Numerik I

K. Böhmer, B. Schmitt

Sommer-Semester 1998

Inhaltsverzeichnis

1 Einleitung **3**

2 Lineare Gleichungssysteme **7**

2.1 Beispiele 7

2.2 Hilfsmittel 9

2.3 Der Gauß-Algorithmus 12

LR-Zerlegung 14

Pivotisierung 15

2.4 Fehlerausbreitung, Kondition 19

Schranken für $\kappa(A)$ 21

2.5 Iterationsverfahren für lineare Gleichungssysteme 23

Gesamt- und Einzelschrittverfahren 25

Relaxationsverfahren 28

3 Polynom-Interpolation **31**

3.1 Problemstellung 31

3.2 Lagrange-Interpolation für Polynome 31

3.3 Newton-Interpolation 34

3.4 Entwicklung nach Orthogonalpolynomen, Drei-Term-Rekursionen 39

3.5 Vergleich der Verfahren 42

3.6 Interpolationsfehler, optimale Knoten 43

4 Spline-Funktionen **48**

4.1 Definition 48

4.2 Existenz des Interpolations-Splines 50

4.3 Konvergenz des Interpolations-Splines 52

4.4 Spline-Darstellungen, Beziér-, B-Splines 55

4.5 Lokale Spline-Approximationen 59

5 Numerische Integration und Differentiation **62**

5.1 Quadratur . . . 62

5.2 Newton-Cotes-Formeln . . . 63

5.3 Gauß-Quadratur . . . 66

5.4 Adaptive Integration . . . 69

5.5 Richardson-Extrapolation, Romberg-Verfahren . . . 71

5.6 Numerische Differentiation . . . 76

6 Nichtlineare Gleichungssysteme **78**

6.1 Nullstellen einer skalaren Funktion . . . 78

Newton-Verfahren . . . 80

Bisektionsverfahren . . . 82

Sekantenverfahren, Regula falsi . . . 82

Interpolationsverfahren höherer Ordnung . . . 85

6.2 Newtonverfahren im $\mathbf{R}^n$ . . . 85

7 Rundungsfehler - Analyse **90**

7.1 Zahldarstellung und Rundungsfehler . . . 90

7.2 Rundungsfehleranalyse einiger Algorithmen . . . 94

Approximationsverfahren . . . 94

Gauß-Algorithmus . . . 95

Fixpunkt-Iteration . . . 97

Index **98**

1 Einleitung

Die Numerik befaßt sich mit der "Lösung" analytisch-mathematischer Modelle auf Computern. Da die dabei verwendeten Computer-Modelle zwar meist groß, aber endlich sind, kann man nur Näherungen solcher Lösungen berechnen. Eine wesentliche Aufgabe der Numerik ist daher die Betrachtung von Fehlern. Allerdings stellt der Übergang zum numerischen Modell nur der allerletzte Schritt bei der Modellierung realer Situationen dar, wobei in der Regel in jeder Stufe schon Vereinfachungen stattfinden.

Modellierungs-Schema:

Reale Situation, Beschreibung durch physikalische Gesetze, Experiment

↓

Entwicklung mathematischer Modelle, meist unter Vereinfachungen

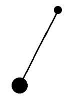
↓

Mathematische Analyse: Existenz? Eindeutigkeit? Qualitative Aussagen

↙ Theorie ↘ Numerik

Theorie	Numerik
Explizite Lösung (Spezialfälle), Reihenentwicklungen; Eigenschaften (∀ Lösungen,..)	Entwicklung numerischer Verfahren mit Fehleranalyse
↓	↓
Lösungsdiskussion, asymptotische Aussagen, Parameterabhängigkeit (Optimierung?)	Implementierung, Lösung vieler Einzelfälle, Simulation, Optimierung

Beispiel Pendel



$\varphi'' + \omega^2 \sin\varphi = 0$, ohne Reibung etc.

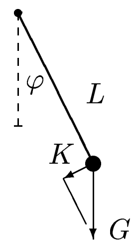
AWP eindeutig lösbar, RWP?

Approx. φ'', Fehler $\sim h^2$

$\varphi'' + \omega^2\varphi = 0$, 'Frequenz'

Subprobleme: Gleichungssysteme, ..

Beispiel: Physikalisches Pendel



Kräftegleichgewicht: Kraft $K = -G\sin\varphi$, Gewicht $G = mg$.

Newtonsches Gesetz: $mL\varphi''(t) = -G\sin\varphi$. Mit $\omega^2 := g/L$ folgt für den Winkel $\varphi(t)$ die Gleichung:

$$\varphi''(t) + \omega^2 \sin\varphi(t) = 0. \qquad (1.0.1)$$

Bei dieser Modellierung wurden Luftwiderstand, Reibung, etc. vernachlässigt. Die Gleichung (1.0.1) ist eine *Differentialgleichung* und beschreibt die Menge *aller* möglichen Bewegungen des Pendels. Einzelne Lösungen werden daraus ausgewählt durch zusätzliche Bedingungen.

Beim *Anfangswertproblem (AWP)* sind Start-Winkel und -Geschwindigkeit bekannt, $\varphi(0) = \alpha$, $\varphi'(0) = \beta$. Beim *Randwertproblem (RWP)* werden Start- und End-Winkel zu bestimmten Zeiten vorgegeben, $\varphi(0) = \alpha$, $\varphi(T) = \gamma$ ($\alpha = \gamma = 0$: Schwingung mit Halb-Periode T). Die Lösbarkeit (Existenz) des Anfangswertproblems folgt aus dem Satz von Picard-Lindelöf, allerdings gilt praktisch:

$\Longrightarrow$ $\boxed{\text{Weder AWP noch RWP zur Gleichung (1.0.1) sind } \textit{explizit} \text{ lösbar!}}$ $\Longleftarrow$

Konsequenz: Approximationsverfahren für die Lösung sind erforderlich. Mögliche Ansätze:

- **mathematisch**: Für 'kleine' Winkel φ gilt $\sin\varphi \cong \varphi$. Statt (1.0.1) wird die Differentialgleichung

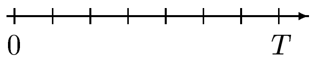
$$\varphi''(t) + \omega^2\varphi(t) = 0 \quad \Rightarrow \quad \varphi(t) = a\cos\omega t + b\sin\omega t, \tag{1.0.2}$$

  mit bekannter allgemeiner Lösung verwendet.

- **numerisch**: Die 'Differentiale' werden ersetzt durch endliche Differenzen. Analysis:

$$\frac{d\varphi}{dt}(t) = \varphi'(t) \cong \frac{\varphi(t+h) - \varphi(t)}{h}, \quad \varphi''(t) \cong \frac{\varphi(t+h) - 2\varphi(t) + \varphi(t-h)}{h^2}. \tag{1.0.3}$$

  Zur numerischen Approximation wird h 'klein', aber fest, gewählt, und Werte nur in endlich vielen Punkten $t_j = jh,\ j = 0, \ldots, m$ mit $h = T/m$ verwendet:



  Dies führt für Näherungen $y_j \cong \varphi(t_j)$ auf Gleichungen

$$0 = \varphi''(t_j) + \omega^2 \sin\varphi(t_j) \cong \frac{y_{j-1} - 2y_j + y_{j+1}}{h^2} + \omega^2 \sin y_j \stackrel{!}{=} 0. \tag{1.0.4}$$

  Zusammen mit den obigen Randbedingungen $y_0 = \alpha$, $y_m = \gamma$ ist dies ist ein nichtlineares Gleichungssystem im $\mathbf{R}^{m-1}$. Der lineare Fall wird noch einmal ausführlicher diskutiert.

Beispiel 1.0.1 *Differenzenverfahren für lineares Randwertproblem:*

$$\begin{gathered} \varphi''(t) + f(t)\varphi(t) = g(t) \\ \varphi(0) = \alpha,\ \varphi(1) = \gamma \end{gathered} \tag{1.0.5}$$

Die Ersetzung der zweiten Ableitung nach (1.0.3) führt hier mit den Bezeichnungen

$$y_j \approx \varphi(t_j), \quad t_j = j \cdot h, \quad h = \frac{1}{m}, \quad f_j := f(t_j), \quad g_j := g(t_j),$$

auf das lineare Gleichungssystem

$$\begin{gathered} \frac{y_{j+1} - 2y_j + y_{j-1}}{h^2} + f_j y_j = g_j \qquad j = 1, \cdots, m-1 \\ y_0 = \alpha, \quad y_m = \beta. \end{gathered} \tag{1.0.6}$$

Ausführlich ausgeschrieben sieht man die Struktur

$$\begin{pmatrix} -2+h^2 f_1 & 1 & 0 & \cdots & 0 \\ 1 & -2+h^2 f_2 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ & & 1 & -2+h^2 f_{m-2} & 1 \\ 0 & \cdots & 0 & 1 & -2+h^2 f_{m-1} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{m-2} \\ y_{m-1} \end{pmatrix} = \begin{pmatrix} h^2 g_1 - \alpha \\ h^2 g_2 \\ \vdots \\ h^2 g_{m-2} \\ h^2 g_{m-1} - \beta \end{pmatrix}.$$

Die Matrix hat eine sehr spezielle Gestalt, die einer *Tridiagonalmatrix*. Matrizen ähnlicher Form treten auch in anderen Situationen auf und spielen daher im weiteren eine wesentliche Rolle.

Die numerische Lösung der Differenzengleichung (1.0.4) führt auch im Bezug zum Ausgangsproblem auf folgende Fragen:

- Auflösung nichtlinearer Gleichungssysteme ($\rightarrow$ §6), dabei: Auflösung linearer Gleichungssysteme ($\rightarrow$ §2),
- Wie genau sind die Näherungen $y_k \cong \varphi(t_k)$ theoretisch? ($\rightarrow$ Numerik IIB)
- Kann $\varphi(t)$ überall in $[0, T]$ approximiert werden? ($\rightarrow$ §3,4)
- Berechnung von $\sin y$? ($\rightarrow$ §3)
- Reelle Zahlen werden im Computer approximiert, wie wirken sich Rundungsfehler und ihre Fortpflanzung auf die Ergebnisse aus? ($\rightarrow$ §7).

Fehlerbehandlung

Da überall in der Numerik mit Approximationen gearbeitet wird und insbesondere Computer reelle Zahlen nicht exakt darstellen können, hat die Analyse von Fehlern und der Versuch, sie zu beherrschen, eine zentrale Bedeutung. Zugänge sind:

- theoretische Analyse der Fehlerquellen und ihrer Auswirkungen, damit Konstruktion und Verwendung von Fehler*schätzungen* in den Verfahren. Ein Versagen der Verfahren ist (hoffentlich nur in Ausnahmefällen) möglich!
- in Teilbereichen (nicht-/ lineare Algebra) sind exakte Fehler*einschließungen* auf Computern durchführbar durch sichere Arithmetiken (Programmbibliothek ACRITH, Sprache PASCAL-SC): Exakte Aussagen, aber teuer mit begrenztem Einsatzbereich.

Informationsquellen

Für die Lösung praktischer Probleme durch numerische Verfahren sollte man außer den üblichen Literatur-Quellen auch vorhandene Software heranziehen. Ein Überblick:

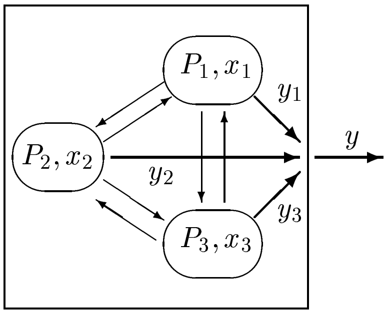
Form	Art	Namen/Quellen
Literatur klassisch:	Lehrbücher und Zeitschriften	Bibliotheken der Universität
Literatur elektronisch:	Kurzreferate zu mathematischen Veröffentlichungen erscheinen schon lange gedruckt im 'Zentralblatt für Mathematik' und den 'Mathematical Reviews'. Mittlerweile kann diese Information in elektronischen Literatur-Datenbanken (z.B. auf CD-ROM) mit Computerunterstützung ausgewertet werden	CompactMATH, MathSci auf UB-Server
Software kommerziell:	Kommerzielle Software-Bibliotheken (Unterprogramme) mit fertigen Lösungsalgorithmen für numerische Standardprobleme	NAG, IMSL auf Großrechner, MATLAB
Software im Internet:	Bibliotheken/Spezial-Software für viele Probleme (BLAS, LINPACK, LAPACK)	elib (elib.zib-berlin.de), Netlib (www.netlib.org)

Da nur in einfachen Ausnahmefällen allgemein einsetzbare Standard-Programme existieren, erfordert die Verwendung solcher Bibliotheken Numerik-Kenntnisse, um Auswahl und Einsatz sinnvoll durchführen zu können.

2 Lineare Gleichungssysteme

Viele Näherungsverfahren der Numerik bei Differentialgleichungen (vgl. Beispiel 1.0.1), Integralgleichungen, etc., führen auf *Lineare Gleichungssysteme* (LGSe). In einigen Anwendungen können Modelle direkt als LGSe formuliert werden. Realistische Aussagen erhält man dabei meist nur für große Modelldimensionen.

2.1 Beispiele



Beispiel 2.1.1 *Input-Output-Analyse*: Zur Analyse der Wechselwirkungen in einer Volkswirtschaft und der Auswirkungen geänderter Nachfrage nimmt man eine Unterteilung in Produktionsbereiche $P_j,\ j = 1,\dots,n$, vor. Mit y_j wird die Endnachfrage der Verbraucher nach Produkten aus P_j bezeichnet, zusammengefaßt zu $y \in \mathbf{R}^n$. Bei der Bestimmung der dazu erforderlichen Produktionsmengen x_i (in P_i) muß berücksichtigt werden, daß zur Herstellung eines Produktes in P_i Teile aus anderen Bereichen benötigt werden (Auto: Maschinen, Stahl, Kunststoff, Elektroteile). Diese Beziehungen faßt man in einer Bedarfsmatrix $B = (b_{ij})$ zusammen:

$$b_{ij} : \begin{cases} \text{Bedarf an Produkten aus } P_i \text{ zur Produktion einer} \\ \text{Einheit (Geldwert) in } P_j,\ b_{ij} \geq 0. \end{cases} \tag{2.1.1}$$

Die benötigte Gesamtproduktion $x = (x_i) \in \mathbf{R}^n$ ergibt sich also aus der Bilanzgleichung

$$x = \underbrace{y}_{\text{End-Nachfrage}} + \underbrace{Bx}_{\text{interne Nachfrage}}$$

$$\Leftrightarrow \qquad x - Bx = y \ \Leftrightarrow\ (I - B)x = y. \tag{2.1.2}$$

Somit beschreibt die Matrix $(I - B)^{-1}$ die Auswirkungen einer sich ändernden Endnachfrage y.

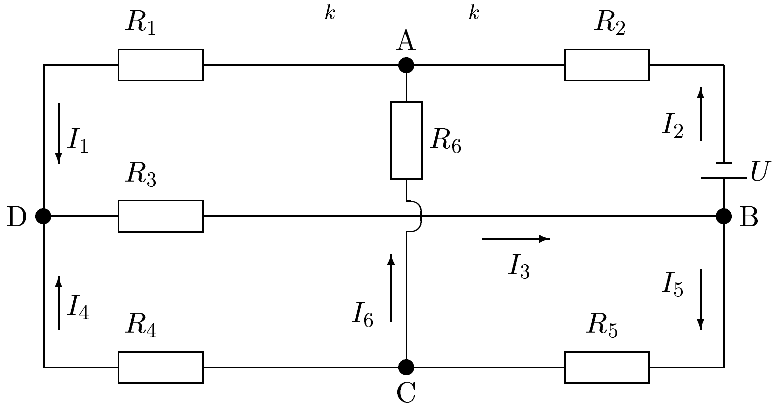
Bemerkung: Auch andere Probleme führen auf Bilanzgleichungen wie in Beispiel 1. In der Computergrafik, z.B., ergibt sich die absolute Helligkeit und Farbe eines Flächenelements aus der direkt und indirekt empfangenen Lichtmenge (Radiosity-Verfahren).

Beispiel 2.1.2 *Elektrisches Netzwerk*: Netzwerke ohne kapazitive, induktive oder aktive Bauteile führen auf LGSe. Zur Berechnung werden die angelegten Einzelspannungen U_k mit den Teilwiderständen R_k und den Teilströmen I_k durch die Kirchhoff'schen Regeln verknüpft:
a) *Knotenregel:* Unter Beachtung der Richtung aller auf einen Knoten zu- bzw. abfließenden Ströme I_k gilt für jeden Knoten

$$\sum_k I_k = 0$$

b) *Maschenregel:* Für jeden geschlossenen Teilstromkreis gilt (unter Berücksichtigung der Richtungen)

$$\sum_k I_k R_k = \sum_k U_k$$



Für das abgebildete Netzwerk ergibt sich so das folgende LGS:

$$\text{Knoten}\quad \begin{array}{rrrrrrrcl} A: & -I_1 & +I_2 & & & & +I_6 & = & 0 \\ B: & & -I_2 & +I_3 & & -I_5 & & = & 0 \\ C: & & & & -I_4 & +I_5 & -I_6 & = & 0 \\ D: & I_1 & & -I_3 & +I_4 & & & = & 0 \end{array} \tag{2.1.3}$$

$$\text{Maschen}\quad \begin{array}{rrrrrrrcl} ABC: & & I_2R_2 & & & -I_5R_5 & -I_6R_6 & = & U \\ ADB: & I_1R_1 & +I_2R_2 & +I_3R_3 & & & & = & U \\ ADC: & I_1R_1 & & & -I_4R_4 & & +I_6R_6 & = & 0 \\ BDC: & & & -I_3R_3 & -I_4R_4 & -I_5R_5 & & = & 0 \end{array} \tag{2.1.4}$$

Die Gleichungen sind sicher *nicht linear unabhängig*, die linear abhängigen Gleichungen sind zu eliminieren. Eine Addition der Gleichungen eins, zwei, drei, vier bzw. fünf, sieben und Subtraktion von sechs und acht ergibt jeweils null. Damit bleiben sechs Gleichungen mit sechs Unbekannten. In Matrixform geht (2.1.3,2.1.4) so über in

$$\begin{array}{c} A \\ B \\ C \\ ACB \\ ADB \\ ADC \end{array} \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & R_2 & 0 & 0 & -R_5 & -R_6 \\ R_1 & R_2 & R_3 & 0 & 0 & 0 \\ R_1 & 0 & 0 & -R_4 & 0 & R_6 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ U \\ U \\ 0 \end{pmatrix} \tag{2.1.5}$$

Bei allen Beispielen stellen sich sofort folgende Fragen:
— Wann besitzen die LGSe eine Lösung (hinreichende Kriterien)?
— Wie löst man solche LGSe numerisch günstig auf (für große Dimensionen, bei spezieller Struktur)?
— Spiegeln sich spezielle Eigenschaften der rechten Seite, z.B. positive Nachfrage y bzw. Spannung U in Beispiel 1 bzw. 2, auch in der Lösung x bzw. $(I_1, \ldots, I_6)$ wieder?

2.2 Hilfsmittel

Im $\mathbf{R}^n$ bzw. $\mathbf{C}^n$ (Sammelbegriff $\mathbf{K}^n$) werden Spaltenvektoren betrachtet

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

Eine wichtige Klasse von Normen sind, mit $p \in \mathbf{R}$, $p \geq 1$, die Hölder-Normen

$$\|x\|_p := \left(\sum_{j=1}^n |x_j|^p \right)^{1/p}, \tag{2.2.1}$$

mit den wichtigsten Vertretern

$$\|x\|_1 := \sum_{j=1}^n |x_j| \qquad \text{Summennorm} \tag{2.2.2}$$

$$\|x\|_2 := \sqrt{\sum_{j=1}^n |x_j|^2} \qquad \text{Euklidnorm} \tag{2.2.3}$$

$$\|x\|_\infty := \max_{j=1}^n |x_j|, \qquad \text{Maximumnorm} \tag{2.2.4}$$

Mit diesen Normen $\|\cdot\|_p$ ist $\mathbf{K}^n$ ein Banachraum. Da alle Normen im $\mathbf{K}^n$ äquivalent sind, existieren Konstanten mit

$$\|x\|_p \leq c_{pq}\|x\|_q \quad \forall x \in \mathbf{K}^n,\ p, q \geq 1.$$

Diese c_{pq} hängen im allgemeinen aber von n ab. Lineare Abbildungen von $\mathbf{K}^n \to \mathbf{K}^m$ werden (in der Einheits-Basis) durch Matrizen

$$A = (a_{ij})_{i=1,j=1}^{m,n} \in \mathbf{K}^{m \times n}$$

beschrieben. Jedes Paar von Vektornormen in $\mathbf{K}^m, \mathbf{K}^n$ (mit gleichem Index p) *induziert* eine zugehörige Matrixnorm, die mit dem gleichen Symbol bezeichnet wird

$$\|A\|_p := \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\|x\|_p = 1} \|Ax\|_p. \tag{2.2.5}$$

In (2.2.5) wird das Supremum als Maximum angenommen (Grund?). Die durch (2.2.2ff) *induzierten Normen* sind:

$$\|A\|_\infty = \max_{i=1}^m \sum_{j=1}^n |a_{ij}| \qquad \text{Zeilensummennorm,} \tag{2.2.6}$$

$$\|A\|_1 = \max_{j=1}^n \sum_{i=1}^m |a_{ij}| \qquad \text{Spaltensummennorm,} \tag{2.2.7}$$

$$\|A\|_2 = \varrho(A^*A)^{1/2} = \sqrt{\max\nolimits_{j=1}^n \lambda_j(A^*A)} \qquad \text{Spektralnorm,} \tag{2.2.8}$$

wobei $\lambda_j(A^*A)$ die Eigenwerte von A^*A sind und $A^* = \bar{A}^\top$. Eine Matrixnorm heißt *verträglich* mit den Vektornormen, wenn gilt

$$\|Ax\| \leq \|A\| \|x\| \quad \forall x \in \mathbf{K}^n, \; A \in \mathbf{K}^{m \times n}. \tag{2.2.9}$$

Als obere Schranke für $\|A\|_2$ betrachtet man oft eine einfachere, die

$$\|A\|_F := \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \qquad \text{Frobeniusnorm.} \tag{2.2.10}$$

Sie ist verträglich mit der Euklidnorm, denn

$$\|Ax\|_2 \leq \|A\|_F \|x\|_2.$$

Allerdings ist sie nicht die induzierte Norm, da z.B. $\|I\|_F = \sqrt{n} > \|I\|_2 = 1$. Mit I wird die Einheitsmatrix, $I = (\delta_{ij})$ bezeichnet. Für induzierte Nomen und für $\|\cdot\|_F$ gilt die Produktformel

$$\|AB\| \leq \|A\| \, \|B\| \tag{2.2.11}$$

In dieser Vorlesung werden nur Normen benutzt, *die (2.2.11) erfüllen.* Für viele Überlegungen zu quadratischen Matrizen $A \in \mathbf{K}^{n \times n}$ ist der *Spektralradius*

$$\varrho(A) := \max\{|\lambda_j(A)| : \; \lambda_j(A) \text{ EW zu } A\} \tag{2.2.12}$$

eine wichtige Größe. Wegen $\varrho \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = 0$ ist $\varrho(A)$ sicherlich keine Norm. Der folgende Satz zeigt aber, daß der Spektralradius eine untere Schranke für jede beliebige Matrixnorm ist, sich von geeigneten Normen beliebig wenig unterscheidet und für diagonalisierbare Matrizen sogar gleich einer Norm sein kann. Da hier spezielle Matrixtypen angesprochen werden werden Definitionen und Eigenschaften in Bezug auf die Jordan-Normalform kurz zusammengestellt.

Eigenschaft	Definition	Jordan-Normalform $A = S^{-1}\Lambda S$
A diagonalisierbar	$\exists$ Eigenvektor-Basis	Λ diagonal
A normal	$A^*A = AA^*$	Λ komplex diagonal, S unitär: $S^* = S^{-1}$
A hermitesch	$A^* = A$	Λ reell diagonal, S unitär
A herm.positiv definit	$A^* = A$, $x^*Ax > 0 \forall x \neq 0$	Eigenwerte positiv reell

Satz 2.2.1 *a) Für jede Matrixnorm $\|\cdot\|$ gilt*

$$\varrho(A) \leq \|A\|.$$

b) Für jede Matrix A und jedes $\varepsilon > 0$ existiert eine (spezielle) Norm mit

$$\|A\|_M \leq \varrho(A) + \varepsilon. \tag{2.2.13}$$

c) Für diagonalisierbare Matrizen A kann in (2.2.13) $\varepsilon = 0$ gewählt werden, für hermitesche, reell-symmetrische und normale Matrizen gilt $\|A\|_2 = \varrho(A)$.

Beweis a) x sei Eigenvektor von $A \Rightarrow\ |\lambda|\|x\| = \|\lambda x\| = \|Ax\| \leq \|A\|\|x\|$.
b) Die Jordan-Normalform von A sei

$$\Lambda = SAS^{-1} = \begin{pmatrix} \lambda_1 & \delta_1 & & \\ & \lambda_2 & \delta_2 & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

mit $\delta_i \in \{0, 1\}, i = 1, \ldots, n-1$. Eine weitere Ähnlichkeitstransformation mit der Diagonalmatrix

$$P = \begin{pmatrix} \varepsilon & & & \\ & \varepsilon^2 & & \\ & & \ddots & \\ & & & \varepsilon^n \end{pmatrix}$$

führt Λ über in

$$\tilde{\Lambda} = P^{-1}\Lambda P = (P^{-1}S)A(S^{-1}P).$$

Da $\tilde{\Lambda}$ die gleiche Hauptdiagonale wie Λ, in der Nebendiagonale aber Elemente $\varepsilon\delta_j$ hat, gilt

$$\|\tilde{\Lambda}\|_\infty = \begin{cases} \max_{j=1}^n\{|\lambda_j| + \varepsilon|\delta_j|\} \leq \varrho(\tilde{\Lambda}) + \varepsilon = \varrho(A) + \epsilon, & \text{wenn ein } \delta_j \neq 0, \\ \max_{j=1}^n\{|\lambda_j|\} = \varrho(\tilde{\Lambda}) = \varrho(A), & \text{wenn alle } \delta_j = 0. \end{cases}$$

Für die Matrixnorm

$$\|B\|_M := \|MBM^{-1}\|_\infty, \quad M := P^{-1}S,$$

gilt also b). Bei diagonalisierbaren Matrizen ist $\delta_j \equiv 0$. Dies zeigt die erste Behauptung in c). Die in c) genannten Spezialfälle mit der Norm $\|\cdot\|_2$ folgen aus der Orthogonalität der Eigenvektoren bei normalen Matrizen, d.h., aus $S^{-1} = S^*$. Dann ist $\|S^{-1}\Lambda S\|_2 = \|\Lambda\|_2$. ∎

Der Satz erleichtert folgende Regularitätsaussage für Matrizen der Form $A = I - B$.

Satz 2.2.2 *(Neumannreihe) Unter der Voraussetzung*

$$\varrho(B) < 1 \tag{2.2.14}$$

a) ist das Gleichungssystem $(I - B)x = b$ eindeutig lösbar,

b) gilt $\qquad (I - B)^{-1} = \sum_{j=0}^{\infty} B^j,$

c) gibt es eine Matrixnorm so, daß $\|B\| < 1$ ist. Für diese gilt

$$\|(I - B)^{-1}\| \leq \frac{1}{1 - \|B\|}. \tag{2.2.15}$$

Bemerkung: Dieser Satz wird für Störungsaussagen sehr häufig benutzt, (2.2.15) gilt natürlich in jeder Norm, für die $\|B\| < 1$ ist.

Beweis Da $I - B$ regulär ist genau dann, wenn die Gleichung $x = Bx$ nur die triviale Lösung $x = 0$ hat, d.h. $\lambda = 1$ kein EW von B ist, folgt die Existenz von $(I - B)^{-1}$ aus (2.2.14).

Mit den Teilsummen $\sum_{k=0}^{m} B^k$ gilt

$$S_m := (I - B) \sum_{k=0}^{m} B^k = I - B + B - B^2 + \ldots - B^{m+1} = I - B^{m+1}.$$

Nach Satz 2.2.1 existiert eine Norm mit $\|B\| < 1$, da $\varrho(B) < 1$. In dieser Norm gilt $\|S_m - I\| = \|B^{m+1}\| \leq \|B\|^{m+1} \to 0$ für $m \to \infty$ und somit $S_m \to I$, also

$$S_m \to (I - B) \sum_{k=0}^{\infty} B^k = I, \quad m \to \infty.$$

Durch Betrachtung von $(I - B)^{-1} S_m$ folgt die Behauptung. Außerdem ist

$$\|(I - B)^{-1}\| = \|\sum_{k=0}^{\infty} B^k\| \leq \sum_{k=0}^{\infty} \|B\|^k = \frac{1}{1 - \|B\|}. \qquad \blacksquare$$

2.3 Der Gauß-Algorithmus

Zur Lösung von linearen Gleichungssystemen

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \; i = 1, \ldots, n \quad \Leftrightarrow \quad Ax = b, \tag{2.3.1}$$

$A \in \mathbf{K}^{n \times n}$, gibt es sowohl *direkte* als auch *iterative* Verfahren. Bei den direkten Verfahren formt man das gegebene System in einer endlichen Anzahl von Schritten in eine leicht auflösbare Form um, ändert also insbesondere die Matrix A. Bei iterativen Verfahren (vgl. §2.5) konstruiert man dagegen eine Folge von Näherungen, die gegen die Lösung konvergiert. Hierbei wird die Matrix nicht verändert, was in vielen Fällen ("dünnbesetzte" Matrix) vorteilhaft sein kann.

Der Gauß-Algorithmus ist das Standardverfahren zur direkten Auflösung und beruht auf der Beobachtung, daß die Lösung x von (2.3.1) unverändert bleibt, wenn Gleichungen addiert oder subtrahiert werden (allgemein: Bildung von Linearkombination). Ziel ist die Konstruktion eines einfach lösbaren Systems in Dreieckform:

$$\left.\begin{array}{rrrrcl} r_{11}x_1 & +r_{12}x_2 & +\ldots & +r_{1n}x_n & = & b'_1 \\ & r_{22}x_2 & +\ldots & +r_{2n}x_n & = & b'_2 \\ & & \ddots & \vdots & & \vdots \\ & & & r_{nn}x_n & = & b'_n \end{array}\right\} \quad \Leftrightarrow \quad Rx = b'. \tag{2.3.2}$$

Dieses kann, beginnend mit x_n, schrittweise aufgelöst werden, falls alle $r_{ii} \neq 0$ sind. Die Matrix A sei im folgenden regulär. Dann erhält man ein solches Dreiecksystem (2.3.2) in $n-1$ Schritten auf folgende Weise.

1. für $a_{11} = 0$ wird eine Zeile mit $a_{k1} \neq 0$ gesucht und diese dann mit der ersten vertauscht (*Pivot-Schritt*, k-te Zeile = Pivot-Zeile);

2. es wird dasjenige Vielfache der ersten Zeile zur i-ten Zeile ($i > 1$) addiert, das dort den Koeffizienten bei x_1 zu Null macht (*Eliminationsschritt*, Faktor $-a_{i1}/a_{11}$). Nach Behandlung aller Zeilen $i = 2, \ldots, n$ geht das LGS über in die Form

$$A^{(2)}x = b^{(2)} \quad \text{mit} \quad A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \ldots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix}$$

3. die Schritte 1 und 2 werden auf das Restsystem angewendet mit

$$B^{(2)} := \begin{pmatrix} a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \ddots & \vdots \\ a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix}. \tag{2.3.3}$$

4. Nach $n-1$ Schritten mit Matrizen $A^{(1)} := A, A^{(2)}, \ldots, A^{(n-1)}$ ergibt sich ein Dreicksystem (2.3.2) mit $R = A^{(n)}$, $b' = b^{(n)}$, das sich einfach auflösen läßt.

Die Eliminationskoeffizienten

$$l_{ij} := \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}$$

haben eine besondere Bedeutung, die später erläutert wird. Man speichert sie bei Durchführung des Algorithmus an die freiwerdenden Plätze der $a_{ij}^{(j+1)} = 0$ $(i > j)$.

Algorithmus 2.3.1 Gauß-Algorithmus ohne Pivotisierung, durchgeführt im Speicherplatz der Originalmatrix A (d.h. $l_{ij} \mapsto a_{ij}$),

$$\boxed{\begin{array}{l} \text{für } j := 1, \ldots, n-1: \\ \quad \text{für } i := j+1, \ldots, n: \\ \quad\quad l_{ij} := a_{ij}/a_{jj};\ b_i := b_i - l_{ij}b_j; \\ \quad\quad \text{für } k := j+1, \ldots, n\text{: } a_{ik} := a_{ik} - l_{ij}a_{jk}; \\ x_n := b_n/a_{nn}; \\ \text{für } i := n-1, n-2, \ldots, 1: \\ \quad x_i := \Big(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\Big)/a_{ii}; \end{array}} \tag{2.3.4}$$

Beim Rechenaufwand werden alle arithmetischen Operationen gezählt (kurz FLOP: floating point operation), man gibt dabei aber meist nur den am schnellsten wachsenden Anteil an. Der

Algorithmus enthält drei gestaffelte Schleifen (j, i, k), der Hauptaufwand entsteht daher in der 4. Zeile (innerste Schleife):

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

Dieser Hauptanteil dient aber nur zur Umformung der Matrix A und kann bei einer erneuten Lösung des gleichen LGS $Ay = c$ mit einer anderen rechten Seite c eingespart werden. Im obigen Algorithmus müssen dann nur die die rechte Seite b betreffenden Schritte (3. Zeile des Algor.) wiederholt werden. Dazu wurden die Größen l_{ij} gespeichert. Der Aufwand ist dann

$$\sum_{j=1}^{n-1} 2(n-j) = n(n-1) \text{ Operationen für } b^{(n)}.$$

Die abschließende Auflösung des Dreiecksystems $Rx = b^{(n)}$ benötigt ebenfalls

$$1 + \sum_{i=1}^{n-1} [1 + 2(n-i)] = n^2 \text{ Operationen.}$$

Rechenaufwand Der Gauß-Algorithmus benötigt i.w. $\frac{2}{3} n^3$ arithmetische Operationen. Die nochmalige Auflösung mit neuer rechter Seite b erfordert nur $2n^2$ Operationen.

LR-Zerlegung

Die Umformung beim System $A^{(j)}x = b^{(j)}$ in einem Eliminationsschritts ohne Pivotisierung,

$$\boxed{\begin{array}{l} \text{für } i = j+1, \ldots, n: \\ \quad b_i^{(j+1)} := b_i^{(j)} - l_{ij} b_j^{(j)}; \\ \quad \text{für } k := j+1, \ldots, n: \\ \qquad a_{ik}^{(j+1)} := a_{ik}^{(j)} - l_{ij} a_{jk}^{(j)}; \end{array}}$$

entspricht der Multiplikation dieses Systems mit einer speziellen Matrix:

$$A^{(j+1)}x = L_j^{-1} A^{(j)} x = L_j^{-1} b^{(j)} = b^{(j+1)},$$

wobei

$$L_j^{-1} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{j+1,j} & 1 & & \\ & & \vdots & & \ddots & \\ & & -l_{nj} & & & 1 \end{pmatrix} = L_j^{-1} = I - l_j e_j^{\mathsf{T}}, \quad l_j := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ l_{j+1,j} \\ \vdots \\ l_{nj} \end{pmatrix}. \tag{2.3.5}$$

Wegen $e_j^\top l_j = 0$ ($e_j = j$-ter Einheitsvektor) gilt $(I - l_j e_j^\top)(I + l_j e_j^\top) = I$, also $L_j = I + l_j e_j^\top$. Somit wird die obere Dreieckmatrix $R = A^{(n)}$ erzeugt durch Multiplikation mit L-Matrizen:

$$R = A^{(n)} = L_{n-1}^{-1} A^{(n-1)} = \ldots = L_{n-1}^{-1} \cdots L_1^{-1} A \quad \Leftrightarrow$$

$$A = L_1 L_2 \cdots L_{n-1} R =: LR. \tag{2.3.6}$$

Da die Menge der unteren bzw. oberen Dreieckmatrizen abgeschlossen ist unter Matrixmultiplikation, ist L eine *untere* und R eine *obere* Dreieckmatrix:

$$A = LR = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ * & 1 & & \\ \vdots & & \ddots & \\ * & * & \ldots & 1 \end{pmatrix} \cdot \begin{pmatrix} * & * & \ldots & * \\ 0 & * & & * \\ & & \ddots & \vdots \\ 0 & & & * \end{pmatrix} \tag{2.3.7}$$

Satz 2.3.2 *Der Gaußalgorithmus (2.3.4) (ohne Pivotisierung) erzeugt eine* LR-Zerlegung *der Matrix A. Der Aufwand dafür ist $\frac{2}{3}n^3$ Operationen. Danach reduziert sich jede Lösung von $Ax = b$ auf*

$$b = Ax = L \underbrace{(Rx)}_{b^{(n)}} \quad \Leftrightarrow \quad Lb^{(n)} = b, \; Rx = b^{(n)}, \tag{2.3.8}$$

also auf die Auflösung von zwei LGSen mit Dreieckmatrizen mit $2n^2$ Operationen. Die beiden Faktoren L und R werden bei (2.3.4) wieder in A abgespeichert mit Ausnahme der Hauptdiagonalen von L.

Bemerkung: Die LR-Zerlegung kann auch direkt erzeugt werden ohne die Zwischenmatrizen $A^{(j)}$. Die Identität $A = L \cdot R$ wird dabei als Gleichungssystem für L und R aufgefaßt:

$$a_{ij} = \sum_{k=1}^{\min\{i,j\}} l_{ik} r_{kj} \quad \Longleftrightarrow \quad \begin{cases} a_{ij} = \sum\limits_{k=1}^{j-1} l_{ik} r_{kj} + l_{ij} r_{jj}, & i > j, \\ a_{ij} = \sum\limits_{k=1}^{i-1} l_{ik} r_{kj} + r_{ij}, & i \le j. \end{cases} \tag{2.3.9}$$

Es können verschiedene Indexfolgen (i, j) angegeben werden, für die man diese Gleichungen direkt nach den l_{ij} (obere Gleichungen) bzw. r_{ij} (untere Gleichungen) auflösen kann. Die Effizienz ist dabei sprach- und maschinenabhängig (Matrizen zeilen-/spaltenweise gespeichert?) und führt auf die Algorithmen von Banachiewicz, Crout, etc.

Pivotisierung

Im allgemeinen ist der Gauß-Algorithmus nicht ohne Zeilenvertauschungen durchführbar. Daher stellen sich zwei Fragen:

— Für welche Problemklassen ist keine Pivotisierung nötig?
— Welche Pivotstrategien sind für die übrigen sinnvoll?

Bei der Diskussion ist die Tatsache hilfreich, daß der Gauß-Algorithmus bestimmte Strukturen bzw. Eigenschaften der Matrix A erhält.

Definition 2.3.3 *A sei eine $n \times n$-Matrix. A heißt*

— streng diagonaldominant*, wenn*

$$|a_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|, \quad i = 1, \dots, n. \tag{2.3.10}$$

— $(2m+1)$-Bandmatrix *(Tridiagonalmatrix für $m = 1$), wenn*

$$a_{ij} = 0 \qquad \text{für } |i-j| > m.$$

Bei definiten und streng diagonaldominanten Matrizen sind insbesondere die Hauptdiagonalelemente von Null verschieden. Die Erhaltung bestimmter Strukturen ermöglicht außerdem oft eine Reduktion des Aufwands beim Gauß-Algorithmus.

Satz 2.3.4 *a) In der Matrix A gelte $a_{11} \neq 0$. Wenn die Matrix A hermitesch, positiv (negativ) definit, streng diagonaldominant oder $2m+1$-Bandmatrix ist, dann hat im Gauß-Algorithmus die Restmatrix $B^{(2)}$ (2.3.3) wieder die gleiche Eigenschaft.*

b) Bei definiten oder streng diagonaldominanten Matrizen ist der Gauß-Algorithmus ohne Pivotisierung durchführbar.

c) Für eine positiv (negativ) definite Matrix $A \in \mathbf{K}^{n\times n}$ existiert genau eine Dreieckzerlegung $A = \tilde{L}\tilde{R}$ $(A = -\tilde{L}\tilde{R})$ mit $\tilde{l}_{ii} = \tilde{r}_{ii} > 0$, $i = 1, \dots, n$, und $\tilde{L}^* = \tilde{R}$. Diese Cholesky-Zerlegung *$A = \tilde{L}\tilde{L}^*$ erfordert etwa den halben Rechenaufwand der gewöhnlichen LR-Zerlegung.*

Beweis a) Für $a_{11} \neq 0$ erzeugt der erste Eliminationsschritt folgende Umformung. Aus

$$A = A^{(1)} = \begin{pmatrix} a_{11} & z^{\mathsf{T}} \\ s & B^{(1)} \end{pmatrix} \text{ mit } s = \begin{pmatrix} a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}, \; z^{\mathsf{T}} = (a_{12}, \dots, a_{1n}), \quad \text{wird}$$

$$A^{(2)} = \begin{pmatrix} a_{11} & z^{\mathsf{T}} \\ 0 & B^{(2)} \end{pmatrix}, \quad B^{(2)} := \Big(a_{ik} - \frac{1}{a_{11}} a_{i1} a_{1k}\Big)_{i,k=2}^{n} = B^{(1)} - \frac{1}{a_{11}}\, s z^{\mathsf{T}}.$$

Daraus ergeben sich die Aussagen. Als Beispiel sei A hermitesch. Dann gilt $a_{11} \in \mathbf{R}$, $s = \bar{z}$ und $B^{(1)} = B^{(1)^*}$. Daher ist auch $B^{(2)}$ hermitesch.

b) Exemplarisch wird die Definitheit nachgewiesen. Bei positiv definiter Matrix A gilt $x^* A x > 0$ für alle $x \in \mathbf{K}^n, x \neq 0$. Für $x = (1, 0, \dots, 0)^{\mathsf{T}}$ folgt $a_{11} > 0$, im ersten Schritt war also keine Zeilenvertauschung notwendig. Nun sei mit $y \in \mathbf{K}^{n-1}, y \neq 0$, speziell

$$\hat{x} = \begin{pmatrix} \frac{-1}{a_{11}} s^* y \\ y \end{pmatrix} \Rightarrow A\hat{x} = \begin{pmatrix} a_{11} & s^* \\ s & B^{(1)} \end{pmatrix} \begin{pmatrix} \frac{-1}{a_{11}} s^* y \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ B^{(1)} y - \frac{1}{a_{11}} s s^* y \end{pmatrix} = \begin{pmatrix} 0 \\ B^{(2)} y \end{pmatrix}.$$

Also ist $y^* B^{(2)} y = \hat{x}^* A \hat{x} > 0$, d.h., $B^{(2)}$ wieder positiv definit.

c) Die Existenz einer Dreieckzerlegung $A = \tilde{L}\tilde{R}$ mit $\tilde{l}_{ii} = \tilde{r}_{ii} > 0$ folgt sofort aus dem zweiten

Beweisteil. Es kann nämlich mit $D := \mathrm{diag}(\sqrt{r_{11}}, \ldots, \sqrt{r_{nn}})$ gesetzt werden $\tilde{L} := LD$, $\tilde{R} := D^{-1}R$. Durch Induktion läßt sich auch zeigen, daß diese Zerlegung eindeutig ist. Wegen $\tilde{L}\tilde{R} = A = A^* = \tilde{R}^*\tilde{L}^*$ erhält man dann sofort $\tilde{L} = \tilde{R}^*$ und $\tilde{R} = \tilde{L}^*$. ■

Damit ist in vielen wichtigen Fällen sichergestellt, daß der Gauß-Algorithmus ohne Pivotisierung durchführbar ist.

