$$\|\boldsymbol{r}(\mu)\|_2 = \min \Leftrightarrow \mu = \frac{\boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}}{\|\boldsymbol{x}\|_2^2}.$$

Beweis: $\|\boldsymbol{r}(\mu)\|_2 = \min$ ist ein lineares Ausgleichsproblem zur Bestimmung von μ mit der Koeffizientenmatrix $\boldsymbol{x}$ und der rechten Seite $\boldsymbol{A}\boldsymbol{x}$. Die Normalgleichung hierzu lautet

$$\boldsymbol{x}^H \boldsymbol{x}\mu = \boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}.$$

■

Auf den ersten Blick scheint es eine Schwierigkeit bei der inversen Iteration zu sein, dass die Koeffizientenmatrix $\lambda_{(m)}\boldsymbol{I} - \boldsymbol{A}$ des zu lösenden linearen Gleichungssystems bei Konvergenz von $\lambda_{(m)}$ gegen einen Eigenwert von $\boldsymbol{A}$ mehr und mehr singulär wird, die Kondition also wächst und damit der Fehler der Lösung des Gleichungssystems wächst. Man kann jedoch zeigen (vgl. Chatelin [12], pp. 217 ff.), dass (bei gut konditionierten Problemen) der Fehler, der bei der Lösung von $(\lambda_{(m)}\boldsymbol{I} - \boldsymbol{A})\boldsymbol{v}^{m+1} = \boldsymbol{u}^m$ gemacht wird, vornehmlich die Richtung des Eigenvektors zu λ, also die gewünschte Richtung, hat.

6.3 Der QR Algorithmus

Der QR Algorithmus dient dazu, alle Eigenwerte einer Matrix $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ zu bestimmen. Er wurde von Francis [25], [26] und Kublanovskaya [46] unabhängig voneinander im Jahr 1961 eingeführt. Er wird heute als das beste Verfahren zur Lösung der vollständigen Eigenwertaufgabe für nichtsymmetrische Probleme angesehen.

Es sei $\boldsymbol{A}_0 := \boldsymbol{A}$. Die Grundform des QR Algorithmus ist gegeben durch

Algorithmus 6.19 (QR Algorithmus; Grundform)

```
for i=0,1,... until convergence do
   Zerlege A_i=Q_iR_i;                (QR Zerlegung)
   A_{i+1}=R_iQ_i;
end
```

Die durch den QR Algorithmus erzeugten Matrizen $\boldsymbol{A}_i$ sind einander ähnlich und damit ähnlich zu $\boldsymbol{A}$, denn es gilt

$$\boldsymbol{A}_{i+1} = \boldsymbol{R}_i\boldsymbol{Q}_i = \boldsymbol{Q}_i^H(\boldsymbol{Q}_i\boldsymbol{R}_i)\boldsymbol{Q}_i = \boldsymbol{Q}_i^H\boldsymbol{A}_i\boldsymbol{Q}_i.$$

Die Konvergenz des QR Algorithmus wird durch den folgenden Satz beschrieben.

Satz 6.20 *Es seien die Eigenwerte von* $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ *dem Betrage nach voneinander verschieden und angeordnet gemäß*

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0. \tag{6.9}$$

Es sei $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1}$*, und es besitze die Matrix* $\boldsymbol{Y} := \boldsymbol{X}^{-1}$ *eine LR Zerlegung.*

Dann konvergiert der QR Algorithmus im Wesentlichen, d.h. für $\boldsymbol{A}_i := (a_{jk}^{(i)})_{j,k}$ *gilt*

$$\lim_{i\to\infty} a_{jk}^{(i)} = 0 \textit{für} j > k$$
$$\lim_{i\to\infty} a_{kk}^{(i)} = \lambda_k \textit{für} k = 1, \ldots, n.$$

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 6.4. ■

Bemerkung 6.21 Am Ende sind die Eigenwerte von $\boldsymbol{A}$ in $\boldsymbol{A}_i$ der Größe der Beträge nach geordnet. □

Beispiel 6.22 Die Matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & -1 & -1 \\ 4 & 6 & 3 \\ -4 & -4 & -1 \end{pmatrix}$$

besitzt die Eigenwerte $\lambda_1 = 3$, $\lambda_2 = 2$ und $\lambda_3 = 1$, und für die Matrix der Eigenvektoren gilt (bis auf die Normierung)

$$\boldsymbol{X} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 1 & 2 \\ -1 & 0 & -2 \end{pmatrix} \quad \text{und} \quad \boldsymbol{X}^{-1} = \begin{pmatrix} 2 & 2 & 1 \\ 0 & 1 & 1 \\ -1 & -1 & -1 \end{pmatrix}.$$

Die Voraussetzungen von Satz 6.20 sind also erfüllt. Man erhält

$$\boldsymbol{A}_{10} = \begin{pmatrix} 3.000034 & -0.577363 & 8.981447 \\ 9.78e-6 & 1.999995 & 1.4142593 \\ -6.91e-6 & 3.99e-6 & 0.999972 \end{pmatrix}$$ □

Bemerkung 6.23 Wegen (6.9) ist $\boldsymbol{A}$ diagonalisierbar, d.h. $\boldsymbol{X}$ und damit $\boldsymbol{Y}$ existieren. Die einzige Forderung in Satz 6.20 an die Matrix $\boldsymbol{X}$ ist also, dass $\boldsymbol{Y} := \boldsymbol{X}^{-1}$ eine LR Zerlegung besitzt. Verzichtet man hierauf, so erhält man

$$\lim_{i\to\infty} a_{kk}^{(i)} = \lambda_{\pi(k)}$$

für eine Permutation π von $\{1, \ldots, n\}$ (vgl. Wilkinson [82], p. 519 ff.). □

Beispiel 6.24 Für die Matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & -1 \\ -2 & -2 & 2 \end{pmatrix}$$

mit den Eigenwerten $\lambda_1 = 3$, $\lambda_2 = 2$, $\lambda_3 = 1$ und den Eigenvektoren

$$\boldsymbol{X} = \begin{pmatrix} -1 & -1 & -1 \\ 2 & 1 & 1 \\ -2 & -1 & 0 \end{pmatrix} \quad \text{und} \quad \boldsymbol{X}^{-1} = \begin{pmatrix} 1 & 1 & 0 \\ -2 & -2 & -1 \\ 0 & 1 & 1 \end{pmatrix}.$$

ist die Voraussetzung von Satz 6.20 nicht erfüllt. Die Matrix $\boldsymbol{X}^{-1}(1:2, 1:2)$ ist singulär. Der QR Algorithmus liefert nach 30 Schritten

$$\boldsymbol{A}_{30} = \begin{pmatrix} 3.0000058 & -2.0000014 & 2.9999975 \\ 1.16e-6 & 0.9999989 & -0.9999978 \\ -1.16e-6 & 6.69e-5 & 1.9999953 \end{pmatrix}.$$

Die Diagonalelemente scheinen also gegen die Eigenwerte zu konvergieren, wobei sie allerdings nicht nach der Betragsgöße geordnet sind. Diese Konvergenz würde

auch bei exakter Rechnung eintreten (vgl. Bemerkung 6.23). Iteriert man weiter, so erhält man (durch Rundungsfehler)

$$\boldsymbol{A}_{70} = \begin{pmatrix} 3 + 5.2e-13 & -3.5355303 & 0.7071247 \\ 1.48e-13 & 1.9999949 & -1.0000051 \\ -7.52e-19 & -5.07e-6 & 1.0000051 \end{pmatrix},$$

also doch noch Konvergenz gegen eine Dreiecksmatrix mit der Betragsgröße nach geordneten Eigenwerten. □

Bemerkung 6.25 Ist $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$, so gilt $\boldsymbol{Q}_k \in \mathbb{R}^{(n,n)}$, $\boldsymbol{R}_k \in \mathbb{R}^{(n,n)}$ für alle k, und der ganze Algorithmus verläuft in $\mathbb{R}$. Sind alle Eigenwerte betragsmäßig verschieden, so sind sie alle reell, und der QR–Algorithmus konvergiert. Besitzt $\boldsymbol{A}$ jedoch komplexe Eigenwerte, so kann die Grundform nicht konvergieren.

Ist $\boldsymbol{A} \in \mathbb{C}^{(n,n)}$ Hermitesch, so sind alle $\boldsymbol{A}_m$ Hermitesch, und $\boldsymbol{A}_m$ konvergiert unter den Voraussetzungen von Satz 6.20 gegen eine Diagonalmatrix. □

Ein Nachteil der Grundform des QR Algorithmus ist, dass sie sehr aufwendig ist. Ist $\boldsymbol{A}$ voll besetzt, so benötigt man in jedem Schritt $O(n^3)$ Operationen zur Bestimmung der QR Zerlegung.

Wir beschleunigen nun den QR Algorithmus auf zwei Weisen. Erstens erhöhen wir mit Hilfe von Shifts die Konvergenzgeschwindigkeit, und zweitens zeigen wir, dass die Hessenberg Gestalt einer Matrix im Laufe eines QR Schritts erhalten bleibt. Da die QR Zerlegung einer Hessenberg Matrix nur $O(n^2)$ Operationen benötigt, werden wir Matrizen vor Anwendung des QR Algorithmus zunächst unitär auf Hessenberg Gestalt transformieren.

Sei $\boldsymbol{A}_1 := \boldsymbol{A}$. Wir betrachten dann die geshiftete Version des QR Algorithmus:

Algorithmus 6.26 (QR Algorithmus mit Shifts)

```
for i=0,1,... until convergence do
   Waehle geeigneten Shift kappa_i;
   Zerlege A_i-kappa_i I=Q_iR_i;
   A_{i+1}=R_iQ_i+kappa_i I;
end
```

Für die hierdurch erzeugten Matrizen gilt

$$\boldsymbol{R}_i = \boldsymbol{Q}_i^H(\boldsymbol{A}_i - \kappa_i \boldsymbol{I}),$$

und

$$\boldsymbol{A}_{i+1} = \boldsymbol{Q}_i^H(\boldsymbol{A}_i - \kappa_i \boldsymbol{I})\boldsymbol{Q}_i + \kappa_i \boldsymbol{I} = \boldsymbol{Q}_i^H \boldsymbol{A}_i \boldsymbol{Q}_i. \tag{6.10}$$

Die $\boldsymbol{A}_i$ sind also alle der Matrix $\boldsymbol{A}$ ähnlich und haben dieselben Eigenwerte. Um die Parameter κ_i geeignet zu wählen, überlegen wir uns einen Zusammenhang von Algorithmus 6.26 und der inversen Iteration.

Satz 6.27 *Es seien*

$$\boldsymbol{U}_i := \boldsymbol{Q}_1\boldsymbol{Q}_2 \dots \boldsymbol{Q}_i, \quad \boldsymbol{S}_i := \boldsymbol{R}_i\boldsymbol{R}_{i-1} \dots \boldsymbol{R}_1.$$

Dann gilt

$$\boldsymbol{U}_i\boldsymbol{S}_i = (\boldsymbol{A} - \kappa_i \boldsymbol{I})(\boldsymbol{A} - \kappa_{i-1}\boldsymbol{I}) \dots (\boldsymbol{A} - \kappa_1 \boldsymbol{I}). \tag{6.11}$$

Beweis: Zunächst folgt aus (6.10) durch Induktion

$$\boldsymbol{A}_{i+1} = \boldsymbol{U}_i^H \boldsymbol{A} \boldsymbol{U}_i. \tag{6.12}$$

Für $i = 1$ besagt (6.11)

$$\boldsymbol{U}_1\boldsymbol{S}_1 = \boldsymbol{Q}_1\boldsymbol{R}_1 = \boldsymbol{A} - \kappa_1\boldsymbol{I},$$

und dies ist gerade die Zerlegung im ersten Schritt von Algorithmus 6.26.

Sei (6.11) bereits für $i-1$ bewiesen. Dann folgt aus der Definition von $\boldsymbol{A}_{i+1}$

$$\boldsymbol{R}_i = (\boldsymbol{A}_{i+1} - \kappa_i\boldsymbol{I})\boldsymbol{Q}_i^H = \boldsymbol{U}_i^H(\boldsymbol{A} - \kappa_i\boldsymbol{I})\boldsymbol{U}_i\boldsymbol{Q}_i^H = \boldsymbol{U}_i^H(\boldsymbol{A} - \kappa_i\boldsymbol{I})\boldsymbol{U}_{i-1},$$

und hieraus erhält man durch Rechtsmultiplikation mit $\boldsymbol{S}_{i-1}$ und Linksmultiplikation mit $\boldsymbol{U}_i$

$$\boldsymbol{U}_i\boldsymbol{S}_i = (\boldsymbol{A} - \kappa_i\boldsymbol{I})\boldsymbol{U}_{i-1}\boldsymbol{S}_{i-1} = (\boldsymbol{A} - \kappa_i\boldsymbol{I})(\boldsymbol{A} - \kappa_{i-1}\boldsymbol{I})\dots(\boldsymbol{A} - \kappa_1\boldsymbol{I})$$

nach Induktionsannahme. ■

Aus der Darstellung (6.11) folgt sofort

$$(\boldsymbol{A}^H - \bar{\kappa}_i\boldsymbol{I})^{-1}\dots(\boldsymbol{A}^H - \bar{\kappa}_1\boldsymbol{I})^{-1}\boldsymbol{e}^n = \boldsymbol{U}_i(\boldsymbol{S}_i^H)^{-1}\boldsymbol{e}^n.$$

Da mit $\boldsymbol{S}_i^H$ auch $(\boldsymbol{S}_i^H)^{-1}$ eine untere Dreiecksmatrix ist, gilt

$$\boldsymbol{U}_i(\boldsymbol{S}_i^H)^{-1}\boldsymbol{e}^n = \sigma_i\boldsymbol{U}_i\boldsymbol{e}^n$$

für ein $\sigma_i \in \mathbb{C}$, d.h. die letzte Spalte von $\boldsymbol{U}_i$ ist eine Näherung für einen Eigenvektor von $\boldsymbol{A}^H$, die man ausgehend von $\boldsymbol{e}^n$ mit der inversen Iteration mit den Shifts $\bar{\kappa}_1, \dots, \bar{\kappa}_i$ erhält.

Man kann also schnelle Konvergenz erwarten, wenn man $\bar{\kappa}_i$ als Rayleighquotienten von $\boldsymbol{A}^H$ an der Stelle $\boldsymbol{u}_n^{i-1}$, der letzten Spalte von $\boldsymbol{U}_{i-1}$, wählt. Es ist

$$\overline{\kappa}_i = (\boldsymbol{u}_n^{i-1})^H\boldsymbol{A}^H\boldsymbol{u}_n^{i-1} = \overline{(\boldsymbol{u}_n^{i-1})^H\boldsymbol{A}\boldsymbol{u}_n^{i-1}} \overset{(6.12)}{=} \overline{(\boldsymbol{e}^n)^T\boldsymbol{A}_i\boldsymbol{e}^n} =: \overline{a_{nn}^{(i)}},$$

d.h.

$$\kappa_i = a_{nn}^{(i)}.$$

Dieser Shift heißt Rayleigh Quotienten Shift.

Eine weitere Verbesserung des Verfahrens (Verkleinerung des Aufwandes) erhält man, wenn man $\boldsymbol{A}$ zunächst unitär auf obere Hessenberg Gestalt transformiert. Dabei sagt man, dass eine Matrix $\boldsymbol{A}$ **Hessenberg Gestalt** hat, wenn

$$a_{ij} = 0 \quad \text{für alle } i > j+1.$$

Diese Gestalt bleibt nämlich, wie wir noch sehen werden, während des gesamten QR Algorithmus erhalten. Wir bestimmen daher eine unitäre Matrix $\boldsymbol{U}$, so dass $\boldsymbol{U}^H\boldsymbol{A}\boldsymbol{U}$ obere Hessenberg Gestalt hat.

$\boldsymbol{U}$ können wir als Produkt von $n-2$ Spiegelungen der Gestalt $\boldsymbol{I}-2\boldsymbol{u}\boldsymbol{u}^H$ aufbauen: Sei

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & \boldsymbol{c}^T \\ \boldsymbol{b} & \boldsymbol{B} \end{pmatrix} \text{ mit } \boldsymbol{b} \neq \boldsymbol{0}.$$

Wir bestimmen dann $\boldsymbol{w} \in \mathbb{C}^{n-1}$ mit $\|\boldsymbol{w}\|_2 = 1$ und

$$\boldsymbol{Q}_1\boldsymbol{b} := (\boldsymbol{I}_{n-1} - 2\boldsymbol{w}\boldsymbol{w}^H)\boldsymbol{b} = k\boldsymbol{e}^1,$$

und hiermit $\boldsymbol{P}_1 = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{Q}_1 \end{pmatrix}$.

Dann gilt

$$\boldsymbol{A}_1 := \boldsymbol{P}_1 \boldsymbol{A} \boldsymbol{P}_1 = \left(\begin{array}{c|c} a_{11} & \boldsymbol{c}^T \boldsymbol{Q}_1 \\ \hline k & \\ 0 & \\ \vdots & \boldsymbol{Q}_1 \boldsymbol{B} \boldsymbol{Q}_1 \\ 0 & \end{array}\right).$$

Damit hat die erste Spalte schon die Gestalt, die wir in einer Hessenberg Matrix benötigen. Die Spalten $2, \ldots, n-2$ können wir dann genauso behandeln (vgl. die Vorgehensweise bei der QR Zerlegung einer Matrix).

Sei $\boldsymbol{A} =: \boldsymbol{A}_1$ obere Hessenberg Matrix und $\kappa_1 \in \mathbb{C}$ ein Shift-Parameter, z.B. $\kappa_1 := a_{nn}^{(1)}$. Wir multiplizieren dann für $i = 1, \ldots, n-1$ die Matrix

$$\boldsymbol{U}_{i-1,i} \ldots \boldsymbol{U}_{12} (\boldsymbol{A} - \kappa_1 \boldsymbol{I})$$

mit einer ebenen Drehung $\boldsymbol{U}_{i,i+1}$, so dass das Element an der Stelle $(i+1, i)$ annulliert wird. Dann besitzt

$$\boldsymbol{R}_1 := \boldsymbol{U}_{n-1,n} \ldots \boldsymbol{U}_{12} (\boldsymbol{A} - \kappa_1 \boldsymbol{I})$$

obere Dreiecksgestalt, und $\boldsymbol{Q}_1 := \boldsymbol{U}_{12}^H \ldots \boldsymbol{U}_{n-1,n}^H$ ist der unitäre Anteil in der QR Zerlegung von $\boldsymbol{A} - \kappa_1 \boldsymbol{I}$ in Algorithmus 6.26. Die neue Iterierte $\boldsymbol{A}_2$ erhält man dann gemäß

$$\boldsymbol{A}_2 = \boldsymbol{R}_1 \boldsymbol{U}_{12}^H \ldots \boldsymbol{U}_{n-1,n}^H + \kappa_1 \boldsymbol{I}.$$

Da die Multiplikation von rechts mit $\boldsymbol{U}_{i,i+1}^H$ nur die i-te und $(i+1)$-te Spalte verändert, ist klar, dass $\boldsymbol{A}_2$ wieder obere Hessenberg Matrix ist. Die obere Hessenberg Gestalt der Matrizen bleibt also im Verlaufe des QR Algorithmus erhalten. Da ein QR Schritt für eine Hessenberg Matrix nur $O(n^2)$ Operationen erfordert, lohnt sich diese vorbereitende Transformation von $\boldsymbol{A}$.

Ist $\boldsymbol{A}$ Hermitesch, so sind alle $\boldsymbol{A}_i$ Hermitesch (vgl. (6.12)). Transformiert man also $\boldsymbol{A}$ zunächst auf Tridiagonalgestalt, so bleiben alle $\boldsymbol{A}_m$ wie eben tridiagonal, und man benötigt sogar in jedem QR Schritt nur $O(n)$ Operationen.

Wendet man den QR Algorithmus mit Shift $\kappa = a_{nn}$ wiederholt an, so wird $a_{n,n-1}^{(i)}$ rasch (quadratisch; vgl. Lemma 6.18) gegen 0 gehen. Man erhält eine Matrix der Gestalt

$$\boldsymbol{A}_i = \left(\begin{array}{cccccc} + & + & \ldots & + & + & * \\ + & + & \ldots & + & + & * \\ 0 & + & \ldots & + & + & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & \ldots & + & + & * \\ \hline 0 & 0 & \ldots & 0 & \approx 0 & * \end{array}\right) = \boldsymbol{U}_i^H \boldsymbol{A} \boldsymbol{U}_i.$$

$a_{nn}^{(i)}$ ist also eine Näherung für einen Eigenwert von $\boldsymbol{A}$ (nicht notwendig wie in Satz 6.20 für den betragskleinsten Eigenwert, da die Shifts diese Eigenschaft zerstören), und die Eigenwerte der führenden $(n-1, n-1)$ Hauptuntermatrix (oben $+$) von $\boldsymbol{A}_i$ sind Näherungen für die übrigen Eigenwerte von $\boldsymbol{A}$. Man kann also mit einer verkleinerten Matrix den Algorithmus fortsetzen.

Darüber hinaus gehen auch die übrigen Subdiagonalelemente von $\boldsymbol{A}_i$ gegen 0 (vgl. Satz 6.20). Ist eines dieser Elemente klein genug, etwa $a_{k+1,k}^{(i)} \approx 0$, so kann

man offenbar zwei kleinere Hessenberg Matrizen

$$\begin{pmatrix} a_{11}^{(i)} & a_{12}^{(i)} & \dots & a_{1,k-1}^{(i)} & a_{1k}^{(i)} \\ a_{21}^{(i)} & a_{22}^{(i)} & \dots & a_{2,k-1}^{(i)} & a_{2k}^{(i)} \\ 0 & a_{32}^{(i)} & \dots & a_{3,k-1}^{(i)} & a_{3k}^{(i)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a_{k,k-1}^{(i)} & a_{kk}^{(i)} \end{pmatrix}, \quad \begin{pmatrix} a_{k+1,k+1}^{(i)} & a_{k+1,k+2}^{(i)} & \dots & a_{k+1,n-1}^{(i)} & a_{k+1,n}^{(i)} \\ a_{k+2,k+1}^{(i)} & a_{k+2,k+2}^{(i)} & \dots & a_{k+2,n-1}^{(i)} & a_{k+2,n}^{(i)} \\ 0 & a_{k+3,k+2}^{(i)} & \dots & a_{k+3,n-1}^{(i)} & a_{k+3,n}^{(i)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a_{n,n-1}^{(i)} & a_{nn}^{(i)} \end{pmatrix}$$

mit dem QR Algorithmus behandeln.

Als Kriterium für klein genug wählt man mit $\varepsilon = 10^{-t}$, wobei t die Zahl der signifikanten Stellen bezeichnet,

$$|a_{k+1,k}^{(i)}| \leqslant \varepsilon \min\{|a_{kk}^{(i)}|, |a_{k+1,k+1}^{(i)}|\}$$

oder (weniger einschneidend)

$$|a_{k+1,k}^{(i)}| \leqslant \varepsilon(|a_{kk}^{(i)}| + |a_{k+1,k+1}^{(i)}|).$$

Beispiel 6.28 Wir demonstrieren den Konvergenzverlauf für die Hessenberg Matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 1 & 0 \\ 0 & 3 & 2 & 1 \\ 0 & 0 & 2 & 4 \end{pmatrix} \in \mathbb{R}^{(4,4)}.$$

Die folgende Tabelle enthält die Subdiagonalelemente bei dem oben beschriebenen Abbruch mit $t = 16$:

i	a_{21}	a_{32}	a_{43}
1	2	3	2
2	$1.83e0$	$-2.23e0$	$1.61e0$
3	$1.65e0$	$1.43e0$	$3.55e-1$
4	$6.78e-1$	$-3.65e0$	$3.82e-2$
5	$7.63e-1$	$1.17e0$	$-1.63e-3$
6	$8.32e-1$	$-5.32e-1$	$3.93e-6$
7	$-1.18e0$	$1.52e-1$	$-3.03e-11$
8	$1.64e0$	$-4.97e-2$	$1.62e-21$
9	$-3.22e0$	$2.89e-5$	
10	$1.81e0$	$5.33e-10$	
11	$-5.32e-1$	$-2.48e-19$	
12	$-4.46e-3$		
13	$5.89e-8$		
14	$-1.06e-17$		

Nach einer Anlaufphase wird die Zahl der gültigen Stellen des letzten berücksichtigten Diagonalelements von Schritt zu Schritt ungefähr verdoppelt. Dies zeigt die quadratische Konvergenz des Verfahrens.

Für die Grundform des QR Algorithmus wird diese Genauigkeit erst nach ca. 300 Schritten erreicht. □

Kapitel 7

Numerische Lösung nichtlinearer Gleichungssysteme

In den vorhergehenden Abschnitten haben wir lineare Probleme betrachtet. Wir untersuchen nun die numerische Bestimmung der Lösungen von nichtlinearen Gleichungen oder Gleichungssystemen. Es sei $f : \mathbb{R}^n \supset D \to \mathbb{R}^n$ gegeben. Wir betrachten das **Nullstellenproblem**

$$f(\boldsymbol{x}) = \mathbf{0}. \tag{7.1}$$

In Abschnitt 7.1 betrachten wir (7.1) in der Gestalt eines äquivalenten **Fixpunktproblems**

$$\phi(\boldsymbol{x}) = \boldsymbol{x} \tag{7.2}$$

und erinnern an den Fixpunktsatz für kontrahierende Abbildungen.

7.1 Fixpunktsatz für kontrahierende Abbildungen

Wir betrachten das Fixpunktproblem

$$\phi(\boldsymbol{x}) = \boldsymbol{x}, \tag{7.3}$$

mit einer gegebenen Funktion $\phi : \mathbb{R}^n \supset D \to \mathbb{R}^n$. Dieses ist äquivalent dem Nullstellenproblem (7.1), wenn wir z.B. $\phi(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{A}f(\boldsymbol{x})$ mit einer regulären Matrix $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ wählen.

Obwohl die meisten Aussagen auch in Banachräumen oder vollständigen metrischen Räumen richtig sind, beschränken wir uns auf den $\mathbb{R}^n$. Es bezeichne $\|\cdot\|$ irgendeine Vektornorm im $\mathbb{R}^n$ oder eine passende Matrixnorm.

Für das Problem (7.3) liegt es nahe, eine Näherung $\boldsymbol{x}^0$ für eine Lösung zu wählen und ausgehend hiervon die Folge

$$\boldsymbol{x}^{m+1} = \phi(\boldsymbol{x}^m) \tag{7.4}$$

iterativ zu bestimmen.

Beispiel 7.1 Wir betrachten das reelle Fixpunktproblem

$$x = 0.2\exp(x). \tag{7.5}$$

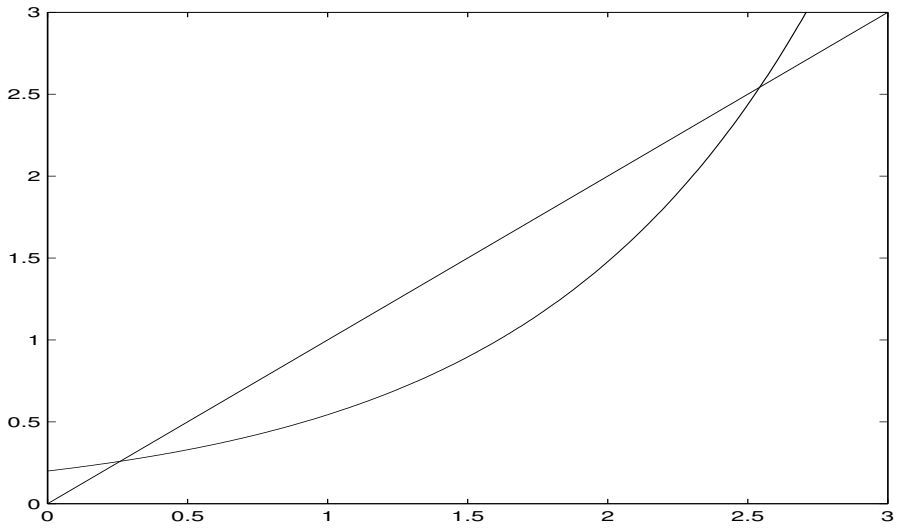

Abbildung 7.1: Zu Beispiel 7.1

m	x_m	x_m	x_m
0	0.000000	2.542600	2.542700
1	0.200000	2.542536	2.542790
2	0.244281	2.542374	2.543021
3	0.255340	2.541962	2.543606
4	0.258180	2.540914	2.545094
5	0.258914	2.538252	2.548886
⋮	⋮	⋮	⋮
10	0.259171	2.138065	3.378282
11	0.259171	1.696602	5.864073

Tabelle 7.1: Fixpunktiteration; zu Beispiel 7.1

Der Graph von $\phi(x) := 0.2\exp(x)$ ist in Abbildung 7.1 dargestellt, und man liest ab, dass ϕ (wenigstens) zwei Fixpunkte besitzt, einen in der Nähe von 0.25 und einen in der Nähe von 2.5 (und wenn man genauer zeichnen würde, würde man erkennen, dass der zweite Fixpunkt in der Nähe von 2.5426 liegt).

Tabelle 7.1 enthält die Ergebnisse der Iteration (7.4) für die Startwerte $x_0 = 0$, $x_0 = 2.5426$ und $x_0 = 2.5427$.

Dieses Beispiel vermittelt den Eindruck, dass nicht alle Fixpunkte einer Funktion ϕ mit der Iteration (7.4) erreichbar sind, auch dann nicht, wenn man sehr nahe bei dem Fixpunkt startet. □

Eine hinreichende Bedingung dafür, dass die Iteration (7.4) gegen einen Fixpunkt von ϕ konvergiert, liefert der Fixpunktsatz für kontrahierende Abbildungen. Aus ihm folgt zugleich die Existenz eines Fixpunktes.

Definition 7.2 $\phi : D \to \mathbb{R}^n$ heißt **Lipschitz stetig** in D, wenn es eine Konstante $q \geqslant 0$ gibt mit

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\| \leqslant q\|\boldsymbol{x} - \boldsymbol{y}\| \quad \text{für alle } \boldsymbol{x}, \boldsymbol{y} \in D. \tag{7.6}$$

Eine Lipschitz stetige Abbildung $\phi : D \to \mathbb{R}^n$ heißt **kontrahierend** (bzgl. der Vektornorm $\|\cdot\|$), wenn ϕ eine Lipschitz Konstante $q < 1$ besitzt. q heißt dann die

Kontraktionskonstante von ϕ.

Bemerkung 7.3 Aus der Äquivalenz aller Vektornormen auf dem $\mathbb{R}^n$ folgt, dass aus der Lipschitz Stetigkeit von ϕ bzgl. einer Norm folgt, dass ϕ auch bzgl. jeder anderen Vektornorm auf $\mathbb{R}^n$ Lipschitz stetig ist. Für die Kontraktivität einer Abbildung ist dies nicht richtig; sie kann von der gewählten Norm abhängen. Man mache sich dies an einer Drehstreckung des $\mathbb{R}^2$ klar, die bzgl. der Euklidischen Norm sicher kontrahierend ist, wenn der Streckungsfaktor kleiner als 1 ist, bzgl. der Maximumnorm aber nicht, wenn der Streckungsfaktor genügend nahe bei 1 liegt und der Drehwinkel klein genug ist. □

Die Lipschitz Stetigkeit und die Kontraktivität einer Funktion kann man häufig mit Hilfe des Mittelwertsatzes der Differentialrechnung nachweisen.

Satz 7.4

1. *Es sei $\phi : [a,b] \to \mathbb{R}$ stetig differenzierbar. ϕ ist genau dann kontrahierend mit der Kontraktionskonstante q, wenn gilt*

$$q := \max_{x\in[a,b]} |\phi'(x)| < 1.$$

2. *Es sei $D \subset \mathbb{R}^n$ konvex und $\phi : D \to \mathbb{R}^n$ stetig differenzierbar. Es bezeichne $\|\cdot\|$ eine Vektornorm und eine passende Matrixnorm, und es gelte*

$$\|\phi'(\boldsymbol{x})\| \leqslant q < 1 \quad \textit{für alle } \boldsymbol{x} \in D.$$

Dann ist ϕ kontrahierend in D bzgl. $\|\cdot\|$ mit der Kontraktionskonstante q.

Beweis: (i): Ist ϕ kontrahierend auf $[a,b]$ mit der Konstante $q < 1$, so ist

$$|\phi(x) - \phi(y)| \leqslant q|x-y| \quad \text{für alle } x, y \in [a,b].$$

Es folgt

$$\left|\frac{\phi(x)-\phi(y)}{x-y}\right| \leqslant q \quad \text{für alle } x, y \in [a,b], x \neq y,$$

und daher

$$\lim_{y\to x}\left|\frac{\phi(x)-\phi(y)}{x-y}\right| = |\phi'(x)| \leqslant q \quad \text{für alle } x \in [a,b].$$

Ist umgekehrt $|\phi'(x)| \leqslant q < 1$ für alle $x \in [a,b]$, so gilt nach dem Mittelwertsatz mit einem $\xi \in (a,b)$

$$|\phi(x) - \phi(y)| = |\phi'(\xi)| \cdot |x-y|,$$

und daher

$$|\phi(x) - \phi(y)| \leqslant q|x-y| \quad \text{für alle } x, y \in [a,b].$$

(ii): Da D konvex ist, liegt mit $\boldsymbol{x}, \boldsymbol{y} \in D$ auch die Verbindungsgerade in D, und nach dem Mittelwertsatz gilt

$$\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{y})\| \leqslant \sup_{t\in[0,1]} \|\phi'(t\boldsymbol{x} + (1-t)\boldsymbol{y})\| \cdot \|\boldsymbol{x} - \boldsymbol{y}\| \leqslant q\|\boldsymbol{x} - \boldsymbol{y}\|.$$

■

Satz 7.5 (Fixpunktsatz für kontrahierende Abbildungen) *Es sei $D \subset \mathbb{R}^n$ eine abgeschlossene Menge, $\phi : D \to \mathbb{R}^n$ eine kontrahierende Abbildung mit der Kontraktionskonstante q, und es gelte $\phi(D) \subset D$.*

Dann gilt

1. *ϕ hat genau einen Fixpunkt $\bar{\boldsymbol{x}} \in D$.*
2. *Für jeden Startwert $\boldsymbol{x}^0 \in D$ konvergiert die durch $\boldsymbol{x}^{m+1} := \phi(\boldsymbol{x}^m)$ definierte Folge gegen $\bar{\boldsymbol{x}}$.*
3. *Es gelten die Fehlerabschätzungen*

$$\|\bar{\boldsymbol{x}} - \boldsymbol{x}^m\| \leqslant \frac{q^m}{1-q} \|\boldsymbol{x}^1 - \boldsymbol{x}^0\| \tag{7.7}$$

$$\|\bar{\boldsymbol{x}} - \boldsymbol{x}^m\| \leqslant \frac{q}{1-q} \|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\|. \tag{7.8}$$

Beweis: Die Existenz eines Fixpunktes zeigen wir konstruktiv. Sei $\boldsymbol{x}^0 \in D$ beliebig gewählt. Wegen $\phi(D) \subset D$ ist dann die Folge $\{\boldsymbol{x}^m\}$, $\boldsymbol{x}^{m+1} := \phi(\boldsymbol{x}^m)$, definiert und $\{\boldsymbol{x}^m\} \subset D$.

Aus

$$\|\boldsymbol{x}^{m+1} - \boldsymbol{x}^m\| = \|\phi(\boldsymbol{x}^m) - \phi(\boldsymbol{x}^{m-1})\|| \leqslant q \|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\|$$

erhält man durch Induktion für alle $k \in \mathbb{N}$

$$\|\boldsymbol{x}^{m+k} - \boldsymbol{x}^{m+k-1}\| \leqslant q^k \|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\|,$$

und daher

$$\begin{aligned}
\|\boldsymbol{x}^{m+p} - \boldsymbol{x}^m\| &= \left\| \sum_{k=1}^{p} (\boldsymbol{x}^{m+k} - \boldsymbol{x}^{m+k-1}) \right\| \\
&\leqslant \sum_{k=1}^{p} \|\boldsymbol{x}^{m+k} - \boldsymbol{x}^{m+k-1}\| \leqslant \sum_{k=1}^{p} q^k \|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\| \\
&\leqslant q \|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\| \sum_{k=0}^{p-1} q^k \leqslant \frac{q}{1-q} \|\boldsymbol{x}^m - \boldsymbol{x}^{m-1}\| \\
&\leqslant \frac{q^m}{1-q} \|\boldsymbol{x}^1 - \boldsymbol{x}^0\| \to 0.
\end{aligned} \tag{7.9}$$

Nach dem Cauchyschen Konvergenzkriterium ist $\{\boldsymbol{x}^m\}$ konvergent gegen ein $\bar{\boldsymbol{x}}$, da D abgeschlossen ist, gilt $\bar{\boldsymbol{x}} \in D$, und wegen der Stetigkeit von ϕ ist $\bar{\boldsymbol{x}}$ ein Fixpunkt von ϕ.

