Numerik I

Bernhard Schmitt

Sommer-Semester 2017

Inhaltsverzeichnis

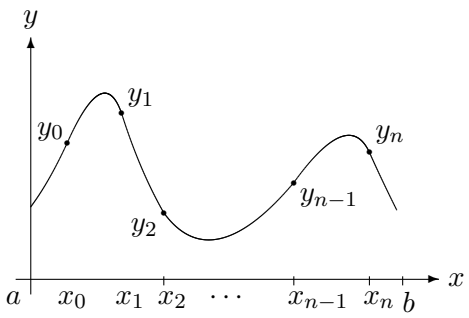
1 Einleitung

Die Aufgabe der *Numerik* ist die "Lösung" von mathematischen Modellen für Probleme aus der realen Welt (Wettervorhersage, Fahrzeugentwicklung, Finanz-Modelle) mit Hilfe von Computern. Allerdings gibt es nur für eine geringe Zahl solcher Probleme exakte Modelle, in der Regel müssen Modelle mit Vereinfachungen arbeiten, um überhaupt mathematisch fassbar zu sein. In der Übertragung von Modellergebnissen auf die Realität ist es daher wichtig, das Ausmaß der *Modellfehler* abzuschätzen. Bei der Verwendung von Computern gilt das sogar auf der grundlegendsten Ebene, denn die reellen Zahlen der Analysis sind auch nur Modelle, die es in der Realität nicht gibt, selbst die einfachsten arithmetischen Rechnungen mit Zahlen können ungenau sein aufgrund von *Rundungsfehlern*. Auch komplexere Aufgaben der Analysis ("infinitesimale") wie Differentiation oder Integration sind auf Computern nicht exakt durchführbar, sondern müssen durch Ausdrücke mit kleinen endlichen Größen approximiert werden. In diesen Fällen enthalten die Näherungen *Diskretisierungsfehler*. Daher ist es eine wesentliche Aufgabe der Numerik außer der Berechnung von Näherungslösungen für gegebene Problem auch eine Identifikation von Fehlerquellen und möglichst eine Schätzung der Fehlergröße anzugeben. Ein wichtiger Aspekt numerischer Verfahren ist aber natürlich auch der *Rechenaufwand*.

2 Interpolation und Quadratur

2.1 Interpolations-Polynome

Beobachtet man sich stetig ändernde Größen (Bewegung, Finanzdaten), so lassen sich immer nur endlich viele diskrete Werte tatsächlich messen. Zwischenwerte berechnet man mit einfachen Ersatzfunktionen, die die vorhandenen Daten berücksichtigen. Polynome bieten sich als Ersatzfunktionen an, da sie nur die Operationen Addition und Multiplikation verwenden und *linear* von den Koeffizienten abhängen. Die Einbeziehung der Messdaten geschieht am einfachsten durch *Interpolation*, dabei nimmt die Ersatzfunktion in bestimmten Parameterwerten, den *Stützstellen*, die Datenwerte exakt an. Später werden auch Splinefunktionen behandelt, welche sich stückweise aus Polynomen zusammensetzen.



Die Aufgabenstellung wird jetzt präzisiert.

Definition 2.1.1 *Gegeben sei ein Gitter $\{x_0, \ldots, x_n\}$ paarweise verschiedener Stützstellen,*

$$a \le x_0, \ldots, x_n \le b, \quad x_i \ne x_j \text{ für } i \ne j, \tag{2.1.1}$$

und ein Datenvektor $y := (y_0, \ldots, y_n)^\mathsf{T}$. Ein Polynom p mit

$$p(x_i) = y_i, \quad i = 0, 1, \ldots, n. \tag{2.1.2}$$

heißt (Lagrange-) Interpolationspolynom.

Die Interpolations-Bedingungen (2.1.2) führen im Prinzip auf ein lineares Gleichungssystem für die Koeffizienten a_j eines Polynoms vom Grad n in der Potenz-Darstellung

$$p_n(x) = \sum_{j=0}^{n} a_j x^j, \quad \Pi_n := \{p_n : \ a_j \in \mathbb{R}, j = 0, \ldots, n\}. \tag{2.1.3}$$

Das Lagrange-Interpolationspolynom kann aber direkt konstruiert werden:

Satz 2.1.2 *Zu einem Gitter (2.1.1) und beliebigen Werten y_i, $i = 0, \ldots, n$, existiert genau ein Interpolationspolynom vom Grad n, das die Forderungen (2.1.2) erfüllt. Es hat die Form*

$$p := p_y := \sum_{i=0}^{n} y_i L_i, \quad L_i(x) := \prod_{\substack{j=0 \\ j \ne i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{2.1.4}$$

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

Die Lagrange-Polynome L_i bilden nach Konstruktion eine Kardinal-Basis,

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

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

Beweis zu Satz 2.1.2 Wegen der Kardinalität der L_i gilt für p aus (2.1.4)

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

Zur Eindeutigkeit: wenn zwei Polynome $p, \bar{p} \in \Pi_n$ beide (2.1.2) erfüllen, so folgt

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

Das Differenzpolynom vom Grad n hat also $n+1$ Nullstellen und nach dem Fundamentalsatz der Algebra sind p und $\bar{p}$ identisch. ∎

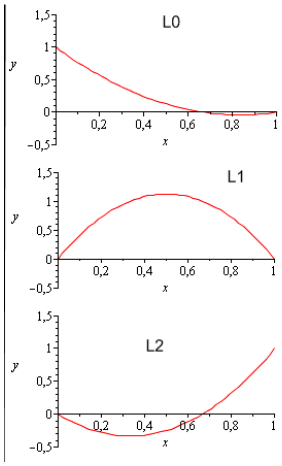
Den Rechenaufwand bei der Interpolation gibt man abhängig vom Polynomgrad n an. Wenn die Differenzen $x - x_j$ getrennt berechnet werden, sind zur Auswertung von (2.1.4) jeweils $n^2 + 2n$ Additionen, $n(n+1)$ Multiplikationen und $n(n+1)$ Divisionen nötig. Daher ist der

Rechenaufwand für eine Auswertung von (2.1.4): $3n^2 + \mathcal{O}(n)$ Operationen.

Dabei gibt man üblicherweise nur den am schnellsten wachsenden Term $3n^2$ an. Für theoretische Überlegungen ist die explizite Darstellung (2.1.4) mit dem Kardinalsystem L_i sehr bequem und ein Ansatz für andere Verfahren etwa bei der numerischen Integration (Quadratur). Für die praktische Beurteilung muß man den genauen Einsatzbereich präzisieren. Ist nur ein einziger Datensatz mit $y_i \in \mathbb{R}$, $i = 0, \ldots, n$, zu interpolieren, ist die Darstellung (2.1.4) zu teuer und wird kaum verwendet. Bei vektorwertigen Daten $y_i \in \mathbb{R}^d$, $i = 0, \ldots, n$, (z.B. Bahndaten einer Differentialgleichung) sieht das anders aus, $p(x) = \sum_{i=0}^{n} L_i(x) y_i$ beschreibt eine Kurve im $\mathbb{R}^d$, die teurere Berechnung der $L_i(x)$ amortisiert sich über die Dimension d insbesondere für $d \gg n$.

Beispiel 2.1.3 Es wird die Interpolation zum Gitter $x_0 = 0$, $x_1 = \frac{2}{3}$, $x_2 = 1$ betrachtet. Die zugehörigen Lagrangepolynome (2.1.4) sind daher

$$\begin{aligned} L_0(x) &= \frac{(x-2/3)(x-1)}{(0-2/3)(0-1)} &&= (\tfrac{3}{2}x - 1)(x-1) &&= \tfrac{3}{2}x^2 - \tfrac{5}{2}x + 1, \\ L_1(x) &= \frac{(x-0)(x-1)}{(2/3-0)(2/3-1)} &&= \tfrac{9}{2}x(1-x) &&= \tfrac{9}{2}(x - x^2), \\ L_2(x) &= \frac{x(x-2/3)}{1(1-2/3)} &&= x(3x-2) &&= 3x^2 - 2x. \end{aligned}$$



Für abstrakte Datenwerte y_0, y_1, y_2 oder vorgegebene $y^{\mathsf{T}} = (1, \frac{1}{3}, \frac{1}{2})$ bekommt man das Interpolationspolynom (2.1.4) explizit in der Form

$$\begin{aligned} p(x) &= (\frac{3}{2}x - 1)(x-1)y_0 + \frac{9}{2}x(1-x)y_1 + x(3x-2)y_2 \\ &= (\frac{3}{2}x - 1)(x-1) + \frac{3}{2}x(1-x) + \frac{1}{2}x(3x-2) = \frac{3}{2}x^2 - 2x + 1. \end{aligned}$$

Newton-Interpolation

Für skalare Daten $y_i \in \mathbb{R}$ kann man einen induktiver Ansatz für das Interpolationspolynom verwenden, der die einfache Hinzunahme zusätzlicher Stützstellen ermöglicht:

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

Nach Ansatz verschwindet der zweite Summand in den ersten n Punkten $x_0, \ldots, x_{n-1}$. Wenn also p_{n-1} die Interpolationsaufgabe in den ersten n Punkten erfüllt, $p_{n-1}(x_i) = y_i$, $i = 0, \ldots, n-1$, tut dies auch p_n. Den Wert a_n kann man dann aus der zusätzlichen Interpolationsbedingung $p_n(x_n) = y_n$ berechnen, zweckmäßiger ist aber ein weiter unten formulierter Algorithmus. Beginnend mit $p_0(x) \equiv y_0 = a_0$ bekommt p_n induktiv also die folgende Gestalt

$$\begin{aligned} p_n(x) &= a_0 + (x - x_0)a_1 + \ldots + (x - x_0) \cdots (x - x_{n-1}) a_n \\ &= a_0 + (x - x_0)\Big(\cdots \Big(a_{n-2} + (x - x_{n-2}) \Big(a_{n-1} + (x - x_{n-1}) a_n \Big) \Big) \cdots \Big) \end{aligned} \tag{2.1.5}$$

Dies ist die *Newton-Darstellung* des Interpolationspolynoms. Die Klammerung in (2.1.5) deutet bereits das Berechnungsverfahren zur Auswertung $p_n(x)$ an:

Algorithmus 2.1.4 Horner-Schema zur Auswertung von $p_n(x)$ bei bekannten a_i.

$q := a_n;$

für $k = n-1, n-2, \ldots, 0$: { $q := a_k + (x - x_k)q;$ }

$p_n(x) = q.$

Rechenaufwand Hornerschema: $3n$ Operationen

Die Koeffizienten a_i in (2.1.5) berechnet man über folgenden Algorithmus, dessen Begründung einige Zusatzüberlegungen erfordert.

Definition 2.1.5 (und Algorithmus) *Die k-ten* dividierten Differenzen *oder k-ten Differenzenquotienten der Daten y bezgl. der Stützstellen x werden bestimmt durch den Algorithmus*

$$\begin{array}{l} \textit{für } i = 0, 1, \ldots, n\textit{:} \ \{\ y[x_i] := y_i;\ \} \\ \textit{für } k = 1, \ldots, n\textit{:} \ \{ \\ \quad \textit{für } i = 0, \ldots, n-k\textit{:} \ \{ \\ \qquad y[x_i, \ldots, x_{i+k}] := \dfrac{y[x_{i+1}, \ldots, x_{i+k}] - y[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i};\ \} \\ \} \end{array} \tag{2.1.6}$$

Das Verfahren (2.1.6) erzeugt folgendes Dreieckschema spaltenweise ($k = 1, 2, \ldots, n$).

x_i	$y_i = y[x_i]$	$k=1$	$k=2$		$k=n$
x_0	$\underline{y_0}$				
x_1	y_1	$\underline{y[x_0, x_1]}$			
x_2	y_2	$y[x_1, x_2]$	$\underline{y[x_0, x_1, x_2]}$		
$\vdots$	$\vdots$	$\vdots$		$\ddots$	
x_n	y_n	$y[x_{n-1}, x_n]$	$y[x_{n-2}, x_{n-1}, x_n]$	$\cdots$	$\underline{y[x_0, \ldots, x_n]}$

(2.1.7)

Im Newtonpolynom (2.1.5) werden nur die unterstrichenen Differenzen $y[x_0, \ldots, x_i]$ benötigt, s.u., (2.1.11). Daher genügt zur Durchführung ein lineares Feld $[0, 1, \ldots, n]$, in dem der Reihe nach zunächst die Werte $y_n, \ldots, y_1$ rückwärts durch die ersten dividierten Differenzen, dann die ersten dividierten Differenzen $y[x_1, x_2]$,..., $y[x_{n-1}, x_n]$ durch die zweiten überschrieben werden, usw. Am Ende bleiben die in (2.1.11) 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 (2.1.6): $3(n + (n-1) + \cdots + 1) = \frac{3}{2}n^2 + \mathcal{O}(n)$ Operationen.