Bemerkung: Bei großen praktischen Problemen können LGSe mit $n \gg 10^6$ Unbekannten auftreten. Die Matrizen haben dabei aber nur wenige Elemente $a_{ij} \neq 0$ (Anteil $< 10\%$, "dünnbesetzt"). Bei anderen Besetzungen der Matrix, außer der Bandstruktur, wird die ursprüngliche dünne Struktur bei der Elimination in der Regel aufgefüllt ("fill-in"). Der Grad des Auffüllens hängt aber von der Numerierung der Gleichungen/Variablen ab und kann so beeinflußt werden. Bei geeigneter Implementierung bestimmt v.a. die Zahl der nichttrivialen Elemente der LR-Zerlegungder den Rechenaufwand (z.B. Bandstruktur: $O(m^2n)$). Ohne die Ausnutzung solcher Struktureigenschaften wären sehr große Probleme auch auf modernen Supercomputern nicht lösbar.

Die aktuelle Top-500-Liste der Supercomputer (Quelle: Netlib/HRZ Mannheim 11'97) weist folgende tatsächlichen Leistungen aus (GFLOPS=10^9 Operationen pro Sekunde). Die genannte Dimension n gibt dabei die Größe eines vollbesetzten LGS an, das in 1 sec Rechenzeit gelöst werden kann. Bei einer Rechenzeit von einem Tag sind die entsprechenden Dimensionen (nur) um den Faktor 44 größer.

Platz	Typ	Ort	Prozessoren	GFLOPS	n
1	Intel	Sandia/USA	9152	1338	11019
2	SGI/Cray T3E	USA	1248	634	8590
3	SGI/Cray T3E	Meteo/GrBr	840	430	7548
9	SGI/Cray T3E	MPI/Garching	672	196	5808
??	IBM SP 2	Uni Marburg	35	6.3	1847

In der Theorie ist eine Zeilenvertauschung beim Gauß-Algorithmus nur dann erforderlich, wenn das Pivotelement verschwindet, d.h., $a_{jj}^{(j)} = 0$. Bei der praktischen Rechnung im Computer mit endlichen Zahldarstellungen sollten aber auch zu kleine Pivotelemente vermieden werden, wie das folgende Beispiel zeigt (genauere Analyse in §7.)

Beispiel 2.3.5 Alle (Zwischen-)Ergebnisse werden in der Gleitkomma-Darstellung auf zwei Stellen gerundet. Aus

$$\begin{array}{rcrcr} 0.01\,x_1 & + & 2\,x_2 & = & 1 \\ x_1 & + & x_2 & = & 3 \end{array} \quad \text{wird} \quad \begin{array}{rcrcr} 0.01\,x_1 & + & 2x_2 & = & 1 \\ 0\cdot x_1 & - & 200\,x_2 & = & -97 \end{array}, \tag{2.3.11}$$

denn $l_{21} = a_{21}/a_{11} = 1/0.01 = 1.0 \cdot 10^2$ (Gleitpunktdarstellung) $\Rightarrow$

$$\begin{aligned} a_{22}^{(2)} &= a_{22} - l_{21}a_{12} = 1 - 100 \cdot 2 = -199 \qquad \text{Rundung: } \widetilde{a_{22}^{(2)}} = -2 \cdot 10^2 \\ b_2^{(2)} &= b_2 - l_{21}b_1 = 3 - 100 = -97 \quad \text{(ist exakt).} \end{aligned}$$

Damit erhält man

$$\widetilde{x_2} = -97/(-200) \doteq 0.49, \quad \widetilde{x_1} = (1 - 2 \cdot 0.49)/0.01 = 2.0.$$

Die exakte Lösung ist jedoch $(2.512.., 0.487..)$. Vertauscht man die beiden Gleichungen in (2.3.11), so ergibt die Elimination dagegen

$$\begin{array}{rrl} x_1 & +x_2 & = 3 \\ 0.01x_1 & +2x_2 & = 1 \end{array} \quad \rightarrow \quad \begin{array}{rrl} x_1 & +x_2 & = 3 \\ & 2x_2 & = 0.97 \end{array} \quad \rightarrow \quad \begin{array}{rl} x_1 & = 2.5 \\ x_2 & = 0.49 \end{array}$$

Um die gezeigten Ungenauigkeiten zu verhindern, verwendet man *Pivot-Strategien*: Dabei wird im Schritt j zur Vertauschung mit Zeile j die Zeile $p \in \{j, \dots, n\}$ nach folgender Regel gewählt:

1. Spaltenpivotisierung, absolutes Maximum:
$$|a_{pj}^{(j)}| = \max_{k=j}^{n} |a_{kj}^{(j)}|.$$
2. Spaltenpivotisierung, relatives Maximum:
$$\frac{|a_{pj}^{(j)}|}{\sum_{\nu=j}^{n} |a_{p\nu}^{(j)}|} = \max_{k=j}^{n} \frac{|a_{kj}^{(j)}|}{\sum_{\nu=j}^{n} |a_{k\nu}^{(j)}|}.$$
3. Vollständige Pivotisierung, zusätzlich werden auch Spaltenvertauschungen (Umnumerierung der Variablen) vorgenommen
$$|a_{pq}^{(j)}| = \max_{i,k=j}^{n} |a_{ik}^{(j)}|.$$

Wegen des hohen Aufwands werden die Strategien 2) und 3) nur in Ausnahmefällen benutzt. Strategie 2) unterdrückt den Einfluß der Multiplikation von Zeilen mit einer Konstanten (Skalierung). Ein ähnlicher Effekt ist bei 1) erzielbar durch *Äquilibrierung* der Matrix, bei der zu Beginn alle Zeilen auf Summennorm 1 skaliert werden. Auch die einfache Strategie 1) erhöht den Aufwand um 50% :

Rechenaufwand für den Gauß-Algorithmus mit Pivotisierung 1): n^3 Operationen

Inverse Matrix A^{-1}:

Die Lösung von $Ax = b$ läßt sich kompakt in der Form $x = A^{-1}b$ angeben. Bei praktischer Rechnung lohnt es sich aber selbst zur Lösung vieler LGSe $Ax^{(j)} = b^{(j)}$, $j = 1, 2, \dots$, **nicht**, die Inverse A^{-1} explizit zu berechnen. Der Rechenaufwand für das Produkt $x = A^{-1}b$ ist der *gleiche*, wie der zur Lösung der beiden Dreiecksysteme bei $LRx = b$, nämlich $2n^2$ (vgl. Berechnung nach (2.3.4)), der Rechenaufwand für A^{-1} amortisiert sich daher nie.

Manchmal ist es aber für theoretische Aussagen günstig, die Elemente von A^{-1} zu kennen (z.B. Input-Output-Analyse: Wechselwirkungen zwischen Produktionszweigen). Bei gegebener LR-Zerlegung von A ergeben sich die Spalten von

$$A^{-1} = C = \left(c^{(1)}, \dots, c^{(n)}\right) \quad \text{aus} \quad LRc^{(j)} = e^{(j)},\ j = 1, \dots, n. \tag{2.3.12}$$

Unter Ausnutzung der speziellen Gestalt der Einheitsvektoren $e^{(j)}$ benötigt man zur Berechnung $\frac{4}{3}n^3$ Operationen zusätzlich zur LR-Zerlegung, also $2n^3$ Operationen insgesamt.

2.4 Fehlerausbreitung, Kondition

Die Auswirkungen von Rundungsfehlern bei der Durchführung des Gauß-Algorithmus werden später (§7) diskutiert. Aber auch die Verfälschung von Eingangsdaten (z.B. durch Rundung auf endliche Genauigkeit) kann sich bei fast-singulären LGSen katastrophal auf die Lösung auswirken. Zur quantitativen Untersuchung dieser Auswirkungen wird angenommen, daß statt des ursprünglichen Problems $Ax = b$ mit regulärer Matrix A das Problem

$$(A + A')\tilde{x} = b + b', \quad \tilde{x} = x + x' \tag{2.4.1}$$

mit gestörter Matrix $\tilde{A} = A + A'$ und gestörter rechter Seite $\tilde{b} = b + b'$ exakt gelöst wird. Die Rundungsfehleranalyse des Gauß-Algorithmus wird übrigens in §7 auf diesen Fall zurückgeführt.

Zur Abschätzung der Störung $x' = \tilde{x} - x$ von x wird die Gleichung mit A^{-1} multipliziert:

$$(I + A^{-1}A')\tilde{x} = A^{-1}(b + b') = x + A^{-1}b'. \tag{2.4.2}$$

Nach Satz 2.2.2 (v. Neumann) ist dieses LGS lösbar, wenn in einer Norm gilt

$$\|A^{-1}A'\| \le \|A^{-1}\|\,\|A'\| =: q < 1. \tag{2.4.3}$$

Dann folgt die Schranke

$$\|\tilde{x}\| \le \frac{\|x + A^{-1}b'\|}{1 - q}. \tag{2.4.4}$$

Nach (2.4.2) gilt auch $x' = \tilde{x} - x = A^{-1}b' - A^{-1}A'\tilde{x}$. Daraus folgt mit (2.4.3) und (2.4.4)

$$\begin{aligned} \|x'\| = \|\tilde{x} - x\| &\le \|A^{-1}b'\| + q\|\tilde{x}\| \\ &\le \|A^{-1}b'\| + \frac{q}{1-q}(\|x\| + \|A^{-1}b'\|) \\ &\le \frac{q}{1-q}\|x\| + \frac{1}{1-q}\|A^{-1}\|\|b'\|. \end{aligned} \tag{2.4.5}$$

Besonders aussagekräftig sind Abschätzungen, die *skalierungsinvariant* sind, bei einer Multiplikation des LGS mit Konstanten also unverändert bleiben. Zu diesem Zweck betrachtet man relative Fehler. Mit $\|b\| = \|Ax\| \le \|A\|\|x\|$ folgt aus (2.4.5):

$$\frac{\|x'\|}{\|x\|} \le \frac{q}{1-q} + \frac{\|A^{-1}\| \cdot \|A\|}{1-q}\,\frac{\|b'\|}{\|b\|}.$$

Bezieht man sich auch bei $A' = \tilde{A} - A$ auf den relativen Fehler, etwa in

$$q = \|A^{-1}\| \|A'\| = \|A^{-1}\| \cdot \|A\| \, \frac{\|A'\|}{\|A\|},$$

dann taucht in der Abschätzung außer den Fehlern nur **eine** zusätzliche Konstante auf:

Definition 2.4.1 *Zu einer gegebenen Matrixnorm $\|\cdot\|$ heißt*

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

die Konditionszahl *der Matrix A. Bei Verwendung spezieller Normen übernimmt man den Index, z.B.,* $\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Die Konditionszahl beschreibt die Empfindlichkeit des LGS gegenüber Störungen. Für verträgliche Matrixnormen gilt $\kappa(A) \geq 1$ wegen $1 \leq \|I\| = \|AA^{-1}\| \leq \|A\| \|A^{-1}\|$. Die gerade durchgeführte Diskussion hat folgende Fehlerabschätzung bewiesen:

Satz 2.4.2 *Im gestörten System (2.4.1) werden folgende relative Fehler betrachtet:*

$$\varphi_x := \frac{\|\tilde{x} - x\|}{\|x\|} = \frac{\|x'\|}{\|x\|}, \quad \varphi_b := \frac{\|\tilde{b} - b\|}{\|b\|} = \frac{\|b'\|}{\|b\|}, \quad \varphi_A := \frac{\|\tilde{A} - A\|}{\|A\|} = \frac{\|A'\|}{\|A\|}.$$

Damit ist das gestörte LGS (2.4.1) eindeutig lösbar für

$$\kappa(A) \cdot \varphi_A < 1. \tag{2.4.6}$$

Der relative Fehler in der Lösung ist beschränkt durch

$$\varphi_x \leq \frac{\kappa(A)}{1 - \kappa(A)\varphi_A} \, \big(\varphi_A + \varphi_b\big). \tag{2.4.7}$$

Für große $\kappa(A) \gg 1$ ist das Problem *schlecht konditioniert.* Dann können schon kleine Störungen der Daten zu erheblichen Änderungen der Lösung des LGS führen.

Bei der Berechnung einer Lösung von (2.4.1) können weitere (Rundungs-) Fehler auftreten. Die Genauigkeit einer Näherungslösung y kann mit Hilfe des *Defekts* oder *Residuums*

$$r := \tilde{A}y - \tilde{b}, \qquad \tilde{A} = A + A', \; \tilde{b} = b + b', \tag{2.4.8}$$

überprüft werden. Dazu gilt

Satz 2.4.3 *Mit den Bezeichnungen von Satz 2.4.2 gilt für den relativen Fehler einer Näherung y zur Lösung $x = A^{-1}b$ mit Defekt r (2.4.8) die Abschätzung*

$$\frac{\|y - x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\varphi_A} \left(\varphi_A + \varphi_b + \frac{\|r\|}{\|b\|}\right). \tag{2.4.9}$$

Beweis y unterscheidet sich von $\tilde{x}$ aus Satz 2.4.2 folgendermaßen:

$$\tilde{A}\tilde{x} = \tilde{b}, \quad \tilde{A}y = \tilde{b} + r \quad \Rightarrow \quad y - \tilde{x} = \tilde{A}^{-1}r.$$

Da (2.4.2), (2.4.4) der Abschätzung

$$\|\tilde{A}^{-1}\| = \|(A + A')^{-1}\| \le \frac{\|A^{-1}\|}{1 - \|A^{-1}A'\|},$$

entsprechen, folgt mit den Schranken aus Satz 2.4.2 die Behauptung,

$$\frac{\|y - x\|}{\|x\|} \le \frac{\|\tilde{x} - x\|}{\|x\|} + \frac{\|\tilde{A}^{-1}r\|}{\|x\|} \le \varphi_x + \frac{\|A^{-1}\|}{1 - \kappa(A)\varphi_A} \cdot \frac{\|A\|\|r\|}{\|b\|}. \qquad \blacksquare$$

Auch hier geht die Konditionszahl entscheidend ein. Wegen dieser Bedeutung sollte bei der praktischen Lösung eines LGS immer auch die Kondition der Matrix (ab-)geschätzt werden, um die Genauigkeit der tatsächlich berechneten Lösung beurteilen zu können.

Schranken für $\kappa(A)$

Da die Zeilensummennorm $\|\cdot\|_\infty$ einer Matrix einfacher bestimmt werden kann als, z.B., $\|\cdot\|_2$, werden hier Ansätze diskutiert, diese Norm der Inversen $\|A^{-1}\|_\infty$ abzuschätzen:

1. Für streng diagonaldominante Matrizen (2.3.10) ist $A = D(I - D^{-1}B)$ mit $D = \mathrm{diag}(a_{ii})$ und $\|D^{-1}B\|_\infty < 1$. Aus Satz 2.2.2 folgt daher

$$\kappa_\infty(A) \le \|A\|_\infty \frac{\|D^{-1}\|_\infty}{1 - \|D^{-1}B\|_\infty}.$$

2. Da die LR-Zerlegung $A = LR$ bei der Auflösung berechnet wird, kann mit der folgenden Methode mit akzeptablem Rechenaufwand $O(n^2)$ auch eine Schranke für $\|A^{-1}\|_\infty \le \|L^{-1}\|_\infty \|R^{-1}\|_\infty$ berechnet werden. Grundlage sind die folgenden Eigenschaften nichtnegativer Matrizen, die hier nur für R formuliert werden. Ungleichungen und Beträge bei Matrizen sind dabei komponentenweise zu verstehen.

   (a) Für $B = (b_{ij})$ mit $b_{ij} \ge 0 \,\forall i, j$ (kurz: $B \ge 0$) gilt

   $$\|B\|_\infty = \|Be\|_\infty \,, \; e = (1, 1, \ldots, 1)^\mathsf{T}.$$

   (b) Für $C^{-1} \ge 0$ ist $\|C^{-1}\|_\infty = \|C^{-1}e\|_\infty = \|z\|_\infty$, wobei z das LGS $Cz = e$ löst. Hiermit kann die Norm $\|C^{-1}\|_\infty$ berechnet werden, ohne C^{-1} explizit zu kennen.

   (c) Die Norm $\|\cdot\|_\infty$ ist monoton, d.h. aus

   $$|b_{ij}| \le |c_{ij}| \;\forall i, j \;(\text{kurz: } |B| \le |C|), \quad \text{folgt} \quad \|B\|_\infty \le \|C\|_\infty.$$

(d) Zerlegt man $R = D(I - B)$ mit $D = \operatorname{diag}(r_{ii})$, dann ist B strikt obere Dreieckmatrix mit $\rho(B) = 0 \Rightarrow B^n = 0$. Aus dem Satz 2.2.2 (Neumannreihe) folgt durch komponentenweise Abschätzung

$$|R^{-1}| = |(I-B)^{-1}D^{-1}| = |\sum_{j=0}^{n-1} B^j||D^{-1}| \le \sum_{j=0}^{n-1} |B|^j|D|^{-1} = \Big(|D|(I-|B|)\Big)^{-1}.$$

Die Eigenschaften (a) - (d) führen auf

Satz 2.4.4 *$R \in \mathbf{R}^{n\times n}$ sei eine reguläre obere Dreieckmatrix. Der (positive) Vektor $z \in \mathbf{R}^n$ sei definiert durch das LGS*

$$|r_{ii}|z_i - \sum_{j=i+1}^{n} |r_{ij}|z_j = 1, \quad i = n, n-1, \ldots, 1. \tag{2.4.10}$$

Dann gilt

$$\|R^{-1}\|_\infty \le \|z\|_\infty.$$

Beispiel 2.4.5

$$A = \begin{pmatrix} 2 & 3 & -5 \\ 4 & 8 & -3 \\ -6 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & 5 & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 & -5 \\ 0 & 2 & 7 \\ 0 & 0 & -46 \end{pmatrix} = L \cdot R.$$

Zur Abschätzung von $\|R^{-1}\|_\infty$ ist zu lösen

$$\begin{pmatrix} 2 & -3 & -5 \\ 0 & 2 & -7 \\ 0 & 0 & 46 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1.418 \\ 0.576 \\ 0.0217 \end{pmatrix}.$$

Dies führt auf die Schranke $\|R^{-1}\|_\infty \le \|z\|_\infty \doteq 1.418$. Für $\|L^{-1}\|_\infty$ ergibt sich analog

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & -5 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 19 \end{pmatrix}.$$

Hier erhält man die Schranke $\|L^{-1}\|_\infty \le \|u\|_\infty = 19$, zusammen mit der ersten führt sie auf die Abschätzung $\|A^{-1}\|_\infty \le 26.95$ (zum Vergleich: $\|A^{-1}\|_\infty \doteq 0.451$). Mit $\|A\|_\infty = 15$ folgt die Schranke $\kappa_\infty(A) \le 404.25$ für die Konditionszahl.

Spalten-Pivotisierung (hier: Vertauschung der Zeilen 1 und 3) verbessert auch die berechneten Schranken, es ist

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} A = \begin{pmatrix} -6 & 1 & 4 \\ 4 & 8 & -3 \\ 2 & 3 & -5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{2}{3} & 1 & 0 \\ -\frac{1}{3} & \frac{5}{13} & 1 \end{pmatrix} \begin{pmatrix} -6 & 1 & 4 \\ 0 & \frac{26}{3} & -\frac{1}{3} \\ 0 & 0 & -\frac{46}{13} \end{pmatrix} =: \hat{L}\hat{R}.$$

Als Abschätzung erhält man jetzt $\|\hat{R}^{-1}\|_\infty \le 0.376$, $\|\hat{L}^{-1}\|_\infty \le 2.616$ und daher die viel schärfere Schranke $\|A^{-1}\|_\infty \le 0.984$, da $\|A^{-1}\|_\infty = \|(PA)^{-1}\|_\infty$ mit $P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = P^{-1}$.

2.5 Iterationsverfahren für lineare Gleichungssysteme

Wenn selbst die exakte Lösung eines LGSs nur eine Approximation für ein Ausgangsproblem darstellt, reicht auch hier eine Näherungslösung aus. Für lineare Gleichungssysteme der Form $(I-B)x = r$ (vgl. Input-Output-Analyse) mit "kleiner" Matrix B, d.h., Spektralradius $\rho(B) < 1$, ist eine explizite Darstellung der Lösung durch Satz 2.2.2 (Neumannreihe) bekannt,

$$x = (I-B)^{-1}r = \sum_{j=0}^{\infty} B^j r = r + B\sum_{j=0}^{\infty} B^j r. \tag{2.5.1}$$

Die Partialsummen dieser Reihe sind daher konvergierende Näherungen für x und können über folgende Rekursion bzw. *Iteration* $x^{(0)} := 0$,

$$x^{(k+1)} := Bx^{(k)} + r, \quad k = 0, 1, \dots, \tag{2.5.2}$$

berechnet werden: $x^{(1)} = r,\ x^{(2)} = r + Br,\ x^{(3)} = r + Br + B^2r, \dots$. Diese Konvergenzaussage läßt sich erheblich verallgemeinern. Die Iteration (2.5.2) dient nämlich zur Bestimmung eines *Fixpunkts* der Abbildung $x \mapsto g(x) := Bx + r$, d.h. eines Vektors mit

$$z = g(z), \qquad g : \mathbf{R}^n \to \mathbf{R}^n. \tag{2.5.3}$$

Allgemein gilt für *kontrahierende* Abbildungen g der

Satz 2.5.1 (Banachscher Fixpunksatz) *Gegeben sei ein Banachraum H und eine Abbildung $g: H \to H$. Gelten auf einer abgeschlossenen Teilmenge $\Omega \subset H$ und für $q \in \mathbf{R}, 0 < q < 1$, die Aussagen*

$$g(\Omega) \subseteq \Omega \qquad (\text{"}g \text{ bildet } \Omega \text{ auf sich ab"}) \tag{2.5.4}$$

$$\|g(x) - g(y)\| \le q\|x - y\| \ \forall x, y \in \Omega, \quad (\text{"}g \text{ ist kontrahierend"}) \tag{2.5.5}$$

dann besitzt g genau *einen Fixpunkt $z = g(z) \in \Omega$. Dieser kann mit einem beliebigen Startwert $x^{(0)} \in \Omega$ durch die* Iteration

$$x^{(k+1)} := g(x^{(k)}),\ k = 0, 1, \dots, \tag{2.5.6}$$

berechnet werden, es ist $\lim_{k\to\infty} x^{(k)} = z$*. Quantitativ gelten die Fehlerabschätzungen*

$$\|x^{(k)} - z\| \le \frac{q}{1-q}\|x^{(k)} - x^{(k-1)}\| \quad (\text{"a-posteriori"}) \tag{2.5.7}$$

$$\le \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\| \quad (\text{"a-priori"}). \tag{2.5.8}$$

Bemerkungen: 1) Eine Bedingung der Form (2.5.5) ($q > 0$) heißt *Lipschitz-Bedingung*.

2) Typische Anwendungen der Fehlerschranken sind folgende. (2.5.7): Verwendung als Abbruchkriterium in der Iteration, d.h., wenn $x^{(k)}$ berechnet ist. (2.5.8): Abschätzung der für eine bestimmte Genauigkeit erforderliche Anzahl von Iterationen, d.h., des Aufwands.

Beweis Wegen (2.5.5) gilt

$$\begin{aligned}\|x^{(k+1)}-x^{(k)}\| &= \|g(x^{(k)})-g(x^{(k-1)})\| \le q\|x^{(k)}-x^{(k-1)}\| \le \dots \\ &\le q^k\|x^{(1)}-x^{(0)}\|.\end{aligned}$$

Analog ergibt sich, für beliebiges $m>0$, aus der Dreiecksungleichung die Schranke

$$\begin{aligned}\|x^{(k+m)}-x^{(k)}\| &\le \sum_{j=0}^{m-1}\|x^{(k+j+1)}-x^{(k+j)}\| \le \sum_{j=1}^{m} q^j\|x^{(k)}-x^{(k-1)}\| \\ &\le \frac{q}{1-q}\|x^{(k)}-x^{(k-1)}\| \le \frac{q^k}{1-q}\|x^{(1)}-x^{(0)}\| \ \to 0,\ k\to\infty. \qquad (2.5.9)\end{aligned}$$

Daher ist $\{x^{(k)}\}$ Cauchyfolge, wegen der Vollständigkeit von Ω also auch konvergent, der Grenzwert $\hat{x}=\lim_{k\to\infty}x^{(k)}$ existiert also. Aus der Stetigkeit von g folgt nun

$$\hat{x}=\lim_{k\to\infty}x^{(k)}=\lim_{k\to\infty}x^{(k+1)}=\lim_{k\to\infty}g(x^{(k)})=g(\lim_{k\to\infty}x^{(k)})=g(\hat{x}),$$

d.h. $\hat{x}$ ist Fixpunkt von g. Dieser ist eindeutig, denn wegen $q<1$ gilt

$$\|\hat{x}-z\|=\|g(\hat{x})-g(z)\|\le q\|\hat{x}-z\| \quad \Rightarrow \quad \hat{x}=z.$$

Nach (2.5.9) folgen aus $x^{(k)}-z=\lim_{m\to\infty}(x^{(k)}-x^{(k+m)})$ die Schranken wie

$$\|x^{(k)}-z\|\le\sum_{j=0}^{\infty}q^{j+1}\|x^{(k)}-x^{(k-1)}\|=\frac{q}{1-q}\|x^{(k)}-x^{(k-1)}\|. \qquad \blacksquare$$

Dieser Satz ist die Grundlage der meisten Konvergenzaussagen bei Iterationsverfahren, auch bei nichtlinearen Problemen (vgl. §6). Wegen Satz 2.2.1 kann die Bedingung (2.5.5), $\|B\|<1$, bei der affin-linearen Funktion (2.5.2) natürlich durch die Bedingung $\rho(B)<1$ an den Spektralradius ersetzt werden. Außerdem gilt hier die Umkehrung

Satz 2.5.2 *Für $\rho(B)>1$ gibt es Startvektoren $x^{(0)}\in\mathbf{R}^n$ so, daß (2.5.2) divergiert.*

Beweis (-idee): Wähle $x^{(0)}=z+y$ so, daß $By=\lambda y$ mit $|\lambda|>1$. $\blacksquare$

Außer in einigen Spezialfällen liegt die Matrix A des LGSs natürlich nicht in der Form $A=I-B$ vor. Oft kann man sie aber als *Störung* eines Hauptteils M schreiben. Von praktischem Interesse sind dabei allerdings nur Hauptteile M, die auf einfach auflösbare Gleichungssysteme führen (M, z.B., Diagonal-, Dreieck-Matrix).

Definition 2.5.3 *Die Zerlegung einer Matrix*

$$A=M-N \qquad (2.5.10)$$

heißt regulär, *wenn M regulär ist und wenn gilt*

$$\rho(M^{-1}N) \ < \ 1. \qquad (2.5.11)$$

Nach Satz 2.5.1 definiert jede reguläre Zerlegung von A ein konvergentes Iterationsverfahren:

$$b = Ax = Mx - Nx \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b,$$

d.h. (2.5.3) mit $B := M^{-1}N = I - M^{-1}A,\ r := M^{-1}b$. Dazu gehört ein Iterationsverfahren, das man bei der Rechnung in der Form

$$Mx^{(k+1)} := Nx^{(k)} + b \iff M(x^{(k+1)} - x^{(k)}) := b - Ax^{(k)} \tag{2.5.12}$$

$x^{(0)} := 0$, durchführt, allerdings besser in der äquivalenten Form

$$x^{(k+1)} := M^{-1}Nx^{(k)} + M^{-1}b \iff x^{(k+1)} := M^{-1}b + (I - M^{-1}A)x^{(k)}. \tag{2.5.13}$$

analysiert (keine Berechnung von M^{-1}!). Eine Zerlegung ist umso günstiger, je kleiner der Konvergenzfaktor $\rho(M^{-1}N)$ ist und je weniger Aufwand ein Schritt (2.5.12) kostet. Beide Kriterien lassen sich auf folgende Weise miteinander in Beziehung setzen.

Bemerkung: Benutzt man in Satz 2.5.1 die bestmögliche Konstante $q = \rho(M^{-1}N)$, so folgt aus (2.5.8), daß zur Verkleinerung des Startfehlers $\|x^{(1)} - x^{(0)}\|$ auf die Endgenauigkeit $\|x^{(k)} - z\| \leq 10^{-\ell}\|x^{(1)} - x^{(0)}\|$ ungefähr

$$k = \frac{\ell}{-\log_{10}\rho(M^{-1}N)} \text{ Schritte}$$

nötig sind. Die Reduktion des Fehlers um $\frac{1}{10}$ erfordert bei einer gegebenen Zerlegung den Aufwand

$$\frac{1}{-\log_{10}\rho(M^{-1}N)} \times [\text{Aufwand für einen Schritt (2.5.12)}]. \tag{2.5.14}$$

Verschiedene Zerlegungen von A lassen sich daher anhand dieses "Effizienz-Maßes" vergleichen.

Eine einfache Anwendung der Iteration beim Gauß-Algorithmus ist die *Nachiteration:* Bei sehr schlecht konditionierten linearen Gleichungssystemen mit $\kappa(A) \gg 1$ kann die mit Hilfe des Gaußalgorithmus berechnete Lösung durch Rundungsfehler stark verfälscht sein. Dann gilt für die berechnete LR-Zerlegung nur noch $\tilde{L}\tilde{R} \cong A$. In diesem Fall kann diese Lösung durch (2.5.12) verbessert werden mit $M := \tilde{L}\tilde{R}$ wenn $\rho(I - \tilde{R}^{-1}\tilde{L}^{-1}A) < 1$ (vgl. auch Satz 2.4.3), d.h., mit der Nachiteration

$$\tilde{L}\tilde{R}(x^{(k+1)} - x^{(k)}) = b - Ax^{(k)}.$$

Der Speicherbedarf verdoppelt sich allerdings, da die Originalmatrix A zur Berechnung des Defekts $b - Ax^{(k)}$ benötigt wird. Die Defektberechnung sollte mit doppelter Zahlengenauigkeit erfolgen.

Gesamt- und Einzelschrittverfahren

Beispiel 2.5.4 Betrachtet werde das folgende 2×2-Gleichungssystem mit Lösung $(2,1)^{\mathsf{T}}$.

$$\begin{array}{rcrcl c rcl l}
x_1 & + & 0.01x_2 & = & 2.01 & \iff & x_1 & = & 2.01 & -0.01x_2 \\
-0.02x_1 & + & 4x_2 & = & 3.96 & \iff & 4x_2 & = & 3.96 & \underbrace{+0.02x_1}_{\text{“Störung”}}
\end{array}$$

"Sukzessives Einsetzen", beginnend mit $x^{(0)} = \binom{0}{0}$ liefert der Reihe nach die Vektoren

$x^{(0)}$	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
0	2.01	2.0001	1.9999995
0	0.99	1.00005	1.0000005

Dabei wurde die Diagonale der Matrix $\begin{pmatrix} 1 & 0.01 \\ -0.02 & 4 \end{pmatrix}$ als *Hauptteil* M benutzt, der Rest als "Störung". Als bessere (?!) Variante kann man zur Berechnung von $x_2^{(k)}$ statt $x_1^{(k-1)}$ den schon bekannten Wert $x_1^{(k)}$ benutzen. Zugrunde liegt beiden Verfahren die Zerlegung der Matrix $A = L + D + R$ in Diagonale $D = diag(a_{11}, a_{22}, \ldots, a_{nn})$ $(a_{ii} \neq 0)$ und linkes und rechtes Dreieck,

$$A = \begin{pmatrix} \diagdown & & R \\ & D & \\ L & & \diagdown \end{pmatrix}, \quad \text{d.h., } L = \begin{pmatrix} 0 & \cdots & 0 \\ * & \ddots & \vdots \\ * & * & 0 \end{pmatrix}, \quad R = \begin{pmatrix} 0 & * & * \\ \vdots & \ddots & * \\ 0 & \cdots & 0 \end{pmatrix} \tag{2.5.15}$$

Gesamtschrittverfahren, Jacobi-Iteration: Zerlegung $M := D,\ N := -L - R$, Iterationsmatrix ist $B_G = -D^{-1}(L + R)$.

$$\left\{ \begin{array}{rcll} Dx^{(k+1)} & = & b - (L+R)x^{(k)}, & k = 0, 1, \ldots \\ x_i^{(k+1)} & = & \frac{1}{a_{ii}}[b_i - \sum\limits_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum\limits_{j=i+1}^{n} a_{ij}x_j^{(k)}], & i = 1, \ldots, n. \end{array} \right. \tag{2.5.16}$$

Einzelschrittverfahren, Gauß-Seidel-Iteration: Zerlegung $M := D + L,\ N := -R$, Iterationsmatrix ist $B_E = -(D + L)^{-1}R$.

$$\left\{ \begin{array}{rcll} (D+L)x^{(k+1)} & = & b - Rx^{(k)}, & k = 0, 1, \ldots \\ x_i^{(k+1)} & = & \frac{1}{a_{ii}}[b_i - \sum\limits_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum\limits_{j=i+1}^{n} a_{ij}x_j^{(k)}], & i = 1, \ldots, n. \end{array} \right. \tag{2.5.17}$$

Die Formeln (2.5.16) und (2.5.17) besitzen den gleichen Rechenaufwand pro Iterationsschritt, in (2.5.17) werden in der Zeile i allerdings die jeweils aktuellsten Näherungen $x_j^{(k+1)}$ eingesetzt. Der Rechenaufwand beträgt bei beiden Verfahren maximal $2n^2$ Operationen. Für Matrizen aber, bei denen nur wenige Elemente von Null verschieden sind, gilt genauer

Rechenaufwand pro Schritt (2.5.16), (2.5.17) = 2·(Anzahl der nichttrivialen Matrixelemente)

Diese einfache Abhängigkeit ist ein entscheidender Vorteil von Iterationsverfahren, da auch eine unregelmäßige *dünne Besetzung* von A direkt ausgenutzt werden kann, im Gegensatz zu den Eliminationsverfahren.

Ein einfaches, hinreichendes Kriterium für die Konvergenz beider Iterationen ist die *Diagonaldominanz* von A.

Satz 2.5.5 *Die Matrix A sei streng diagonaldominant, (2.3.10). Dann konvergieren Gesamt- und Einzelschrittverfahren. Genauer gilt*

$$\|B_E\|_\infty = \|(D + L)^{-1}R\|_\infty \leq \|D^{-1}(L + R)\|_\infty = \|B_G\|_\infty < 1. \tag{2.5.18}$$

Somit gelten die Abschätzungen (2.5.8) in der Maximumnorm mit $q = \|B_G\|_\infty$ bzw. $q = \|B_E\|_\infty$.

Beweis Die betrachtete Matrixnorm ist $\|.\|_\infty$.

a) Die strenge Diagonaldominanz ist äquivalent mit $\|B_G\| = \max_i \sum_{j\neq i} |a_{ij}/a_{ii}| < 1$.

b) Es wird die stärkere Aussage

$$\|C\| \leq \|B_G\|, \quad C := (|D| - |L|)^{-1}|R| \tag{2.5.19}$$

bewiesen. Daraus folgt die Behauptung (2.5.18) mit $\|B_E\| \leq \|C\| \leq \|B_G\|$, da wie im (vorgezogenen) Beweis von Satz 2.4.4 gilt

$$|B_E| = |(D+L)^{-1}R| \leq (|D| - |L|)^{-1}|R| = C.$$

Zur Ungleichung (2.5.19): Da $C \geq 0$ gilt, ist $\|C\| = \|z\|_\infty =: q$ wobei $Ce =: z,\ e = (1,\ldots,1)^\mathsf{T}$. Mit

$$(I - |D^{-1}L|)^{-1}|D^{-1}R|e = z \iff z = |D^{-1}L|z + |D^{-1}R|e$$

folgt

$$\begin{aligned} q &= \|z\|_\infty = \max_i \{ \sum_{j<i} |\frac{a_{ij}}{a_{ii}}| z_j + \sum_{j>i} |\frac{a_{ij}}{a_{ii}}| \} \\ &\leq \max_i \{ q \sum_{j<i} |\frac{a_{ij}}{a_{ii}}| + \sum_{j>i} |\frac{a_{ij}}{a_{ii}}| \} = \|D^{-1}(qL + R)\|. \end{aligned} \tag{2.5.20}$$

Nach Division durch q (der Fall $q = 0$ ist trivial) wird daraus die Ungleichung

$$1 \leq \max_i \{ \sum_{j<i} |\frac{a_{ij}}{a_{ii}}| + \frac{1}{q} \sum_{j>i} |\frac{a_{ij}}{a_{ii}}| \} = \|D^{-1}(L + \frac{1}{q}R)\|,$$

die wegen der Diagonaldominanz aber nur für $q < 1$ erfüllt sein kann. Also gilt $\|C\| = q \leq \|D^{-1}(qL+R)\| \leq \|D^{-1}(L+R)\| = \|B_G\|$, d.h. (2.5.20)⇒(2.5.19)⇒(2.5.18). ∎

Der Satz zeigt, daß bezüglich der Zeilensummennorm das Einzelschrittverfahren nicht schlechter konvergiert als das Gesamtschrittverfahren. Für die Konvergenz entscheidend ist aber der Spektralradius. Auf diesen läßt sich die Aussage in einem wichtigen Spezialfall übertragen.

Satz 2.5.6 (Stein-Rosenberg) *Die Matrix $B_G = -D^{-1}(L+R)$ sei nicht-negativ, $B_G \geq 0$. Dann gilt einer der Fälle*

$$\left.\begin{array}{rlcccc} 0 = & \rho(B_E) & = & \rho(B_G) & & \\ 0 < & \rho(B_E) & < & \rho(B_G) & < & 1 \\ & \rho(B_E) & = & \rho(B_G) & = & 1 \\ & \rho(B_E) & > & \rho(B_G) & > & 1 \end{array}\right\} \iff \begin{array}{c} |\rho(B_G) - 1| \leq |\rho(B_E) - 1|, \\ \textit{mit Gleichheit genau für } \rho \in \{0,1\}. \end{array}$$

Im nichttrivialen Konvergenzfall konvergiert also das Einzelschrittverfahren für solche Matrizen schneller. Für die Konvergenz von Gesamt- und Einzelschritt-Verfahren gibt es folgende *hinreichenden* Kriterien:

1. (2.3.10), A ist zeilenweise streng diagonaldominant: $|a_{ii}| - \sum_{j\neq i} |a_{ij}| > 0 \ \forall i$.
2. A ist spaltenweise streng diagonaldominant: $|a_{jj}| - \sum_{i\neq j} |a_{ij}| > 0 \ \forall j$.
3. A ist hermitesch, positiv definit.
4. schwache Zeilen- bzw. Spalten-Diagonaldominanz: A ist unzerlegbar und $|a_{ii}| - \sum_{j\neq i} |a_{ij}| \geq 0$ bzw. $|a_{jj}| - \sum_{i\neq j} |a_{ij}| \geq 0$ mit echter Ungleichung "$>$" an mindestens einer Stelle.

Im letzten Fall mußte ausgeschlossen werden, daß das LGS in kleinere Teilprobleme zerfällt:

Definition 2.5.7 *Eine Matrix $A = (a_{ij})$ heißt* unzerlegbar, *wenn keine Umordnung $(i_1, i_2, \ldots, i_n)$ der Indizes $(1, 2, \ldots, n)$ existiert so, daß die Matrix dann folgende Struktur hat*

$$\left(a_{i_k i_j}\right)_{k,j=1}^{n} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}.$$

Relaxationsverfahren

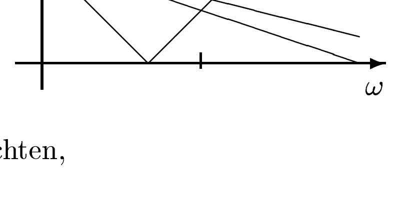
Die gerade beschriebenen Verfahren können oft durch Einführung eines Beschleunigungsparameters ohne höheren Rechenaufwand verbessert werden. Zur Vereinfachung der Darstellung wird schon von einer Matrix der Form $A = I - B$ ausgegangen. Nach (2.5.13) ist dies keine Einschränkung. Mit $0 < \omega \in \mathbf{R}$ wird das Iterationsverfahrens (2.5.2) folgendermaßen modifiziert:

$$x^{(k+1)} := (1-\omega)x^{(k)} + \omega(Bx^{(k)} + r) = x^{(k)} + \omega(r - Ax^{(k)}), \tag{2.5.21}$$

$k = 0, 1, \ldots$. Die Iterationsmatrix ist hier demnach

$$B(\omega) := (1-\omega)I + \omega B = I - \omega A, \tag{2.5.22}$$

wobei $B(1) = B$. Beim Ansatz (2.5.21),(2.5.22) verfolgt man natürlich das Ziel, ω so zu bestimmen, daß der Konvergenzfaktor $\rho(B(\omega))$ möglichst klein wird. Dabei gilt $\rho(B(0)) = 1$.