$\bar{\boldsymbol{x}}$ ist der einzige Fixpunkt von ϕ in D, denn ist $\hat{x}$ ein beliebiger Fixpunkt von ϕ in D, so gilt

$$\|\bar{\boldsymbol{x}} - \hat{x}\| = \|\phi(\bar{\boldsymbol{x}}) - \phi(\hat{x})\| \leqslant q \|\bar{\boldsymbol{x}} - \hat{x}\|,$$

d.h. $0 \leqslant (1-q) \|\bar{\boldsymbol{x}} - \hat{x}\| \leqslant 0$, und wegen $q < 1$ folgt $\bar{\boldsymbol{x}} = \hat{x}$.

Die Fehlerabschätzungen (7.7) und (7.8) erhält man unmittelbar aus (7.9) mit $p \to \infty$. ■

Bemerkung 7.6 Die Ungleichung (7.9) zeigt, dass Abschätzung (7.8) bessere Fehlerschranken liefert als die Abschätzung (7.7).

Der Vorteil der Abschätzung (7.7) liegt darin, dass man mit ihr schon zu Beginn der Rechnung den Fehler einer Iterierten $\boldsymbol{x}^m$ abschätzen kann, ohne $\boldsymbol{x}^m$ berechnet zu haben. Man kann mit ihr also abschätzen, wie hoch der Aufwand sein wird, um eine vorgegebene Genauigkeit ε zu erreichen. Es gilt ja

$$\|\bar{\boldsymbol{x}} - \boldsymbol{x}^m\| \leqslant \frac{q^m}{1-q} \|\boldsymbol{x}^1 - \boldsymbol{x}^0\| \leqslant \varepsilon, \quad \text{falls } m \geqslant \frac{\ln\left(\frac{\varepsilon(1-q)}{\|\boldsymbol{x}^1 - \boldsymbol{x}^0\|}\right)}{\ln q}.$$

m	$x(n)$	a priori	a posteriori	Fehler
0	0.500000000000000			
1	0.877582561890373	$2.00E+00$	$2.00E+00$	$1.38E-01$
2	0.639012494165259	$1.69E+00$	$1.27E+00$	$1.00E-01$
3	0.802685100682335	$1.42E+00$	$8.69E-01$	$6.36E-02$
4	0.694778026788006	$1.19E+00$	$5.73E-01$	$4.43E-02$
5	0.768195831282016	$1.00E+00$	$3.90E-01$	$2.91E-02$
6	0.719165445942419	$8.46E-01$	$2.60E-01$	$1.99E-02$
7	0.752355759421527	$7.12E-01$	$1.76E-01$	$1.33E-02$
8	0.730081063137823	$5.99E-01$	$1.18E-01$	$9.00E-03$
9	0.745120341351440	$5.04E-01$	$7.98E-02$	$6.04E-03$
10	0.735006309014843	$4.24E-01$	$5.37E-02$	$4.08E-03$
20	0.739006779780813	$7.55E-02$	$1.03E-03$	$7.84E-05$
30	0.739083626103480	$1.34E-02$	$1.99E-05$	$1.51E-06$
40	0.739085104225471	$2.39E-03$	$3.82E-07$	$2.90E-08$
50	0.739085132657536	$4.25E-04$	$7.35E-09$	$5.58E-10$

Tabelle 7.2: Fixpunktiteration mit Abschätzung

(7.7) heißt daher eine **a priori Abschätzung** . Mit (7.8) kann man den Fehler von $\boldsymbol{x}^m$ erst abschätzen, nachdem man $\boldsymbol{x}^m$ berechnet hat. (7.8) ist eine **a posteriori Abschätzung**. Sie kann benutzt werden, um während der Rechnung die Güte der Näherung zu kontrollieren und bei ausreichender Genauigkeit die Iteration abzubrechen. □

Beispiel 7.7 Wir betrachten das Fixpunktproblem

$$x = \phi(x) := \cos x, \quad x \in I := [0,1].$$

Da ϕ monoton fallend in $[0,1]$ ist, folgt aus $\phi(0) = 1 \in I$ und $\phi(1) = 0.54 \in I$, dass $\phi(I) \subset I$ gilt.

ϕ ist kontrahierend auf I mit der Kontraktionskonstante $q = 0.85$, denn

$$\max_{x\in I} |\phi'(x)| = \max_{x\in I} |\sin x| = \sin 1 \leqslant 0.85 =: q.$$

Man erhält die Näherungen und Fehlerabschätzungen in Tabelle 7.2. Während die a posteriori Abschätzung ziemlich realistische Werte für den Fehler liefert, wird der Fehler durch die a priori Abschätzung sehr stark überschätzt. □

Oft ist es schwierig, insbesondere bei der Anwendung auf Funktionen von mehreren Veränderlichen, eine Menge D zu bestimmen, die durch ϕ in sich abgebildet wird. Hilfreich hierbei ist

Korollar 7.8 *Es sei $D \subset \mathbb{R}^n$ und $\phi : D \to \mathbb{R}^n$. Es sei ϕ kontrahierend in der Kugel $K := \{\boldsymbol{y} \in R^n : \|\boldsymbol{y} - \boldsymbol{z}\| \leqslant r\}$ mit der Kontraktionskonstante q, und es gelte die Kugelbedingung*

$$\|\phi(\boldsymbol{z}) - \boldsymbol{z}\| \leqslant (1-q)r. \tag{7.10}$$

Dann gelten die Aussagen (i) - (iii) von Satz 7.5 mit K an Stelle von D.

Beweis: Wir haben nur zu zeigen, dass $\phi(K) \subset K$ gilt. Für $\boldsymbol{y} \in K$ ist

$$\|\phi(\boldsymbol{y}) - \boldsymbol{z}\| \leqslant \|\phi(\boldsymbol{y}) - \phi(\boldsymbol{z})\| + \|\phi(\boldsymbol{z}) - \boldsymbol{z}\| \leqslant q\|\boldsymbol{y} - \boldsymbol{z}\| + (1-q)r \leqslant r.$$

■

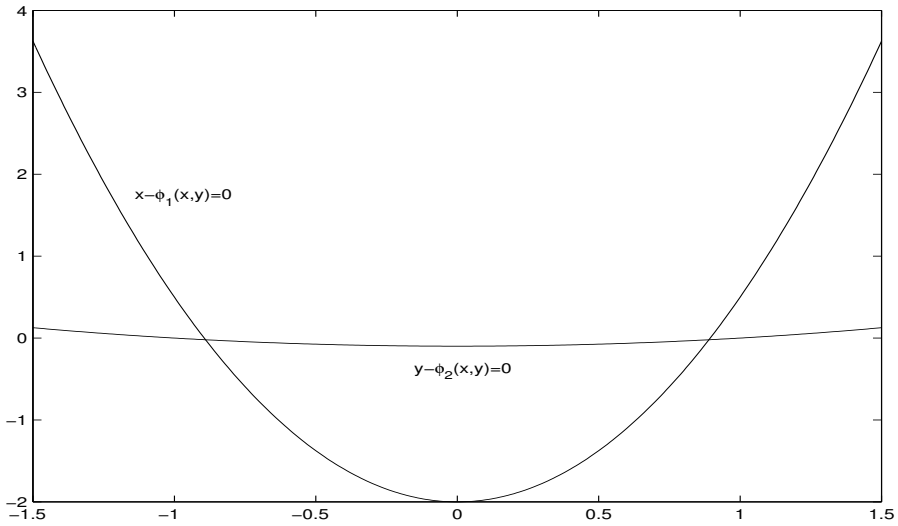

Abbildung 7.2: Zu Beispiel 7.9

Beispiel 7.9 Wir betrachten das Fixpunktproblem

$$\begin{pmatrix} x \\ y \end{pmatrix} = \phi(x,y) := \begin{pmatrix} 0.4 - 0.5x^2 + x + 0.2y \\ 0.1(x^2 + y^2 - 1) \end{pmatrix}. \tag{7.11}$$

Abbildung 7.2 zeigt, dass in der Nähe des Punktes $(x_0, y_0)^T := (1,0)^T$ eine Lösung liegt.

Es gilt

$$\left\| \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \phi(x_0, y_0) \right\|_\infty = \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0.9 \\ 0 \end{pmatrix} \right\|_\infty = 0.1$$

und wegen

$$\phi'(x,y) = \begin{pmatrix} 1-x & 0.2 \\ 0.2x & 0.2y \end{pmatrix}$$

gilt in der Kugel (bzgl. der ∞-Norm)

$$K_r(x_0, y_0) = [1-r, 1+r] \times [-r, r]$$

(im Falle $r \leqslant 1$)

$$\begin{aligned} \|\phi'(x,y)\|_\infty &= \max\{|1-x| + 0.2, 0.2|x| + 0.2|y|) \\ &\leqslant \max(r + 0.2, 0.2(1+r) + 0.2r) = 0.2 + r. \end{aligned}$$

Die Kugelbedingung (7.10) lautet also

$$0.1 \leqslant r(1 - (r + 0.2)) = 0.8r - r^2,$$

und diese ist erfüllt für

$$r \in [0.4 - \sqrt{0.06}, 0.4 + \sqrt{0.06}] \subset [0.16, 0.64].$$

Daher besitzt ϕ genau eine Lösung in der Kugel $K_{0.16}(x_0, y_0)$. Mit der Fixpunktiteration erhält man hierfür die Näherung $(0.88977, -0.02079)^T$. □

Wir haben schon in Beispiel 7.1 gesehen, dass nicht jeder Fixpunkt $\bar{\boldsymbol{x}}$ einer Abbildung ϕ mit der Iteration $\boldsymbol{x}^{m+1} := \phi(\boldsymbol{x}^m)$ erreicht werden kann.

Definition 7.10 Ein Fixpunkt $\bar{\boldsymbol{x}}$ von $\phi : D \to \mathbb{R}^n$, $D \subset \mathbb{R}^n$, heißt **anziehend**, wenn die Fixpunktiteration für jeden genügend nahe bei $\bar{\boldsymbol{x}}$ liegenden Startwert $\boldsymbol{x}^0$ gegen $\bar{\boldsymbol{x}}$ konvergiert (d.h. es gibt ein $\varepsilon > 0$, so dass $\|\boldsymbol{x}^0 - \bar{\boldsymbol{x}}\| \leqslant \varepsilon$, $\boldsymbol{x}^{m+1} := \phi(\boldsymbol{x}^m)$, zur Folge hat $\lim_{m\to\infty} \boldsymbol{x}^m = \bar{\boldsymbol{x}}$); er heißt **abstoßend**, wenn es eine Kugel gibt mit dem Mittelpunkt $\bar{\boldsymbol{x}}$, so dass für jeden Startwert $\boldsymbol{x}^0 \neq \bar{\boldsymbol{x}}$ aus dieser Kugel die Fixpunktiteration aus der Kugel herausführt (d.h. es gibt ein $\varepsilon > 0$, so dass zu jedem $\boldsymbol{x}^0 \in \mathbb{R}^n$ mit $\|\boldsymbol{x}^0 - \bar{\boldsymbol{x}}\| < \varepsilon$, $\boldsymbol{x}^0 \neq \bar{\boldsymbol{x}}$, ein $m \in \mathbb{N}$ existiert mit $\|\boldsymbol{x}^m - \bar{\boldsymbol{x}}\| \geqslant \varepsilon$, wobei $\boldsymbol{x}^{m+1} := \phi(\boldsymbol{x}^m)$).

Im Falle einer stetig differenzierbaren reellen Funktion $\phi : I \to \mathbb{R}$ ist ein Fixpunkt $\bar{x}$ anziehend, falls $|\phi'(\bar{x})| < 1$ gilt, denn zu $\varepsilon > 0$ mit $|\phi'(\bar{x})| + \varepsilon =: q < 1$ gibt es ein $\delta > 0$ mit

$$|\phi'(x)| \leqslant |\phi'(\bar{x})| + |\phi'(x) - \phi'(\bar{x})| \leqslant |\phi'(\bar{x})| + \varepsilon = q < 1$$

für alle $x \in [\bar{x}-\delta, \bar{x}+\delta] =: J$, d.h. ϕ ist kontrahierend auf J, und wegen $|\bar{x}-\phi(\bar{x})| = 0 \leqslant (1-q)\,\delta$ wird J nach Korollar 7.8 durch ϕ in sich abgebildet.

Ist $|\phi'(\bar{x})| > 1$, so gibt es ein Intervall $J := [\bar{x} - r, \bar{x} + r]$ mit

$$|\phi'(x)| \geqslant |\phi'(\bar{x})| - |\phi'(x) - \phi'(\bar{x})| \geqslant q > 1$$

für alle $x \in J$, und daher gilt für $x \in J$ mit einem $\xi = \bar{x} + \theta\,(x - \bar{x})$

$$|\bar{x} - \phi(x)| = |\phi(\bar{x}) - \phi(x)| = |\phi'(\xi)| \cdot |\bar{x} - x| \geqslant q\,|\bar{x} - x|,$$

und es folgt, dass $\bar{x}$ abstoßend ist.

Ist $|\phi'(\bar{x})| = 1$, so kann $\bar{x}$ anziehend oder abstoßend oder keines von beiden sein. Dies zeigen die Beispiele $\phi(x) := x - x^3$, $\phi(x) := x + x^3$ und $\phi(x) := x - x^2$ für den Fixpunkt $\bar{x} = 0$.

7.2 Nullstellen reeller Funktionen

Wir betrachten für das Nullstellenproblem

$$f(x) = 0 \tag{7.12}$$

für die reelle Funktion $f : [a, b] \to \mathbb{R}$ zwei Typen von Verfahren. Zunächst untersuchen wir Methoden, die sich durch Linearisierung von f ergeben, danach Verfahren, die auf dem Zwischenwertsatz beruhen und Einschließungen der Nullstelle liefern. Verbreiteter ist der erste Typ von Methoden (er ist auch der Ausgangspunkt für die Entwicklung von Methoden für nichtlineare Systeme von Gleichungen), effizienter der zweite Typ.

7.2.1 Newton Verfahren

Wir betrachten die nichtlineare Gleichung (7.12) und setzen voraus, dass $f : [a, b] \to \mathbb{R}$ differenzierbar ist.

Um eine gegebene Näherung x_0 für eine Nullstelle $\bar{x} \in (a, b)$ von f zu verbessern, ersetzen wir f durch ihre Linearisierung

$$\tilde{f}(x) = f(x_0) + f'(x_0)\,(x - x_0)$$

in x_0. Gilt $f'(x_0) \neq 0$, so besitzt die Ersatzfunktion $\tilde{f}$ genau eine Nullstelle

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)},$$

die wir als neue Näherung für $\bar{x}$ auffassen.

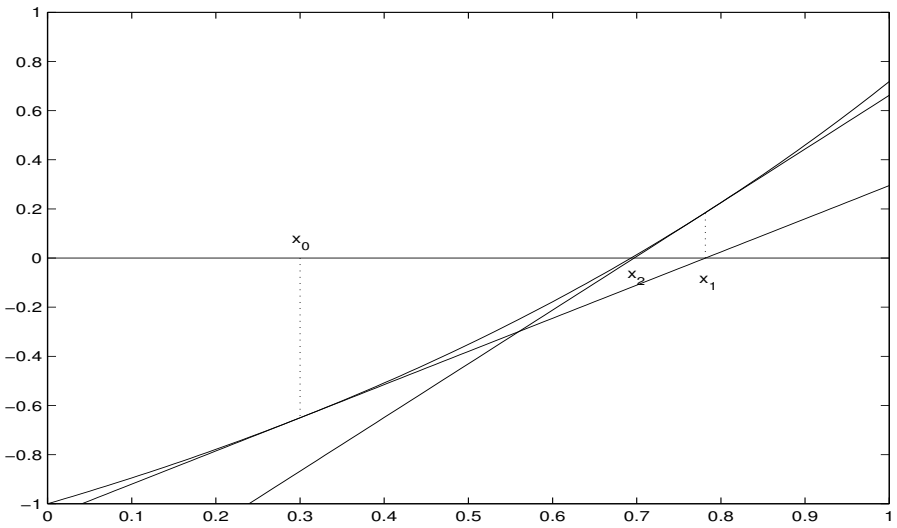

Abbildung 7.3: Newton Verfahren

m	x_m	Fehler
0	2.5	$4.26e-02$
1	2.5443	$1.56e-03$
2	2.5426434	$2.01e-06$
3	2.5426413577769	$3.34e-12$

Tabelle 7.3: Newton Verfahren; Beispiel 7.11

Wiederholt man diesen Schritt iterativ mit x_n an Stelle von x_0, so erhält man das **Newton Verfahren**

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)}. \tag{7.13}$$

Beispiel 7.11 Wir bestimmen den größeren der beiden Fixpunkte von $\phi(x) = 0.2\exp(x)$. Aus Abbildung 7.1 liest man die Anfangsnäherung $x_0 = 2.5$ ab, und mit $f(x) := x - 0.2\exp(x)$ lautet das Newton Verfahren

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n - 0.2\exp(x_n)}{1 - 0.2\exp(x_n)}.$$

Hiermit erhält man die Näherungen und Fehler in Tabelle 7.3. □

Das Newton Verfahren kann geschrieben werden als

$$x_{n+1} = \phi(x_n), \quad n \in \mathbb{N}_0,$$

mit

$$\phi(x) := x - \frac{f(x)}{f'(x)}.$$

ϕ ist auf $D := \{x \in [a,b] : f'(x) \neq 0\}$ definiert, und die Nullstellen von f in D sind genau die Fixpunkte von ϕ in D. Wendet man den Fixpunktsatz für kontrahierende Abbildungen (Satz 7.5) auf dieses ϕ an, so erhält man

Satz 7.12 *Es sei $f : I \to \mathbb{R}$ zweimal stetig differenzierbar und $\bar{x}$ eine einfache Nullstelle von f im Innern von I. Dann gibt es ein $r > 0$, so dass das Newton Verfahren für alle Startwerte $x_0 \in I$ mit $|\bar{x} - x_0| \leqslant r$ gegen $\bar{x}$ konvergiert.*

Beweis: $\bar{x}$ ist Fixpunkt von

$$\phi(x) := x - \frac{f(x)}{f'(x)}.$$

Wegen der Stetigkeit von f' und $f'(\bar{x}) \neq 0$ gilt $f'(x) \neq 0$ für alle $x \in J := [\bar{x}-\rho, \bar{x}+\rho]$ mit einem geeigneten $\rho > 0$. Daher ist ϕ in J definiert und stetig differenzierbar mit

$$\phi'(x) = \frac{f(x)\, f''(x)}{(f'(x))^2},$$

d.h. $\phi'(\bar{x}) = 0$, und $\bar{x}$ ist ein anziehender Fixpunkt von ϕ. ■

Satz 7.12 ist ein typischer lokaler Konvergenzsatz: Wenn der Startwert x_0 nur genügend nahe an der Lösung liegt ($|x - x_0| \leqslant r$), so konvergiert das Verfahren. Über die Größe von r wird nichts ausgesagt.

Aus den Abschätzungen von Satz 7.5 folgt, dass der Fehler beim Newton Verfahren wie eine geometrische Folge gegen 0 geht. Tatsächlich gilt nach dem Taylorschen Satz

$$\begin{aligned} |x_{n+1} - \bar{x}| &= \left| x_n - \frac{f(x_n)}{f'(x_n)} - \bar{x} \right| = \left| \frac{f(\bar{x}) - f(x_n) - f'(x_n)\,(\bar{x} - x_n)}{f'(x_n)} \right| \\ &= \frac{1}{2} \left| \frac{f''(\xi_n)}{f'(x_n)}\, (\bar{x} - x_n)^2 \right|, \qquad \xi_n = \bar{x} + \theta_n\,(\bar{x} - x_n). \end{aligned}$$

Da zu $\varepsilon > 0$ ein $r > 0$ existiert mit

$$|f'(x)| \geqslant \frac{1}{2}|f'(\bar{x})| \quad \text{und} \quad |f''(x)| \leqslant \varepsilon + |f''(\bar{x})|$$

für alle x mit $|\bar{x} - x| \leqslant r$ und da nach Satz 7.12 x_n gegen $\bar{x}$ konvergiert, erhält man

$$|x_{n+1} - \bar{x}| \leqslant \frac{|f''(\bar{x})| + \varepsilon}{|f'(\bar{x})|}\, |\bar{x} - x_n|^2 =: C\, |\bar{x} - x_n|^2.$$

Bis auf die multiplikative Konstante C wird also der Fehler beim Newton Verfahren von Schritt zu Schritt quadriert, was die rasche Konvergenz des Verfahrens in Beispiel 7.11 erklärt.

Um die Konvergenzgeschwindigkeit von Folgen zu vergleichen, führen wir die folgenden Begriffe ein:

Definition 7.13 Sei $\{\boldsymbol{x}^m\} \subset \mathbb{R}^n$ eine konvergente Folge mit $\lim_{m\to\infty} \boldsymbol{x}^m = \bar{\boldsymbol{x}}$. Die Folge $\{\boldsymbol{x}^m\}$ **besitzt wenigstens die Q-Ordnung** $p \in [1, \infty)$, wenn es eine Konstante $C > 0$ und ein $m_0 \in \mathbb{N}$ gibt mit

$$\frac{\|\bar{\boldsymbol{x}} - \boldsymbol{x}^{m+1}\|}{\|\bar{\boldsymbol{x}} - \boldsymbol{x}^m\|^p} \leqslant C \qquad \text{für alle } m \geqslant m_0.$$

Ist $p = 1$, so fordern wir zusätzlich $C < 1$, damit die lokale Konvergenz überhaupt gesichert ist.

Das maximale p mit dieser Eigenschaft heißt die **Q-Konvergenzordnung** der Folge.

Ein Iterationsverfahren hat die Q-Konvergenzordnung p, wenn jede mit ihm erzeugte, konvergente Folge die Q-Konvergenzordnung p hat.

Eine Folge (ein Verfahren) **konvergiert Q-linear**, falls $p = 1$ ist, **Q-quadratisch** im Falle $p = 2$ und **Q-superlinear**, falls

$$\limsup_{m\to\infty} \frac{\|\bar{\boldsymbol{x}} - \boldsymbol{x}^{m+1}\|}{\|\bar{\boldsymbol{x}} - \boldsymbol{x}^{m}\|} = 0$$

gilt.

Wir werden später noch den Begriff der Konvergenzordnung modifizieren. Zur Unterscheidung verwenden wir für die hier eingeführten Konvergenzbegriffe das Präfix Q, da sich die Ordnung auf Quotienten von aufeinanderfolgenden Fehlern bezieht.

Die durch Satz 7.5 erfassten Verfahren konvergieren wenigstens Q-linear, das Newton Verfahren konvergiert lokal Q-quadratisch gegen einfache Nullstellen von f, wenn f zweimal stetig differenzierbar in einer Umgebung der Nullstelle ist.

Allgemeiner erhält man

Satz 7.14 *Sei $I := [a, b] \subset \mathbb{R}$, $p > 1$ und $\bar{x} \in (a, b)$ ein Fixpunkt von $\phi \in C^p[a, b]$ mit $\phi^{(j)}(\bar{x}) = 0$, $j = 1, 2, \ldots, p - 1$.*

Dann gibt es eine Umgebung $U(\bar{x}) \subset (a, b)$ von $\bar{x}$, so dass für alle Startwerte $x_0 \in U(x)$ die Folge $x_{n+1} = \phi(x_n)$ von mindestens der Q-Ordnung p gegen $\bar{x}$ konvergiert.

Beweis: Dass das Verfahren lokal konvergent ist, erhält man wieder aus Korollar 7.8. Nach dem Taylorschen Satz gilt für $x_n \in (a, b)$,

$$|x_{n+1} - \bar{x}| = |\phi(x_n) - \phi(\bar{x})| = \left| \frac{\phi^{(p)}(\xi)}{p!} (\bar{x} - x_n)^p \right|$$

für ein $\xi \in (a, b)$, mit

$$C := \max \left\{ \frac{1}{p!} |\phi^{(p)}(\xi)| \,:\, \xi \in [a, b] \right\}$$

also die Abschätzung

$$|x_{n+1} - \bar{x}| \leqslant C |x_n - \bar{x}|^p.$$

■

Bemerkung 7.15 Mit

$$\phi(x) := x - \frac{f(x)}{f'(x)}$$

erhält man aus Satz 7.14 wegen

$$\phi'(x) = \frac{f(x)\ f''(x)}{(f'(x))^2}$$

wieder die Q-quadratische Konvergenz des Newton Verfahrens gegen einfache Nullstellen von f (allerdings unter der stärkeren Voraussetzung $f \in C^3[a, b]$). □

Ist $\bar{x}$ eine mehrfache Nullstelle von f, so konvergiert das Newton Verfahren immer noch lokal, aber nur linear, also wie vom Fixpunktsatz für kontrahierende Abbildungen (Satz 7.5) vorhergesagt. Genauer gilt Satz 7.16:

n	Newton für f	(7.14) mit $\alpha = 3$	Newton für g
0	$1.00e+00$	$1.00e+00$	$1.00e+00$
1	$6.55e-01$	$3.46e-02$	$6.48e-02$
2	$4.34e-01$	$1.38e-06$	$1.81e-05$
3	$2.88e-01$	$2.33e-13$	$3.40e-14$

Tabelle 7.4: Newton Verfahren

Satz 7.16 *Ist $f : [a,b] \to \mathbb{R}$ $(k+3)$-mal stetig differenzierbar, und besitzt f eine Nullstelle $\bar{x} \in (a,b)$ der Vielfachheit $k \geqslant 2$, so konvergiert das Verfahren*

$$x_{n+1} := x_n - \alpha \frac{f(x_n)}{f'(x_n)} \tag{7.14}$$

für alle $\alpha \in (0, 2k)$ lokal Q-linear gegen $\bar{x}$. Für $\alpha = k$ ist die Konvergenz sogar Q-quadratisch.

Beweis: Es ist $f(x) = (x-\bar{x})^k\, g(x)$, wobei $g(\bar{x}) \neq 0$ gilt und g dreimal stetig differenzierbar ist. Hiermit kann man die Iterationsvorschrift (7.14) schreiben als

$$x_{n+1} = \phi(x_n), \quad \phi(x) = x - \alpha \frac{(x-\bar{x})\, g(x)}{k\, g(x) + (x-\bar{x})\, g'(x)}.$$

ϕ ist sicher in einer Umgebung von $\bar{x}$ definiert und differenzierbar, und es gilt mit $h(x) := k\, g(x) + (x-\bar{x})\, g'(x)$

$$\phi'(x) = 1 - \alpha \frac{h(x)\Big(g(x) + (x-\bar{x})\, g'(x)\Big) - (x-\bar{x})\, g(x)\, h'(x)}{(h(x))^2},$$

d.h.

$$\phi'(\bar{x}) = 1 - \alpha \frac{h(\bar{x})\, g(\bar{x})}{(h(\bar{x}))^2} = 1 - \frac{\alpha}{k}.$$

Für $\alpha \in (0, 2k)$ gilt $|\phi'(\bar{x})| < 1$, und $\bar{x}$ ist ein anziehender Fixpunkt von ϕ. Im Falle $\alpha = k$ gilt $\phi'(\bar{x}) = 0$, und die quadratische Konvergenz folgt aus Satz 7.14. ■

Im allgemeinen ist die Vielfachheit einer Nullstelle nicht bekannt. Ist $\bar{x}$ eine k-fache Nullstelle von f, so ist $\bar{x}$ eine $(k-1)$-fache Nullstelle von f', und daher hat

$$g(x) := \frac{f(x)}{f'(x)}$$

die einfache Nullstelle $\bar{x}$. Das Newton Verfahren für g konvergiert also Q-quadratisch.

Beispiel 7.17 Für die dreifache Nullstelle $\bar{x} = 0$ von

$$f(x) = \sin x - x$$

erhält man die Fehler in Tabelle 7.4.

Da die Berechnung der Ableitung $f'(x)$ oft aufwendig ist, ersetzt man im Newton Verfahren $f'(x_n)$ durch $f'(x_0)$, berechnet also die Ableitung nur im Startwert x_0. Man erhält dann das **vereinfachte Newton Verfahren**

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_0)}. \tag{7.15}$$

Hierfür gilt der folgende Konvergenzsatz

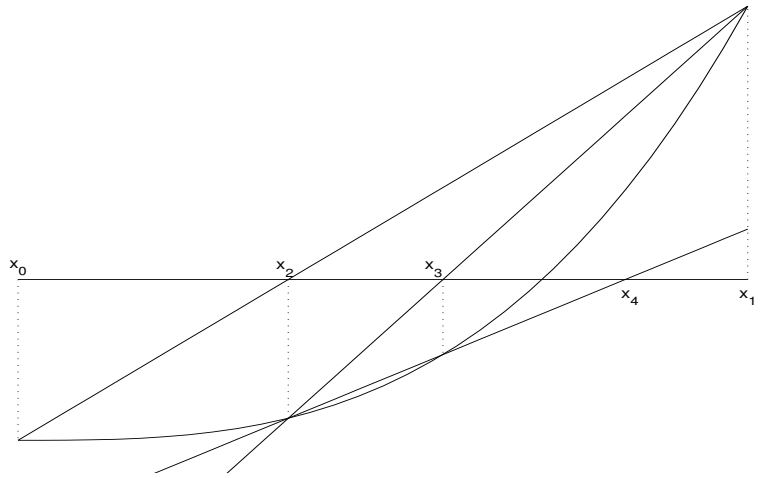

Abbildung 7.4: Sekantenverfahren

Satz 7.18 *Ist $f : [a,b] \to \mathbb{R}$ stetig differenzierbar und $\bar{x} \in (a,b)$ eine einfache Nullstelle von f, so gibt es ein $r > 0$, so dass das vereinfachte Newton Verfahren (7.15) für alle $x_0 \in [\bar{x}-r, \bar{x}+r]$ gegen $\bar{x}$ konvergiert. Die Konvergenz ist Q-linear.*

Beweis: Wir wenden Korollar 7.8 für festes $x_0 \in I_r := [\bar{x}-r, \bar{x}+r]$ und geeignetes $r > 0$ auf die Iterationsfunktion

$$\phi(x; x_0) := x - \frac{f(x)}{f'(x_0)}$$

an.

Wegen $\phi(\bar{x}; x_0) = \bar{x}$ muss das Intervall I_r nur so klein gewählt werden, dass für jedes $x_0 \in I_r$ die Funktion $\phi(\cdot; x_0) : I_r \to \mathbb{R}$ kontrahierend auf I_r ist, d.h. so dass für ein $q \in [0,1)$

$$\max_{x \in I_r} |1 - \frac{f'(x)}{f'(x_0)}| \leqslant q$$

gilt. Dass diese Wahl von r möglich ist, folgt aus der Stetigkeit der Funktion $\psi(x, x_0) := 1 - \dfrac{f'(x)}{f'(x_0)}$ in einer Umgebung des Punktes $(\bar{x}, \bar{x})$ und $\psi(\bar{x}, \bar{x}) = 0$. ■

Eine weitere Möglichkeit zur Gewinnung eines ableitungsfreien Verfahrens ist, die Ableitung $f'(x_n)$ im Newton Verfahren durch den Differenzenquotienten

$$\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

(die Funktion f also durch ihre Sekante zu den Stützstellen x_n und x_{n-1}) zu ersetzen. Man erhält dann das **Sekantenverfahren** (manchmal auch nicht ganz korrekt **regula falsi Verfahren**)

$$x_{n+1} := x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n).$$

Wie das Newton Verfahren konvergiert das Sekantenverfahren lokal, d.h. wenn die Startwerte x_0 und x_1 genügend nahe bei der Nullstelle $\bar{x}$ gewählt werden, falls $\bar{x}$ eine einfache Nullstelle ist.

Für das Sekantenverfahren verwenden wir einen schwächeren Konvergenzordnungsbegriff als den der Q-Ordnung.

Definition 7.19 Eine Folge $\{\boldsymbol{x}^m\} \subset \mathbb{R}^n$ konvergiert gegen $\bar{\boldsymbol{x}}$ von wenigstens der **R-Ordnung** p, wenn es eine reelle Nullfolge $\{r_m\}$ gibt mit

$$\|\boldsymbol{x}^m - \bar{\boldsymbol{x}}\| \leqslant r_m,$$

die wenigstens von der Q-Ordnung p konvergiert.

Die R-Ordnung wird für Mehrpunktverfahren (die $\boldsymbol{x}^{m+1}$ unter Benutzung von mehreren Vorgängern $\boldsymbol{x}^m, \ldots, \boldsymbol{x}^{m-k}$ bestimmen) wie das Sekantenverfahren häufig mit Hilfe der positiven Nullstelle (engl.: root) eines Polynoms bestimmt. Dies erklärt das Präfix R.

Satz 7.20 *Sei f zweimal stetig differenzierbar in der Umgebung einer einfachen Nullstelle $\bar{x}$ von f. Dann konvergiert das Sekantenverfahren lokal von wenigstens der R-Ordnung $p = \frac{1}{2}(1+\sqrt{5})$.*

Beweis: Es gilt

$$\begin{aligned} x_{n+1} - \bar{x} &= x_n - \bar{x} - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \\ &= x_n - \bar{x} - \frac{f(x_n)}{[x_n, x_{n-1}]}, \end{aligned}$$

und wegen

$$f(x_n) = \frac{f(x_n) - f(\bar{x})}{x_n - \bar{x}}(x_n - \bar{x}) = [x_n, \bar{x}](x_n - \bar{x})$$

erhält man weiter

$$\begin{aligned} x_{n+1} - \bar{x} &= (x_n - \bar{x}) \left(1 - \frac{[x_n, \bar{x}]}{[x_n, x_{n-1}]}\right) \\ &= (x_n - \bar{x})(x_{n-1} - \bar{x}) \frac{[x_{n-1}, x_n, \bar{x}]}{[x_n, x_{n-1}]} \\ &= (x_n - \bar{x})(x_{n-1} - \bar{x}) \frac{0.5 f''(\xi_2)}{f'(\xi_1)} \end{aligned}$$

mit $\xi_1 \in I(x_{n-1}, x_n)$ und $\xi_2 \in I(x_{n-1}, x_n, \bar{x})$.

Wegen $f'(\bar{x}) \neq 0$ und der Stetigkeit von f' und f'' gibt es $\varepsilon > 0$ und $M > 0$, so dass

$$\left| \frac{1}{2} \frac{f''(\xi_2)}{f'(\xi_1)} \right| \leqslant M \quad \text{für alle } \xi_1, \xi_2 \text{ mit } |\bar{x} - \xi_j| \leqslant \varepsilon.$$

Sei $e_n := M|\bar{x} - x_n|$ und $e_0, e_1 < \min\{1, \varepsilon M\}$. Dann folgt

$$e_{n+1} \leqslant e_n e_{n-1} \quad \text{für alle } n \in \mathbb{N}.$$

Aus dieser Ungleichung erhält man nun, dass das Sekantenverfahren konvergiert mit der R-Konvergenzordnung $p := \frac{1}{2}(1+\sqrt{5}) = 1.618\ldots$.

Es sei nämlich $k := \max\{e_0, e_1^{1/p}\} < 1$. Dann gilt

$$e_n \leqslant k^{(p^n)}, \quad \text{für alle } n \geqslant 0,$$

denn für $n = 0$ und $n = 1$ ist diese Ungleichung nach Wahl von k richtig, und aus ihrer Gültigkeit für $0, 1, \ldots, n$ folgt wegen $p^2 = p + 1$

$$e_{n+1} \leqslant e_n e_{n-1} \leqslant k^{(p^n)} k^{(p^{n-1})} = k^{(p^{n-1})(p+1)} = k^{(p^{n+1})}.$$

■

Da das Sekantenverfahren nur eine Funktionsauswertung in jedem Schritt benötigt (aus dem vorhergehenden Schritt ist $f(x_{n-1})$ ja bekannt) und keine Auswertung der Ableitung, ist es effizienter als das Newton Verfahren. Es gilt nämlich

$$k^{(p^{n+2})} = \left(k^{(p^n)}\right)^{p^2} = \left(k^{(p^n)}\right)^{p+1},$$

und daher wird der Fehler e_{2n} der Folge x_{2n} (in der nur jeder zweite Sekantenschritt gezählt wird) durch eine Folge majorisiert, die von der Ordnung $p + 1 = 2.618\ldots$ konvergiert. Bei gleichem Aufwand konvergiert das Sekantenverfahren also schneller als das Newton Verfahren.