Die Beziehung der Differenzenquotienten (2.1.6) zu den Koeffizienten a_i im Newtonpolynom (2.1.5) bekommt man aus dem folgenden Neville-Algorithmus und dem Satz 2.1.7.

Definition 2.1.6 (und Neville-Algorithmus) *Es seien $p_{ik} \in \Pi_k$ Abschnittspolynome mit*

$$p_{ik}(x_j) = y_j, \quad j = i, \ldots, i+k. \tag{2.1.8}$$

Man erhält die Werte dieser p_{ik} und den des Polynoms p_n zu (2.1.2) an einer Stelle x durch

$$\boxed{\begin{array}{l} \textit{für } i = 0, 1, \ldots, n: \quad \{\ p_{i0}(x) := y_i;\} \\ \textit{für } k = 1, \ldots, n: \{ \\ \quad \textit{für } i = 0, \ldots, n-k: \{ \\ \qquad p_{ik}(x) := \dfrac{(x - x_i)p_{i+1,k-1}(x) + (x_{i+k} - x)p_{i,k-1}(x)}{x_{i+k} - x_i}; \}\} \\ p_n(x) = p_{0n}(x). \end{array}} \tag{2.1.9}$$

Der Neville-Algorithmus liefert (mit $O(n^2)$-Aufwand) nur einen einzigen Polynomwert und ist daher nur in Sonderfällen von praktischem Interesse. Er stellt aber die Verbindung zu den dividierten Differenzen aus (2.1.6) her.

Satz 2.1.7 *a) Die in (2.1.9) berechneten Polynome und die in (2.1.8) definierten sind gleich.*

b) Die dividierten Differenzen in (2.1.6) stimmen mit den höchsten Koeffizienten der Teilpolynome in (2.1.8) überein:

$$y[x_i, \ldots, x_{i+k}] = \frac{1}{k!}\, p_{ik}^{(k)}, \quad 0 \le i \le i + k \le n. \tag{2.1.10}$$

Dieser Wert ist unabhängig von der Reihenfolge der Wertepaare $(x_i, y_i), \ldots, (x_{i+k}, y_{i+k})$.

c) Das Lagrange-Interpolationspolynom (2.1.2) ist in der Newtonform (2.1.5) darstellbar als

$$p_n(x) = y[x_0] + (x - x_0)\, y[x_0, x_1] + \ldots + (x - x_0) \cdots (x - x_{n-1})\, y[x_0, \ldots, x_n]. \tag{2.1.11}$$

Beweis Durch Induktion über k, die Aussagen zu (2.1.8) bis (2.1.11) sind offensichtlich richtig für $k = 0$.

a) Unter der Induktionsvoraussetzung, dass $p_{i,k-1}(x)$ und $p_{i+1,k-1}(x)$ ihre jeweiligen Interpolationsbedingungen in (2.1.8) erfüllen, gilt für das in (2.1.9) definierte $p_{ik}(x)$ die Aussage

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

Für $j = i$ und $j = i + k$ treten nämlich in (2.1.9) nur der zweite bzw. erste Summand im Zähler auf, für die Fälle $j = i + 1, \ldots, i + k - 1$ kann man aufgrund der Induktionsvoraussetzung den Faktor y_j ausklammern, die Vorfaktoren summieren zu eins.

b) Nun wird angenommen, (2.1.10) sei richtig für $k - 1$. Dann gilt nach (2.1.5) und (2.1.8)

$$p_{ik}(x) = p_{i,k-1}(x) + (x - x_i) \cdots (x - x_{i+k-1}) \cdot a_{ik}. \tag{2.1.12}$$

Die k-te Ableitung dieser Identität zeigt $p_{ik}^{(k)} = k!\, a_{ik}$ und die Behauptung (2.1.10) ist daher äquivalent zu

$$a_{ik} = y[x_i, \ldots, x_{i+k}]. \tag{2.1.13}$$

Um dies zu zeigen, betrachtet man die höchsten Koeffizienten a_{ik} im Neville-Algorithmus (2.1.9). Mit der Induktionsvoraussetzung erhält man tatsächlich die Differenzen-Rekursion (2.1.6) für die Werte (2.1.13):

$$a_{ik} = \frac{a_{i+1,k-1} - a_{i,k-1}}{x_{i+k} - x_i} \overset{(I.V.)}{=} \frac{y[x_{i+1}, \ldots, x_{i+k}] - y[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i} \overset{(2.1.6)}{=} y[x_i, \ldots, x_{i+k}].$$

c) Ist der Spezialfall von (2.1.12) und (2.1.13) für $i = 0$ und $k = 0, \ldots, n$. ■

Beispiel 2.1.8 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 Daten y_j berücksichtigt. Wenn diese allerdings als *Funktionswerte* einer glatten Funktion f interpretiert werden können, lässt sich die Abweichung von p_n und f abschätzen.

Satz 2.1.9 *Es sei $f \in C^{n+1}[a,b]$ und $y_j = f(x_j)$, $j = 0, \ldots, n$, mit einem Gitter (2.1.1).*

a) Es gilt

$$f[x_i, \ldots, x_{i+k}] := y[x_i, \ldots, x_{i+k}] = \frac{1}{k!}\, f^{(k)}(\xi), \tag{2.1.14}$$

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

b) Für $f(x) = c_k x^k + \ldots \in \Pi_k$ ist

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

c) Für den Interpolationsfehler gilt mit dem Knotenpolynom $\omega_{n+1}(x) := (x - x_0) \ldots (x - x_n)$ und einem $\xi \in (a,b)$ die Darstellung

$$f(x) - p_n(x) \;=\; \omega_{n+1}(x) f[x_0, \ldots, x_n, x] \;=\; \omega_{n+1}(x) \frac{f^{(n+1)}(\xi)}{(n+1)!}\,. \tag{2.1.15}$$

Beweis Die Interpolationsfehler der Abschnittspolynome werden mit $r_{ik}(x) := f(x) - p_{ik}(x)$ bezeichnet.

a) Nach (2.1.10) ist $y[x_i, \cdots, x_{i+k}] = p_{ik}^{(k)}(x)/k!$. Nun verschwindet r_{ik} 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

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

b) Dies ist ein Spezialfall von a).

c) Der Interpolationsfehler wird an einer festen Stelle $x = \bar{x}$ dargestellt und das erweiterte Interpolationspolynom $p_{0,n+1}$ mit der zusätzlichen Stützstelle $x_{n+1} := \bar{x}$, $y_{n+1} := f(\bar{x})$, betrachtet. Nach Satz 2.1.7 gilt hierfür

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

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

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

Die Definition von ω_{n+1} und (2.1.14) liefern schließlich (2.1.15). ∎

Wegen der Beziehung (2.1.14) liegt es nahe, den Grenzübergang von mehreren Knoten in eine gemeinsame Stelle zu betrachten. Bekanntlich ist

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

Analoge Aussagen gelten in (2.1.14), wenn mehr als zwei Knoten zusammenrücken. Das Interpolationsproblem ist allerdings nur dann sinnvoll gestellt, wenn an einer Stelle Funktions- und Ableitungswerte bis zu einer bestimmten Ordnung lückenlos vorgeschrieben sind. Die Definition 2.1.1 kann daher verallgemeinert werden.

Definition 2.1.10 *Gegeben sei ein Gitter $\{x_0, \ldots, x_n\}$ beliebiger Stützstellen $x_i \in [a,b]$, zusammenfallende Stützstellen seien zusammenhängend nummeriert ($x_i = x_j$ für $i < j \Rightarrow x_k = x_i$ für $i \le k \le j$). Beim Hermite-Interpolationsproblem wird zu einem Datenvektor $(y_0, \ldots, y_n)$ ein Polynom $p \in \Pi_n$ gesucht mit*

$$p^{(j)}(x_i) = y_{i+j},\ x_i = x_{i+1} = \ldots = x_{i+j},\ 0 \le i \le n.$$

Für die Hermite-Interpolationsaufgabe sind also die folgenden Werte vorzugeben,

$$f(x_i), f'(x_i), \cdots, f^{(j)}(x_i), \quad \text{wenn } x_i = x_{i+1} = \cdots = x_{i+j}. \tag{2.1.16}$$

Im Einklang mit (2.1.14) definiert man mit diesen Ableitungen Differenzenquotienten durch

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

Für alle solchen *Mehrfachenknoten* (2.1.16) werden diese Werte in das Differenzentableau (2.1.7) eingetragen. Die noch fehlenden Differenzenquotienten berechnet man dann nach (2.1.6).

Beispiel 2.1.11 Differenzentableau für $n = 4,\ x_1 = x_2 = x_3$:

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	$y[x_1, x_4]$	$y[x_1, x_1, x_4]$	$y[x_1, x_1, x_1, x_4]$	$y[x_0, x_1, x_1, x_1, x_4]$

Bemerkung: Man kann das Hornerschema aus Algorithmus 2.1.3 verallgemeinern zur Berechnung von Ableitungswerten von Polynomen.

Satz 2.1.9 erlaubt eine Gegenüberstellung von Taylor- und Newton-Entwicklung für $f \in C^{n+1}[a,b]$. Mit den analog zu ω_{n+1} definierten Polynomen $\omega_k(x) = (x-x_0)\cdots(x-x_{k-1})$, vgl. (2.1.15), erhält man für die Entwicklung nach

$$\text{Taylor:} \quad f(x) = \sum_{k=0}^{n}(x-x_0)^k \, \frac{f^{(k)}(x_0)}{k!} + (x-x_0)^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!},$$

$$\text{Newton:} \quad f(x) = \sum_{k=0}^{n} \omega_k(x) \; f[x_0,\cdots,x_k] + \; \omega_{n+1}(x) \; \frac{f^{(n+1)}(\eta)}{(n+1)!},$$

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

Interpolationsfehler und optimale Knoten

Wenn die Plazierung der "Eingangsdaten" (x_i, y_i) nicht fest vorgegeben ist, kann der Interpolationsfehler ohne Mehraufwand durch geeignete Wahl der Stützstellen x_i stark reduziert werden. Dazu setzt man an der Darstellung (2.1.15) des Interpolationsfehlers an. Wenn eine *gleichmäßig* gute Approximation im Intervall $[a,b]$ erzielt werden soll, benutzt man die Supremumnorm

$$\|g\|_\infty := \sup\{|g(x)| : \; x \in [a,b]\}, \quad g \in C[a,b]$$

des Fehlers und versucht, diese möglichst klein zu machen. Für eine Funktion $f \in C^{n+1}[a,b]$ führt die Fehlerdarstellung (2.1.15) mit $\omega_{n+1}(x) = (x-x_0)\cdots(x-x_n)$ auf die Schranke

$$\|f - p_n\|_\infty \le \|\omega_{n+1}\|_\infty \frac{\|f^{(n+1)}\|_\infty}{(n+1)!}. \tag{2.1.17}$$

Bei festem Polynomgrad n ist der Anteil $\|f^{(n+1)}\|_\infty$ durch die Funktion f fest vorgegeben. Dagegen kann man aber die Norm $\|\omega_{n+1}\|_\infty$ des Knotenpolynoms durch geeignete Wahl der Stützstellen beeinflussen. Es ist sogar eine optimale Wahl möglich mit Knoten $x_0, \ldots, x_n$, die

$$\|\omega_{n+1}\|_\infty = \min_{\xi_j \in [a,b]} \max_{x \in [a,b]} |x-\xi_0||x-\xi_1|\cdots|x-\xi_n| \tag{2.1.18}$$

erfüllen. In diesem Zusammenhang spielen folgende Polynome eine Rolle

Definition 2.1.12 *Die Funktionen*

$$T_n(x) := \cos(n \arccos x), \; |x| \le 1, \quad n = 0,1,.., \tag{2.1.19}$$

heißen Tschebyscheff-Polynome *und die Nullstellen von* T_{n+1}

$$x_k := x_{k,n} := \cos\Big(\frac{2k+1}{2n+2}\pi\Big), \quad k = 0,\ldots,n \tag{2.1.20}$$

Tschebyscheff-Punkte oder -Knoten.

Satz 2.1.13 *Die in (2.1.19) definierten Funktionen T_n sind Polynome vom Grad n. Sie können ausgehend von $T_0 = 1,\ T_1(x) = x$, schrittweise berechnet werden durch die Drei-Term-Rekursion*

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),\ n = 1, 2, \ldots. \tag{2.1.21}$$

Die Punkte $x_k = x_{k,n}$ aus (2.1.20) sind die Nullstellen des Polynoms T_{n+1}. Die Polynome T_n haben die Form

$$T_n(x) = 2^{n-1}x^n + \ldots, \quad T_n \in \Pi_n, \tag{2.1.22}$$

ihre Supremum-Norm auf $[-1, 1]$ ist eins, für $n \in \mathbb{N}_0$ gilt

$$\|T_n\|_{[-1,1],\infty} = \max_{x\in[-1,1]} |T_n(x)| = 1. \tag{2.1.23}$$

Beweis Mit $t := \arccos x$ folgen die Aussagen für T_0, T_1 unmittelbar aus (2.1.19). Die Rekursion in (2.1.21) ergibt sich aus dem Additionstheorem

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

An (2.1.21) erkennt man, dass die $T_n \in \Pi_n$ die Form (2.1.22) besitzen und an (2.1.19), dass die x_k aus (2.1.20) die Nullstellen von T_{n+1} sind und (2.1.23) gilt. ∎

Umskalierte Tschebyscheff-Polynome führen tatsächlich auf optimale Interpolationsknoten.