Eine Optimierung des Parameters ω ist einfach, wenn A und B nur reelle Eigenwerte besitzen. Die Eigenwerte von A seien daher $\lambda_n \leq \ldots \leq \lambda_1$, die von B sind dann $\mu_i = 1 - \lambda_i$. Für $\lambda_n > 0$ existiert offensichtlich ein $\omega > 0$ mit $\rho(B(\omega)) = \max_i |1 - \omega\lambda_i| < 1$.
Dabei muß man nur die flachste und steilste Gerade betrachten,

$$\rho(B(\omega)) = \max\{|1 - \omega\lambda_1|, |1 - \omega\lambda_n|\}. \tag{2.5.23}$$

Ein optimaler Parameter $\hat{\omega}$ ergibt sich im Schnittpunkt dieser beiden (geknickten) Geraden:

$$-(1 - \omega\lambda_1) = 1 - \omega\lambda_n \quad \Longleftrightarrow \quad \frac{1}{\hat{\omega}} = \frac{\lambda_1 + \lambda_n}{2}.$$

Satz 2.5.8 *Die Eigenwerte λ_i von A seien reell und positiv, $0 < \lambda_n \leq \ldots \leq \lambda_2 \leq \lambda_1$. Dann ist der optimale Relaxationsparameter $\hat{\omega}$ bei (2.5.21) gegeben durch*

$$\hat{\omega} = \frac{2}{\lambda_1 + \lambda_n} \quad \textit{mit} \quad \rho\Big(B(\hat{\omega})\Big) = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} < 1.$$

Bemerkung: 1) Die Konvergenz des einfachen Verfahrens ($\omega = 1$) war nicht vorausgesetzt, d.h. $\mu_1 < -1$, $\lambda_1 > 2$, ist zugelassen.

2) Falls der Mittelpunkt des Spektrums von A kleiner Eins ist, $\frac{1}{2}(\lambda_1 + \lambda_n) < 1$, gilt $\hat{\omega} > 1$. Man spricht dann von *Überrelaxation.*

3) Die exakte Kenntnis der Eigenwerte von A ist nicht erforderlich, eine (gute) Einschließung $[\lambda_n, \lambda_1] \subseteq [a, b]$ reicht dazu aus. Die Approximation von Eigenwerten (sogar im Rahmen von (2.5.21)) wird in der Numerik IIA behandelt.

Beispiel 2.5.9 Das folgende LGS hat die Lösung $(0, -1, 1)^\mathsf{T}$, die Matrix die Eigenwerte $1, 1, 4$.

$$\begin{pmatrix} 2 & -1 & 1 \\ 0 & 2 & 2 \\ -1 & 2 & 2 \end{pmatrix} x = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}$$

Der Konvergenzfaktor aus Satz 2.5.8 ist $3/5 = 0.6$ bei $\hat{\omega} := 2/(1+4) = 0.4$. Beginnend mit $x^{(0)} = b$ erhält man durch (2.5.21) die Approximationen:

k=	1	2	3	4	5	6	7
$x_1 =$	1.2	0.72	0.432	0.2592	0.15552	0.09331	0.05599
$x_2 =$	0	-0.64	-0.64	-0.8704	-0.8704	-0.95334	-0.95334
$x_3 =$	0.8	0.64	0.928	0.8704	0.97408	0.95334	0.99067

Das Verfahren (2.5.21) läßt sich sehr einfach analysieren, da die Iterationsmatrix $B(\omega)$ linear von ω abhängt. Es hat aber, etwa in Verbindung mit dem Einzelschrittverfahren mit $B = -(D+L)^{-1}R$ den Nachteil, daß für den Vektor $x^{(k)}$ und $(D+L)^{-1}Rx^{(k)}$ getrennte Speicherplätze erforderlich sind. Geschickter ist es, die Relaxation direkt in jeder Komponente durchzuführen. Für das LGS $Ax = b$ hat ein einzelner solcher Schritt die einfache Gestalt

Algorithmus 2.5.10 *SOR-Verfahren (successive overrelaxation)*

```
für i := 1 bis n:
   s := b_i;
   für j := 1 bis n: s := s − a_ij * x_j;
   x_i := x_i + ω * s/a_ii;
```
(2.5.24)

Mit Iterationsindizes $k = 0, 1, \ldots,$ versehen lautet ein Verfahrensschritt somit

$$a_{ii}x_i^{(k+1)} := a_{ii}x_i^{(k)} + \omega[b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j \geq i} a_{ij}x_j^{(k)}], \quad i = 1, \ldots, n\ .$$

In der Matrix-Schreibweise

$$(D + \omega L)x^{(k+1)} := [(1 - \omega)D - \omega R]x^{(k)} + \omega b \tag{2.5.25}$$

wird allerdings erkennbar, daß die Iterationsmatrix

$$B_S(\omega) := (D + \omega L)^{-1}[(1 - \omega)D - \omega R]$$

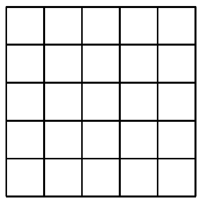
nichtlinear vom Parameter ω abhängt. Eine Analyse des Verfahrens ist für diese Vorlesung zu aufwendig. Als Beispiel einer Konvergenzaussage sei zitiert

Satz 2.5.11 *Für symmetrische und positiv definite Matrizen A gilt*

$$|\omega - 1| \le \rho(B_S(\omega)) < 1 \quad \forall 0 < \omega < 2.$$

Meist wird eine Beschleunigung nur für $\omega > 1$ erreicht, d.h. bei *Überrelaxation*. Eine sehr genaue Analyse ist bei bestimmten, praktisch wichtigen Matrixstrukturen möglich.

Demo-Beispiel: Poisson-Gleichung auf Einheitsquadrat mit einem $10\times$ 10-Gitter, $n = 100$. Das LGS besteht i.w. aus Gleichungen der Form $4x_i - x_{i-1} - x_{i+1} - x_{i-10} - x_{i+10} = 0$, wobei für bestimmte Komponenten, z.B., solche mit Indizes außerhalb $\{1, \ldots, 100\}$ feste Werte eingesetzt werden. Theoretisch berechenbar ist $\rho(B_G) = 0.959$, $\rho(B_E) = \rho(B_G)^2 = 0.9206$, $\rho(B_S(\hat{\omega})) = 0.560 \cong \rho(B_E)^7$.



Also hat eine SOR-Iteration mit optimalem Parameter in diesem Fall den gleichen Effekt wie ca. 7 Einzelschritt- oder 14 Gesamtschritt-Iterationen (fast) ohne Mehraufwand.

Allgemeiner gilt für große $n = m^2$ hier $\varrho(B_E) \cong 1 - \pi^2/m^2$, also $-1/\log_{10}\varrho(B_E) \cong 0.23m^2 = 0.23n$, sowie bei optimalem Parameter $\varrho(B_S) \cong 1 - 2\pi/m$, d.h., $-1/\log_{10}\varrho(B_S) \cong 0.37m = 0.37\sqrt{n}$. Da die Iterationsverfahren ca. $10n$ Operationen pro Schritt benötigen, führt die Annahme, daß der Anfangsfehler um 10^{-8} verkleinert werden soll, auf folgende Aufwandsschätzungen, vgl. (2.5.14). Zum Vergleich ist der Aufwand beim Gauß-Algorithmus für Bandmatrizen angegeben.

Verfahren	ESV	SOR	Gauß
Aufwand ca.	$20n^2$	$30n\sqrt{n}$	$2n^2$

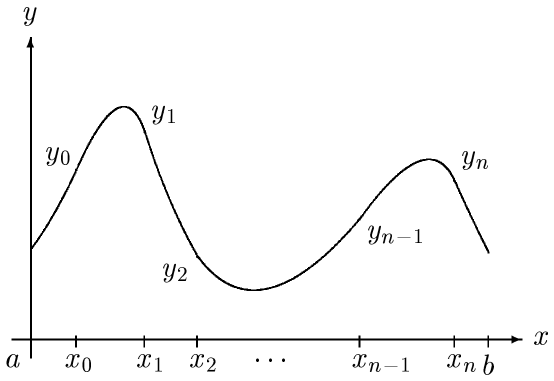
Für große n ist das SOR-Verfahren hier also der Band-Variante des Gauß-Algorithmus überlegen.

3 Polynom-Interpolation

3.1 Problemstellung

In vielen Situationen sind Datenwerte für bestimmte Größen eines Parameters (etwa Zeitpunkte) bekannt, z.B., die Position eines Fahrzeuges oder Flugkörpers oder auch Finanzdaten. Nicht in allen Fällen ist es sinnvoll, diese Daten als Funktionswerte einer (glatten) Funktion zu interpretieren (Börsenwerte?).

Zur Berechnung von Zwischenwerten verwendet man einfache Ersatzfunktionen, die die vorhandenen Daten einbeziehen. Zunächst werden als Ersatzfunktionen Polynome betrachtet, die in bestimmten Parameterwerten ("Stützstellen") vorgegebene Datenwerte annehmen, d.h., *interpolieren*. Im nächsten Paragraphen werden auch Splinefunktionen behandelt, die sich stückweise aus Polynomen zusammensetzen.



3.2 Lagrange-Interpolation für Polynome

Definition 3.2.1 *Zum Gitter* $\{x_0, \ldots, x_n\}$ *mit*

$$a \leq x_0, \ldots, x_n \leq b, \quad x_i \neq x_j \text{ für } i \neq j, \tag{3.2.1}$$

und dem Datenvektor $y := (y_0, \ldots, y_n)^\top$ *sei* p *ein Polynom mit*

$$p(x_i) = y_i, \quad i = 0, 1, \ldots, n. \tag{3.2.2}$$

Dann heißt p Interpolationspolynom. *Ein Polynom, das (3.2.2) erfüllt, nennt man genauer auch* Lagrange-Interpolationspolynom. *Als* Hermite-Interpolationspolynom *bezeichnet man ein Polynom* p *zu den allgemeineren Bedingungen*

$$p^{(j)}(x_i) = y_{ij}, \quad j = 0, \ldots, m_i - 1, \; i = 0, \ldots, n. \tag{3.2.3}$$

Die Bedingungen (3.2.2) führen im Prinzip auf ein lineares Gleichungssystem für die Koeffizienten a_j eines Polynoms vom Grad n in der Darstellung

$$p_n(x) = \sum_{j=0}^{n} a_j x^j, \quad \Pi_n := \{p_n : \; a_j \in \mathbf{R}, j = 0, \ldots, n\}. \tag{3.2.4}$$

Das Lagrange-Interpolationspolynom kann aber direkt konstruiert werden:

Satz 3.2.2 *Zu beliebigen Wertepaaren (x_i, y_i), $i = 0, \dots, n$, mit paarweise verschiedenen Stützstellen $x_i \neq x_j$ (für $i \neq j$) existiert genau ein Interpolationspolynom vom Grad n zu (3.2.2). Es hat die Form*

$$p := p_y := \sum_{i=0}^{n} y_i L_i, \quad L_i(x) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{3.2.5}$$

Die Abbildung $\mathbf{R}^{n+1} \to \Pi_n$, $y \mapsto p_y$ ist linear.

Offensichtlich sind die L_i Polynome einer Kardinal-Basis,

$$L_i \in \Pi_n, \quad L_i(x_k) = \delta_{ik} = \begin{cases} 1 & \text{für} \quad i = k, \\ 0 & \text{für} \quad i \neq k \end{cases},$$

in den Stützstellen x_j verschwindet daher genau ein Summand von p_y nicht.

Eines von vielen Kriterien zur Beurteilung numerischer Algorithmen ist der Rechenaufwand. Die Operationsanzahlen wachsen mit dem Polynomgrad n. Wenn die Differenzen $x - x_j$ getrennt berechnet werden, sind zur Auswertung von (3.2.5) jeweils $n^2 + 2n$ Aditionen, $n(n+1)$ Multiplikationen und $n(n+1)$ Divisionen nötig. Für große n ist der am stärksten wachsende Anteil (hier die n^2-Terme) entscheidend, die anderen werden als $\mathcal{O}(n)$-Ausdruck zusammengefaßt. Dies ergibt den

Rechenaufwand für eine Auswertung von (3.2.5): $3n^2 + \mathcal{O}(n)$ Operationen.

Beweis zu Satz 3.2.2 Wegen der genannten Eigenschaften der L_i gilt für p aus (3.2.5)

$$p \in \Pi_n, \quad p(x_k) = \sum_{i=0}^{n} y_i L_i(x_k) = y_k, \quad k = 0, \dots, n.$$

Zur Eindeutigkeit: wenn zwei Polynome $p, \bar{p} \in \Pi_n$ beide (3.2.2) erfüllen, so folgt

$$(p - \bar{p})(x_i) = 0, \quad i = 0, \dots, n.$$

Damit sind nach dem Fundamentalsatz der Algebra p und $\bar{p}$ identisch. ■

Im Vergleich zu anderen Darstellungen hat (3.2.5) einen sehr hohen Rechenaufwand pro Auswertung. Für die praktische Anwendung ist eine andere Darstellung der L_i aus (3.2.5)besser geeignet. Mit

$$b_i^{(n)} := b_i := \frac{1}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}, \quad \omega(x) := \prod_{j=0}^{n} (x - x_j) \tag{3.2.6}$$

ist $L_i(x) = b_i \omega(x)/(x - x_i)$ für $x \neq x_i$. Damit erhält man für das Polynom p aus (3.2.5) die Form

$$p_n(x) = \omega(x) \sum_{i=0}^{n} \frac{b_i}{x - x_i} y_i \quad \text{für } x \neq x_i.$$

Speziell ergibt sich mit dem Sonderfall

$$p_n(x) \equiv 1 \iff y_i \equiv 1 \ \forall i$$

aus der letzten Gleichung die Relation

$$1 \equiv \omega(x) \sum_{i=0}^{n} \frac{b_i}{x - x_i}, \quad \text{also} \quad \omega(x) = 1 \Big/ \sum_{i=0}^{n} \frac{b_i}{x - x_i} \qquad \text{für } x \neq x_j \ \forall j. \tag{3.2.7}$$

Dies zeigt folgende Darstellung von $p(x)$ als ein gewichtetes Mittel.

Satz 3.2.3 *Das Lagrange-Interpolationspolynom ist darstellbar in der* baryzentrischen Form

$$p_n(x) = \begin{cases} \dfrac{\displaystyle\sum_{i=0}^{n} \frac{b_i}{x - x_i} y_i}{\displaystyle\sum_{i=0}^{n} \frac{b_i}{x - x_i}} & \textit{für} \quad x \neq x_i \, \forall i, \\[2ex] y_j & \textit{wenn} \quad x = x_j. \end{cases} \tag{3.2.8}$$

Zur Auswertung von (3.2.8) berechnet man zweckmäßigerweise zunächst die vom Punkt x und den Datenwerten y_i unabhängigen Gewichte b_i. Man kann zeigen, daß die Auswertung von (3.2.8) "numerisch stabiler" ist ($\rightarrow$ §7) als (3.2.5).

Bei Hinzunahme einer Stützstelle ergibt sich der Übergang von den $b_i^{(n)}$ in (3.2.6) zu den ersten $n+1$ Koeffizienten $b_i^{(n+1)}$, $i = 0, \ldots, n$, folgendermaßen:

$$b_i^{(n+1)} := b_i^{(n)} / (x_i - x_{n+1}), \quad i = 0, 1, \ldots, n. \tag{3.2.9}$$

Für den offenen Fall $n+1$ wird (3.2.7) umgeformt zu

$$1 \equiv \sum_{i=0}^{n+1} b_i^{(n+1)} \prod_{\substack{j=0 \\ j \neq i}}^{n+1} (x - x_j) = x^{n+1} \sum_{i=0}^{n+1} b_i^{(n+1)} + \ldots \tag{3.2.10}$$

Durch Koeffizientenvergleich folgt also

$$\sum_{i=0}^{n+1} b_i^{(n+1)} = 0 \qquad \text{für } n \geq 0. \tag{3.2.11}$$

Daraus kann auch noch $b_{n+1}^{(n+1)}$ bestimmt werden und es ergibt sich

Algorithmus 3.2.4 Rekursive Berechnung der Gewichtsfaktoren $b_i^{(n)}$ in (3.2.6):

$b_0^{(0)} := 1;$

für $k := 1$ bis n:

  für $i := 0$ bis $k - 1$:

    $b_i^{(k)} := b_i^{(k-1)} / (x_i - x_k);$

  $b_k^{(k)} := -\sum_{i=0}^{k-1} b_i^{(k)};$

(3.2.12)

Dieser Algorithmus (3.2.12) erfordert $\sum_{k=1}^{n} k = \frac{n^2}{2} + \mathcal{O}(n)$ Divisionen und doppelt soviel Additionen, also ist der

Rechenaufwand Baryzentrische Darstellung:

$$\begin{array}{rl} \frac{3}{2}n^2 + \mathcal{O}(n) & \text{Operationen zur Berechnung der } \{b_i^{(n)}\} \text{ in (3.2.12),} \\ 5n + \mathcal{O}(1) & \text{Operationen zur Auswertung von } p(x) \text{ in (3.2.8).} \end{array}$$

Man beachte, daß im Dreieck-Schema (3.2.12) die $b_i^{(k)}$ die $b_i^{(k-1)}$ überschreiben dürfen, daher wird also nur ein lineares Feld $[0, \ldots, n]$ benötigt.

Beispiel 3.2.5 Interpolation auf einem *äquidistanten Gitter*: Mit

$$x_i := x_0 + ih, \; i = 0, \ldots, n$$

erhält man

$$\frac{1}{b_i} = \prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j) = h^n \prod_{\substack{j=0 \\ j \neq i}}^{n} (i - j) = h^n (-1)^{n-i} i!(n-i)! \;. \tag{3.2.13}$$

Dies ergibt

$$b_i = \frac{(-1)^n}{h^n n!} \cdot \frac{n!(-1)^i}{i!(n-i)!} = \frac{(-1)^n}{h^n n!} \cdot (-1)^i \binom{n}{i}.$$

Kürzt man in der Darstellung (3.2.8) hierbei den von i unabhängigen Vorfaktor heraus und führt die Größen $\tilde{b}_i := (-1)^i \binom{n}{i}$ ein, so ergibt sich für den Fall der äquidistanten Knoten die folgende vereinfachte Formel für das Interpolationspolynom p,

$$p_n(x) = \sum_{i=0}^{n} \frac{(-1)^i}{x - x_i} \binom{n}{i} y_i \Big/ \sum_{i=0}^{n} \frac{(-1)^i}{x - x_i} \binom{n}{i} \qquad \text{für } x \neq x_j \; \forall j. \tag{3.2.14}$$

3.3 Newton-Interpolation

Um beim konkreten Einsatz eventuell zusätzliche Stützstellen schnell berücksichtigen zu können, ist folgender Ansatz zweckmäßig

$$p_n(x) = p_{n-1}(x) + (x - x_0) \cdots (x - x_{n-1}) a_n.$$

Dies ist die Newton-Darstellung des Interpolationspolynoms. Darin verschwindet der zweite Summand in den Punkten $x_0, \ldots, x_{n-1}$. Man setzt also induktiv voraus, daß p_{n-1} die Interpolationsaufgabe in den ersten n Punkten erfüllt, $p_{n-1}(x_i) = y_i, \; i = 0, \ldots, n-1$. Den Wert a_n berechnet man dann aus der zusätzlichen Interpolationsbedingung $p_n(x_n) = y_n$, zweckmäßiger ist aber ein weiter unten formulierter Algorithmus. Beginnend mit $p_0(x) \equiv y_0 = a_0$ setzt man also für p_n die folgende Form an:

$$\begin{aligned} p_n(x) &= a_0 + (x - x_0) a_1 + \ldots + (x - x_0) \cdots (x - x_{n-1}) a_n \qquad (3.3.1) \\ &= a_0 + (x - x_0)[\cdots [a_{n-2} + (x - x_{n-2})[a_{n-1} + (x - x_{n-1}) a_n]] \cdots] \end{aligned}$$

Diese Klammerung deutet bereits an, wie $p_n(x)$ am günstigsten zu berechnen ist:

Algorithmus 3.3.1 Horner-Schema zur Auswertung von $p_n(x)$ bei bekannten a_i.

$q := a_n;$
für $k = n-1, n-2, \ldots, 0$:
$\quad q := a_k + (x - x_k)q;$
$p_n(x) = q.$

Rechenaufwand Hornerschema: $3n$ Operationen:

Für den Spezialfall $x_i = 0\ \forall i$ erhält man $p_n(x) = \sum_{i=0}^{n} a_i x^i$ und braucht für die Auswertung dieser Polynomform nach dem Hornerschema nur $2n$ Operationen.

Das Hauptproblem bei (3.3.1), die Bestimmung der Koeffizienten a_i, wird nun behandelt.

Definition 3.3.2 (und Algorithmus) *Die ℓ-ten* dividierten Differenzen *oder ℓ-ten Differenzenquotienten der Daten y bezgl. der Stützstellen x werden bestimmt durch den Algorithmus*

$$\boxed{\begin{array}{l} \textit{für } i = 0, 1, \ldots, n\textit{:} \quad y[x_i] := y_i; \\ \textit{für } l = 1, \ldots, n\textit{:} \\ \quad \textit{für } i = 0, \ldots, n-l\textit{:} \\ \qquad y[x_i, \ldots, x_{i+l}] := \dfrac{y[x_{i+1}, \ldots, x_{i+l}] - y[x_i, \ldots, x_{i+l-1}]}{x_{i+l} - x_i}. \end{array}} \tag{3.3.2}$$

Bemerkungen: Die Berechnung nach (3.3.2) führt auf folgendes Dreieckschema, das spaltenweise $(l = 1, 2, \ldots, n)$ erzeugt wird.

$$\begin{array}{c|cccccc} x_i & y_i = y[x_i] & l=1 & l=2 & & l=n \\ \hline x_0 & y_0 & & & & \\ x_1 & y_1 & y[x_0,x_1] & & & \\ x_2 & y_2 & y[x_1,x_2] & y[x_0,x_1,x_2] & & \\ \vdots & \vdots & \vdots & & \ddots & \\ x_n & y_n & y[x_{n-1},x_n] & y[x_{n-2},x_{n-1},x_n] & \cdots & y[x_0,\ldots,x_n] \end{array} \tag{3.3.3}$$

Im Newtonpolynom (3.3.1) werden nur die Differenzen $y[x_0, \ldots, x_i]$ benötigt, s.u., (3.3.7). Daher genügt zur Durchführung ein lineares Feld $[0, 1, \ldots, n]$, in dem der Reihe nach zunächst die Werte $y_1, \ldots, y_n$ durch die ersten dividierten Differenzen, dann die ersten dividierten Differenzen $y[x_1, x_2]$,..., $y[x_{n-1}, x_{n-2}]$ durch die zweiten überschrieben werden, usw. Am Ende bleiben die in (3.3.7) benötigten Werte übrig. Es ist auch eine zeilenweise Berechnung möglich, wenn man mit dem höchsten Index beginnt. Dabei kann dann bei Hinzunahme zusätzlicher Interpolationspunkte das Differenzenschema einfach ergänzt werden.

Rechenaufwand für (3.3.2): $\quad 3[n + (n-1) + \cdots + 1] = \frac{3}{2}n^2 + \mathcal{O}(n)$ Operationen.

Der Zusammenhang der in (3.3.2) eingeführten Differenzenquotienten mit den Koeffizienten a_i aus (3.3.1) ergibt sich aus dem folgenden Neville-Algorithmus und dem anschließenden Satz 3.3.4.

Definition 3.3.3 (und Neville-Algorithmus) *Es seien $p_{i,\ell} \in \Pi_\ell$ Abschnittspolynome mit*

$$p_{i,\ell}(x_j) = y_j, \quad j = i, \ldots, i+\ell. \tag{3.3.4}$$

Man erhält die Werte dieser $p_{i,\ell}$ und den des Polynoms p_n zu (3.2.2) an einer Stelle x durch

$$\boxed{\begin{array}{l} \text{für } i = 0,1,\ldots,n\text{:} \quad p_{i,0}(x) := y_i; \\ \text{für } \ell = 1,\ldots,n\text{:} \\ \quad \text{für } i = 0,\ldots,n-\ell\text{:} \\ \qquad p_{i,\ell}(x) := \dfrac{(x-x_i)p_{i+1,\ell-1}(x) + (x_{i+\ell}-x)p_{i,\ell-1}(x)}{x_{i+\ell}-x_i}; \\ p_n(x) = p_{0,n}(x). \end{array}} \tag{3.3.5}$$

Für $x \in (x_i, x_{i+\ell})$ ist $p_{i,\ell}(x)$ eine "konvexe Linearkombination" der $p_{i,\ell-1}(x)$ und $p_{i+1,\ell-1}(x)$. Dadurch werden Rundungsfehler im Algorithmus nicht verstärkt (vgl. §7).

<u>Rechenaufwand</u> für den Neville-Algorithmus (3.3.5) bei vorberechneten Differenzen $x - x_i$:

$$\frac{5}{2}n^2 + \mathcal{O}(n) \text{ Operationen.}$$

Den Zusammenhang zwischen den dividierten Differenzen aus (3.3.2), den Teilpolynomen $p_{i,\ell}$ aus (3.3.4) und den Koeffizienten a_i aus der Newton-Interpolationsform (3.3.1) gibt der folgende

Satz 3.3.4 *a) Die in (3.3.5) berechneten Polynome und die in (3.3.4) definierten sind gleich.*

b) Die dividierten Differenzen in (3.3.2) stimmen mit den höchsten Koeffizienten der Teilpolynome in (3.3.4) überein:

$$y[x_i, \ldots, x_{i+\ell}] = \frac{1}{\ell!}\, p_{i,\ell}^{(\ell)}, \quad 0 \le i \le i+\ell \le n. \tag{3.3.6}$$

c) Das Lagrange-Interpolationspolynom (3.2.2) ist in der Newtonform (3.3.1) darstellbar als

$$p_n(x) = y[x_0] + (x-x_0)\, y[x_0,x_1] + \ldots + (x-x_0)\cdots(x-x_{n-1})\, y[x_0,\ldots,x_n]. \tag{3.3.7}$$

Beweis Durch Induktion, die Aussagen bezüglich (3.3.4) bis (3.3.7) sind offensichtlich richtig für $n = \ell = 0$.

a) Unter der Induktionsvoraussetzung, daß $p_{i,\ell-1}(x)$ und $p_{i+1,\ell-1}(x)$ ihre Interpolationsbedingungen in (3.3.4) erfüllen, gilt für das in (3.3.5) definierte $p_{i,\ell}(x)$ die Aussage

$$p_{i,\ell}(x_j) = y_j, \quad j = i, \ldots, i+\ell.$$

Für $j = i$ und $j = i + \ell$ treten nämlich in (3.3.5) nur der zweite bzw. erste Summand im Zähler auf, für die Fälle $j = i + 1, \ldots, i + \ell - 1$ kann man aufgrund der Induktionsvoraussetzung den Faktor y_j ausklammern, die Vorfaktoren ergeben zusammen eins.

b) Nun wird angenommen, (3.3.6) sei richtig für $\ell - 1$. Dann gilt nach (3.3.1) und (3.3.4)

$$p_{i,\ell}(x) = p_{i,\ell-1}(x) + (x - x_i) \cdots (x - x_{i+\ell-1}) \cdot a. \tag{3.3.8}$$

Zunächst wird gezeigt

$$a = y[x_i, \ldots, x_{i+\ell}]. \tag{3.3.9}$$

Durch Vergleich der Koeffizienten von x^ℓ in (3.3.8) und (3.3.5) ergibt sich mit der Induktionsvoraussetzung die Gleichung

$$a = \frac{y[x_{i+1}, \ldots, x_{i+\ell}] - y[x_i, \ldots, x_{i+\ell-1}]}{x_{i+\ell} - x_i}.$$

Dies ist die Rekursion (3.3.2) für (3.3.9). Durch Differentiation von (3.3.8) folgt (3.3.6).

c) Ist der Spezialfall von (3.3.8) und (3.3.9) für $i = 0$ und $\ell = 0, \ldots, n$. ∎

Beispiel 3.3.5 Newton-Polynom durch vier Interpolationspunkte (Werte in der Tabelle):

i	x_i	y_i			
0	-1	$\boxed{2}$			
1	0	4	$\boxed{2}$		
2	2	6	1	$\boxed{-\frac{1}{3}}$	
3	3	12	6	$\frac{5}{3}$	$\boxed{\frac{1}{2}}$

Die eingerahmten Werte sind die Koeffizienten im Interpolationspolynom

$$p_3(x) = 2 + 2(x+1) - \frac{1}{3}(x+1)x + \frac{1}{2}(x+1)x(x-2).$$

Bei den besprochenen Verfahren werden keinerlei spezielle Eigenschaften der verwendeten Datenpaare (x_j, y_j) berücksichtigt. Wenn die y_j allerdings *Funktionswerte* einer bestimmten Funktion f sind, stellt sich die Frage, wie stark p und f voneinander abweichen. Diese und andere Eigenschaften sind im folgenden Satz zusammengefaßt.

Satz 3.3.6 *a) Das Interpolationspolynom p_n aus (3.2.5) hängt linear vom Datenvektor y ab.*

b) Die dividierte Differenz $y[x_i, \ldots, x_{i+k}]$ ist unabhängig von der Reihenfolge der Wertepaare $(x_i, y_i), \ldots, (x_{i+k}, y_{i+k})$.

c) Sind die y_i Funktionswerte einer Funktion $f \in C^n[a,b]$, d.h., $y_i = f(x_i)$ mit $x_i \in [a,b]$, so gilt

$$f[x_i, \ldots, x_{i+k}] := y[x_i, \ldots, x_{i+k}] = \frac{1}{k!}\, f^{(k)}(\xi), \tag{3.3.10}$$

wobei $\xi \in (\min_{j=i}^{i+k} x_j, \max_{j=i}^{i+k} x_j)$, $0 \le i \le i + k \le n$.

d) Für $f(x) = c_k x^k + \ldots \in \Pi_k$ gilt

$$f[x_0, \ldots, x_n] = \begin{cases} c_k & \text{für} \quad k = n, \\ 0 & \text{für} \quad k < n. \end{cases}$$

e) Für den Interpolationsfehler $r_n(x) := f(x) - p_n(x)$, $x \in [a, b]$, gilt bei einer Funktion $f \in C^{n+1}[a, b]$ mit $\omega_{n+1}(x) := (x - x_0) \ldots (x - x_n)$ und einem $\xi \in (a, b)$ die Darstellung

$$r_n(x) \;=\; \omega_{n+1}(x) f[x_0, \ldots, x_n, x] \;=\; \omega_{n+1}(x) \frac{f^{(n+1)}(\xi)}{(n+1)!}\,. \tag{3.3.11}$$

Beweis Teil a) ist nach Definition offensichtlich, b) folgt aus Satz 3.2.2 und (3.3.6).

c) Nach (3.3.6) ist $y[x_i, \cdots, x_{i+k}] = p_{i,k}^{(k)}(x)/k!$. Nun verschwindet $r_k = f - p_{i,k}$ in den Punkten $x = x_i, \cdots, x_{i+k}$. Nach dem iterierten Satz von Rolle existiert $\xi \in \left(\min_{j=i}^{i+k} x_j, \max_{j=i}^{i+k} x_j\right)$ mit

$$0 = r_k^{(k)}(\xi) = f^{(k)}(\xi) - p_{i,k}^{(k)}(\xi).$$

d) Dies ist ein Spezialfall von c).

e) Der Interpolationsfehler wird an einer festen Stelle $x = \bar{x}$ dargestellt und das Interpolationspolynom $p_{0,n+1}$ mit der zusätzlichen Stützstelle $x_{n+1} := \bar{x}, y_{n+1} := f(\bar{x})$, betrachtet. Nach Satz 3.3.4 gilt hierfür

$$p_{0,n+1}(x) = p_{0,n}(x) + (x - x_0) \cdots (x - x_n) f[x_0, \cdots, x_n, x_{n+1} = \bar{x}].$$

Wegen $p_{0,n+1}(\bar{x}) = f(\bar{x})$ folgt in $\bar{x}$ für den Fehler

$$r_n(\bar{x}) = f(\bar{x}) - p_{0,n}(\bar{x}) = (\bar{x} - x_0) \cdots (\bar{x} - x_n)\, f[x_0, \cdots, x_n, \bar{x}].$$

Die Definition von ω_{n+1} und (3.3.10) liefern schließlich (3.3.11). ■

Wegen der Beziehung (3.3.10) liegt es nahe, das Hermite-Interpolationsproblem (3.2.3) durch Grenzübergang von mehreren Knoten in eine gemeinsame Stelle zu behandeln. Bekanntlich ist

$$\lim_{x_{i+1} \to x_i} f[x_i, x_{i+1}] = f'(x_i) \qquad \text{für } f \in C^1[a, b].$$

Analoge Aussagen gelten in (3.3.10), wenn mehr als zwei Knoten zusammenrücken. Das Interpolationsproblem ist allerdings nur dann sinnvoll gestellt, wenn, wie in (3.2.3) gefordert, an einer Stelle Funktions- und Ableitungswerte lückenlos vorgeschrieben sind. Es seien also für die Hermite-Interpolationsaufgabe bei einer genügend oft differenzierbaren Funktion f die folgenden Werte bekannt

$$f(x_i), f'(x_i), \cdots, f^{(j)}(x_i), \qquad \text{wenn } x_i = x_{i+1} = \cdots = x_{i+j}. \tag{3.3.12}$$

Mit diesen Ableitungen werden im Einklang mit (3.3.10) die Differenzenquotienten definiert durch

$$f[x_i, \cdots, x_{i+k}] := \frac{1}{k!} f^{(k)}(x_i), \ 0 \le k \le j.$$

Für all solche *Mehrfachenknoten* (3.3.12) werden diese Werte in das Differenzentableau (3.3.3) eingetragen. Die noch fehlenden Differenzenquotienten berechnet man dann nach (3.3.2).

Beispiel 3.3.7 Differenzentableau für $n = 4,\ x_1 = x_2 = x_3$:

$$
\begin{array}{c|lllll}
x_0 & y_0 & & & & \\
x_1 & y_1 & y[x_0,x_1] & & & \\
x_1 & y_1 & y_1' & y[x_0,x_1,x_1] & & \\
x_1 & y_1 & y_1' & \frac{1}{2}y_1'' & y[x_0,x_1,x_1,x_1] & \\
x_4 & y_4 & y[x_1,x_4] & y[x_1,x_1,x_4] & y[x_1,x_1,x_1,x_4] & y[x_0,x_1,x_1,x_1,x_4]
\end{array}
$$

Bemerkungen: Man kann das Hornerschema aus Algorithmus 1 verallgemeinern zur Berechnung von Ableitungswerten von Polynomen.

Satz 3.3.6 erlaubt eine Gegenüberstellung von Taylor- und Newton-Entwicklung für $f \in C^{n+1}[a,b]$. Mit den analog zu ω_{n+1} definierten $\omega_k(x) = (x-x_0)\cdots(x-x_{k-1})$, vgl. (3.3.11), erhält man für die Entwicklung nach

$$
\begin{array}{lll}
\text{Taylor:} & f(x) = & \displaystyle\sum_{k=0}^{n}(x-x_0)^k\,\frac{f^{(k)}(x_0)}{k!} + (x-x_0)^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!}, \\
\text{Newton:} & f(x) = & \displaystyle\sum_{k=0}^{n}\omega_k(x) f[x_0,\cdots,x_k] + \ \omega_{n+1}(x)\ \frac{f^{(n+1)}(\eta)}{(n+1)!},
\end{array}
$$

$\xi, \eta \in (a,b)$. Insbesondere geht für $x_j \to x_0,\ j = 1,\cdots,n$, die Newton-Darstellung über in die Taylor-Darstellung.

3.4 Entwicklung nach Orthogonalpolynomen, Drei-Term-Rekursionen

Die Vorteile von Orthogonalbasen sind aus der Linearen Algebra bekannt. Um diese auch bei der Interpolation nutzen zu können werden in $\Pi_n \subseteq C[a,b]$ Familien von Orthogonalpolynomen bezüglich geeigneter Skalarprodukte eingeführt. Diese lassen sich *rekursiv* durch *Drei-Term-Rekursionen* berechnen. Der wichtigste Fall ist die Entwicklung nach *Tschebyscheff-Polynomen.*

Zu einer positiven Gewichtsfunktion $g : [a,b] \mapsto \mathbf{R}_+$ und $f, h \in C[a,b]$ sei

$$
\begin{array}{lllll}
(f,h) & := & (f,h)_g & := & \int_a^b f(x)h(x)g(x)\ dx, \quad \text{bzw.} \\
(f,h) & := & (f,h)_g & := & \sum_{k=0}^{n} f(x_k)h(x_k)g(x_k)
\end{array}
\tag{3.4.1}
$$

mit $a \le x_0 < \ldots < x_n \le b$. Durch (3.4.1) sind Innenprodukte für Π_n definiert. Über Zusammenhänge zwischen den beiden Innenprodukten aus (3.4.1) in Π_n wird in §5 nachgedacht. Eine Familie von Orthogonalpolynomen ist definiert durch die Relationen

$$
P_k \in \Pi_k \quad \text{und} \quad (P_j, P_k) = \delta_{jk}\cdot c_k,\ 0 \le j,k \le n. \tag{3.4.2}
$$

Dabei bezeichnet δ_{jk} das übliche Kronecker-Symbol, die Werte c_k sind entweder in geeigneter Weise vorgegeben (z.B. $c_k \equiv 1$) oder berechnen sich aus der Forderung

$$P_0(x) \equiv 1,\; P_k(x) = x^k + \ldots \tag{3.4.3}$$

Durch (3.4.1) bis (3.4.3) ist die Familie von Orthogonalpolynomen wohldefiniert und kann durch das Schmidtsche Orthogonalisierungsverfahren bestimmt werden.

Orthogonalsysteme bieten den Vorteil, daß die Entwicklungskoeffizienten zur Best-Approximation einer Funktion f explizit bekannt sind, mit

$$\alpha_j := (f, P_j)/(P_j, P_j),\; 0 \le j \le n,\; \text{ist} \quad p_k := \sum_{j=0}^{k} \alpha_j P_j \approx f \tag{3.4.4}$$

Bestapproximierende vom Grad $0 \le k \le n$. Hier werden nur die Tschebyscheff-Polynome behandelt, im §5 treten aber bei der numerischen Integration weitere Klassen von Orthogonalpolynomen auf.

Definition 3.4.1 *Man bezeichnet die Funktionen*

$$T_n(x) := \cos(n \arccos x),\; |x| \le 1, \quad n = 0, 1, .., \tag{3.4.5}$$

als Tschebyscheff-Polynome *und ihre Nullstellen*

$$x_k := x_{k,n} := \cos\Big(\frac{2k+1}{2n+2}\pi\Big), \quad k = 0, \ldots, n \tag{3.4.6}$$

als Tschebyscheff-Punkte oder -Knoten.

Satz 3.4.2 *Die in (3.4.5) definierten Funktionen T_n sind Polynome vom Grad n. Sie können mit $T_0 = 1,\; T_1(x) = x$, rekursiv berechnet werden durch die Drei-Term-Rekursion*

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),\; n = 1, 2, \ldots. \tag{3.4.7}$$

Die $x_k = x_{k,n}$ aus (3.4.6) sind die Nullstellen von T_{n+1}. Die Polynome T_n haben die Form

$$T_n(x) = 2^{n-1}x^n + \ldots \quad \in \Pi_n, \tag{3.4.8}$$

ihre Supremum-Norm ist eins, für $n = 0, 1, \ldots$ gilt

$$\|T_n\|_{[-1,1],\infty} = \max_{x \in [-1,1]} |T_n(x)| = 1. \tag{3.4.9}$$

Beweis Mit $t := \arccos x$ folgen die Aussagen für T_0, T_1 unmittelbar aus (3.4.5). Die Rekursion in (3.4.7) ergibt sich aus dem Additionstheorem

$$\cos(n+1)t + \cos(n-1)t = 2\cos t \cdot \cos nt.$$

An (3.4.7) erkennt man, daß die $T_n \in \Pi_n$ die Form (3.4.8) besitzen und an (3.4.5), daß die x_k aus (3.4.6) die Nullstellen von T_{n+1} sind und (3.4.9) gilt. ∎

Satz 3.4.3 *Die Tschebyscheff-Polynome T_k, $k = 0, 1, \dots, n$, bilden ein Orthogonalsystem bezüglich des Innenprodukts ($f, h \in \Pi_n$)*

$$(f, h)_n := \sum_{k=0}^{n} f(x_k) h(x_k) \qquad \textit{mit } x_k = x_{k,n} \textit{ aus (3.4.6).} \tag{3.4.10}$$

Zur Vereinfachung der Formeln (wegen (3.4.13)) verwendet man bei Entwicklungen nach Tschebyscheff-Polynomen (wie bei Fourierentwicklungen) statt (3.4.4) die Darstellung

$$f \approx \frac{c_0}{2} + \sum_{j=1}^{n} c_j T_j =: p_n. \tag{3.4.11}$$

Die Koeffizienten ergeben sich bei den speziellen Knoten (3.4.6) in (3.4.10) durch

$$c_j = \frac{2}{n+1}(f, T_j)_n = \frac{2}{n+1} \sum_{k=0}^{n} f(x_k) \cos\left(j \frac{2k+1}{2n+2}\pi\right), \quad j = 0, 1, \dots, n. \tag{3.4.12}$$

Das Polynom p_n aus (3.4.11) interpoliert f in $x_{k,n}, k = 0, \dots, n$.