7.2.2 Einschließende Verfahren

Ist $f : [a, b] \to \mathbb{R}$ stetig und gilt $f(a)f(b) \leqslant 0$, so besitzt f in $[a, b]$ eine Nullstelle $\bar{x}$. Bewiesen wird dieser Satz mit Hilfe der **Bisektion**, die zugleich ein numerisches Verfahren liefert:

Algorithmus 7.21 (Bisektion)

```
fa=f(a); fb=f(b);
repeat
   c=0.5(a+b); fc=f(c);
   if fa*fc < 0
      b=c; fb=fc;
   else
      a=c; fa=fc;
   end
until abs(a-b) < eps
```

Dieses Verfahren konvergiert R-linear, denn die Länge des Intervalls, das die Nullstelle von f enthält wird in jedem Schritt halbiert. Um eine Dezimalstelle zu gewinnen, sind also (etwas mehr als) drei Funktionsauswertungen erforderlich.

Es konvergiert aber i.a. nicht Q-linear, wie das folgende Beispiel zeigt. Damit ist der Begriff der Q-Konvergenzordnung echt stärker als der der R-Konvergenzordnung.

Beispiel 7.22 Die stetige Funktion f besitze in

$$\bar{x} = \frac{1}{7} = \sum_{\nu=1}^{\infty} \left(\frac{1}{8}\right)^{\nu}$$

eine Nullstelle, und es gelte $f(x) \neq 0$ für $x \neq \frac{1}{7}$.

Führt man für diese Funktion das Bisektionsverfahren mit dem Startwert $a_0 = 0$

und $b_0 = 1$ durch, so erhält man die Iterierten

$$\begin{aligned}
x_1 &= \frac{1}{2},\ x_2 = \frac{1}{2} - \frac{1}{4},\ x_3 = \frac{1}{2} - \frac{1}{4} - \frac{1}{8} \\
x_4 &= \left(\frac{1}{2} - \frac{1}{4} - \frac{1}{8}\right) + \frac{1}{16},\ x_5 = \left(\frac{1}{2} - \frac{1}{4} - \frac{1}{8}\right) + \frac{1}{16} - \frac{1}{32} \\
x_6 &= \left(\frac{1}{2} - \frac{1}{4} - \frac{1}{8}\right) + \frac{1}{16} - \frac{1}{32} - \frac{1}{64} \\
x_7 &= \left(\frac{1}{2} - \frac{1}{4} - \frac{1}{8}\right) + \left(\frac{1}{16} - \frac{1}{32} - \frac{1}{64}\right) + \frac{1}{128}
\end{aligned}$$

d.h.

$$x_{3n} = \sum_{\nu=1}^{n} (4-2-1)\left(\frac{1}{8}\right)^{\nu},\ x_{3n+1} = x_{3n} + 4\left(\frac{1}{8}\right)^{n+1},\ x_{3n+2} = x_{3n} + (4-2)\left(\frac{1}{8}\right)^{n+1}.$$

Damit gilt für den Fehler

$$\begin{aligned}
|\bar{x} - x_{3n}| &= \left| \sum_{\nu=n+1}^{\infty} \left(\frac{1}{8}\right)^{\nu} \right| = \frac{1}{7} \cdot \left(\frac{1}{8}\right)^{n} \\
|\bar{x} - x_{3n+1}| &= \left| \bar{x} - x_{3n} - 4\left(\frac{1}{8}\right)^{n+1} \right| = \frac{5}{14} \cdot \left(\frac{1}{8}\right)^{n} \\
|\bar{x} - x_{3n+2}| &= \left| \bar{x} - x_{3n} - 2\left(\frac{1}{8}\right)^{n+1} \right| = \frac{3}{28} \cdot \left(\frac{1}{8}\right)^{n}.
\end{aligned}$$

Der Fehler fällt also nicht monoton, sondern wird im Schritt von x_{3n} zu x_{3n+1} für alle $n \in \mathbb{N}$ sogar wieder vergrößert, und damit liegt keine Q-lineare Konvergenz vor. □

Da die Funktionswerte $f(a)$ und $f(b)$ bekannt sind, kann man an Stelle des Intervallmittelpunktes als neuen Punkt (wie beim Sekantenverfahren) die Nullstelle der Sekante durch die Punkte $(a, f(a))$ und $(b, f(b))$ wählen. Man erhält die **regula falsi**:

Algorithmus 7.23 (regula falsi)

```
fa=f(a); fb=f(b);
repeat
   c=a-(b-a)*fa/(fb-fa); fc=f(c);
   if fa*fc < 0
      b=c; fb=fc;
   else
      a=c; fa=fc;
   end
until abs(a-b) < eps
```

Nachteil der regula falsi ist, dass in vielen Fällen einer der beiden Intervallendpunkte „hängen bleibt“, nämlich dann, wenn ein Intervall erreicht ist, in dem f konvex oder konkav ist. Ist f (wie in Abbildung 7.5) konvex und monoton wachsend in $[a, b]$, so liegt die Nullstelle der Sekante links von der Nullstelle von f, und der rechte Intervallendpunkt bleibt unverändert. Dies gilt auch für die folgenden Schritte.

Das regula falsi Verfahren wird beschleunigt, indem man den Funktionswert auf der hängen bleibenden Seite modifiziert, wenn zwei aufeinanderfolgende Schritte auf

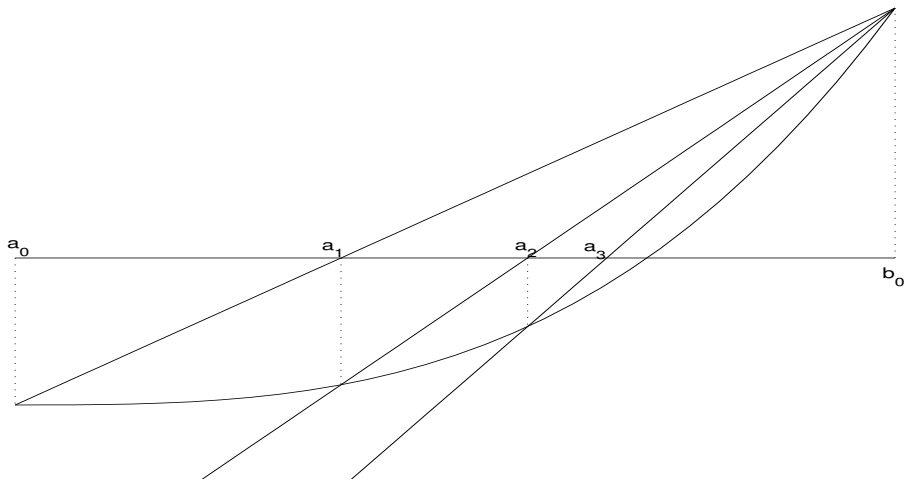

Abbildung 7.5: regula falsi

derselben Seite der Nullstelle liegen (tatsächlich: wenn die Funktionswerte gleiches Vorzeichen haben).

Ein erstes Verfahren dieses Typs ist das **Illinois Verfahren**. Es wurde vermutlich ca. 1950 am Rechenzentrum der University of Illinois eingeführt. In Dowell und Jarrett [22] wurde gezeigt, dass die R-Konvergenzordnung $\sqrt[3]{3} \approx 1.442$ ist.

Beim Illinois Verfahren wird der Funktionswert auf der hängenden Seite stets halbiert. Abbildung 7.6 zeigt die ersten Schritte des Illinois Verfahrens. Nach der Bestimmung von x_3 wird dem rechts liegenden Punkt x_1 als neuer Funktionswert $0.5f(x_1)$ für den Sekantenschritt zugeordnet. Dadurch wird x_4 größer als die Nullstelle von f.

Algorithmus 7.24 (Illinois Verfahren)

```
f1=f(x1); f2=f(x2);
repeat
   x3=x2-(x2-x1)*f2/(f2-f1); f3=f(x3);
   if f2*f3 < 0
      x1=x2; f1=f2; x2=x3; f2=f3;
   else
      x2=x3; f2=f3; f1=0.5*f1;
   end
until abs(x1-x2) < eps
```

Etwas bessere Konvergenzeigenschaften hat das **Pegasus Verfahren**, dessen Ursprung ebenfalls im Dunkeln liegt. Bei ihm wird im Falle $f_2 * f_3 > 0$ (mit den Bezeichnungen aus Algorithmus 7.24) der Funktionswert f_1 multipliziert mit $f_2/(f_2 + f_3) \in (0, 1)$. Dieses Verfahren hat (vgl. Dowell und Jarrett [23]) die R-Konvergenzordnung $\sqrt[4]{7.275} \approx 1.642$, ist also sogar noch etwas schneller als das Sekantenverfahren und liefert zusätzlich eine Fehlerabschätzung in Form einer Einschließung.

Algorithmus 7.25 (Pegasus Verfahren)

```
f1=f(x1); f2=f(x2);
repeat
   x3=x2-(x2-x1)*f2/(f2-f1); f3=f(x3);
```

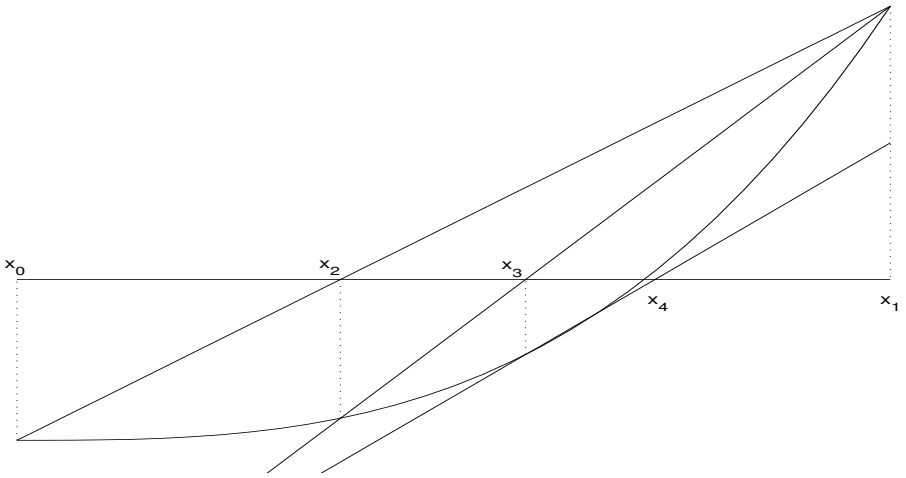

Abbildung 7.6: Illinois Verfahren

```
   if f2*f3 < 0
      x1=x2; f1=f2; x2=x3; f2=f3;
   else
      f1=f1*f2/(f2+f3); x2=x3; f2=f3;
   end
until abs(x1-x2) < eps
```

Eine weitere Verbesserung wurde von Anderson und Björck [4] eingeführt. Im Falle $f_2 * f_3 > 0$ wird die folgende Modifikation vorgenommen: Es wird eine Parabel durch (x_1, f_1), (x_2, f_2) und (x_3, f_3) gelegt und die Tangente im mittleren Punkt (x_3, f_3) an die Parabel konstruiert. Schneidet diese Tangente die x-Achse zwischen den Punkten x_1 und x_3, so wird dieser Punkt als die nächste Näherung x_4 für die Nullstelle gewählt. Ist dies nicht der Fall, so wird ein Illinois Schritt ausgeführt. Die Konstruktion des Punktes x_4 ist in Abbildung 7.7 demonstriert. q bezeichnet den Graphen der interpolierenden quadratischen Funktion.

Die Lagrangesche Interpolationsformel liefert für die interpolierende quadratische Funktion

$$q(x) = f_1 \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + f_2 \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + f_3 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)},$$

mit der Ableitung in x_3

$$m = f_1 \frac{x_3-x_2}{(x_1-x_2)(x_1-x_3)} + f_2 \frac{x_3-x_1}{(x_2-x_1)(x_2-x_3)} + f_3 \frac{2x_3-x_1-x_2}{(x_3-x_1)(x_3-x_2)},$$

und unter Berücksichtigung von

$$x_3 = x_2 - \frac{x_1-x_2}{f_1-f_2} f_2 = x_1 - \frac{x_1-x_2}{f_1-f_2} f_1$$

rechnet man leicht nach, dass gilt

$$f_1^* := f_3 + m(x_1-x_3) = f_1 \left(1 - \frac{f_3}{f_2}\right).$$

Man kann x_4 also durch einen Sekantenschritt mit den Punkten (x_3, f_3) und (x_1, f_1^*) berechnen.

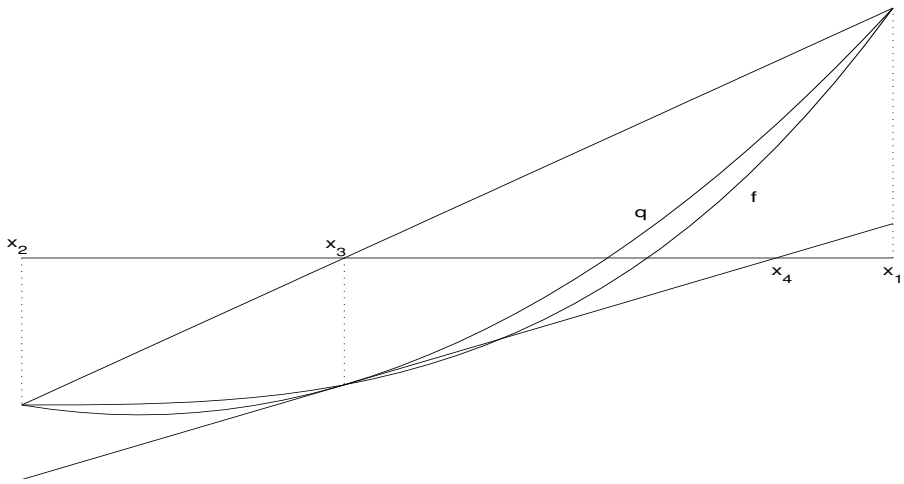

Abbildung 7.7: Anderson Björck Verfahren

Ist $|f_2| < |f_3|$, so haben f_1 und f_1^* entgegengesetztes Vorzeichen, und x_4 liegt nicht zwischen x_1 und x_3. In diesem Fall wird ein Illinois Schritt ausgeführt. Man erhält das folgende Verfahren:

Algorithmus 7.26 (Anderson Björck Verfahren)

```
f1=f(x1); f2=f(x2);
repeat
   x3=x2-(x2-x1)*f2/(f2-f1); f3=f(x3);
   if f2*f3 < 0
      x1=x2; f1=f2; x2=x3; f2=f3;
   else
      if |f2| < |f3|
         g=0.5;
      else
         g=1-f3/f2;
      end
      f1=g*f1;
      x2=x3; f2=f3;
   end
until abs(x1-x2) < eps
```

Anders als beim Illinois und beim Pegasus Verfahren ist nicht klar, in welcher Folge asymptotisch Sekantenschritte und modifizierte Sekantenschritte im Anderson Björck Verfahren auftreten. Man erhält daher nicht eine feste Konvergenzordnung, sondern je nach der Folge der Schritte ergibt sich eine R-Konvergenzordnung von 1.682 oder 1.710 (vgl. Anderson und Björck [4]).

Eine weitere Modifikation wurde von King [42] für das Pegasus Verfahren vorgeschlagen. Es werden niemals nacheinander zwei Sekantenschritte zugelassen, sondern auf jeden Sekantenschritt muss ein modifizierter Schritt folgen.

Man geht aus von zwei Punkten x_0 und x_1 mit $f_0 f_1 < 0$, wobei $f_j := f(x_j)$ gesetzt ist.

Algorithmus 7.27 (King Verfahren)

```
repeat
```

m	reg. falsi	Illinois	Pegasus	And. Björck	King	ABK
1	$1.17e-01$	$1.17e-01$	$1.17e-01$	$1.17e-01$	$1.17e-01$	$1.17e-01$
2	$5.02e-02$	$5.02e-02$	$5.02e-02$	$5.02e-02$	$2.02e-02$	$8.48e-03$
3	$2.10e-02$	$6.00e-03$	$8.56e-03$	$1.49e-03$	$6.61e-04$	$8.35e-04$
4	$8.76e-03$	$2.45e-04$	$2.34e-05$	$6.11e-05$	$1.07e-05$	$3.63e-07$
5	$3.62e-03$	$1.16e-06$	$1.59e-07$	$7.24e-08$	$5.73e-11$	$2.41e-10$
6	$1.50e-03$	$1.15e-06$	$2.95e-12$	$2.66e-15$	0	0
7	$6.18e-04$	$1.06e-12$	0			
8	$2.55e-04$	0				

Tabelle 7.5: Einschließende Verfahren

```
   x2=x1-(x0-x1)*f1/(f0-f1); f2=f(x2);
   if f1*f2 < 0
      vertausche (x0,f0) mit (x1,f1);
   end
   repeat
      f0=f0*f1/(f1+f2); x1=x2; f1=f2;
      x2=x1-(x0-x1)*f1/(f0-f1); f2=f(x2);
   until f1*f2 <= 0
   x0=x1; f0=f1; x1=x2; f1=f2;
until abs(x0-x1) < eps
```

Es wurde von King gezeigt, dass für dieses **King Verfahren** die R-Konvergenzgeschwindigkeit auf 1.710 bzw. 1.732 gehoben wird. Dieselbe Konvergenzgeschwindigkeit erreicht man, wenn man die Modifikation von King in das Verfahren von Anderson und Björck einbaut. Das entstehende Verfahren heißt **Anderson Björck King Verfahren**.

Beispiel 7.28 Wir wenden die einschließenden Verfahren auf das Problem an, die dritte Wurzel von 2 zu bestimmen. Als Startwerte verwenden wir $x_0 = 1$ und $x_1 = 2$. Tabelle 7.5 enthält die Fehler für die verschiedenen Verfahren. □

Im letzten Beispiel erreicht man die volle Genauigkeit von 16 Stellen mit dem Newton Verfahren mit dem Startwert $x_0 = 1$ nach 5 Schritten. Man beachte aber, dass jeder Newton Schritt zwei Funktionsauswertungen benötigt, die 5 Newton Schritte also vergleichbar mit 10 Pegasus Schritten sind.

7.3 Newton Verfahren für Systeme

Wir betrachten das Fixpunktproblem

$$\phi(\boldsymbol{x}) = \boldsymbol{x} \tag{7.16}$$

mit $\phi : \mathbb{R}^n \supset D \to \mathbb{R}^n$.

Ähnlich wie im reellen Fall gilt die folgende lokale Konvergenzaussage für die Fixpunktiteration

$$\boldsymbol{x}^{m+1} := \phi(\boldsymbol{x}^m). \tag{7.17}$$

Satz 7.29 (Satz von Ostrowski) *$\phi : \mathbb{R}^n \supset D \to \mathbb{R}^n$ habe einen Fixpunkt $\bar{\boldsymbol{x}} \in \overset{\circ}{D}$ und sei differenzierbar in $\bar{\boldsymbol{x}}$. Gilt dann für den Spektralradius $\rho(\phi'(\bar{\boldsymbol{x}})) = \sigma < 1$, so*

existiert eine Umgebung $U(\bar{\boldsymbol{x}}) \subset D$, so dass die Iteration (7.17) für jeden Startwert $\boldsymbol{x}^0 \in U(\bar{\boldsymbol{x}})$ (Q-linear) gegen $\bar{\boldsymbol{x}}$ konvergiert.

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 8.3. ■

Bemerkung 7.30 Ist ϕ stetig differenzierbar in $\bar{\boldsymbol{x}}$, so erhält man das Ergebnis schneller aus Korollar 7.8 Es gibt nämlich zu $\varepsilon > 0$ ein $\delta > 0$, so dass

$$\|\phi'(\boldsymbol{x})\| \leqslant \sigma + 2\varepsilon = \alpha \quad \text{für alle } \boldsymbol{x} \text{ mit } \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \leqslant \delta.$$

Damit ist nach Satz 7.4 ϕ kontrahierend auf $U(\bar{\boldsymbol{x}})$, und wegen

$$\|\bar{\boldsymbol{x}} - \phi(\bar{\boldsymbol{x}})\| = 0 \leqslant (1-q)\delta$$

sind nach Korollar 7.8 die Voraussetzungen des Fixpunktsatzes für kontrahierende Abbildungen in $U(\bar{\boldsymbol{x}})$ erfüllt. □

Ist $\phi'(\bar{\boldsymbol{x}}) = \boldsymbol{O}$, so erhält man wieder quadratische Konvergenz.

Satz 7.31 *Die Voraussetzungen von Satz 7.29 seien erfüllt, es sei ϕ zweimal stetig differenzierbar in einer Umgebung von $\bar{\boldsymbol{x}}$, und es gelte $\phi'(\bar{\boldsymbol{x}}) = \boldsymbol{O}$. Dann konvergiert das Iterationsverfahren (7.17) lokal Q-quadratisch gegen $\bar{\boldsymbol{x}}$.*

Beweis: Nach Satz 7.29 konvergiert die Folge (7.17) lokal gegen $\bar{\boldsymbol{x}}$. Durch Taylorentwicklung erhält man für $\boldsymbol{x} \in U(\bar{\boldsymbol{x}})$

$$\phi_i(\boldsymbol{x}) - \phi_i(\bar{\boldsymbol{x}}) = \frac{1}{2} \sum_{j,k=1}^{n} \frac{\partial^2 \phi_i}{\partial x_j \partial x_k}(\boldsymbol{\xi}^i)(x_j - \bar{x}_j)(x_k - \bar{x}_k),$$

und hieraus folgt mit

$$M := \frac{1}{2} \max_{i,j,k} \max_{\boldsymbol{x} \in U(\bar{\boldsymbol{x}})} \left| \frac{\partial^2 \phi_i}{\partial x_j \partial x_k}(x) \right|$$

die quadratische Konvergenz:

$$\|\boldsymbol{x}^{m+1} - \bar{\boldsymbol{x}}\|_\infty \leqslant M n^2 \|\boldsymbol{x}^m - \bar{\boldsymbol{x}}\|_\infty^2.$$

■

Wir betrachten nun das Nullstellenproblem

$$f(\boldsymbol{x}) = \boldsymbol{0} \tag{7.18}$$

mit $f : \mathbb{R}^n \supset D \to \mathbb{R}^n$.

Für den Fall $n = 1$ ist das folgende Prinzip zur Gewinnung von Iterationsverfahren bekannt: Die nichtlineare Funktion f wird im m-ten Schritt durch eine einfachere Funktion $\tilde{f}_m : \mathbb{R}^n \supset D_m \to \mathbb{R}^n$ ersetzt, die f in einer Umgebung der letzten Iterierten $\boldsymbol{x}^m$ approximiert und für die die Ersatzaufgabe $\tilde{f}_m(\boldsymbol{x}) = \boldsymbol{0}$ leicht lösbar ist. Die Lösung von $\tilde{f}_m(\boldsymbol{x}) = \boldsymbol{0}$ wird dann als neue Iterierte gewählt.

Wir betrachten nur affin lineare Approximationen

$$\tilde{f}_m(\boldsymbol{x}) = \boldsymbol{a}^m + \boldsymbol{A}_m(\boldsymbol{x} - \boldsymbol{x}^m), \ \boldsymbol{x} \in \mathbb{R}^n, \tag{7.19}$$

mit $\boldsymbol{a}^m \in \mathbb{R}^n$, $\boldsymbol{A}_m \in \mathbb{R}^{(n,n)}$.

Ist f differenzierbar, so erhält man lokal (in einer Umgebung von $\boldsymbol{x}^m$) mit den Ansatz (7.19) die beste Approximation durch

$$\tilde{f}_m(\boldsymbol{x}) = f(\boldsymbol{x}^m) + f'(\boldsymbol{x}^m)(\boldsymbol{x} - \boldsymbol{x}^m) \tag{7.20}$$

nach Definition der Ableitung.

Ist $f'(\boldsymbol{x}^m)$ regulär, so ist die $(m+1)$-te Iterierte die Lösung des linearen Gleichungssystems

$$f(\boldsymbol{x}^m) + f'(\boldsymbol{x}^m)(\boldsymbol{x} - \boldsymbol{x}^m) = \boldsymbol{0} \tag{7.21}$$

oder (obwohl man bei der numerischen Ausführung des Verfahrens niemals die Inverse berechnen wird)

$$\boldsymbol{x}^{m+1} = \boldsymbol{x}^m - f'(\boldsymbol{x}^m)^{-1} f(\boldsymbol{x}^m). \tag{7.22}$$

Das so definierte Verfahren heißt **Newton Verfahren**.

Setzt man genügend hohe Differenzierbarkeit für f voraus, so liefert Satz 7.31 die quadratische Konvergenz des Newton Verfahrens.

Satz 7.32 *$f : \mathbb{R}^n \supset D \to \mathbb{R}^n$ besitze die reguläre Nullstelle $\bar{\boldsymbol{x}} \in D$, d.h. es sei $f'(\bar{\boldsymbol{x}})$ regulär und die Ableitung f' in einer Umgebung von $\bar{\boldsymbol{x}}$ Lipschitz stetig.*

Dann existieren Zahlen $\delta > 0$ und $Q > 0$ mit $Q\delta < 1$, so dass das Newton Verfahren (7.22) für jeden Startwert $\boldsymbol{x}^0 \in S := \{\boldsymbol{x} : \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \leqslant \delta\}$ durchführbar ist, die Iterierten in S verbleiben und gegen $\bar{\boldsymbol{x}}$ konvergieren. Zudem gilt

$$\|\boldsymbol{x}^{m+1} - \bar{\boldsymbol{x}}\| \leqslant Q\|\boldsymbol{x}^m - \bar{\boldsymbol{x}}\|^2 \quad \textit{für alle } m \geqslant 0. \tag{7.23}$$

Bemerkung 7.33 Unter den Voraussetzungen von Satz 7.32 ist $\bar{\boldsymbol{x}}$ die einzige Nullstelle von f in S, denn ist $\bar{\boldsymbol{y}} \in S$ eine weitere Nullstelle, so wähle man $\boldsymbol{x}^0 := \bar{\boldsymbol{y}}$. Dann gilt $\boldsymbol{x}^m = \bar{\boldsymbol{y}}$ für alle m und daher $\bar{\boldsymbol{x}} = \bar{\boldsymbol{y}}$. Diese Überlegung zeigt, dass reguläre Nullstellen im anschaulichen Sinne isoliert sind. □

Bemerkung 7.34 Existiert $f''(\bar{\boldsymbol{x}})$ und ist $f''(\bar{\boldsymbol{x}})$ definit, d.h. $\boldsymbol{h}^T f''(\bar{\boldsymbol{x}})\boldsymbol{h} \neq 0$ für alle $\boldsymbol{h} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$, so kann man sogar zeigen, dass das Newton Verfahren von genau der Q-Ordnung 2 konvergiert, d.h. zusätzlich zu (7.23) gilt mit einem $\tilde{Q} > 0$ (vgl. Schwetlick [63], p. 98)

$$\|\boldsymbol{x}^{m+1} - \bar{\boldsymbol{x}}\| \geqslant \tilde{Q}\|\boldsymbol{x}^m - \bar{\boldsymbol{x}}\|^2 \quad \text{für alle } m \geqslant 0. \qquad \square$$

Satz 7.32 ist ein typischer lokaler Konvergenzsatz: Unter geeigneten Glattheits- und Regularitätsvoraussetzungen wird die Konvergenz eines Verfahrens für genügend gute Startwerte gesichert. Da in den Voraussetzungen die Lösung $\bar{\boldsymbol{x}}$ explizit vorkommt, lassen sie sich für eine konkrete Aufgabenstellung jedoch nicht überprüfen. Falls dies für Existenzaussagen oder Fehlerabschätzungen erforderlich ist, muss man auf semilokale Konvergenzsätze zurückgreifen, bei denen unter überprüfbaren Voraussetzungen sowohl die Existenz einer Lösung als auch die Konvergenz des Verfahrens gegen die Lösung bewiesen wird.

Der Einzugsbereich einer Nullstelle $\bar{\boldsymbol{x}}$ von f für das Newton Verfahren (d.h. die Menge aller Startwerte $\boldsymbol{x}^0$, für die das Newton Verfahren gegen $\bar{\boldsymbol{x}}$ konvergiert) ist häufig sehr klein. Er kann manchmal durch Einführung einer Dämpfung vergrößert werden.

Es sei $\boldsymbol{h}^m := f'(\boldsymbol{x}^m)^{-1} f(\boldsymbol{x}^m)$ die Verbesserung nach dem Newton Verfahren ausgehend von $\boldsymbol{x}^m$. Wir setzen

$$\boldsymbol{x}^{m+1} := \boldsymbol{x}^m - \lambda_m \boldsymbol{h}^m, \tag{7.24}$$

wobei der Dämpfungsparameter $\lambda_m \in (0,1]$ so gewählt wird, dass „die Größe des Funktionswerts f mit jedem Schritt verkleinert wird".

Die Größe von $f(\boldsymbol{x})$ messen wir mit der Testfunktion

$$t(\boldsymbol{x}) := \|f(\boldsymbol{x})\|_2^2.$$

Wir wählen also $\lambda_m \in (0,1]$ so, dass $t(\boldsymbol{x}^{m+1}) < t(\boldsymbol{x}^m)$ für alle $m \in \mathbb{N}_0$ gilt. Dass dies immer möglich ist, wenn die Nullstelle noch nicht erreicht ist, zeigt Satz 7.35

Satz 7.35 *Es sei $f : \mathbb{R}^n \supset D \to \mathbb{R}^n$ stetig differenzierbar, $\tilde{\boldsymbol{x}} \in D$ mit $f(\tilde{\boldsymbol{x}}) \neq \mathbf{0}$, $f'(\tilde{\boldsymbol{x}})$ regulär und $\boldsymbol{h} := f'(\tilde{\boldsymbol{x}})^{-1} f(\tilde{\boldsymbol{x}})$ die Verbesserung nach dem Newton Verfahren.*

Dann existiert ein $\mu > 0$, so dass

$$t(\tilde{\boldsymbol{x}} - \lambda \boldsymbol{h}) < t(\tilde{\boldsymbol{x}}) \quad \textit{für alle } \lambda \in (0, \mu].$$

Beweis: Es ist $t(\boldsymbol{x}) = f(\boldsymbol{x})^T f(\boldsymbol{x})$, und daher

$$\mathrm{grad} t(\boldsymbol{x}) = 2 f(\boldsymbol{x})^T f'(\boldsymbol{x}).$$

Es sei $\phi(\lambda) := t(\tilde{\boldsymbol{x}} - \lambda \boldsymbol{h})$. Dann ist ϕ in einer Umgebung von $\lambda = 0$ stetig differenzierbar mit

$$\phi(0) = t(\tilde{\boldsymbol{x}})$$

und

$$\phi'(\lambda) = -\mathrm{grad} t(\tilde{\boldsymbol{x}} - \lambda \boldsymbol{h}) \boldsymbol{h} = -2 f(\tilde{\boldsymbol{x}} - \lambda \boldsymbol{h})^T f'(\tilde{\boldsymbol{x}} - \lambda \boldsymbol{h}) \boldsymbol{h},$$

d.h.

$$\phi'(0) = -2 f(\tilde{\boldsymbol{x}})^T f'(\tilde{\boldsymbol{x}}) f'(\tilde{\boldsymbol{x}})^{-1} f(\tilde{\boldsymbol{x}}) = -2 \|f(\tilde{\boldsymbol{x}})\|_2^2 < 0,$$

d.h. ϕ ist in einer Umgebung von 0 streng monoton fallend. ■

Einen geeigneten Dämpfungsparameter λ_m kann man durch fortgesetztes Halbieren bestimmen. Man erhält dann das **gedämpfte Newton Verfahren**:

`Bestimme`	$\boldsymbol{h}^m \in \mathbb{R}^n$ `mit` $f'(\boldsymbol{x}^m)\boldsymbol{h}^m = f(\boldsymbol{x}^m)$
`bestimme`	$\ell := \min\{k \in \mathbb{N}_0 : t(\boldsymbol{x}^m - 2^{-k}\boldsymbol{h}^m) < t(\boldsymbol{x}^m)\}$
`setze`	$\boldsymbol{x}^{m+1} := \boldsymbol{x}^m - 2^{-\ell}\boldsymbol{h}^m$

Beispiel 7.36

$$f(\boldsymbol{x}) = \begin{pmatrix} (x_1^2 + x_2^2)(1 + 0.8x_1 + 0.6x_2) - 1 \\ (x_1^2 + x_2^2)(1 - 0.6x_1 + 0.8x_2) - 2x_1 \end{pmatrix} = \mathbf{0}.$$

Mit dem Startwert $\boldsymbol{x}^0 = (0.55, -1)^T$ erhält man mit dem Newton Verfahren die Näherungen in Tabelle 7.6. Man entfernt sich also sehr weit von der Nullstelle von f und wird zufällig im zwölften Schritt in den näheren Einzugsbereich der Nullstelle getragen.

Mit dem gedämpften Newton Verfahren erhält man die Werte aus Tabelle 7.7. Mit dem geänderten Startwert $\boldsymbol{x}^0 = (0.54, -1)^T$ für das gedämpften Newton Verfahren erhält man die Näherungen in Tabelle 7.8.

Das gedämpfte Newton Verfahren führt also nicht notwendig in eine Nullstelle von f, also in ein globales Minimum von t, sondern es kann auch (wie im obigen Fall) in einem lokalen Minimum stecken bleiben.

Das Newton Verfahren findet (nach einigem Herumirren) nach dreißig Iterationen die Nullstelle von f. □

m	x_1^m	x_2^m	$t(\boldsymbol{x}^m)$
0	0.55000000000000000	−1.00000000000000000	$1.62E+0000$
1	−14.43530223571625730	−33.68447879289723070	$2.24E+0009$
2	−9.55846899009341918	−22.65588840979738190	$1.97E+0008$
3	−6.30263512634718873	−15.30801418517524050	$1.73E+0007$
4	−4.12540490095915977	−10.41578314254021010	$1.52E+0006$
5	−2.66453732230278943	−7.16283531774792886	$1.33E+0005$
6	−1.67817129098230139	−5.00476389670139821	$1.16E+0004$
7	−1.00582442516326456	−3.57701132447667355	$1.02E+0003$
8	−0.54362763832473274	−2.63207040180621126	$9.00E+0001$
9	−0.22705037437102526	−1.99656005864760327	$8.40E+0000$
10	−0.01136811408793452	−1.53983061247833520	$9.68E-0001$
11	0.16037483148821532	−1.11595852255304619	$2.68E-0001$
12	1.29720182573823523	2.35416946826005801	$7.32E+0002$
13	0.83216662389917077	1.52437646280603078	$5.84E+0001$
14	0.53744240258463048	1.02770844932461125	$3.96E+0000$
15	0.40668900211956326	0.76355080012724792	$1.56E-0001$
16	0.38264871526056001	0.67356137691691342	$1.11E-0003$
17	0.38202607642813437	0.66409468893962469	$1.00E-0007$
18	0.38203126706979271	0.66400128301154753	$8.77E-0016$
19	0.38203126811166926	0.66400127421998055	$6.75E-0032$
20	0.38203126811166927	0.66400127421998048	$5.88E-0039$

Tabelle 7.6: Newton Verfahren, ungedämpft

m	x_1^m	x_2^m	$t(\boldsymbol{x}^m)$
0	0.55000000000000000	−1.00000000000000000	$1.62E+0000$
1	0.53536591578543334	−1.03191843632118870	$1.62E+0000$
2	0.56110003914930182	−0.97315152261970942	$1.62E+0000$
3	0.52522028267090552	−1.04907142845895020	$1.60E+0000$
4	0.58022724802414461	−0.91988676253078696	$1.59E+0000$
5	0.44186642606308465	−1.20117318695284988	$1.57E+0000$
6	0.78715923029833133	0.06828325649427959	$1.47E+0000$
7	0.64691727552902532	0.76081952209075180	$9.44E-0001$
8	0.42036801216744384	0.70268466773197415	$3.33E-0002$
9	0.38320847227569939	0.66655934000194405	$9.38E-0005$
10	0.38203275158923288	0.66400946937820756	$8.35E-0010$
11	0.38203126811259582	0.66400127429259763	$6.18E-0020$
12	0.38203126811166927	0.66400127421998048	$1.47E-0038$

Tabelle 7.7: Newton Verfahren, gedämpft, 1

m	x_1^m	x_2^m	$t(\boldsymbol{x}^m)$
0	0.54000000000000000	−1.00000000000000000	$1.544E+0000$
1	0.52226320324077548	−1.03837727037179463	$1.541E+0000$
2	0.54674729383789015	−0.98233280952907319	$1.536E+0000$
3	0.51049823387024609	−1.05912654060680413	$1.526E+0000$
4	0.55730345511559661	−0.94800525500289117	$1.509E+0000$
5	0.47836282966206385	−1.10979544203828296	$1.471E+0000$
6	0.61791730287807408	−0.73956641616568136	$1.443E+0000$
7	0.34190731543662328	−1.31095184668199709	$1.333E+0000$
8	0.47291974083537622	−0.57110672040690209	$8.306E-0001$
9	0.23567800090906323	−1.21822240324259060	$5.097E-0001$
10	0.32376926330177793	−0.93765575143268807	$4.505E-0001$
11	0.24244040787039246	−1.13335030418307687	$4.038E-0001$
12	0.27401919832012331	−1.04385654817449567	$3.955E-0001$
13	0.25802477791587531	−1.08563557440662883	$3.925E-0001$
14	0.26512517724780759	−1.06641164648454920	$3.925E-0001$
15	0.25920807677376168	−1.08216418620516801	$3.922E-0001$
16	0.26221776870739162	−1.07404006558494690	$3.921E-0001$
17	0.25966889237959624	−1.08087102592776657	$3.921E-0001$

Tabelle 7.8: Newton Verfahren, gedämpft, 2

Das Newton Verfahren ist sehr aufwendig. In jedem Schritt hat man die Jacobimatrix $f'(\boldsymbol{x}^m)$ und die n-Vektorfunktion $f(\boldsymbol{x}^m)$, also $n(n+1)$ reelle Funktionen, auszuwerten. Der Aufwand wird wesentlich kleiner, wenn man iteriert nach

$$\boldsymbol{x}^{m+1} = \boldsymbol{x}^m - f'(\boldsymbol{x}^0)^{-1} f(\boldsymbol{x}^m). \tag{7.25}$$

Dies ist das **vereinfachte Newton Verfahren** . Die Jacobimatrix wird also nur im ersten Schritt berechnet und in den folgenden Schritten unverändert übernommen. Man hat in (7.25) eine Folge von linearen Gleichungssystemen mit sich ändernden rechten Seiten, aber derselben Koeffizientenmatrix $f'(\boldsymbol{x}^0)$ zu lösen, was man sehr effizient mit der LR Zerlegung oder QR Zerlegung tun kann.