Satz 2.1.14 *Die optimalen Knoten in (2.1.18) sind die* Tschebyscheff-*Punkte*

$$x_k := \frac{a+b}{2} - \frac{b-a}{2}\cos\Big(\frac{2k+1}{2n+2}\pi\Big), \quad k = 0, \ldots, n. \tag{2.1.24}$$

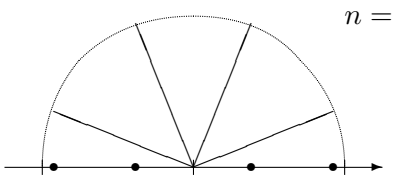
Der Minimalwert der Norm ist

$$\|\omega_{n+1}\|_\infty = \max_{x\in[a,b]} |\omega_{n+1}(x)| = 2\left(\frac{b-a}{4}\right)^{n+1}. \tag{2.1.25}$$

Bemerkung: Die Knoten ergeben sich aus den in (2.1.20) genannten durch affine Transformation des Intervalls

$$\left\{\begin{array}{ccc} [-1,1] & \to & [a,b] \\ \xi & \mapsto & \frac{a+b}{2} - \frac{b-a}{2}\xi = x \end{array}\right. . \tag{2.1.26}$$

Die Punkte $\xi_k = \cos\frac{2k+1}{2n+2}\pi = Re\exp\left(i\frac{2k+1}{2n+2}\pi\right)$ sind 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 $[a, b] = [-1, 1]$ gilt $x_k = -\xi_k,\ k = 0, \ldots, n$, also

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

Daher ist ω_{n+1}, bis auf einen konstanten Vorfaktor, das Tschebyscheff-Polynom. Ein Vergleich des höchsten Koeffizienten mit (2.1.22) 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 $\pm 2^{-n}$ daher in $t_j = -\cos\frac{j\pi}{n+1}$, $j = 0, \ldots, n+1$ (*Alternanden*). 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

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

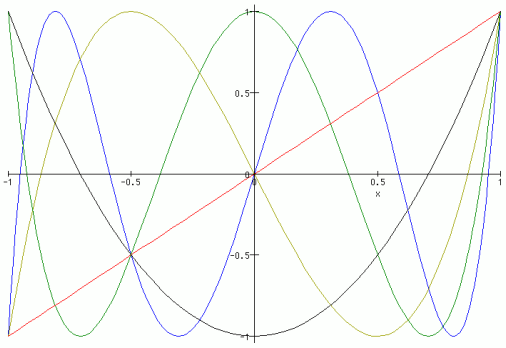
Das Polynom $\omega_{n+1} - p_{n+1}$ vom Grad n (!) hätte dann aber mindestens $n+1$ Nullstellen. Dies führt zu einem Widerspruch. Durch die Transformation (2.1.26) wird

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

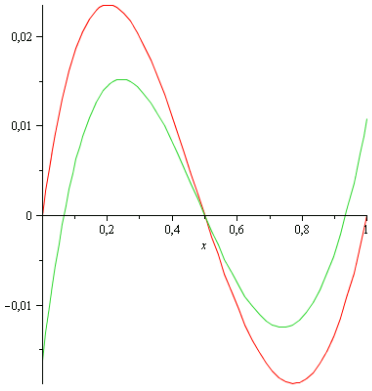
die Normierung des höchsten Koeffizienten ergibt damit

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

Daraus folgt die Schranke (2.1.25). ∎



Die nebenstehende Graphik der Tschebyscheffpolynome $T_1, \ldots, T_5$ zeigt die auch im Beweis verwendeten Eigenschaften dieser Polynomfamilie. Die Polynome von (un-) geradem Grad sind (un-) gerade Funktionen, die expliziten Formeln sind $T_2 = 2x^2 - 1$, $T_3(x) = 4x^3 - 3x$, $T_4(x) = 8x^4 - 8x^2 + 1$, $T_5(x) = 16x^5 - 20x^3 + 5x$.



Beispiel 2.1.15 Die Funktion $f(x) = \sin(\frac{\pi}{2}x)$ soll im Intervall $[0,1]$ durch ein quadratisches Polynom approximiert werden. Dazu wird die Interpolation in den Stellen $x^\mathsf{T} = (0, \frac{1}{2}, 1)$ verglichen mit der in Tschebyscheffknoten $\hat{x} = (\frac{1}{4}(2-\sqrt{3}), \frac{1}{2}, \frac{1}{4}(2+\sqrt{3}))$. Die nebenstehende Grafik zeigt dazu die Interpolationsfehler. Die beiden Tabellen enthalten jeweils das Differenzentableau, das Polynom und den Fehler, links zu äquidistanten Punkten, rechts zu Tschebyscheffknoten.

0	0		
0.5	0.70710678	1.41421356	
1	1	0.58578643	−0.82842712

$p(x) = x(1.82842712 - 0.82842712x)$

Fehler $\cong 0.0235$

0.06698729	0.10502933		
0.5	0.70710678	1.39043829	
0.93301270	0.99446912	0.66363490	−0.83924026

$\hat{p}(x) = -0.01622158 + 1.86627686x\ -0.83924026x^2$

Fehler $= 0.0162$

Wegen des niedrigen Polynomgrads ist der Genauigkeitsunterschied in diesem Beispiel nur gering. Dies ändert sich bei höheren Polynomgraden. Nach dem Satz von Weierstraß kann jede stetige Funktion auf einem kompakten Intervall zwar beliebig genau durch Polynome approximiert werden. Dieses Ergebnis überträgt sich aber nicht unbesehen auf Interpolationspolynome, selbst wenn immer mehr Stützstellen verwendet werden. Mit der Angabe (2.1.25) hat die Fehleraussage (2.1.17) für Tschebyscheffknoten die Form

$$\|p_n - f\|_\infty \leq\ 2\Big(\frac{b-a}{4}\Big)^{n+1}\ \frac{\|f^{(n+1)}\|}{(n+1)!}. \tag{2.1.27}$$

Der Vorfaktor bei $f^{(n+1)}$ ist dabei kleiner als bei allen anderen Stützstellen, etwa bei äquidistanten, $x_i = a + (b-a)i/n$. Bei der Formel (2.1.27) ist aber zu beachten, dass sich bei Anhebung des Polynomgrads n auch die Ordnung der Ableitung $f^{(n+1)}$ erhöht. Schon lange bekannt ist das Beispiel von Runge mit der Funktion

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

wo bei äquidistanten Stützstellen das Anwachsen der Ableitungen für $n \to \infty$ verhindert, dass der Fehler klein wird. Die Folge der Interpolationspolynome (p_n) $(n \to \infty)$ divergiert in diesem Beispiel sogar in Randnähe bei ± 1. Bei Tschebyscheffknoten ist wegen der Alternanden-Eigenschaft (vgl. Beweis) der Fehlerverlauf viel gleichmäßiger. Für $f \in C^{m+1}[-1,1], m \geq 1$ lässt sich hier sogar allgemein Konvergenz nachweisen in der Form [Schaback, 10.4.5]

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

2.2 Quadratur

Die Differentiation von Funktionen erfolgt nach festen, wohlbekannten Regeln. Dagegen ist die Berechnung von Integralen eine Kunst ohne klare Regeln, viele Integrale besitzen sogar keine explizit angebbare Stammfunktion. Daher ist der Bedarf nach numerischen Approximationen offensichtlich. Generell kann man zur Approximation einer aufwändigen, oder nicht explizit durchführbare Operation mit einer Funktion f so vorgehen, dass man f durch ein einfacheres Modell annähert und die Operation dann mit diesem Modell ausführt. Bei der *Integration* bietet sich dazu die Polynom-Interpolierende mit Darstellung in der Lagrange-Basis an:

$$\begin{aligned} f(x) \cong \quad & p_n(x) && = \sum_{j=0}^{n} L_j(x) f(x_j) && \stackrel{?}{\Longrightarrow} \\ \int_a^b f(x)dx \cong \quad & \int_a^b p_n(x)dx && = \sum_{j=0}^{n} \Big(\underbrace{\int_a^b L_j(x)dx}_{=:\alpha_j} \Big) f(x_j) && = \sum_{j=0}^{n} \alpha_j f(x_j). \end{aligned} \tag{2.2.1}$$

Die Näherungen für das Integral ist hier einfach eine Linearkombinationen von Funktionswerten.

Bekanntlich ist Integrierbarkeit eine schwache Einschränkung an Funktionen. Allerdings hängen Fehleraussagen der Polynominterpolation von Ableitungen der Funktion ab. Daher ist es vorteilhaft, Integranden mit Singularitäten als Produkt einer glatten Funktion f und einer festen, schwierigen *Gewichtsfunktion* $g(x) > 0$ (z.B. $\sqrt{x}, 1/\sqrt{x}, ..$) aufzufassen.

Definition 2.2.1 *Falls das Integral* $\int_a^b f(x)g(x)dx$ *existiert, heißt*

$$\sum_{j=0}^{n} \alpha_j f(x_j) = \int_a^b f(x)g(x)dx - R_n(f) \tag{2.2.2}$$

Quadraturformel *und* $R_n(f)$ *der* Quadraturfehler *bzw. das Restglied. Die Koeffizienten* α_j *heißen* Gewichte *zu den* Knoten

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

Die Formel (2.2.2) besitzt die Ordnung $m \in \mathbb{N}$*, wenn gilt* $R_n(p) = 0 \quad \forall p \in \Pi_{m-1}$.

Für die speziellen Quadraturformeln (2.2.1) aus der Integration des Interpolationspolynoms aus Π_n hat man explizit die Gewichte

$$\alpha_j = \int_a^b L_j(x)g(x)dx, \;\; j = 0, \ldots, n. \tag{2.2.3}$$

Die Darstellung (2.1.15) des Interpolationsfehlers, $r_n = f - p_n = \omega_{n+1}(x) f[x_0, \ldots, x_n, x]$ mit dem Knotenpolynom $\omega_{n+1}(x) = (x - x_0) \cdots (x - x_n)$ führt für den Quadraturfehler auf die Formel

$$\begin{aligned} R_n(f) &= \int_a^b f(x)g(x)dx - \sum_{j=0}^{n} \alpha_j f(x_j) = \int_a^b r_n(x)g(x)dx \\ &= \int_a^b \omega_{n+1}(x) f[x_0, \ldots, x_n, x] g(x)dx. \end{aligned} \tag{2.2.4}$$

Für glattes $f \in C^{n+1}[a,b]$ bekommt man durch Betragsabschätzung die Schranke

$$|R_n(f)| \le \int_a^b |\omega_{n+1}(x)| g(x) dx \; \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty. \tag{2.2.5}$$

Satz 2.2.2 *Für die Ordnung m einer* interpolatorischen *Quadraturformel (2.2.2), (2.2.3) gilt* $m \ge n+1$.

Die Betragsschranke lässt sich verschärfen, wenn das Knotenpolynom ω_{n+1} im Intervall keine Vorzeichenwechsel besitzt, wenn also

$$\omega_{n+1}(x) = (x-x_0)\cdots(x-x_n) \ge 0 \;\forall x \in [a,b] \quad (\text{bzw. } \le 0 \;\forall..). \tag{2.2.6}$$

Nach dem Mittelwertsatz der Integralrechnung gibt es dann Zwischenstellen $\xi_1, \xi_2 \in (a,b)$ mit

$$R_n(f) = \int_a^b \omega_{n+1}(x) g(x) dx \cdot f[x_0, \ldots, x_n, \xi_1] = \varrho_n \cdot f^{(n+1)}(\xi_2), \tag{2.2.7}$$

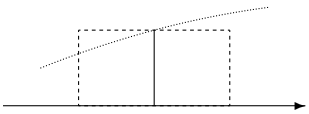
mit der Fehlerkonstanten

$$\varrho_n := \frac{1}{(n+1)!} \int_a^b \omega_{n+1}(x) g(x) dx.$$

Newton-Cotes-Formeln

Für eine spezifische Quadraturformel sind Gewicht g und Knoten zu wählen. Mit trivialem Gewicht $g \equiv 1$ und einfachen *äquidistanten* 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") bekommt man die *Newton-Cotes-Formeln*. Für die einfachsten Spezialfälle kann man dabei mit Zusatzüberlegungen die Fehlerdarstellung (2.2.7) einsetzen. In den folgenden Formeln steht daher ξ jeweils für eine unbekannte Zwischenstelle in (a,b).