Beweis Die Aussagen in (3.4.10), (3.4.12) folgen aus der bekannten Beziehung

$$\begin{aligned}(T_i, T_j)_n &= \sum_{k=0}^{n} T_i(x_k) T_j(x_k) = \sum_{k=0}^{n} \cos\left(i \frac{2k+1}{2n+2}\pi\right) \cos\left(j \frac{2k+1}{2n+2}\pi\right) \\ &= \begin{cases} 0 & \text{für} \quad i \neq j, \\ \frac{n+1}{2} & \text{für} \quad 0 < i = j < n+1, \\ n+1 & \text{für} \quad i = j = 0. \end{cases}\end{aligned} \tag{3.4.13}$$

Die Relation $(T_0, T_0)_n = 2(T_j, T_j)_n, 1 \leq j \leq n$, in Formel (3.4.13) ist der Grund für den Übergang von (3.4.5) zu (3.4.11). Da p_n nach einer Orthogonalbasis entwickelt wurde, ist

$$(f - p_n, f - p_n)_n = \min_{q \in \Pi_n} (f - q, f - q)_n = \min_{q \in \Pi_n} \sum_{k=0}^{n} [f(x_k) - q(x_k)]^2 = 0.$$

Das Minimum wird im Interpolationspolynom an die Funktion f zu den Punkten x_k aus (3.4.6) angenommen, mit dem p_n offensichtlich übereinstimmt. ■

Rechenaufwand zur Bestimmung der c_j aus (3.4.12):

$$2n^2 + \mathcal{O}(n) \text{ Operationen } + n^2 \cos\text{-Funktionsauswertungen.}$$

Die Drei-Term-Rekursion (3.4.7) ist auch bei der Auswertung eines Polynoms (3.4.11) einsetzbar. Analoge Algorithmen sind für alle Orthogonalfamilien mit Drei-Term-Rekursionen möglich.

Algorithmus 3.4.4 Clenshaw-Algorithmus

$$\boxed{\begin{array}{l} d_{n+1} := 0;\ d_n := c_n; \\ \text{für } k = n-1, n-2, \dots, 1: \\ \quad d_k := c_k + 2x d_{k+1} - d_{k+2}; \\ p_n(x) = c_0/2 + x d_1 - d_2; \end{array}} \tag{3.4.14}$$

zur Berechnung von

$$p_n(x) = \frac{c_0}{2} + \sum_{j=1}^{n} c_j T_j(x) \qquad \text{für } x \in \mathbf{R}, (\text{ i.a. } |x| \le 1). \tag{3.4.15}$$

Rechenaufwand von (3.4.14): $3n + \mathcal{O}(1)$ Operationen.

Der Clenshaw-Algorithmus ergibt sich einfach durch Vertauschung der Multiplikationen in folgender Darstellung. Der Polynomwert (3.4.15) entspricht einem Innenprodukt

$$p_n(x) - \frac{c_0}{2} = (c_1, \ldots, c_n)\Big(T_1(x), \ldots, T_n(x)\Big)^{\mathsf{T}}.$$

Da der Vektor $t := \Big(T_1(x), \ldots, T_n(x)\Big)^{\mathsf{T}}$ aufgrund der Rekursion (3.4.7) das Dreieck-System

$$\begin{pmatrix} 1 & & & & \\ -2x & 1 & & & \\ 1 & -2x & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2x & 1 \end{pmatrix} \begin{pmatrix} T_1(x) \\ T_2(x) \\ T_3(x) \\ \vdots \\ T_n(x) \end{pmatrix} = \begin{pmatrix} x \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \Longleftrightarrow: \quad Bt = r$$

löst, gilt $t = B^{-1}r$. Der Vektor $d := (B^{\mathsf{T}})^{-1}c$ wird in der Rekursion (3.4.14) berechnet, es folgt

$$p_n(x) - \frac{c_0}{2} = c^{\mathsf{T}}t = c^{\mathsf{T}}B^{-1}r = d^{\mathsf{T}}r = xd_1 - d_2.$$

Bemerkung: Wegen (3.4.5) und der daraus abgeleiteten Normabschätzung (3.4.9) ist der Fehlerverlauf bei einer (abgebrochenen) Tschebyscheff-Entwicklung wesentlich gleichförmiger als etwa beim Newton-Polynom mit Hornerschema und sonstigen Knoten.

3.5 Vergleich der Verfahren

Verfahren	Rechenaufwand(Oper.): Vorbereitung,	1 Auswertung	Bemerkungen
Lagrange origin.	$L_i(x)$: $3n^2$	$2n$	L_i abhängig von x und Gitter, Stabilitätsprobleme
Lagr. baryzentr.	b_i : $\frac{3}{2}n^2$	$5n$	b_i nur gitterabhängig, flexibel, stabil
Newton	$y[.]$: $\frac{3}{2}n^2$	$3n$	$y[x_{0,\ldots}]$ abhängig von Gitter und Daten, flexibel, Stabilitätsprobleme
Aitken-Neville	—	$\frac{5}{2}n^2$	für Spezialanwendungen
Clenshaw, n bel. Clenshaw, $n = 2^m$	c_i : $2n^2$ $m \cdot n$	$3n$ $3n$	Einfach, stabiler als Newton. F.F.T.(Numerik IIA), spezielle Gitter, Übergang von 2^m zu 2^{m+1}.

In den Kapiteln 3.2 bis 3.4 wurden eine Reihe verschiedener Methoden zur Bestimmung von

Interpolationspolynomen vorgestellt. In 3.2 und 3.3 waren die Knoten völlig willkürlich vorgebbar, in 3.4 wurde nur das Intervall $[-1,1]$ behandelt bei Verwendung der Knoten (3.4.6). In der obigen Tabelle werden die verschiedenen Methoden gegenüberstellt. Dabei wird der Rechenaufwand getrennt nach der Vorberechnung von Koeffizienten, der bei jeder Interpolation einmalig auftritt, und der Polynomauswertung bei bekannten Koeffizienten.

3.6 Interpolationsfehler, optimale Knoten

Bisher wurde davon ausgegangen, daß die "Eingangsdaten" (x_i, y_i) festgelegt sind und daher nur die Berechnung und Auswertung des Interpolationspolynoms behandelt. Wenn das Ziel der Interpolation aber die Approximation einer gegebenen Funktion ist, kann der Fehler ohne Mehraufwand durch geeignete Wahl der Interpolationsknoten stark verkleinert werden. Dazu setzt man an der Darstellung (3.3.11) des Interpolationsfehlers an und benutzt die Supremumsnorm

$$\|g\|_\infty := \sup\{|g(x)| : x \in [\alpha,\beta]\}, \quad g \in C[\alpha,\beta].$$

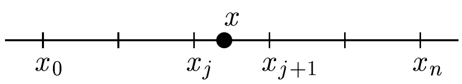
Für eine "glatte" Funktion $f \in C^{n+1}[\alpha,\beta]$ erhält man aus der Fehlerdarstellung (3.3.11) mit $\omega_{n+1}(x) := (x-x_0)\cdots(x-x_n)$ und $x \in [\alpha,\beta]$, $\xi \in (\alpha,\beta)$, die Schranke

$$|f(x) - p_n(x)| = |\omega_{n+1}(x)| \frac{|f^{(n+1)}(\xi)|}{(n+1)!} \le |\omega_{n+1}(x)| \frac{\|f^{(n+1)}\|_\infty}{(n+1)!}. \tag{3.6.1}$$

Bei festem Polynomgrad n ist der Anteil $\|f^{(n+1)}\|_\infty$ durch die Funktion f vorgegeben. Dagegen kann aber der Beitrag $|\omega_{n+1}(x)|$ durch geeignete Wahl der Stützstellen beeinflußt werden. Zwei mögliche Situationen werden diskutiert:

Problem 1: Es sind sehr viele Funktionswerte von f gegeben an äquidistanten Stellen $x_i = \alpha + ih,\ i = 0,1,\ldots$ (z.B. Funktionstafel, Meßreihe). Welche Knoten eignen sich, bei festem n, am besten zur Auswertung von f an einer bestimmten Stelle x?

Es sei nun speziell $x \in [x_j, x_{j+1}],\ 0 \le j < n$.



Dann folgt

$$\begin{aligned} |\omega_{n+1}(x)| &= |x-x_0|\cdots|x-x_j||x-x_{j+1}|\cdots|x-x_n| \\ &\le h(j+1-0)h(j+1-1)\cdots hh\cdots h(n-j) = h^{n+1}(j+1)!(n-j)! \\ &= h^{n+1}\frac{(n+1)!}{\binom{n+1}{j+1}}. \end{aligned} \tag{3.6.2}$$

Da die Binomialkoeffizienten bei $\frac{n+1}{2}$ den größten Wert haben, ist der Fehler minimal für

$$j+1 = \lfloor \frac{n+1}{2} \rfloor,$$

d.h., es ist am günstigsten, links und rechts von x etwa gleich viele Stützstellen zu wählen.

Problem 2: Wenn f durch das Polynom p_n *gleichmäßig* möglichst gut approximiert werden soll, ist der maximale Fehler im Intervall $[\alpha,\beta]$ zu betrachten. Dieser erfüllt nach (3.6.1)

$$\|f-p_n\|_\infty \le \|\omega_{n+1}\|_\infty \frac{\|f^{(n+1)}\|_\infty}{(n+1)!}. \tag{3.6.3}$$

Da in dieser Abschätzung nur der erste Faktor beeinflußt werden kann, soll dieser minimiert werden. Dazu sind diejenigen Knoten $x_0,\ldots,x_n$ zu suchen mit

$$\|\omega_{n+1}\|_\infty = \min_{\xi_j\in[\alpha,\beta]} \max_{x\in[\alpha,\beta]} |x-\xi_0||x-\xi_1|\cdots|x-\xi_n|. \tag{3.6.4}$$

Die Lösung dieses Problems kann explizit angegeben werden, es gilt

Satz 3.6.1 *Die optimalen Knoten in (3.6.4) sind die* Tschebyscheff-*Punkte*

$$x_k := \frac{\alpha+\beta}{2} - \frac{\beta-\alpha}{2}\cos\Big(\frac{2k+1}{2n+2}\pi\Big), \quad k=0,\ldots,n. \tag{3.6.5}$$

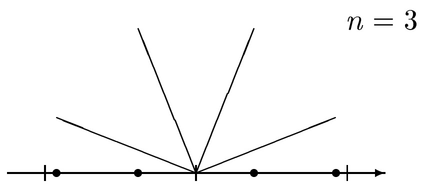
Der Minimalwert der Norm ist

$$\|\omega_{n+1}\|_\infty = \max_{x\in[\alpha,\beta]} |\omega_{n+1}(x)| = 2\left(\frac{\beta-\alpha}{4}\right)^{n+1}. \tag{3.6.6}$$

Bemerkung: Die Knoten ergeben sich aus den in (3.4.6) genannten durch affine Transformation des Intervalls

$$\begin{cases} [-1,1] & \to \quad [\alpha,\beta] \\ \quad\xi & \mapsto \quad \frac{\alpha+\beta}{2}-\frac{\beta-\alpha}{2}\xi = x \end{cases}. \tag{3.6.7}$$

Die Punkte $\xi_k = \cos\frac{2k+1}{2n+2}\pi = Re\exp\left(i\frac{2k+1}{2n+2}\pi\right)$ sind die Realteile der $(4n+4)$-ten komplexen Einheitswurzeln. Sie liegen am Rand des Intervalls $[-1,1]$ *viel dichter* als im Innern.



Beweis des Satzes: Beim Standardintervall $[\alpha,\beta]=[-1,1]$ gilt $x_k=-\xi_k,\ k=0,\ldots,n$, also

$$T_{n+1}(x_k) = \cos\left((n+1)\arccos x_k\right) = \cos\Big((n+1)\frac{2k+1}{2n+2}\pi\Big) = \cos\big(\frac{\pi}{2}+k\pi\big) = 0.$$

Daher ist ω_{n+1}, bis auf eine Konstante, das Tschebyscheff-Polynom. Ein Vergleich des höchsten Koeffizienten mit (3.4.8) liefert genauer

$$\omega_{n+1}(x) = x^{n+1}+\ldots = 2^{-n}T_{n+1}(x).$$

Die Funktion $\cos(n+1)\varphi$ nimmt ihre Extrema ±1 in den Stellen $\varphi_j=\frac{j\pi}{n+1}$ an, das Polynom ω_{n+1} seine Extrema $\pm2^{-n}$ daher in $t_j=\cos\frac{j\pi}{n+1},\ j=0,\ldots,n+1$. Daher gilt

$$\|\omega_{n+1}\|_\infty = 2^{-n}.$$

Dies ist tatsächlich der Minimalwert dieser Norm, gäbe es nämlich ein Polynom der Form $p_{n+1} = x^{n+1} + \ldots$ mit *kleinerer* Norm, dann müßte für dessen Differenz zu ω_{n+1} in den Stellen t_j gelten

$$\omega_{n+1}(t_j) - p_{n+1}(t_j) \begin{cases} < 0 & \text{für} \quad n+1-j \text{ gerade} \\ > 0 & \text{für} \quad n+1-j \text{ ungerade} \end{cases}, \ j = 0, \ldots, n+1.$$

Das Polynom $\omega_{n+1} - p_{n+1}$ vom Grad n (!) hätte dann aber mindestens $n+1$ Nullstellen. Dies führt zu einem Widerspruch.

Durch die Transformation (3.6.7) wird

$$T_{n+1}(\xi) = T_{n+1}\Big(\frac{\alpha+\beta-2x}{\beta-\alpha}\Big) = 2^n\Big(\frac{-2}{\beta-\alpha}\Big)^{n+1} x^{n+1} + \ldots,$$

die Normierung des höchsten Koeffizienten ergibt damit

$$\omega_{n+1}(x) = \Big(-\frac{\beta-\alpha}{2}\Big)^{n+1} 2^{-n} T_{n+1}(\xi).$$

Daraus folgt die Schranke (3.6.6). ■

Nach dem Satz von Weierstraß ist auf einem kompakten Intervall jede stetige Funktion beliebig genau durch Polynome approximierbar. Dieses Ergebnis läßt sich aber nicht unbesehen auf Interpolationspolynome übertragen, selbst wenn immer mehr Stützstellen verwendet werden. Dazu wird die Fehleraussage (3.6.3) für zwei Klassen von Knoten betrachtet. Für äquidistante Punkte $x_j = \alpha + j(\beta-\alpha)/n,\ j = 0, \ldots, n$, ergibt sich aus (3.6.2) die einfache Schranke

$$\|\omega_{n+1}\|_\infty \le \Big(\frac{\beta-\alpha}{n}\Big)^{n+1} n! \le c\Big(\frac{\beta-\alpha}{e}\Big)^{n+1}\frac{1}{\sqrt{n}}, \qquad \text{äquidistant},$$

deren asymptotisches Verhalten mit der Stirlingschen Formel ($n! \sim (n/e)^n\sqrt{2\pi n}$) approximiert wurde. Aus dieser Aussage und der aus (3.6.6) folgen explizite Schranken für die Interpolationsfehler bei äquidistanten und Tschebyscheff-Knoten:

$$\|p_n - f\|_\infty \le \frac{c}{\sqrt{n}}\Big(\frac{\beta-\alpha}{e}\Big)^{n+1}\frac{\|f^{(n+1)}\|}{(n+1)!}, \quad \text{äquidistant}, \tag{3.6.8}$$

$$\|p_n - f\|_\infty \le 2\Big(\frac{\beta-\alpha}{4}\Big)^{n+1}\frac{\|f^{(n+1)}\|}{(n+1)!}, \quad \text{Tschebyscheff}. \tag{3.6.9}$$

Bei diesen Formeln ist zu beachten, daß sich bei einer Vergrößerung des Polynomgrads n auch die Ordnung der Ableitung $f^{(n+1)}$ erhöht. Schon lange bekannt ist das Beispiel

$$f(x) := \frac{1}{1+25x^2}, \quad x \in [-1, 1],$$

wo bei äquidistanten Stützstellen das Anwachsen der Ableitungen für $n \to \infty$ verhindert, daß der Fehler klein wird. Tatsächlich divergiert die Folge der Interpolationspolynome $\{p_n\}$ $(n \to \infty)$ bei diesem Beispiel in Randnähe bei ± 1. Im Vergleich von (3.6.8) und (3.6.9) verbessert der Übergang zu Tschebyscheffpunkten den wichtigsten Vorfaktor e^{-n} auf 4^{-n}. Für Tschebyschefknoten und $f \in C^{m+1}[-1,1], m \geq 1$, läßt sich sogar Konvergenz nachweisen in der Form [Schaback, 10.4.5]

$$\|p_n - f\|_\infty = o\Big(\frac{\log n}{n^m}\Big),\ n \to \infty$$

Diese Beobachtungen legen die Frage nahe, wie weit man mit dem *interpolierenden* Polynom noch vom *bestapproximierenden* entfernt ist, d.h. von demjenigen $\hat{p}_n \in \Pi_n$ mit

$$\|\hat{p}_n - f\|_\infty = \min_{p_n \in \Pi_n} \|p_n - f\|_\infty =: \operatorname{dist}(f, \Pi_n). \tag{3.6.10}$$

Dazu wird der Interpolationsprozeß als ein Operator

$$P_n : \left\{ \begin{array}{ccl} C[\alpha,\beta] & \mapsto & \Pi_n, \\ g & \mapsto & P_n g,\quad (P_n g)(x_i) = g(x_i),\ i = 0,\ldots,n, \end{array} \right. \tag{3.6.11}$$

betrachtet. Dieser Operator ist linear und ein Projektor, $P_n^2 = P_n$, denn Polynome $p \in \Pi_n$ werden reproduziert, $P_n p = p$. Die Norm dieses Projektors kann aus den Lagrange-Polynomen berechnet werden. Dazu sei

$$\lambda_n(x) := \sum_{i=0}^{n} |L_i(x)| \qquad \text{die } \textit{Lebesgue-Funktion.} \tag{3.6.12}$$

Diese tritt im folgenden Satz auf, der ein sehr allgemeines Beweisprinzip benutzt.

Satz 3.6.2 *Für $f \in C[\alpha,\beta]$ gilt*

$$\begin{aligned} \|P_n f - f\|_\infty &\leq (1 + \|P_n\|)\ \mathit{dist}(f, \Pi_n), && (3.6.13)\\ \|P_n\| &= \|\lambda_n\|_\infty \geq 1, && (3.6.14) \end{aligned}$$

wobei $\|P_n\| := \sup\Big\{\|P_n g\|_\infty / \|g\|_\infty :\ g \in C[\alpha,\beta],\ g \not\equiv 0\Big\}$.

Bemerkung: Dies zeigt, daß der Interpolationsfehler höchstens um den Faktor $1 + \|P_n\| \geq 2$ vom minimalen Fehler (3.6.10) abweicht. Daher interessieren Interpolationsknoten mit möglichst kleinem Wert $\|P_n\| = \|\lambda_n\|_\infty$.

Beweis Mit einem beliebigem $p \in \Pi_n$ gilt wegen $P_n p = p$ die Aussage

$$\begin{aligned} \|P_n f - f\|_\infty &= \|P_n f - P_n p + p - f\|_\infty = \|(I - P_n)(p - f)\|_\infty \\ &\leq (1 + \|P_n\|)\|p - f\|_\infty. \end{aligned}$$

Auf der rechten Seite kann nun zum Infimum von $\|p - f\|_\infty$ übergegangen werden.

Für die Norm $\|P_n\|$ folgt aus der Lagrange-Darstellung

$$\|P_n g\|_\infty = \|\sum_{i=0}^{n} g(x_i) L_i\|_\infty \leq \|g\|_\infty \|\sum_{i=0}^{n} |L_i|\|_\infty = \|g\|_\infty \|\lambda_n\|_\infty,$$

d.h., $\|P_n\| \le \|\lambda_n\|_\infty$. Zum Nachweis der Gleichheit sei $\hat{x} \in [\alpha, \beta]$ eine Maximalstelle von λ_n, also $\lambda_n(\hat{x}) = \|\lambda_n\|_\infty$ und g eine spezielle Funktion mit $\|g\|_\infty = 1$ und $g(x_i) = \text{sign } L_i(\hat{x})$. Hierfür gilt

$$\|P_n g\|_\infty \ge |P_n g(\hat{x})| = |\sum_{i=0}^{n} L_i(\hat{x}) \text{ sign } L_i(\hat{x})| = \lambda_n(\hat{x}).$$

Dies zeigt die Gleichung in (3.6.14). Die untere Schranke für $\|P_n\|$ gilt wegen $\|p\| = \|P_n p\|_\infty \le \|P_n\| \|p\|_\infty \; \forall p \in \Pi_n$. ■

Die Bestimmung eines bestapproximierenden Polynoms (3.6.10) ist ein sehr aufwendiger Prozeß. Zur Klärung der Frage, ob im konkreten Fall nicht eine gute Interpolierende ausreicht, kann der "Verlustfaktor" $1 + \|P_n\|$ herangezogen werden. Für Tschebyscheffknoten wächst er wie $1.52 + \frac{2}{\pi} \ln(n+1)$. Etwas günstiger verhält sich dieser Faktor für die *gestreckten Tschebyscheffknoten*

$$x_k = \frac{\alpha+\beta}{2} - \frac{\beta-\alpha}{2} \frac{\cos \frac{2k+1}{2n+2}\pi}{\cos \frac{\pi}{2n+2}}, \; k = 0, \dots, n,. \tag{3.6.15}$$

bei denen $x_0 = \alpha, x_n = \beta$ gilt. Die folgende Tabelle enthält die entsprechenden Werte für einige Polynomgrade.

$n =$	5	8	10	15	20	
$1 + \|P_n\| \cong$	3.1	3.4	3.5	3.7	3.9	Tschebyscheffknoten (3.6.5)
	2.6	2.9	3.0	3.3	3.5	gestreckte Tscheb.Knoten (3.6.15)

4 Spline-Funktionen

4.1 Definition

Der Fehler bei der Interpolation einer Funktion $f \in C^{k+1}[a,b]$ durch ein Polynom p_k vom Grad k mit Stützstellen $\xi_0, \ldots, \xi_k \in [a,b]$ hat nach (3.6.1) die Gestalt ($x \in [a,b]$)

$$|f(x) - p_k(x)| \leq |(x-\xi_0)\cdots(x-\xi_k)| \frac{\|f^{(k+1)}\|_\infty}{(k+1)!} \leq \frac{(b-a)^{k+1}}{(k+1)!} \|f^{(k+1)}\|_\infty. \tag{4.1.1}$$

Im letzten Paragraph wurde ein Beispiel genannt, bei dem für $k \to \infty$ *i.a. keine Konvergenz* $\|f - p_k\|_\infty \to 0$ garantiert werden kann. Dagegen folgt aus (4.1.1) trivialerweise

$$|f - p_k| \to 0 \text{ für } (b-a) \to 0, \; k \text{ fest.}$$

Dies nutzt man aus durch Unterteilung des Gesamtintervalls $[a,b]$ mit einem Gitter

$$\Delta: \quad a = x_0 < x_1 < \ldots < x_n = b, \quad h_i := x_{i+1} - x_i, \; i = 0, \ldots, n-1, \; H := \max_{i=0}^{n-1} h_i, \tag{4.1.2}$$

und approximiert die Funktion f in jedem Teilintervall $[x_i, x_{i+1}]$ durch ein Polynom *festen Grades.*

Definition 4.1.1 *Zu einem Gitter Δ nach (4.1.2) und $m, k \in \mathbf{N}_0$, $0 \leq m < k$, wird der Raum $S^m_{\Delta,k}$ aller* Spline-*Funktionen s bezeichnet, die*

a) lokal, in jedem Teilintervall $(x_i, x_{i+1}]$, Polynome vom Grad k sind, d.h., $s|_{(x_i,x_{i+1}]} \in \Pi_k$,

b) global m-mal stetig differenzierbar sind, d.h., $s \in C^m[a,b]$.

Hier wird konkret nur der Fall der *kubischen Splines* ($k = 3$) mit $m \in \{1, 2\}$ behandelt, einige Aussagen werden aber allgemeiner formuliert.

Anwendungsgebiete von Splines:

— Approximation (nicht-analytischer) Funktionen
— Approximation von Meßdaten
— Kurven- und Flächendarstellung in der Computer-Graphik (CAD)
— Lösung von gewöhnlichen und partiellen Differentialgleichungen (FEM, Finite-Elemente-Methode)

Von besonderer Bedeutung sind die sogenannten *natürlichen Splines* aus $S^{2m-2}_{\Delta,2m-1}$ wegen der folgenden *Minimaleigenschaft.*

Satz 4.1.2 *Mit einem Gitter Δ nach (4.1.2) und $f \in C^2[a,b]$ erfülle die Splinefunktion $s \in S^2_{\Delta,3}$ die Interpolationsbedingungen*

$$s(x_i) = f(x_i), \; i = 0, 1, \ldots, n, \tag{4.1.3}$$

sowie eine der Bedingungen

$$\begin{aligned} &a) \quad s''(a) = s''(b) = 0, \\ &b) \quad s'(a) = f'(a),\ s'(b) = f'(b). \end{aligned}$$

Dann gilt

$$\|f'' - s''\|_2^2 = \|f''\|_2^2 - \|s''\|_2^2 \geq 0, \tag{4.1.4}$$

mit der Norm $\|g\|_2^2 := \int_a^b g(x)^2 dx$. *Der so definierte Spline ist eindeutig bestimmt.*

Beweis Mit dem L_2-Innenprodukt $(f,g)_2 := \int_a^b f(x)g(x)\,dx$ gilt

$$\begin{aligned} \|f'' - s''\|_2^2 - \|f''\|_2^2 + \|s''\|_2^2 &= -2(f'', s'')_2 + 2(s'', s'')_2 = 2(s'' - f'', s'')_2 \\ &= 2\int_a^b [s''(x) - f''(x)]s''(x)\,dx. \end{aligned}$$

Durch partielle Integration auf den einzelnen Teilintervallen $[x_i, x_{i+1}]$ ergibt sich hier

$$\begin{aligned} \int_{x_i}^{x_{i+1}} [s'' - f'']s''\,dx &= (s' - f')s''\Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} (s' - f')s'''\,dx \\ &= (s' - f')s''\Big|_{x_i}^{x_{i+1}} - \underbrace{(s - f)}_{=0,(4.1.3)} s'''\Big|_{x_i}^{x_{i+1}} + \int_{x_i}^{x_{i+1}} (s - f) \underbrace{s^{(4)}}_{=0, s\in\Pi_3} dx \end{aligned}$$

Der verbleibende Beitrag $(s' - f')s''$ ist stetig, bei Summation über die Teilintervalle zeigt sich

$$\int_a^b [s'' - f'']s''\,dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} \ldots = \Big(s'(x) - f'(x)\Big)s''(x)\Big|_a^b = 0.$$

Dabei verschwindet die Klammer, wenn Bedingung b) erfüllt ist, bzw. die zweite Ableitung bei Bedingung a).

Zum Nachweis der Eindeutigkeit seien $s, \bar{s}$ zwei Splines, die die Voraussetzungen erfüllen. Für beide gilt dann (4.1.4), also

$$0 \leq \|s''\|_2^2 - \|\bar{s}''\|_2^2 = \|s'' - \bar{s}''\|_2^2 = \|\bar{s}''\|_2^2 - \|s''\|_2^2 \leq 0.$$

Daher ist $s'' - \bar{s}'' \equiv 0$, also $s(x) - \bar{s}(x) = cx + d,\ c, d \in \mathbf{R}$. Da die Differenz wegen (4.1.3) aber in allen x_i verschwindet, gilt $s = \bar{s}$. ■

Bemerkung: Unter allen Funktionen, die die Werte $(x_i, f(x_i))$ interpolieren, hat der natürliche kubische Spline nach (4.1.4) minimale zweite Ableitung im quadratischen Mittel, bei b) etwa ist

$$\|s''\|_2 = \min\{\|g''\|_2 :\ g \in C^2[a,b], g(x_i) = f(x_i)\,\forall i,\ g''(a) = g''(b) = 0\}.$$

In der Mechanik entspricht $\|g''\|_2$ der Biegenergie eines elastischen Lineals, wie es früher im Schiffsbau eingesetzt wurde (*Strak-Latte*, engl. *Spline*).

4.2 Existenz des Interpolations-Splines

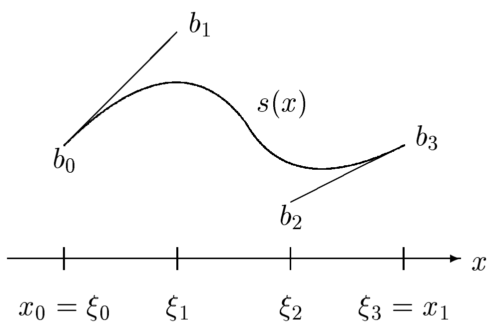
Für den Interpolationsspline $s \in S^2_{\Delta,3}$, der gegebene Werte y_i interpoliert, $s(x_i) = y_i$, $i = 0, \dots, n$, wird in jedem Teilintervall eine Darstellung als Polynom benötigt. Dazu bietet sich die folgende *Beziér-Bernstein-Darstellung* an, die auf dem Standardintervall $[0, h]$, $h > 0$, lautet

$$s(x) = \frac{1}{h^k} \sum_{j=0}^{k} \binom{k}{j} x^j (h-x)^{k-j}\, b_j \tag{4.2.1}$$

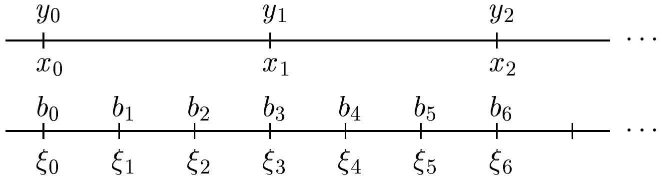
mit Koeffizienten $b_0, \dots, b_k$. Diese Darstellung besitzt folgende Eigenschaften:

$$\left.\begin{array}{ll} (a) & b_j \equiv 1 \;\Rightarrow\; s(x) = \frac{1}{h^k} \sum_{j=0}^{k} \binom{k}{j} x^j (h-x)^{k-j} = \frac{1}{h^k}(x+h-x)^k \equiv 1 \\ (b) & s(0) = b_0, \quad s(h) = b_k \\ (c) & s'(x) = \frac{1}{h^k}[-k(h-x)^{k-1} b_0 + k(h-kx)(h-x)^{k-2} b_1 + x(\dots)] \Rightarrow \\ & s'(0) = \dfrac{b_1 - b_0}{h/k}, \quad s'(h) = \dfrac{b_k - b_{k-1}}{h/k} \\ (d) & s''(0) = \frac{k(k-1)}{h^2}(b_0 - 2b_1 + b_2), \quad s''(h) = \frac{k(k-1)}{h^2}(b_{k-2} - 2b_{k-1} + b_k). \end{array}\right\} \tag{4.2.2}$$

Die Eigenschaft (c) führt bei $k = 3$ auf eine einfache *geometrische* Interpretation der Koeffizienten b_j. Werden die Paare $(\xi_j, b_j)^\top$ mit $\xi_j = jh/3$ betrachtet, dann ist die Gerade $\overline{(\xi_0, b_0)^\top\ (\xi_1, b_1)^\top}$ die Tangente an s in $x = \xi_0 = 0$ und $\overline{(\xi_2, b_2)^\top\ (\xi_3, b_3)^\top}$ die Tangente an s in $x = \xi_3 = h$. Durch b_0, b_3 werden also die Werte von s in den Randpunkten beeinflußt, durch b_1, b_2 die dortigen Ableitungen. Man bezeichnet die Koeffizienten oft als *Kontrollpunkte*.



Der Spline s wird auf dem Gesamtintervall $[a, b]$ durch lokale Darstellungen (4.2.1) zusammengesetzt. Dazu wird ein Zusatzgitter eingeführt,



$$\begin{array}{l} \xi_{ik+j} := x_i + \frac{j}{k} h_i, \\ j = 0, \dots, k-1, \\ i = 0, \dots, n-1, \end{array} \tag{4.2.3}$$

Die *Stetigkeitsbedingungen* an s im Punkt $x_j = \xi_{kj}$, $1 \le j \le n-1$, mit Polynomstücken (4.2.1) lauten für

C^0: ist erfüllt mit $b_{kj} = y_j$.

C^1: Nach (4.2.2,c) entspricht die Stetigkeit der 1. Ableitung

$$s'(x_j-) = \frac{k}{h_{j-1}}(b_{kj} - b_{kj-1}) \stackrel{!}{=} \frac{k}{h_j}(b_{kj+1} - b_{kj}) = s'(x_j+) \quad \Longleftrightarrow$$

$$\frac{1}{h_{j-1}}b_{kj-1}+\frac{1}{h_j}b_{kj+1}=(\frac{1}{h_{j-1}}+\frac{1}{h_j})b_{kj}=(\frac{1}{h_{j-1}}+\frac{1}{h_j})y_j. \tag{4.2.4}$$

2. Ableitung: Nach (4.2.2,d) ist die Bedingung $s''(x_j-)\stackrel{!}{=}s''(x_j+)$ äquivalent mit

$$\frac{1}{h_{j-1}^2}(b_{kj-2}-2b_{kj-1}+b_{kj})=\frac{1}{h_j^2}(b_{kj}-2b_{kj+1}+b_{kj+2})=:\mu_j \tag{4.2.5}$$

Im folgenden werden jetzt die Werte $\mu_j=\frac{1}{6}s''(x_j)$ aus (4.2.5) als eigentliche Parameter des Splines interpretiert, die Koeffizienten b lassen sich daraus bei $k=3$ berechnen. Die Beziehungen (4.2.5) an den beiden Rändern von $[x_j,x_{j+1}]$,

$$\begin{aligned} 2b_{3j+1}-b_{3j+2} &= y_j-h_j^2\mu_j\\ -b_{3j+1}+2b_{3j+2} &= y_{j+1}-h_j^2\mu_{j+1}\end{aligned}$$

können nämlich nach b_{3j+1},b_{3j+2} aufgelöst werden und ergeben

$$\left.\begin{aligned} b_{3j+1} &= \tfrac13(2y_j+y_{j+1})-\tfrac13 h_j^2(2\mu_j+\mu_{j+1})\\ b_{3j+2} &= \tfrac13(y_j+2y_{j+1})-\tfrac13 h_j^2(\mu_j+2\mu_{j+1})\end{aligned}\right\}. \tag{4.2.6}$$

Diese Gleichungen führen auf folgende Formulierung der C^1-Bedingungen (4.2.4):

$$h_{j-1}\mu_{j-1}+2(h_{j-1}+h_j)\mu_j+h_j\mu_{j+1}=\frac{y_{j+1}-y_j}{h_j}-\frac{y_j-y_{j-1}}{h_{j-1}},\quad j=1,\ldots,n-1.$$

Bei Division durch $h_{j-1}+h_j=x_{j+1}-x_{j-1}$ kann insbesondere die rechte Seite nach (3.3.2) einfacher als dividierte Differenz dargestellt werden:

$$\underbrace{\frac{h_{j-1}}{h_{j-1}+h_j}}_{=:\,\alpha_j}\mu_{j-1}+2\mu_j+\underbrace{\frac{h_j}{h_{j-1}+h_j}}_{=:\,\beta_j}\mu_{j+1}=y[x_{j-1},x_j,x_{j+1}],\quad j=1,\ldots,n-1. \tag{4.2.7}$$

Zusätzlich sind noch die Randbedingungen aus Satz 4.1.2 zu berücksichtigen. Im

Fall a) $s''(a)=s''(b)=0$, bedeuten diese einfach

$$\mu_0=\mu_n=0.$$

Nach Streichung dieser beiden Unbekannten ist (4.2.7) ein quadratisches Gleichungssystem. Dagegen gilt im

Fall b) $s'(a)=y_0',\ s'(b)=y_n'$, nach (4.2.2,c) und (4.2.6) am linken Rand

$$y_0+\frac{h_0}{3}y_0'=b_1=\frac13(2y_0+y_1)-\frac{h_0^2}{3}(2\mu_0+\mu_1).$$

Da rechts analoges gilt, führen diese Bedingungen b) also auf die zusätzlichen Gleichungen

$$\left.\begin{aligned} 2\mu_0+\mu_1 &= \tfrac{1}{h_0}\left(\tfrac{y_1-y_0}{h_0}-y_0'\right)=y[x_0,x_0,x_1],\\ \mu_{n-1}+2\mu_n &= y[x_{n-1},x_n,x_n].\end{aligned}\right\} \tag{4.2.8}$$

In beiden Fällen ergeben sich die Momente $\mu = (\mu_j)_j$ als Lösung eines Gleichungssystems

$$A\mu = r$$

mit einer *Tridiagonalmatrix*

$$A = \underbrace{\begin{pmatrix} 2 & \beta_1 & & & \\ \alpha_2 & 2 & \beta_2 & & \\ & \alpha_3 & 2 & \beta_3 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}}_{\in \mathbf{R}^{(n-1)\times(n-1)}} \quad \text{bzw.} \quad A = \underbrace{\begin{pmatrix} 2 & \beta_0 & & & \\ \alpha_1 & 2 & \beta_1 & & \\ & \alpha_2 & 2 & \beta_2 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}}_{\in \mathbf{R}^{(n+1)\times(n+1)}}. \tag{4.2.9}$$

Satz 4.2.1 *Für die Matrix A gilt in beiden Fällen von (4.2.9) die Schranke*

$$\|A^{-1}\|_\infty \le 1.$$

Beweis Nach (4.2.7), (4.2.8) ist $\alpha_j, \beta_j \ge 0,\ \alpha_j + \beta_j = 1$, (mit $\alpha_0 := \beta_{n+1} := 0$). Diese Matrix ist diagonaldominant, daher gilt für jeden Vektor r und μ mit $A\mu = r$ und jeden Index j:

$$2|\mu_j| = |r_j - \alpha_j\mu_{j-1} - \beta_j\mu_{j+1}| \le |r_j| + \underbrace{(\alpha_j + \beta_j)}_{\le 1}\|\mu\|_\infty.$$

Betrachtet man auf der linken Seite den Index mit $|\mu_j| = \|\mu\|_\infty$, sieht man

$$2\|\mu\|_\infty \le \|r\|_\infty + \|\mu\|_\infty \ \Rightarrow \|\mu\|_\infty = \|A^{-1}r\|_\infty \le \|r\|_\infty. \qquad \blacksquare$$

Da also insbesondere die Gleichungssysteme mit A immer lösbar sind, folgt

Satz 4.2.2 *Für beliebige Gitter Δ und beliebige Datenwerte y_j, y_0', y_n', existiert der eindeutig bestimmte Interpolationsspline $s \in S^2_{\Delta,3}$ (vgl. Satz 4.1.2) mit $s(x_j) = y_j,\ j = 0, \dots, n$ sowie den Randbedingungen $s''(a) = s''(b) = 0$ oder $s'(a) = y_0', s'(b) = y_n'$.*

Mit Hilfe von Satz 4.2.1 läßt sich auch einfach eine a-priori-Schranke für den Spline herleiten, bei einem äquidistantes Gitter ($h_j \equiv h$), etwa gilt

$$|s(x)| \le 3\,\|y\|_\infty \quad \Big(+\,h\max\{|y_0'|, |y_n'|\}\Big), \quad x \in [a, b].$$

Dagegen existiert eine solche Schranke für Interpolationspolynome nicht mit einem für alle n beschränkten Vorfaktor.

4.3 Konvergenz des Interpolations-Splines

Wie nach der einleitenden Bemerkung zu erwarten, konvergiert der Interpolationsspline einer glatten Funktion gegen diese bei feiner werdendem Gitter ($H \to 0$). Überraschenderweise zeigt sich außerdem, daß dabei *keine weiteren* Voraussetzungen über das Gitter benötigt werden und daß sogar die ersten drei *Ableitungen* des Splines gegen die der Funktion konvergieren. Zum Nachweis dieser Aussagen wird folgendes Hilfsmittel benötigt.

Satz 4.3.1 *Verschwindet eine Funktion $g \in C^2[a,b]$ in den Randpunkten, $g(a) = g(b) = 0$, gilt*

$$\|g\|_\infty \le \frac{(b-a)^2}{8}\|g''\|_\infty, \quad \|g'\|_\infty \le (b-a)\|g''\|_\infty.$$

Beweis Taylorentwicklung um $x \in (a,b)$ ergibt

$$\begin{aligned} 0 = g(a) = &\quad g(x) + (a-x)g'(x) + \tfrac{1}{2}(a-x)^2 g''(\xi_1),\ \xi_1 \in (a,b), \\ 0 = g(b) = &\quad g(x) + (b-x)g'(x) + \tfrac{1}{2}(b-x)^2 g''(\xi_2),\ \xi_2 \in (a,b). \end{aligned}$$

Durch Multiplikation der Gleichungen mit $(b-x)$ bzw. $(x-a)$ und Addition wird $g'(x)$ eliminiert und man erhält

$$0 = (b-x+x-a)g(x) + \frac{1}{2}(b-x)(x-a)[(x-a)g''(\xi_1) + (b-x)g''(\xi_2)].$$

Daraus folgt dann die erste Ungleichung,

$$|g(x)| \le \frac{(b-x)(x-a)}{2(b-a)}(|x-a| + |b-x|)\|g''\|_\infty \le \frac{(b-a)^2}{8}\|g''\|_\infty.$$

Die zweite Ungleichung gilt, da nach dem Satz von Rolle ein $x_0 \in [a,b]$ existiert mit $g'(x_0) = 0$ und daher

$$|g'(x)| = |g'(x_0) + (x-x_0)g''(\xi)| = |x-x_0||g''(\xi)| \le (b-a)\|g''\|_\infty. \qquad \blacksquare$$

Die Konvergenz des Interpolationssplines nach Satz 4.1.2,b wird im folgenden Satz beschrieben. Der Fehler in einem Punkt x hängt dabei sowohl von der maximalen Schrittweite H als auch von der lokalen h_j ab. Da die Fehlerschranke für die Funktionswerte vereinfacht die Form $\|s-f\| = O(H^4)$ $(H \to 0)$ hat, sagt man, der Spline konvergiere mit *Ordnung* 4.