Ist f stetig differenzierbar in einer Umgebung einer Nullstelle $\bar{\boldsymbol{x}}$ von f und ist $f'(\bar{\boldsymbol{x}})$ regulär, so ist $f'(\boldsymbol{x}^0)$ auch für alle genügend nahe bei $\bar{\boldsymbol{x}}$ liegenden $\boldsymbol{x}^0$ regulär, und die Iterationsfunktion des vereinfachten Newton Verfahrens

$$\phi(\boldsymbol{x}) = \boldsymbol{x} - f'(\boldsymbol{x}^0)^{-1} f(\boldsymbol{x})$$

ist definiert. Wörtlich wie im eindimensionalen Fall kann man hiermit die lokale Konvergenz des vereinfachten Newton Verfahrens zeigen. Diese ist jedoch nur linear.

7.4 Quasi-Newton Verfahren

Der größte Nachteil des Newton Verfahrens ist, dass in jedem Schritt $n(n+1)$ reelle Funktionen ausgewertet werden müssen. Wir wollen in diesem Abschnitt Verfahren besprechen, die mit n Funktionsauswertungen auskommen (mit weniger wird es kaum gehen!) und ebenfalls superlinear konvergieren.

Es seien eine Näherung $\boldsymbol{x}^m$ mit $f(\boldsymbol{x}^m) \neq \boldsymbol{0}$ für eine reguläre Nullstelle $\bar{\boldsymbol{x}}$ von f und eine reguläre Approximation $\boldsymbol{B}_m$ für $f'(\boldsymbol{x}^m)$ bekannt. Wir bestimmen dann eine neue Näherung für $\bar{\boldsymbol{x}}$ durch

$$\boldsymbol{x}^{m+1} = \boldsymbol{x}^m - \boldsymbol{B}_m^{-1} f(\boldsymbol{x}^m). \tag{7.26}$$

Da $\boldsymbol{B}_m$ regulär ist und $f(\boldsymbol{x}^m) \neq \boldsymbol{0}$, ist die Änderung

$$\boldsymbol{s}^m := \boldsymbol{x}^{m+1} - \boldsymbol{x}^m \tag{7.27}$$

von $\boldsymbol{0}$ verschieden. Die neue Approximation $\boldsymbol{B}_{m+1}$ für $f'(\boldsymbol{x}^{m+1})$ soll dann so bestimmt werden, dass gilt

$$\boldsymbol{B}_{m+1}(\boldsymbol{x}^{m+1} - \boldsymbol{x}^m) = f(\boldsymbol{x}^{m+1}) - f(\boldsymbol{x}^m). \tag{7.28}$$

Dabei soll nur $\boldsymbol{B}_m$, die im gerade ausgeführten Schritt eingetretene Änderung $\boldsymbol{s}^m$ der Argumente und die zugehörige Änderung der Funktionswerte

$$f(\boldsymbol{x}^{m+1}) - f(\boldsymbol{x}^m) =: \boldsymbol{y}^m \tag{7.29}$$

verwendet werden. Insbesondere sollen keine zusätzlichen Funktionsauswertungen benutzt werden.

Wir suchen also $\boldsymbol{B}_{m+1}$ in der Form

$$\boldsymbol{B}_{m+1} = \Phi(\boldsymbol{B}_m, \boldsymbol{s}^m, \boldsymbol{y}^m) \tag{7.30}$$

mit einer **Aufdatierungsfunktion**

$$\Phi : \mathbb{R}^{(n,n)} \times \mathbb{R}^n \times \mathbb{R}^n \supset D_\Phi \to \mathbb{R}^{(n,n)}.$$

Definition 7.37 Die Vorschrift (7.30) heißt **Aufdatierungsformel** . Verfahren der Gestalt (7.26), (7.30), die der Bedingung (7.28) genügen, heißen **Quasi-Newton Verfahren**.

$\boldsymbol{B}_{m+1}$ ist durch die Bedingung (7.28) (außer im Falle $n = 1$, in dem man das Sekantenverfahren erhält) nicht eindeutig festgelegt. Die zu (7.28) äquivalente Bedingung

$$(\boldsymbol{B}_{m+1} - \boldsymbol{B}_m)\boldsymbol{s}^m = \boldsymbol{y}^m - \boldsymbol{B}_m\boldsymbol{s}^m \quad (= f(\boldsymbol{x}^{m+1}) \neq \boldsymbol{0}) \tag{7.31}$$

zeigt, dass durch sie die Änderung von $\boldsymbol{B}_m$ nur in bezug auf die Richtung $\boldsymbol{s}^m$ bestimmt wird.

Wir fordern zusätzlich, dass die Matrizen $\boldsymbol{B}_{m+1}$ hinsichtlich ihrer Wirkung auf zu $\boldsymbol{s}^m$ orthogonale Richtungen unverändert bleiben:

$$(\boldsymbol{B}_{m+1} - \boldsymbol{B}_m)\boldsymbol{s} = \boldsymbol{0} \quad \text{für alle } \boldsymbol{s} \in \mathbb{R}^n \text{ mit } \boldsymbol{s}^T\boldsymbol{s}^m = 0 \tag{7.32}$$

Wegen (7.32) besitzt die Matrix $\boldsymbol{B}_{m+1} - \boldsymbol{B}_m$ den Rang 1, ist also von der Gestalt

$$\boldsymbol{B}_{m+1} - \boldsymbol{B}_m = \boldsymbol{u}^m(\boldsymbol{v}^m)^T, \quad \boldsymbol{u}^m \neq \boldsymbol{0},\ \boldsymbol{v}^m \neq \boldsymbol{0}.$$

Aus (7.32) folgt

$$(\boldsymbol{B}_{m+1} - \boldsymbol{B}_m)\boldsymbol{s} = \boldsymbol{u}^m(\boldsymbol{v}^m)^T\boldsymbol{s} = \boldsymbol{0} \quad \text{für alle } \boldsymbol{s} \text{ mit } \boldsymbol{s} \perp \boldsymbol{s}^m$$

d.h.

$$(\boldsymbol{v}^m)^T\boldsymbol{s} = 0 \quad \text{für alle } \boldsymbol{s} \text{ mit } \boldsymbol{s} \perp \boldsymbol{s}^m$$

und daher ohne Beschränkung der Allgemeinheit $\boldsymbol{v}^m = \boldsymbol{s}^m$.

Die Quasi-Newton Gleichung (7.31) liefert

$$(\boldsymbol{B}_{m+1} - \boldsymbol{B}_m)\boldsymbol{s}^m = \boldsymbol{u}^m(\boldsymbol{s}^m)^T\boldsymbol{s}^m = \boldsymbol{y}^m - \boldsymbol{B}_m\boldsymbol{s}^m,$$

d.h.

$$\boldsymbol{u}^m = \frac{1}{\|\boldsymbol{s}^m\|_2^2}\,(\boldsymbol{y}^m - \boldsymbol{B}_m\boldsymbol{s}^m),$$

und es ist

$$\boldsymbol{B}_{m+1} = \boldsymbol{B}_m + \frac{1}{\|\boldsymbol{s}^m\|_2^2}\,(\boldsymbol{y}^m - \boldsymbol{B}_m\boldsymbol{s}^m)(\boldsymbol{s}^m)^T. \tag{7.33}$$

Die zu (7.33) gehörige Aufdatierungsfunktion

$$\Phi(\boldsymbol{B}, \boldsymbol{s}, \boldsymbol{y}) = \boldsymbol{B} + \frac{1}{\|\boldsymbol{s}\|_2^2}(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{s})\boldsymbol{s}^T$$

ist auf der Menge

$$D_\Phi := \{(\boldsymbol{A}, \boldsymbol{s}, \boldsymbol{y}) \,:\, \boldsymbol{A} \in \mathbb{R}^{(n,n)}, \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{s} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}\}$$

definiert.

Definition 7.38 Das durch (7.33) gegebene Quasi-Newton Verfahren heißt **Broyden Rang-1-Verfahren**

Die Lösung des linearen Gleichungssystems $\boldsymbol{B}_m\boldsymbol{s}^m = -f(\boldsymbol{x}^m)$ in m-ten Schritt des Broyden Rang-1-Verfahrens kann vermieden werden, denn es gilt

Lemma 7.39 *Seien $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ und $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ regulär. Dann ist die Rang 1 Modifikation $\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T$ von $\boldsymbol{A}$ genau dann regulär, wenn $\sigma := 1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u} \neq 0$.*

Ist $\sigma \neq 0$, so gilt

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} - \frac{1}{\sigma}\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}. \tag{7.34}$$

Definition 7.40 Die Aufdatierungsformel (7.34) für die Inverse einer Rang-1-Modifikation einer regulären Matrix heißt **Sherman Morrison Formel**.

Beweis: Für $\sigma \neq 0$ ist die rechte Seite von (7.34) definiert und man rechnet leicht nach, dass

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)(\boldsymbol{A}^{-1} - \frac{1}{\sigma}\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}) = \boldsymbol{I}$$

gilt.

Für $\sigma = 0$ ist $\boldsymbol{z} := \boldsymbol{A}^{-1}\boldsymbol{u} \neq \boldsymbol{0}$ und $(\boldsymbol{A}+\boldsymbol{u}\boldsymbol{v}^T)\boldsymbol{z} = \boldsymbol{0}$, d.h. $\boldsymbol{A}+\boldsymbol{u}\boldsymbol{v}^T$ ist singulär. ■

Es sei $\boldsymbol{H}_m := \boldsymbol{B}_m^{-1}$ und $\boldsymbol{B}_{m+1}$ die mit dem Broyden Rang-1-Verfahren aufdatierte Matrix. Dann ist $\boldsymbol{H}_{m+1} := \boldsymbol{B}_{m+1}^{-1}$ nach Lemma 7.39 genau dann definiert, wenn $(\boldsymbol{s}^m)^T\boldsymbol{H}_m\boldsymbol{y}^m \neq 0$ gilt, und

$$\boldsymbol{H}_{m+1} = \boldsymbol{H}_m + \frac{1}{(\boldsymbol{s}^m)^T\boldsymbol{H}_m\boldsymbol{y}^m} \cdot (\boldsymbol{s}^m - \boldsymbol{H}_m\boldsymbol{y}^m)(\boldsymbol{s}^m)^T\boldsymbol{H}_m \tag{7.35}$$

(wir wählen nämlich $\boldsymbol{u} = \boldsymbol{y}^m - \boldsymbol{B}_m\boldsymbol{s}^m$ und $\boldsymbol{v} = \boldsymbol{s}^m/\|\boldsymbol{s}^m\|_2^2$).

Hiermit kann man das Broyden Verfahren schreiben als

$$\boldsymbol{x}^{m+1} = \boldsymbol{x}^m - \boldsymbol{H}_m f(\boldsymbol{x}^m), \tag{7.36}$$

und man benötigt offenbar in jedem Schritt sowohl für die Aufdatierung von $\boldsymbol{H}_{m+1}$ als auch für die Berechnung der neuen Näherung $\boldsymbol{x}^{m+1}$ nur $O(n^2)$ Operationen neben den n Funktionsauswertungen.

m	x_1^m	x_2^m	$t(\boldsymbol{x}^m)$
0	$0.550000000000000e+00$	$-1.000000000000000e+00$	$1.62E+00$
1	$4.559000000000002e-01$	$2.693250000000003e-01$	$7.47e-01$
2	$1.543981538116330e+00$	$1.501304221151375e+00$	$1.91e+02$
3	$-1.025397030327286e-01$	$1.001738027581978e+00$	$4.68e+00$
4	$1.155418539286380e+00$	$-4.749815441829222e-01$	$8.31e+00$
5	$3.598779288346202e-01$	$2.769192432357461e-01$	$7.53e-01$
6	$3.059435797761894e-01$	$5.139683793110996e-01$	$2.27e-01$
7	$3.168501618408199e-01$	$6.429750746281397e-01$	$2.70e-02$
8	$3.428381587159949e-01$	$6.199750293193117e-01$	$3.16e-02$
9	$3.593134438436643e-01$	$6.428635461736542e-01$	$8.76e-03$
10	$3.823317197048445e-01$	$6.654991281621061e-01$	$2.82e-05$
11	$3.819657986428381e-01$	$6.638948931722465e-01$	$1.76e-07$
12	$3.820297255805328e-01$	$6.639992049514174e-01$	$7.24e-11$
13	$3.820312595808607e-01$	$6.640012640813945e-01$	$1.85e-15$
14	$3.820312680827534e-01$	$6.640012741848831e-01$	$2.19e-20$
15	$3.820312681116868e-01$	$6.640012742200018e-01$	$8.07e-27$
16	$3.820312681116693e-01$	$6.640012742199805e-01$	$1.23e-32$

Tabelle 7.9: Broyden Verfahren

Satz 7.41 *Es sei $\bar{\boldsymbol{x}}$ eine reguläre Nullstelle von f. Dann konvergiert das Broyden-Verfahren lokal Q-superlinear gegen $\bar{\boldsymbol{x}}$.*

Beweis: Siehe dazu das Skript „Grundlagen der Numerischen Mathematik“ von Heinrich Voß, Abschnitt 8.6. ■

Beispiel 7.42 Wir betrachten erneut Beispiel 7.36

$$f(\boldsymbol{x}) = \begin{pmatrix} (x_1^2 + x_2^2)\,(1 + 0.8x_1 + 0.6x_2) - 1 \\ (x_1^2 + x_2^2)\,(1 - 0.6x_1 + 0.8x_2) - 2x_1 \end{pmatrix} = \boldsymbol{0}.$$

Mit dem Startwert $\boldsymbol{x}^0 = (0.55, -1)^T$ und $\boldsymbol{B} = \boldsymbol{E}_2$ erhält man die Näherungen in Tabelle 7.9. $t(\boldsymbol{x}^m)$ bezeichnet wie vorher $t(\boldsymbol{x}^m) = \|f(\boldsymbol{x}^m)\|_2^2$. □

7.5 Nichtlineare Ausgleichsprobleme

Wir betrachten das **nichtlineare Ausgleichsproblem** .

> *Es seien $f : \mathbb{R}^n \supset D \to \mathbb{R}^m$ und $\boldsymbol{y} \in \mathbb{R}^m$ gegeben mit $m \geqslant n$. Man bestimme $\boldsymbol{x} \in \mathbb{R}^n$, so dass*
>
> $$\|f(\boldsymbol{x}) - \boldsymbol{y}\|_2 \tag{7.37}$$
>
> *minimal wird.*

Es sei $\boldsymbol{x}^0$ eine Näherung für ein Minimum. Wir linearisieren f an der Stelle $\boldsymbol{x}^0$, ersetzen also $f(\boldsymbol{x})$ durch

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}^0) + \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{x}^0),$$

wobei $\boldsymbol{J} := f'(\boldsymbol{x}^0)$ die Jacobi Matrix von f an der Stelle $\boldsymbol{x}^0$ bezeichnet.

Setzt man $\tilde{f}$ für f ein, so erhält man das lineare Ausgleichsproblem

Bestimme $\boldsymbol{\xi} \in \mathbb{R}^n$, so dass

$$\|\boldsymbol{J}\boldsymbol{\xi} - \boldsymbol{r}\|_2 \tag{7.38}$$

minimal ist. Dabei bezeichnet $\boldsymbol{r} := \boldsymbol{y} - f(\boldsymbol{x}^0)$ das Residuum im Punkt $\boldsymbol{x}^0$.

Dann ist $\boldsymbol{x}^1 := \boldsymbol{x}^0 + \boldsymbol{\xi}$ i.a. keine bessere Näherung für eine Lösung des nichtlinearen Ausgleichsproblems als $\boldsymbol{x}^0$, aber es gilt Satz 7.43

Satz 7.43 *Es seien $\boldsymbol{x}^0 \in \mathbb{R}^n$ mit Rang$\boldsymbol{J} = n$, $\boldsymbol{r} \neq \boldsymbol{0}$ und $\boldsymbol{\xi} \in \mathbb{R}^n$ die Lösung des linearen Ausgleichsproblems*

$$\|\boldsymbol{J}\boldsymbol{\xi} - \boldsymbol{r}\|_2 = \min. \tag{7.39}$$

Ist $\boldsymbol{\xi} \neq \boldsymbol{0}$, so existiert $\bar{t} > 0$, so dass

$$\phi(t) := \|\boldsymbol{y} - f(\boldsymbol{x}^0 + t\boldsymbol{\xi})\|_2^2, t \in (0, \bar{t})$$

streng monoton fällt, d.h. $\boldsymbol{\xi}$ ist eine Abstiegsrichtung für $\|f(\boldsymbol{x}) - \boldsymbol{y}\|_2$ in $\boldsymbol{x}^0$.

Beweis: ϕ ist stetig differenzierbar und

$$\begin{aligned}\phi'(0) &= \frac{d}{dt}\Big\{\Big(\boldsymbol{y} - f(\boldsymbol{x}^0 + t\boldsymbol{\xi})\Big)^T\Big(\boldsymbol{y} - f(\boldsymbol{x}^0 + t\boldsymbol{\xi})\Big)\Big\}\Big|_{t=0} \\ &= -2(\boldsymbol{J}\boldsymbol{\xi})^T(\boldsymbol{y} - f(\boldsymbol{x}^0)).\end{aligned}$$

Da $\boldsymbol{\xi}$ das lineare Ausgleichsproblem löst, ist $\boldsymbol{\xi}$ auch Lösung der zugehörigen Normalgleichungen

$$\boldsymbol{J}^T\boldsymbol{J}\boldsymbol{\xi} = \boldsymbol{J}^T\boldsymbol{r},$$

d.h.

$$\phi'(0) = -2\boldsymbol{\xi}^T\boldsymbol{J}^T\boldsymbol{r} = -2\boldsymbol{\xi}^T\boldsymbol{J}^T\boldsymbol{J}\boldsymbol{\xi} = -2\|\boldsymbol{J}\boldsymbol{\xi}\|_2^2 < 0.$$

■

Dieses Ergebnis legt nun nahe, in Richtung der Lösung des linearisierten Ausgleichsproblems fortzuschreiten und ähnlich wie beim gedämpften Newton Verfahren die Schrittweite durch fortgesetzte Halbierung so klein zu wählen, dass die Norm in (7.37) verkleinert wird. Wiederholt man diesen Schritt, so erhält man das **Gauß Newton Verfahren**, genauer das **gedämpfte Gauß Newton Verfahren**

Algorithmus 7.44 (Gedämpftes Gauß Newton Verfahren)
For $i = 1, 2, \ldots$ until convergence do
Berechne die Lösung $\boldsymbol{\xi}^i$ des linearen Ausgleichsproblems

$$\|f'(\boldsymbol{x}^{i-1})\boldsymbol{\xi} - (\boldsymbol{y} - f(\boldsymbol{x}^{i-1}))\|_2 = \min$$

Bestimme das minimale $\ell \in \mathbb{N}_0$ mit

$$\|\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}^{i-1} + 2^{-\ell}\boldsymbol{\xi}^i)\|_2^2 < \|\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}^{i-1})\|_2^2$$

Setze $\boldsymbol{x}^i := \boldsymbol{x}^{i-1} + 2^{-\ell}\boldsymbol{\xi}^i$.

Ohne Dämpfung, d.h. für $\ell = 0$ in jedem Schritt, wurde dieses Verfahren von Gauß benutzt, um Bahnen von Planetoiden vorherzusagen.

m	x_0^m	y_0^m	r^m
0	10.00000000000000	0.00000000000000	5.00000000000000
1	6.14876214033877	2.15725426511744	8.04036508586118
2	3.90540362558127	3.86325842696234	1.23334135895049
3	3.30542254249197	3.27561123666543	1.00518648399760
4	2.01445622380190	1.88326627386490	2.37615193167457
5	1.28012769560372	1.20757631375102	3.63420894517193
6	1.05146620440749	1.00051076514201	3.94851196633348
7	1.03357181946508	0.98450202132343	3.97248530883056
8	1.03339414968563	0.98435519042950	3.97270001636676
9	1.03339325417899	0.98435449459529	3.97270097982488
10	1.03339324956150	0.98435449093551	3.97270098483765
11	1.03339324955561	0.98435449093087	3.97270098484402
12	1.03339324955506	0.98435449093043	3.97270098484462
13	1.03339324955506	0.98435449093043	3.97270098484462

Tabelle 7.10: Gauß Newton Verfahren

Beispiel 7.45 Gegeben seien die Punkte

x_j	1	1.5	2	2.5	3	3.5	4	4.5	5
y_j	5	4.9	4.8	4.7	4.4	4.1	3.7	2.9	1

in der Ebene. Man bestimme durch Ausgleich einen Kreis

$$K = \{(x,y)^T \,:\, \sqrt{(x-x_0)^2+(y-y_0)^2} = r\},$$

der diesen Punkten möglichst nahe ist, d.h. mit

$$f_i(x_0,y_0,r) := \sqrt{(x_i-x_0)^2+(y_i-y_0)^2} - r, i = 1,\ldots,9$$

löse man das nichtlineare Ausgleichsproblem

$$\|(f_i(x_0,y_0,r))_{i=1,\ldots,9}\|_2 = \min$$

Mit dem Gauß Newton Verfahren und den (unsinnigen) Startwerten $x_0^0 := 10$, $y_0^0 = 0$, $r_0^0 := 5$ erhält man die Näherungen in Tabelle 7.10. □

Das Gauß Newton Verfahren ist eine Liniensuchmethode. Zur Näherung $\boldsymbol{x}^0$ wird mit dem linearen Ausgleichsproblem (7.39) eine Abstiegsrichtung $\boldsymbol{\xi}$ für $\|\boldsymbol{y}-f(\boldsymbol{x}^0+t\boldsymbol{\xi})\|_2$ berechnet und dann hierzu durch fortgesetzte Halbierung eine geeignete Suchlänge bestimmt, so dass die Norm in (7.37) verkleinert wird.

Neben diesen Liniensuchmethoden werden in der nichtlinearen Optimierung Vertrauensbereichsmethoden (engl. trust region methods) betrachtet, bei denen die Suchrichtung und Suchlänge simultan bestimmt werden. Für nichtlineare Ausgleichsprobleme haben sie die folgende Gestalt:

> Zur Näherungslösung $\boldsymbol{x}^0$ und zum Radius $\Delta > 0$ des Vertrauensbereichs bestimme man $\boldsymbol{x} := \boldsymbol{x}^0 + \boldsymbol{\xi}$ mit
>
> $$\|\boldsymbol{\xi}\|_2 \leqslant \Delta \quad \text{und} \quad \|\boldsymbol{J}\boldsymbol{\xi} - \boldsymbol{r}\|_2 = \min \tag{7.40}$$

Ist dann

$$\|f(\boldsymbol{x}) - \boldsymbol{y}\|_2 < \|f(\boldsymbol{x}^0) - \boldsymbol{y}\|_2, \tag{7.41}$$

so war der Schritt erfolgreich. $\boldsymbol{x}$ wird als neue Näherung akzeptiert, und der Radius Δ des Vertrauensbereichs vergrößert (z.B. verdoppelt), wenn die Abnahme in (7.41)

sehr stark war. Ist (7.41) nicht erfüllt, wird der Schritt mit einem verkleinerten (z.B. halbierten) Δ wiederholt. Dieses sog. **Levenberg-Marquardt-Verfahren** wurde erstmals von Levenberg [47] und Marquardt [51] (allerdings mit einer anderen Motivation) zur Lösung von nichtlinearen Ausgleichsproblemen vorgeschlagen.

Die Kuhn-Tucker Bedingungen sagen, dass (7.40) äquivalent ist dem Problem

Bestimme $\boldsymbol{\xi}$ und $\lambda \geqslant 0$ mit

$$(\boldsymbol{J}^T\boldsymbol{J} + \lambda \boldsymbol{I})\boldsymbol{\xi} = \boldsymbol{J}^T\boldsymbol{r}, \tag{7.42}$$

$$\lambda(\Delta - \|\boldsymbol{\xi}\|_2) = 0. \tag{7.43}$$

(7.42) sind die Normalgleichungen des linearen Ausgleichsproblems

$$\left\| \begin{pmatrix} \boldsymbol{J} \\ \sqrt{\lambda}\boldsymbol{I} \end{pmatrix} \boldsymbol{\xi} - \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{0} \end{pmatrix} \right\|_2 = \min \tag{7.44}$$

so dass man das System (7.42) lösen kann, ohne die Matrix $\boldsymbol{J}^T\boldsymbol{J}$ zu berechnen. Hierzu verwendet man die QR–Zerlegung

$$\begin{pmatrix} \boldsymbol{J} \\ \sqrt{\lambda}\boldsymbol{I} \end{pmatrix} = \boldsymbol{Q}_\lambda \begin{pmatrix} \boldsymbol{R}_\lambda \\ \boldsymbol{O} \end{pmatrix}. \tag{7.45}$$

Um die Bedingung (7.43) (wenigstens näherungsweise) zu erfüllen oder wenn ein Schritt des Vertrauensbereichs Verfahrens verworfen wird, muss das Ausgleichsproblem (7.44) für verschiedene λ gelöst werden. Dies kann auf folgende Weise effizient durchgeführt werden:

Wir bestimmen zunächst mit Householder Transformationen die QR–Zerlegung

$$\boldsymbol{J} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O} \end{pmatrix}.$$

Dann gilt

$$\begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O} \\ \sqrt{\lambda}\boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}^T & \\ & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{J} \\ \sqrt{\lambda}\boldsymbol{I} \end{pmatrix}. \tag{7.46}$$

Die Matrix auf der linken Seite ist schon fast eine obere Dreiecksmatrix. Man muss nur noch die Elemente der Diagonale von $\sqrt{\lambda}\boldsymbol{I}$ annullieren. Dies kann man mit Givens Rotationen ausführen:

Wir bezeichnen die ersten n Zeilen der linken Seite von (7.46) immer mit $\boldsymbol{R}$ und die letzten n-Zeilen (die zunächst mit $\sqrt{\lambda}\boldsymbol{I}$ besetzt sind) mit $\boldsymbol{N}$. Durch Kombination der n-ten Zeile von $\boldsymbol{R}$ und der n-ten Zeile von $\boldsymbol{N}$ kann man das n-te Diagonalelement von $\boldsymbol{N}$ eliminieren. Hat man bereits die Zeilen $n, n-1, \ldots, i+1$ von $\boldsymbol{N}$ behandelt, so kombiniert man die Zeilen i von $\boldsymbol{R}$ und von $\boldsymbol{N}$ und erzeugt eine 0 in der Position (i,i) von $\boldsymbol{N}$. Hierdurch werden die Elemente an den Stellen $(i,i+1), \ldots, (i,n)$ in $\boldsymbol{N}$ aufgefüllt. Man kann sie aber nach einander durch Kombination der i-ten Zeile von $\boldsymbol{N}$ mit den Zeilen $i+1, i+2, \ldots, n$ von $\boldsymbol{R}$ anullieren.

Damit kann man für jedes λ ausgehend von (7.46) durch $n(n+1)/2$ Givens Rotationen die QR–Zerlegung (7.45) bestimmen. Dies kann insbesondere im Fall $n \ll m$ sehr viel Arbeit ersparen.

Ausgleichsprobleme sind häufig sehr schlecht skaliert. Es kommt vor, dass einige der zu berechnenden x_j die Größenordnung 10^4 haben und andere 10^{-6}. In diesem Fall kann das Levenberg-Marquardt Verfahren ein sehr schlechtes Verhalten haben. Eine Möglichkeit, der schlechten Skalierung zu begegnen, besteht darin, statt der

kugelförmigen Vertrauensbereiche ellipsoidförmige zu verwenden. Der k-te Schritt des Verfahrens erhält dann die Gestalt: Bestimme $\boldsymbol{\xi}$, so dass

$$\|\boldsymbol{D}_k\boldsymbol{\xi}\|_2 \leqslant \Delta_k \quad \text{und} \quad \|\boldsymbol{J}_k\boldsymbol{\xi} - \boldsymbol{r}_k\|_2 = \min \tag{7.47}$$

wobei $\boldsymbol{D}_k$ eine Diagonalmatrix mit positiven Diagonalelementen bezeichnet. In den Kuhn-Tucker Bedingungen ist dann (7.42) durch

$$(\boldsymbol{J}_k^T\boldsymbol{J}_k + \lambda\boldsymbol{D}_k)\boldsymbol{\xi} = \boldsymbol{J}_k^T\boldsymbol{r}_k \tag{7.48}$$

zu ersetzen, und dies ist äquivalent dem linearen Ausgleichsproblem

$$\left\| \begin{pmatrix} \boldsymbol{J}_k \\ \sqrt{\lambda}\boldsymbol{D}_k \end{pmatrix} \boldsymbol{\xi} - \begin{pmatrix} \boldsymbol{r}_k \\ \boldsymbol{0} \end{pmatrix} \right\|_2 = \min \tag{7.49}$$

Die Diagonalelemente von $\boldsymbol{D}_k$ können dabei an die Größenordnungen der Komponenten der Näherungslösung von Schritt zu Schritt angepasst werden. An der Technik, die QR–Zerlegung der Koeffizientenmatrix von (7.49) für verschiedene λ aus der von $\boldsymbol{J}_k$ mit Givens Rotationen zu berechnen, ändert sich nichts.

Kapitel 8

Einschrittverfahren

8.1 Das Eulersche Polygonzugverfahren

Wir betrachten die Anfangswertaufgabe

$$y' = f(x, y), \quad y(a) = y_0, \tag{8.1}$$

wobei die Lösung y im Intervall $[a, b]$ gesucht ist.

Dabei kann $\boldsymbol{y}$ auch vektorwertig, also (8.1) ein Differentialgleichungssystem erster Ordnung sein.

Es sei $a = x_0 < x_1 < x_2 < \cdots < x_N =: b$ eine (nicht notwendig äquidistante) Zerlegung von $[a, b]$. Wir nehmen diese Zerlegung zunächst als gegeben an. Tatsächlich wird die Folge der x_j im Verfahren mitbestimmt und an das Verhalten der Lösung der Anfangswertaufgabe angepasst.

Da $f(x_n, y(x_n))$ gerade die Steigung $y'(x_n)$ der gesuchten Lösung $y(x)$ von (8.1) ist, gilt näherungsweise bei nicht zu großer Schrittweite $h_n := x_{n+1} - x_n$

$$\frac{1}{h_n}\left(y(x_{n+1}) - y(x_n)\right) \approx f(x_n, y(x_n)),$$

d.h.

$$y(x_{n+1}) = y(x_n) + h_n\, f(x_n, y(x_n)) + \varepsilon_n. \tag{8.2}$$

Wir vernachlässigen nun in (8.2) den Fehler ε_n. Dann wird die entstehende Gleichung nicht mehr durch die Lösung $y(x_n)$ von (8.1) an den Knoten x_n erfüllt, sondern nur noch durch Näherungswerte y_n für $y(x_n)$. Wir bestimmen also die y_n ausgehend von y_0 durch das Verfahren

$$y_{n+1} = y_n + h_n\, f(x_n, y_n), \quad n = 0, 1, \ldots, N-1, \tag{8.3}$$

wobei $h_n := x_{n+1} - x_n$ ist.

Definition 8.1 Das durch (8.3) beschriebene Verfahren zur approximativen Lösung der Anfangswertaufgabe (8.1) heißt das **Eulersche Polygonzugverfahren**. Es wurde 1768 von L. Euler beschrieben.

Beispiel 8.2 Die Anfangswertaufgabe

$$y' = y^2, \; y(0.8) = \frac{5}{6}, \quad x \in [0.8, 1.8]$$

besitzt die Lösung $y(x) = \dfrac{1}{2 - x}$.

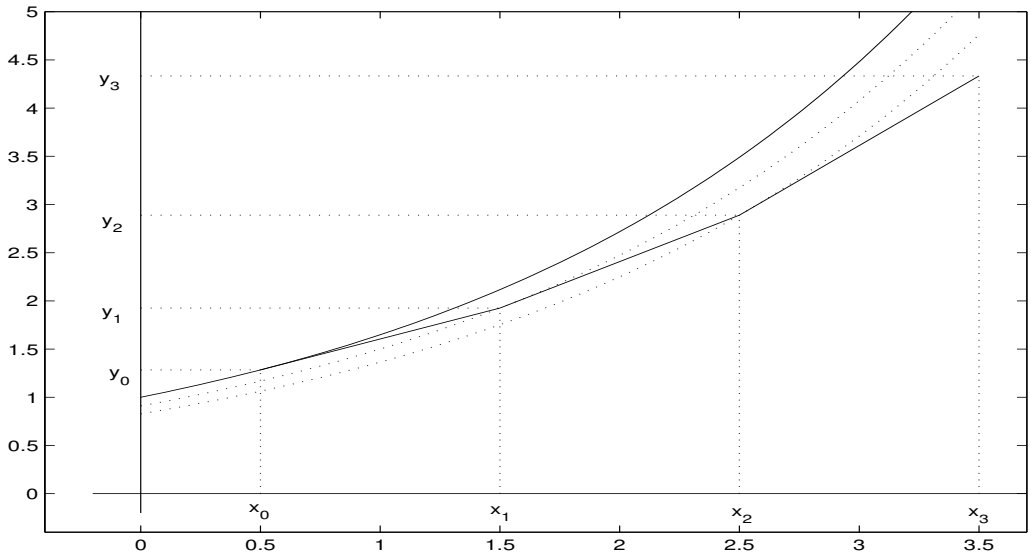

Abbildung 8.1: Knotenpunkte und Näherungswerte

x	$N = 100$	$N = 200$	$N = 400$
0.80	$0.00E+0$	$0.00E+0$	$0.00E+0$
0.90	$-7.09E-4$	$-3.57E-4$	$-1.79E-4$
1.00	$-1.79E-3$	$-9.04E-4$	$-4.54E-4$
1.10	$-3.49E-3$	$-1.76E-3$	$-8.84E-4$
1.20	$-6.20E-3$	$-3.13E-3$	$-1.57E-3$
1.30	$-1.07E-2$	$-5.43E-3$	$-2.73E-3$
1.40	$-1.86E-2$	$-9.47E-3$	$-4.77E-3$
1.50	$-3.35E-2$	$-1.71E-2$	$-8.66E-3$
1.60	$-6.48E-2$	$-3.33E-2$	$-1.69E-2$
1.70	$-1.41E-1$	$-7.38E-2$	$-3.77E-2$
1.80	$-3.90E-1$	$-2.08E-1$	$-1.08E-1$

Tabelle 8.1: Fehlertabelle von Beispiel 8.2

Mit den äquidistanten Schrittweiten $h = \frac{1}{100}, \frac{1}{200}$ und $\frac{1}{400}$ liefert das Verfahren (8.3) Näherungen, deren Fehler in der Tabelle 8.1 enthalten sind. Man liest aus der Tabelle ab, dass die Fehler bei Halbierung der Schrittweite ebenfalls halbiert werden. □

Wir wollen nun im allgemeinen Fall den entstandenen Fehler abschätzen. Dieser setzt sich aus zwei Anteilen zusammen: Wir haben im n-ten Schritt die Lösung $y(x_{n+1}; x_n, y(x_n))$ der Differentialgleichung $y' = f(x, y)$ mit dem Anfangswert $y(x_n)$ an der Stelle x_n zu bestimmen. Statt dessen betrachten wir die Anfangswertaufgabe

$$y' = f(x, y),\ y(x_n) = y_n$$

mit dem "falschen" Anfangswert y_n, und wir lösen diese auch nur näherungsweise, indem wir den Abbruchfehler vernachlässigen. Tatsächlich werden bei der Realisierung des Verfahrens auf einem Rechner noch bei der Auswertung von $f(x_n, y_n)$ und den Rechenoperationen Rundungsfehler gemacht. Diese wollen wir aber bei den folgenden Betrachtungen außer Acht lassen.