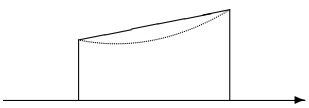
Rechteckregel: $n = 0$, offene Formel, Ordnung 2:



$$\int_a^b f(x)dx = (b-a)\, f\Big(\frac{a+b}{2}\Big) + \frac{(b-a)^3}{24} f''(\xi). \tag{2.2.8}$$

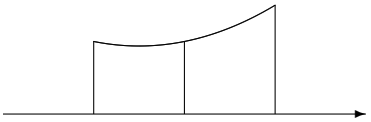
Elementargeometrisch: die Rechteckfläche ist Breite $(b-a)$ mal Höhe $f(\frac{a+b}{2})$, aber auch jedes Trapez mit x-Achse und einer Geraden durch $(\frac{a+b}{2}, f(\frac{a+b}{2}))$ als Seitenlinien hat die gleiche Fläche!

Trapezregel: $n = 1$, abgeschlossene Formel, Ordnung 2:



$$\int_a^b f(x)dx = \frac{b-a}{2}\Big[f(a)+f(b)\Big] - \frac{(b-a)^3}{12} f''(\xi). \tag{2.2.9}$$

Simpsonregel: $n = 2$, abgeschlossene Formel, Ordnung 4 (Keplersche Fassregel):



$$\int_a^b f(x)dx = \frac{b-a}{6}\Big[f(a) + 4f(\frac{a+b}{2}) + f(b)\Big] - \frac{(b-a)^5}{2880} f^{(4)}(\xi). \tag{2.2.10}$$

Eine Tabelle der Gewichte α_j und der Fehlerkoeffizienten ϱ_n für weitere Ordnungen folgt weiter unten. Zunächst wird aber nachgeprüft, dass bei der Rechteck- und Simpsonregel die Ordnung tatsächlich $m = n + 2$ ist statt $n + 1$, wie nach Satz 2.2.2 zu erwarten.

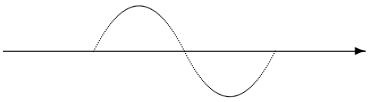
Beweis von (2.2.10): Für $a = 0$ (oBdA) sind die Lagrangepolynome zu $x_0 = 0,\ x_1 = h,\ x_2 = 2h$:

$$L_0(x) = \frac{(x-h)(x-2h)}{2h^2},\ L_1(x) = \frac{x(x-2h)}{-h^2},\ L_2(x) = \frac{x(x-h)}{2h^2} \quad \in \Pi_2.$$

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 (2.2.5) bzw. (2.2.7) stimmen allerdings noch nicht mit der in (2.2.10) überein. In einer Zusatzüberlegung wird eine *zusätzliche* Interpolationsbedingung $p'(x_1) = f'(x_1)$ herangezogen. Diese Zusatzinformation geht aber überhaupt nicht ein, da ihr Gewicht



$$\int_0^{2h} \omega_3(x)dx = \int_0^{2h} x(x-h)(x-2h)dx = 0 \tag{2.2.11}$$

null ist. Denn in der Newton-Darstellung gilt 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 somit

$$\int_0^{2h} p_3(x)dx = \int_0^{2h} p_2(x)dx + \underbrace{\int_0^{2h} \omega_3(x)dx}_{=0} f[x_0, x_1, x_1, x_2] = \alpha_0 f(x_0) + \alpha_1 f(x_1) + \alpha_2 f(x_2).$$

Also sind die Integrale über p_2 und p_3 gleich, sogar kubische Polynome werden exakt integriert, und es gilt (2.2.4) sogar mit dem Polynomgrad $n = 3$ und auch (2.2.7) 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 dort $n + 2$ statt $n + 1$. Die folgende Tabelle enthält die Gewichte, Fehlerkonstanten und Ordnungen der Newton-Cotes-Formeln. Da die Gewichte i.w. rational sind, erfolgt die Angabe in der Form $\alpha_j = (b-a)\beta_j/\gamma$,

$$\int_a^b f(x)dx = \frac{b-a}{\gamma}\sum_{j=0}^{n} \beta_j f(x_j) + c\Big(\frac{b-a}{n}\Big)^{m+1} f^{(m)}(\xi),\ \xi \in (a,b). \tag{2.2.12}$$

Fehlende Koeffizienten $\beta_5, \ldots, \beta_8$ sind symmetrisch zu ergänzen. Die Gewichte für $n \ge 8$ sind nicht mehr positiv. Dies erhöht die Anfälligkeit der Formeln für Rundungsfehler geringfügig.

Abgeschlossene Newton-Cotes-Formeln, $x_j = a + jh,\ j = 0, \ldots, n,\ h = \frac{b-a}{n}$.

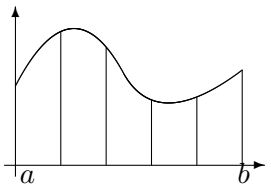
n	γ	β_0	β_1	β_2	β_3	β_4	c	m
1	2	1	1				-1/12	2
2	6	1	4	1			-1/90	4
3	8	1	3	3	1		-3/80	4
4	90	7	32	12	32	7	-8/945	6
5	288	19	75	50	50	75	-275/12096	6
6	840	41	216	27	272	27	-9/1400	8
7	17280	751	3577	1323	2989	2989	-8183/518400	8
8	28350	989	5888	-928	10496	-4540	-2368/467775	10

Für die Konvergenz ($n \to \infty$) der Quadraturformel (2.2.12) 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 wird der Fehler

$$R_n(f) = c \left(\frac{b-a}{n} \right)^{m+1} f^{(m)}(\xi)$$

sehr schnell klein für $(b-a) \to 0$. Dazu kann man die Additivität des Integrals bei Teilintervallen ausnutzen durch *Unterteilung* des Gesamtintervalls, man wendet eine Quadraturformel fester Ordnung *iteriert* an. Bei der Trapezregel (2.2.9), z.B., ergibt sich mit Teilintervallen $[x_{i-1}, x_i],\ x_j = a + jh,\ j = 0, \ldots, n,\ h := (b-a)/n$, die einfach aufgebaute Näherung

$$\begin{aligned} \int_a^b f(x)dx &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx \cong \sum_{i=1}^{n} \frac{h}{2} \Big[f(x_i) + f(x_{i-1}) \Big] \\ &= \frac{h}{2} \Big[f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n) \Big] =: T_h(f). \end{aligned}$$



mit folgender Fehleraussage.

Satz 2.2.3 *Für $n \in \mathbb{N}$ sei $h := (b-a)/n,\ x_i = a + ih,\ f_i := f(x_i),\ i = 0, \ldots, n$. Dann gilt mit einer Zwischenstelle $\xi \in (a, b)$*

a) für die iterierte Trapezregel bei $f \in C^2[a, b]$:

$$T_h(f) = \frac{h}{2} \Big(f_0 + 2f_1 + \ldots + 2f_{n-1} + f_n \Big) = \int_a^b f(x)dx + \frac{b-a}{12} h^2 f''(\xi), \tag{2.2.13}$$

b) für die iterierte Simpsonregel bei $f \in C^4[a, b]$ und n gerade:

$$S_h(f) := \frac{h}{3} \Big(f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 4f_{n-1} + f_n \Big) = \int_a^b f(x)dx + \frac{b-a}{180} h^4 f^{(4)}(\xi). \tag{2.2.14}$$

Beweis Die Beweise von a) und b) verlaufen analog. Bei b) wird die Formel (2.2.10) über die $\frac{n}{2} =: k$ Intervalle $[x_{2i-2}, x_{2i}]$ summiert. Beim Restglied erhält man dabei mit $\xi_i \in (x_{2i-2}, x_{2i})$

$$-\frac{1}{2880} \sum_{i=1}^{k} (x_{2i} - x_{2i-2})^5 f^{(4)}(\xi_i) = -\frac{32}{2880} h^5 \sum_{i=1}^{k} f^{(4)}(\xi_i) = -\frac{b-a}{180} h^4 f^{(4)}(\xi). \qquad \blacksquare$$

Bemerkung: Man beachte, dass auch die iterierten Formeln (2.2.13), (2.2.14) die Gestalt (2.2.2) aus Definition 2.2.1 besitzen, sogar mit positiven Gewichten, aber nicht in der Form (2.2.3).

Durch die Summation über die Teilintervalle geht eine der h-Potenzen im Fehler verloren und man sieht, dass es in (2.2.12) sinnvoll war als Ordnung der Formel die Zahl m und nicht $m+1$ anzugeben. Ein großer praktischer Vorteil der iterierten Trapez- und Simpsonregel (z.B. gegenüber der Rechteckregel) ist, dass man bei einer Verdopplung der Intervallzahl die alten Funktionswerte wiederverwenden kann, sogar in sehr einfacher Form. Z.B. gilt für die Trapezregel

$$T_{h/2}(f) = \frac{1}{2}T_h(f) + \frac{b-a}{2n}\sum_{j=1}^{n} f(a + \frac{b-a}{2n}(2j-1)). \tag{2.2.15}$$

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 gilt

$$R_n(f) \sim \frac{1}{n^2} \text{ (Trapezregel)} \quad \text{bzw.} \quad R_n(f) \sim \frac{1}{n^4} \text{ (Simpsonregel)}$$

bei einem genügend glatten 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 2.2.4 $\int_1^2 \frac{dx}{x} = \ln 2 = 0.6931472...$

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 macht es interessant, mit einer festen Knotenzahl $n+1$ eine möglichst hohe Ordnung zu erreichen. Dies erreicht man bei der folgenden Klasse von Quadraturformeln.

Gauß-Quadratur

Bei geradem Polynomgrad n hatten die Newton-Cotes-Formeln eine erhöhte Konvergenzordnung, da der erste Fehlerterm im Newton-Polynom verschwindet wegen der Eigenschaft

$$\int_a^b \omega_{n+1}(x)dx = 0$$

des Knotenpolynoms $\omega_{n+1} = (x-x_0)\cdots(x-x_n)$. Das Prinzip lässt sich verallgemeinern, durch geeignete Wahl der Knoten x_i kann man weitere Fehlerterme eliminieren und die Ordnung sogar verdoppeln. Dann gilt für den Quadraturfehler (2.2.2)

$$R_n(f) = 0 \quad \forall f \in \Pi_{2n+1}. \tag{2.2.16}$$

Zur Herleitung sei nun $f \in \Pi_{2n+1}$ und $p_n(x) = \sum_{i=0}^{n} f(x_i)L_i(x)$ das Interpolationspolynom dazu in Π_n. Also gilt hier

$$f = p_n + \omega_{n+1} \cdot q_n \quad \text{mit } q_n \in \Pi_n.$$

Die Forderung (2.2.16) entspricht somit 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 (2.2.16) offensichtlich genau dann erfüllt, wenn

$$\int_a^b \omega_{n+1}(x)q(x)g(x)dx = 0 \quad \forall q \in \Pi_n. \tag{2.2.17}$$

Prinzipiell gilt dazu: Für eine auf (a, b) positive Gewichtsfunktion g stellt die *Bilinearform*

$$(u, v)_g := \int_a^b u(x)v(x)\,g(x)dx \tag{2.2.18}$$

ein *Innenprodukt* in $C[a, b]$ dar. Die Forderung (2.2.17) entspricht daher der Orthogonalität

$$\omega_{n+1} \perp_g \Pi_n \quad \Longleftrightarrow \quad (\omega_{n+1}, q)_g = 0 \;\forall\, q \in \Pi_n. \tag{2.2.19}$$

Ordnung $2n + 2$ erreicht man also in (2.2.16), *wenn ω_{n+1} g-orthogonal zu allen Polynomen vom Grad n ist!*

Die Konstruktion einer Familie von Orthogonalpolynomen $\omega_k,\ k \in \mathbb{N}$, zum Innenprodukt (2.2.18) lässt sich einfach mit dem Gram-Schmidt-Orthogonalisierungsverfahren durchführen ausgehend von der Monom-Basis $\{1, x, x^2, \ldots\}$. Diese Orthogonalpolynome besitzen glücklicherweise nur einfache, reelle Nullstellen im Intervall (a, b), die als Knoten $x_0, \ldots, x_n$ einer Quadraturformel verwendbar sind. Der nächste Satz fasst diese Aussagen zusammen, später folgt ein Überblick über einige wichtige Klassen von Orthogonalpolynomen.

Satz 2.2.5 *Die Stützstellen $x_i,\ i = 0, \ldots, n$, der Quadraturformel (2.2.2), (2.2.3) seien die Nullstellen des Orthogonalpolynoms vom Grad $n+1$ zur Gewichtsfunktion $g \in C[a, b]$ mit $g > 0$ in (a, b), es gelte also $\Pi_{n+1} \ni \omega_{n+1} \perp_g \Pi_n$. Dann besitzt diese* Gauß-sche Quadraturformel *die* Ordnung $2n+2$, *es ist $R_n(f) = 0\ \forall f \in \Pi_{2n+1}$. Für Integranden $f \in C^{2n+2}[a, b]$ gilt mit $\xi \in (a, b)$ die Fehleraussage*

$$R_n(f) = \int_a^b f(x)g(x)dx - \sum_{i=0}^{n} \alpha_i f(x_i) = \frac{1}{(2n+2)!} \int_a^b \omega_{n+1}^2(x)g(x)dx\, f^{(2n+2)}(\xi). \tag{2.2.20}$$

Bemerkung: Gegenüber den Newton-Cotes-Formeln erhält man also ungefähr die doppelte Konvergenzordnung ohne Mehraufwand. Bei iterierter Anwendung gleicht sich dieser Vorteil aber teilweise aus, da man mit Gaußformeln bei Verdopplung der Intervallzahl alte Funktionswerte nicht wiederverwenden kann. Als Bestandteile anderer numerischer Verfahren, z.B. der sogenannten Finite-Element-Methode, können Gaußformeln aber sehr effizient sein.