Satz 4.3.2 *Gegeben sei $f \in C^4[a,b]$ und ein Gitter Δ. Dann gelten für den Interpolationsspline $s \in S^2_{\Delta,3}$ mit*

$$s(x_j) = f(x_j),\ j = 0,\ldots,n, \quad s'(a) = f'(a),\ s'(b) = f'(b),$$

und seine Ableitungen folgende Fehlerschranken. Für $x \in (x_j, x_{j+1}]$, $0 \le j < n$, ist

$$|s^{(i)}(x) - f^{(i)}(x)| \le c_i H^2 h_j^{2-i}\|f^{(4)}\|_\infty, \quad 0 \le i \le 3.$$

Beweis Es wird eine weitere Splinefunktion $\hat{s} \in S^0_{\Delta,3}$ eingeführt, die die Bedingungen

$$\hat{s}(x_j) = f(x_j),\ \hat{s}''(x_j) = f''(x_j), \quad j = 0,\ldots,n,$$

erfüllt. Dann ist $\hat{b}_{3j} = b_{3j} = f(x_j) =: f_j$ und nach (4.2.6) gilt mit $\hat{\mu}_j := \frac{1}{6}\hat{s}''(x_j)$ die Beziehung

$$\left\{\begin{array}{lcl} \hat{b}_{3j+1} & = & \frac{1}{3}(2f_j + f_{j+1}) - \frac{1}{3}h_j^2(2\hat{\mu}_j + \hat{\mu}_{j+1}) \\ \hat{b}_{3j+2} & = & \frac{1}{3}(f_j + 2f_{j+1}) - \frac{1}{3}h_j^2(\hat{\mu}_j + 2\hat{\mu}_{j+1}) \end{array}\right. .$$

Daraus folgt

$$|b_{3j+\nu} - \hat{b}_{3j+\nu}| \le h_j^2 \|\mu - \hat{\mu}\|_\infty, \quad \nu = 1, 2, \tag{4.3.1}$$

und aus der Beziérdarstellung (4.2.1) in $(x_j, x_{j+1}]$,

$$s(x) - \hat{s}(x) = \frac{3}{h_j^3} \sum_{\nu=1}^{2} (x - x_j)^\nu (x_{j+1} - x)^{3-\nu} (b_{3j+\nu} - \hat{b}_{3j+\nu}),$$

ergibt sich für die Ableitungen der Differenz die Schranke

$$\begin{aligned} |s^{(i)}(x) - \hat{s}^{(i)}(x)| &\le \frac{3}{h_j^3} \sum_{\nu=1}^{2} \Big| \frac{d^i}{dx^i} [(x - x_j)^\nu (x_{j+1} - x)^{3-\nu}] \Big| \max_\nu |b_{3j+\nu} - \hat{b}_{3j+\nu}| \\ &\le \frac{\tilde{c}_i}{h_j^i} \max_\nu |b_{3j+\nu} - \hat{b}_{3j+\nu}|, \end{aligned}$$

mit $\tilde{c}_0 = \frac{3}{4}$, $\tilde{c}_1 = 3$, $\tilde{c}_2 = 18$, $\tilde{c}_3 = 36$. Mit dieser Aussage und (4.3.1) wurde also bisher gezeigt

$$|s^{(i)}(x) - \hat{s}^{(i)}(x)| \le \tilde{c}_i h_j^{2-i} \|\mu - \hat{\mu}\|_\infty, \ x \in (x_j, x_{j+1}]. \tag{4.3.2}$$

Da die Momente von s aus dem Gleichungssystem $A\mu = r$, (4.2.7), (4.2.8) stammen und nach Satz 4.2.1 $\|A^{-1}\|_\infty \le 1$ gilt, folgt

$$\mu - \hat{\mu} = A^{-1}(r - A\hat{\mu}) \quad \Rightarrow \quad \|\mu - \hat{\mu}\|_\infty \le \|r - A\hat{\mu}\|_\infty.$$

Die Komponenten r_j und $(A\hat{\mu})_j$ der Differenz können durch Taylorentwicklung um x_j verglichen werden. Für $1 \le j < n$ ist

$$\begin{aligned} 6(A\hat{\mu})_j &= \frac{1}{h_{j-1} + h_j} \Big(h_{j-1} f''(x_{j-1}) + 2(h_{j-1} + h_j) f''(x_j) + h_j f''(x_{j+1}) \Big) \\ &= 3f''(x_j) + (h_j - h_{j-1}) f'''(x_j) + \frac{1}{2} \frac{1}{h_{j-1} + h_j} [h_{j-1}^3 f^{(4)}(\eta_1) + h_j^3 f^{(4)}(\eta_2)], \\ r_j &= f[x_{j-1}, x_j, x_{j+1}] = \frac{1}{h_{j-1} + h_j} \Big(\frac{1}{h_j}(f_{j+1} - f_j) - \frac{1}{h_{j-1}}(f_j - f_{j-1}) \Big) \\ &= \frac{1}{2} f''(x_j) + \frac{1}{6}(h_j - h_{j-1}) f'''(x_j) + \frac{1}{24} \frac{1}{h_{j-1} + h_j} [h_{j-1}^3 f^{(4)}(\eta_3) + h_j^3 f^{(4)}(\eta_4)], \end{aligned}$$

mit Zwischenstellen $\eta_\nu \in (x_{j-1}, x_{j+1})$. Für die gesuchte Differenz folgt daher die Schranke

$$|r_j - (A\hat{\mu})_j| \le (\frac{1}{12} + \frac{1}{24}) \frac{h_{j-1}^3 + h_j^3}{h_{j-1} + h_j} \|f^{(4)}\|_\infty \le \frac{H^2}{8} \|f^{(4)}\|_\infty,$$

die sich analog auch in den Fällen $j = 0, n$ nachweisen läßt. Nach Einsetzen dieser Ergebnisse in (4.3.2) zeigt sich für die Differenz der beiden Spline-Funktionen die Aussage

$$|s^{(i)}(x) - \hat{s}^{(i)}(x)| \le \frac{1}{8} \tilde{c}_i H^2 h_j^{2-i} \|f^{(4)}\|_\infty, \ x \in (x_j, x_{j+1}].$$

Zum Nachweis der Behauptung ist nun nur noch zu zeigen, daß eine solche Abschätzung auch für die Differenz $\hat{s} - f$ gilt. Nach Konstruktion verschwinden $\hat{s} - f$ und $\hat{s}'' - f''$ in allen Gitterpunkten x_j, $j = 0, \ldots, n$. Daher besitzt in jedem Teilintervall $(x_j, x_{j+1}]$ die Funktion

$\hat{s} - f$	$\hat{s}' - f'$	$\hat{s}'' - f''$	$\hat{s}''' - f'''$
2 Nullstellen	1 Nullstelle	2 Nullstellen	1 Nullstelle

.

Bei der ersten und dritten Ableitung wurde dazu der Satz von Rolle angewendet. Daher folgen aus Satz 4.3.1 rekursiv die Schranken

$$|\hat{s}'' - f''| \le \frac{1}{8} h_j^2 \|f^{(4)}\|_\infty, \quad |\hat{s}''' - f'''| \le h_j \|f^{(4)}\|_\infty \quad \Rightarrow$$
$$|\hat{s} - f| \le \frac{1}{64} h_j^4 \|f^{(4)}\|_\infty, \quad |\hat{s}' - f'| \le \frac{1}{8} h_j^3 \|f^{(4)}\|_\infty.$$

Die Behauptung ergibt sich jetzt aus den Schranken für die beiden Differenzen in $s^{(i)} - f^{(i)} = s^{(i)} - \hat{s}^{(i)} + \hat{s}^{(i)} - f^{(i)}$. ■

4.4 Spline-Darstellungen, Beziér-, B-Splines

Zur Darstellung von Splines sind zwei Methoden verbreitet. Die erste, mit Beziér-Stücken (4.2.1) bietet sich an, wenn sowohl C^2- als auch C^1- oder C^0-Splines verwendet werden sollen. Allerdings benutzt diese Darstellung bei C^2-Splines unnötig viele Parameter. Dies wird bei Verwendung einer Basis von $S^2_{\Delta,3}$ aus B-Splines vermieden.

Die **Beziér-Darstellung** wird nur für äquidistante Gitter behandelt. Dies ist aber bei einer der wichtigsten Anwendungen, der Parameterdarstellung von Kurven $(s_1(x), s_2(x), \ldots)$ im $\mathbf{R}^2$ bzw. $\mathbf{R}^3$ keine starke Einschränkung. In diesem Fall, $h_j \equiv h$, gibt es für die verschiedenen Stetigkeitsbedingungen einfache geometrische Interpretationen.

- Die C^1-Bedingung (4.2.4) lautet $b_{3j} = \frac{1}{2}(b_{3j-1} + b_{3j+1})$. Da auch $\xi_{3j} = \frac{1}{2}(\xi_{3j-1} + \xi_{3j+1})$ gilt, ist also der $\mathbf{R}^2$-Punkt $(\xi_{3j}, b_{3j})^\mathsf{T}$ Mittelpunkt der Verbindungsstrecke von $(\xi_{3j-1}, b_{3j-1})^\mathsf{T}$ und $(\xi_{3j+1}, b_{3j+1})^\mathsf{T}$.

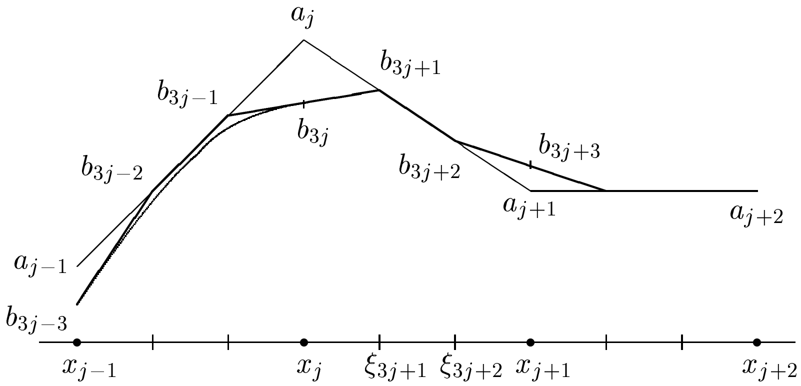
- Spezielle Teilausdrücke in der C^2-Bedingung werden als neue Parameter eingeführt, aus (4.2.5) wird so:

$$\begin{array}{rcll} 2b_{3j-1} - b_{3j-2} & = & 2b_{3j+1} - b_{3j+2} & =: a_j, \qquad \text{außerdem ist} \\ 2\xi_{3j-1} - \xi_{3j-2} & = & 2\xi_{3j+1} - \xi_{3j+2} & = \xi_{3j} = x_j. \end{array}$$

  Daher liegt der Punkt $(x_j, a_j)^\mathsf{T}$ im Schnittpunkt der Geraden durch $(\xi_\nu, b_\nu)^\mathsf{T}$ mit $\nu = 3j-2, 3j-1$ sowie $\nu = 3j+1, 3j+2$. Umgekehrt dritteln die Punkte $(\xi_{3j+\nu}, b_{3j+\nu})^\mathsf{T}$, $\nu = 1, 2$, die Strecke von $(x_j, a_j)^\mathsf{T}$ nach $(x_{j+1}, a_{j+1})^\mathsf{T}$, denn

$$\left.\begin{array}{r} 2b_{3j+1} - b_{3j+2} = a_j \\ -b_{3j+1} + 2b_{3j+2} = a_{j+1} \end{array}\right\} \quad \Rightarrow \quad \left\{\begin{array}{l} b_{3j+1} = \frac{1}{3}(2a_j + a_{j+1}) \\ b_{3j+2} = \frac{1}{3}(a_j + 2a_{j+1}) \end{array}\right. . \tag{4.4.1}$$

Diese Beziehungen der Beziér-Koeffizienten beim C^2-Spline erlauben folgende geometrische Interpretation (Hilfslinien), die analog auch für Kurven $(s_1(x), s_2(x))^\mathsf{T}$ gilt. Der Spline ist hier nur links von x_j skizziert:



Mit (4.4.1) können die C^1-Bedingungen umformuliert werden,

$$2b_{3j} = b_{3j-1} + b_{3j+1} = \frac{1}{3}(a_{j-1} + 4a_j + a_{j+1}).$$

Daher gilt für die Koeffizienten a_j des C^2-Interpolationssplines das Gleichungssystem

$$a_{j-1} + 4a_j + a_{j+1} = 6y_j, \quad j = 1, \ldots, n-1, \tag{4.4.2}$$

mit den Randbedingungen

$$\begin{array}{lll} a_0 = y_0, & a_n = y_n, & \text{bei } s''(a) = s''(b) = 0, \text{ bzw.} \\ 2a_0 + a_1 = 3y_0 + hy_0', & a_{n-1} + 2a_n = 3y_n - hy_n' & \text{bei } s'(a) = y_0', s'(b) = y_n', \end{array} \tag{4.4.3}$$

das zur Berechnung der a_j dienen kann. Die Beziér-Koeffizienten $\{b_\nu\}$ ergeben sich daraus nach (4.4.1).

Zur *Auswertung* eines Splines s in Beziér-Darstellung kann natürlich die Formel (4.2.1) direkt verwendet werden, nach (4.2.2a) ist sie eine konvexe Linearkombination und daher numerisch stabil. Wegen der auftretenden Binomialkoeffizienten läßt sich die Auswertung aber (analog zum Pascalschen Dreieck) auch durch mehrfache lineare Interpolation realisieren. Dazu sei für $x \in [x_j, x_{j+1}]$

$$s(x) = \sum_{i=0}^{k} \binom{k}{i} \xi^i (1-\xi)^{k-i} b_{kj+i}, \quad \xi := \frac{x - x_j}{h}.$$

Dann ergibt sich $s(x) = b_0^{(k)}$ aus dem Dreieckschema

$$\begin{array}{lll} b_i^{(0)} & := & b_{kj+i}, \qquad\qquad\qquad\qquad i = 0, \ldots, k; \\ b_i^{(\nu)} & := & (1-\xi) b_i^{(\nu-1)} + \xi b_{i+1}^{(\nu-1)}, \quad i = 0, \ldots, k-\nu; \ \nu = 1, \ldots, k. \end{array} \tag{4.4.4}$$

Auch (4.4.4) enthält nur konvexe Linearkombinationen.

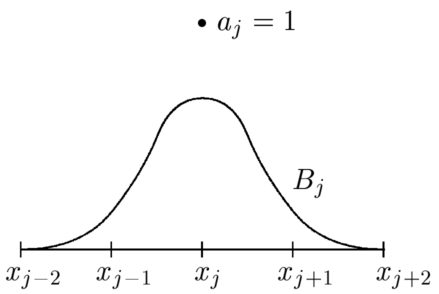
Beweis von (4.4.4): Die Identität $b_i^{(\nu)} = \sum_{\mu=0}^{\nu} \binom{\nu}{\mu} \xi^\mu (1-\xi)^{\nu-\mu} b_{i+\mu}^{(0)}$ läßt sich leicht durch Induktion über ν verifizieren. ■

B-Splines: Die wesentlichen Parameter des Splines s sind die Größen a_j. Da sich die Funktionswerte des Splines daraus durch lineare Operationen (4.4.1), (4.4.4) berechnen existiert eine

Basisdarstellung der Form

$$s(x) = \sum_{j=0}^{n} a_j B_j(x), \quad x \in [a,b]. \tag{4.4.5}$$

Für jedes j, $0 \le j \le n$, entspricht hier die Funktion B_j dem Spline, den man für die speziellen Koeffizienten $(a_m)_m = (\delta_{jm})_m$ erhält.

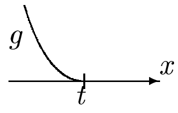


Dieser Spline hat im Innern des Intervalls die gezeigte Gestalt (vgl. oben die geometrische Interpretation). Der Träger von B_j für $2 \le j \le n-2$, ist $[x_{j-2}, x_{j+2}]$, umfaßt also $4 = k+1$ Teilintervalle. In Randnähe ergeben sich etwas unterschiedliche Formen.

Diese Basisfunktionen werden B-Splines genannt. Aus der Herleitung ist ersichtlich, daß diese eine *Zerlegung der Eins* bilden mit $B_j \ge 0$, $\sum_j B_j(x) \equiv 1$, wie dies bei den Bernsteinpolynomen, vgl. (4.2.2,a), schon der Fall war. Die B-Splines können aber auch vollkommen unabhängig davon und für beliebige Gitter Δ definiert werden. Die Bezeichnungen vereinfachen sich in Randnähe, wenn das Gitter um $2k = 6$ Punkte erweitert wird,

$$\overline{\Delta}: \quad x_{-3} < x_{-2} < x_{-1} < x_0 < \ldots < x_n < x_{n+1} < \ldots < x_{n+3}, \tag{4.4.6}$$

z.B. durch $x_{-1} := 2x_0 - x_{-1}$, $x_{n+1} := 2x_n - x_{n-1}, \ldots$. Eine geschlossene Darstellung der B-Splines kann mit Hilfe der folgenden Funktion von zwei Argumenten angegeben werden.



$$g(t;x) := (t-x)_+^3 = \begin{cases} (t-x)^3 & \text{für} \quad t-x \ge 0, \\ 0 & \text{für} \quad t-x < 0. \end{cases} \tag{4.4.7}$$

Für festes t ist offensichtlich $g(t;\cdot) \in C^2(\mathbf{R})$.

Definition 4.4.1 *Zum Gitter* $\overline{\Delta}$ *werden die kubischen (normalisierten)* B-Splines *definiert durch die dividierte Differenz*

$$B_i(x) := (x_{i+2} - x_{i-2})\, g[x_{i-2}, x_{i-1}, \ldots, x_{i+2}; x], \quad i = -1, \ldots, n+1.$$

Die Differenzen sind dabei nach (3.3.2) bezüglich des ersten Arguments t von g zu bilden.

Satz 4.4.2 *Die B-Splines besitzen folgende Eigenschaften:*

$$B_i(x) = 0 \quad \forall x \notin (x_{i-2}, x_{i+2}), \tag{4.4.8}$$

$$B_i(x) > 0 \quad \forall x \in (x_{i-2}, x_{i+2}), \tag{4.4.9}$$

$$\sum_{i=-1}^{n+1} B_i(x) = \sum_{i=j-1}^{j+2} B_i(x) \equiv 1, \;\; \textit{wenn } x \in (x_j, x_{j+1}],\ 0 \le j < n. \tag{4.4.10}$$

Beweis (4.4.8): Für $x \geq x_{i+2}$ ist $g(x_j, x) = 0, j = i-2, .., i+2$. Für $x \leq x_{i-2}$ ist der Ausdruck $g[x_{i-2}, x_{i-1}, \ldots, x_{i+2}; x]$ vierte Differenz des kubischen Polynoms $(t-x)^3$ in t, also ebenfalls 0.

(4.4.9): vgl. Böhmer/Spline-Funktionen, §6.3.

(4.4.10): Nach Definition ist $B_i(x) = g[x_{i-1}, .., x_{i+2}; x] - g[x_{i-2}, .., x_{i+1}; x]$. Nach (4.4.8) ist für $x \in (x_j, x_{j+1}]$:

$$\begin{aligned}\sum_i B_i(x) &= \sum_{i=j-1}^{j+2} B_i(x) \\ &= \underbrace{g[x_{j+1}, .., x_{j+4}; x]}_{1} - g[x_j, .., x_{j+3}; x] + g[x_j, .., x_{j+3}; x] + \ldots - \underbrace{g[x_{j-3}, .., x_j; x]}_{0} = 1.\end{aligned}$$

Der Wert 1 ergibt sich vorne aus Satz 3.3.6d, da $g(t; \cdot) = t^3 + \ldots$ ist und der Wert 0 am Ende, da $g \equiv 0$ für die auftretenden Argumentwerte. ∎

Durch die Eigenschaften (4.4.8-4.4.10) ist umgekehrt der B-Spline B_i eindeutig festgelegt und stimmt daher mit dem aus (4.4.5) überein. Zur numerischen Berechnung ist der Zugang über die Differenzen nicht sehr geeignet, da bei der Differenzbildung Rundungsfehler verstärkt werden. Ähnlich wie bei der Beziér-Darstellung in (4.4.4) kann auch für einen Spline in B-Spline-Darstellung ein stabiles Rekursionsschema angegeben werden, wobei bei jeder Auswertung höchstens 4 Koeffizienten a eingehen (vgl. Stoer, §2.4.5).

Sowohl B-Spline- als auch Beziér-Darstellung bieten eine Reihe von **praktischen Vorteilen** bei der Handhabung, die darauf beruhen, daß die verwendeten Basisfunktionen eine *lokale Zerlegung der Eins* bilden:

Der Funktionswert $s(x)$ an einer beliebigen Stelle x im Intervall ist eine konvexe Linearkombination von $k+1=4$ Koeffizienten.

Konsequenzen:

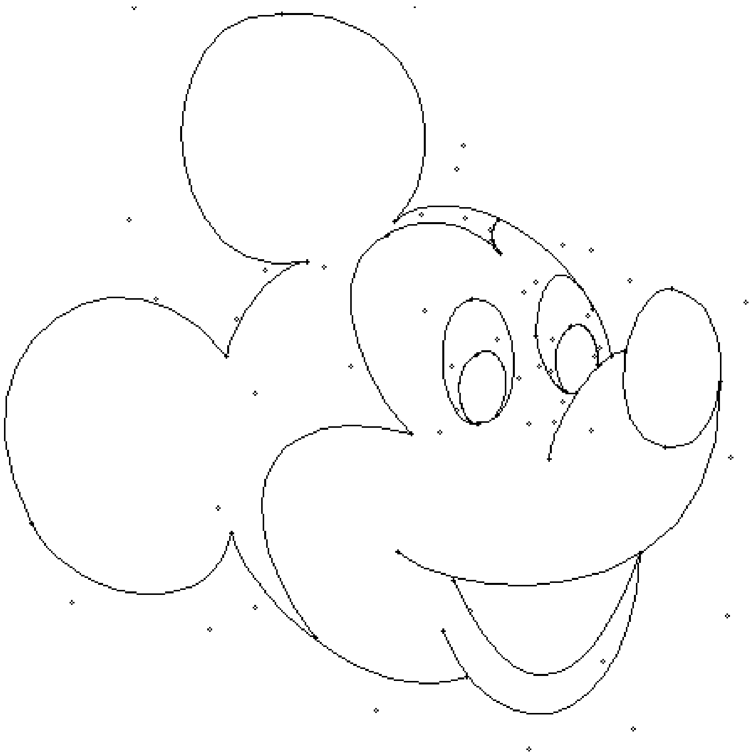
— Bei Auswertung des Splines treten keine zusätzlichen Rundungsfehler auf.

— Die Koeffizienten schließen, als Punkte betrachtet, den Funktionsgraphen ein. Graphik-Algorithmen wie Skalierung, Clipping (Abschneiden bei Fenstern, Verdeckung) können eine Vorentscheidung anhand der Koeffizienten treffen.

— Bei Koordinatentransformationen (Zoom) sind alle Koeffizienten gleich zu behandeln.

— Änderungen einzelner Koeffizienten haben nur lokale Änderungen des Splines zur Folge ($\rightarrow$ interaktiver Entwurf, CAD).

Beispiel 4.4.3 $\mathbf{R}^2$-Graphik aus kubischen Beziér-Splines mit Kontrollpunkten:



4.5 Lokale Spline-Approximationen

Beim Interpolationsspline ist zur Bestimmung des Splines ein Gleichungssystem zu lösen. Daher wirken sich einzelne (große) Funktionswerte auf den gesamten Spline aus. Beide Probleme entfallen, wenn man auf die exakte Interpolation verzichtet und explizite Formeln der Spline-Koeffizienten betrachtet. Den einfachsten Fall behandelt

Satz 4.5.1 *Die Funktion f sei Lipschitz-stetig auf $\mathbf{R}$ mit der Lipschitz-Konstanten L. Mit den Eigenschaften (4.4.8)-(4.4.10) der Basis-Funktionen B_j gilt für den Spline*

$$s(x) := \sum_{j=-1}^{n+1} f(x_j) B_j(x)$$

im Intervall $[a,b] = [x_0, x_n]$ die Konvergenzaussage

$$\|s - f\|_\infty \leq 2HL.$$

Beweis Für $x \in (x_j, x_{j+1}]$ ist $1 = \sum_{i=j-1}^{j+2} B_i(x)$ und daher

$$|s(x) - f(x)| = |\sum_{i=j-1}^{j+2} (f(x_i) - f(x))B_i(x)| \leq \max_{|x-y|\leq 2H} |f(x) - f(y)| \leq 2HL. \qquad \blacksquare$$

Allgemeiner werden Approximationen der Form

$$Qf := \sum_{i=-1}^{n+1} a_i(f) B_i \tag{4.5.1}$$

betrachtet für $f \in C[a-d, b+d]$ mit $d := \max\{|x_0 - x_{-3}|, |x_{n+3} - x_n|\}$. Diese heißen *lokale Spline-Approximation*, wenn $a_i(f)$ nur von Funktionswerten im Träger von B_i, also dem Intervall $[x_{i-2}, x_{i+2}]$, abhängt. Besonders interessant sind dabei lineare Funktionale a_i, z.B.,

$$a_i(f) = \sum_{j=1}^{\ell} \alpha_{ij} f(\xi_{ij}), \quad \xi_{ij} \in [x_{i-2}, x_{i+2}]. \tag{4.5.2}$$

Dann ist nämlich Q ein linearer Operator. Bei (4.5.2) gilt zunächst

$$|a_i(f)| \leq \sum_{j=1}^{\ell} |\alpha_{ij}| \sup\{|f(x)| : \; x \in [x_{i-2}, x_{i+2}]\} =: A_i \|f\|_{[x_{i-2}, x_{i+2}]}. \tag{4.5.3}$$

Daraus ergibt sich eine *lokale* Abschätzung für Qf, denn für $x \in [x_j, x_{j+1}]$ ist

$$|Qf(x)| \leq \sum_{i=j-1}^{j+2} |a_i(f)| B_i(x) \leq \max_{i=j-1}^{j+2} |a_i(f)| \leq \max_{i=j-1}^{j+2} A_i \|f\|_{[x_{j-3}, x_{j+4}]}. \tag{4.5.4}$$

Diese lokale Schranke ermöglicht nach einem einfachen Prinzip (vgl. Satz 3.6.2) folgende Konvergenzaussage.

Satz 4.5.2 *Zu einem erweiterten Gitter $\bar{\Delta}$ sei ein lokaler Approximationsoperator Q definiert, der (4.5.4) erfüllt. Wenn Q Polynome vom Grad m reproduziert, d.h., $Qp = p \,\forall p \in \Pi_m$ gilt, dann gilt für $f \in C^{m+1}[a-d, b+d]$ die Fehlerschranke*

$$\|f - Qf\|_{[a,b]} \leq c\Big(1 + \max_{i=-1}^{n+1} A_i\Big) H^{m+1} \|f^{(m+1)}\|_{[a-d, b+d]}$$

Beweis Nach Voraussetzung ist mit jedem beliebigen Polynom $p \in \Pi_m$

$$f - Qf = f - p - (Qf - p) = f - p - Q(f - p).$$

Mit (4.5.4) folgt daraus für $x \in (x_j, x_{j+1}]$, $0 \leq j < n$, dann

$$|f(x) - Qf(x)| \leq |f(x) - p(x)| + |Q(f-p)(x)| \leq (1 + \max_{i=-1}^{n+1} A_i) \|f - p\|_{[x_{j-3}, x_{j+4}]}.$$

Nun kann ein Polynom p so gewählt werden (z.B. durch Interpolation), daß

$$\|f - p\|_{[x_{j-3}, x_{j+4}]} \leq c H^{m+1} \|f^{(m+1)}\|_{[x_{j-3}, x_{j+4}]}. \qquad \blacksquare$$

Die optimale Fehleraussage für kubische Splines, $\mathcal{O}(H^4)$, folgt also aus diesem Satz, wenn Q kubische Polynome reproduziert und die Konstante $\max_i A_i$ nicht vom Gitter $\bar{\Delta}$ abhängt.

Beispiel 4.5.3 (Schoenberg, H^2-Approximation): Bei beliebigem Gitter sei

$$a_i(f) := f(\bar{x}_i),\ \bar{x}_i := \frac{1}{3}(x_{i-1} + x_i + x_{i+1}).$$

Q reproduziert lineare Funktionen. Für äquidistantes Gitter stimmt Q mit der Approximation aus Satz 4.5.1 überein, da $\bar{x}_i = x_i$.

Beispiel 4.5.4 (H^4-Approximation) Die Gestalt der Funktionale (4.5.2) für diese Approximation mit $\ell = 3$ ist sehr kompliziert bei allgemeinem Gitter, bei äquidistantem Gitter vereinfachen sich diese zu

$$a_i(f) := f(x_i) - \frac{h^2}{3} f[x_{i-1}, x_i, x_{i+1}],$$

(vgl. dazu (4.2.7)). Q reproduziert kubische Polynome.

Auch ohne Auflösung eines linearen Gleichungssystems (4.2.7) bzw. (4.4.2) kann also eine H^4-Approximation einer C^4-Funktion konstruiert werden, wenn auf die strenge Interpolationsforderung aus Satz 4.3.2 verzichtet wird.

5 Numerische Integration und Differentiation

In der Praxis stellt sich oft die Aufgabe, mit einer gegebenen Funktion f eine aufwendige, oder explizit nicht mögliche Operation durchzuführen. Als Ausweg kann man die Funktion durch eine "einfachere" ersetzen, für die die Operation (einfach) ausführbar ist. Bei *Integration und Differentiation* bietet sich die Polynom-Interpolierende in der Lagrange-Form aus §3.2 an:

$$\begin{aligned} f(x) &\cong & p_n(x) &= \sum_{j=0}^n L_j(x) f(x_j) & &\overset{?}{\Longrightarrow} \\ \int_a^b f(x)dx &\cong & \int_a^b p_n(x)dx &= \sum_{j=0}^n \Big(\int_a^b L_j(x)dx\Big) f(x_j) & &=: \sum_{j=0}^n \alpha_j f(x_j), \\ f'(0) &\cong & p_n'(0) &= \sum_{j=0}^n \ L_j'(0) \ f(x_j) & &=: \sum_{j=0}^n \delta_j f(x_j). \end{aligned} \tag{5.0.1}$$

Diese Näherungen für Integral- und Ableitungswerte sind also einfach Linearkombinationen von Funktionswerten. Daher werden jetzt solche Approximationen behandelt, zunächst für die Integration.

5.1 Quadratur

Bei nicht stetig differenzierbaren Integranden ist es für Fehleraussagen vorteilhaft, diese als Produkt einer glatten Funktion und einer speziellen, schwierigen *Gewichtsfunktion* $g(x) > 0$ (z.B. $\sqrt{x}, 1/\sqrt{x}, ..$) aufzufassen. Im folgenden wird daher die Integration mit einer festen Gewichtsfunktion g betrachtet.

Definition 5.1.1 *Falls $\int_a^b f(x)g(x)dx$ existiert, heißt*

$$\int_a^b f(x)g(x)dx = \sum_{j=0}^n \alpha_j f(x_j) + R_n(f) \tag{5.1.1}$$

eine Quadraturformel *mit den* Gewichten α_j *zu den* Knoten

$$a \le x_0 < x_1 < \ldots < x_n \le b.$$

Dabei ist $R_n(f)$ der Quadraturfehler *bzw. das Restglied und man sagt, die Formel (5.1.1) besitze die* Ordnung $m \in \mathbf{N}$, *wenn gilt*

$$R_n(p) = 0 \quad \forall p \in \Pi_{m-1}.$$

Für die speziellen Quadraturformeln, die sich wie in (5.0.1) durch Integration des Interpolationspolynoms ergeben, sind die Gewichte

$$\alpha_j = \int_a^b L_j(x)g(x)dx, \ j = 0, \ldots, n. \tag{5.1.2}$$

Aus der Darstellung (3.3.11) des Interpolationsfehlers $r_n = f - p_n$ folgt

$$\begin{aligned} R_n(f) &= \int_a^b f(x)g(x)dx - \sum_{j=0}^{n} \alpha_j f(x_j) = \int_a^b r_n(x)g(x)dx \\ &= \int_a^b \omega_{n+1}(x) f[x_0,\ldots,x_n,x] g(x)dx. \end{aligned} \tag{5.1.3}$$

Für $f \in C^{n+1}[a,b]$ führt eine Betragsabschätzung auf die Schranke

$$|R_n(f)| \le \int_a^b |\omega_{n+1}(x)| g(x)dx \ \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty. \tag{5.1.4}$$

Damit folgt insbesondere der

Satz 5.1.2 *Die Ordnung einer* interpolatorischen *Quadraturformel (5.1.1), (5.1.2) ist mindestens* $n+1$.

Die Betragsschranke kann noch verschärft werden, wenn das Knotenpolynom ω_{n+1} nur ein Vorzeichen besitzt, d.h., wenn

$$\omega_{n+1}(x) = (x-x_0)\cdots(x-x_n) \ge 0 \ \forall x \in [a,b] \quad (\text{bzw. } \le 0\ \forall ..). \tag{5.1.5}$$

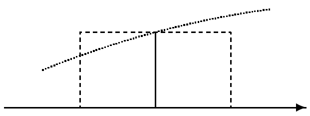
Dann gilt nach dem Mittelwertsatz der Integralrechnung mit $\xi_1, \xi_2 \in (a,b)$, sogar

$$\begin{aligned} R_n(f) &= \int_a^b \omega_{n+1}(x)g(x)dx \cdot f[x_0,\ldots,x_n,\xi_1] \\ &= \varrho_n \cdot f^{(n+1)}(\xi_2), \quad \varrho_n := \frac{1}{(n+1)!} \int_a^b \omega_{n+1}(x)g(x)dx. \end{aligned} \tag{5.1.6}$$

5.2 Newton-Cotes-Formeln

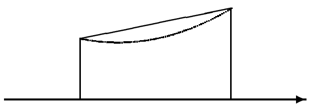
Speziell für $g \equiv 1$ und äquidistante Knoten $x_j = x_0 + jh, j = 0,\ldots,n$, die die Randpunkte enthalten (d.h. $x_0 = a,\ h = (b-a)/n$, "abgeschlossene Formeln") oder ausschließen ($a < x_0,\ x_n < b$, "offene Formeln") ergeben sich die Newton-Cotes-Formeln. Die einfachsten Spezialfälle sind:

Rechteckregel: $n = 0$, offene Formel, Ordnung 2:



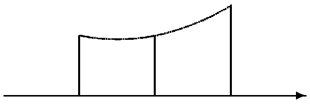
$$\int_a^b f(x)dx = (b-a)\, f\Big(\frac{a+b}{2}\Big) + \frac{(b-a)^3}{24} f''(\xi). \tag{5.2.1}$$

Trapezregel: $n = 1$, abgeschlossene Formel, Ordnung 2:



$$\int_a^b f(x)dx = \frac{b-a}{2}\Big[f(a) + f(b)\Big] - \frac{(b-a)^3}{12} f''(\xi). \tag{5.2.2}$$

Simpsonregel: $n = 2$, abgeschlossene Formel, Ordnung 4:



$$\int_a^b f(x)dx = \frac{b-a}{6}\Big[f(a) + 4f(\frac{a+b}{2}) + f(b)\Big] - \frac{(b-a)^5}{2880}f^{(4)}(\xi). \tag{5.2.3}$$

In diesen Formeln steht ξ jeweils für eine unbekannte Zwischenstelle in (a, b). Eine Tabelle der Gewichte α_j und der Fehlerkoeffizienten ϱ_n für weitere Ordnungen folgt weiter unten. Zunächst wird nachgeprüft, daß bei der Rechteck- und Simpsonregel die Ordnung tatsächlich $n + 2$ ist, statt $n + 1$, wie nach Satz 5.1.2 zu erwarten.

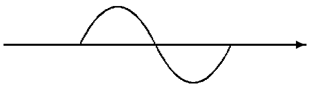
Beweis von (5.2.3): Für $a = 0$ (oBdA) sind die Lagrangepolynome zu $x_0 = 0,\ x_1 = h,\ x_2 = 2h$:

$$L_0(x) = \frac{(x-h)(x-2h)}{2h^2},\ L_1(x) = \frac{x(x-2h)}{-h^2},\ L_2(x) = \frac{x(x-h)}{2h^2} \quad \in \Pi_2.$$

Daraus berechnen sich die Gewichte

$$\alpha_0 = \int_0^{2h} L_0(x)dx = \frac{1}{2h^2}[\frac{1}{3}x^3 - \frac{3h}{2}x^2 + 2h^2x]_0^{2h} = \frac{h}{3} = \alpha_2, \quad \alpha_1 = \int_0^{2h} L_1(x)dx = \frac{4}{3}h.$$

Die Fehlerdarstellungen (5.1.4) bzw. (5.1.6) stimmen allerdings noch nicht mit der in (5.2.3) überein. Dazu ist die Zusatzüberlegung erforderlich, daß wegen



$$\int_0^{2h} \omega_3(x)dx = \int_0^{2h} x(x-h)(x-2h)dx = 0 \tag{5.2.4}$$

sogar kubische Polynome exakt integriert werden! Dazu wird eine zusätzliche Interpolationsbedingung $p'(x_1) = f'(x_1)$ betrachtet. Nach (3.3.7) folgt dann mit $p_2 = f(x_0)L_0 + f(x_1)L_1 + f(x_2)L_2$ die Darstellung $p_3(x) = p_2(x) + \omega_3(x)f[x_0, x_1, x_1, x_2]$ und

$$\int_0^{2h} p_3(x)dx = \int_0^{2h} p_2(x)dx + \underbrace{\int_0^{2h} \omega_3(x)dx}_{=0} f[x_0, x_1, x_1, x_2] = \alpha_0 f(x_0) + \alpha_1 f(x_1) + \alpha_2 f(x_2).$$

Daher sind die Integrale über p_2 und p_3 identisch und es gilt (5.1.3) sogar mit $n = 3$ und auch (5.1.6) mit $\varrho_3 = -\frac{1}{90}h^5$, da

$$\omega_4(x) = x(x-h)^2(x-2h) \le 0 \ \forall x \in [0, 2h]. \qquad \blacksquare$$

Dieser Beweis überträgt sich auf alle Newton-Cotes-Formeln mit geradem n, die Ordnung ist für diese $n + 2$ statt $n + 1$.

Die folgende Tabelle enthält die Gewichte und Fehlerkonstanten der Newton-Cotes-Formeln. Da die Gewichte rational sind, erfolgt die Angabe in der Form

$$\int_a^b f(x)dx = \frac{b-a}{\gamma}\sum_{j=0}^{n}\beta_j f(x_j) + c\Big(\frac{b-a}{n}\Big)^{m+1} f^{(m)}(\xi),\ \xi \in (a, b). \tag{5.2.5}$$

Abgeschlossene Newton-Cotes-Formeln, $x_j = a + jh,\ j = 0, \ldots, n,\ h = \frac{b-a}{n}$.

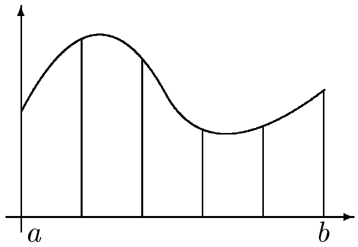
n	γ	β_0	β_1	β_2	β_3	β_4	c	m
1	2	1	1				-1/12	2
2	6	1	4	1			-1/90	4
3	8	1	3	3	1		-3/80	4
4	90	7	32	12	32	7	-8/945	6
5	288	19	75	50	50	75	-275/12096	6
6	840	41	216	27	272	27	-9/1400	8
7	17280	751	3577	1323	2989	2989	-8183/518400	8
8	28350	989	5888	-928	10496	-4540	-2368/467775	10

Die fehlenden Koeffizienten $\beta_5, \ldots, \beta_8$ sind symmetrisch zu ergänzen. Die Gewichte für $n \geq 8$ sind nicht mehr positiv. Dies erhöht etwas die Anfälligkeit der Formeln für Rundungsfehler.

Für die Konvergenz ($n \to \infty$) der Quadraturformel (5.2.5) gilt das gleiche wie bei der Polynom-Interpolation. Das Anwachsen der höheren Ableitungen $f^{(m)},\ m = 2(\lfloor \frac{n}{2} \rfloor + 1)$, kann die Konvergenz zerstören. Andererseits geht der Fehler

$$R_n(f) = c\left(\frac{b-a}{n}\right)^{m+1} f^{(m)}(\xi)$$

sehr schnell gegen null für $(b-a) \to 0$. Dies kann durch Unterteilung des Gesamtintervalls ausgenutzt werden. Dabei wird eine Quadraturformel fester Ordnung *iteriert* angewendet. Bei der Trapezregel (5.2.2), z.B., ergibt sich mit Teilintervallen $[x_{i-1}, x_i],\ x_j = a + jh,\ j = 0, \ldots, n,\ h := (b-a)/n$, die Näherung



$$\begin{aligned}\int_a^b f(x)dx &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \cong \sum_{i=1}^{n} \frac{h}{2}\Big[f(x_i) + f(x_{i-1})\Big] \\ &= \frac{h}{2}\Big[f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)\Big].\end{aligned}$$

Für diese Näherung und die zur Simpsonregel gilt folgende Fehleraussage.