Wir schreiben das Polygonzugverfahren in der Form

$$y_{n+1} - y_n - h_n\, f(x_n, y_n) = 0. \tag{8.4}$$

Setzt man hier an Stelle der Werte y_n die Werte $y(x_n)$ der Lösung von (8.1) an den Knoten x_n ein, so erhält man (vgl. (8.2))

$$y(x_{n+1}) - y(x_n) - h_n\, f(x_n, y(x_n)) =: \varepsilon(x_n, h_n). \tag{8.5}$$

Definition 8.3 $\varepsilon(x_n, h_n)$ heißt der **lokale Fehler** (auch **Abbruchfehler**) des Polygonzugverfahrens an der Stelle x_n bei der Schrittweite h_n.

Subtrahiert man die Gleichung (8.4) von (8.5), so folgt

$$y(x_{n+1}) - y_{n+1} = y(x_n) - y_n + h_n\,(f(x_n, y(x_n)) - f(x_n, y_n)) + \varepsilon(x_n, h_n). \tag{8.6}$$

Definition 8.4

$$\delta_{n+1} := |y(x_{n+1}) - y_{n+1}|$$

heißt der **Fehler** oder (zur besseren Unterscheidung) **globale Fehler** des Polygonzugverfahrens an der Stelle x_{n+1}.

Um den globalen Fehler abschätzen zu können, setzen wir voraus, dass f auf $[a,b]\times\mathbb{R}$ einer globalen Lipschitz Bedingung

$$|f(x,y) - f(x,z)| \leqslant L|y-z| \quad \text{für alle } y, z \in \mathbb{R} \text{ und alle } x \in [a,b]$$

bzgl. y genügt. Eine Lipschitz Bedingung in einem Rechteck (wie in dem Satz von Picard-Lindelöf) würde auch genügen. Man müsste dann nur aufpassen, dass die Näherungslösungen dieses Rechteck nicht verlassen.

Dann folgt aus (8.6) mit der Dreiecksungleichung und $\varepsilon_n := \varepsilon(x_n, h_n)$

$$\delta_{n+1} \leqslant (1 + Lh_n)\,\delta_n + |\varepsilon_n|, \tag{8.7}$$

und durch vollständige Induktion erhält man hieraus

$$\delta_n \leqslant \left(\delta_0 + \sum_{j=0}^{n-1} |\varepsilon_j|\right) \cdot \exp\left(\sum_{j=0}^{n-1} h_j\, L\right), \tag{8.8}$$

denn für $n = 0$ ist diese Aussage trivial, und ist (8.8) für ein $n < N$ erfüllt, so folgt

$$\begin{aligned}
\delta_{n+1} &\leqslant (1 + Lh_n)\,\delta_n + |\varepsilon_n| \leqslant \exp(Lh_n)\,\delta_n + |\varepsilon_n| \\
&\leqslant \exp(Lh_n) \cdot \left(\delta_0 + \sum_{j=0}^{n-1} |\varepsilon_j|\right) \cdot \exp\left(\sum_{j=0}^{n-1} h_j\, L\right) + |\varepsilon_n| \\
&= \left(\delta_0 + \sum_{j=0}^{n-1} |\varepsilon_j|\right) \cdot \exp\left(\sum_{j=0}^{n} h_j\, L\right) + |\varepsilon_n| \\
&\leqslant \left(\delta_0 + \sum_{j=0}^{n} |\varepsilon_j|\right) \cdot \exp\left(\sum_{j=0}^{n} h_j\, L\right)
\end{aligned}$$

Es ist

$$\sum_{j=0}^{n} h_j \leqslant \sum_{j=0}^{N-1} h_j = b - a$$

und $\delta_0 = |y(a) - y_0| = 0$. Daher folgt

$$\delta_n \leqslant e^{(b-a)L} \sum_{j=0}^{N-1} |\varepsilon_j|.$$

Bis auf eine multiplikative Konstante lässt sich der globale Fehler also durch die Summe der lokalen Fehler abschätzen.

Wir setzen nun weiter voraus, dass die Lösung y von (8.1) zweimal stetig differenzierbar ist. Dies ist z.B. erfüllt, wenn die rechte Seite f stetig differenzierbar in einer offenen Menge ist, die den Graphen $\{(x, y(x)) : a \leqslant x \leqslant b\}$ enthält. Dann gilt nach dem Taylorschen Satz für den lokalen Fehler

$$\varepsilon_j = y(x_j + h_j) - y(x_j) - h_j y'(x_j) = \frac{1}{2} h_j^2 y''(x_j + \theta_j h_j)$$

mit einem $\theta_j \in (0, 1)$, und daher folgt für den globalen Fehler

$$\begin{aligned}
\delta_n &\leqslant \frac{1}{2} e^{(b-a)L} \sum_{j=0}^{N-1} h_j^2 |y''(x_j + \theta_j h_j)| \\
&\leqslant \frac{1}{2}\, e^{(b-a)L} \max_{a \leqslant x \leqslant b} |y''(x)| \sum_{j=0}^{N-1} h_j^2 \\
&\leqslant \frac{1}{2}\, e^{(b-a)L} \max_{a \leqslant x \leqslant b} |y''(x)| \max_{j=0,\ldots,N-1} h_j \sum_{j=0}^{N-1} h_j \\
&\leqslant \frac{1}{2}\, (b-a)\, e^{(b-a)L} \max_{a \leqslant x \leqslant b} |y''(x)| \max_{j=0,\ldots,N-1} h_j \\
&=: C \cdot \max_{j=0,\ldots,N-1} h_j
\end{aligned}$$

mit einer von den gewählten Schrittweiten h_n unabhängigen Konstante C.

Hieraus liest man ab, dass bei Halbierung der Schrittweiten der Fehler ebenfalls halbiert wird (vgl. Beispiel 8.2). Man liest ferner ab, dass man $O(\delta^{-1})$ Schritte des Polygonzugverfahrens aufwenden muss, um den globalen Fehler δ zu erreichen (für den Fehler $\delta = 10^{-6}$ also $c \cdot 10^6$ Schritte). Dies zeigt, dass das Eulersche Polygonzugverfahren für die Praxis nicht geeignet ist und dass man schnellere Verfahren benötigt.

Wie im Falle der Quadratur wird man in der Praxis nicht mit vorgegebenen Schrittweiten rechnen, sondern die Schrittweite dem Lösungsverhalten anpassen. Dabei schätzt man wie bei den adaptiven Quadraturformeln den lokalen Fehler mit Hilfe einer zweiten Formel.

Wir verwenden hierzu zwei Schritte des Polygonzugverfahrens mit halber Schrittweite:

$$\begin{aligned}
\tilde{y}_{n+\frac{1}{2}} &= y_n + \frac{h_n}{2} f(x_n, y_n) \\
\tilde{y}_{n+1} &= \tilde{y}_{n+\frac{1}{2}} + \frac{h_n}{2} f(x_n + \frac{h_n}{2}, \tilde{y}_{n+\frac{1}{2}}) \\
&= y_n + \frac{h_n}{2} f(x_n, y_n) + \frac{h_n}{2} f(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} f(x_n, y_n)).
\end{aligned}$$

Für den lokalen Fehler gilt mit der Lösung $z(x)$ der Anfangswertaufgabe $y' = f(x,y)$, $y(x_n) = y_n$ (im Falle $z \in C^3[a,b]$) nach dem Taylorschen Satz

$$\begin{aligned}\varepsilon(x_n,h_n) &= z(x_n+h_n) - (y_n + h_n\, f(x_n,y_n)) \\ &= z(x_n) + h_n\, z'(x_n) + \frac{1}{2}\, h_n^2 z''(x_n) + O(h_n^3) - z(x_n) - h_n\, z'(x_n) \\ &= \frac{1}{2}\, h_n^2 z''(x_n) + O(h_n^3) \end{aligned} \tag{8.9}$$

und genauso für die zusammengesetzte Formel

$$\begin{aligned}&\tilde{\varepsilon}(x_n,h_n) = z(x_n+h_n) - \tilde{y}_{n+1} \\ &= y_n + h_n\, f(x_n,y_n) + \frac{1}{2}\, h_n^2 z''(x_n) + O(h_n^3) - y_n - \frac{h_n}{2}\, f(x_n,y_n) \\ &- \frac{h_n}{2}(f(x_n,y_n) + \frac{h_n}{2}\,\frac{\partial}{\partial x}\, f(x_n,y_n) + \frac{h_n}{2}\,\frac{\partial}{\partial y}\, f(x_n,y_n) f(x_n,y_n) + O(h_n^2)) \\ &= \frac{1}{4}\, h_n^2 z''(x_n) + O(h_n^3)\end{aligned}$$

wegen

$$z''(x) = \frac{d}{dx}\, f(x,z(x)) = \frac{\partial}{\partial x}\, f(x,z(x)) + \frac{\partial}{\partial y}\, f(x,z(x))\, z'(x).$$

Durch Subtraktion dieser beiden Formeln erhält man

$$\tilde{y}_{n+1} - y_{n+1} = \frac{1}{4}\, h_n^2 z''(x_n) + O(h_n^3).$$

Setzt man dies in (8.9) unter Vernachlässigung des $O(h_n^3)$-Terms ein, so erhält man die Schätzung für den lokalen Fehler

$$\varepsilon(x_n,h_n) \approx \phi(x_n,h_n) := 2(\tilde{y}_{n+1} - y_{n+1}). \tag{8.10}$$

Zugleich erhält man mit

$$\begin{aligned}\hat{y}_{n+1} &:= 2\tilde{y}_{n+1} - y_{n+1} \\ &= y_n + h_n f(x_n + \frac{1}{2} h_n, y_n + \frac{h_n}{2} f(x_n,y_n))\end{aligned} \tag{8.11}$$

eine Näherung für $y(x_n+h_n)$ mit dem lokalen Fehler

$$\hat{\varepsilon}(x_n,h_n) = 2\,\tilde{\varepsilon}(x_n,h_n) - \varepsilon(x_n,h_n) = O(h_n^3).$$

Verfahren mit dieser Eigenschaft werden wir Verfahren der Ordnung 2 nennen.

Die Formel (8.10) verwenden wir nun zur **Schrittweitensteuerung**:
Wir geben uns eine Toleranz $\tau > 0$ vor und bestimmen die Schrittweite in jedem Schritt so, dass

$$\text{lokaler Fehler } \approx \tau \tag{8.12}$$

gilt.

Approximieren wir $\varepsilon(x_n,h)$ durch $\varepsilon(x_n,h) \approx \gamma h^2$, so kann man γ durch einen Probeschritt der Länge H schätzen:

$$\gamma \approx \frac{1}{H^2}\, \varepsilon(x_n,H).$$

Die optimale Wahl der Schrittweite wäre nach (8.12)

$$\tau = |\varepsilon(x_n,h)| \approx |\gamma| \cdot h^2 \approx \frac{h^2}{H^2}\, |\varepsilon(x_n,H)|,$$

d.h.

$$h = H\sqrt{\frac{\tau}{|\varepsilon(x_n, H)|}}.$$

Der folgende MATLAB-Programmteil verwendet eine ähnliche Schrittweitenkontrolle bei gegebenen Startwerten x und y und gegebener Probeschrittlänge h:

```
v=1.e-5*ones(1,n);
z = f(x,y);
while h>0
  y1 = y + h*z;
  y2 = y + h/2*z;
  y2 = y2 + h/2 * f(x+h/2,y2);
  d=max(v,max(abs(y),abs(y1)));
  phi = 2 * norm((y2 - y1)./d);
  hneu = h * min(max(0.9*sqrt(tol/phi),0.2),10);
  if phi > tol
    h = hneu;
  else
    x = x + h;
    y = 2*y2 - y1;                (*)
    z = f(x,y);
    h = min(b-x,hneu);
    end
  end
```

Bemerkung 8.5 Es wurde der absolute Fehler durch den "relativen" Fehler ersetzt. Ferner wurde dabei der Betrag des Funktionswerts nach unten komponentenweise durch 10^{-5} begrenzt. Zusätzlich wurde die "optimale" Schrittweite *hneu* mit dem Faktor 0.9 verkleinert. Dies verringert die Wahrscheinlichkeit, dass der nächste Schritt verworfen wird. Schließlich wurde durch den minimalen Faktor 0.2 und den maximalen Faktor 10 dafür gesorgt, dass die Schrittweiten von Schritt zu Schritt sich nicht zu stark ändern. Die gewählten Konstanten 0.9, 0.2 und 10 lassen sich durch keine Theorie begründen, sondern haben sich in Codes bewährt. □

Bemerkung 8.6 Nach unserer Herleitung müsste in der Zeile $(*)$ $y = y1$ stehen. Da man aber ohne Mehrkosten die bessere Näherung $y = 2 * y2 - y1$ (Formel der Ordnung 2) zur Verfügung hat, verwendet man diese. Unsere Fehlerschätzung ist damit in der Regel pessimistisch. □

Beispiel 8.7

$$y' = y^2, \quad y(0.8) = \frac{5}{6}.$$

Mit $\tau = 1e-3$ benötigt man 61 Funktionsauswertungen für die numerische Lösung im Intervall $[0.8, 1.8]$. Der maximale absolute Fehler ist dabei $1.61 \cdot 10^{-2}$, der maximale relative Fehler $3.21e-3$, die maximale benutzte Schrittweite ist $3.52 \cdot 10^{-2}$ und die minimale Schrittweite ist $2.86 \cdot 10^{-3}$. Abbildung 8.2 zeigt die berechneten Punkte; die Schrittweite wird kleiner bei größerer Steigung.

Um dieselbe Genauigkeit mit äquidistanter Schrittweite zu erreichen, benötigt man 2776 Funktionsauswertungen. □

8.2 Allgemeine Einschrittverfahren

Das behandelte Polygonzugverfahren ist die einfachste Methode der großen Klasse der **Einschrittverfahren**, bei denen die Näherung y_{n+1} an dem neuen Punkt

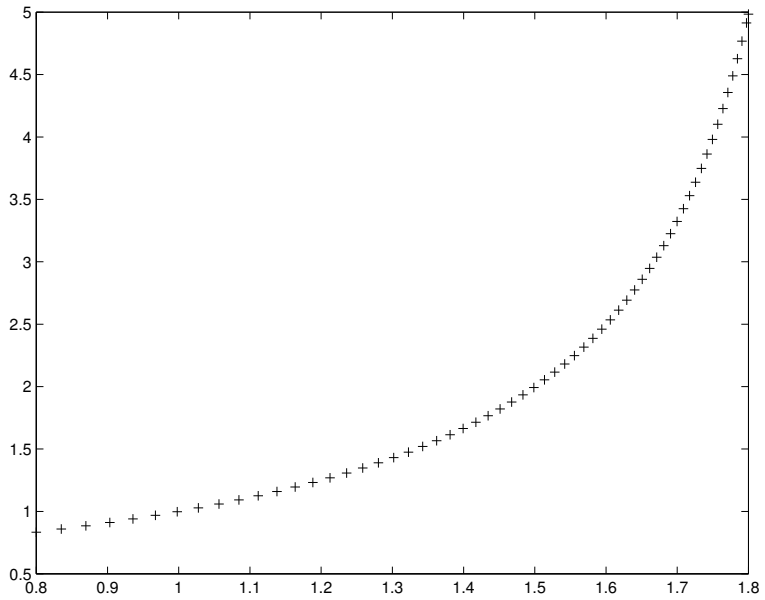

Abbildung 8.2: Schrittweitenkontrolle in Beispiel 8.7

$x_{n+1} := x_n + h_n$ allein aus der Näherung y_n an der Stelle x_n und der Schrittweite h_n berechnet wird. Einschrittverfahren haben also die folgende Gestalt

$$y_{n+1} = y_n + h_n \, \Phi(x_n, y_n, h_n) \tag{8.13}$$

mit einer **Verfahrensfunktion** Φ.

Um die Güte von Einschrittverfahren zu beurteilen, führen wir die folgenden Begriffe ein:

Definition 8.8 Es sei $z(x)$ die Lösung der Anfangswertaufgabe

$$z' = f(x, z(x)), \;\; z(x_n) = y_n.$$

Dann heißt

$$\varepsilon(h) := z(x_n + h) - y_n - h \, \Phi(x_n, y_n, h)$$

der **lokale Fehler** des durch (8.13) definierten Verfahrens.

Das Verfahren (8.13) heißt **konsistent**, falls $\varepsilon(h) = o(h)$ gilt, es heißt **von der Ordnung** p, wenn $\varepsilon(h) = O(h^{p+1})$ gilt.

Wie im Falle des Polygonzugverfahrens gilt:

Satz 8.9 *Es seien sie Näherungen y_n von $y(x_n)$ mit dem Einschrittverfahren*

$$y_{n+1} = y_n + h_n \Phi(x_n, y_n, h_n), \;\; n = 0, 1, \ldots, N,$$

berechnet.

Erfüllt die Verfahrensfunktion Φ eine Lipschitz Bedingung bzgl. y in $[a, b] \times \mathbb{R}$ (es genügt eine Umgebung der Lösung)

$$|\Phi(x, y, h) - \Phi(x, z, h)| \leqslant \Lambda |y - z| \tag{8.14}$$

und ist das Einschrittverfahren konsistent von der Ordnung p

$$|y(x + h) - y(x) - h\Phi(x, y(x), h)| \leqslant C \cdot h^{p+1}, \tag{8.15}$$

so gilt für den globalen Fehler

$$|\delta_n| = |y_n - y(x_n)| \leqslant C(b-a)e^{\Lambda(b-a)}\, h^p, \tag{8.16}$$

wobei

$$h := \max_{j=0,\ldots,N-1} h_j$$

gesetzt ist.

Beweis: Mit wörtlich demselben Beweis wie für das Polygonzugverfahren. ■

Bemerkung 8.10 Die Lipschitz Bedingung für die Verfahrensfunktion Φ erhält man in vielen Fällen aus der Lipschitz Bedingung für die rechte Seite f. □

Bemerkung 8.11 Wie beim Übergang von den Quadraturformeln zu summierten Quadraturformeln verliert man beim Übergang vom lokalen zum globalen Fehler eine h-Potenz. □

Beispiel 8.12 (Polygonzugverfahren) (Euler 1768)
Das einfachste Einschrittverfahren ist das in Abschnitt 8.1 betrachtete Eulersche Polygonzugverfahren

$$\Phi(x, y, h) = f(x, y).$$ □

Beispiel 8.13 (Verbessertes Polygonzugverfahren) (Coriolis 1837, Runge 1895)
Wir haben dieses Verfahren bereits durch Extrapolation aus dem Polygonzugverfahren mit den Schrittweiten h und $\frac{h}{2}$ hergeleitet (vgl. (8.11)):

$$y_{n+1} = y_n + h_n\, f\Big(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\, f(x_n, y_n)\Big).$$

Geometrisch kann man dieses Verfahren so interpretieren: Es wird zunächst eine Schätzung $y_{n+\frac{1}{2}} = y_n + \frac{h_n}{2}\, f(x_n, y_n)$ für $y\Big(x_n + \frac{h_n}{2}\Big)$ ermittelt, und die hiermit berechnete Näherung $f\Big(x_n + \frac{h_n}{2}, y_{n+\frac{1}{2}}\Big) \approx y'\Big(x_n + \frac{h_n}{2}\Big)$ für die Steigung von y im ganzen Intervall $[x_n, x_n + h_n]$ verwendet. Eine Veranschaulichung findet sich in der linken Skizze in Abbildung 8.3.

Wir haben bereits gesehen, dass das verbesserte Polygonzugverfahren die Ordnung 2 besitzt. Erfüllt f eine Lipschitzbedingung

$$|f(x, y) - f(x, z)| \leqslant L|y - z|,$$

so erfüllt auch die Verfahrensfunktion

$$\Phi(x, y, h) = f(x + 0.5h, y + 0.5hf(x, y))$$

wegen

$$\begin{aligned}
&|\Phi(x, y, h) - \Phi(x, z, h)| \\
&= |f(x + 0.5h, y + 0.5hf(x, y)) - f(x + 0.5h, z + 0.5hf(x, z))| \\
&\leqslant L|y + 0.5hf(x, y) - (z + 0.5hf(x, z))| \\
&\leqslant L(|y - z| + 0.5h|f(x, y) - f(x, z)|) \\
&\leqslant L(|y - z| + 0.5hL|y - z|) \\
&\leqslant L(1 + 0.5(b - a)L)|y - z| =: \Lambda|y - z|
\end{aligned}$$

eine Lipschitz Bedingung. Nach Satz 8.9 ist das verbesserte Polygonzugverfahren also nicht nur ein Verfahren der Ordnung 2, sondern der *globale Fehler* konvergiert auch gegen Null wie h^2. □

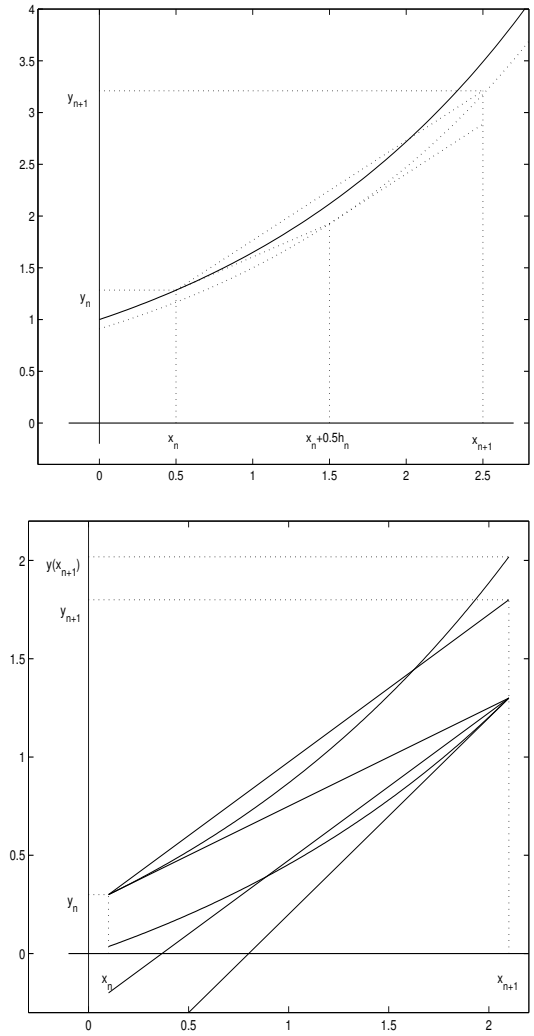

Abbildung 8.3: Verbessertes Polygonzugverfahren / Verfahren von Heun

Beispiel 8.14 Eine naheliegende Idee, Verfahren höherer Ordnung zu entwickeln, liefert der Taylorsche Satz. Ist f differenzierbar, so erhält man wegen

$$\begin{aligned} y''(x) &= \frac{d}{dx} f(x, y(x)) = f_x(x, y(x)) + f_y(x, y(x)) y'(x) \\ &= f_x(x, y(x)) + f_y(x, y(x)) f(x, y(x)) \end{aligned}$$

und

$$\begin{aligned} y(x_n + h) &= y(x_n) + h y'(x_n) + \frac{1}{2} h^2 y''(x_n) + O(h^3) \\ &= y_n + h f(x_n, y_n) + \frac{1}{2} h^2 (f_x(x_n, y_n) + f_y(x_n, y_n) f(x_n, y_n)) + O(h^3) \end{aligned}$$

mit

$$\Phi(x, y, h) = f(x, y) + 0.5h(f_x(x, y) + f_y(x, y) f(x, y))$$

ein Einschrittverfahren der Ordnung 2. Nachteil ist, dass man die Ableitung der rechten Seite benötigt, deren Auswertung häufig aufwendiger ist als die der Funktion.

Besitzt f höhere Ableitungen, so kann man auf dieselbe Weise Einschrittverfahren höherer Ordnung bestimmen. □

Beispiel 8.15 (Verfahren von Heun) (Heun 1900)
Dem verbesserten Polygonzugverfahren verwandt ist das Verfahren von Heun, das in der Literatur machmal auch Verbessertes Polygonzugverfahren genannt wird. Man verwendet den Mittelwert zweier Näherungen

$$k_1 := f(x_n, y_n), \quad k_2 := f(x_n + h_n, y_n + h_n k_1)$$

für die Steigung von y im Intervall $[x_n, x_{n+1}]$ und setzt hiermit

$$y_{n+1} = y_n + h_n \frac{k_1 + k_2}{2}.$$

Dieses Verfahren entspricht der Quadratur des Integrals in

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(t, y(t))\, dt$$

mit der Trapezregel, wenn man den unbekannten Punkt $(x_{n+1}, y(x_{n+1}))$ ersetzt durch $(x_{n+1}, y_n + h_n f(x_n, y_n))$. Eine Veranschaulichung findet man in der rechten Skizze von Abbildung 8.3.

Mit dem Taylorschen Satz kann man zeigen, dass für den lokalen Fehler

$$\varepsilon(h) = z(x_n + h) - y_n - \frac{h}{2}(f(x_n, y_n) + f(x_n + h, y_n + h\, f(x_n, y_n))) = O(h^3)$$

gilt, dass das Verfahren von Heun also wie das verbesserte Polygonzugverfahren die Ordnung 2 besitzt. □

8.3 Explizite Runge-Kutta Verfahren

Beispiel 8.16 (explizite Runge-Kutta Verfahren) (Kutta 1901)
Explizite Runge-Kutta Verfahren sind Verallgemeinerungen der Einschrittverfahren in Beispiel 8.12, Beispiel 8.13 und Beispiel 8.15. Es wird ein gewichtetes Mittel von Approximationen der Steigung der Lösung y im Intervall $[x_n, x_{n+1}]$ bestimmt und hiermit ein Schritt der Länge h_n ausgeführt. Es sei

$$\begin{aligned} k_1 &:= f(x_n, y_n) \\ k_j &:= f(x_n + \alpha_j\, h_n, y_n + h_n \sum_{\ell=1}^{j-1} \beta_{j\ell}\, k_\ell), \quad j = 2, \ldots, s \\ y_{n+1} &:= y_n + h_n \sum_{j=1}^{s} \gamma_j\, k_j. \end{aligned} \tag{8.17}$$

Die Koeffizienten $\alpha_j, \beta_{j\ell}, \gamma_j$ werden dabei so gewählt, dass das Verfahren möglichst hohe Ordnung hat. s heißt die **Stufe** des Runge-Kutta Verfahrens. □

Als Beispiel betrachten wir die Herleitung von zweistufigen Runge-Kutta Verfahren. Es gilt

$$\begin{aligned} y_{n+1} &= y_n + h(\gamma_1 k_1 + \gamma_2 k_2) \\ &= y_n + h(\gamma_1 f(x_n, y_n) + \gamma_2 f(x_n + \alpha_2 h, y_n + h\beta_{21} f(x_n, y_n))). \end{aligned} \tag{8.18}$$

Wir bestimmen die Parameter γ_1, γ_2, α_2 und β_{21} so, dass das Verfahren möglichst große Ordnung hat.

Ist $f \in C^2$ (und damit $y \in C^3$), so gilt nach dem Taylorschen Satz

$$\begin{aligned} y(x+h) &= y(x) + hy'(x) + \frac{1}{2}h^2y''(x) + \frac{1}{6}h^3y'''(x) + o(h^3) \\ &= y(x) + hf(x,y(x)) + \frac{1}{2}h^2\big(f_x(x,y(x)) + f_y(x,y(x))f(x,y(x))\big) \\ &\quad + \frac{1}{6}h^3\big[f_{xx} + 2ff_{xy} + f^2f_{yy} + f_xf_y + ff_y^2\big](x,y(x)) + o(h^3) \end{aligned}$$

und

$$\begin{aligned} &y(x) + h(\gamma_1 f(x,y(x)) + \gamma_2 f(x+\alpha_2 h, y(x) + h\beta_{21}f(x,y(x)))) \\ &= y(x) + h\gamma_1 f(x,y(x)) + h\gamma_2\big[f + \alpha_2 hf_x + \beta_{21}hf_yf + \frac{1}{2}(\alpha_2 h)^2 f_{xx} \\ &\quad + \alpha_2\beta_{21}h^2 f_{xy}f + \frac{1}{2}(\beta_{21}h)^2 f^2 f_{yy}\big](x,y(x)) + o(h^3). \end{aligned}$$

Subtraktion und Ordnen nach Potenzen von h liefert

$$\begin{aligned} &y(x+h) - y(x) - h(\gamma_1 f(x,y(x)) + \gamma_2 f(x+\alpha_2 h, y(x) + h\beta_{21}f(x,y(x)))) \\ &= h(1-\gamma_1-\gamma_2)f(x,y(x)) \\ &+ \frac{1}{2}h^2\big[(1-2\gamma_2\alpha_2)f_x + (1-2\gamma_2\beta_{21})ff_y\big](x,y(x)) \\ &+ \frac{1}{6}h^3\big[(1-3\gamma_2\alpha_2^2)f_{xx} + 2(1-3\gamma_2\alpha_2\beta_{21})f_{xy}f \\ &\quad + (1-3\gamma_2\beta_{21}^2)f_{yy}f^2 + f_xf_y + ff_y^2\big](x,y(x)) + o(h^3). \end{aligned}$$

Da bei keiner Wahl der Parameter der Koeffizient bei h^3 für alle Funktionen f verschwindet, können wir keine höhere Konvergenzordnung als 2 erreichen.

Für Verfahren der Ordnung 2 muss gelten

$$\gamma_1 + \gamma_2 = 1,\ 2\gamma_2\alpha_2 = 1,\ 2\gamma_2\beta_{21} = 1. \tag{8.19}$$

Dieses nichtlineare System von 3 Gleichungen in 4 Unbekannten besitzt unendlich viele Lösungen. Wählt man γ_2 als freien Parameter, so erhält man die Lösungsschar

$$\gamma_1 = 1-\gamma_2,\ \alpha_2 = \beta_{21} = \frac{1}{2\gamma_2},\ \gamma_2 \neq 0. \tag{8.20}$$

Die bereits betrachteten Verfahren der Ordnung 2 sind hierin enthalten. Für $\gamma_2 = 1$ erhält man das verbesserte Polygonzugverfahren, für $\gamma_2 = 0.5$ das Verfahren von Heun.

Unter Verwendung von (8.19) geht der Term bei h^3 über in

$$\frac{1}{6}((1-\frac{3}{4\gamma_2})(f_{xx} + 2ff_{xy} + f^2f_{yy}) + f_xf_y + ff_y^2)(x,y(x)). \tag{8.21}$$

Hier ist die Summe der Beträge der Koeffizienten vor den partiellen Ableitungen minimal für $\gamma_2 = \frac{3}{4}$. In diesem Sinne ist also die folgende Formel unter den Methoden (8.18) optimal:

$$y_{n+1} = y_n + \frac{1}{4}h_n\big(f(x_n,y_n) + 3f(x_n + \frac{2}{3}h_n, y_n + \frac{2}{3}h_nf(x_n,y_n))\big). \tag{8.22}$$

Damit ist nicht gesagt, dass (8.22) unter allen möglichen Formeln (8.18) bei Anwendung auf eine spezielle Differentialgleichung das kleinste führende Fehlerglied hat, da sich in der Entwicklung von

$$y(x+h) - y(x) - h\Phi(x,y(x))$$

h	verb. Poly.	Heun	optimal	Taylor
1/5	$1.01e+0$	$8.51e-1$	$9.58e-1$	$1.16e+0$
1/10	$4.34e-1$	$3.38e-1$	$4.02e-1$	$5.25e-1$
1/20	$1.47e-1$	$1.07e-1$	$1.34e-1$	$1.86e-1$
1/40	$4.27e-2$	$2.98e-2$	$3.84e-2$	$5.54e-2$
1/80	$1.14e-2$	$7.82e-3$	$1.02e-2$	$1.51e-2$
1/160	$2.96e-3$	$2.00e-3$	$2.64e-3$	$3.92e-3$
1/320	$7.51e-4$	$5.04e-4$	$6.69e-4$	$9.99e-4$
1/640	$1.89e-4$	$1.27e-4$	$1.48e-4$	$2.52e-4$
1/1280	$4.75e-5$	$3.17e-5$	$4.22e-5$	$6.33e-5$

Tabelle 8.2: Verfahren der Ordnung 2

Terme mit entgegengesetztem Vorzeichen kompensieren können. Andere Optimalitätskriterien sind denkbar (vgl. Grigorieff [33]).

Beispiel 8.17 Wir betrachten erneut

$$y' = y^2,\ y(0.8) = \frac{5}{6},\ x \in [0.8, 1.8].$$

Dann erhält man mit dem verbesserten Polygonzugverfahren, dem Verfahren von Heun, dem "optimalen" Verfahren aus (8.22) und dem Verfahren aus Beispiel 8.14 die Fehler der Tabelle 8.2. □

Bevor wir Verfahren größerer Ordnung als 2 angeben, schicken wir einige Bemerkungen voraus über die mit einem s-stufigen Verfahren erreichbare Konsistenzordnung. Diese Frage ist keinesfalls leicht zu beantworten, da die beim Taylorabgleich entstehenden Bedingungsgleichungen nichtlinear in den Parametern sind. Eine sorgfältige Untersuchung mit Hilfe von Ordnungsbäumen findet man in *Hairer, Nørsett, Wanner* [38].

Die Zahl der zu erfüllenden Gleichungen steigt mit wachsender Ordnung p sehr stark an, wie die folgende Tabelle zeigt:

Ordnung p	1	2	3	4	5	6	7	8	9	10
Zahl der Gleichungen	1	2	4	8	17	37	85	200	486	1205.

Dabei wurden die Gleichungen

$$\sum_{j=1}^{k-1} \beta_{kj} = \alpha_k,\ k = 2, \ldots, m,$$

die wir stets als erfüllt annehmen, nicht mitgezählt.

Gibt man die Ordnung p vor und bestimmt dazu die Stufenzahl s des Runge-Kutta Verfahrens minimal, so gilt der folgende Zusammenhang

p	1	2	3	4	5	6	7	8	9	10
s	1	2	3	4	6	7	9	10	11	12.

Man sieht, dass sich mit wachsender Ordnung das Verhältnis von erreichbarer Ordnung p zur Zahl der dazu nötigen Stufen (also zur Zahl der Funktionsauswertungen in jedem Schritt) verschlechtert.

Entwickelt man (8.17) bis zu Termen mit h^2, so sieht man unmittelbar:

Satz 8.18 *Das Einschrittverfahren* (8.17) *ist genau dann konsistent, wenn gilt*

$$\sum_{j=1}^{s} \gamma_j = 1.$$

Wir betrachten nun den Fall $s = 4$: Eine etwas mühsame Taylorentwicklung (ähnlich wie im Fall $s = 2$; vgl. Gear [27], p. 32 ff) zeigt, dass man in der Darstellung des lokalen Fehlers durch keine Wahl der Parameter den Koeffizienten bei h^5 unabhängig von f zum Verschwinden bringen kann. Es ist also die Ordnung $p = 4$ erreichbar. Die acht Bedingungen der 10 Parameter (3 Gleichungen zur Bestimmung der Parameter α_j nicht mitgerechnet) lauten:

$$\begin{aligned}
1 &= \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 \\
\frac{1}{2} &= \alpha_2\gamma_2 + \alpha_3\gamma_3 + \alpha_4\gamma_4 \\
\frac{1}{3} &= \alpha_2^2\gamma_2 + \alpha_3^2\gamma_3 + \alpha_4^2\gamma_4 \\
\frac{1}{4} &= \alpha_2^3\gamma_2 + \alpha_3^3\gamma_3 + \alpha_4^3\gamma_4 \\
\frac{1}{6} &= \alpha_3\beta_{43}\gamma_4 + \alpha_2\beta_{42}\gamma_4 + \alpha_2\beta_{32}\gamma_3 \\
\frac{1}{8} &= \alpha_3\alpha_4\beta_{43}\gamma_4 + \alpha_2\alpha_4\beta_{42}\gamma_4 + \alpha_2\alpha_3\beta_{32}\gamma_3 \\
\frac{1}{12} &= \alpha_3^2\beta_{43}\gamma_4 + \alpha_2^2\beta_{42}\gamma_4 + \alpha_2^2\beta_{32}\gamma_3 \\
\frac{1}{24} &= \alpha_2\beta_{32}\beta_{43}\gamma_4.
\end{aligned}$$

Die Parameter sind durch die vorstehenden Gleichungen nicht eindeutig bestimmt. Wir geben 3 verschiedene Formeln an. Dabei verwenden wir das folgende Koeffizientenschema, mit dem man Runge-Kutta Verfahren auf übersichtliche Weise darstellen kann:

$$\begin{array}{c|cccccc}
0 & & & & & & \\
\alpha_2 & \beta_{21} & & & & & \\
\alpha_3 & \beta_{31} & \beta_{32} & & & & \\
\vdots & & & & & & \\
\alpha_s & \beta_{s1} & \beta_{s2} & \cdots & \beta_{s,s-1} & & \\
\hline
 & \gamma_1 & \gamma_2 & \cdots & \gamma_{s-1} & \gamma_s.
\end{array}$$

Die uns bekannten Verfahren der Ordnung 2 kann man damit so schreiben

$$\begin{array}{c|cc}
0 & & \\
1 & 1 & \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
\qquad
\begin{array}{c|cc}
0 & & \\
\frac{1}{2} & \frac{1}{2} & \\
\hline
 & 0 & 1
\end{array}
\qquad
\begin{array}{c|cc}
0 & & \\
\frac{2}{3} & \frac{2}{3} & \\
\hline
 & \frac{1}{4} & \frac{3}{4}
\end{array}$$

Verfahren von Heun verb. Polygonzugverfahren optimales Verfahren

Am bekanntesten ist wohl das **klassische Runge-Kutta Verfahren** (1895) der Ordnung 4.