Beweis Hier ist nur noch zu zeigen, dass das Restglied die von (2.2.7) abweichende Form hat. Analog zum Beweis bei der Simpsonregel (2.2.10) werden formal die zusätzlichen Interpolationsdaten $f'(x_0), \ldots, f'(x_n)$ benutzt. Das Interpolationspolynom $p_{2n+1} \in \Pi_{2n+1}$ zu den Hermite-Bedingungen $p_{2n+1}(x_i) = f(x_i)$, $p'_{2n+1}(x_i) = f'(x_i),\ i = 0, \ldots, n$, hat im Vergleich zum einfachen Polynom $p_n \in \Pi_n$ mit $p_n(x_i) = f(x_i)$, $i = 0, \ldots, n$, in der Newton-Darstellung die Gestalt

$$\begin{aligned} p_{2n+1}(x) &= p_n(x) + \underbrace{\omega_{n+1}(x)}_{\int \ldots = 0} f[x_0, .., x_n, x_0] + \underbrace{\omega_{n+1}(x)(x - x_0)}_{\int \ldots = 0} f[x_0, .., x_n, x_0, x_1] + \\ &\quad \ldots + \underbrace{\omega_{n+1}(x)(x - x_0) \cdots (x - x_{n-1})}_{\int \ldots = 0} f[x_0, \ldots, x_n, x_0, \ldots, x_n]. \end{aligned}$$

Außerdem gilt nach (2.1.15) für dessen Fehler

$$f(x) = p_{2n+1}(x) + \omega_{n+1}^2(x) f[x_0, \ldots, x_n, x_0, \ldots, x_n, x]. \tag{2.2.21}$$

Aufgrund der Orthogonalität (2.2.19) fallen im Integral aber die markierten Zusatzterme weg:

$$\int_a^b \omega_{n+1}(x)(x - x_0) \cdots (x - x_k) g(x) dx = 0 \ \forall k < n \quad \Rightarrow$$

$$\int_a^b p_{2n+1}(x) g(x) dx = \int_a^b p_n(x) g(x) dx = \sum_{i=0}^{n} \alpha_i f(x_i). \tag{2.2.22}$$

Aus (2.2.21) folgt die Aussage (2.2.20), da die Fehlerformel (2.2.7) wegen $\omega_{n+1}^2 \geq 0$ jetzt auf p_{2n+1} angewendet werden kann. ∎

Bemerkung: In (2.2.22) 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: $\beta_i \equiv 0$.

Je nach Art der im Integranden abgespaltenen Singularität ($\rightarrow$ Gewichtsfunktion) erhält man verschiedene Polynomfamilien (vgl. folgende Tabelle). Analog zu den Tschebyscheff-Polynomen aus Defin. 2.1.12 gelten auch für diese jeweils Drei-Term-Rekursionen. Die wichtigste Familie zur Gewichtsfunktion $g \equiv 1$ ist die der Legendrepolynome. Zur Vereinfachung wird meist das Standardintervall $[-1, 1]$, bzw. bei uneigentlichen Integralen $[0, \infty)$, zugrundegelegt. Andere Intervalle werden durch Variablensubstitution darauf zurückgeführt.

Polynom-Orthogonalfamilien nach Satz 2.2.5:

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.	$H_{n+1} = 2xH_n - 2nH_{n-1},\ H_0 = 1,\ H_1 = 2x$

Die angegebenen Polynomfamilien haben eine erhebliche, weitergehende Bedeutung v.a. im Zusammenhang mit Reihenentwicklungen für Lösungen von Differentialgleichungen. Daher gibt

es eine umfangreiche Literatur (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

$$P_n(x) = \frac{n!}{(2n)!}\,\frac{d^n}{dx^n}(x^2-1)^n, \quad n = 0, 1, \dots \ . \tag{2.2.23}$$

Durch partielle Integration kann man mit dieser Formel sofort die Orthogonalitätsbedingungen (2.2.17) verifizieren. Aus der Darstellung (2.2.23) folgt übrigens direkt die Existenz von n einfachen reellen Nullstellen in $(-1, 1)$ nach dem Satz von Rolle.

Zum Vergleich der verschiedenen Quadraturverfahren dient folgendes

Beispiel 2.2.6 $\int_{-1}^{1} e^x dx = e^1 - e^{-1} = 2\sinh 1 = 2.3504024$. Näherungen durch

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 i.w. das Legendrepolynom $\omega_2(x) = x^2 - \frac{1}{3} = \frac{2}{3}P_2(x)$. Dieses besitzt die 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

$$f(x_0) + f(x_1) = 2\cosh(x_1) = 2.342696.. \quad \text{Fehler} = -0.0077..$$

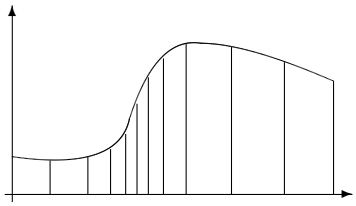
Adaptive Integration

In der Grundform sind die Verfahren recht unhandlich für den praktischen Einsatz. Denn bei gegebenem Integranden sind das Quadraturverfahren und dessen Parameter (Schrittweite h, Ordnung) zu wählen, um den Integralwert mit einer gewünschten Genauigkeit bzw. Toleranz ε möglichst effizient zu berechnen. Falls man den Integranden analytisch kennt, kann man im Prinzip aus der Restglieddarstellung (2.2.5) die Ordnung der Formel und die Zahl der Teilintervalle so bestimmen, dass die Fehlerschranke den Wert ε nicht überschreitet (vgl. Übungen).

Diese Methode ist aufwändig und oft nicht durchführbar. Man umgeht das Problem durch Verbindung der Quadraturformel mit einer Fehler(ab)schätzung. Bei der iterierten Trapezregel etwa gilt für eine Zerlegung in Teilintervalle $[x_{i-1}, x_i]$ in jedem Teil mit einer Zwischenstelle $\xi_i \in (x_{i-1}, x_i)$ die Darstellung:

$$\int_{x_{i-1}}^{x_i} f(x)dx - \frac{1}{2}(x_i - x_{i-1})(f(x_{i-1}) + f(x_i)) = -\frac{1}{12}(x_i - x_{i-1})^3 f''(\xi_i) =: R[x_{i-1}, x_i]. \quad (2.2.24)$$

Der Fehler zu einem einzelnen Teilintervall hängt also nur von der aktuellen Schrittweite $x_i - x_{i-1}$ und dem *lokalen* Wert der zweiten Ableitung zusammen.



Eine effiziente Strategie bei der Quadratur hält die vorgegebene Toleranz mit möglichst wenigen Funktionsauswertungen ein. Im Bereich kleiner Ableitungen kann dazu mit großen Schrittweiten vorgegangen werden, während bei großen Ableitungswerten feinere Unterteilungen nötig sind. Diese erreicht man mit Teil-Intervallen $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, durch *proportionale Gleichverteilung* des Fehlers

$$\Big|R[x_{i-1}, x_i]\Big| \overset{!}{\leq} \frac{x_i - x_{i-1}}{b-a}\varepsilon, \quad i = 1, \ldots, n. \quad (2.2.25)$$

Dann gilt nämlich

$$\left|\int_a^b f(x)dx - \sum_{i=1}^n \frac{x_i - x_{i-1}}{2}[f(x_{i-1}) + f(x_i)]\right| \leq \sum_{i=1}^n |R[x_{i-1}, x_i]| \leq \sum_{i=1}^n \frac{x_i - x_{i-1}}{b-a}\varepsilon = \varepsilon. \quad (2.2.26)$$

Zur Anwendung der Strategie (2.2.25) gibt es zwei Anforderungen:

a) (Ab-) Schätzung des lokalen Fehlers $R[x_{i-1}, x_i]$.

b) Konstruktion einer Unterteilung, die (2.2.25) erfüllt.

Zu a): Außer bei einfachen Integranden, bei denen $\|f''\|_{[x_{i-1},x_i]}$ explizit abgeschätzt werden kann, 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

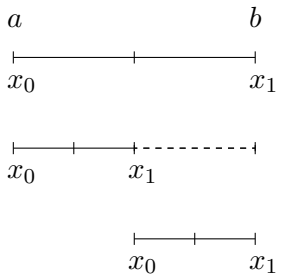
$$f(x_{i-1}) - 2f\Big(\frac{x_{i-1} + x_i}{2}\Big) + f(x_i) = \frac{1}{4}h_i^2 f''(\zeta_i) \cong -\frac{3}{h_i}R[x_{i-1}, x_i]. \quad (2.2.27)$$

Dies folgt aus den Eigenschaften dividierter Differenzen oder durch Taylorentwicklung um $\frac{1}{2}(x_{i-1} + x_i)$. Mit der Approximation (2.2.27) wird das Kriterium (2.2.25) *ersetzt* durch

$$\Big|f(x_{i-1}) - 2f\Big(\frac{x_{i-1} + x_i}{2}\Big) + f(x_i)\Big| \overset{!}{\leq} \frac{3\varepsilon}{b-a}. \quad (2.2.28)$$

Zu b): Ein Gitter mit Eigenschaft (2.2.28) konstruiert man adaptiv, etwa in folgender Weise:

- wähle $x_0 := a,\ x_1 := b$;
- Falls (2.2.28) verletzt ist, halbiere das Intervall, setze $x_1 := (x_0 + x_1)/2$;
- Falls (2.2.28) 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 $x_0 := x_1,\ x_1 := b$.

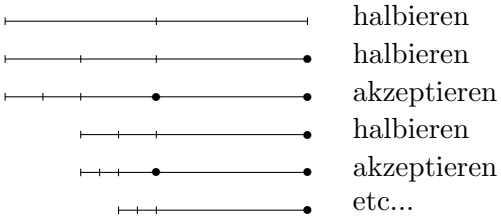


Algorithmus 2.2.7 Einfache adaptive Quadratur

$x0 := a;\ f0 := f(x0);\ x1 := b;\ f1 := f(x1);\ w := 0;\ e := 3\varepsilon/(b-a);$
wiederhole
  $xm := 0.5 * (x0 + x1);\ fm := f(xm);$
  falls $abs(f0 - 2 * fm + f1) <= e$ dann { //akzeptieren
    $w := w + 0.5 * (x1 - x0) * (f0 + f1);$
    $x0 := x1;\ f0 := f1;\ x1 := b;\ f1 := f(x1)\}$
  sonst//halbieren
  { $x1 := xm;\ f1 := fm$ }
bis $x0 >= b$;

In der Praxis sind Funktionsauswertungen zu kostbar, um sie im Falle eines nicht akzeptierten Teilintegrals zu verwerfen. Wenn man die Teilintervallängen auf feste Bruchteile $(b-a)2^{-j}, j \in \mathbb{N}$, einschränkt kann man durch einfache Zusatzbedingungen 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.

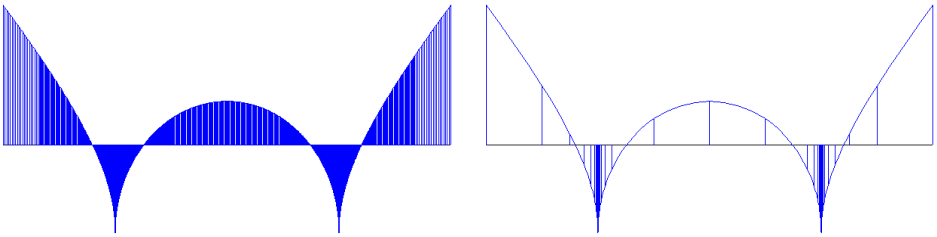
Beispiel:



Schwarz markierte
Gitterwerte stehen im Keller

Professionelle Programme steuern lokal außer der Schrittweite auch die Ordnung der Formel.

Demo-Beispiel: Adaptive Integration von $\int_{-1}^{1} \left(\sqrt{|x^2 - 0.25|} - \frac{1}{3}\right) dx$.



Die Bilder zeigen nur die bei der Quadratur verwendeten Trapeze, bilden aber den Funktions-

verlauf sehr genau nach. Bei Toleranz $\varepsilon = 10^{-6}$ benötigt die Trapezregel (linke Graphik) für das Beispiel 2837 Funktionswerte, während die Simpsonregel (rechts) mit nur 241 Werten auskommt. Beide unterschätzen hier aber den exakten Fehler, der bei der Trapezregel 1.2ε und bei Simpson 5ε ist.