Satz 5.2.1 *Für $n \in \mathbf{N}$ sei $h := (b-a)/n,\ x_i = a + ih,\ f_i := f(x_i),\ i = 0, \ldots, n$. Dann gilt mit einer Zwischenstelle $\xi \in (a, b)$ für die*

a) iterierte Trapezregel bei $f \in C^2[a, b]$:

$$\int_a^b f(x)dx = \frac{h}{2}\Big(f_0 + 2f_1 + \ldots + 2f_{n-1} + f_n\Big) - \frac{b-a}{12}h^2 f''(\xi), \tag{5.2.6}$$

b) iterierte Simpsonregel bei $f \in C^4[a, b]$, n gerade:

$$\int_a^b f(x)dx = \frac{h}{3}\Big(f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 4f_{n-1} + f_n\Big) - \frac{b-a}{180}h^4 f^{(4)}(\xi). \tag{5.2.7}$$

Beweis Die Beweise von a) und b) verlaufen analog. Bei b) wird die Formel (5.2.3) über die $\frac{n}{2} =: k$ Intervalle $[x_{2i-2}, x_{2i}]$ summiert. Beim Restglied erhält man dabei mit $\xi_i \in (x_{2i-2}, x_{2i})$

$$-\frac{1}{2880}\sum_{i=1}^{k}(x_{2i}-x_{2i-2})^5 f^{(4)}(\xi_i) = -\frac{32}{2880}h^5\sum_{i=1}^{k} f^{(4)}(\xi_i) = -\frac{b-a}{180}h^4 f^{(4)}(\xi). \qquad \blacksquare$$

An diesen Darstellungen sieht man, daß es sinnvoll ist, in (5.2.5) als Ordnung der Formel die Zahl m und nicht $m+1$ anzugeben. Ein praktischer Vorteil der iterierten Trapez- und Simpsonregel ist, daß bei einer Verdopplung der Intervallzahl alle alten Funktionswerte verwendbar sind.

Der wesentliche Rechenaufwand bei beiden iterierten Formeln fällt bei den $n+1$ Funktionsauswertungen $f_0, \ldots, f_n$ an. Bei einem Vergleich der Fehler sieht man, daß

$$R_n(f) \sim \frac{1}{n^2} \text{ (Trapezregel)} \quad \text{bzw.} \quad R_n(f) \sim \frac{1}{n^4} \text{ (Simpsonregel)}$$

gilt bei einem genügend oft differenzierbaren Integranden f. Eine Verdopplung der Zahl n der Funktionsauswertungen verkleinert den Fehler $R_n(f)$ bei der Trapezregel um den Faktor $\frac{1}{4}$, bei der Simpsonregel dagegen um den Faktor $\frac{1}{16}$ bei vergleichbarem Aufwand. Daher sind bei *glatten* Integranden Formeln hoher Ordnung günstiger.

Beispiel 5.2.2 $\int_1^2 \frac{dx}{x} = \ln 2 = 0.6931472...$

h	Trapez	Fehler	Simpson	Fehler
1	0.75	0.056		
1/2	0.7083..	0.015..	0.694..	0.00129..
1/4	0.6970..	0.0038..	0.69325..	0.00010..
1/8	0.6941..	0.00097..	0.69315..	0.000007..
1/16	0.69339..	0.0002..	0.6931476..	0.0000004..

Diese Beobachtung führt natürlich sofort auf das Ziel, mit möglichst wenig Funktionsauswertungen eine möglichst hohe Ordnung zu erzielen. Mit der folgenden Klasse von Quadraturformeln wird dieses Ziel erreicht.

5.3 Gauß-Quadratur

Bei den Newton-Cotes-Formeln hatte man für geraden Polynomgrad n eine um eins erhöhte Konvergenzordnung aufgrund der Eigenschaft

$$\int_a^b \omega_{n+1}(x)dx = 0, \quad \omega_{n+1} = (x-x_0)\cdots(x-x_n),$$

die auf die Symmetrie der Knoten x_i zurückzuführen war. Durch geeignete Wahl der Knoten x_i kann die Ordnung sogar verdoppelt werden, es läßt sich erreichen, daß für den Quadraturfehler (5.1.1) gilt

$$R_n(f) = 0 \quad \forall f \in \Pi_{2n+1}. \tag{5.3.1}$$

Es sei nun $f \in \Pi_{2n+1}$ und $p_n(x) = \sum_{i=0}^n f(x_i)L_i(x)$ das zugehörige Interpolationspolynom in Π_n. Da dann gilt $f = p_n + \omega_{n+1} \cdot q_n$ mit $q_n \in \Pi_n$, entspricht die Forderung (5.3.1) der Bedingung

$$R_n(f) = \int_a^b f(x)g(x)dx - \int_a^b p_n(x)g(x)dx = \int_a^b \omega_{n+1}(x)q_n(x)g(x)dx \stackrel{!}{=} 0.$$

Daher ist (5.3.1) offensichtlich genau dann erfüllt, wenn

$$\int_a^b \omega_{n+1}(x)q(x)g(x)dx = 0 \quad \forall q \in \Pi_n. \tag{5.3.2}$$

Für $g > 0$ in (a, b) stellt die Bilinearform

$$(u, v)_g := \int_a^b u(x)v(x)\, g(x)dx \tag{5.3.3}$$

ein Innenprodukt in $C[a, b]$ dar. Die Forderung (5.3.2) bedeutet daher

$$\omega_{n+1} \perp_g \Pi_n \quad \iff \quad (\omega_{n+1}, q)_g = 0 \ \forall q \in \Pi_n, \tag{5.3.4}$$

es muß ω_{n+1} also g-orthogonal zu allen Polynomen vom Grad n gewählt werden!

Eine Familie von Orthogonalpolynomen ω_k, $k \in \mathbf{N}$, zum Innenprodukt (5.3.3) kann mit Hilfe des E. Schmidtschen Orthogonalisierungsverfahrens aus der Basis $\{1, x, x^2, \ldots\}$ konstruiert werden. Diese Orthogonalpolynome besitzen glücklicherweise nur reelle Nullstellen in (a, b), die als Knoten $x_0, \ldots, x_n$ einer Quadraturformel verwendet werden können. Die wesentlichen Aussagen werden im nächsten Satz zusammengefaßt, später folgt ein Überblick über einige wichtige Klassen von Orthogonal-Polynomen.

Satz 5.3.1 *Die Stützstellen x_i, $i = 0, \ldots, n$, der Quadraturformel (5.1.1), (5.1.2) seien als Nullstellen des Orthogonalpolynoms vom Grad $n+1$ zur Gewichtsfunktion $g \in C[a, b]$, $g > 0$ in (a, b), gewählt, d.h. $\omega_{n+1} \in \Pi_{n+1}$, $\omega_{n+1} \perp_g \Pi_n$. Dann besitzt diese* Gauß-sche Quadraturformel *die* Ordnung $2n + 2$, *d.h. es ist $R_n(f) = 0 \ \forall f \in \Pi_{2n+1}$. Für $f \in C^{2n+2}[a, b]$ gilt mit $\xi \in (a, b)$,*

$$R_n(f) = \int_a^b f(x)g(x)dx - \sum_{i=0}^n \alpha_i f(x_i) = \frac{1}{(2n+2)!} \int_a^b \omega_{n+1}^2(x)g(x)dx\, f^{(2n+2)}(\xi). \tag{5.3.5}$$

Bemerkung: Gegenüber den Newton-Cotes-Formeln erhält man also ungefähr die doppelte Konvergenzordnung ohne Mehraufwand. Bei der Verwendung in iterierten Formeln besitzen die Gaußformeln aber praktische Nachteile (vgl. §5.4).

Beweis Hier ist nur noch zu zeigen, daß das Restglied die von (5.1.6) abweichende Form hat. Analog zum Beweis bei der Simpsonregel (5.2.3) werden zusätzliche Interpolationsbedingungen benutzt: $f'(x_0), \ldots, f'(x_n)$. Im Vergleich zum einfachen Polynom $p_n \in \Pi_n$ mit $p_n(x_i) = f(x_i)$, $i = 0, \ldots, n$ hat die Newtonform dieses Interpolationspolynoms $p_{2n+1} \in \Pi_{2n+1}$ mit

$$p_{2n+1}(x_i) = f(x_i), \quad p'_{2n+1}(x_i) = f'(x_i), \ i = 0, \ldots, n,$$

nach (3.3.7) die Gestalt

$$\begin{aligned} p_{2n+1}(x) &= p_n(x) + \omega_{n+1}(x) f[x_0, .., x_n, x_0] + \omega_{n+1}(x)(x-x_0) f[x_0, .., x_n, x_0, x_1] + \\ & \quad \ldots + \omega_{n+1}(x)(x-x_0)\cdots(x-x_{n-1}) f[x_0, \ldots, x_n, x_0, \ldots, x_n]. \end{aligned}$$

Außerdem ist nach (3.3.11)

$$f(x) = p_{2n+1}(x) + \omega_{n+1}^2(x) f[x_0, \ldots, x_n, x_0, \ldots, x_n, x]. \tag{5.3.6}$$

Aufgrund der Orthogonalität (5.3.4) gilt aber

$$\int_a^b \omega_{n+1}(x)(x-x_0)\cdots(x-x_k) g(x) dx = 0 \ \forall k < n, \text{ d.h.,}$$

$$\int_a^b p_{2n+1}(x) g(x) dx = \int_a^b p_n(x) g(x) dx = \sum_{i=0}^n \alpha_i f(x_i). \tag{5.3.7}$$

Aus (5.3.6) folgt die Aussage (5.3.5), da die Fehlerformel (5.1.6) wegen $\omega_{n+1}^2 \geq 0$ jetzt auf p_{2n+1} angewendet werden kann. ■

Bemerkung: In (5.3.7) zeigt sich, daß die Gewichte β_i in der zu p_{2n+1} gehörigen erweiterten Quadraturformel $\int_a^b p_{2n+1}(x) g(x) dx = \sum_{i=0}^n [\alpha_i f(x_i) + \beta_i f'(x_i)]$ alle verschwinden: $\beta_i \equiv 0$.

Zum Vergleich der verschiedenen Methoden dient folgendes

Beispiel 5.3.2 $\int_{-1}^1 e^x dx = e^1 - e^{-1} = 2 \sinh 1 = 2.3504024$. Näherungen

a) Trapezregel, Ordnung 2, 2 Schritte, 3 Punkte:

$$\frac{b-a}{4}[f(-1) + 2f(0) + f(1)] = 2\cosh^2 \frac{1}{2} = 2.54308.., \quad \text{Fehler} = 0.193..$$

b) Simpsonregel, Ordnung 4, 3 Punkte:

$$\frac{b-a}{6}[f(-1) + 4f(0) + f(1)] = \frac{1}{3}[4 + 2\cosh 1] = 2.36205.., \quad \text{Fehler} = 0.012..$$

c) Gaußquadratur mit 2 Punkten, Ordnung 4. Bestimmung der Orthogonalpolynome: $\omega_0 = P_0 \equiv 1$, $\omega_1(x) = P_1(x) = x \perp \omega_0$, Ansatz $\omega_2 = x^2 + ax + b \perp \Pi_1$, d.h.

$$\alpha) \qquad 0 \stackrel{!}{=} \int_{-1}^1 [x^2 + ax + b] dx = \Big[\frac{x^3}{3} + a\frac{x^2}{2} + bx\Big]_{-1}^1 = \frac{2}{3} + 2b \ \Rightarrow b = -\frac{1}{3},$$

$$\beta) \qquad 0 \stackrel{!}{=} \int_{-1}^1 x[x^2 + ax + b] dx = \Big[\frac{x^4}{4} + a\frac{x^3}{3} + b\frac{x^2}{2}\Big]_{-1}^1 = \frac{2}{3}a \ \Rightarrow a = 0.$$

Dies ergibt $\omega_2(x) = x^2 - \frac{1}{3}$ mit den Nullstellen $x_0 = -\frac{1}{\sqrt{3}}$, $x_1 = \frac{1}{\sqrt{3}} = 0.57735..$ Aus Symmetriegründen ist $\alpha_0 = \alpha_1 = 1$. Die Gaußnäherung hat daher den Wert

$$f(x_0) + f(x_1) = 2\cosh(x_1) = 2.342696.. \quad \text{Fehler} = -0.0077..$$

Je nach Art der im Integranden zu berücksichtigenden Singularität (→ Gewichtsfunktion) ergeben sich verschiedene Polynomfamilien (vgl. folgende Tabelle), in denen jeweils auch Drei-Term-Rekursionen gelten (vgl. §3.4). Die wichtigste Familie zur Gewichtsfunktion $g \equiv 1$ ist die der Legendrepolynome. Zur Vereinfachung wird meist das Standardintervall $[-1,1]$ zugrundegelegt, durch Variablensubstitution sind andere Intervalle darauf zurückführbar (wie (3.6.7)).

Polynom-Orthogonalfamilien nach Satz 5.3.1:

Intervall	Gewichtsfunkt.	Bezeichnung	Rekursionsformel
$[-1,1]$	$g \equiv 1$	Legendre-Pol.	$P_{n+1} = \frac{2n+1}{n+1} x P_n - \frac{n}{n+1} P_{n-1},\ P_0 = 1,\ P_1 = x$
$[-1,1]$	$g(x) = \sqrt{1-x^2}$	Tscheby. 2.Art	$U_{n+1} = 2xU_n - U_{n-1},\ U_0 = 1,\ U_1 = 2x$
$(-1,1)$	$g(x) = \frac{1}{\sqrt{1-x^2}}$	Tscheby. 1.Art	$T_{n+1} = 2xT_n - T_{n-1},\ T_0 = 1,\ T_1 = x$ Gewichte konstant, $\alpha_j = \frac{\pi}{n}$
$[0,\infty)$	$g(x) = e^{-x}$	Laguerre-Pol.	$L_{n+1} = \frac{2n+1-x}{n+1} L_n - \frac{n}{n+1} L_{n-1}$ $L_0 = 1,\ L_1 = 1-x$
$(-\infty,\infty)$	$g(x) = e^{-x^2}$	Hermite-Pol.	$H_{n+1} = 2xH_n - 2nH_{n-1},\ H_0 = 1,\ H_1 = 2x$

Über die angegebenen Polynomfamilien gibt es eine umfangreiche Literatur v.a. im Zusammenhang mit Reihenentwicklungen für Lösungen von Differentialgleichungen (z.B. Courant-Hilbert). Quadratur-Knoten und -Gewichte sind tabelliert, die Berechnung ist bei Stoer, §3.5, besprochen.

Eine explizite Darstellung von Orthogonalpolynomen ist oft mit Hilfe von Rodriguez-Formeln möglich. Die Legendre-Polynome, z.B., haben, bis auf Konstanten, die Gestalt

$$P_n(x) = \frac{n!}{(2n)!} \frac{d^n}{dx^n}(x^2-1)^n, \quad n = 0,1,\dots\ . \tag{5.3.8}$$

Diese Formel ergibt sich aus den Orthogonalitätsbedingungen (5.3.2) durch partielle Integration. Aus dieser Darstellung (5.3.8) folgt übrigens direkt die Existenz von n reellen Nullstellen in $(-1,1)$ nach dem Satz von Rolle.

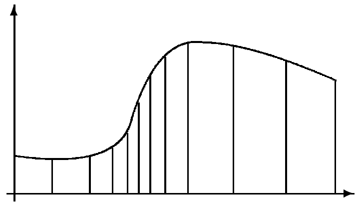
5.4 Adaptive Integration

Bisher wurde überhaupt nicht diskutiert, wie im konkreten Fall, bei gegebenem Integranden, das Quadraturverfahren und dessen Parameter (Schrittweite h, Ordnung) zu wählen sind. Im allgemeinen will man den Integralwert mit einer bestimmten Genauigkeit ε bei minimalem Aufwand berechnen. Falls über den Integranden genügend analytische Information vorhanden ist, kann im Prinzip aus der Restglieddarstellung (5.1.4) die Ordnung der Formel und die Zahl der Teilintervalle so bestimmt werden, daß die Fehlerschranke den Wert ε nicht überschreitet.

Diese aufwendige Methode läßt sich umgehen, wenn die Quadraturformel mit einer Fehler(ab)schätzung verbunden wird. Dies soll bei der iterierten Trapezregel besprochen werden. Wenn das Intervall $[a,b]$ in n Teile $[x_{i-1},x_i]$ zerlegt ist, gilt in jedem Teil mit $\xi_i \in (x_{i-1},x_i)$:

$$\int_{x_{i-1}}^{x_i} f(x)dx - \frac{1}{2}(x_i - x_{i-1})[f(x_{i-1}) + f(x_i)] = -\frac{1}{12}(x_i - x_{i-1})^3 f''(\xi_i) =: R[x_{i-1},x_i]. \tag{5.4.1}$$

Der lokal auftretende Fehler setzt sich also aus der aktuellen Teilintervallänge und dem lokalen Wert der zweiten Ableitung zusammen. Eine effiziente Strategie bei der Quadratur sollte den



vorgegebenen Fehler mit möglichst wenig Funktionsauswertungen erreichen. Im Bereich kleiner Ableitungen kann dazu mit großen Schrittweiten vorgegangen werden, während bei großen Ableitungswerten viel feinere Unterteilungen nötig sind.

Eine gute Strategie in diesem Sinn ist die der *proportionalen Gleichverteilung* des Fehlers

$$\Big|R[x_{i-1},x_i]\Big| \overset{!}{\leq} \frac{x_i - x_{i-1}}{b-a}\varepsilon, \quad i = 1,\dots,n. \tag{5.4.2}$$

Dann gilt nämlich

$$\left|\int_a^b f(x)dx - \sum_{i=1}^n \frac{x_i - x_{i-1}}{2}[f(x_{i-1}) + f(x_i)]\right| \leq \sum_{i=1}^n |R[x_{i-1},x_i]| \leq \sum_{i=1}^n \frac{x_i - x_{i-1}}{b-a}\varepsilon = \varepsilon. \tag{5.4.3}$$

Zur Anwendung der Strategie (5.4.2) sind zwei Fragen zu klären:

a) Wie erhält man eine Abschätzung des lokalen Fehlers $R[x_{i-1},x_i]$?

b) Wie konstruiert man damit eine Unterteilung, die (5.4.2) erfüllt?

Zu a): Außer bei einfachen Integranden, bei denen $\|f''\|_{[x_{i-1},x_i]}$ abgeschätzt werden kann (evtl. mit Intervallrechnung), begnügt man sich mit einer *Schätzung* des lokalen Fehlers $R[x_{i-1},x_i]$. So gilt, z.B., mit $h_i := x_i - x_{i-1}$ für ein $\zeta_i \in (x_{i-1},x_i)$ die Aussage

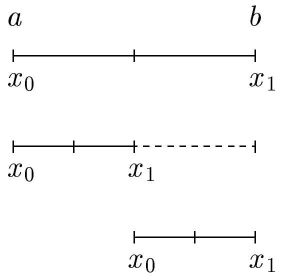
$$f(x_{i-1}) - 2f\Big(\frac{x_{i-1}+x_i}{2}\Big) + f(x_i) = \frac{1}{4}h_i^2 f''(\zeta_i) \cong -\frac{3}{h_i}R[x_{i-1},x_i]. \tag{5.4.4}$$

Dies folgt aus (3.3.10) oder durch Taylorentwicklung um $\frac{1}{2}(x_{i-1}+x_i)$. Mit der Approximation (5.4.4) wird das Kriterium (5.4.2) *ersetzt* durch

$$\Big|f(x_{i-1}) - 2f\Big(\frac{x_{i-1}+x_i}{2}\Big) + f(x_i)\Big| \overset{!}{\leq} \frac{3\varepsilon}{b-a}. \tag{5.4.5}$$

Zu b): Die Gitterkonstruktion zur Einhaltung von (5.4.5) kann ganz einfach über folgendes adaptive Verfahren geschehen:

- wähle $x_0 := a,\ x_1 := b$;
- Falls (5.4.5) verletzt ist, halbiere das Intervall, setze $x_1 := (x_0 + x_1)/2$;
- Falls (5.4.5) gilt, akzeptiere den Integralwert für $\int_{x_0}^{x_1}$ und fahre wie oben fort mit der Integration im Restintervall $\int_{x_1}^{b}$, d.h. mit $x_0 := x_1,\ x_1 := b$.



Der folgende Programmteil beschreibt diesen einfachen Algorithmus.

Algorithmus 5.4.1 Einfache adaptive Quadratur

$x_0 := a;\ f0 := f(x0);\ x1 := b;\ f1 := f(x1);$

$w := 0;\ e := 3\varepsilon/(b-a);$

repeat $xm := 0.5 * (x0 + x1);\ fm := f(xm);$

  if $abs(f0 - 2 * fm + f1) <= e$ then {akzeptieren}

  begin $w := w + 0.5 * (x1 - x0) * (f0 + f1);$

    $x0 := x1;\ f0 := f1;\ x1 := b;\ f1 := f(x1)$

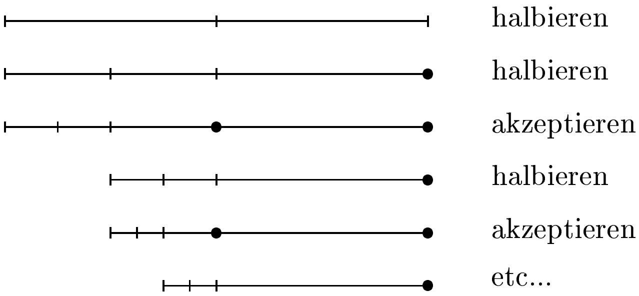
  end else {halbieren}

  begin $x1 := xm;\ f1 := fm$ end

until $x0 >= b;$

In der Praxis sind Funktionsauswertungen zu kostbar, um sie im Falle eines nicht akzeptierten Teilintegrals zu verwerfen. Durch eine Einschränkung der Teilintervallängen auf die Werte $(b-a)2^{-j}, j \in \mathbf{N}$, und einfache Zusatzbedingungen kann man erreichen, daß alle einmal *berechneten* Funktionswerte später auch *verwendet* werden. Die Verwaltung der gespeicherten Funktionswerte läßt sich elegant in einem *Keller*speicher (Stapel, stack, FILO) realisieren.

Beispiel:



Schwarz markierte
Gitterwerte stehen im Keller

Professionelle Programme steuern lokal außer der Schrittweite auch die Ordnung der Formel.

Demo-Beispiel: Adaptive Integration von

$$\int_0^1 e^x\,dx, \quad \int_{-1}^1 \Big(\sqrt{|x^2-0.25|} - \frac{1}{3}\Big)\,dx, \quad \int_0^1 \sin\frac{1}{x^2+0.05}\,dx.$$

5.5 Richardson-Extrapolation, Romberg-Verfahren

Im letzten Abschnitt wurde zum ersten Mal die theoretische Kenntnis des Fehlerverhaltens von Quadraturformeln praktisch ausgenutzt. Die folgende Methode verwendet diese in noch viel stärkerem Maße zu einer erheblichen Verbesserung der Konvergenz. Dazu sei noch einmal an die Werte des Beispiels aus §5.2 erinnert:

Beispiel 5.5.1 Iterierte Trapezregel zu $W := \int_1^2 \frac{1}{x}\,dx = \ln 2 = 0.6931472...$

h	1	1/2	1/4	1/8	1/16
T_h	0.75	0.70833	0.69702	0.694122	0.693391
Fehler	0.0568	0.01518	0.00387	0.000975	0.000244
Fehler-Quotient		3.74	3.92	3.99	3.996

Der Quotient aufeinanderfolgender Fehler konvergiert offensichtlich gegen 4 für $h \to 0$:

$$\frac{W - T_h}{W - T_{h/2}} \to 4, \quad \text{d.h.,} \quad W - T_h - ah^2 = o(h^2).$$

Dabei ist a eine Konstante und $o(h^k)$, $(h \to 0)$, bezeichnet eine Funktion, die schneller gegen Null geht als h^k. Bei Vernachlässigung dieses Terms in der letzten Gleichung kann die unbekannte Konstante a bei Kenntnis von zwei Trapeznäherungen eliminiert werden in folgender Weise:

$$\begin{aligned} a \doteq \frac{1}{h^2}(W - T_h) \quad \Rightarrow \quad W &\doteq T_{h/2} + a\frac{h^2}{4} \doteq T_{h/2} + \frac{h^2}{4}\frac{1}{h^2}(W - T_h), \text{ bzw.} \\ W &\doteq \frac{4}{3}T_{h/2} - \frac{1}{3}T_h =: \widehat{T}_{h/2}. \end{aligned} \tag{5.5.1}$$

Die auf diese Weise neu kombinierten Werte zum letzten Zahlenbeispiel lauten

$h =$ 1	1/2	1/4	1/8	1/16
$\widehat{T}_h =$	0.69444	0.6932	0.69316	0.693147

Diese besitzen ca. die doppelte Anzahl von gültigen Ziffern wie die ursprünglichen Trapezwerte. Der Erfolg des Vorgehens beruht auf folgender Eigenschaft.

Definition 5.5.2 *Die Größe T_h, $0 < h \in \mathbf{R}$, besitzt eine* asymptotische Entwicklung *(nach Potenzen von h^k), wenn $q \in \mathbf{N}$ und von h unabhängige Koeffizienten $a_j \in \mathbf{R}$ existieren so, daß gilt*

$$T_h = W + a_1 h^k + a_2 h^{2k} + \ldots + a_q h^{qk} + \mathcal{O}(h^{qk+1}), \; h \to 0. \tag{5.5.2}$$

Bei der Trapezregel ($k = 2$, s.u.) verändert die Linearkombination (5.5.1) die Entwicklung (5.5.2):

$$\begin{aligned} \widehat{T}_{h/2} = \frac{4}{3}T_{h/2} - \frac{1}{3}T_h &= W + a_1 h^2\left(\frac{4}{3}\frac{1}{4} - \frac{1}{3}\right) + a_2 h^4\left(\frac{4}{3}\frac{1}{16} - \frac{1}{3}\right) + \ldots \\ &= W - \frac{1}{4}a_2 h^4 + \ldots \end{aligned}$$

In der Entwicklung wurde also der h^2-Fehleranteil eliminiert, $\widehat{T}_h$ ist jetzt eine Näherung der Ordnung 4. Genauer gilt $\widehat{T}_{h/2} = W + \hat{a}_2(\frac{h}{2})^4 + \ldots$. Daher kann das *Extrapolationsverfahren* von oben noch einmal auf $\widehat{T}$ angewendet werden. Bevor diese Methode systematischer untersucht wird, sei zunächst die Aussage (5.5.2) für die Trapezregel formuliert.

Satz 5.5.3 (Euler-McLaurin-Summenformel) *Für $f \in C^{2q+2}[a, b]$ besitzt die iterierte Trapezregel (5.2.6) folgende h^2-Entwicklung, $h = (b - a)/n$,*

$$\begin{aligned} T_h &= \frac{h}{2}[f_0 + 2f_1 + \ldots + 2f_{n-1} + f_n] = \int_a^b f(x)dx \\ &\quad + \sum_{j=1}^{q} b_j h^{2j}\left[f^{(2j-1)}(b) - f^{(2j-1)}(a)\right] + c_{2q+2}h^{2q+2}f^{(2q+2)}(\xi), \; \xi \in (a, b). \end{aligned} \tag{5.5.3}$$

Beweis (-Idee, vgl. Stoer) Auf einem einzelnen Intervall, z.B. $[0,h]$, gilt

$$\frac{h}{2}[f(0)+f(h)] = \Big[(x-\frac{h}{2})f(x)\Big]_0^h = \int_0^h \Big[(x-\frac{h}{2})f(x)\Big]'\,dx = \int_0^h f(x)dx + \int_0^h (x-\frac{h}{2})f'(x)dx.$$

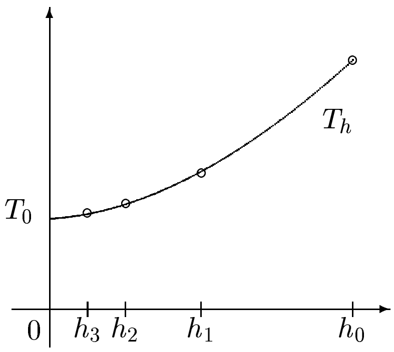
Hier kann nun wieder partiell integriert werden. Dabei lassen sich die Integrationskonstanten so wählen, daß Terme mit ungeraden h-Potenzen nicht auftreten. Im ersten Schritt etwa ist $p_2(x) := \frac{1}{2}(x^2 - hx + h^2/6)$ eine Stammfunktion von $(x-\frac{h}{2})$ mit $\int_0^h p_2(x)dx = 0$. Daher hat p_2 eine Stammfunktion $p_3(x) = \int_0^x p_2(t)dt$ mit $p_3(0) = p_3(h) = 0$ und es gilt

$$\frac{h}{2}[f(0)+f(h)] = \int_0^h f(x)dx + \frac{h^2}{12}[f'(h)-f'(0)] + \int_0^h p_3(x)f'''(x)dx,$$

also $b_1 = 1/12$. Das Gesamtergebnis (5.5.3) folgt durch Summation über die Teilintervalle. ■

Bemerkung: Der Summenformel entnimmt man, z.B., auch folgende Aussage: Wenn der Integrand $(b-a)$-periodisch ist, besitzt die Trapezregel sogar maximale Ordnung $2q+2$.

Extrapolationsverfahren arbeiten nach folgendem Prinzip. Ein Grenzwert $W = \lim_{h\searrow 0} T_h =: T_0$ soll bestimmt werden, wobei T_h aber nur für (einzelne) Werte $h > 0$ tatsächlich berechnet werden kann. Anstatt nun die Elemente einer Folge von Werten $T_{h_0}, T_{h_1}, \ldots$ unabhängig voneinander zu betrachten, ist es günstiger, das Interpolationspolynom zu $(h_0, T_{h_0}), \ldots, (h_m, T_{h_m})$ zu berechnen und dessen Wert in $h = 0$ als Näherung für T_0 heranzuziehen. Zur Auswertung des Polynoms in der Stelle 0 eignet sich am besten der *Neville-Algorithmus* (3.3.5).



Dazu wird aber (5.5.2) als Entwicklung nach der Variablen h^k aufgefaßt. Für die Polynomwerte $p_{ij} = p_{ij}(0)$ bekommt man so den folgenden Algorithmus der *Richardson-Extrapolation:*

$$\begin{aligned} p_{i0} &:= T_{h_i},\ i = 0,\ldots,m \\ p_{ij} &:= p_{i+1,j-1} + \frac{p_{i+1,j-1} - p_{i,j-1}}{(h_i/h_{i+j})^k - 1},\ \left\{\begin{array}{c} i = 0,\ldots,m-j \\ j = 1,\ldots,m \end{array}\right. . \end{aligned} \tag{5.5.4}$$

Die Berechnung erfolgt wieder in einem Dreieckschema spalten- oder zeilenweise. In der Praxis werden meist feste Schrittweitenfolgen, z.B. $h_i/h_{i+j} = 2^j$, verwendet. Dann benötigt (5.5.4) nur 3 Operationen pro Schritt. Die durch die Extrapolation (5.5.4) erreichte Erhöhung der Konvergenzordnung ergibt sich aus

Satz 5.5.4 *Wenn für die Größe T_h eine asymptotische Entwicklung (5.5.2) existiert, dann gilt für die nach (5.5.4) mit $h_0 > h_1 > \ldots$ extrapolierten Werte die Darstellung*

$$p_{ij} = T_0 + h_i^k \cdots h_{i+j}^k[(-1)^j a_{j+1} + \mathcal{O}(h_i^k)], \tag{5.5.5}$$

$h_0 \to 0,\ j < q$. Bei einem konstanten Schrittweitenverhältnis $v = h_i/h_{i-1} < 1$ vereinfacht sich die Darstellung zu

$$p_{ij} = T_0 + h_{i+j}^{k(j+1)}[(-1)^j v^{-kj(j-1)/2} a_{j+1} + \mathcal{O}(h_{i+j}^k)], \tag{5.5.6}$$

$h_0 \to 0,\ j < q$. Die Ordnung der Näherung hat sich also auf $h^{k(j+1)}$ erhöht.

Beweis Alle beteiligten Größen, insbesondere die in der Entwicklung (5.5.2) auftretenden h-Potenzen, werden als Funktionen von $s := h^k$ interpretiert,

$$h^{\ell k} = s^\ell =: g_\ell(s),\ \ell \geq 1, \quad \text{bzw.} \quad 1 = s^0 =: g_0(s).$$

Da die Polynominterpolation (vom Grad j zu $s_i, \ldots, s_{i+j}$) einer beschränkten, linearen Abbildung P_j entspricht, vgl. (3.6.11), kann man sich in der Entwicklung (5.5.2) wegen

$$P_j T_s = T_0 P_j g_0 + \sum_{\ell=1}^{q} a_\ell P_j g_\ell + \ldots$$

bei der Analyse des Verfahrens (5.5.4) auf einzelne Summanden beschränken. Der Wert des Interpolationspolynoms $q_{\ell,j}(s) = (P_j g_\ell)(s)$ vom Grad j zu einem Monom g_ℓ ist im Punkt $s = 0$ bei der konstanten Funktion ($\ell = 0$) $q_{0j}(0) = 1$ und für $0 < \ell \leq j$ dann $q_{\ell j}(0) = 0$. Für $\ell = j+1$ wird aber g_{j+1} nicht mehr exakt interpoliert, wegen $g_{j+1}(0) = 0$ ist der Wert $q_{j+1,j}(0)$ daher gerade der Interpolationsfehler in $s = 0$. Nach (3.3.11) ist dies

$$p_{j+1,j}(0) = g_{j+1}(0) - (0 - s_i)\cdots(0 - s_{i+j}) g_{j+1}[s_i, \ldots, s_{i+j}, 0] = (-1)^j s_i \cdots s_{i+j} \frac{(j+1)!}{(j+1)!}.$$

Somit hat das Interpolationspolynom $P_j(T_0 g_0 + a_1 g_1 + \ldots + a_{j+1} g_{j+1})$ vom Grad j in $s = 0$ den Wert $T_0 + (-1)^j s_i \cdots s_{i+j} a_{j+1}$. Dies entspricht der Behauptung. ∎

Praktische Durchführung

1. In der Trapezregel brauchen nach einer Schrittweitenhalbierung bei der Berechnung von $T_{h/2}$ die in T_h enthaltenen Funktionswerte nicht mehr neu berechnet zu werden. Bei Extrapolation sind daher folgende Schrittweitenfolgen günstig.

   a) *Romberg-Folge*: $h_0 = \beta - \alpha,\ h_i = 2^{-i} h_0$. Diese Quadratur-Methode heißt *Romberg-Verfahren.* In Satz 5.5.4 gilt

   $$p_{ij} = T_0 + h_{i+j}^{2j+2} \bar{a}_{j+1} + \ldots\ .$$

   Wenn hiermit hohe Ordnungen erreicht werden sollen, sind allerdings sehr viele Funktionsauswertungen erforderlich, nämlich $2^j + 1$ für p_{0j}. Eine Formel der Ordnung 16 etwa verwendet hier 129 Punkte, während die Gaußformel dafür mit 8 Punkten auskommt!

   b) *Bulirsch-Folge*: $h_0 = \beta - \alpha$ und für $i = 1, 2, \ldots$

   $$h_i/h_0 = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{12}, \ldots = \begin{cases} 2^{-(i+1)/2}, & i \text{ ungerade}, \\ \frac{2}{3} 2^{-i/2}, & i \text{ gerade}. \end{cases}$$

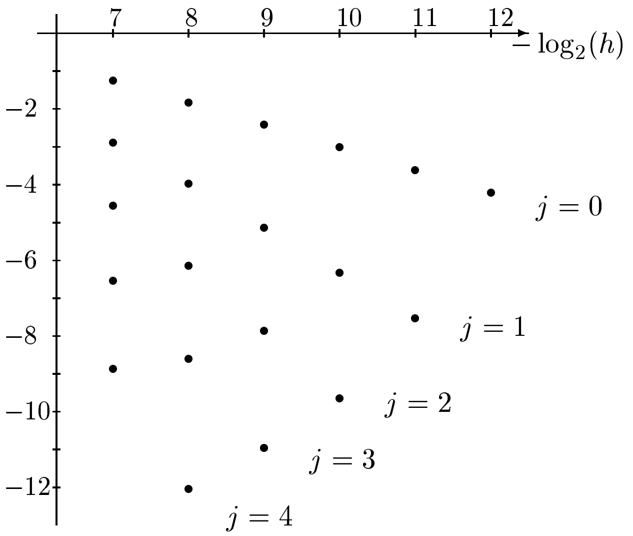
   Die Anzahl der für eine bestimmte Ordnung benötigten Stützstellen wächst hier langsamer, Ordnung 16, z.B., ist jetzt mit 25 Punkten erreichbar.

2. Wie bei den anderen Quadraturverfahren kennt man i. a. nicht die günstigste Ordnung (etwa die Differenzierbarkeitsordnung von f). Daher sollte die Zulässigkeit der Extrapolation in der Praxis durch Überwachung der Konvergenzaussage (5.5.5), (5.5.6) sichergestellt werden. Für die Romberg-Folge gilt nach (5.5.6):
$$p_{ij} = T_0 + h_{i+j}^{2j+2}\bar{a}_{j+1} + h_{i+j}^{2j+4}\bar{a}_{j+2} + \dots \;.$$
Daher ist
$$\begin{aligned} p_{i,j} - p_{i-1,j} &= h_{i+j}^{2j+2}(1-2^{2j+2})\bar{a}_{j+1} + h_{i+j}^{2j+4}(1-2^{2j+4})\bar{a}_{j+2} + \dots \\ p_{i+1,j} - p_{i,j} &= h_{i+j}^{2j+2}(2^{-2j-2}-1)\bar{a}_{j+1} + h_{i+j}^{2j+4}(2^{-2j-4}-1)\bar{a}_{j+2} + \dots \end{aligned}$$
Für den Quotienten dieser Differenzen folgt somit
$$\frac{p_{i,j} - p_{i-1,j}}{p_{i+1,j} - p_{i,j}} = \frac{(1-2^{2j+2})\bar{a}_{j+1} + h_{i+j}^2(\dots)\bar{a}_{j+2} + \dots}{(2^{-2j-2}-1)\bar{a}_{j+1} + h_{i+j}^2(\dots)\bar{a}_{j+2} + \dots} = 2^{2j+2} + \mathcal{O}(h_{i+j}^2).$$
Daher sollte bei einem zu großen Betrag der Abweichung
$$2^{-2j-2}\frac{p_{i,j} - p_{i-1,j}}{p_{i+1,j} - p_{i,j}} - 1$$
auf eine weitere Extrapolation $j \to j+1$ verzichtet werden.

Beispiel 5.5.5 $\int_1^2 \frac{1}{x}dx = \ln 2 = 0.69314718056$, Rombergfolge. Die Tabelle zeigt die Werte p_{ij}, $i = 1, .., 5$, $0 \le j \le i$. Der Wert p_{14} ist auf alle 11 Stellen genau.

h	j=0	1	2	3	4
1/2	0.70833333333	0.69325396825	0.69314790148	0.69314718307	0.69314718056
1/4	0.69702380952	0.69315453065	0.69314719430	0.69314718057	
1/8	0.69412185037	0.69314765282	0.69314718079		
1/16	0.69339120221	0.69314721029			
1/32	0.69320820827				

In der rechtsstehenden Graphik wird die Genauigkeit aller Näherungen in den verschiedenen Spalten der Tabelle gezeigt. An der horizontalen Achse ist der negative Zweier-Logarithmus der Schrittweiten abgetragen, an der vertikalen der Zehner-Logarithmus der Fehler. Die Ordnung der Näherungen in einer Spalte der Tabelle läßt sich an der Steigung der Punktereihen ablesen.



Bemerkung: 1) Die Richardson-Extrapolation beruht allein auf der *Existenz* einer asymptotischen Entwicklung (5.5.2). Solche Entwicklungen existieren auch bei vielen anderen Verfahren (z.B. den Differenzenformeln in §5.6). Dort läßt sich die Extrapolation daher analog einsetzen.

2) Mit dem Romberg-Verfahren kann bei adaptiver Integration auch einfach eine Ordnungssteuerung auf den einzelnen Teilintervallen implementiert werden.

5.6 Numerische Differentiation

Zur Approximation von Ableitungen einer Funktion kann man analog zu den einfachen Quadraturformeln das Interpolationspolynom heranziehen und dessen Ableitungen verwenden:

$$f^{(k)}(\hat{x}) \cong p_n^{(k)}(\hat{x}) = \sum_{j=0}^{n} f(x_j) L^{(k)}(\hat{x}).$$

Da hier weniger Gestaltungsmöglichkeiten bestehen als in §5.1, werden nur kurz einige Beispiele und Besonderheiten beim Fehlerverhalten besprochen. Im folgenden sei $[\alpha,\beta]$ ein (beliebig kleines) Intervall, das die Stelle $\hat{x}$ enthält.