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}.
\end{array}$$

h	klassisch	3/8-Regel	Kuntzmann
1/5	$3.52e-2$	$3.42e-2$	$3.59e-2$
1/10	$3.39e-3$	$3.36e-3$	$3.51e-3$
1/20	$2.50e-4$	$2.38e-4$	$2.59e-4$
1/40	$1.65e-5$	$1.43e-5$	$1.67e-5$
1/80	$1.05e-6$	$8.25e-7$	$1.03e-6$
1/160	$6.58e-8$	$4.82e-8$	$6.37e-8$
1/320	$4.12e-9$	$2.89e-9$	$3.94e-9$

Tabelle 8.3: Verfahren der Ordnung 4

Ausführlich geschrieben lautet dieses

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f\Big(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\, k_1\Big) \\ k_3 &= f\Big(x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\, k_2\Big) \\ k_4 &= f(x_n + h_n, y_n + h_n\, k_3) \\ y_{n+1} &= y_n + h_n\, \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}. \end{aligned}$$

Weitere Verfahren der Ordnung 4 sind die **3/8-Regel** (Kutta 1901)

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array}$$

und die **optimale Formel von Kuntzmann** (Kuntzmann 1959)

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{2}{5} & \frac{2}{5} & & & \\ \frac{3}{5} & -\frac{3}{20} & \frac{3}{4} & & \\ 1 & \frac{19}{44} & -\frac{15}{44} & \frac{40}{44} & \\ \hline & \frac{55}{360} & \frac{125}{360} & \frac{125}{360} & \frac{55}{360}. \end{array}$$

Die 3/8-Regel verallgemeinert die Newtonsche 3/8-tel Regel (oder Keplersche Fassregel oder pulcherima) der numerischen Integration. Die Formel von Kuntzmann erhält man ähnlich wie im Falle $s = 2$, indem man die Parameter so bestimmt, dass die Summe der Beträge der Koeffizienten im führenden Fehlerglied von $y(x+h) - y(x) - h\Phi(x, y(x))$ minimal wird.

Die klassische Runge-Kutta Regel ist die bekannteste, die 3/8-Regel häufig die genaueste der expliziten Runge-Kutta Verfahren der Ordnung 4.

Beispiel 8.19 Wir betrachten erneut

$$y' = y^2,\ y(0.8) = \frac{5}{6},\ x \in [0.8, 1.8].$$

Dann erhält man mit dem klassischen Runge-Kutta Verfahren, der 3/8-Regel und dem optimalen Verfahren von Kuntzmann die Fehler der Tabelle 8.3 □

8.4 Schrittweitensteuerung

Eine Schrittweitensteuerung kann man für die Runge-Kutta Verfahren prinzipiell wie für das Polygonzugverfahren durchführen. Um den Fehler zu schätzen, kann man zwei Schritte mit der halben Schrittweite ausführen. Im Falle der klassischen Runge-Kutta Verfahren hat man dabei die Funktion f an 7 zusätzlichen Punkten auszuwerten, so dass man in jedem Schritt insgesamt 11 Funktionsauswertungen benötigt.

8.5 Eingebettete Runge-Kutta Verfahren

Mit wesentlich weniger Aufwand kommt man bei den **eingebetteten Runge-Kutta Formeln** aus:

Die Idee ist — ähnlich wie bei den Kronrod-Formeln zur Quadratur — von einer Runge-Kutta Formel der Stufe s mit den Zuwächsen $k_1, \ldots, k_s$ und der Ordnung p auszugehen und hierzu bei erhöhter Stufenzahl σ weitere $k_{s+1}, \ldots, k_\sigma$ zu bestimmen, so dass die Formel

$$\tilde{y}_{n+1} = y_n + h_n\Big(\sum_{j=1}^{s} \tilde{\gamma}_j\, k_j + \sum_{j=s+1}^{\sigma} \tilde{\gamma}_j\, k_j\Big)$$

eine höhere Ordnung q als die Ausgangsformel hat.

Dann gilt für die lokalen Fehler $\varepsilon = C\,h^{p+1} + O(h^{p+2})$ und $\tilde{\varepsilon} = O(h^{q+1}) = O(h^{p+2})$, d.h. $\tilde{y}_{n+1} - y_{n+1} = C\,h^{p+1} + O(h^{p+2})$, und hiermit kann man bei vorgegebener Toleranz die optimale Schrittweite wie vorher schätzen.

Man führt also einen Probeschritt der Länge H aus, erhält hieraus

$$C \approx \frac{\tilde{y}_{n+1}(H) - y_{n+1}(H)}{H^{p+1}},$$

und die Forderung $|\varepsilon(x_n, h)| \approx |C| h^{p+1} = \tau$ liefert die neue Schrittweite

$$h = H\left(\frac{\tau}{|\tilde{y}_{n+1}(H) - y_{n+1}(H)|}\right)^{1/(p+1)}.$$

Wir geben einige **eingebettete Formelpaare** an:

Fehlberg (Ordnungen $p = 2$, $q = 3$)

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \\ \hline p=2 & \frac{1}{2} & \frac{1}{2} & \\ q=3 & \frac{1}{6} & \frac{1}{6} & \frac{2}{3}. \end{array}$$

England (Ordnungen $p = 4$, $q = 5$)

$$\begin{array}{c|ccccc} 0 & & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & & & \\ 1 & 0 & -1 & 2 & & \\ \frac{2}{3} & \frac{7}{27} & \frac{10}{27} & 0 & \frac{1}{27} & \\ \frac{1}{5} & \frac{28}{625} & -\frac{125}{625} & \frac{546}{625} & \frac{54}{625} & -\frac{378}{625} \\ \hline p=4 & \frac{1}{6} & 0 & \frac{2}{3} & \frac{1}{6} & 0 \\ q=5 & \frac{14}{336} & 0 & 0 & \frac{35}{336} & \frac{162}{336} \quad \frac{125}{336}. \end{array}$$

Verner (Ordnungen $p = 5$, $q = 6$)

$$
\begin{array}{c|cccccccc}
0 & & & & & & & & \\
\frac{1}{6} & \frac{1}{6} & & & & & & & \\
\frac{4}{15} & \frac{4}{75} & \frac{16}{75} & & & & & & \\
\frac{2}{3} & \frac{5}{6} & -\frac{8}{3} & \frac{5}{2} & & & & & \\
\frac{5}{6} & -\frac{165}{64} & \frac{55}{6} & -\frac{425}{64} & \frac{85}{96} & & & & \\
1 & \frac{12}{5} & -8 & \frac{4015}{612} & -\frac{11}{36} & \frac{88}{255} & & & \\
\frac{1}{15} & -\frac{8263}{15000} & \frac{124}{75} & -\frac{643}{680} & -\frac{81}{250} & \frac{2484}{10625} & 0 & & \\
1 & \frac{3501}{1720} & -\frac{300}{43} & \frac{297275}{52632} & -\frac{319}{2322} & \frac{24068}{84065} & 0 & \frac{3850}{26703} & \\
\hline
p = 5 & \frac{13}{160} & 0 & \frac{2375}{5984} & \frac{5}{16} & \frac{12}{85} & \frac{3}{44} & 0 & \\
q = 6 & \frac{3}{40} & 0 & \frac{875}{2244} & \frac{23}{72} & \frac{264}{1955} & 0 & \frac{125}{11592} & \frac{43}{616}.
\end{array}
$$

Wie beim Polygonzugverfahren mit Schrittweitensteuerung verwendet man bei diesen sehr frühen Formelpaaren im weiteren Verlauf der Integration die Näherung für das Verfahren höherer Ordnung, obwohl der Fehler für das Verfahren niedrigerer Ordnung geschätzt wurde. Für diese Näherung liegt dann zwar keine Fehlerschätzung vor, aber ein Anwender wird sich wohl kaum beschweren, wenn der Code eine höhere Genauigkeit liefert als die Geforderte.

Neuere Formelpaare wurden so konstruiert, dass die neue Näherung y_{n+1} in x_{n+1} so gewählt ist, dass das im nächsten Schritt benötigte $k_1 = f(x_{n+1}, y_{n+1})$ bereits im letzten Schritt berechnet wurde. Verfahren dieses Typs werden als **FSAL-Verfahren** (*first* evaluation of f for the next step *same as last* of the current step). Bei ihnen bekommt man in jedem erfolgreichen Schritt eine Auswertung der rechten Seite geschenkt.

Das bekannteste und wohl am häufigsten verwendete Verfahren, das auch in der ODE-Suite von MATLAB 6.1 als ODE45 implementiert ist, ist das Formelpaar von Dormand und Prince [21], das Runge-Kutta Verfahren der Ordnungen 5 und 4 kombiniert.

Für etwas schwächere Genauigkeitsanforderungen wird eine Kombination von Verfahren der Ordnungen 3 und 2 von Bogacki und Shampine [9] in der ODE-Suite von MATLAB 6.1 als ODE23 bereitgestellt.

Bogacki und Shampine (Ordnungen $p = 3$, $q = 2$)

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{3}{4} & & \frac{3}{4} & & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
\hline
p = 3 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
q = 2 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}.
\end{array}
$$

Dormand und Prince (Ordnungen $p = 5$, $q = 4$)

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
$p = 5$	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$q = 4$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$.

Beispiel 8.20

$$y' = y^2, \quad y(0.8) = \frac{5}{6}.$$

Um die Lösung $y(x) = \dfrac{1}{2-x}$ im Punkte $b = 1.8$ mit höchstens dem absoluten Fehler 0.0005 zu approximieren, benötigt man mit dem Formelpaar von Fehlberg 138 Funktionsauswertungen, mit dem von England 96, mit dem von Verner 84, mit dem Paar von Dormand und Prince 49 und mit dem Paar von Bogacki und Shampine 187 Funktionsauswertungen. □

Kapitel 9

Mehrschrittverfahren

Ein weiterer, häufig benutzter Verfahrenstyp zur numerischen Lösung der Anfangswertaufgabe

$$y' = f(x, y),\ y(a) = y_0 \tag{9.1}$$

sind die linearen **Mehrschrittverfahren**, bei denen man zur Berechnung der Näherung y_{n+k} die bereits ermittelten Näherungen $y_{n+k-1}, y_{n+k-2}, \ldots, y_n$ verwendet. Dazu macht man den Ansatz

$$\sum_{\nu=0}^{k} a_\nu\, y_{n+\nu} = h \sum_{\nu=0}^{k} b_\nu\, f_{n+\nu} \tag{9.2}$$

mit $f_{n+\nu} := f(x_{n+\nu}, y_{n+\nu})$, wobei $a_k \neq 0$ vorausgesetzt wird.

Ist $b_k \neq 0$, so kommt y_{n+k} auf beiden Seiten von (9.2) vor. Es muss dann in jedem Schritt ein (i.a. nichtlineares) Gleichungssystem gelöst werden, um y_{n+k} zu bestimmen. In diesem Fall heißt (9.2) **implizites k-Schritt Verfahren**. Ist $b_k = 0$, so kann man (9.2) sofort nach y_{n+k} auflösen. In diesem Fall heißt (9.2) **explizites k-Schritt Verfahren**.

Offenbar ist der erste Wert, den man mit (9.2) berechnen kann, y_k. Neben dem gegebenen Wert y_0 müssen also zunächst Näherungen $y_1, \ldots, y_{k-1}$ für $y(x_1), \ldots, y(x_{k-1})$ zur Verfügung gestellt werden. Diese können z.B. mit einem Runge-Kutta Verfahren berechnet werden.

Wegen $a_k \neq 0$ können wir o.B.d.A. $a_k = 1$ annehmen. Wir bestimmen die übrigen a_ν, b_ν nun so, dass (9.2) zu einem brauchbaren Verfahren wird.

Den lokalen Fehler von (9.2) erhält man wieder, indem man die exakte Lösung $y(x)$ von (9.1) in (9.2) einsetzt:

Definition 9.1 Der **lokale Fehler** des k-Schritt Verfahrens (9.2) ist definiert durch

$$\varepsilon_n := \sum_{\nu=0}^{k} a_\nu\, y(x_n + \nu h) - h \sum_{\nu=0}^{k} b_\nu\, y'(x_n + \nu h). \tag{9.3}$$

wobei y die Lösung der Anfangswertaufgabe (9.1) bezeichnet.

9.1 Konsistenz

Das Verfahren (9.2) heißt **konsistent**, wenn $\varepsilon_n(h) = o(h)$ gilt, es heißt **von der Ordnung** p, wenn $\varepsilon_n(h) = O(h^{p+1})$ für alle genügend glatten Funktionen y gilt.

Ist y $(m+1)$-mal differenzierbar, so liefert der Taylorsche Satz

$$\varepsilon_n = \sum_{\nu=0}^{k} a_\nu \Big(\sum_{\mu=0}^{m} \frac{y^{(\mu)}(x_n)}{\mu!}\, \nu^\mu h^\mu + \frac{1}{(m+1)!}\, y^{(m+1)}(x_n + \theta_\nu \nu h)\, \nu^{m+1} h^{m+1} \Big)$$
$$- \sum_{\nu=0}^{k} b_\nu \Big(\sum_{\mu=0}^{m-1} \frac{y^{(\mu+1)}(x_n)}{\mu!}\, \nu^\mu h^{\mu+1} + \frac{1}{m!}\, y^{(m+1)}(x_n + \hat{\theta}_\nu \nu h)\, \nu^m h^{m+1} \Big).$$

Damit das Verfahren konsistent ist, müssen sich die Glieder mit dem Faktor h^0 und mit dem Faktor h^1 jeweils gegenseitig aufheben, d.h. es muss gelten

$$\sum_{\nu=0}^{k} a_\nu = 0, \quad \sum_{\nu=0}^{k} (\nu a_\nu - b_\nu) = 0. \tag{9.4}$$

Das Nächstliegende ist nun, die a_ν, b_ν so zu bestimmen, dass in ε_n möglichst hohe Potenzen von h abgeglichen werden.

Dies führt zu einem linearen (wegen $a_k = 1$ inhomogenen) Gleichungssystem.

Für $k = 2$ erhält man

$$\begin{array}{rcrcrcrcrl}
a_0 & + & a_1 & & & & & = & -1 & \text{(siehe (9.4))} \\
 & & a_1 & - & b_0 & - & b_1 & - & b_2 = & -2 \quad \text{(siehe (9.4))} \\
 & & a_1 & & & - & 2b_1 & - & 4b_2 = & -4 \\
 & & a_1 & & & - & 3b_1 & - & 12b_2 = & -8 \\
 & & a_1 & & & - & 4b_1 & - & 32b_2 = & -16
\end{array} \tag{9.5}$$

mit der Lösung $a_0 = -1$, $a_1 = 0$, $b_0 = \frac{1}{3}$, $b_1 = \frac{4}{3}$, $b_2 = \frac{1}{3}$.

Man erhält also das implizite Verfahren der Ordnung 4:

$$y_{n+2} = y_n + \frac{h}{3}\,(f_n + 4f_{n+1} + f_{n+2}).$$

Verlangt man, um die Auflösung einer nichtlinearen Gleichung (bzw. eines nichtlinearen Gleichungssystems) in y_{n+2} in jedem Schritt zu vermeiden, $b_2 = 0$, d.h. ein explizites Verfahren, so kann man nur die ersten vier Gleichungen von (9.5) erfüllen. Lösung hiervon ist $a_0 = -5$, $a_1 = 4$, $b_0 = 2$, $b_1 = 4$, und man erhält das explizite Verfahren der Ordnung 3:

$$y_{n+2} = -4y_{n+1} + 5y_n + 2h(f_n + 2f_{n+1}). \tag{9.6}$$

Wendet man (9.6) auf die Anfangswertaufgabe $y' = 0$, $y(0) = 1$ mit dem (z.B. durch Rundungsfehler verfälschten) Anfangsfeld $y_0 = 1$, $y_1 = 1 + 10^{-15}$ an, so erhält man Tabelle 9.1.

Kleinste Anfangsfehler schaukeln sich also auf und machen das Verfahren trotz der Ordnung 3 völlig unbrauchbar.

Die Fehlerordnung allein ist also kein geeignetes Mittel zur Bewertung eines Mehrschrittverfahrens.

9.2 Stabilität

Für den Fall $f(x, y) \equiv 0$ lautet (9.6)

$$y_{n+2} + 4y_{n+1} - 5y_n = 0. \tag{9.7}$$

Dies ist eine lineare homogene Differenzengleichung mit konstanten Koeffizienten. Der Ansatz $y_n = \lambda^n$ für eine Lösung von (9.7) führt auf die Bedingung

$$\lambda^{n+2} + 4\lambda^{n+1} - 5\lambda^n = 0$$

n	y_n
0	1.00000000000000000
1	1.00000000000000111
2	0.99999999999999556
3	1.00000000000002331
4	0.99999999999988454
5	1.00000000000057843
6	0.99999999999710898
7	1.00000000001445621
8	0.99999999992772004
9	1.00000000036140091
10	0.99999999819299656
16	0.99997176556842504
17	1.00014117215787590
18	0.99929413921062160
19	1.00352930394689310
20	0.98235348026553560
21	1.08823259867232314
22	0.55883700663838543
⋮	⋮
34	$-1.077058079E+0008$
35	$5.385290456E+0008$

Tabelle 9.1: AWA mit Anfangsfehler 10^{-15} ($y \equiv 1$)

d.h. $\lambda^2 + 4\lambda - 5 = 0$ mit den Lösungen $\lambda_1 = 1$, $\lambda_2 = -5$.

Da (9.7) linear und homogen ist, ist auch

$$y_n = A\lambda_1^n + B\lambda_2^n = A + B(-5)^n, \quad A, B \in \mathbb{R},$$

eine Lösung (sogar die allgemeine Lösung). Die Konstanten A und B kann man aus dem Anfangsfeld y_0 und y_1 bestimmen. Man erhält

$$y_n = \frac{1}{6}(5y_0 + y_1) + (-5)^n \frac{1}{6}(y_0 - y_1).$$

Der zweite Term hiervon führt dazu, dass sich die Fehler (bei alternierendem Vorzeichen) aufschaukeln.

Im allgemeinen Fall (9.2) hätte man statt (9.7) für $f(x, y) \equiv 0$ die Differenzengleichung $\sum_{\nu=0}^{k} a_\nu \, y_{n+\nu} = 0$ mit der charakteristischen Gleichung

$$\rho(\lambda) := \sum_{\nu=0}^{k} a_\nu \, \lambda^\nu = 0.$$

Sind $\lambda_1, \ldots, \lambda_r$ die verschiedenen Nullstellen von ρ mit den Vielfachheiten $m_1, \ldots, m_r$, so sind alle Lösungen von

$$\sum_{\nu=0}^{k} a_\nu \, y_{n+\nu} = 0$$

Linearkombinationen von $\lambda_j^n, n\lambda_j^n, \ldots, n^{m_j-1}\lambda_j^n$, $j = 1, \ldots, r$ (vgl. die allgemeine Lösung der homogenen Differentialgleichung mit konstanten Koeffizienten).

Fehler im Anfangsfeld $y_0, \ldots, y_{k-1}$ werden daher nicht verstärkt, wenn $|\lambda_j| \leqslant 1$ für alle Nullstellen λ_j von ρ gilt und die Nullstellen mit $|\lambda_j| = 1$ einfach sind.

Definition 9.2 Das Mehrschrittverfahren (9.2) heißt **stabil**, wenn für alle Nullstellen λ_j des charakteristischen Polynoms $\rho(\lambda)$ gilt $|\lambda_j| \leqslant 1$ und wenn die Nullstellen mit $|\lambda_j| = 1$ einfach sind.

Wegen der Konsistenzbedingung ist stets $\lambda = 1$ eine Nullstelle von ρ. Gilt $|\lambda_j| < 1$ für alle anderen Nullstellen λ_j von ρ, so heißt das Verfahren **stark stabil**.

Die obigen Überlegungen zeigen, dass die Stabilität neben der Konsistenz die Mindestanforderung an ein k-Schritt Verfahren ist. Umgekehrt kann man zeigen, dass konsistente und stabile Verfahren konvergieren (vgl. *Hairer, Nørsett, Wanner*[38], p. 395). Dabei müssen wir die Definition der Konvergenz gegenüber den Einschrittverfahren nun ein wenig modifizieren, da das Verfahren (9.2) nicht nur von dem in (9.1) gegebenen Anfangswert y_0 abhängt, sondern auch von dem gewählten Anfangsfeld $y_1, \ldots, y_{k-1}$.

Definition 9.3 Das lineare k-Schritt Verfahren (9.2) heißt **konvergent**, wenn für jedes Anfangswertproblem (9.1)

$$y(x) - y_n \to 0 \quad \text{für } h \to 0 \text{ und } n \to \infty \text{ mit } x_0 + nh \to x$$

gilt für alle Anfangsfelder $y_1(h), \ldots, y_{k-1}(h)$ mit

$$y(x_0 + jh) - y_j(h) \to 0 \quad \text{für } h \to 0,\ j = 1, \ldots, k-1.$$

Das Verfahren heißt konvergent von der Ordnung p, wenn für jede Anfangswertaufgabe (9.1) mit genügend glatter rechter Seite f es ein $h_0 > 0$ gibt mit

$$\|y(x_0 + jh) - y_j(h)\| \leqslant Ch^p \quad \text{für } h \leqslant h_0$$

für alle Anfangsfelder mit

$$\|y(x_0 + jh) - y_j(h)\| \leqslant C_0 h^p \quad \text{für } h \leqslant h_0 \text{ und } j = 1, \ldots, k-1.$$

Fordert man in (9.5) neben $b_2 = 0$ (Explizitheit), dass $\rho(\lambda)$ die Nullstellen $\lambda_1 = 1$ (Konsistenz) und $\lambda_2 = 0$ (um die Stabilität zu erzwingen) besitzt, so kann man nur die ersten drei Gleichungen von (9.5) erfüllen und erhält das explizite Verfahren der Ordnung 2:

$$y_{n+2} = y_{n+1} + \frac{h}{2}(-f_n + 3f_{n+1}).$$

9.3 Adams-Bashforth Verfahren

Wir geben nun einen Weg an, wie man stark stabile Mehrschrittverfahren konstruieren kann. In Übereinstimmung mit der Literatur numerieren wir dabei nun die an der Mehrschrittformel beteiligten Näherungswerte mit $y_{n-k+1}, \ldots, y_n, y_{n+1}$, wobei $y_{n-k+1}, \ldots, y_n$ als aus den vorhergehenden Schritten bekannt angenommen werden und y_{n+1} in dem aktuellen Schritt zu bestimmen ist.

Für die Lösung y der Anfangswertaufgabe (9.1) gilt

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} y'(t)\,dt = \int_{x_n}^{x_{n+1}} f(t, y(t))\,dt. \tag{9.8}$$

Wir ersetzen daher bei gegebenen Näherungen $y_j \approx y(x_j)$, $j = n, n-1, \ldots, n-k+1$, und damit bekannten Näherungen

$$f_j := f(x_j, y_j) \approx f(x_j, y(x_j)) = y'(x_j),\ j = n, n-1, \ldots, n-k+1,$$

die Funktion y' im Integranden durch ihr Interpolationspolynom

$$p \in \Pi_{k-1} \ : \ p(x_j) = f_j, \ j = n, n-1, \ldots, n-k+1,$$

und berechnen die neue Näherung gemäß

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} p(t)\, dt.$$

An der Lagrangeschen Integrationsformel

$$p(x) = \sum_{j=0}^{k-1} f_{n-j} \cdot \ell_j(x), \ \ell_j(x) := \prod_{\substack{i=0 \\ i \neq j}}^{k-1} (x - x_{n-i}) \Big/ \prod_{\substack{i=0 \\ i \neq j}}^{k-1} (x_{n-j} - x_{n-i}),$$

erkennt man, dass

$$y_{n+1} = y_n + \sum_{j=0}^{k-1} f_{n-j} \int_{x_n}^{x_{n+1}} \ell_j(t)\, dt$$

tatsächlich die Gestalt eines k-Schritt Verfahrens hat.

Mit der Variablentransformation $t := x_n + h \cdot s$ erhält man

$$\int_{x_n}^{x_{n+1}} \ell_j(t)\, dt = h \cdot \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{k-1} (i + s) \Big/ \prod_{\substack{i=0 \\ i \neq j}}^{k-1} (i - j)\, ds =: h\alpha_j.$$

Die Integrale über das Interpolationspolynom lassen sich also schreiben als

$$\int_{x_n}^{x_{n+1}} p(t)\, dt = h \cdot \sum_{j=0}^{k-1} \alpha_j f_{n-j},$$

wobei die Koeffizienten α_j unabhängig von den y_j und von den speziellen Knoten x_j und der Schrittweite h sind, und daher in Dateien bereitgestellt werden können.

Die Mehrstellenformel erhält damit die Gestalt

$$y_{n+1} = y_n + h \cdot \sum_{j=0}^{k-1} \alpha_j f_{n-j}.$$

Das charakteristische Polynom ist (man beachte die geänderte Numerierung der Koeffizienten in der k-Schritt Formel)

$$\rho(\lambda) = \lambda^k - \lambda^{k-1}$$

mit der einfachen Nullstelle $\lambda = 1$ und der $(k-1)$-fachen Nullstelle 0. Die Mehrstellenformel ist also stark stabil.

So konstruierte Mehrstellenformeln heißen **Adams-Bashforth Verfahren**. Sie sind explizit und aus der Fehlerdarstellung des Interpolationspolynoms erhält man, dass ihre Ordnung k ist. Die ersten Adams-Bashforth Formeln sind:

$$\begin{array}{ll}
k = 1: & y_{n+1} = y_n + h f_n \\
k = 2: & y_{n+1} = y_n + 0.5h(3f_n - f_{n-1}) \\
k = 3: & y_{n+1} = y_n + h(23f_n - 16f_{n-1} + 5f_{n-2})/12 \\
k = 4: & y_{n+1} = y_n + h(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3})/24 \\
k = 5: & y_{n+1} = y_n + h(1901f_n - 2774f_{n-1} + 2616f_{n-2} - 1274f_{n-3} + 251f_{n-4})/720
\end{array}$$

Für $k = 1$ ergibt sich also das Eulersche Polygonzugverfahren.

9.4 Adams-Moulton Verfahren

Nachteil der Adams-Bashforth Formeln ist, dass bei ihrer Konstruktion das Interpolationspolynom p im Intervall $[x_n, x_{n+1}]$ verwendet wird, während die Interpolationsknoten außerhalb dieses Intervalls liegen. Wir wissen bereits, dass der Fehler eines Interpolationspolynoms außerhalb des kleinsten Intervalls $[x_{n-k+1}, x_n]$, das alle Knoten enthält, sehr schnell anwächst. Es ist daher naheliegend, die Funktion y' in (9.8) durch das Interpolationspolynom

$$p \in \Pi_k \ : \ p(x_j) = f(x_j, y_j), j = n+1, n, n-1, \ldots, n-k+1$$

zu ersetzen.

Wie eben kann man das Verfahren schreiben als

$$y_{n+1} = y_n + h \sum_{j=0}^{k} \beta_j f_{n+1-j}$$

mit

$$\beta_j := \frac{1}{h} \int_{x_n}^{x_{n+1}} \prod_{\substack{i=0 \\ i \neq j}}^{k} (t - x_{n+1-i}) \Big/ \prod_{\substack{i=0 \\ i \neq j}}^{k} (x_{n+1-j} - x_{n+1-i}) \, dt.$$

Diese Verfahren heißen **Adams-Moulton Verfahren**. Sie sind wie die Adams-Bashforth Verfahren stark stabil und haben die Ordnung $k+1$ (Beachten Sie, dass der Grad des Interpolationspolynoms hier k ist, beim Adams-Bashforth Verfahren aber nur $k-1$). Die ersten Adams-Moulton Formeln sind:

$$\begin{array}{ll}
k = 0: & y_{n+1} = y_n + h f_{n+1} \\
k = 1: & y_{n+1} = y_n + 0.5h(f_{n+1} + f_n) \\
k = 2: & y_{n+1} = y_n + h(5f_{n+1} + 8f_n - f_{n-1})/12 \\
k = 3: & y_{n+1} = y_n + h(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})/24 \\
k = 4: & y_{n+1} = y_n + h(251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3})/720.
\end{array}$$

und für $k = 5$

$$y_{n+1} = y_n + h(475f_{n+1} + 1427f_n - 798f_{n-1} + 482f_{n-2} - 173f_{n-3} + 27f_{n-4})/1440.$$

Das Adams-Moulton Verfahren mit $k = 1$ heißt das **implizite Euler Verfahren**, dasjenige für $k = 2$ die **Trapezregel**.

Die Adams-Moulton Verfahren haben wesentlich bessere Konvergenzeigenschaften als die Adams-Bashforth Verfahren gleicher Ordnung. Nachteilig ist, dass sie implizit sind, man also in jedem Schritt ein nichtlineares Gleichungssystem zu lösen hat.

9.5 Prädiktor-Korrektor Verfahren

Man kombiniert daher beide Verfahren zu einem **Prädiktor-Korrektor Verfahren**: Sind bereits Näherungen $y_j = y(x_j)$, $j = 0, \ldots, n$, bekannt ($n \geqslant k$), so bestimme man mit dem Adams-Bashforth Verfahren der Ordnung k eine vorläufige Näherung

$$\tilde{y}_0 := y_n + h \sum_{j=0}^{k-1} \alpha_j f_{n-j}$$

für $y(x_{n+1})$ und verbessere diese iterativ unter Benutzung der Adams-Moulton Formel der Ordnung $k+1$:

$$\tilde{y}_{i+1} = y_n + h\Big(\beta_0 f(x_{n+1}, \tilde{y}_i) + \sum_{j=1}^{k} \beta_j f_{n+1-j}\Big),\ i = 0, 1, \ldots.$$

Erfüllt f eine Lipschitz Bedingung und ist h genügend klein gewählt, so ist diese Iteration konvergent. In der Regel genügen ein oder zwei Verbesserungsschritte (sonst ist die Schrittweite h zu groß). Das so gefundene $\tilde{y}_1$ oder $\tilde{y}_2$ wird als y_{n+1} gewählt und es wird der nächste Prädiktor-Korrektor-Schritt ausgeführt.

Eine typische Implementierung eines Prädiktor-Korrektor Verfahrens hat also die folgende Gestalt: P (Auswertung der Prädiktorformel) E (Evaluation von f) C (Auswertung der Korrektorformel) E (Evaluation von f) oder PECECE. Mit den Adams-Bashforth und Adams-Moulton Formeln für $k = 1$ (dem expliziten Euler Verfahren und der Trapezregel) erhält man also das PECECE Verfahren

$$\begin{aligned}
Py_{n+1}^{[1]} &= y_n + hf_n \\
Ef_{n+1}^{[1]} &= f(x_{n+1}, y_{n+1}^{[1]}) \\
Cy_{n+1}^{[2]} &= y_n + 0.5h(f_{n+1}^{[1]} + f_n) \\
Ef_{n+1}^{[2]} &= f(x_{n+1}, y_{n+1}^{[2]}) \\
Cy_{n+1} &= y_n + 0.5h(f_{n+1}^{[2]} + f_n) \\
Ef_{n+1} &= f(x_{n+1}, y_{n+1}^{[2]}).
\end{aligned}$$

Vorteil der Mehrschrittverfahren ist, dass auch bei größeren Ordnungen nur in jedem Schritt eine Funktionsauswertung von f im expliziten Fall bzw. 2 oder 3 Auswertungen beim Prädiktor-Korrektor Verfahren benötigt werden, während beim Einschrittverfahren die Zahl der Funktionsauswertungen bei Steigerung der Ordnung sehr rasch wächst. Mehrschrittverfahren werden daher vor allem verwendet, wenn die Auswertung von f sehr teuer ist.

Beispiel 9.4 Wir betrachten erneut die Anfangswertaufgabe

$$y' = y^2,\ y(0.8) = 5/6,\ 0.8 \leqslant x \leqslant 1.8.$$

Mit dem klassischen Runge-Kutta Verfahren mit äquidistanter Schrittweite $h = 1/64$ ist der Fehler im Endpunkt 1.8 des Intervalls $2.55 \cdot 10^{-6}$. Hierzu muss die rechte Seite an $4 \cdot 64 = 256$ Stellen ausgewertet werden.

Mit dem PECE Verfahren mit den Adams-Bashforth und Adams-Moulton Formeln für $k = 5$ erhält man einen vergleichbaren Fehler $2.53 \cdot 10^{-6}$ mit der Schrittweite $h = 1/80$. Hierzu benötigt man zur Bestimmung des Anfangsfeldes y_1, y_2, y_3, y_4 mit dem klassischen Runge-Kutta Verfahren 16 Funktionsauswertungen und für die weiteren 76 Schritte je zwei Auswertungen, isgesamt also 168. Das Prädiktor-Korrektor Verfahren erfordert also wesentlich geringeren Aufwand. □

Nachteil der Mehrschrittverfahren ist, dass die Schrittweitensteuerung komplizierter als beim Einschrittverfahren ist. Man muss

- entweder nicht äquidistante Knoten $x_n, x_{n-1}, \ldots, x_{n-k+1}$ verwenden und kann dann die α_j bzw. β_j nicht einer Tabelle entnehmen, sondern muss sie nach jeder Veränderung der Schrittweite während der nicht äquidistanten Phase neu berechnen
- oder bei geänderter Schrittweite $\tilde{h}$ Näherung für $y(x_n - j \cdot \tilde{h})$ aus einem Interpolationspolynom berechnen.

Der zweite Zugang wird in Hairer, Nørsett, Wanner [38] zusammen mit einer kombinierten Schrittweiten- und Ordnungskontrolle diskutiert.

In MATLAB 6.1 ist als Funktion ODE113 ein PECE Verfahren von Adams-Bashforth-Moulton zur Lösung von nicht-steifen Anfangswertaufgaben implementiert.

9.6 BDF-Verfahren

Die bisherigen Mehrschrittverfahren basierten auf der numerischen Lösung der Integralgleichung (9.8). Die folgende Klasse von Verfahren wird mit Hilfe der numerischen Differentiation konstruiert.

Es seien bereits Näherungen $y_{n-k+1}, \dots, y_n$ der Lösung der Anfangswertaufgabe (9.1) an den Knoten $x_{n-k+1}, \dots, x_n$ bekannt. Um $y_{n+1} \approx y(x_{n+1})$ zu bestimmen betrachten wir das Interpolationspolynom q zu den Daten (x_{n-k+1}, y_{n-k+1}), $\dots$, (x_n, y_n), (x_{n+1}, y_{n+1}). Dann kann man q nach der Newtonschen Darstellung des Interpolationspolynoms mit den **rückwärtsgenommenen Differenzen**

$$\nabla^0 y_n := y_n, \ \nabla^{j+1} y_n := \nabla^j y_n - \nabla^j y_{n-1}$$

schreiben als

$$q(s) = y(x_n + sh) = \sum_{j=0}^{k} (-1)^j \binom{-s+1}{j} \nabla^j y_{n+1}. \tag{9.9}$$

Der Unbekannte Wert y_{n+1} wird nun so bestimmt, dass das Polynom q die Differentialgleichung an einem Gitterpunkt erfüllt:

$$q'(x_{n+\ell}) = f(x_{n+\ell}, y_{n+\ell}), \ \ell \in \{0, 1, \dots\}. \tag{9.10}$$

Für $\ell = 0$ erhält man explizite Formeln, und zwar für $k = 1$ das explizite Euler Verfahren und für $k = 2$ die Mittelpunktregel. Die Formeln für $k \geqslant 3$ sind instabil und daher wertlos.