Bemerkung: Bei der adaptiven Quadratur wurde die Kenntnis der Fehlerstruktur der Trapezregel dazu verwendet eine geeignete Intervallunterteilung zu konstruieren. Die Fehlerstruktur der iterierten Trapezregel kennt man aber noch genauer. Für einen genügend oft diffbaren Integranden f besitzt $T_h(f)$ eine sog. *asymptotische Entwicklung* nach h^2-Potenzen, d.h., es gibt Konstanten $a_i \in \mathbb{R}$ mit

$$T_h(f) = \int_a^b f(x)\,dx + a_2h^2 + a_4h^4 + \ldots + a_{2q}h^{2q} + \mathcal{O}(h^{2q+1}),\ h \to 0. \tag{2.2.29}$$

Durch Linearkombination von Trapezapproximationen zu verschiedenen Schrittweiten h_i kann man Fehlerterme eliminieren und verbesserte Näherungen erzeugen. Eliminiert man etwa den Fehlerterm a_2h^2 mit den Schrittweiten h und $h/2$ bekommt man mit

$$\hat{T}_{h/2}(f) := \frac{4}{3}T_{h/2}(f) - \frac{1}{4}T_h(f)$$

eine Integralnäherung der Ordnung 4. Diese entspricht übrigens genau der Simpsonregel, es gilt $\hat{T}_{h/2}(f) = S_{h/2}(f)$. Dieses Verfahren der sogenannten *Richardson-Extrapolation* kann man wiederholen und es führt mit fortgesetzten Schrittweitenhalbierungen $h_i = h/2^i$ auf die *Romberg-Integration*. Man bekommt so eine einfache Methode, um Quadraturnäherungen sehr hoher Ordnung zu konstruieren. Allerdings sind diese im Vergleich zu anderen Formeln ineffizient, da sie erheblich mehr f-Auswertungen benötigen, auch in verbesserten Varianten.

2.3 Interpolation mit Spline-Funktionen

Wie bei der Quadratur kann man das Problem mit der Konvergenz von Polynomen wachsenden Grads auch bei der Interpolation umgehen durch Unterteilung des Gesamtintervalls. Bei Interpolation einer Funktion $f \in C^{k+1}[\alpha,\beta]$ durch ein Polynom p_k vom Grad k mit Stützstellen $\xi_0,\ldots,\xi_k \in [\alpha,\beta]$ hat der Fehler nach (2.1.17) die Gestalt ($x \in [\alpha,\beta]$)

$$|f(x)-p_k(x)| \le |(x-\xi_0)\cdots(x-\xi_k)|\frac{\|f^{(k+1)}\|_\infty}{(k+1)!} \le \frac{(\beta-\alpha)^{k+1}}{(k+1)!}\|f^{(k+1)}\|_\infty. \tag{2.3.1}$$

Für wachsenden Grad k kann man *i.a. keine Konvergenz* $\|f-p_k\|_\infty \to 0$ garantieren, aus (2.3.1) folgt aber trivialerweise

$$|f-p_k| \to 0 \quad \text{für} \quad (\beta-\alpha)\to 0,\ k \text{ fest.}$$

Dies nutzt man wieder aus durch Unterteilung des Gesamtintervalls $[\alpha,\beta]$ mit einem Gitter

$$\begin{aligned} &\Delta: \quad \alpha = x_0 < x_1 < \ldots < x_n = \beta, \\ &h_i := x_{i+1}-x_i,\ i=0,\ldots,n-1, \quad H := \max_{i=0}^{n-1} h_i, \end{aligned} \tag{2.3.2}$$

und approximiert die Funktion f in den Teilintervallen $[x_i,x_{i+1}]$ durch Polynome *festen Grades*.

Definition 2.3.1 *Zu einem Gitter Δ nach (2.3.2) und $m,k \in \mathbb{N}_0,\ 0 \le m < k$, bezeichnet S_k^m den Raum aller* Spline-*Funktionen s, die*

a) lokal, in jedem Teilintervall $(x_i,x_{i+1}]$, Polynome vom Grad k sind: $s|_{(x_i,x_{i+1}]} \in \Pi_k$,
b) global m-mal stetig differenzierbar sind: $s \in C^m[\alpha,\beta]$.

Der wichtigste Fall ist der der *kubischen Splines* ($k=3$) mit $m \in \{1,2\}$. Splines werden außer zur Approximation von Meßdaten hauptsächlich zur Kurven- und Flächendarstellung in der Computer-Graphik (CAD) und als Finite-Elemente-Methode (FEM) zur Lösung von partiellen Differentialgleichungen eingesetzt. Zur Interpolation wird definiert:

Definition 2.3.2 *Mit einem Gitter Δ nach (2.3.2) und $f \in C^1[\alpha,\beta]$ ist die Funktion $s \in S_3^2$ der kubische* Interpolationsspline*, wenn sie die Interpolationsbedingungen*

$$s(x_i) = f(x_i),\ i=0,1,\ldots,n, \tag{2.3.3}$$

erfüllt und die Randbedingung

$$s'(\alpha) = f'(\alpha),\ s'(\beta) = f'(\beta). \tag{2.3.4}$$

Zur Darstellung von Splines sind zwei Methoden verbreitet. Die *Bézier*-Darstellung (2.3.5) ist flexibler, wenn sowohl C^2- als auch C^1- oder C^0-Splines benutzt werden. Da diese Darstellung bei C^2-Splines unnötig viele Parameter verwendet, sind für S_3^2 aber *B-Splines* günstiger. Die

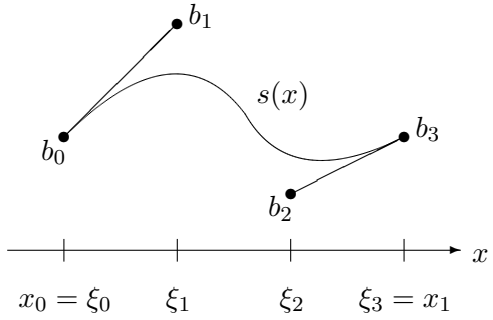
Bézier-Bernstein-Darstellung von kubischen Polynomen auf dem Standardintervall $[0,h]$, $h>0$, lautet

$$s(x) = \frac{1}{h^k}\sum_{j=0}^{k}\binom{k}{j}x^j(h-x)^{k-j}\,b_j \tag{2.3.5}$$

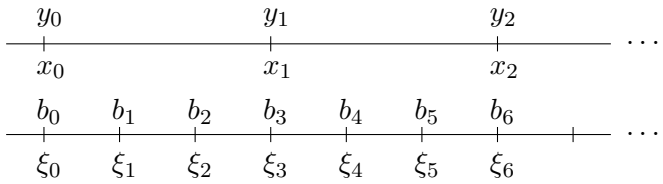
mit Koeffizienten $b_0,\ldots,b_k$. Diese Darstellung besitzt folgende Eigenschaften:

$$\left.\begin{array}{ll}
(a) & b_j\equiv 1 \;\Rightarrow\; s(x)=\frac{1}{h^k}\sum_{j=0}^{k}\binom{k}{j}x^j(h-x)^{k-j}=\frac{1}{h^k}(x+h-x)^k\equiv 1\\
(b) & s(0)=b_0,\quad s(h)=b_k\\
(c) & s'(x)=\frac{1}{h^k}[-k(h-x)^{k-1}b_0+k(h-kx)(h-x)^{k-2}b_1+x(\ldots)]\Rightarrow\\
 & s'(0)=\dfrac{b_1-b_0}{h/k},\quad s'(h)=\dfrac{b_k-b_{k-1}}{h/k}\\
(d) & s''(0)=\frac{k(k-1)}{h^2}(b_0-2b_1+b_2),\quad s''(h)=\frac{k(k-1)}{h^2}(b_{k-2}-2b_{k-1}+b_k).
\end{array}\right\} \tag{2.3.6}$$

Die Eigenschaft (c) führt bei $k=3$ auf eine einfache *geometrische* Interpretation der Koeffizienten b_j. Werden die Paare $(\xi_j,b_j)^\mathsf{T}$ mit $\xi_j=jh/3$ betrachtet, dann ist die Gerade $\overline{(\xi_0,b_0)^\mathsf{T}\;(\xi_1,b_1)^\mathsf{T}}$ die Tangente an s in $x=\xi_0=0$ und $\overline{(\xi_2,b_2)^\mathsf{T}\;(\xi_3,b_3)^\mathsf{T}}$ 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 $[\alpha,\beta]$ durch lokale Darstellungen (2.3.5) zusammengesetzt. Dazu wird ein Zusatzgitter eingeführt,



$$\begin{array}{c}\xi_{ik+j}:=x_i+\frac{j}{k}h_i,\\ j=0,\ldots,k-1,\\ i=0,\ldots,n-1,\end{array} \tag{2.3.7}$$

Die **Bézier-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 $\mathbb{R}^2$ bzw. $\mathbb{R}^3$ keine starke Einschränkung. In diesem Fall, $h_j\equiv h$, gibt es für die verschiedenen Stetigkeitsbedingungen einfache geometrische Interpretationen.

- C^0 ist erfüllt mit $b_{3j}=y_j$
- 1. Ableitung, deren Stetigkeit bedeutet nach (2.3.6,c)

$$s'(x_j-)=\frac{3}{h}(b_{3j}-b_{3j-1})\stackrel{!}{=}\frac{3}{h}(b_{3j+1}-b_{3j})=s'(x_j+)\quad\Longleftrightarrow$$

$$\frac{1}{2}(b_{3j-1}+b_{3j+1})=b_{3j}=y_j. \tag{2.3.8}$$

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

- 2. Ableitung: Nach (2.3.6,d) ist die Bedingung $s''(x_j-) \stackrel{!}{=} s''(x_j+)$ äquivalent mit

$$\frac{1}{h^2}(b_{3j-2} - 2b_{3j-1} + b_{3j}) = \frac{1}{h^2}(b_{3j} - 2b_{3j+1} + b_{3j+2}) \tag{2.3.9}$$

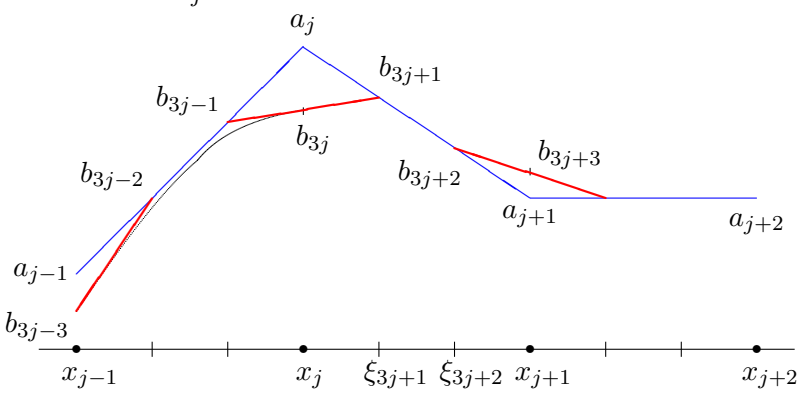
Hier fällt b_{3j} weg und es ist sinnvoll, spezielle Teilausdrücke als neue Parameter einzuführen. Aus (2.3.9) wird so:

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

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

$$\left.\begin{array}{r} 2b_{3j+1} - b_{3j+2} = a_j \\ -b_{3j+1} + 2b_{3j+2} = a_{j+1} \end{array}\right\} \quad \Rightarrow \quad \left\{\begin{array}{l} b_{3j+1} = \frac{1}{3}(2a_j + a_{j+1}) \\ b_{3j+2} = \frac{1}{3}(a_j + 2a_{j+1}) \end{array}\right. . \tag{2.3.10}$$

Diese Beziehungen der Bézier-Koeffizienten beim C^2-Spline erlauben folgende geometrische Interpretation (Hilfslinien: C^1 rot, C^2 blau), 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 (2.3.10) können auch die C^1-Bedingungen (2.3.8) noch umformuliert werden,

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

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

$$a_{j-1} + 4a_j + a_{j+1} = 6y_j, \quad j = 1, \ldots, n-1, \tag{2.3.11}$$

das durch die Randbedingungen (2.3.4) zu ergänzen ist:

$$2a_0 + a_1 = 3y_0 + hy'_0, \quad a_{n-1} + 2a_n = 3y_n - hy'_n \tag{2.3.12}$$

Denn nach (2.3.10) gilt $hs'(\alpha) = 3(b_1 - b_0) = 2a_0 + a_1 - 3y_0$. Die Bézier-Koeffizienten $\{b_\nu\}$ ergeben sich aus den $\{a_j\}$ nach (2.3.10).