Satz 5.6.1 *Es sei $p_n \in \Pi_n$ das Interpolationspolynom zu $f \in C^{n+1}[\alpha,\beta]$ und einfachen Stützstellen $x_j \in [\alpha,\beta]$, $j = 0,\ldots,n$. Dann gilt für $k \le n$ und $\hat{x} \in [\alpha,\beta]$ die Aussage*

$$\begin{aligned} p_n^{(k)}(\hat{x}) - f^{(k)}(\hat{x}) &= r_{kn}(\hat{x}) \qquad \textit{mit} \\ r_{kn}(x) &= (x - x_0^{(k)}) \cdots (x - x_{n-k}^{(k)}) \frac{f^{(n+1)}(\xi_k)}{(n+1-k)!}, \end{aligned} \tag{5.6.1}$$

$\xi_k \in (\alpha,\beta)$, wobei $\alpha < x_0^{(k)} < \ldots < x_{n-k}^{(k)} < \beta$. Mit $\varrho := \max_{j=0}^{n} |\hat{x} - x_j| \le \beta - \alpha$ gilt daher

$$|r_{kn}(\hat{x})| \le \frac{\varrho^{n+1-k}}{(n+1-k)!} \|f^{(n+1)}\|_\infty.$$

Beweis Iterierter Satz von Rolle, vgl. [Stummel-Hainer]

Bemerkung: Nach (5.6.1) gibt es offensichtlich in $[\alpha,\beta]$ Stellen, in denen der Fehler verschwindet. Durch geeignete Wahl der Stützstellen kann $\hat{x}$ in die Nähe solcher Punkte gebracht werden um, wie bei der Quadratur, eine bessere Konvergenz zu erhalten. Zur Untersuchung wird wieder eine zusätzliche Interpolationsstelle x_{n+1} herangezogen. In der Newtondarstellung ist

$$f(x) = p_n(x) + \omega_{n+1}(x) f[x_0,\ldots,x_{n+1}] + r_{n+1}(x),$$

$\omega_{n+1}(x) = (x - x_0) \cdots (x - x_n)$. Aus dem letzten Satz folgt dann

$$f^{(k)}(\hat{x}) - p_n^{(k)}(\hat{x}) = \omega_{n+1}^{(k)}(\hat{x}) f[x_0,\ldots,x_{n+1}] + r_{k,n+1}(\hat{x}), \tag{5.6.2}$$

mit $|r_{k,n+1}(\hat{x})| \le \varrho^{n+2-k} \|f^{(n+2)}\|_\infty / (n+2-k)!$. Daher wird also bei Auswertung in den Nullstellen von $\omega_{n+1}^{(k)}$ eine um eins erhöhte Konvergenzordnung erreicht für $\varrho \to 0$. Bei Approximationen mit minimaler Knotenzahl, d.h. $n = k$, gilt dazu

$$\omega_{n+1}(x) = x^{n+1} - \Big(\sum_{j=0}^{n} x_j\Big)x^n + \ldots \quad \Rightarrow \quad \omega_{n+1}^{(n)}(x) = (n+1)!\,x - n!\sum_{j=0}^{n} x_j.$$

Die Ableitung $p_n^{(n)} \equiv n!f[x_0,\ldots,x_n]$ ist konstant (vgl. (3.3.10)). Wegen (5.6.2) ist dieser Wert aber eine besonders gute Approximation für die Ableitung $f^{(n)}(x)$ im Mittelpunkt der Knoten

$$\hat{x} = \frac{1}{n+1}\sum_{j=0}^{n} x_j, \tag{5.6.3}$$

z.B., wenn die Knoten um $\hat{x}$ symmetrisch verteilt sind. In diesem Fall gilt also

$$\Big|f^{(n)}(\hat{x}) - n!f[x_0,\ldots,x_n]\Big| \le \frac{\varrho^2}{2}\|f^{(n+2)}\|_\infty.$$

Bei äquidistanten Knoten $x_j = x_0 + jh$ ist $\varrho \le nh$, in den einfachsten Fällen ergeben sich folgende Approximationen (sog. *Differenzenformeln*) für $\hat{x} = 0$:

$$\begin{aligned} f'(0) &= \frac{1}{h}[f(h) - f(0)] + chf''(\xi) \\ f'(0) &= \frac{1}{2h}[f(h) - f(-h)] + ch^2 f^{(3)}(\xi), \quad \text{vgl. (5.6.3)} \\ f''(0) &= \frac{1}{h^2}[f(-h) - 2f(0) + f(h)] + ch^2 f^{(4)}(\xi), \quad \text{vgl. (5.6.3)} \end{aligned}$$

Die Fehlerkonstante c ist dabei i.a. unterschiedlich. Eine Formel höherer Ordnung ist

$$f'(0) = \frac{1}{12h}[f(-2h) - 8f(-h) + 8f(h) - f(2h)] + ch^4 f^{(5)}(\xi). \tag{5.6.4}$$

Die Differenzenformeln sind umso genauer, je kleiner die Schrittweite h ist. Allerdings wird dann in der Formel durch die kleinen Zahlen h oder h^2 dividiert. Daher wird der bei der Berechnung der Werte $f(x_j)$ im Computer gemachte Fehler vergrößert und macht die Verwendung sehr kleiner Schrittweiten unsinnig. Da dieser Effekt hier besonders kritisch ist, soll er an der symmetrischen ersten Differenz erläutert werden, detaillierter wird das Problem in §7 untersucht. Daher seien jetzt $\tilde{f}(x_j) = f(x_j) + \epsilon_j$ die tatsächlich berechneten Funktionswerte, deren Fehler ϵ_j durch eine kleine Konstante $\mathcal{E}$ (ca. Maschinengenauigkeit, z.B. 10^{-11}) beschränkt sind, $|\epsilon_j| \le \mathcal{E}$. Dann gilt für den Gesamtfehler der tatsächlich *berechneten* Approximation

$$\Big|\frac{1}{2h}[\tilde{f}(h) - \tilde{f}(-h)] - f'(0)\Big| = \Big|ch^2 f''(\xi) + \frac{\epsilon_1 - \epsilon_0}{2h}\Big| \le \underbrace{ch^2\|f''\|_\infty}_{\searrow 0} + \underbrace{\frac{\mathcal{E}}{h}}_{\nearrow\infty} \quad (h \to 0). \tag{5.6.5}$$

Die Schranke wird minimal bei $\hat{h} \cong \mathcal{E}^{1/3}$ mit einem Minimalwert $\cong \mathcal{E}^{2/3}$. Man kann also bei diesem Beispiel nicht erwarten, eine Genauigkeit von mehr als $\frac{2}{3}$ der im Computer verfügbaren Stellen zu erreichen. Bei Formeln höherer Konvergenzordnung, wie (5.6.4), ist die erreichbare Genauigkeit besser.

6 Nichtlineare Gleichungssysteme

Viele praktische Probleme führen auf nichtlineare Gleichungen. Dies war beim Pendel §1 der Fall und gilt auch in den Problemen aus §2.1, wenn man kompliziertere Wechselwirkungen betrachtet. Die unbekannten Größen $x \in \mathbf{R}^n$ sind dann implizit durch Bedingungen in einer der beiden Formen

$$f(x) = 0, \qquad f : \mathbf{R}^n \mapsto \mathbf{R}^n, \tag{6.0.1}$$

$$x = g(x), \qquad g : \mathbf{R}^n \mapsto \mathbf{R}^n, \tag{6.0.2}$$

definiert, durch ein *nichtlineares Gleichungssystem*. In ausführlicher Form geschrieben heißt das

$$\left\{\begin{array}{ll} f_1(x_1,\dots,x_n) & = 0 \\ & \vdots \\ f_n(x_1,\dots,x_n) & = 0 \end{array}\right. \qquad \text{bzw.} \qquad \left\{\begin{array}{ll} x_1 = & g_1(x_1,\dots,x_n) \\ \vdots & \\ x_n = & g_n(x_1,\dots,x_n) \end{array}\right. .$$

Dabei wird im folgenden mindestens einmalige stetige Differenzierbarkeit der Funktionen f bzw. g angenommen. Die Gleichungen (6.0.1) und (6.0.2) sind Standardformen solcher Probleme, (6.0.1) heißt *Nullstellen-*, (6.0.2) *Fixpunktproblem.*

Für das Fixpunktproblem (6.0.2) kennt man im Banachschen Fixpunktsatz (Satz 2.5.1) ein hinreichendes Existenzkriterium *und* ein Lösungsverfahren (die Iteration). Das Nullstellenproblem (6.0.1) wird meist durch Umformungen auf die Gestalt (6.0.2) gebracht, auf die der Fixpunktsatz anwendbar ist. Der einfachste Spezialfall $n = 1$ wird zunächst betrachtet.

6.1 Nullstellen einer skalaren Funktion

Für reelle Funktionen g können die Aussagen des Fixpunktsatzes durch Monotoniebetrachtungen verfeinert werden. Es sei $g \in C^1[a, b]$.

- Die Existenz eines Fixpunkts (oder einer Nullstelle) läßt sich schon aus dem Zwischenwertsatz folgern:
$$a \le g(a),\ g(b) \le b \ \Rightarrow\ \exists\ z = g(z) \in [a, b].$$

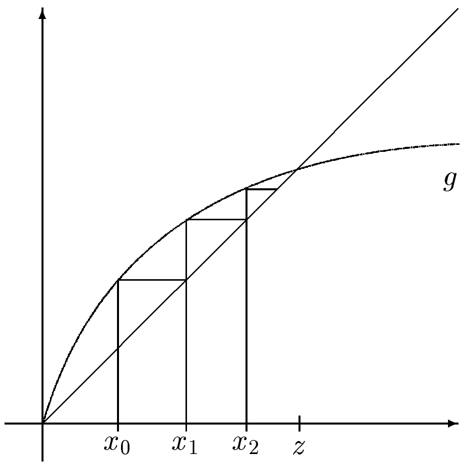
- Die Überprüfung der Lipschitzbedingung aus dem Banachschen Fixpunktsatz kann zur Vereinfachung auf den Fixpunkt z selbst beschränkt werden. Wenn $|g'(z)| < 1$ gilt, folgt aus der Stetigkeit von g' die Existenz einer ganzen Umgebung von z, in der Konvergenz eintritt. Zusätzlich wirkt sich das Vorzeichen von $g'(z)$ auf die Art der Konvergenz aus, bei positiver Steigung von g ist die Konvergenz monoton, bei negativer alternierend. Die Umgebung von z, in der Konvergenz gegen z eintritt, der sogenannte *Einzugsbereich* von z, kann sogar wesentlich größer sein als der Bereich $\{x : |g'(x)| < 1\}$ aus dem Fixpunktsatz. Im folgenden linken Beispiel hat die Funktion g den weiteren Fixpunkt 0 mit $g'(0) > 1$

("abstoßender Fixpunkt"). Offensichtlich konvergiert die Iteration hier für alle Startwerte $x_0 \in (0, z]$ gegen z. Dies läßt sich induktiv ganz einfach aus den Eigenschaften

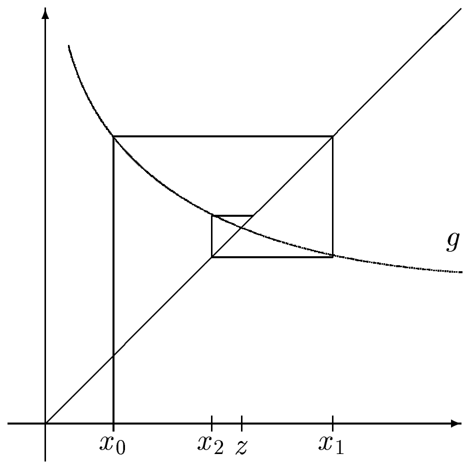
$$x < g(x) < g(z) \ \forall x \in (0, z)$$

herleiten, da dann $x_k < g(x_k) = x_{k+1} < g(z) = z$ gilt.

$0 < g'(z) < 1$: monotone Konvergenz



$-1 < g'(z) < 0$: alternierende Konvergenz



- Im Fall $g'(z) = 0$ tritt ein weiterer Effekt auf:

Beispiel 6.1.1 $f(x) = x^2 - a = 0 \, (a > 0) \iff x = \frac{a}{x} \iff 2x = x + \frac{a}{x}$.

Es sei $g(x) := \frac{1}{2}(a + \frac{a}{x}) \ \Rightarrow$ der Fixpunkt ist $z = \sqrt{a} = g(z)$. Die Iteration $x_{k+1} := g(x_k)$ heißt *Heron-Verfahren.* Die nebenstehende Tabelle zeigt die Iterierten für $a = 2$. Offensichtlich verdoppelt sich die Anzahl der richtigen Ziffern bei jedem Schritt. Dieser Effekt ist mit dem Banachschen Fixpunktsatz alleine nicht erklärbar.

k	x_k
0	1
1	1.5
2	1.4166
3	1.4142156
4	1.414213562

Eine Analyse ergibt im speziellen Beispiel die Identität

$$x_{k+1} - z = \frac{x_k^2 + a}{2x_k} - \sqrt{a} = \frac{1}{2x_k}(x_k^2 - 2x_k\sqrt{a} + a) = \frac{1}{2x_k}(x_k - z)^2,$$

bzw. allgemein mit der Eigenschaft $g'(z) = \frac{1}{2}(1 - a/z^2) = 0$ und dem Satz von Taylor

$$x_{k+1} - z = g(x_k) - g(z) = \underbrace{g'(z)}_{=0}(x_k - z) + \frac{1}{2}g''(\xi_k)(x_k - z)^2. \tag{6.1.1}$$

Aus beiden Formeln folgt, daß der Fehler bei jedem Iterationsschritt ungefähr quadriert wird mit der Konsequenz

$$|x_k - z| \le 10^{-m} \quad \Rightarrow \quad |x_{k+1} - z| \le \frac{1}{2}\|g''\| \, 10^{-2m}.$$

Dies entspricht ungefähr einer Verdopplung der Zahl der gültigen Ziffern pro Schritt.

Zur Beschreibung solcher Effekte führt man die *Konvergenzordnung* von Iterationsverfahren ein.

Definition 6.1.2 *Eine Folge $\{x_k\}$ mit $\lim_{k\to\infty} x_k = z$, bzw. ein sie erzeugendes Iterationsverfahren, heißen konvergent von der* Ordnung *$p \geq 1$, wenn $k_0 \in \mathbf{N}$, $q \in \mathbf{R}_+$ (wobei $q < 1$ bei Ordnung $p = 1$) existieren mit*

$$|x_{k+1} - z| \leq q\,|x_k - z|^p \quad \forall k \geq k_0. \tag{6.1.2}$$

Im allgemeinen wird als Ordnung das maximal mögliche p angegeben. Für $p = 1, 2, 3$ spricht man speziell von linearer, quadratischer und kubischer Konvergenz. Die Ordnung eines Iterationsverfahrens läßt sich an der Taylor-Entwicklung der Iterationsfunktion g im Fixpunkt z ablesen (wie in (6.1.1)).

Satz 6.1.3 *Es sei $z = g(z) \in U$ ein Fixpunkt von $g \in C^p(U)$, $p > 1$, U offen. Es gelte*

$$g'(z) = \ldots = g^{(p-1)}(z) = 0, \quad g^{(p)}(z) \neq 0.$$

Dann konvergiert die Iterationsfolge $\{x_k\}$, $x_{k+1} := g(x_k)$, $k = 0, 1, \ldots$, für Startwerte x_0 genügend nahe an z mit der Ordnung p gegen den Punkt z.

Beweis Wenn $0 < |x_0 - z| \leq \varepsilon$ mit genügend kleinem ε gilt, ist

$$x_{k+1} - z = g(x_k) - g(z) = g'(z)(x_k - z) + \ldots + \frac{g^{(p-1)}(z)}{(p-1)!}(x_k - z)^{p-1} + \frac{g^{(p)}(\xi_k)}{p!}(x_k - z)^p$$

wobei rechts alle Summanden, bis auf den letzten, verschwinden ($g^{(p)}(\xi) \neq 0$ wegen der Stetigkeit). Daher gilt (6.1.2). Die Konvergenz folgt für $q\varepsilon^{p-1} < 1$ aus

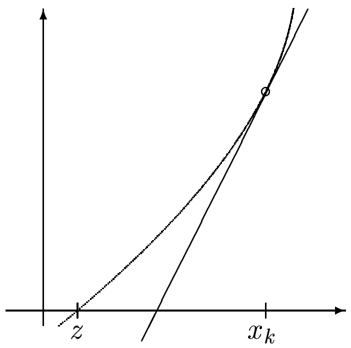
$$|x_{k+1} - z| \leq q|x_k - z|^p \leq (q\varepsilon^{p-1})|x_k - z|. \qquad \blacksquare$$

Bei der Konstruktion von Iterationsverfahren zur Nullstellenbestimmung geht man ähnlich vor wie früher: Die Funktion f wird durch ein einfacheres Modell approximiert (z.B. ein Polynom), für das sich eine Nullstelle einfach berechnen läßt. Dazu gibt es verschiedene Ansätze.

Newton-Verfahren

Approximation der Funktion f durch Abschnitte des Taylorpolynoms:

$$f(z) = \underbrace{\underbrace{0 \stackrel{!}{=} f(x_k) + f'(x_k)(z - x_k)}_{\to \text{ Newton-Verfahren}} + \frac{1}{2}f''(x_k)(z - x_k)^2}_{\to \text{ Newton-Raphson-Verfahren}} + \ldots$$



Das lineare Modell führt auf das *Newtonverfahren*

$$x_{k+1} := g_N(x_k), \quad g_N(x) := x - \frac{f(x)}{f'(x)}\,, \tag{6.1.3}$$

das quadratische Modell auf das *Newton-Raphson-Verfahren*

$$x_{k+1} := g_R(x_k), \quad g_R(x) := x - \frac{2f(x)}{f'(x) + \operatorname{sign} f' \sqrt{f'(x)^2 - 2f(x)f''(x)}}\,. \tag{6.1.4}$$

Nur das Newtonverfahren wird im folgenden genauer analysiert. Offensichtlich gilt

$$z = g_N(z) = z - \frac{f(z)}{f'(z)} \quad \Rightarrow \quad f(z) = 0.$$

Daher ist jeder Fixpunkt von g_N auch Nullstelle von f. Zur Anwendung von Satz 6.1.3 sind für $f'(z) \neq 0$ Ableitungen von g_N zu berechnen:

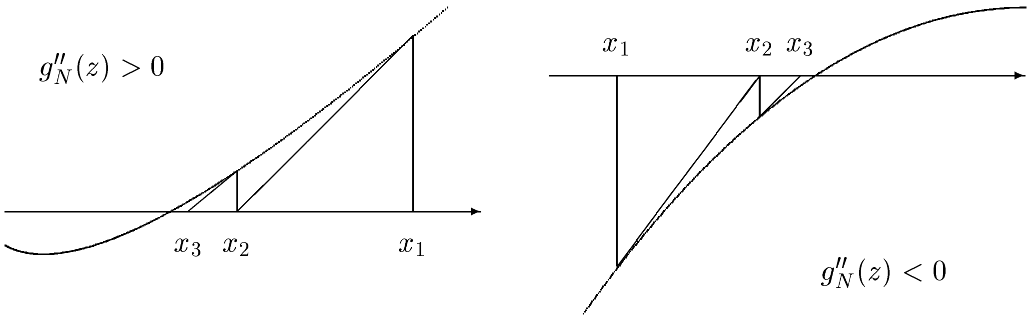
$$\begin{aligned} g_N'(x) &= 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = f(x)\frac{f''(x)}{f'(x)^2} &&= 0 \qquad \text{in } x = z, \\ g_N''(x) &= \frac{f'(x)^2 f''(x) + f(x)[\cdots]}{f'(x)^3} &&= \frac{f''(z)}{f'(z)} \quad \text{in } x = z, \end{aligned} \tag{6.1.5}$$

Hieraus folgt, daß (für $f'(z) \neq 0$) das Newtonverfahren *quadratisch* konvergiert und in einem Wendepunkt von f (mit $f''(z) = 0$) sogar kubisch. Das Newton-Raphson-Verfahren (6.1.4) konvergiert i.a. kubisch.

Außerdem verläuft die Konvergenz beim Newton-Verfahren meist *monoton*, im Beispiel 6.1.1, z.B., galt ab dem zweiten Schritt $x_k \searrow \sqrt{a}$. Dies folgt aus der Fehlerdarstellung (6.1.1),

$$x_{k+1} - z = \frac{1}{2} g_N''(\xi_k) \underbrace{(x_k - z)^2}_{\geq 0} \quad \Rightarrow \; \operatorname{sign}(x_{k+1} - z) = \operatorname{sign}\, g_N''(z) = \operatorname{sign}\, \frac{f''(z)}{f'(z)}$$

nahe bei z.



Bemerkung: 1) Wie die meisten Verfahren höherer Ordnung konvergiert das Newtonverfahren i.a. nur lokal, d.h., für genügend kleine Startfehler $|x_0 - z|$. Eine konkrete Aussage über dessen zulässige Größe folgt im nächsten Abschnitt für den $\mathbf{R}^n$. Der *Einzugsbereich* des Newtonverfahrens kann in ungünstigen Fällen sehr klein sein. Dann ist es sehr schwer, einen geeigneten Startwert x_0 zu finden ($\rightarrow$ Numerik 2A).

2) Auch bei mehrfachen Nullstellen, d.h., für $f'(z) = 0$, ist das Newtonverfahren noch verwendbar. Bei $f(x) = (x-z)^m r(x),\ r(z) \neq 0$, gilt nämlich in der Nähe von z

$$g_N(x) = x - \frac{(x-z)^m r(x)}{m(x-z)^{m-1}r(x) + (x-z)^m r'(x)} = x - \frac{(x-z)r(x)}{mr(x) + (x-z)r'(x)} \quad \Rightarrow$$

$$x_{k+1} - z = g_N(x_k) - z = (x_k - z)\Big(1 - \frac{1}{m + (x_k - z)r'(x_k)/r(x_k)}\Big) \cong (1 - \frac{1}{m})(x_k - z).$$

Daher konvergiert das Newtonverfahren noch linear mit dem Kontraktionsfaktor $1 - \frac{1}{m} < 1$.

Bisektionsverfahren

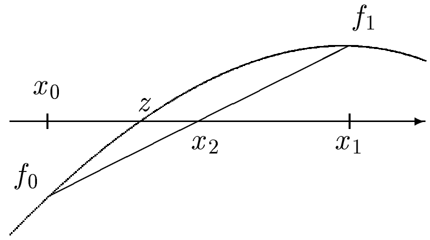
Ein einfaches Verfahren zur Nullstellenbestimmung beruht auf dem Zwischenwertsatz für stetige Funktionen. Bei jedem Vorzeichenwechsel der Funktion, d.h., Stellen $x_0 < y_0$ mit $f(x_0)f(y_0) < 0$, ist die Existenz einer Nullstelle $z \in (x_0, y_0)$ gesichert. Mit einem solchen Anfangsintervall (x_0, y_0) kann die Einschließung durch Intervallhalbierung ("Bisektion") verbessert werden:

$$[x_{k+1}, y_{k+1}] := \begin{cases} [m_k, y_k] & \text{für} \quad f(m_k)f(y_k) < 0 \\ [x_k, m_k] & \text{für} \quad f(x_k)f(m_k) < 0 \end{cases} \quad \text{mit } m_k := (x_k + y_k)/2. \tag{6.1.6}$$

Die Fallunterscheidung hält die Bedingung $f(x_{k+1})f(y_{k+1}) < 0$ aufrecht wobei $y_{k+1} - x_{k+1} = \frac{1}{2}(y_k - x_k)$. Daher zeigt dieses einfache Verfahren immerhin lineare Konvergenz mit dem Faktor $q = 1/2$ ohne jegliche Differenzierbarkeitsvoraussetzung an f und kann, z.B., dazu dienen, verläßliche Startwerte für Newton- oder andere Verfahren zu berechnen.

Sekantenverfahren, Regula falsi

Zur Approximation der Funktion f bei der Nullstellenbestimmung ist außer dem Taylor- auch ein Interpolationspolynom einsetzbar. Ausgehend von zwei Funktions-Wertepaaren $(x_0, f_0), (x_1, f_1)$ läßt sich eine verbesserte Näherung über die Nullstelle der linearen Interpolierenden (*Sekante*) bestimmen:



$$s(x) = f[x_1] + (x - x_1)f[x_1, x_0] = f_1 + (x - x_1)\frac{f_1 - f_0}{x_1 - x_0} \stackrel{!}{=} 0.$$

Dies führt auf den neuen Wert

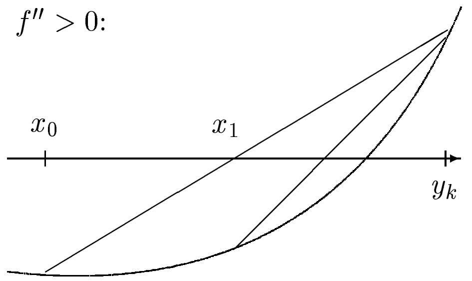
$$x_2 = x_1 - f_1\frac{x_1 - x_0}{f_1 - f_0} = \frac{x_0 f_1 - x_1 f_0}{f_1 - f_0}. \tag{6.1.7}$$

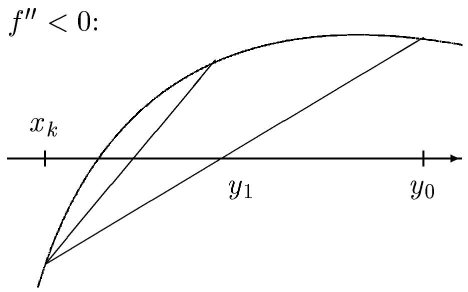
Die erste Form betont die Ähnlichkeit mit dem Newtonverfahren, da $(x_1 - x_0)/(f_1 - f_0) \cong 1/f'(x_1)$. Bei Wiederholung der Konstruktion (6.1.7) mit $\{x_1, x_2\}$ kann die nächste Näherung allerdings wieder schlechter als x_0 sein. Dies wird verhindert bei der

Regula falsi Mit (6.1.7) wird wie beim Bisektionsverfahren stets eine *Einschließung* (x_k, y_k) der Nullstelle z konstruiert, d.h., der Ablauf entspricht

$$\text{(6.1.6) mit } m_k := \frac{x_k f(y_k) - y_k f(x_k)}{f(y_k) - f(x_k)}. \tag{6.1.8}$$

Da dieses Verfahren die Bedingung $f(x_{k+1})f(y_{k+1}) < 0$ erzwingt, wird einer der Randpunkte nie verbessert, wenn f'' in der Nähe von z ein festes Vorzeichen besitzt:





Daher hat dieses Verfahren i.a. nur die Konvergenzordnung eins. Bei Verzicht auf die Einschließungseigenschaft erhält man das

Sekantenverfahren Ohne Beachtung des Vorzeichenwechsels der Funktionswerte wird (6.1.7) als Iterationsvorschrift angewendet:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}, \; k = 1, 2, \ldots \tag{6.1.9}$$

Dieses Verfahren hat nun eine andere Struktur als die Fixpunkt-Iteration. Da zur Konstruktion von x_{k+1} zwei frühere Iterierte verwendet werden, handelt es sich um eine *mehrstufige* Iteration.

Definition 6.1.4 *Ein Iterationsverfahren der Form*

$$x_{k+1} := G(x_k, x_{k-1}, \ldots, x_{k-m+1}), \; k \geq m-1,$$

mit $G : \mathbf{R}^m \mapsto \mathbf{R}$ *heißt* m-stufiges Verfahren.

Zur Untersuchung der Konvergenz des Verfahrens wird die Fortpflanzung der Fehler betrachtet. Aus (6.1.9) folgt

$$\begin{aligned} x_{k+1} - z &= x_k - z - \frac{f(x_k) - f(z)}{f[x_{k-1}, x_k]} = (x_k - z)\frac{f[x_{k-1}, x_k] - f[x_k, z]}{f[x_{k-1}, x_k]} \\ &= \frac{(x_k - z)(x_{k-1} - z)}{f[x_{k-1}, x_k]} \; \frac{f[x_{k-1}, x_k] - f[x_k, z]}{x_{k-1} - z} \\ &= (x_k - z)(x_{k-1} - z)\frac{f[x_{k-1}, x_k, z]}{f[x_{k-1}, x_k]}. \end{aligned} \tag{6.1.10}$$

Aus (3.3.10) folgt, daß Stellen ξ_k, η_k aus der konvexen Hülle von x_k, x_{k-1}, z existieren mit

$$f[x_{k-1}, x_k] = f'(\xi_k), \; f[x_{k-1}, x_k, z] = \frac{1}{2} f''(\eta_k).$$

Für $f'(z) \neq 0$ gibt es daher eine Umgebung von z, in der (6.1.9) konvergiert.

Satz 6.1.5 *Es sei* $U = [z-r, z+r], r > 0$, *eine Umgebung der Nullstelle* z *von* $f \in C^2(U)$. *Es gelte*

$$|f'(x)| \geq m_1 > 0, \; \frac{1}{2}|f''(x)| \leq M_2 \; \forall x \in U.$$

Dann konvergiert das Sekantenverfahren (6.1.9) für zwei beliebige Startwerte $x_0, x_1 \in U$ mit $M|x_0 - z| < 1,\ M|x_1 - z| < 1$ $(M := M_2/m_1)$ gegen die Nullstelle z. Die Konvergenzordnung ist dabei mindestens $p_S = \frac{1}{2}(1+\sqrt{5}) = 1.618..$.

Beweis Mit $e_j := |x_j - z|$, $j \geq 0$, lautet (6.1.10) kurz

$$e_{k+1} = e_k\, e_{k-1} \left| \frac{f[x_{k-1}, x_k, z]}{f[x_{k-1}, x_k]} \right|. \tag{6.1.11}$$

a) Nach (6.1.11) gilt mit $q := M \max\{e_0, e_1\} < 1$ induktiv

$$e_{k+1} \leq M e_k e_{k-1} \leq q e_k < e_k,\ \ k \geq 1.$$

Somit übertragen sich die benötigten Eigenschaften auf e_{k+1}, es gilt $e_{k+1} < r, M e_{k+1} < q$ und es folgt $e_{k+1} \leq q^k e_1 \to 0$.

b) Es wird nur der Fall $f''(z) \neq 0$ betrachtet und gezeigt, daß das Verfahren dann genau von der Ordnung $p = p_S$, mit $p^2 = p + 1$, konvergiert. Dazu sei $\hat{q} := \frac{1}{2} f''(z)/f'(z) \neq 0$ und $e_0, e_1 \neq 0$. Nach Teil a) und (6.1.11) gilt dann $e_{k+1} = q_k e_k e_{k-1}$ mit $q_k \neq 0$ für $k \geq k_0$, $q_k \to \hat{q}$, und die neue Folge $c_k := e_{k+1}/e_k^p$ ist daher wohldefiniert. Für sie gilt

$$c_k = \frac{e_{k+1}}{e_k^p} = q_k e_k^{1-p} e_{k-1} = q_k c_{k-1}^{1-p} e_{k-1}^{1+p-p^2} = q_k c_{k-1}^{1-p}.$$

Dies ist eine einstufige Rekursionsformel, zu der wegen $0 < p - 1 < 1$ eine beschränkte Folge $\{c_k\}$ gehört (Übung). Daraus folgt die Behauptung, $e_{k+1} \leq C e_k^p$. ∎

Bemerkungen: 1) Die Ordnung des Sekantenverfahrens ist etwas geringer als die des Newtonverfahrens. Allerdings benötigt ein Iterationsschritt (6.1.9) nur *eine* neue Funktionsauswertung, während beim Newtonverfahren $f(x_k)$ und $f'(x_k)$ zu berechnen sind. Dies ist bei einem Effizienzvergleich zu berücksichtigen: Mit Hilfe von $k = 2m$ Funktionsauswertungen wird der Anfangsfehler e_0 i.w. reduziert auf

$$e_0^{p_N^{k/2}} \text{ (Newton-Verf.)} \quad \text{bzw.} \quad e_0^{p_S^k} \text{ (Sekanten-Verf.)}.$$

Die *effektiven* Ordnungen (bezogen auf die Zahl der Funktionsauswertungen) sind also für Newton- und Sekanten-Verfahren

$$p_N^{1/2} = \sqrt{2} = 1.414\ldots < p_S = 1.618\ldots\ .$$

Da das Sekantenverfahren somit eine bessere effektive Ordnung besitzt und außerdem keine Ableitungen benötigt, ist es für skalare Probleme dem Newtonverfahren i.a. vorzuziehen (HP-Rechner: Solve-Taste).

Beispiel 6.1.6 $f(x) := \cos(x)\cosh(x) + 1$. Anwendung von Newton- und Sekantenverfahren:

Newton	Sekanten
$x_0=1.8$	$x_0=1.8$, $x_1=1.9$
	$x_2=\underline{1.87}36979$
$x_1=\underline{1.87}95674$	$x_3=\underline{1.875}0786$
	$x_4=\underline{1.875104}095$
$x_2=\underline{1.8751}186$	$x_5=\underline{1.875104069}$

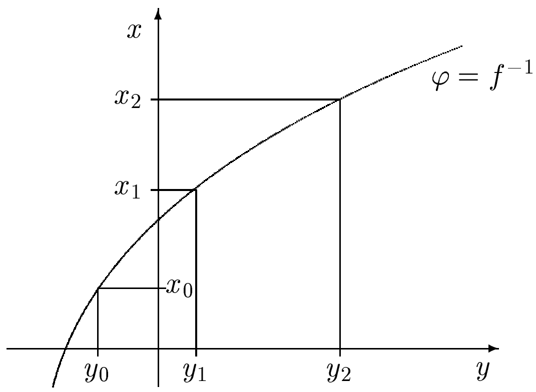
2) Die Vorteile von Regula falsi und Sekantenverfahren lassen sich durch einfache Überwachung vereinigen. Zusätzlich zur Folge $\{x_k\}$ aus (6.1.9) wird die bisher beste Einschließung

$$a_k := \max\{x_j : x_j \le z, j \le k\}, \quad b_k := \min\{x_j : x_j \ge z, j \le k\},$$

gespeichert und ein $x_{k+1} \notin [a_k, b_k]$ durch den besseren Randwert ersetzt. Weitere Varianten sind in der Literatur beschrieben, ihre Konvergenzordnungordnungen reduzieren sich geringfügig auf ungefähr 1.6.

Interpolationsverfahren höherer Ordnung

Versucht man, bessere Verfahren durch Übergang zu Interpolationspolynomen höherer Ordnung von f zu konstruieren, handelt man sich das Problem ein, dessen Nullstellen bestimmen zu müssen. Dies kann man umgehen durch Betrachtung der (lokalen) Umkehrfunktion $\varphi := f^{-1}$. Wenn nämlich z Nullstelle von f ist, $f(z) = 0$, dann gilt natürlich $z = \varphi(0)$. Dazu werden die Wertepaare $(x_j, y_j = f(x_j))$, $j = k, k-1, \ldots, k-m$, als Daten der Umkehrfunktion interpretiert, $(y_j, \varphi(y_j) = x_j)$, und so interpoliert:



$$q_m \in \Pi_m : \quad q_m(y_j) = x_j,\ j = k, k-1, \ldots, k-m.$$

Die nächste Iterierte ist dann einfach $x_{k+1} = q_m(0)$. Die Verbesserungen gegenüber dem Sekantenverfahren sind aber nicht mehr groß (Ordnung $< 2\ \forall m$).

6.2 Newtonverfahren im $\mathbf{R}^n$

Im Gegensatz zum skalaren Fall $n = 1$ gibt es bei der Lösung nichtlinearer Gleichungs*systeme* (6.0.1) mit $n > 1$ kaum eine Alternative zum Newtonverfahren. Nach Definition der Ableitung in einer Stelle $\bar{x} \in \mathbf{R}^n$,

$$f'(\bar{x}) := \frac{\partial f}{\partial x}(\bar{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\bar{x}) & \ldots & \frac{\partial f_1}{\partial x_n}(\bar{x}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\bar{x}) & \ldots & \frac{\partial f_n}{\partial x_n}(\bar{x}) \end{pmatrix} = \begin{pmatrix} \nabla f_1 \\ \vdots \\ \nabla f_n \end{pmatrix} \quad \text{zu } f = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}, \tag{6.2.1}$$

gilt mit der Matrix $A := f'(\bar{x})$ die Aussage

$$f(x) = f(\bar{x}) + A(x - \bar{x}) + o(\|x - \bar{x}\|),\ x \to \bar{x}. \tag{6.2.2}$$

Falls der Abstand $\|z - \bar{x}\|$ zu einer Nullstelle z klein genug ist, kann zur Approximation von z das *lineare Modell* benutzt werden:

$$0 = f(z) = f(\bar{x}) + A(z - \bar{x}) + \ldots \quad \Rightarrow \quad z \cong \bar{x} - A^{-1} f(\bar{x}).$$

Dies führt auf das folgende *Newtonverfahren* im $\mathbf{R}^n$, bei dem die Iteration

$$f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = -f(x^{(k)}), \quad k = 0, 1, \ldots, \tag{6.2.3}$$

durchgeführt wird mit einem Startvektor $x^{(0)} \in \mathbf{R}^n$. In jedem Schritt (6.2.3) ist dabei ein lineares Gleichungssystem zu lösen, mit dessen Lösung die aktuelle Näherung korrigiert wird,

$$\begin{array}{rcll} A_k s^{(k)} & = & -f(x^{(k)}) & \quad \text{mit Matrix } A_k = f'(x^{(k)}), \\ x^{(k+1)} & := & x^{(k)} + s^{(k)}. & \end{array} \tag{6.2.4}$$

Die Inverse A_k^{-1} wird dazu aber i.a. *nicht* berechnet (vgl. §2). Die Gestalt des Einzugsbereichs der Iteration (6.2.3) ist in der Regel sehr kompliziert. Unter geeigneten Voraussetzungen kann aber eine Kugel um z angegeben werden, in der Konvergenz $x^{(k)} \to z$ eintritt für jeden Startwert $x^{(0)}$ aus der Kugel. Dazu sei an den Mittelwertsatz im $\mathbf{R}^n$ erinnert:

Satz 6.2.1 *Es sei $D \subseteq \mathbf{R}^n$ offen, $f \in C^2(D)$ und $D_0 \subseteq D$ konvex und abgeschlossen. Dann gelten für $x, y \in D_0$ die Aussagen*

$$\begin{aligned} f(x) - f(y) &= \int_0^1 f'\Big(y + t(x-y)\Big)\, dt\ (x - y), && (6.2.5) \\ \|f(x) - f(y)\| &\le L_1 \|x - y\|, \quad L_1 := \max\{\|f'(x)\| : x \in D_0\}, && (6.2.6) \\ \Big(f'(x) - f'(y)\Big) v &= \int_0^1 f''(y + t(x-y))\, dt\ [v, x - y]\ \forall v \in \mathbf{R}^n, && (6.2.7) \\ \|f'(x) - f'(y)\| &\le L_2 \|x - y\|, \quad L_2 := \max\{\|f''(x)\| : x \in D_0\}. && (6.2.8) \end{aligned}$$

Bei (6.2.7) ist zu beachten, daß f'' eine bilineare Abbildung ist, $f''(x) : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$. Die Komponenten f_i'' sind symmetrische Matrizen und $f''(x)[v, y] = \Big(v^T f_i''(x) y\Big)_{i=1}^n$.

Beweis Für die skalaren Funktionen $\varphi_i(t) := f_i(y + t(x-y))$, $t \in [0, 1]$, gilt nach der Kettenregel

$$\begin{aligned} \frac{d\varphi_i}{dt}(t) &= \operatorname{grad} f_i(y + t(x-y)) \cdot (x - y) \quad \Rightarrow \\ \varphi_i(1) - \varphi_i(0) &= \int_0^1 \varphi_i'(t)\, dt = \int_0^1 \operatorname{grad} f_i(y + t(x-y))\, dt \cdot (x - y), \end{aligned}$$

d.h., (6.2.5) nach (6.2.1). Die Schranke (6.2.6) ergibt sich daraus sofort, denn

$$\|f(x) - f(y)\| \le \|\int_0^1 f'(y + t(x-y)) dt\|\, \|x - y\| \le \max_{t \in [0,1]} \|f'(y + t(x-y))\|\, \|x - y\|.$$

Der Beweis von (6.2.7),(6.2.8) geht analog. ■

Mit diesen Hilfsmitteln läßt sich nun die lokale Konvergenz des Newtonverfahrens mit Ordnung zwei zeigen.

Satz 6.2.2 *Es sei $D \subseteq \mathbf{R}^n$ offen, $f \in C^2(D)$ und $z \in D$ mit $f(z) = 0$. Die Ableitung $f'(z)$ in der Lösung sei regulär und es gelte*

$$\begin{aligned} \|f'(z)^{-1}\| &\le \frac{1}{\lambda_1}, && (6.2.9) \\ \sup_{x \in D} \|f''(x)\| &\le L_2. && (6.2.10) \end{aligned}$$

Die Kugel $K_\delta := \{x \in \mathbf{R}^n : \|x - z\| \le \delta\}$ sei in D enthalten,

$$K_\delta \subseteq D \quad \textit{für} \quad \delta < \frac{2\lambda_1}{3L_2}.$$

Dann konvergiert das Newtonverfahren (6.2.3) für jeden Startvektor $x^{(0)} \in K_\delta$ gegen z. Die Konvergenz ist quadratisch,

$$\|x^{(k+1)} - z\| \le \frac{1}{2}\frac{L_2}{\lambda_1 - \delta L_2}\|x^{(k)} - z\|^2. \tag{6.2.11}$$

Beweis O.B.d.A. wird der erste Schritt mit $x^{(0)} =: x$ betrachtet. Es gilt nach (6.2.3) und (6.2.5)

$$\begin{aligned} \|x^{(1)} - z\| &= \|\,x - z - f'(x)^{-1}(f(x) - f(z))\,\| \\ &= \|\Big(I - f'(x)^{-1}\int_0^1 f'(z + t(x - z))dt\Big)(x - z)\| \\ &\le \|f'(x)^{-1}\| \int_0^1 \|f'(x) - f'(z + t(x - z))\|dt\ \|x - z\|. && (6.2.12) \end{aligned}$$

Mit der Abkürzung $y = z + t(x - z)$, d.h., $x - y = (1 - t)(x - z)$, läßt sich nach (6.2.8) in K_δ die Differenz der Ableitungen abschätzen durch

$$\int_0^1 \|f'(x) - f'(z + t(x - z))\|dt \le L_2 \int_0^1 (1 - t)dt\ \|x - z\| = \frac{1}{2}L_2\|x - z\|. \tag{6.2.13}$$

Als nächstes wird die Regularität der Ableitung $f'(x)$ gezeigt. Es ist $f'(z)^{-1}f'(x) = I + f'(z)^{-1}(f'(x) - f'(z))$. Nach Voraussetzung (6.2.9) ist der zweite Summand klein,

$$\|f'(z)^{-1}(f'(x) - f'(z))\| \le \frac{L_2}{\lambda_1}\|x - z\| \le \frac{L_2\delta}{\lambda_1} < \frac{2}{3} < 1.$$

Nach dem Satz von Neumann, Satz 2.2.2, existiert daher $f'(x)^{-1}$ und nach (2.2.15) ist

$$\|f'(x)^{-1}\| \le \|f'(z)^{-1}\|\frac{1}{1 - L_2\delta/\lambda_1} \le \frac{1}{\lambda_1 - L_2\delta}.$$

Diese Schranke und (6.2.13) führen mit (6.2.12) auf die letzte Behauptung (6.2.11),

$$\|x^{(1)} - z\| \le \frac{1}{\lambda_1 - \delta L_2}\frac{1}{2}L_2\|x - z\|^2.$$

Damit ist die quadratische Konvergenz gesichert, wenn noch $\|x^{(1)} - z\| \le q\|x - z\|,\ q < 1$, gilt. Dies kann für $\delta < 2\lambda_1/(3L_2)$ garantiert werden:

$$\|x^{(1)} - z\| \le \underbrace{\frac{1}{2}\frac{L_2\delta}{\lambda_1 - \delta L_2}}_{=:\,q<1}\|x - z\| < \frac{1}{3}\frac{\lambda_1}{\lambda_1 - 2\lambda_1/3}\|x - z\| = \|x - z\|. \qquad \blacksquare$$

Durchführung des Newtonverfahrens

1. Satz 6.2.2 garantiert Konvergenz nur in einer, eventuell sehr kleinen, Kugel um die Nullstelle (*lokale Konvergenz*). Das globale Konvergenzverhalten des Verfahrens kann verbessert werden, indem der Fortschritt der Iteration anhand der Norm $\|f(x)\|_2$ überprüft wird. Man versucht dabei, die Funktion ("Zielfunktion")

   $$\varphi(x) := \|f(x)\|_2^2 \qquad \text{zu } \textbf{minimieren}.$$

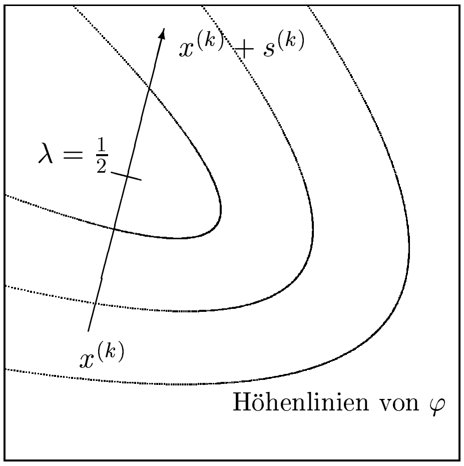
   Dazu wird der Vektor des Newtonschritts $s^{(k)} := -f'(x^{(k)})^{-1}f(x^{(k)})$ nur als Such*richtung* benutzt, und der nächste Punkt $x^{(k+1)}$ so auf dem Strahl

   $$x^{(k+1)} := x^{(k)} + \lambda_k s^{(k)}, \quad 0 < \lambda_k \le 1,$$

   bestimmt, daß $\varphi(x^{(k+1)}) < \varphi(x^{(k)})$ gilt. Man führt eine sogenannte *Liniensuche* durch. Da

   $$\frac{d}{d\lambda}\varphi(x^{(k)} + \lambda s^{(k)})\Big|_{\lambda=0} = -2\varphi(x^{(k)}) < 0$$

   gilt, existiert ein solcher Punkt für genügend kleines λ. In der Praxis kann dazu die Strecke $x^{(k)} \ldots x^{(k)} + s^{(k)}$ solange halbiert werden, bis φ einen kleineren Wert annimmt. Diese Wahl führt auf $\lambda_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$. Da dann $\lambda_k \le 1$ gilt, spricht man vom *gedämpften Newtonverfahren.* Nahe bei der Nullstelle wählt diese Methode automatisch wieder $\lambda_k = 1$, sodaß die quadratische Konvergenz nicht zerstört wird.