Für $\ell = 1$ erhält man implizite Formeln, die **BDF Methoden** (backward differentiation formulas)

$$\sum_{j=0}^{k} \alpha_j \nabla^j y_{n+1} = h f_{n+1}$$

mit den Koeffizienten

$$\alpha_j = (-1)^j \frac{d}{ds} \binom{-s+1}{j} \Big|_{s=1}.$$

Mit

$$(-1)^j \binom{-s+1}{j} = \frac{1}{j!}(s-1)s(s+1)\dots(s+j-2)$$

folgt

$$\alpha_0 = 0, \ \alpha_j = \frac{1}{j} \text{ für } j \geqslant 1,$$

und daher

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = h f_{n+1}. \tag{9.11}$$

Für $k \leqslant 6$ rechnet man leicht aus, dass gilt

$$
\begin{aligned}
&k = 1 : y_{n+1} - y_n = hf_{n+1} \\
&k = 2 : 3y_{n+1} - 4y_n + y_{n-1} = 2hf_{n+1} \\
&k = 3 : 11y_{n+1} - 18y_n + 9y_{n-1} - 2y_{n-2} = 6hf_{n+1} \\
&k = 4 : 25y_{n+1} - 48y_n + 36y_{n-1} - 16y_{n-2} + 3y_{n-3} = 12hf_{n+1} \\
&k = 5 : 137y_{n+1} - 300y_n + 300y_{n-1} - 200y_{n-2} + 75y_{n-3} - 12y_{n-4} = 60hf_{n+1} \\
&k = 6 : 147y_{n+1} - 360y_n + 450y_{n-1} - 400y_{n-2} + 225y_{n-3} - 72y_{n-4} + 10y_{n-5} = 60hf_{n+1}.
\end{aligned}
$$

Durch Diskussion des Polynoms

$$\rho(\lambda) = \sum_{j=1}^{k} \frac{1}{j} \lambda^{k-j} (\lambda - 1)^j$$

sieht man für $k \leqslant 6$ leicht, dass diese BDF-Formeln stabil sind. Für $k \geqslant 7$ sind sie instabil (vgl. *Hairer, Nørsett, Wanner* [38], p. 380).

Kapitel 10

Steife Probleme

10.1 Motivation

Es gibt Differentialgleichungen mit Lösungen, zu deren Approximation bei Anwendung expliziter Verfahren viel kleinere Schrittweiten benötigt werden, als man erwartet. Diese Probleme werden **steif** genannt.

Beispiel 10.1 Die Anfangswertaufgabe

$$y' = -\lambda(y - e^{-x}) - e^{-x}, \quad y(0) = 1 \tag{10.1}$$

besitzt für alle $\lambda \in \mathbb{R}$ die eindeutige Lösung $y(x) = e^{-x}$. Tabelle 10.1 und Tabelle 10.2 enthalten die Fehler der Näherungslösungen bei konstanter Schrittweite $h = 0.01$ für das Polygonzugverfahren, das verbesserte Polygonzugverfahren und das klassische Runge-Kutta Verfahren für die Parameter $\lambda = 1$ und $\lambda = 1000$.

Abbildung 10.1 und Abbildung 10.2 zeigen die Lösungen der Anfangswertaufgaben

$$y' = -\lambda(y - e^{-x}) - e^{-x}, \; y(x_0) = y_0$$

für verschiedene Werte von x_0 und y_0 für $\lambda = 1$ und $\lambda = 20$.

Wir möchten die Lösung $y(x) = e^{-x}$ von (10.1) für großes λ mit einem numerischen Verfahren verfolgen. Durch (Rundungs- oder Verfahrens-) Fehler werden wir ein wenig von der Lösung weggeführt. Ist $y_n \neq e^{-x_n}$, so konvergiert die Lösung

$$y(x; x_n, y_n) = (y_n - e^{-x_n})e^{-\lambda(x-x_n)} + e^{-x}$$

x	Polygonzug	verb. Polygonzug	Runge-Kutta
0.00	$0.00E+0$	$0.00E+0$	$0.00E+0$
0.10	$-4.55E-4$	$1.52E-6$	$7.60E-12$
0.20	$-8.24E-4$	$2.75E-6$	$1.38E-11$
0.30	$-1.12E-3$	$3.73E-6$	$1.87E-11$
0.40	$-1.35E-3$	$4.50E-6$	$2.25E-11$
0.50	$-1.52E-3$	$5.09E-6$	$2.55E-11$
0.60	$-1.66E-3$	$5.53E-6$	$2.77E-11$
0.70	$-1.75E-3$	$5.83E-6$	$2.92E-11$
0.80	$-1.81E-3$	$6.04E-6$	$3.02E-11$
0.90	$-1.84E-3$	$6.14E-6$	$3.07E-11$
1.00	$-1.85E-3$	$6.18E-6$	$3.09E-11$

Tabelle 10.1: Fehler für $\lambda = 1$

x	Polygonzug	verb. Polygonzug	Runge-Kutta
0.00	$0.00E+0$	$0.00E+0$	$0.00E+0$
0.01	$-4.98E-5$	$1.25E-4$	$1.04E-3$
0.02	$3.99E-4$	$5.24E-3$	$3.04E-1$
0.03	$-3.64E-3$	$2.15E-1$	$8.83E+1$
0.04	$3.27E-2$	$8.82E+0$	$2.57E+4$
0.05	$-2.95E-1$	$3.61E+2$	$7.48E+6$
0.06	$2.65E+0$	$1.48E+4$	$2.18E+9$
0.07	$-2.39E+1$	$6.08E+5$	$6.33E+11$
0.08	$2.15E+2$	$2.49E+7$	$1.84E+14$
0.09	$-1.93E+3$	$1.02E+9$	$5.36E+16$
0.10	$1.74E+4$	$4.19E+10$	$1.56E+19$

Tabelle 10.2: Näherungen für $\lambda = 1000$

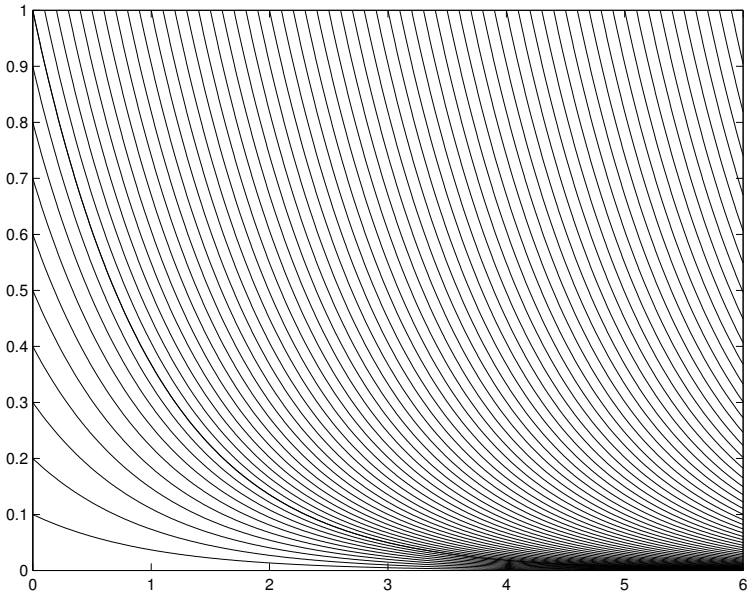

Abbildung 10.1: Lösungen für $\lambda = 1$

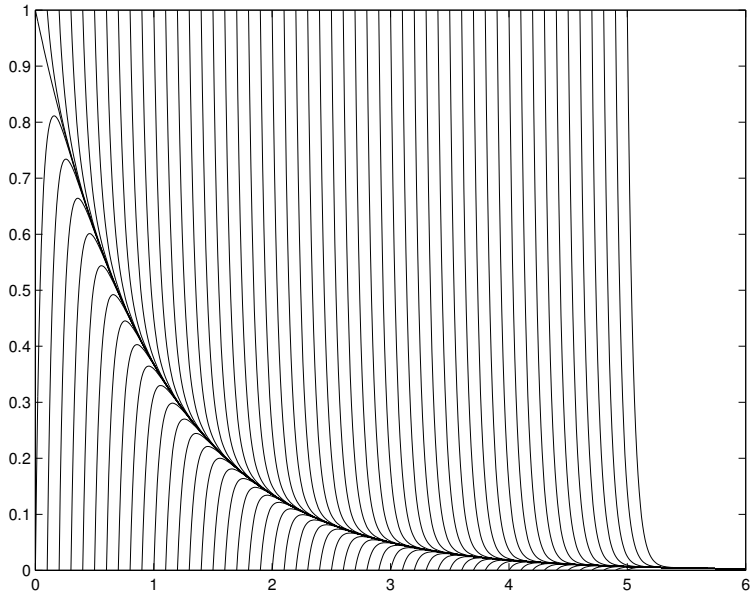

Abbildung 10.2: Lösungen für $\lambda = 20$

der Anfangswertaufgabe für großes λ sehr rasch gegen die quasi stationäre Lösung $\tilde{y}(x) = e^{-x}$. Die dem Betrage nach sehr große Steigung

$$y'(x_n; x_n, y_n) = -\lambda(y_n - e^{-x_n}) - e^{-x_n}$$

führt dazu, dass für das Euler Verfahren (und für die anderen Einschrittverfahren genauso) über das Ziel hinausgeschossen wird und die Fehler sich aufschaukeln. □

10.2 Stabilitätsgebiete

Wendet man das Eulersche Polygonzugverfahren auf die Testgleichung

$$y' = \lambda y, \ \lambda < 0, \tag{10.2}$$

an, so erhält man bei konstanter Schrittweite $h > 0$

$$y_{n+1} = y_n + h\lambda y_n,$$

und daher

$$y_n = (1 + h\lambda)^n y_0.$$

Für

$$|1 + h\lambda| > 1$$

explodiert die numerische Lösung y_n, und zwar ist dieses Aufschaukeln um so rascher, je kleiner λ ist, je schneller die Lösung der Anfangswertaufgabe also abklingt. Das Mindeste, was man von einem Verfahren erwarten muss, ist aber, dass die numerische Lösung bei nicht zu kleinen Schrittweiten ebenfalls abklingt.

Das einfachste Verfahren, dessen numerische Lösung der Testgleichung (10.2) bei annehmbaren Schrittweiten das Abklingverhalten der Lösung der Anfangswertaufgabe reproduziert, ist das **implizite Euler Verfahren**

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + h\boldsymbol{f}(x_{n+1}, \boldsymbol{y}^{n+1}).$$

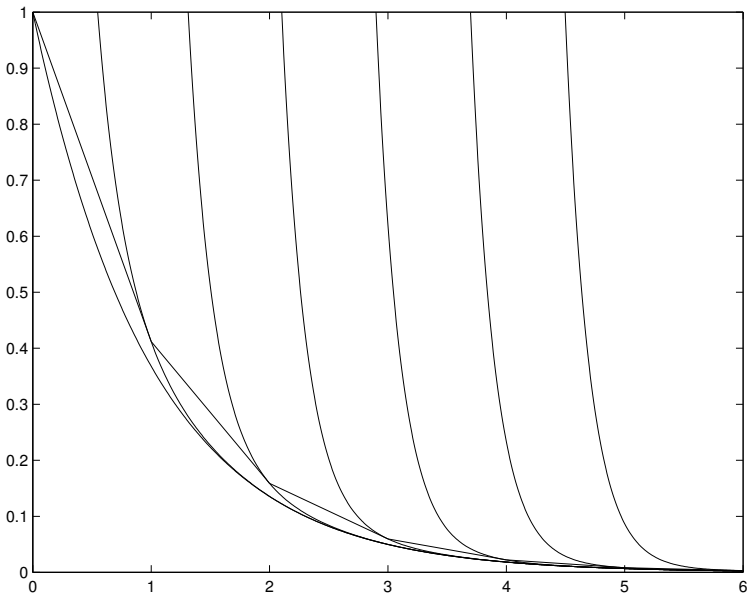

Abbildung 10.3: Implizites Euler Verfahren

Mit ihm erhält man für (10.2)

$$y_{n+1} = y_n + h\lambda y_{n+1},$$

d.h.

$$y_n = \Big(\frac{1}{1-h\lambda}\Big)^n y_0 \to 0 \quad \text{für } n \to \infty$$

für jede Schrittweite $h > 0$. Man geht also einen linearen Schritt mit der Steigung weiter, die in dem Richtungsfeld der Differentialgleichung dort herrscht, wo man hinkommt (Abbildung 10.3 zeigt 6 Schritte des impliziten Euler Verfahrens für Beispiel 10.1 mit $\lambda = -5$ und der Schrittweite $h = 1$).

Der Preis, den man für dieses verbesserte Stabilitätsverhalten zu zahlen hat, ist, dass man im allgemeinen Fall in jedem Schritt ein nichtlineares Gleichungssystem

$$\boldsymbol{F}(\boldsymbol{y}^{n+1}) = \boldsymbol{y}^{n+1} - \boldsymbol{y}^n - h\boldsymbol{f}(x_{n+1}, \boldsymbol{y}^{n+1}) = \boldsymbol{0}$$

zu lösen hat. Dies kann man z.B. mit dem Newton Verfahren mit dem Startwert $\boldsymbol{y}^n$ tun.

Bemerkung 10.2 Die Testgleichung (10.2) ist aussagekräftig für allgemeinere Systeme, denn ist die Matrix $\boldsymbol{A} \in \mathbb{R}^{(n,n)}$ in dem linearen System

$$\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y} + \boldsymbol{g} \tag{10.3}$$

diagonalisierbar und gilt

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{\Lambda} = \operatorname{diag}\{\lambda_1, \ldots, \lambda_n\}$$

mit einer regulären Matrix $\boldsymbol{X}$, so erhält man mit der Variablentransformation $\boldsymbol{z} := \boldsymbol{X}^{-1}\boldsymbol{y}$

$$\boldsymbol{z}' = \boldsymbol{X}^{-1}\boldsymbol{y}' = \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^{-1}\boldsymbol{y} + \boldsymbol{X}^{-1}\boldsymbol{g} = \boldsymbol{\Lambda}\boldsymbol{z} + \boldsymbol{X}^{-1}\boldsymbol{g} =: \boldsymbol{\Lambda}\boldsymbol{z} + \tilde{g},$$

d.h. das entkoppelte System

$$z_j' = \lambda_j z_j + \tilde{g}_j, \quad j = 1, \dots, n. \tag{10.4}$$

Wendet man auf das System (10.3) ein Mehrschrittverfahren

$$\sum_{j=0}^{m} \alpha_j \boldsymbol{y}^{k-j} + h \sum_{j=0}^{m} \beta_j (\boldsymbol{A}\boldsymbol{y}^{k-j} + \boldsymbol{g}^{k-j}) = \boldsymbol{0}$$

an, so ist dieses mit $\boldsymbol{z}^i := \boldsymbol{X}^{-1}\boldsymbol{y}^i$ äquivalent zu

$$\sum_{j=0}^{m} \alpha_j \boldsymbol{z}^{k-j} + h \sum_{j=0}^{m} \beta_j (\boldsymbol{\Lambda}\boldsymbol{z}^{k-j} + \tilde{g}^{k-j}) = \boldsymbol{0},$$

d.h. zu dem Mehrschrittverfahren

$$\sum_{j=0}^{m} \alpha_j z_{i,k-j} + h\lambda_i \sum_{j=0}^{m} \beta_j (z_{i,k-j} + \tilde{g}_{i,k-j}) = 0, \quad i = 1, \dots, n,$$

für die skalaren Gleichungen (10.4). □

Das Abklingverhalten des Mehrschrittverfahrens für das System (10.3) wird also bestimmt durch das Abklingverhalten für die skalaren Gleichungen (10.2) für die Eigenwerte $\lambda = \lambda_j$ der Matrix $\boldsymbol{A}$. Da diese nicht notwendig reell sind, auch wenn $\boldsymbol{A}$ nur reelle Einträge besitzt, müssen wir die Testgleichung (10.2) für $\lambda \in \mathbb{C}$ mit negativem Realteil untersuchen.

Wendet man das explizite Runge-Kutta Verfahren 8.17 von Seite 134 auf das System (10.3) an, so ist dieses wieder äquivalent der Anwendung auf die skalaren Gleichungen. Das Abklingverhalten der numerischen Lösung wird also wieder bestimmt durch das Abklingverhalten für die Testgleichung (10.2). Man erhält

$$\begin{aligned} k_1 &:= \lambda y_n \\ k_j &:= \lambda\Big(y_n + h \sum_{\ell=1}^{j-1} \beta_{j\ell}\, k_\ell\Big), \quad j = 2, \dots, s, \\ y_{n+1} &:= y_n + h \sum_{j=1}^{s} \gamma_j\, k_j. \end{aligned} \tag{10.5}$$

Setzt man die k_j nacheinander ein, so folgt

$$y_{n+1} = R(h\lambda) y_n$$

mit

$$R(z) = 1 + z \sum_{j=1}^{s} \gamma_j + z^2 \sum_{j=1}^{s} \sum_{k=1}^{j-1} \gamma_j \beta_{jk} + z^3 \sum_{j=1}^{s} \sum_{k,\ell=1}^{j-1} \gamma_j \beta_{jk} \beta_{k\ell} + \dots \in \Pi_s,$$

und die Folge $\{y_n\}$ ist offenbar beschränkt, wenn $|R(z)| \leqslant 1$ gilt. Beachten Sie, dass diese Bedingung nur von $z := h\lambda$ abhängt.

Wir definieren

Definition 10.3 Das **Stabilitätsgebiet** $S \subset \mathbb{C}$ eines Verfahrens ist die Menge aller $z := h\lambda \in \mathbb{C}$, so dass für alle Startwerte die erzeugte Folge $\{y_n\}$ mit der Schrittweite h für die Testgleichung (10.2) beschränkt ist.

Das Stabilitätsgebiet eines Verfahrens hat die folgende Bedeutung. Will man ein lineares Differentialgleichungssystem $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{y}$ lösen und besitzt die Matrix $\boldsymbol{A}$ die Eigenwerte λ_j, $j = 1, \dots, n$, mit $\Re\lambda_j < 0$, so kann man das System nur dann stabil mit diesem Verfahren lösen, wenn die Schrittweite h so klein gewählt ist, dass die Zahlen $h\lambda_j$ für alle $j = 1, \dots, n$ in dem Stabilitätsgebiet S des Verfahrens liegen.

10.2.1 A-Stabilität

Wünschenswert ist für ein Verfahren, dass sein Stabilitätsgebiet die linke Halbebene umfasst.

Definition 10.4 Ein Verfahren zur Lösung von Anfangswertaufgaben heißt **A-stabil**, wenn gilt

$$S \supset \mathbb{C}_- := \{z \in \mathbb{C} : \Re z \leqslant 0\}.$$

Für das explizite Runge-Kutta Verfahren (8.17) von Seite 134 der Stufe s ist die Funktion R ein Polynom vom Höchstgrad s. Das Stabilitätsgebiet

$$S = \{z \in \mathbb{C} : |R(z)| \leqslant 1\}$$

ist daher mit Sicherheit eine beschränkte Menge in $\mathbb{C}$. Explizite Runge-Kutta Verfahren sind also niemals A-stabil.

Beispiel 10.5 Für das (explizite) Polygonzugverfahren ist das Stabilitätsgebiet

$$S_{\text{expliziter Euler}} = \{z \in \mathbb{C} : |1 + z| \leqslant 1\},$$

für das implizite Euler Verfahren

$$S_{\text{impliziter Euler}} = \{z \in \mathbb{C} : |1 - z| \geqslant 1\} \supset \mathbb{C}_-.$$

Das implizite Euler Verfahren ist also A-stabil. □

Beispiel 10.6 Wendet man das verbesserte Polygonzugverfahren auf die Testgleichung an, so erhält man

$$y_{n+1} = y_n + h\lambda\Big(y_n + \frac{1}{2}h\lambda y_n\Big) = \Big(1 + (h\lambda) + \frac{1}{2}(h\lambda)^2\Big)y_n.$$

Es ist also

$$R(z) = 1 + z + \frac{1}{2}z^2.$$

Für das Verfahren von Heun erhält

$$y_{n+1} = y_n + \frac{h}{2}\Big(\lambda y_n + \lambda(y_n + h\lambda y_n)\Big) = \Big(1 + h\lambda + \frac{1}{2}(h\lambda)^2\Big)y_n,$$

d.h. dieselbe Funktion R wie für das verbesserte Polygonzugverfahren. Das Stabilitätsgebiet $S = \{z \in \mathbb{C} : |R(z)| \leqslant 1\}$ ist in Abbildung 10.4 dargestellt.

Dass man für das verbesserte Polygonzugverfahren und das Verfahren von Heun dieselbe Funktion R erhalten hat, ist kein Zufall. Besitzt das Runge-Kutta Verfahren die Ordnung p, so gilt

$$R(z) = \sum_{j=0}^{p} \frac{z^j}{j!} + O(z^{p+1}) \quad \text{für } z \to 0, \tag{10.6}$$

denn die Lösung der Anfangswertaufgabe $y' = y$, $y(0) = 1$, ist $y(x) = e^x$, und daher muss für die numerische Lösung $y_1 = R(h)y_0 = R(h)$ im ersten Schritt gelten

$$e^z - R(z) = e^h - R(h) = O(h^{p+1}) = O(z^{p+1}).$$

Da R ein Polynom ist, folgt hieraus (10.6).

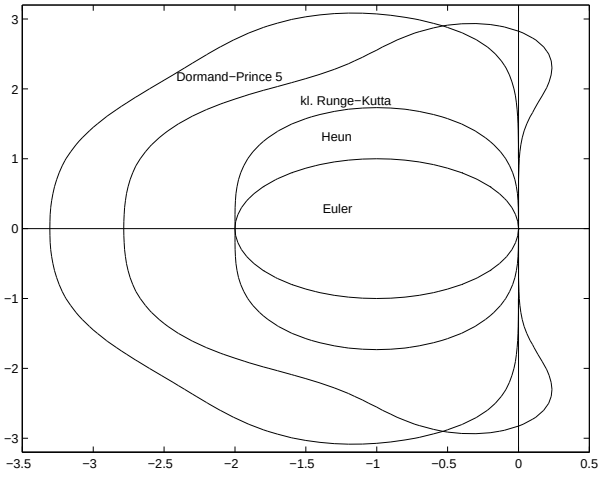

Abbildung 10.4: Stabilitätsgebiete von Runge-Kutta Verfahren

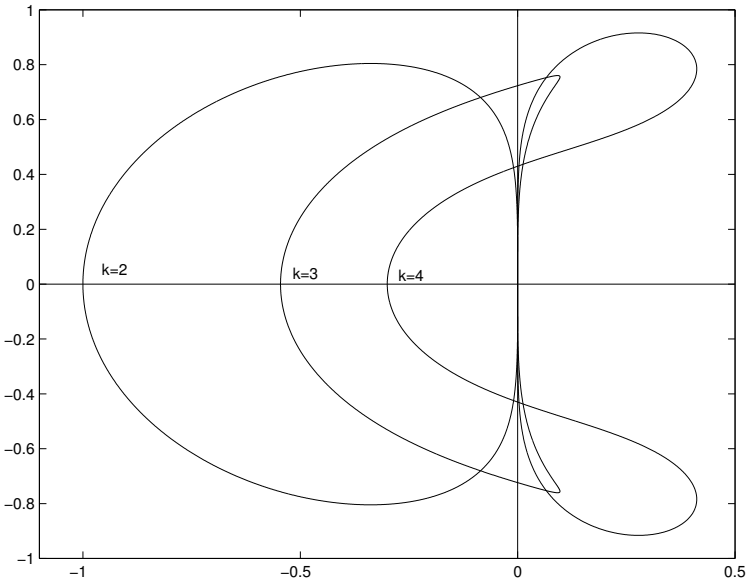

Abbildung 10.5: Stabilitätsgebiete von Adams-Bashforth Verfahren

Gilt zusätzlich $s = p$, so ist R ein Polynom vom Grade p, und es folgt

$$R(z) = \sum_{j=0}^{p} \frac{z^j}{j!}.$$

Für das klassische Runge-Kutta Verfahren, die 3/8-Regel und das Verfahren von Kuntzmann erhält man also

$$R(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4.$$

Auch hierfür findet man das Stabilitätsgebiet in Abbildung 10.4.

Das Verfahren von Dormand und Prince ist von der Ordnung $p = 5$ und benötigt $s = 6$ Stufen (die siebte Stufe wird nur für die Fehlerschätzung verwendet). Die Funktion R hat also die Gestalt

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \alpha z^6,$$

und durch direkte Rechnung erhält man $\alpha = \frac{1}{600}$. Auch hierfür findet man das Stabilitätsgebiet in Abbildung 10.4. □

Beispiel 10.7 Auch für die Adams-Bashforth Verfahren sind die Stabilitätsgebiete beschränkt. Für $k = 2, 3, 4$ sind sie in Abbildung 10.5 dargestellt. Die Stabilitätsgebiete werden also sehr rasch kleiner.

Wir betrachten nun die Adams-Moulton Verfahren

$$y_{n+1} - y_n = h \sum_{\nu=0}^{k} \beta_\nu f_{n+1-\nu}.$$

Für $k = 0$ hat man das implizite Euler Verfahren, das A-stabil ist. Auch die **Trapezregel** (k=1),

$$y_{n+1} = y_n + 0.5h(f_{n+1} + f_n) \tag{10.7}$$

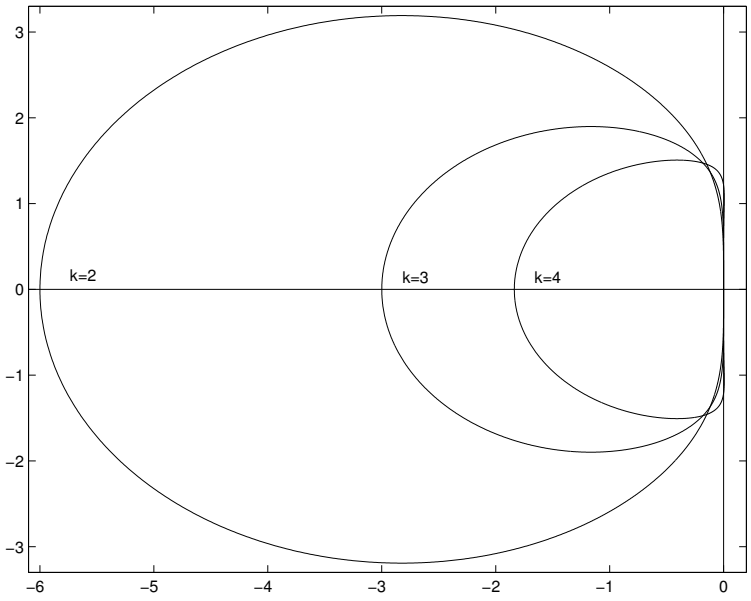

Abbildung 10.6: Stabilitätsgebiete von Adams-Moulton Verfahren

ist A-stabil, denn wendet man die Trapezregel auf die Testgleichung an, so erhält man

$$y_{n+1} = y_n + 0.5h(\lambda y_{n+1} + \lambda y_n),$$

d.h.

$$y_{n+1} = \frac{h\lambda - (-2)}{h\lambda - 2} y_n,$$

und damit ist

$$S_{\text{Trapezregel}} = \{z \in \mathbb{C} : \left|\frac{z+2}{z-2}\right| \leqslant 1\} = C_-.$$

Für $k = 2, 3, 4$ enthält Abbildung 10.6 die Stabilitätsgebiete der Adams-Moulton Verfahren. Diese sind wie die der Adams-Bashforth Verfahren beschränkt, ihre Größe nimmt aber wesentlich langsamer ab. □

Für die BDF-Verfahren

$$\sum_{\nu=1}^{k} \frac{1}{\nu} \nabla^\nu y_{n+1} = h f_{n+1}$$

erhält man

$$z = \sum_{\mu=1}^{k} \frac{1}{\mu} \Big(1 - \frac{1}{\zeta}\Big)^\mu,$$

und hiermit die Stabilitätsgebiete in Abbildung 10.7.

10.2.2 A(α)-Stabilität

Für $k \geqslant 3$ sind die BDF-Verfahren wieder nicht A-stabil. Sie haben aber die Eigenschaft, dass ein Sektor

$$S_\alpha := \{z \in \mathbb{C} : |\arg(-z)| \leqslant \alpha\} \tag{10.8}$$

in der komplexen Ebene im Stabilitätsgebiet enthalten ist.

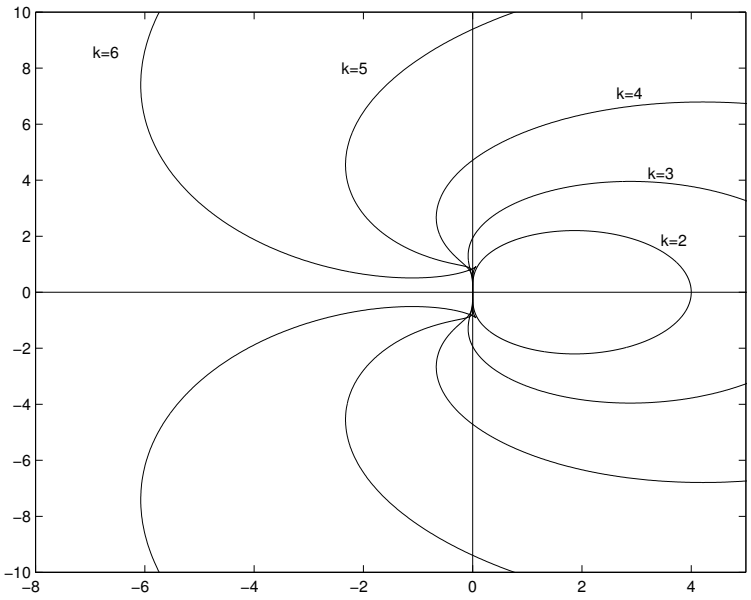

Abbildung 10.7: Stabilitätsgebiete von BDF-Verfahren

Bei vielen Anwendungen weiß man, dass die Eigenwerte der Linearisierung in einem festen Sektor der Gestalt (10.8) liegen, so dass man stabiles Verhalten der numerischen Lösung für alle Schrittweiten erwarten kann, wenn das Stabilitätsgebiet des benutzten Verfahrens diesen Sektor umfasst. Man betrachtet daher als Abschwächung der A-Stabilität

Definition 10.8 Ein Verfahren heißt **A(α)-stabil**, wenn sein Stabilitätsgebiet den Sektor der Gestalt (10.8) enthält. Es heißt **A(0)-stabil**, wenn es A(α)-stabil für ein $\alpha > 0$ ist.

Die BDF-Verfahren sind für $3 \leqslant k \leqslant 6$ A(α)-stabil mit den Öffnungswinkeln

k	1	2	3	4	5	6
α	90°	90°	86.0°	73.3°	51.8°	17.8°

Wegen dieser (gegenüber den Einschrittverfahren und den Adams-Bashforth und Adams-Moulton Verfahren) wesentlich besseren Stabilitätseigenschaften wurde die BDF-Verfahren schon früh für steife Probleme vorgeschlagen (vgl. *Curtiss, Hirschfelder* [14]). Eine Reihe von Codes basiert auf diesen Methoden.

Wir haben bisher als A-stabile Verfahren nur das implizite Euler Verfahren (der Ordnung 1) und die Trapezregel (der Ordnung 2) gefunden. Der nächste Satz von *Dahlquist* [15] zeigt, dass man keine besseren A-stabilen Mehrschrittverfahren finden kann. Einen Beweis findet man in *Hairer, Wanner* [39], p. 247 ff.

Satz 10.9 (Dahlquist) *1. Explizite Mehrschrittverfahren sind niemals A-stabil.*

2. Die Ordnung eines A-stabilen impliziten Mehrschrittverfahrens ist höchstens 2*.*

3. Die Trapezregel (10.7) *ist das A-stabile Verfahren der Ordnung* 2 *mit der kleinsten Fehlerkonstanten.*

10.3 Implizite Runge-Kutta Verfahren

A-stabile Verfahren (oder wenigstens Verfahren mit einem größeren Stabilitätsgebiet) kann man als Verallgemeinerung der expliziten Runge-Kutta Verfahren erhalten. Bei diesen verwenden wir zur Berechnung von k_j nur die Werte von y_n und k_i, $i = 1, \ldots, j-1$.

Definition 10.10 Es seien $\beta_{ij} \in \mathbb{R}, \gamma_i \in \mathbb{R}$, $i, j = 1, \ldots, s$, gegeben und α_i mit

$$\alpha_i = \sum_{j=1}^{s} \beta_{ij}, \ i = 1, \ldots, s. \tag{10.9}$$

Dann heißt das Verfahren

$$\begin{aligned} k_i &= f\Big(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j\Big), \ i = 1, \ldots, s, \\ y_{n+1} &= y_n + h \sum_{i=1}^{s} \gamma_i k_i \end{aligned} \tag{10.10}$$

Runge-Kutta Verfahren mit s Stufen.

Bemerkung 10.11 Gilt $\beta_{ij} = 0$ für $i \leqslant j$, so handelt es sich um ein explizites Runge-Kutta Verfahren. Gilt $\beta_{ij} \neq 0$ für ein $i \leqslant j$, so nennen wir das Runge-Kutta Verfahren **implizit**. □

Beispiel 10.12 Das implizite Euler Verfahren

$$y_{n+1} = y_n + h f(x_{n+1}, y_{n+1})$$

ist offenbar ein einstufiges implizites Runge-Kutta Verfahren.

Ein weiteres naheliegendes Verfahren ist die **implizite Mittelpunktregel**

$$\begin{aligned} k_1 &= f\Big(x_n + \tfrac{h}{2}, y_n + \tfrac{1}{2} h k_1\Big) \\ y_{n+1} &= y_n + h k_1, \end{aligned} \tag{10.11}$$

die offenbar ebenfalls ein einstufiges implizites Runge-Kutta Verfahren ist.

Die Trapezregel

$$y_{n+1} = y_n + \frac{1}{2} h \Big(f(x_n, y_n) + f(x_{n+1}, y_{n+1}) \Big)$$

kann als zweistufiges implizites Runge-Kutta Verfahren aufgefasst werden:

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f\Big(x_n + h, y_n + h(\frac{1}{2} k_1 + \frac{1}{2} k_2)\Big), \\ y_{n+1} &= y_n + h\Big(\frac{1}{2} k_1 + \frac{1}{2} k_2\Big). \end{aligned}$$

□

Wie in Kapitel 8 die expliziten Runge-Kutta Verfahren stellt man auch die impliziten Verfahren übersichtlich in einem Tableau dar:

$$\begin{array}{c|cccc} \alpha_1 & \beta_{11} & \beta_{12} & \ldots & \beta_{1s} \\ \alpha_2 & \beta_{21} & \beta_{22} & \ldots & \beta_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_s & \beta_{s1} & \beta_{s2} & \ldots & \beta_{ss} \\ \hline & \gamma_1 & \gamma_2 & \ldots & \gamma_s. \end{array}$$

Hiermit erhalten die Verfahren aus Beispiel 10.12 die Gestalt

1	1
	1

implizites Euler V.

0.5	0.5
	1

Mittelpunktregel

0	0	0
1	0.5	0.5
	0.5	0.5

Trapezregel.

Nachteil der impliziten Runge-Kutta Verfahren ist, dass die k_j nicht nacheinander berechnet werden können, sondern dass in jedem Schritt ein (i.a. nichtlineares) Gleichungssystem von $s \cdot N$ Gleichungen in den $s \cdot N$ Unbekannten $k_1, \ldots, k_s$ gelöst werden muss, wobei N die Dimension des Differentialgleichungssystems 8.1 von Seite 125 bezeichnet. Eine naheliegende Frage ist, unter welchen Bedingungen das System (eindeutig) lösbar ist.

Satz 10.13 *Es sei $f : [a,b] \times \mathbb{R}^N \to \mathbb{R}^N$ stetig und erfülle eine Lipschitz Bedingung bzgl. des zweiten Arguments mit der Lipschitz Konstante L. Erfüllt die Schrittweite $h > 0$ die Bedingung*

$$h < \frac{1}{L \max\limits_{i=1,\ldots,s} \sum\limits_{j=1}^{s} |\beta_{ij}|}, \tag{10.12}$$

so ist das Gleichungssystem

$$k_i = f\Big(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j\Big), \ i = 1, \ldots, s, \tag{10.13}$$

eindeutig lösbar.

Beweis: Wir zeigen, dass die Abbildung

$$T : (k_i)_{i=1,\ldots,s} \mapsto \Big(f(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j)\Big)_{i=1,\ldots,s}$$

kontrahierend auf $\mathbb{R}^{sN}$ ist und daher einen eindeutigen Fixpunkt besitzt.