Satz 2.3.3 *Gegeben sei ein Gitter Δ, (2.3.2), mit konstanter Schrittweite h. Dann existiert ein eindeutiger Interpolationsspline zu Definition 2.3.2 (Abkürzung $y_i = f(x_i)$, $y_i' = f'(x_i)$). Für den durch (2.3.11), (2.3.12) bestimmten Koeffizientenvektor a gilt $\|a\|_\infty \le 3\|y\|_\infty + h\max\{|y_0'|, |y_n'|\}$.*

Beweis Die Matrix des Gleichungssystems (2.3.11), (2.3.12) ist eine symmetrische Tridiagonalmatrix

$$A = \begin{pmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{2.3.13}$$

Wegen der Randbedingung (2.3.12) ist das erste und letzte Hauptdiagonalelment 2. Die Matrix A in (2.3.13) ist regulär, denn die Norm der Lösung lässt sich einfach abschätzen. Aus (2.3.11), (2.3.12) folgt

$$\begin{aligned} 4|a_j| - 2\|a\|_\infty \le |4a_j + a_{j-1} + a_{j+1}| &= 6|y_j| \le 6\|y\|_\infty \qquad \forall j = 1, \dots, n-1, \\ 2|a_j| - \|a\|_\infty = |2a_j + a_{j\pm1}| &= |3y_j \pm hy_j'| \le 3\|y\|_\infty + h\|y'\|_\infty,\ j \in \{0, n\}. \end{aligned}$$

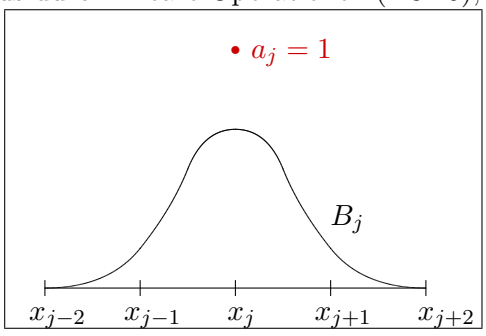
Man betrachtet nun die Ungleichung, in der die betrags-maximale Komponente $|a_k| = \|a\|_\infty$ auftritt. Im Fall $1 \le k < n$ bekommt man so $4\|a\|_\infty - 2\|a\|_\infty \le 6\|y\|_\infty$ und in Fall $k \in \{0, n\}$ analog $\|a\|_\infty \le 3\|y\|_\infty + h\|y'\|_\infty$. Die homogene Gleichung hat also nur die triviale Lösung $a = 0$, A ist daher regulär. ■

Die Eigenschaften der Matrix (2.3.13) werden in §4 wieder aufgegriffen. Dort zeigt sich auch, dass der Gauß-Algorithmus das Tridiagonalsystem (2.3.11) effizient auflösen kann mit dem geringen Aufwand $O(n)$. Die *Auswertung* eines Splines s in Bézier-Darstellung erfolgt, z.B., mit der Formel (2.3.5), nach (2.3.6a) ist sie eine konvexe Linearkombination (Betragssume der Vorfaktoren der b_j ist eins) und daher numerisch stabil.

B-Splines: Die wesentlichen Parameter des Splines s sind die Größen a_j. Die Koeffizienten b_i und Funktionswerte $s(x)$ des Splines berechnet man daraus durch lineare Operationen (2.3.10), (2.3.5). Daher existiert eine Basisdarstellung der Form

$$s(x) = \sum_{j=0}^{n} a_j B_j(x), \quad x \in [\alpha, \beta]. \tag{2.3.14}$$

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



Dieser Spline hat im Innern des Intervalls die im Bild gezeigte Gestalt (vgl. die geometrische Interpretation oben). 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. Zur Vereinfachung der Definition am Rand verwendet man ein erweitertes Gitter $\bar{\Delta}$ mit $2k = 6$ zusätzlichen Punkten, bei äquidistanten Punkten ist also $\bar{\Delta} = \{x_i = x_0 + ih : i = -3, \dots, n+3\}$. In diesem Fall sind die B-Splines verschobene Kopien eines Standardsplines, $B_j(x) = \hat{B}(x - x_j)$, aus dem sie durch Translation

hervorgehen. Es gilt die einfache Darstellung mit der gestreckten Variablen $\xi := x/h$

$$\hat{B}(x) = \begin{cases} \frac{1}{6}(\xi+2)^3 & \xi \in (-2,-1] \\ \frac{2}{3} - \xi^2 - \frac{1}{2}\xi^3 & \xi \in (-1,0], \\ \frac{2}{3} - \xi^2 + \frac{1}{2}\xi^3 & \xi \in (0,1], \\ \frac{1}{6}(2-\xi)^3 & \xi \in (1,2], \\ 0 & \xi \notin (-2,2]. \end{cases} \tag{2.3.15}$$

B-Splines kann man auf Gittern mit beliebigen Schrittweiten definieren. Auch diese allgemeinen B-Splines bilden eine sogenannte *lokale Zerlegung der Eins* (vgl. auch (2.3.6,a)) wodurch wichtige Beweisprinzipien einsetzbar sind. Die Eigenschaften dieser Zerlegung enthält der folgende, für (2.3.15) elementar nachprüfbare, Satz.

Satz 2.3.4 *Die B-Splines bilden eine* lokale Zerlegung der Eins*, es gilt:*

$$\begin{aligned} B_i(x) = 0 \quad & \forall x \notin (x_{i-2}, x_{i+2}), \\ B_i(x) > 0 \quad & \forall x \in (x_{i-2}, x_{i+2}), \\ \sum_{i=-1}^{n+1} B_i(x) = \sum_{i=j-1}^{j+2} B_i(x) \equiv 1, \ & \textit{wenn } x \in (x_j, x_{j+1}],\ 0 \le j < n. \end{aligned} \tag{2.3.16}$$

Die Eigenschaften (2.3.16) legen umgekehrt den B-Spline B_i eindeutig fest, er stimmt daher mit dem aus (2.3.15) überein. Für einen Spline in B-Spline-Darstellung kann ein stabiles Rekursionsschema angegeben werden, wobei bei jeder Auswertung höchstens 4 Koeffizienten a eingehen (vgl. Stoer, §2.4.5).

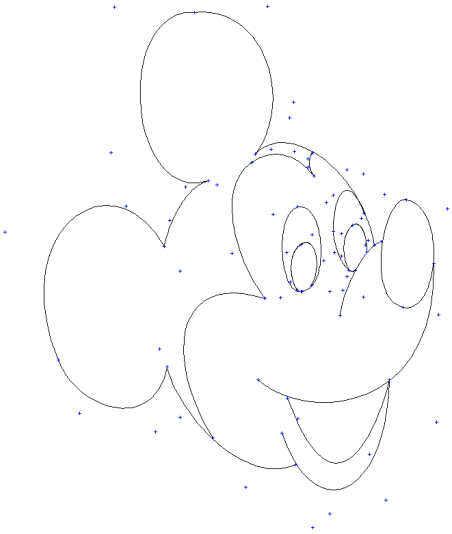
B-Spline- und Bézier-Darstellung bieten **praktische Vorteile** bei der Handhabung, die darauf beruhen, dass die benutzten Basisfunktionen die erwähnte *lokale Zerlegung der Eins* bilden:

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

Konsequenzen aus der Eigenschaft "Zerlegung der Eins":

- Bei Auswertung des Splines treten keine zusätzlichen Rundungsfehler auf, da sogar $\sum |B_i(x)| \equiv 1$ gilt: $|\sum_j a_j B_j - \sum_j \tilde{a}_j B_j| \le \max_j |a_j - \tilde{a}_j|$.
- Die Koeffizienten schließen, als Punkte der Ebene 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 2.3.5 $\mathbb{R}^2$-Graphik aus kubischen Bézier-Spline-Kurvenstücken mit Kontrollpunkten



Lokale Spline-Approximationen

Die Lösung des linearen Gleichungssystems (2.3.11) verursacht einen Rechenaufwand $O(n)$ beim Interpolationsspline. Mit geringfügigen Einbußen kann man auch dieses System noch einsparen, indem man die Koeffizienten a_j einer B-Spline-Darstellung (2.3.14) in geeigneter Weise explizit abhängig von f vorgibt. Dies führt auf *Lokale Spline-Approximationen* der Form

$$Q:\ f \mapsto s(x) := \sum_{j=-1}^{n+1} a_j(f) B_j(x), \tag{2.3.17}$$

auf dem erweiterten Gitter $\bar{\Delta}$, wobei $a_j(f)$ explizit mit Hilfe von f gegeben ist. Ein solches $s = Qf$ interpoliert f i.d.R. nicht mehr exakt, kann aber dennoch eine genaue Approximation sein. Dabei spielt die Eigenschaft aus Satz 2.3.4 (*lokale Zerlegung der Eins*) eine zentrale Rolle.

Beispiel 2.3.6 Die einfachste Approximation bekommt man mit $a_i(f) = f(x_i)$. Der Einfachheit halber wird Schrittweite $h = 1$ verwendet. Betrachtet man den Spline zur linearen Funktion $f(x) = x$ auf dem Interval $(0, 1]$, erhält man mit $B_i(x) = \hat{B}(x - i)$ und $\xi \equiv x$:

$$\begin{aligned}
s(x) &= a_{-1}(f)\hat{B}(x+1) + a_0(f)\hat{B}(x) + a_1(f)\hat{B}(x-1) + a_2(f)\hat{B}(x-2) \\
&= -\hat{B}\underbrace{(x+1)}_{\in(1,2]} + 0\cdot\hat{B}(x) + 1\cdot\hat{B}\underbrace{(x-1)}_{\in(-1,0]} + 2\cdot\hat{B}\underbrace{(x-2)}_{\in(-2,-1]} \\
&\overset{(2.3.15}{=} -\frac{1}{6}(1-x)^3 + (\frac{2}{3} - (x-1)^2 - \frac{1}{2}(x-1)^3) + \frac{2}{6}(x-2+2)^3 \\
&= -\frac{1}{6} + \frac{1}{3}x - \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{2}{3} - x^2 + 2x - 1 - \frac{1}{2}x^3 + \frac{3}{2}x^2 - \frac{3}{2}x + \frac{1}{2} + \frac{1}{3}x^3 \\
&= x.
\end{aligned}$$

Lineare Funktionen werden also exakt wiedergegeben. Dies ist wichtig im folgenden Satz 2.3.7.

Als Verallgemeinerung des Beispiels betrachtet man lineare Funktionale der Form

$$a_i(f) = \sum_{j=1}^{\ell} \alpha_{ij} f(\xi_{ij}), \quad \xi_{ij} \in [x_{i-2}, x_{i+2}]. \tag{2.3.18}$$

Dann ist nämlich die Abbildung Q aus (2.3.17) ein linearer Operator. Für (2.3.18) gilt zunächst

$$|a_i(f)| \le \sum_{j=1}^{\ell} |\alpha_{ij}| \sup\{|f(x)| :\ x \in [x_{i-2}, x_{i+2}]\} =: A_i \|f\|_{[x_{i-2}, x_{i+2}]}. \tag{2.3.19}$$

Damit bekommt man eine *lokale* Abschätzung für Qf, denn für $x \in (x_j, x_{j+1}]$ ist

$$|Qf(x)| \le \sum_{i=j-1}^{j+2} |a_i(f)| B_i(x) \le \max_{i=j-1}^{j+2} |a_i(f)| \le \max_{i=j-1}^{j+2} A_i \|f\|_{[x_{j-3}, x_{j+4}]}. \tag{2.3.20}$$

Nach einem einfachen Prinzip folgt aus dieser lokalen Schranke eine Konvergenzaussage.

Satz 2.3.7 *Zu einem erweiterten Gitter $\bar{\Delta}$ sei ein lokaler Approximationsoperator Q definiert, der (2.3.20) erfüllt. Wenn Q auf $[\alpha, \beta]$ Polynome vom Grad m reproduziert, d.h., $(Qp - p)|_{[\alpha,\beta]} = 0 \ \forall p \in \Pi_m$ ist, dann gilt für $f \in C^{m+1}[\alpha - d, \beta + d]$ die Fehlerschranke*

$$\|f - Qf\|_{[\alpha,\beta]} \le c\Big(1 + \max_{i=-1}^{n+1} A_i\Big) h^{m+1} \|f^{(m+1)}\|_{[\alpha-d,\beta+d]}, \ \ h \le d/3.$$

Beweis Nach Voraussetzung gilt mit beliebigen Polynomen $p \in \Pi_m$, dass

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

auf $[\alpha, \beta]$. Mit (2.3.20) folgt daraus für $x \in (x_j, x_{j+1}]$, $0 \le j < n$, dann

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

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

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

Die lokale Approximation aus Beispiel 2.3.6 erfüllt die Voraussetzung mit $m = 1$, der Fehler ist $O(h^2)$. Für eine bessere Approximation von f ergänzt man $f(x_i)$ durch Differenzen:

$$a_j(f) := f(x_j) - \frac{h^2}{3} f[x_{j-1}, x_j, x_{j+1}] = \frac{1}{6}\Big(- f(x_{j-1}) + 8f(x_j) - f(x_{j+1})\Big). \tag{2.3.21}$$

Für die Funktionswerte dieses lokalen Splines folgt aus (2.3.15) direkt

$$\begin{aligned} s(x_j) &= \frac{1}{6}(a_{j-1}(f) + 4a_j(f) + a_{j+1}(f)) \\ &= f(x_j) - \frac{1}{36}(f_{j-2} - 4f_{j-1} + 6f_j - 4f_{j+1} + f_{j+2}) \\ &= f(x_j) - \frac{2}{3} h^4 f[x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+2}]. \end{aligned}$$

Hier werden 4 zusätzliche Funktionswerte außerhalb des Gitters $x_0, \ldots, x_n$ verwendet. In der Formel ist für $f \in C^4$ die dividierte Differenz beschränkt bei $h \to 0$, die Abweichung dieses lokalen Splines von der Funktion ist in den Stützstellen also nur $O(h^4)$. Tatsächlich reproduziert (2.3.21) kubische Polynome und mit Satz 2.3.7 folgt auch die globale Konvergenzaussage

$$\|f - Qf\|_{[\alpha,\beta]} = O(h^4)$$

die optimal ist bei kubischen Splines. Mit einem aufwändigeren Beweis kann man auch für den Interpolationsspline aus Definition 2.3.2 Konvergenz $\|s - f\|_\infty = O(h^4)$ zeigen.