2. Im $\mathbf{R}^n$ sind pro Schritt des Newtonverfahrens n^2+n Komponenten von f und f' zu berechnen und eine LR-Zerlegung durchzuführen (Aufwand $O(n^3)$ Oper.). Wenn $x^{(k)}$ genügend nahe bei z liegt, kann evtl. die Ableitungsmatrix (und ihre LR-Zerlegung) von früher übernommen werden. Dies führt zum *vereinfachten Newtonverfahren*

   $$f'(x^{(0)})(x^{(k+1)} - x^{(k)}) = -f(x^{(k)}), \quad k = 0, 1, \ldots. \tag{6.2.14}$$

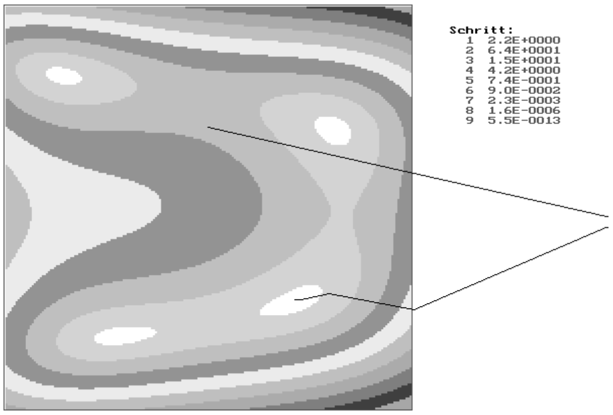
   Bei den Gleichungssystemen ändert sich jetzt nur die rechte Seite (vgl. §2.3), $-f(x^{(k)})$, (Aufwand $O(n^2)$ Operationen nach Berechnung der LR-Zerlegung). Die Konvergenz der vereinfachten Iteration (6.2.14) ist dafür nur noch linear mit dem asymptotischen Konvergenzfaktor $\|I - f'(x^{(0)})^{-1} f'(z)\| =: q$. Wenn q genügend klein und keine extreme Endgenauigkeit erforderlich ist, kann dies schneller zum Erfolg führen, da das volle Newtonverfahren n-mal so teuer ist.

Demo-Beispiel Im Quadrat $[-1, 1] \times [-1, 1] \subseteq \mathbf{R}^2$ besitzt die Funktion

$$f(x) = \begin{pmatrix} x_1^2 + 9x_2^2 + 2x_1 - 3 \\ 4x_1^2 - x_2^2 - x_2 - 1 \end{pmatrix}$$

4 Nullstellen bei $(-0.72, 0.66)$, $(0.62, 0.39)$, $(0.43, -0.46)$ und $(-0.44, -0.64)$. Die normale Newton-Iteration mit dem Startwert $(0, 0.4)$ erzeugt die Iterierten (12-stellige Rechnung)

k	$x_1^{(k)}$	$x_2^{(k)}$	$\varphi(x^{(k)})$
0	0.000000000	0.400000000	2.2E+00
1	3.900000000	-0.466666667	6.4E+01
2	1.974863569	-0.097182979	1.5E+01
3	1.024133838	-0.509788636	4.2E+00
4	0.603428523	-0.429913469	7.4E-01
5	0.457214218	-0.460213498	9.0E-02
6	0.433986060	-0.464807423	2.3E-03
7	0.433367993	-0.464932100	1.6E-06
8	0.433367555	-0.464932188	0.0E+00



Der erste Schritt führt nach weit außerhalb des Quadrats mit einer starken Vergrößerung des Wertes φ. Später konvergiert die Iteration offensichtlich quadratisch gegen eine Lösung in der unteren Halbebene. Bei einer Liniensuche im ersten Schritt wäre die nähergelegene Nullstelle bei $(0.62, 0.39)$ gefunden worden.

7 Rundungsfehler - Analyse

Bisher wurden Verfahren fast ausschließlich auf der Basis einer Rechnung mit reellen Zahlen diskutiert. Da in einem Computer aber nur endlich viele Zahldarstellungen (Maschinenzahlen) zur Approximation reeller Zahlen zur Verfügung stehen, treten i.a. bei jeder Dateneingabe und -verknüpfung Fehler auf, die sich — eventuell katastrophal — auf das Endergebnis auswirken können.

7.1 Zahldarstellung und Rundungsfehler

Die gängige (Dezimal-) Darstellung reeller Zahlen, z.B. e=2.71828.., entspricht der Angabe der Koeffizienten in der Reihen-Entwicklung

$$e = 2 \cdot 10^0 + 7 \cdot 10^{-1} + 1 \cdot 10^{-2} + 8 \cdot 10^{-3} + \ldots$$

zur *Basis* 10. Auf Computern ist eine Zahldarstellung günstiger zu einer Basis B, die eine Zweierpotenz ist, z.B., $B = 2,\ B = 16$. Jede reelle Zahl x ist darstellbar in der Form

$$x = \pm(x_m B^m + x_{m-1} B^{m-1} + \ldots + x_1 B + x_0 + x_{-1} B^{-1} + \ldots), \tag{7.1.1}$$

mit $x_j \in \{0, 1, \ldots, B-1\}$, wenn $1 < B \in \mathbf{N}$. Diese Darstellung ist nur dann eindeutig, wenn von zwei möglichen Darstellungen die endliche gewählt wird (man beachte: $1.999\ldots = 2$). Eine Abbildung mit möglichst kleinem *relativem* Fehler von (7.1.1) auf eine endliche Zahlmenge ist bei der Gleitpunktdarstellung möglich. Bei dieser werden der höchste Exponent m und die ersten ℓ Ziffern $x_m, \ldots, x_{m-\ell+1}$ angegeben. Der zulässige Exponentenbereich muß dabei natürlich auch beschränkt werden. Außerdem ist eine Standardisierung in folgender Weise üblich:

Definition 7.1.1 *Die Menge M der* normalisierten Gleitpunktzahlen *enthält die Elemente $x = 0$ und die Zahlen*

$$\begin{aligned} x \quad &= \quad \pm 0.\xi_1\xi_2\ldots\xi_\ell \cdot B^d := \pm(\xi_1 B^{-1} + \xi_2 B^{-2} + \ldots + \xi_\ell B^{-\ell})B^d, \\ & \qquad \textit{mit } \xi_j \in \{0, 1, \ldots, B-1\},\ \xi_1 \neq 0,\ d \in \mathbf{Z},\ |d| \le e. \end{aligned} \tag{7.1.2}$$

Im folgenden wird allerdings ein unbeschränkter Exponentenbereich ($e = \infty$) angenommen.

Beispiel 7.1.2 Binärdarstellung, $B = 2$. Wegen der Normalisierung $\xi_1 \neq 0$ gilt immer $\xi_1 = 1$. Daher kann an Stelle von ξ_1 das Vorzeichen gespeichert werden. In Turbo-Pascal wird eine *real*-Zahl in 5+1=6 Byte gespeichert:

$\pm 0.\xi_1\xi_2\ldots\xi_\ell \cdot B^d \quad \mapsto$

d	$\pm\xi_2..\xi_8$	$\xi_9..\xi_{16}$	$\cdots$	$\cdots$	$\ldots\xi_\ell$

Hier gilt also $\ell = 40, e = 128$. Die Zahlgenauigkeit beträgt ca. 12 Dezimalstellen, die Größe der Zahlen ist durch $2^d \cong 10^{38}$ beschränkt.

Zentraler Bestandteil jeder Computerarithmetik ist die Zuordnung von reellen Zahlen zu Gleitpunktzahlen, also die Abbildung

$$rd : \begin{cases} \mathbf{R} & \mapsto & M \\ x & \mapsto & rd(x) \end{cases}, \text{ die "Rundung".}$$

Eine triviale Variante dieser Rundung ist das Abschneiden der Darstellung (7.1.1): $\xi_1 \dots \xi_\ell = x_m \dots x_{m-\ell+1}$. Günstiger ist aber die gewöhnliche Rundung, bei der die Funktion rd jeweils auf die nächstgelegene Maschinenzahl abbildet. Bei Dezimalzahlen etwa wird ab 5 aufwärts gerundet.

$$rd(\pm 0.x_1 x_2 \dots x_\ell x_{\ell+1} \dots \cdot B^d) := \begin{cases} \pm 0.x_1 x_2 \dots x_\ell \cdot B^d & \text{falls} \quad x_{\ell+1} < B/2 \\ \pm(0.x_1 x_2 \dots x_\ell + B^{-\ell}) \cdot B^d & \text{falls} \quad x_{\ell+1} \geq B/2 \end{cases}. \tag{7.1.3}$$

Bei dieser Rundung gilt für den *relativen Fehler* folgendes Aussage.

Satz 7.1.3 *Für $x \neq 0$ gilt mit (7.1.3)*

$$\left|\frac{rd(x) - x}{x}\right| \leq \frac{1}{2} B^{1-\ell} =: \mathcal{E} \qquad \Longleftrightarrow \qquad rd(x) = x(1+\epsilon),\ |\epsilon| \leq \mathcal{E}.$$

Die Größe $\mathcal{E}$ heißt *Maschinengenauigkeit*, bei $B = 2$ ist $\mathcal{E} = 2^{-\ell}$.

Beweis In (7.1.3) ist $|x| \geq B^{d-1}$. In beiden Fällen gilt dort

$$|rd(x) - x| \leq \frac{B}{2} B^{-\ell-1} B^d \leq \frac{1}{2} B^{1-\ell} B^{d-1} \leq |x| \cdot \mathcal{E}. \qquad \blacksquare$$

Da das Ergebnis der Verknüpfung von zwei Maschinenzahlen i.a. nicht wieder eine solche ist, werden bei jeder Verknüpfung neue Rundungsfehler erzeugt. Daher müssen bei der Analyse Maschinenoperationen von reellen Verknüpfungen unterschieden werden:

Definition 7.1.4 *Für $a, b \in M$ und $* \in \{+, -, \times, /\}$ bezeichne (mit $b \neq 0$ im Fall der Division)*

$$a \mathbin{\tilde{*}} b = gl(a * b)$$

das Ergebnis der entsprechenden (geräteabhängigen) Gleitpunktoperation.

Bei der Konstruktion eines Rechners (Hardware) ist anzustreben, daß der Fehler bei diesen Operationen den zu erwartenden Rundungsfehler nicht übersteigt. Dies kann durch Verwendung eines Hilfsregisters ("Akkumulator") mit doppelter Stellenzahl 2ℓ garantiert werden. Dort wird die Operation mit 2ℓ Stellen ausgeführt und dann auf ℓ Stellen gerundet.

Satz 7.1.5 *Für die Gleitpunktoperationen zu $* \in \{+, -, \times, /\}$ auf einem ℓ-ziffrigen Rechner mit doppeltlangem Akkumulator (2ℓ Stellen) gilt für $a, b \in M$ ($b \neq 0$ bei Division)*

$$a \mathbin{\tilde{*}} b = (a * b)(1 + \delta), \quad |\delta| \leq \mathcal{E} = \frac{1}{2} B^{1-\ell}. \tag{7.1.4}$$

Erläuterung: Bei Addition von $a = \pm 0.\alpha_1 \ldots \alpha_\ell B^c$, $b = \pm 0.\beta_1 \ldots \beta_\ell B^d$, (oBdA $c \geq d$) ist als erstes eine Exponentenanpassung durchzuführen:

$$\begin{array}{l} \boxed{\pm 0.\alpha_1\alpha_2 \ldots \alpha_\ell} \\ \xleftrightarrow[c-d]{} \boxed{\pm 0.\beta_1\beta_2 \ldots \beta_\ell} \end{array}$$

Man sieht leicht, daß für $c-d \leq \ell$ das Ergebnis exakt mit 2ℓ Stellen darstellbar ist, für $c-d > \ell$ andererseits gilt $gl(a+b) = a$. Das Ergebnis der Multiplikation ist ebenfalls mit 2ℓ Stellen exakt darstellbar.

Die wichtigste Konsequenz der Rundung ist der Verlust der *Assoziativität* bei Gleitpunktoperationen. Eine wichtige Aufgabe der Fehleranalyse ist daher die Identifikation *günstiger* Anordnungen bei mehrfachen Verknüpfungen.

Beispiel 7.1.6 $\ell = 3$, $a = 0.721$, $b = 0.457_{10}2$, $c = -0.456_{10}2$. Je nach Klammerung folgt

$$\begin{aligned} (a \tilde{+} b) \tilde{+} c = &\quad (0.464_{10}2) \tilde{+} (-0.456_{10}2) &&= 0.800_{10}0, \\ a \tilde{+} (b \tilde{+} c) = &\quad 0.721 \tilde{+} (0.1) &&= 0.821_{10}0. \end{aligned}$$

Beide Ergebnisse sind offensichtlich verschieden.

Der in diesem Beispiel bei der ersten Version beobachtete Effekt wird *Auslöschung* genannt: Besitzen zwei Zahlen $a, b \in M$ gleiche Exponenten und gleiche führende Ziffern, so ist bei Subtraktion das Ergebnis *exakt*! Sind in den Operanden aber *alte* Fehler vorhanden, werden diese sehr verstärkt, da dann $|a-b| \ll |a|+|b|$ ist in (7.1.4). Dies muß bei der Fehlerfortpflanzung besonders beachtet werden.

Zunächst wird die Bildung mehrfacher Produkte und Summen betrachtet, also

$$w_n := a_1 * a_2 * \ldots * a_n, \quad n \in \mathbf{N}, \tag{7.1.5}$$

mit $* \in \{+, \times\}$. Das numerische Ergebnis $\widetilde{w_n}$ werde dabei rekursiv berechnet mit gerundeten Werten $\widetilde{a_j}$,

$$\widetilde{w_n} := gl(\ldots gl(gl(\widetilde{a_1} * \widetilde{a_2}) * \widetilde{a_3}) \ldots) = gl(\widetilde{w_{n-1}} * \widetilde{a_n}), \tag{7.1.6}$$

wobei $\widetilde{a_i} = a_i(1+\delta_i)$, $|\delta_i| \leq \mathcal{E}$. Für die weiteren relativen Fehler gelte ebenfalls $|\epsilon_i| \leq \mathcal{E}$.

Produkt: Hier gilt

$$\begin{aligned} \widetilde{w_2} &= gl(\widetilde{a_1} \cdot \widetilde{a_2}) = gl(a_1(1+\delta_1)a_2(1+\delta_2)) = a_1 a_2 (1+\delta_1)(1+\delta_2)(1+\epsilon_2) \\ &\ldots \\ \widetilde{w_n} &= gl(\widetilde{w_{n-1}} \cdot \widetilde{a_n}) = \widetilde{w_{n-1}} \cdot a_n (1+\delta_n)(1+\epsilon_n), \end{aligned}$$

Also

$$\begin{gathered} \widetilde{w_n} = w_n(1+\gamma_n) \qquad \text{mit} \\ (1-\mathcal{E})^{2n-1} \leq 1+\gamma_n \leq (1+\mathcal{E})^{2n-1}. \end{gathered} \tag{7.1.7}$$

Ausdrücke dieser Form treten jetzt öfter auf. Nach dem Mittelwertsatz gilt $(1+x)^m = 1 + mx(1+\xi)^{m-1}$ mit $|\xi| \le |x|$. Daraus folgt

$$|(1+x)^m - 1| \le m|x|(1+|x|)^{m-1} \le m|x|e^{(m-1)|x|} \le 1.06m|x| \quad \text{für } (m-1)|x| \le 0.1\ . \tag{7.1.8}$$

Daher gilt in (7.1.7) für γ_n die Aussage

$$|\gamma_n| \le 1.06(2n-1)\mathcal{E}, \quad \text{falls } 2(n-1)\mathcal{E} \le 0.1\ . \tag{7.1.9}$$

Summe: Bei der Addition ist die Situation komplizierter, hier ist

$$\begin{aligned}
\widetilde{w_2} &= gl(\widetilde{a_1} + \widetilde{a_2}) = (a_1(1+\delta_1) + a_2(1+\delta_2))(1+\epsilon_2) \\
&= a_1(1+\delta_1)(1+\epsilon_2) + a_2(1+\delta_2)(1+\epsilon_2) \\
\dots \\
\widetilde{w_n} &= gl(\widetilde{w_{n-1}} + \widetilde{a_n}) = (\widetilde{w_{n-1}} + a_n(1+\delta_n))(1+\epsilon_n) \\
&= [\widetilde{w_{n-2}} + a_{n-1}(1+\delta_{n-1})](1+\epsilon_{n-1})(1+\epsilon_n) + a_n(1+\delta_n)(1+\epsilon_n) = \dots \\
&= a_1(1+\sigma_1) + a_2(1+\sigma_2) + \dots + a_n(1+\sigma_n),
\end{aligned}$$

mit $(1+\sigma_i) = (1+\delta_i)(1+\epsilon_i)\cdots(1+\epsilon_n)$. Aus (7.1.8) folgt daher

$$|\sigma_i| \le 1.06(n-i+2)\mathcal{E}, \quad \text{wenn } (n-i+1)\mathcal{E} \le 0.1\ . \tag{7.1.10}$$

Bemerkung: Bei der Produktbildung ist in der Fehlerschranke keine Bevorzugung einer bestimmten Anordnung der Operanden erkennbar. Bei der Summe dagegen liefern die ersten Summanden den größten relativen Fehlerbeitrag. Daher ist zu erwarten, daß der absolute Fehler in $\widetilde{w_n}$ kleiner wird, wenn zuerst die *betragskleinsten* Koeffizienten a_i summiert werden, da dann die Schranke

$$|\widetilde{w_n} - w_n| \le 1.06\mathcal{E} \sum_{i=1}^{n} (n-i+2)|a_i|$$

minimal wird. Daher sollte man, z.B., bei der Summation (monoton) konvergenter Reihen mit den letzten, d.h. kleinsten, Koeffizienten beginnen ("Rückwärts-Summation").

Ungünstige arithmetische Ausdrücke: Aufgrund der Rundungsfehler geht bei Gleitpunktrechnung nicht nur die Assoziativität von Summe und Produkt verloren, auch andere Gesetze gelten im allgemeinen nicht mehr (exakt). Daher können verschiedene, im Reellen äquivalente Ausdrücke, wie z.B. die der binomischen Formel $a^2 - b^2 = (a+b)(a-b)$, sehr unterschiedliche numerische Ergebnisse liefern. Dazu werde die Funktion

$$f(x) := \sqrt{1+x^2} - x \tag{7.1.11}$$

betrachtet. Für $x \ge 0$ ist $0 \le f(x) \le 1$. Da $|f'(x)| = |\frac{x}{\sqrt{1+x^2}} - 1| \le 1$ ist für $x \ge 0$, führt eine Störung im Argument x bei exakter Rechnung nur zu einer geringen Verfälschung im Funktionswert f, die nicht größer als die Abweichung in x selbst ist. Werden allerdings die Teilschritte in (7.1.11) nur mit endlicher Genauigkeit ausgeführt, sieht das Ergebnis schlechter aus. Selbst wenn nur bei der Addition in (7.1.11) ein relativer Fehler ϵ, $|\epsilon| \le \mathcal{E}$, erzeugt wird, gilt

$$\widetilde{f}(x) = \sqrt{1 \widetilde{+} x^2} - x = \sqrt{(1+x^2)(1+\epsilon)} - x = f(x) + \frac{\epsilon}{2}\sqrt{1+x^2} + \mathcal{O}(\epsilon^2).$$

Für große $x \to \infty$ ist also ein großer absoluter Fehler $\cong \frac{1}{2}x\epsilon$ zu erwarten, der relative Fehler ist wegen $|f| \le 1$ noch größer. In der Formel (7.1.11) tritt nämlich bei der Differenzbildung der beiden großen Ausdrücke $\sqrt{1+x^2} \cong x$ und x eine *Auslöschung* führender Stellen auf. Diese läßt sich vermeiden durch Umformung in die äquivalente Form

$$f(x) = \sqrt{1+x^2} - x = \frac{1+x^2-x^2}{\sqrt{1+x^2}+x} = \frac{1}{x+\sqrt{1+x^2}}. \tag{7.1.12}$$

Beispiel 7.1.7 Formel (7.1.11) wird mit 4-ziffriger Dezimalrechnung in $x = 9.999$ ausgewertet, bei exakter Rechnung ist $f(9.999) = 0.04988\ldots$ Dagegen ist $x \widetilde{*} x = 99.98$, $1 \widetilde{+} 99.98 = 101.0$, $\widetilde{\sqrt{101}} = 10.05$ und daher $\widetilde{f}(x) = 0.05100$. Dies entspricht einem relativen Fehler von $0.022 \cong 450 \cdot \mathcal{E}$. Die letzte, äquivalente Form in (7.1.12) ergibt dagegen den korrekten 4-stelligen Wert 0.04988. Für $x \ge 100$ schließlich liefert 4-stellige Rechnung in (7.1.11) immer den Wert $\widetilde{f}(x) = 0$.

7.2 Rundungsfehleranalyse einiger Algorithmen

Für einige der früher behandelten Verfahren werden nun Konsequenzen aus dem Auftreten von Rundungsfehlern behandelt. Bei numerischen Verfahren, die selbst nur Näherungen mathematischer Lösungen liefern, kommen daher zum schon diskutierten Approximationsfehler des (abstrakten) Verfahrens bei Durchführung auf einem Rechner noch Rundungsfehler hinzu, im *Gesamtfehler* des realen Verfahrens sind also beide zu berücksichtigen.

Approximationsverfahren

Bei Interpolation, Quadratur und Differentiation hängt die Güte der Approximation von einem Parameter n (Grad, Knotenzahl,..) ab. Bei Vergrößerung von n wird der Verfahrensfehler in der Regel kleiner, allerdings treten dabei i.a. vermehrt Rundungsfehler auf.

Beispiel 7.2.1 Die konvergente Reihe $s := \sum_{k=1}^{\infty} a_k$ soll durch Partialsummen $s_n := \sum_{k=1}^{n} a_k$ angenähert werden, vorausgesetzt sei $|a_k| \le ck^{-m-1}$ $\forall k$. Dann gilt für den Approximationsfehler (Integralkriterium)

$$|s_n - s| \le \sum_{k=n+1}^{\infty} |a_k| \le c \sum_{k=n+1}^{\infty} k^{-m-1} \le c\int_n^{\infty} x^{-m-1}dx = \frac{c}{mn^m}.$$

Statt s_n wird aber $\widetilde{s_n}$ berechnet mit $|\widetilde{s_n} - s_n| \le 1.06\mathcal{E}(n+1)c(1+\frac{1}{m})$, vgl. (7.1.10). Also ist

$$|\widetilde{s_n} - s| \le \frac{c_0}{n^m} + c_1 n \cdot \mathcal{E}.$$

Beispiel 7.2.2 Approximation der Ableitung $f'(0)$ durch den symmetrischen Differenzenquotient $D_n = \frac{n}{2}[f(\frac{1}{n}) - f(-\frac{1}{n})]$ mit Schrittweite $h = \frac{1}{n}$. Nach (5.6.5) gilt

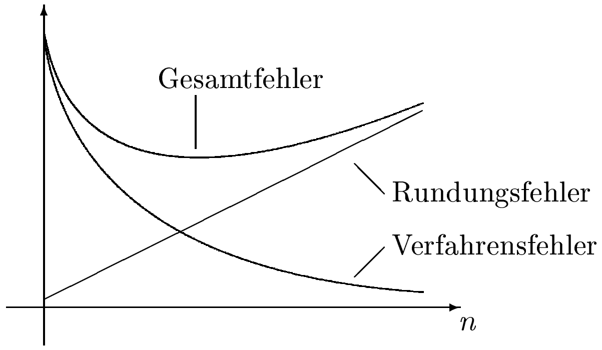
$$|\widetilde{D_n} - f'(0)| \leq \frac{c_0}{n^2} + c_1 n \cdot \mathcal{E}.$$

Beispiel 7.2.3 Approximation des Integrals $I := \int_a^b f(x)dx$ durch die iterierte Trapezregel mit $h = (b-a)/n, x_j := a + jh$:

$$T_h = h\Big[\frac{1}{2}f(a) + \sum_{j=1}^{n-1} f(x_j) + \frac{1}{2}f(b)\Big].$$

Für die tatsächlich berechneten Funktionswerte $\widetilde{f_j}$ gelte $\widetilde{f_j} = f(x_j)(1+\delta_j)$ mit $|\delta_j| \leq \delta \cdot \mathcal{E}$. Nach (7.1.10) gilt für den numerisch berechneten Wert $\widetilde{T_h}$ eine Abschätzung $|\widetilde{T_h} - T_h| = \frac{b-a}{n}|\frac{1}{2}f_0\sigma_0 + \sum_j f_j\sigma_j + \frac{1}{2}f_n\sigma_n| \leq c(b-a)\|f\|_\infty n \cdot \mathcal{E}$. Mit (5.2.6) führt dies auf die Abschätzung für den Gesamtfehler

$$|\widetilde{T_h} - I| \leq |T_h - I| + |\widetilde{T_h} - T_h| \leq c_0 h^2 + c_1 n \cdot \mathcal{E}.$$



In all diesen Fällen verhält sich der Fehler so, wie in der Graphik gezeigt: Dem für $n \to \infty$ kleiner werdenden Verfahrensfehler wirkt ein wachsender Rundungsfehler entgegen, da sich beide i. a. addieren. Die mit diesen realen Verfahren überhaupt erreichbare Genauigkeit ist daher begrenzt. Als Beispiel werde ein Verfahren der Ordnung m mit linear anwachsenden Rundungsfehlern betrachtet. Für den Gesamtfehler sei also eine Schranke F_n der Form

$$F_n = \frac{c_0}{n^m} + c_1\mathcal{E} \cdot n$$

bekannt. Die Ableitung dieses Ausdrucks nach n ist $\partial F_n/\partial n = -mc_0/n^{m+1} + c_1\mathcal{E}$, sie wächst monoton und führt auf die Minimalstelle

$$n_{\min} \cong \Big(\frac{c_0 m}{c_1\mathcal{E}}\Big)^{1/(m+1)} \quad \Rightarrow \quad F_{\min} \geq c_2\mathcal{E}^{m/(m+1)}.$$

Dies bedeutet, daß auf einem Rechner mit ℓ Stellen bei einem Verfahren der Ordnung m die Genauigkeit auf ca. $\frac{m}{m+1}\ell$ Stellen beschränkt ist. Daher können nur Verfahren hoher Ordnung Genauigkeiten im Bereich der vollen Maschinengenauigkeit erreichen.

Gauß-Algorithmus

Da der (reelle) Gauß-Algorithmus die exakte Lösung eines linearen Gleichungssystems liefert, treten kein Verfahrens- sondern nur Rundungsfehler auf. Bei der Beurteilung von numerischen

Algorithmen sollte man sich allgemein zunächst klar machen, daß ein ungenaues numerisches Ergebnis zu einem gegebenen Problem (mindestens) zwei Ursachen haben kann:

1. Das *mathematische* Problem reagiert empfindlich auf Störungen der Eingangsdaten. Bei linearen Gleichungssystemen ist dies bei einer großen Konditionszahl $\kappa(A) \gg 1$ der Fall, vgl. (2.4.7). Dann kann aber natürlich nicht erwartet werden, daß der *numerische* Algorithmus gute Ergebnisse liefert.

2. Das mathematische Problem ist unempfindlich gegen Störungen, der *numerische* Algorithmus erzeugt große Fehler.

Um hier die "Verantwortlichkeiten" klar zu machen, beschränkt man sich meist auf eine sogenannte *Rückwärtsanalyse* der Algorithmen. Dem entspricht folgende Begriffsbildung.

> *Ein Algorithmus heißt gut konditioniert, wenn er das* exakte *Ergebnis zu einem wenig gestörten Ausgangsproblem liefert.*

Dieser Begriff schließt also nur den zweiten Fall aus, im ersten Fall sind große Fehler im Ergebnis auf die schlechten Eigenschaften des mathematischen Problems zurückzuführen.

Beim Gauß-Algorithmus ist in diesem Sinne der Nachweis zu führen, daß er bei Anwendung auf ein $n \times n$-Gleichungssystem $Ax = b$ die exakte Lösung eines gestörten Problems

$$(A + A')\widetilde{x} = b + b', \quad \text{mit } \|A'\|, \|b'\| \le c \cdot \mathcal{E} \tag{7.2.1}$$

berechnet, wobei die Konstante c nicht zu groß ist. Konkret läßt sich folgendes beweisen.

Satz 7.2.4 *Der Gauß-Algorithmus (ohne Pivotisierung) liefert beim System $Ax = b$ eine Lösung des gestörten Systems (7.2.1), wobei für A' elementweise gilt*

$$|A'| \le \mathcal{E} \cdot n(3|A| + 5|\widetilde{L}||\widetilde{R}|) + O(\mathcal{E}^2). \tag{7.2.2}$$

Dabei sind $\widetilde{L}, \widetilde{R}$ die tatsächlich berechneten LR-Faktoren von A.

Als wichtigster Anteil ist das Betragsprodukt $|\widetilde{L}||\widetilde{R}|$ zu erkennen, dessen Elemente möglicherweise viel größere Werte als bei $|A|$ haben können. Dies ist besonders dann der Fall, wenn kleine Pivotelemente auftreten, da $l_{ij} = a_{ij}^{(j)}/a_{jj}^{(j)}$ ist, vgl. (2.3.4). Der Satz ist aber auch bei Pivotisierung anwendbar, da der Gauß-Algorithmus bei Spaltenpivotisierung die LR-Zerlegung eines zeilenpermutierten Systems $(PA)x = Pb$ bestimmt, P ist eine Permutationsmatrix. Bei dieser Pivotisierungsstrategie gilt $|\widetilde{l_{ij}}| \le 1$, d.h. $\|\widetilde{L}\|_\infty \le n$ für den berechneten L-Faktor in $LR = PA$, vgl. Beispiel 2.4.5.

Trotz dieser günstigen Aussagen zum Gauß-Algorithmus können große Fehler in einer Lösung auftreten, wenn die Konditionszahl der Matrix groß ist, $\kappa(A) \gg 1$. Dann ist aber, wie erläutert, schon das Ausgangsproblem fehleranfällig.

Fixpunkt-Iteration

Die zentrale Eigenschaft kontraktiver Iterationsverfahren bei linearen und nichtlinearen Gleichungssystemen ist die Tatsache, daß der Anfangsfehler in jedem Schritt um einen festen Faktor verkleinert wird. Dies trifft dann natürlich auch auf Fehler zu, die erst im Verlauf der Rechnung durch Rundung eingeschleppt wurden. Daher können Iterationsverfahren den Fixpunkt im wesentlichen mit derjenigen Genauigkeit bestimmen, die bei der Auswertung der Funktion g erreicht wird.

Satz 7.2.5 *Die Funktion g erfülle die Voraussetzungen (2.5.4),(2.5.5) des Banach'schen Fixpunktsatzes:*

$$\begin{aligned} g(\Omega) &\subseteq \Omega, \quad \Omega \subseteq H \\ \|g(x) - g(y)\| &\leq q\|x - y\| \ \forall x, y \in \Omega, \ \textit{mit } 0 < q < 1. \end{aligned}$$

Der Fixpunkt von g sei z. Für die tatsächlich berechnete Funktion $\widetilde{g}$ gelte

$$\|\widetilde{g}(x) - g(x)\| \leq \epsilon \ \forall x \in \Omega. \tag{7.2.3}$$

Dann gilt bei der gestörten Iteration

$$\widetilde{x}^{(k+1)} := \widetilde{g}(\widetilde{x}^{(k)}), \tag{7.2.4}$$

die a-posteriori-Schranke

$$\|\widetilde{x}^{(k)} - z\| \leq \frac{1}{1-q}\Big(q\|\widetilde{x}^{(k)} - \widetilde{x}^{(k-1)}\| + \epsilon\Big). \tag{7.2.5}$$

Beweis Mit der Aussage (2.5.7)

$$\|g(x) - z\| \leq \frac{q}{1-q}\|g(x) - x\|$$

aus dem Fixpunktsatz 2.5.1 folgt

$$\begin{aligned} \|\widetilde{x}^{(k+1)} - z\| &= \|\widetilde{g}(\widetilde{x}^{(k)}) - g(\widetilde{x}^{(k)}) + g(\widetilde{x}^{(k)}) - z\| \\ &\leq \epsilon + \|g(\widetilde{x}^{(k)}) - z\| \\ &\leq \epsilon + \frac{q}{1-q}\|g(\widetilde{x}^{(k)}) - \widetilde{x}^{(k)}\| \\ &\leq (1 + \frac{q}{1-q})\epsilon + \frac{q}{1-q}\|\widetilde{g}(\widetilde{x}^{(k)}) - \widetilde{x}^{(k)}\|. \qquad \blacksquare \end{aligned}$$

Der Satz zeigt, daß bei der Iteration der Gesamtfehler $\epsilon/(1-q)$ i.a. nicht unterschritten werden kann. Daher lohnt es sich auch nicht, den Iterationsfehler wesentlich unter diesen Wert zu drücken. Ein Abbruchkriterium für die gestörte Iteration (7.2.4), das bei beiden Fehleranteilen in (7.2.5) für gleiche Größenordnung sorgt, ist

$$q\|\widetilde{x}^{(k)} - \widetilde{x}^{(k-1)}\| \overset{!}{\leq} \epsilon.$$

Mit diesem Kriterium wird dann die bestmögliche Genauigkeit beim Gesamtfehler im wesentlichen erreicht.

Index

a-posteriori, 23, 97
a-priori, 23
adaptive Integration, 70
Äquilibrierung, 18
äquidistant, 34, 45
Akkumulator, 91
Assoziativität, 92
asymptotische Entwicklung, 72
Auslöschung, 92, 94

B-Spline, 57
Banachiewicz, 15
Banachscher Fixpunktsatz, 23, 97
baryzentrische Form, 33, 42
Bernstein, 50
Beziér, 50
Bisektion, 82
Bulirsch-Folge, 74

Cholesky-Zerlegung, 16
Clenshaw-Algorithmus, 41, 42
Crout, 15

Defekt, 20
Differentiation, 62, 76
Differenzen
- -Formel, 77
- -Quotient, 35, 39, 95
- -Verfahren, 4
- dividierte, 35, 36, 57

Drei-Term-Rekursion, 39, 68
Dreieck
- -Matrix, 15, 22
- -Schema, 34, 35, 73
- -System, 12, 42

Eigenwert, 29
Einzelschritt-Verfahren, 26
Einzugsbereich, 78, 81
Elektrisches Netzwerk, 7
Euler-McLaurin-Summenformel, 72
Extrapolations-Verfahren, 72

Fehler
- -Abschätzung, 23, 27, 43
- -Gleichverteilung, 70
- -Schätzung, 69, 70
- relativer, 19, 91

fill-in, 17
Fixpunkt, 23, 78, 97
- abstoßender, 79

Gauß
- -Algorithmus, 12, 13, 25, 96
  - Rundungsfehler, 95
- -Quadratur, 68
- -Seidel-Iteration, 26

Gesamt-Fehler, 77, 94
Gesamtschritt-Verfahren, 26
Gewichtsfunktion, 39, 62, 68
Gleichungssystem
- linear, 7
- nichtlinear, 78, 85
- Störung, 19

Gleitpunktzahl, 17, 90

Hölder-Normen, 9
Heron-Verfahren, 79
Horner-Schema, 35

Innenprodukt, 39, 67
Input-Output-Analyse, 7, 19, 23
Integration, 62
Interpolations-
- Fehler, 38, 43, 45
- Polynom, 31, 67, 76, 82, 85
  - Hermite-, 31
  - Lagrange-, 31, 42, 62

Newton-, 34, 42, 67, 76
Problem
Hermite-, 38
Punkte, neue, 35
Iteration, 23
lineare, 25
mehrstufig, 83
Rundungsfehler, 97

Jacobi-Iteration, 26
Jordan-Normalform, 11

Keller-Speicher, 71
Kirchhoff'sche Regeln, 7
Konditionszahl, 20, 96
Schranken, 21
kontrahierende Abbildung, 23
Kontrollpunkte, 50
Konvergenz, 65
-Faktor, 25
-Ordnung, 53, 84
effektive, 84
alternierend, 78
linear, 80
lokal, 81
monoton, 78, 81
quadratisch, 80, 81

Lebesgue-Funktion, 46
Liniensuche, 88
Lipschitz-Bedingung, 23, 78
LR-Zerlegung, 15, 16, 21, 25, 88, 96

Maschinengenauigkeit, 77, 91
Matrix, 9
-Norm, induzierte, 9
Band-, 16
dünnbesetzt, 17, 26
diagonaldominant, 16, 21, 26
diagonalisierbar, 10
hermitesch, 10, 28
inverse, 18
normal, 10
positiv definit, 28
unzerlegbar, 28
Minimierungsverfahren, 88
Mittelwertsatz, 63, 86

Nach-Iteration, 25
Neumann-Reihe, 11, 22, 23, 87
Neville-Algorithmus, 36, 42, 73
Newton
-Cotes-Formeln, 63, 65
-Polynom, 34
-Raphson-Verfahren, 81
-Verfahren, 80, 84
gedämpftes, 88
vereinfachtes, 88
Nullstelle, 78
mehrfache, 81

Ordnung, 53, 74, 76, 95
eines Iterationsverfahrens, 80

Pendel, 3
Pivotisierung, 13, 15
Spalten-, 18, 96
Strategien, 18
Polynom
bestapproximierendes, 46
Legendre-, 69
Orthogonal-, 39, 67, 68
Tschebyscheff-, 40

Quadratur, 62
-Fehler, 62, 71
-Formel, 62
Gauß-, 67
Gewichte, 62, 68
iteriert, 65, 67
Ordnung, 62, 67

Rückwärts
-Analyse, 96

-Summation, 93
Rechenaufwand, 14, 16, 18, 26, 32, 34–36, 41, 42, 66
Rechteckregel, 63
regula falsi, 82
Reihenentwicklungen, 69
Relaxation, 28
Über-, 29, 30
Parameter, 29
Residuum, 20
Restglied, 62, 67, 69
Richardson-Extrapolation, 73
Rodriguez-Formel, 69
Romberg-Folge, 74
Rundung, 91
Rundungsfehler, 17, 19, 25, 77, 91

Satz von
Neumann, 11
Rolle, 38, 69, 76
Stein-Rosenberg, 27
Weierstraß, 45
Sekanten-Verfahren, 83
Simpsonregel, 64, 68
iteriert, 65
SOR-Verfahren, 29
Spektral
-Norm, 9
-Radius, 10, 24, 27
Spline
-Approximation, lokale, 60
-Funktion, 48
kubischer, 48
Minimaleigenschaft, 48
natürlicher, 48
Stützstellen, optimale, 43

Taylor-Entwicklung, 39, 80
Trapezregel, 63, 68, 95
iteriert, 65, 69, 72
Tridiagonalmatrix, 5, 16, 52
Tschebyscheff
-Polynome, 40
-Punkte, 40, 44, 45
gestreckte, 47

Umkehrfunktion, 85

Volkswirtschaft, 7

Zeilen-Vertauschung, 13, 17
Zerlegung der Eins, 57, 58
Zerlegung, reguläre, 24