Es ist

$$\begin{aligned}
&\|T((k_i)_{i=1,\ldots,s}) - T((\tilde{k}_i)_{i=1,\ldots,s})\|_\infty \\
&= \max_{i=1,\ldots,s} \Big\| f(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j) - f(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} \tilde{k}_j) \Big\|_\infty \\
&\leqslant \max_{i=1,\ldots,s} L \Big\| \Big(y_n + h \sum_{j=1}^{s} \beta_{ij} k_j\Big) - \Big(y_n + h \sum_{j=1}^{s} \beta_{ij} \tilde{k}_j\Big) \Big\|_\infty \\
&= Lh \max_{i=1,\ldots,s} \Big\| \sum_{j=1}^{s} \beta_{ij} (k_j - \tilde{k}_j) \Big\|_\infty \\
&\leqslant Lh \|(k_j)_{j=1,\ldots,s} - (\tilde{k}_j)_{j=1,\ldots,s}\|_\infty \max_{i=1\ldots,s} \sum_{j=1}^{s} |\beta_{ij}|.
\end{aligned}$$

■

Erfüllt die rechte Seite f nur eine Lipschitzbedingung in einer Umgebung von (x_n, y_n), so bleibt die Aussage des Satzes immer noch richtig, wobei die Schrittweite eventuell verkleinert werden muss, um zu sichern, dass das Argument von f in dieser Umgebung bleibt.

Aus dem Fixpunktsatz für kontrahierende Abbildungen folgt zugleich, dass die Fixpunktiteration gegen die Lösung $(k_i)_{i=1,\dots,s}$ von (10.13) konvergiert, wenn die Schrittweite h die Bedingung (10.12) erfüllt. Für steife Probleme ist diese Aussage aber wertlos, da hierfür i.a. die Lipschitzkonstante recht groß sein wird. Bei steifen Systemen muss das Gleichungssystem (10.13) immer mit dem Newton Verfahren oder einer verwandten Methode gelöst werden.

Ein eleganter Weg zur Konstruktion impliziter Runge-Kutta Verfahren hoher Ordnung ist die **Kollokation**. Er besteht darin, ein Polynom $u(x)$ vom Grad s zu bestimmen, so dass die Ableitung (ein Polynom vom Grad $s-1$) an s gegebenen Stellen $x_n + \alpha_i h$, $i = 1, \dots, s$, mit dem Vektorfeld der Differentialgleichung übereinstimmt, d.h.

$$u(x_n) = y_n \tag{10.14}$$
$$u'(x_n + \alpha_i h) = f(x_n + \alpha_i h, u(x_n + \alpha_i h)),\ i = 1, \dots, s. \tag{10.15}$$

Die Näherung für die Lösung an der Stelle $x_n + h$ ist dann

$$y_{n+1} := u(x_n + h). \tag{10.16}$$

Dass dieses Vorgehen als implizites Runge-Kutta Verfahren gedeutet werden kann, sieht man so ein: Es sei

$$k_i := u'(x_n + \alpha_i h).$$

Dann gilt nach der Lagrangeschen Interpolationsformal

$$u'(x_n + th) = \sum_{j=1}^{s} k_j \ell_j(t), \quad \ell_j(t) = \prod_{k \neq j} \frac{t - \alpha_k}{\alpha_j - \alpha_k}. \tag{10.17}$$

Integriert man diese Gleichung, so erhält man mit

$$k_i = u'(x_n + \alpha_i h)$$
$$\beta_{ij} := \int_0^{\alpha_i} \ell_j(t)\, dt$$
$$\gamma_j := \int_0^1 \ell_j(t)\, dt$$

die Formel (10.10) des Runge-Kutta Verfahrens, denn es gilt

$$y_{n+1} = u(x_n + h) = u(x_n) + h \int_0^1 u'(x_n + th)\, dt$$
$$= y_n + h \sum_{j=1}^{s} \int_0^1 k_j \ell_j(t)\, dt = y_n + h \sum_{j=1}^{s} \gamma_j k_j.$$

Zu zeigen bleibt

$$k_i = f(x_n + \alpha_i h, y_n + h \sum_{j=1}^{s} \beta_{ij} k_j).$$

Dies folgt aus

$$k_i = u'(x_n + \alpha_i h) = f(x_n + \alpha_i h, u(x_n + \alpha_i h))$$

und

$$u(x_n+\alpha_i h) = y_n + h\int_0^{\alpha_i} u'(x_n+th)\,dt = y_n + h\int_0^{\alpha_i}\sum_{j=1}^{s} u'(x_n+\alpha_j h)\ell_j(t)\,dt$$
$$= y_n + h\sum_{j=1}^{s}\beta_{ij}u'(x_n+\alpha_j h) = y_n + h\sum_{j=1}^{s}\beta_{ij}k_j.$$

Für die Konsistenzordnung gilt

Satz 10.14 *Dem Kollokationsverfahren zu den Knoten α_j ist in natürlicher Weise eine Quadraturformel mit den Knoten α_j und den Gewichten $\gamma_j := \int_0^1 \ell_j(t)\,dt$ zugeordnet:*

$$\int_{x_n}^{x_n+h} f(x)\,dx = h\sum_{j=1}^{s}\gamma_j f(x_n+\alpha_j h) + O(h^{p+1}). \tag{10.18}$$

Besitzt diese die Ordnung p, so hat das Kollokationsverfahren (10.14), (10.15), (10.16) ebenfalls die Ordnung p.

Beweis: s. *Deuflhard, Bornemann* [19] p. 244 ff. ■

Nach diesem Ergebnis ist klar, welche Ordnung für implizite Runge-Kutta Verfahren man durch Kollokation maximal erreichen kann und wie man diese erreicht.

Sind die α_j die Knoten der Gaußschen Quadraturformel der Ordnung 2s, d.h. die Nullstellen des s-ten (geschifteten) Legendre Polynoms

$$\frac{d^s}{dx^s}\Big(x^s(1-x)^s\Big),$$

und sind γ_j die zugehörigen Gewichte, so heißt das durch (10.14), (10.14), (10.16) definierte implizite Runge-Kutta Verfahren **s-stufiges Gauß Verfahren** . Das s-stufige Gauß Verfahren hat die Ordnung $2s$.

Es kann keine impliziten Runge-Kutta Formeln der Stufe s mit höherer Konsistenzordnung als $2s$ geben, denn sonst hätte man zugleich durch (10.18) eine Quadraturformel mit s Knoten und einer höheren Ordnung als $2s$ gefunden.

Man kann ferner zeigen, dass das Gauß Verfahren der Ordnung $2s$ A-stabil ist (vgl. *Hairer, Wanner* [39] p. 72).

Das einstufige Gauß Verfahren ist die implizite Mittelpunktregel, das zweistufige Gauß Verfahren wird durch das Tableau

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

gegeben und das dreistufige Verfahren durch

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

10.4 Rosenbrock Verfahren

Wir haben bereits bemerkt, dass die nichtlinearen Gleichungssysteme, die in den impliziten Runge-Kutta Verfahren auftreten, nicht durch sukzessive Iteration gelöst werden können, sondern dass eine Linearisierung verwendet werden muss. Um dies zu umgehen, wurde von *Rosenbrock* [61] 1962 vorgeschlagen, nur den linearen Anteil der rechten Seite mit einem impliziten Verfahren zu behandeln und den Rest mit einem expliziten Verfahren. Wir erläutern das Vorgehen für das autonome System

$$\boldsymbol{y}' = f(\boldsymbol{y}). \tag{10.19}$$

Dies bedeutet keine Einschränkung, denn das nichtautonome System

$$\boldsymbol{y}' = f(x, \boldsymbol{y})$$

kann man durch Hinzunahme der Differentialgleichung

$$x' = 1$$

in ein autonomes System

$$\begin{array}{rcl} \boldsymbol{y}' & = & f(x, \boldsymbol{y}) \\ x' & = & 1 \end{array} \tag{10.20}$$

transformieren.

(10.19) schreiben wir mit $\boldsymbol{J} := f'(\boldsymbol{y}_n)$ als

$$\boldsymbol{y}' = (f(\boldsymbol{y}_n) + \boldsymbol{J}(\boldsymbol{y} - \boldsymbol{y}_n)) + (f(\boldsymbol{y}) - f(\boldsymbol{y}_n) - \boldsymbol{J}(\boldsymbol{y} - \boldsymbol{y}_n)).$$

Wir zerlegen die rechte Seite also in die Linearisierung von f bei $\boldsymbol{y}_n$ (von der wir annehmen können, dass sie für das steife Verhalten verantwortlich ist) und (bei kleiner Schrittweite) einen kleinen Rest. Wir integrieren den linearen Anteil implizit und den kleinen Anteil explizit. Da dabei von beiden Formeln der konstante Anteil $f(\boldsymbol{y}_n) - \boldsymbol{J}\boldsymbol{y}_n$ exakt integriert werden wird, wenden wir die Idee auf die Zerlegung

$$\boldsymbol{y}' = \boldsymbol{J}\boldsymbol{y}(x) + (f(\boldsymbol{y}(x)) - \boldsymbol{J}\boldsymbol{y}(x)) \tag{10.21}$$

der rechten Seite von (10.19) an und berechnen hierfür für $i = 1, \ldots, s$ die "Steigungen"

$$\boldsymbol{k}_i = \boldsymbol{J}\Big(\boldsymbol{y} + h\sum_{j=1}^{i} \tilde{\beta}_{ij}\boldsymbol{k}_j\Big) + \Big(f(\boldsymbol{y} + h\sum_{j=1}^{i-1} \beta_{ij}\boldsymbol{k}_j) - \boldsymbol{J}(\boldsymbol{y} + h\sum_{j=1}^{i-1} \beta_{ij}\boldsymbol{k}_j)\Big). \tag{10.22}$$

Beachten Sie, dass das aus (10.22) zu berechnende $\boldsymbol{k}_i$ auf der rechten Seite nur linear auftritt und nur die bereits bekannten $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{i-1}$ auch im Argument der nichtlinearen Funktion f. Man kann also $\boldsymbol{k}_i$ aus einem linearen Gleichungssystem berechnen.

Definition 10.15 Ein Schritt eines **linear impliziten Runge-Kutta Verfahrens** oder **Rosenbrock Verfahrens** mit s Stufen für das autonome System (10.19) besteht aus den folgenden Schritten

$$\begin{array}{lrcl} (i) & \boldsymbol{J} & = & f'(\boldsymbol{y}_n) \\ (ii) & (\boldsymbol{I} - h\tilde{\beta}_{ii}\boldsymbol{J})\boldsymbol{k}_i & = & h\sum_{j=1}^{i-1}(\tilde{\beta}_{ij} - \beta_{ij})\boldsymbol{J}\boldsymbol{k}_j + f(\boldsymbol{y}_n + h\sum_{j=1}^{i-1}\beta_{ij}\boldsymbol{k}_j), \\ & & & i = 1, \ldots, s \\ (iii) & \boldsymbol{y}_{n+1} & = & \boldsymbol{y}_n + h\sum_{j=1}^{s}\gamma_j\boldsymbol{k}_j \end{array} \tag{10.23}$$

In jedem Schritt sind s lineare Gleichungssysteme zu lösen.

Beispiel 10.16 Das einfachste linear implizite Runge-Kutta Verfahren ist das **linear implizite Euler Verfahren**

$$\begin{aligned} (\boldsymbol{I} - hf'(\boldsymbol{y}_n))\boldsymbol{k}_1 &= f(\boldsymbol{y}_n) \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k}_1 \end{aligned} \tag{10.24}$$

Dieses besitzt dieselbe Stabilitätsfunktion wie das implizite Euler Verfahren. Die Konsistenzordnung ist ebenfalls 1. □

Bemerkung 10.17 Wendet man das linear implizite Verfahren (10.23) auf das "autonomisierte" System (10.20) an, so kann man die zu der Variablen x gehörige Gleichung explizit ausrechnen. Man erhält die folgende Form des Verfahrens

$$\begin{aligned} \boldsymbol{J} &= \frac{\partial}{\partial \boldsymbol{y}} f(x_n, \boldsymbol{y}_n), \\ (\boldsymbol{I} - h\tilde{\beta}_{ii}\boldsymbol{J})\boldsymbol{k}_i &= f(x_n + \alpha_i h, \boldsymbol{y}_n + h\sum_{j=1}^{i-1} \beta_{ij}\boldsymbol{k}_j) + \delta_i h \frac{\partial}{\partial x} f(x_n, \boldsymbol{y}_n) \\ &\quad + h\boldsymbol{J}\sum_{j=1}^{i-1}(\tilde{\beta}_{ij} - \beta_{ij})\boldsymbol{k}_j, \ i = 1, \ldots, s, \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\sum_{j=1}^{s} \gamma_j \boldsymbol{k}_j \end{aligned} \tag{10.25}$$

Dabei ist

$$\alpha_i := \sum_{j=1}^{i-1} \beta_{ij}, \ \delta_i := \tilde{\beta}_{ii} + \sum_{j=1}^{i-1}(\tilde{\beta}_{ij} - \beta_{ij}).$$

□

Beispiel 10.18 Für nichtautonome Anfangswertaufgaben erhält das linear implizite Euler Verfahren die Gestalt

$$\begin{aligned} \left(\boldsymbol{I} - h\frac{\partial}{\partial \boldsymbol{y}} f(x_n, \boldsymbol{y}_n)\right)\boldsymbol{k}_1 &= f(x_n, \boldsymbol{y}_n) + h\frac{\partial}{\partial x} f(x_n, \boldsymbol{y}_n) \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k}_1 \end{aligned} \tag{10.26}$$

Rosenbrock Verfahren höherer Ordnung wurden von Kaps und Rentrop (1979), Shampine (1982) und van Veldhuizen (1984) konstruiert. Kaps und Ostermann (1989) konstruierten eingebettete linear implizite Runge-Kutta Verfahren. Sie können ähnlich leicht implementiert werden wie explizite Runge-Kutta Verfahren. Den Code ROS4 findet man in

http://www.unige.ch/math/folks/hairer/

Bemerkung 10.19 In der ODE-Suite von MATLAB ist als ODE23s das folgende eingebettete Paar von linear impliziten Verfahren der Ordnungen 2 und 3 mit der FSAL Eigenschaft implementiert:

$$\begin{aligned} \boldsymbol{f}_0 &= f(x_n, \boldsymbol{y}_n) \\ \boldsymbol{W}\boldsymbol{k}_1 &= \boldsymbol{f}_0 + hd\boldsymbol{t} \\ \boldsymbol{f}_1 &= f(x_n + 0.5h, \boldsymbol{y}_n + 0.5h\boldsymbol{k}_1) \\ \boldsymbol{W}\boldsymbol{k}_2 &= \boldsymbol{f}_1 - \boldsymbol{k}_1 + \boldsymbol{W}\boldsymbol{k}_1 \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k}_2 \\ \boldsymbol{f}_2 &= f(x_{n+1}, \boldsymbol{y}_{n+1}) \\ \boldsymbol{W}\boldsymbol{k}_3 &= \boldsymbol{f}_2 - (6 + \sqrt{2})(\boldsymbol{k}_2 - \boldsymbol{f}_1) - 2(\boldsymbol{k}_1 - \boldsymbol{f}_0) + hd\boldsymbol{t} \\ \text{Fehler} &\approx h(\boldsymbol{k}_1 - 2\boldsymbol{k}_2 + \boldsymbol{k}_3)/6 \end{aligned}$$

mit $d = 1/(2+\sqrt{2})$, $\boldsymbol{W} = \boldsymbol{I} - hd\boldsymbol{J}$ und

$$\boldsymbol{J} \approx \frac{\partial}{\partial \boldsymbol{y}} f(x_n, \boldsymbol{y}_n) \quad \text{und} \quad \boldsymbol{t} \approx \frac{\partial}{\partial \boldsymbol{x}} f(x_n, \boldsymbol{y}_n).$$

War ein Schritt erfolgreich, so kann $\boldsymbol{f}_2$ dieses Schrittes offenbar als $\boldsymbol{f}_0$ des folgenden Schrittes verwendet werden.

In MATLAB 6.1 sind 2 Methoden zur Lösung steifer Systeme implementiert. Das Programm ode15s verwendet BDF-Formeln oder **NDF Verfahren** der Ordnung $k \in \{1,2,3,4,5\}$. NDF sind Modifikationen der BDF Methoden, die ebenfalls A(α)-stabil sind. Sie besitzen eine etwas größere Genauigkeit als die BDF-Methoden, wobei der Öffnungswinkel α des maximalen im Stabilitätsgebiet enthaltenen Sektors aber nur wenig verkleinert wird. Ihre Konstruktion ist in *Shampine, Reichel* [65] beschrieben.

Das Programm ode23s verwendet ein Rosenbrock Verfahren der Ordnung 2, wobei der Fehler mit einer Modifikation der Ordnung 3 geschätzt wird. Es ist geeignet, wenn die Genauigkeitsansprüche nicht zu hoch sind.

Beispiel 10.20 Das folgende Beispiel, das die Entwicklung der Konzentrationen dreier Komponenten in einer chemischen Reaktion beschreibt, wurde von *Robertson* [59] 1967 angegeben und wird häufig verwendet, um die Eigenschaften von steifen Lösern zu demonstrieren:

$$\begin{array}{lll} y_1' &= -0.04y_1 + 10^4 y_2 y_3, & y_1(0) = 1 \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2, & y_2(0) = 0 \\ y_3' &= 3 \cdot 10^7 y_2^2, & y_3(0) = 0. \end{array}$$

Abbildung 10.8 zeigt die Lösungen dieses Systems in halblogarithmischer Darstellung.

Zur Lösung dieses Systems im Intervall $[0, 10^6]$ (mit den voreingestellten Genauigkeitsschranken) benötigt das NDF Verfahren ode15s 146 Schritte. Mit dem impliziten Euler Verfahren benötigt man 310 Schritte. Qualitativ dasselbe Bild erhält man mit dem Rosenbrock Verfahren ode23s mit 61 Schritten.

Der Versuch, die Aufgabe mit dem nicht für steife Probleme geeigneten Verfahren ode23 zu lösen, benötigt in dem Intervall $[0,1]$ schon 860 Schritte. Einen Ausschnitt des Graphen der zweiten Komponente der erzeugten Näherungslösung findet man in Abbildung 10.9. □

Es gibt viele (auch einander widersprechende) Definitionen der Steifheit. Häufig wird gesagt, dass ein Problem steif ist, wenn es verschieden schnell abklingende Lösungen besitzt, z.B. weil die Jacobi Matrix der rechten Seite Eigenwerte mit sehr unterschiedlichen (negativen) Realteilen besitzt.

Dies trifft jedoch nicht den Kern. Will man das schnelle Abklingen von Lösungskomponenten darstellen, so ist man gezwungen, die Lösungen mit sehr kleinen Schrittweiten zu approximieren, und hierzu kann man dann ein explizites Verfahren verwenden, denn diese sind in der Regel billiger als implizite Verfahren. Will man dagegen (wie in Beispiel 10.1) eine sich langsam ändernde Lösung verfolgen, was eigentlich mit großen Schrittweiten möglich sein sollte, und wird ein explizites Verfahren durch sehr schnell abklingende Lösungsanteile zu sehr kleinen Schrittweiten gezwungen, so nennen wir ein Problem steif.

10.5 Bemerkungen zur Wahl der Verfahren

Einer Anfangswertaufgabe sieht man nicht unmittelbar an, ob ihre Lösung steif ist. Es gibt einige Aufgabenklassen, bei denen man weiß, dass steife Lösungen zu erwarten sind wie z.B. Gleichungen, die chemische Reaktionen mit sehr unterschiedlichen

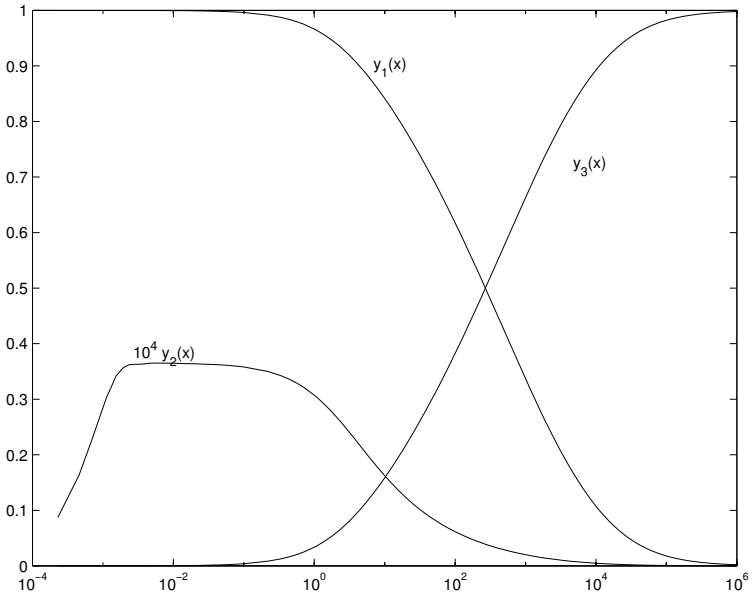

Abbildung 10.8: Lösungen von Beispiel 10.20 mit ode15s

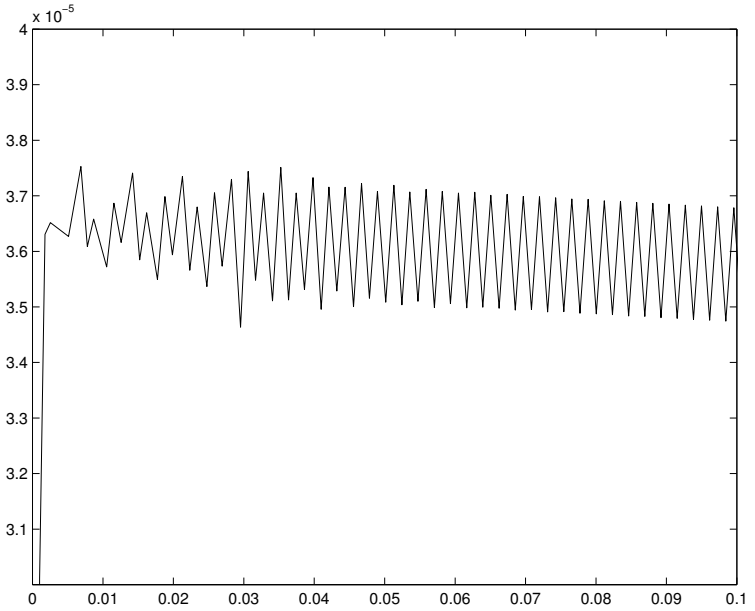

Abbildung 10.9: Lösungen von Beispiel 10.20 mit ode23

Reaktionsgeschwindigkeiten beschreiben, Systeme, die sich mit der Linienmethode aus parabolischen oder hyperbolischen Anfangsrandwertaufgaben ergeben, oder singulär gestörte Probleme wie die van der Pol Gleichung mit sehr großem Parameter. In diesem Fall wird man sofort steife Löser verwenden.

Liegen keine guten Gründe dafür vor, dass eine steife Lösung zu erwarten ist, wird man zuerst versuchen, das gegebene Problem mit einem nicht-steifen Löser zu behandeln, denn explizite (eigebettete) Runge-Kutta Verfahren oder Mehrschrittverfahren vom Adams Typ sind wesentlich billiger als steife Löser. Bei steifen Lösern hat man ja in jedem Schritt ein nichtlineares Gleichungssystem zu lösen und hierzu die Jacobimatrix der rechten Seite oder eine Näherung davon in jedem Schritt zu bestimmen. Beobachtet man, dass der Lösungsprozess nur sehr langsam voranschreitet, wird man zu einem steifen Löser wechseln.

Runge-Kutta Verfahren ermöglichen eine einfache Schrittweitensteuerung, haben aber den Nachteil gegenüber den Adams Verfahren, dass in jedem Schritt die rechte Seite an mehreren Stellen ausgewertet werden muss (für das Verfahren von Dormand und Prince der Ordnung 5 an 6 Stellen). Beim Prädiktor-Korrektor Verfahren kann man hohe Ordnungen mit 2 (im Fall PECE) oder 3 (im Fall PECECE) Auswertungen erreichen. Man wird daher ein Mehrschrittverfahren verwenden, wenn die Auswertung der rechten Seite der Differentialgleichung sehr teuer ist. In beiden Fällen wird man Verfahren hoher Ordnung nur dann verwenden, wenn die rechte Seite der Differentialgleichung sehr glatt ist. Man verwendet ja den Taylorschen Satz, um Methoden hoher Konsistenzordnung zu entwickeln.

Eine Regel für die Auswahl steifer Löser ist nicht so einfach zu formulieren. Einen Anhaltspunkt geben die Stabilitätsgebiete der Verfahren. BDF Formeln sind A(α)-stabil, wobei für große Ordnungen die Winkel α klein sind. Wenn man weiß, dass die Eigenwerte der Linearisierung der rechten Seite in der Nähe der negativen reellen Achse liegen (bei Linienmethoden für parabolische Aufgaben liegen sie sogar auf der negativen reellen Achse), so wird man BDF Formeln wählen. Weiß man, dass Eigenwerte der Jacobimatrix näher an der imaginären Achse als an der negativen reellen Achse liegen (bei Linienmethoden für hyperbolische Probleme liegen sie auf der imaginären Achse), so wird man Rosenbrockmethoden oder Extrapolationsverfahren verwenden.

Literaturverzeichnis

[1] *M. Abramowitz, I.A. Stegun (eds.):* Handbook of Mathematical Functions. Dover Publications, New York, 1971

[2] *E.L. Allgower, K. Georg:* Numerical Continuation Methods. Springer, Berlin, 1990

[3] *E. Anderson, Z. Bai, C, Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen:* LAPACK Users' Guide. Third Edition, SIAM, Philadelphia, 2000

[4] *N. Anderson, A. Björck:* A new high order method of regula falsi type for computing a root of a equation. BIT 13, 253 – 264 (1973)

[5] *U.M. Ascher, L. Petzold:* Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations. SIAM, Philadelphia 1998

[6] *Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst:* (eds.), Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia, 2000

[7] *A. Björck:* Least Squares Methods. In P.G. Ciarlet, J.L. Lions (eds.), Handbook of Numerical Analysis. Vol. I, pp. 465 – 652, North Holland, Amsterdam, 1990

[8] *A. Björk:* Numerical Methods for Least Squares Problems. SIAM, Philadelphia, 1996

[9] *P. Bogacki, L.F. Shampine:* A 3(2) pair of Runge–Kutta formulas. Appl. Math. Lett. 2, 1 – 9 (1989)

[10] *D. Braess:* Nonlinear Approximation Theory. Springer, Berlin, 1986

[11] *R. Bulirsch:* Bemerkungen zur Romberg Integration. Numer. Math. 6, 6 – 16 (1964)

[12] *F. Chatelin:* Eigenvalues of Matrices. Wiley, Chichester, 1993

[13] *L. Collatz:* Eigenwertaufgaben mit technischen Anwendungen. Akademische Verlagsgesellschaft, Leipzig 1963

[14] *C.F. Curtiss, J.O. Hirschfelder:* Integration of stiff equations. Proc. Nat. Acad. Sci. 38, 235 – 243 (1952)

[15] *G. Dahlquist:* A special stability problem for linear multistep methods. BIT 3, 27 – 43 (1963)

[16] *C. de Boor:* A Practical Guide to Splines. Springer, New York, 1978

[17] *J.W. Demmel:* Applied Numerical Linear Algebra. SIAM, Philadelphia, 1997

[18] *J.E. Dennis, Jr., R.B. Schnabel:* Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM, Philadelphia, 1996

[19] *P. Deuflhard, F. Bornemann:* Numerische Mathematik II. Integration gewöhnlicher Differentialgleichungen. De Gruyter, Berlin 1994

[20] *P. Deuflhard, A. Hohmann:* Numerische Mathematik I. Eine algorithmisch orientierte Einführung. 2. Aufl., De Gruyter, Berlin 1993

[21] *J.R. Dormand, P.J. Prince:* A family of embedded Runge–Kutta formulae. J. Comput. Appl. Math. 6, 19 – 26 (1980)

[22] *M. Dowell, P. Jarrat:* A modified regula falsi method for computing the root of an equation. BIT 11, 168 – 174 (1971)

[23] *M. Dowell, P. Jarrat:* The "Pegasus" method for computing the root of an equation. BIT 12, 503 – 508 (1972)

[24] *H.W. Engl:* Integralgleichungen. Springer, Wien 1997

[25] *J.G.F. Francis:* The QR transformation. A unitary analogue to the LR transformation. Part 1. Computer J. 4, 256 – 272 (1961)

[26] *J.G.F. Francis:* The QR transformation. A unitary analogue to the LR transformation. Part 2 Computer J. 4, 332 – 345 (1961)

[27] *C.W. Gear:* Numerical Initial Value Problems in Ordinary Differential Equations. Prentice Hall, Englewood Cliffs 1971

[28] *W.M. Gentleman:* Least squares computations by Givens transformations without square roots. J. Inst. Maths. Applic. 12, 329 – 336 (1973)

[29] *P.E. Gill, W. Murray, M.H. Wright:* Practical Optimization. Academic Press, London, 1981

[30] *W. Givens:* Computation of plane unitary rotations transforming a general matrix to triangular form. SIAM J. Appl. Math. 6, 26 – 50 (1958)

[31] *G.H. Golub:* Numerical methods for solving linear least squares problems. Numer. Math. 7, 206 – 216 (1965)

[32] *G.H. Golub, C.F. Van Loan:* Matrix Computations. 3rd ed., The John Hopkins University Press, Baltimore, 1996

[33] *R.D. Grigorieff:* Numerik gewöhnlicher Differentialgleichungen I. Teubner Verlag, Stuttgart 1972

[34] *R.D. Grigorieff:* Numerik gewöhnlicher Differentialgleichungen II. Teubner Verlag, Stuttgart 1977

[35] *S. Hammarling:* A note on modifications of the Givens plane rotation. J. Inst. Maths. Applic. 13, 215 – 218 (1974)

[36] *P.C. Hansen:* Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. SIAM, Philadelphia 1998

[37] *P.C. Hansen:* The L-curve and its use in the numerical treatment of inverse problems. In P. Johnston (ed): Computational Inverse Problems in Electrocardiography. WIT Press, Southampton 2000

[38] *E. Hairer, S.P. Norsett, G. Wanner:* Solving Ordinary Differential Equations I. Nonstiff Problems. 2. Auflage. Springer Verlag, Berlin 1993

[39] *E. Hairer, G. Wanner:* Solving Ordinary Differential Equations II. Stiff and Differential–Algebraic Problems. 2. Auflage. Springer Verlag, Berlin 1996

[40] *C.G.J. Jacobi:* Über ein leichtes Verfahren, die in der Theorie der Sekularstörungen vorkommenden Gleichungen numerisch aufzulösen. Crelle Journal für die reine und angewandte Mathematik 30, 51 – 94 (1846)

[41] *A. Kielbasinski, H. Schwetlick:* Numerische Lineare Algebra. VEB Deutscher Verlag der Wissenschaften, Berlin 1988

[42] *R.F. King:* An improved Pegasus-method for root finding. BIT 13, 423 – 427 (1973)

[43] *A.R. Krommer, C.W. Ueberhuber:* Computational Integration. SIAM, Philadelphia, 1998

[44] *A.S. Kronrod:* Integration with control of accuracy. Soviet Physics Dokl. 9, 17 – 19 (1964)

[45] *A.S. Kronrod:* Nodes and weights of quadrature formulas. Consultants Bureau, New York, 1965

[46] *V.N. Kublanowskaya:* On some algorithms for the solution of the complete eigenvalue problem. USSR Comp. Math. Phys. 3, 637 – 657 (1961)

[47] *K. Levenberg:* A Method for the solution of certain non-linear problems in least squares. Quart.Appl.Math. 2, 164 – 168 (1944)

[48] *A.K. Louis:* Inverse und schlecht gestellte Probleme Teubner, Stuttgart 1989

[49] *W. Mackens, H. Voß:* Mathematik I für Studierende der Ingenieurwissenschaften. HECO-Verlag, Alsdorf 1993

[50] *W. Mackens, H. Voß:* Aufgaben und Lösungen zur Mathematik I für Studierende der Ingenieurwissenschaften. HECO-Verlag, Alsdorf 1994

[51] *D.W. Marquardt:* An algorithm for least squares estmation of non-linear parameters. SIAM J. 11, 431 – 441 (1963)

[52] *G. Meinardus, G. März:* Praktische Mathematik I Bibliographisches Institut, Mannheim 1979

[53] *J. Nocedal, S.J. Wright:* Numerical Optimization Springer, New York, 1999

[54] *J.M. Ortega, W. Rheinboldt:* Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York – London, 1970

[55] *B.N. Parlett:* The Symmetric Eigenvalue Problem. SIAM, Classics in Applied Mathematics 20, Philadelphia, 1998

[56] *R. Piessens, E. de Doncker-Kapenga, C.W. Überhuber, D.K. Kahaner:* QUADPACK. A soubroutine package for automatic integration. Springer Verlag, Berlin, 1983

[57] *D.L. Philips:* A technique for the numerical solution of certain integral equations of the first kind. J. Assoc. Comput. Mach. 9, 84 – 97 (1962)

[58] *W.H. Press, B.P. Lannery, S.A. Teukolsky, W.T. Vetterling:* Numerical Recipes in FORTRAN – The Art of Scientific Computing. 2nd edition. Cambridge University Press, Cambridge, 1992

[59] *H.H. Robertson:* The solution of a set of reaction rate equations. In J. Walsh (ed.), Numerical Analysis: An Introduction. Academic Press, London 1967, pp. 178 – 182

[60] *W. Romberg:* Vereinfachte numerische Integration. Det Kongelige Norske Videnskabers Forhandlinger, Bind 28 (7), 1955

[61] *H.H. Rosenbrock:* Some general implicit processes for the numerical solution of differential equations. Computer J. 5, 329 – 330 (1962/63)

[62] *H.R. Schwarz:* Numerische Mathematik. Teubner, Stuttgart 1988

[63] *H. Schwetlick:* Numerische Lösung nichtlinearer Gleichungen. Oldenbourg, München – Wien 1979

[64] *H. Schwetlick, H. Kretzschmar:* Numerische Verfahren für Naturwissenschaftler und Ingenieure Fachbuchverlag Leipzig, Leipzig 1991

[65] *L.F. Shampine, M.W. Reichel:* The MATLAB ODE suite. SIAM J. Sci. Comput. 18, 1 – 22 (1997)

[66] *H.J. Stetter:* Analysis of Discretization Methods for Ordinary Differential Equations. Springer Verlag, Berlin 1973

[67] *G.W. Stewart:* Introduction to Matrix Computations. Academic Press, New York, 1973.

[68] *G.W. Stewart:* Matrix Algorithms. Vol. 1: Basic Decompositions. SIAM, Philadelphia, 1998

[69] *G.W. Stewart:* Matrix Algorithms. Vol. 2: Eigensystems. SIAM, Philadelphia, 2001

[70] *J. Stoer:* Einführung in die Numerische Mathematik I. 6. Auflage. Springer Verlag, Berlin 1993 Elsevier, 1987

[71] *J. Stoer, R. Bulirsch:* Einführung in die Numerische Mathematik II. 3. Auflage, Springer, Berlin 1990

[72] *J. Stoer, R. Bulirsch:* Introduction to Numerical Analysis. Springer, New York, 1980

[73] *K. Strehmel, R. Weiner:* Numerik gewöhnlicher Differentialgleichungen Teubner, Stuttgart 1995

[74] *A.N. Tichonov:* Solution of incorrectly formulated problems and the regularization method. Soviet. Math. Dokl. 4, 1035 – 1038 (1963). Englische Übersetzung von Dokl. Akad. Nauk. SSSR 151, 501 – 504 (1963)

[75] *C. Überhuber:* Computer Numerik 1. Springer, Berlin 1995

[76] *C. Überhuber:* Computer Numerik 2. Springer, Berlin 1995

[77] *H. Voss:* Grundlagen der Numerischen Mathematik. Skript, TU Hamburg-Harburg (2002).

[78] *H. Voss:* Numerische Behandlung von Differentialgleichungen. Skript, TU Hamburg-Harburg (2002).

[79] *D.S. Watkins:* Fundamentals of Matrix Computations. 2nd Edition. Wiley, New York, 2002

[80] *H. Werner, R. Schaback:* Praktische Mathematik II Springer Verlag, Berlin 1972

[81] *H. Wielandt:* Beiträge zur mathematischen Behandlung komplexer Eigenwertprobleme, Teil V: Bestimmung höherer Eigenwerte durch gebrochene Iteration.

[82] *J.H. Wilkinson:* The Algebraic Eigenvalue Problem. Clarendon Press, Oxford, 1965.

[83] *R. Zurmühl:* Matrizen und ihre technischen Anwendungen. Springer Verlag, Berlin 1964

Index