3 Rechnerarithmetik und Kondition

In Computern stehen nur endlich viele Zahldarstellungen zur Approximation reeller Zahlen zur Verfügung, die Maschinenzahlen. Bei numerischer Rechnungen und fast jeder Daten-Eingabe treten daher Fehler auf, die das Ergebnis verfälschen. Zur quantitativen Beschreibung der damit verbundenen Risiken dient der Begriff der *Kondition* eines Problems.

3.1 Zahldarstellung und Rundungsfehler

Die gängige (Dezimal-) Darstellung reeller Zahlen gibt die Koeffizienten in der Entwicklung

$$e = 2.71828.. \iff e = 2 \cdot 10^0 + 7 \cdot 10^{-1} + 1 \cdot 10^{-2} + 8 \cdot 10^{-3} + \dots$$

als Reihe zur *Basis* 10 an. 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

$$x = \pm(x_m B^m + x_{m-1} B^{m-1} + \ldots + x_1 B + x_0 + x_{-1} B^{-1} + \ldots), \tag{3.1.1}$$

mit $x_j \in \{0, 1, \ldots, B-1\}$, wenn $1 < B \in \mathbb{N}$. Diese Darstellung ist nur dann eindeutig, wenn von zwei möglichen Darstellungen die endliche gewählt wird (etwa: $1.999\ldots = 2$). Bei *Gleitpunktzahlen* ("floating point representation") kann die Zahldarstellung mit einer endlichen Zahlmenge M viele Größenordnungen abdecken. Dabei bekommt man eine Abbildung $\mathbb{R} \to M$ mit kleinem *relativem* Fehler dadurch, dass man den höchsten Exponent m und die ersten ℓ Ziffern $x_m, \ldots, x_{m-\ell+1}$ angibt. Der zulässige Exponentenbereich wird natürlich auch eingeschränkt. Außerdem standardisiert man so, dass die erste (Nachkomma-) Stelle nicht null ist:

Definition 3.1.1 *Die Menge M der* normalisierten Gleitpunktzahlen *enthält das Element $x = 0$ und die Zahlen*

$$\begin{aligned} x \;&=\; \pm 0.\xi_1\xi_2\ldots\xi_\ell \cdot B^d := \pm(\xi_1 B^{-1} + \xi_2 B^{-2} + \ldots + \xi_\ell B^{-\ell}) B^d, \\ &\quad \textit{mit } \xi_j \in \{0, 1, \ldots, B-1\},\ \xi_1 \neq 0,\ d \in \mathbf{Z},\ |d| < e. \end{aligned} \tag{3.1.2}$$

Die Zahl $\xi_1\xi_2\ldots\xi_\ell$ heißt *Mantisse*, für die Analyse wird ein unbeschränkter Exponentenbereich ($e = \infty$) angenommen.

Beispiel 3.1.2 Binärdarstellung, $B = 2$. Da wegen der Normalisierung $\xi_1 = 1$ gilt, wird dieser Wert nicht gespeichert. Der IEEE-Standard speichert eine *real*-Zahl (double precision) in 64 Bit mit 11 Bit Exponent und 52(+1) Bit Mantisse. Der Exponent wird verschoben, mit der Binärdarstellung $d + e = d_{10}..d_1 d_0$ ist die Darstellung

$$\pm 0.\xi_1\xi_2\ldots\xi_\ell \cdot B^d \quad \mapsto \quad \boxed{\pm d_{10}..d_4}\boxed{d_3..d_0\,\xi_2..\xi_5}\boxed{\xi_6...\xi_{13}}\cdots\boxed{\ldots\xi_\ell}$$

(Die Speicherung dieser 8 Byte in PCs erfolgt aber in umgekehrter Reihenfolge) Hier ist also $\ell = 53, e = 1023$. Die Zahlgenauigkeit beträgt ca. 15 Dezimalstellen, die Größe der Zahlen ist durch $2^d \cong 10^{308}$ beschränkt.

Zentraler Bestandteil jeder Computerarithmetik ist die Zuordnung von reellen Zahlen zu Gleitpunktzahlen, also die Abbildung

$$rd: \left\{ \begin{array}{ccc} \mathbb{R} & \mapsto & M \\ x & \mapsto & rd(x) \end{array} \right. , \text{ die } \textit{Rundung.}$$

Günstiger als die triviale Rundung durch Abschneiden der Darstellung $\xi_1 \dots \xi_\ell = x_m \dots x_{m-\ell+1}$ in (3.1.1) ist die gewöhnliche Rundung, die jeweils auf die nächstgelegene Maschinenzahl abbildet. Bei Dezimalzahlen etwa wird ab 5 aufwärts gerundet, allgemein wird mit $x_1 \neq 0$ definiert.

$$rd(\pm 0.x_1x_2 \dots x_\ell x_{\ell+1} \dots \cdot B^d) := \left\{ \begin{array}{lll} \pm 0.x_1x_2 \dots x_\ell \cdot B^d & \text{falls} & x_{\ell+1} < B/2 \\ \pm(0.x_1x_2 \dots x_\ell + B^{-\ell}) \cdot B^d & \text{falls} & x_{\ell+1} \geq B/2 \end{array} \right. . \quad (3.1.3)$$

Bei dieser Rundung gilt für den *relativen Fehler* folgende Aussage.

Satz 3.1.3 *Für $x \neq 0$ gilt mit (3.1.3)*

$$\left| \frac{rd(x) - x}{x} \right| \leq \frac{1}{2} B^{1-\ell} =: \mathcal{E} \quad \Longleftrightarrow \quad rd(x) = x(1+\epsilon),\ |\epsilon| \leq \mathcal{E}.$$

Beweis In (3.1.3) ist $|x| \geq B^{d-1}$. In beiden Fällen gilt dort

$$|rd(x) - x| \leq \frac{B}{2} B^{-\ell-1} B^d \leq \frac{1}{2} B^{1-\ell} B^{d-1} \leq |x| \cdot \mathcal{E}. \qquad \blacksquare$$

Die Größe $\mathcal{E}$ heißt *Maschinengenauigkeit*, bei $B = 2$ ist $\mathcal{E} = 2^{-\ell}$ und im IEEE-Standard gilt $\mathcal{E} = 2^{-53} \cong 10^{-16}$. Die Verknüpfung von zwei Maschinenzahlen ergibt i.a. nicht wieder eine solche. Dabei werden neue Rundungsfehler erzeugt. Daher muss man bei der Analyse Maschinenoperationen von reellen Verknüpfungen unterscheiden:

Definition 3.1.4 *Für $a, b \in M$ und $* \in \{+, -, \cdot, /\}$ bezeichne (mit $b \neq 0$ im Fall der Division)*

$$a \mathbin{\widetilde{*}} b = gl(a * b)$$

das Ergebnis der entsprechenden (geräteabhängigen) Gleitpunktoperation.

Der IEEE-Standard schreibt vor, dass der Fehler bei diesen Operationen den Rundungsfehler des korrekten Ergebnisses 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. Daher gilt für die Gleitpunktoperationen $* \in \{+, -, \cdot, /\}$ mit $a, b \in M$ ($b \neq 0$ bei Division) die Beziehung

$$a \mathbin{\widetilde{*}} b = (a * b)(1 + \delta), \quad |\delta| \leq \mathcal{E} = \frac{1}{2} B^{1-\ell}. \quad (3.1.4)$$

Die wichtigste Folge der Rundung ist der Verlust der *Assoziativität* bei Gleitpunktoperationen. Aufgabe der Fehleranalyse ist daher auch die Identifikation *günstiger* Anordnungen bei mehrfachen Verknüpfungen. Zunächst werden einige elementare Spezialfälle diskutiert.

Beispiel 3.1.5 $B = 10,\ \ell = 3,\ a = 0.721,\ b = 0.457_{10}2,\ c = -0.456_{10}2$. Die folgenden Varianten liefern offensichtlich verschiedene Ergebnisse:

$$\begin{aligned}(a\widetilde{+}b)\widetilde{+}c = &\quad (0.464_{10}2)\widetilde{+}(-0.456_{10}2) &= 0.800_{10}0,\\ a\widetilde{+}(b\widetilde{+}c) = &\quad 0.721\widetilde{+}(0.1) &= 0.821_{10}0.\end{aligned}$$

Der negative Effekt in der ersten Variante wird *Auslöschung* genannt: Besitzen zwei Zahlen $a, b \in M$ gleiche Exponenten und gleiche führende Ziffern, ist bei Subtraktion das Ergebnis *exakt*! Sind in den Operanden aber *alte* Fehler vorhanden, werden diese wegen der Betrachtung relativer Fehler sehr verstärkt, da dann $|a - b| \ll |a| + |b|$ ist in (3.1.4). Dies muss bei der Fehlerfortpflanzung besonders beachtet werden.

Die erste Anwendung betrachtet die Bildung mehrfacher Produkte und Summen, mit $* \in \{+, \cdot\}$:

$$w_n := a_1 * a_2 * \ldots * a_n, \quad n \in \mathbb{N}, \tag{3.1.5}$$

Das numerische Ergebnis $\widetilde{w_n}$ werde dabei sequentiell berechnet mit gerundeten Werten $\widetilde{a_j}$,

$$\widetilde{w_n} := gl(\ldots gl(gl(\widetilde{a_1} * \widetilde{a_2}) * \widetilde{a_3}) \ldots) = gl(\widetilde{w_{n-1}} * \widetilde{a_n}), \tag{3.1.6}$$

wobei $\widetilde{a_i} = a_i(1 + \delta_i),\ |\delta_i| \le \mathcal{E}$ ist. Für die weiteren relativen Fehler gelte ebenfalls $|\epsilon_i| \le \mathcal{E}$.

Produkt: Hier gilt induktiv

$$\begin{aligned}\widetilde{w_2} &= gl(\widetilde{a_1} \cdot \widetilde{a_2}) = gl(a_1(1+\delta_1)a_2(1+\delta_2)) = a_1a_2(1+\delta_1)(1+\delta_2)(1+\epsilon_2)\\ &\ldots\\ \widetilde{w_n} &= gl(\widetilde{w_{n-1}} \cdot \widetilde{a_n}) = \widetilde{w_{n-1}} \cdot a_n(1+\delta_n)(1+\epsilon_n)\\ &= a_1a_2\cdots a_n(1+\delta_1)\cdots(1+\delta_n)(1+\epsilon_2)\cdots(1+\epsilon_n) = w_n(1+\gamma_n)\end{aligned}$$

mit

$$(1-\mathcal{E})^{2n-1} \le 1 + \gamma_n \le (1+\mathcal{E})^{2n-1}. \tag{3.1.7}$$

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

$$|(1+x)^m - 1| \le m|x|(1+|x|)^{m-1} \le m|x|e^{(m-1)|x|} \le 1.06m|x| \ \text{ für } (m-1)|x| \le 0.05\ . \tag{3.1.8}$$

Daher gilt für die Produktbildung in (3.1.7) für γ_n die Aussage

$$|\gamma_n| \le 1.06(2n-1)\mathcal{E}, \quad \text{falls } 2(n-1)\mathcal{E} \le 0.05\ . \tag{3.1.9}$$

Summe: Bei der Addition ist die Situation komplizierter, hier ist

$$\begin{aligned}\widetilde{w_2} &= gl(\widetilde{a_1} + \widetilde{a_2}) = (a_1(1+\delta_1) + a_2(1+\delta_2))(1+\epsilon_2)\\ &= a_1(1+\delta_1)(1+\epsilon_2) + a_2(1+\delta_2)(1+\epsilon_2) = a_1(1+\sigma_{21}) + a_2(1+\sigma_{22})\end{aligned}$$

$$
\begin{aligned}
&\dots\\
\widetilde{w_n} &= gl(\widetilde{w_{n-1}} + \widetilde{a_n}) = (\widetilde{w_{n-1}} + a_n(1+\delta_n))(1+\epsilon_n)\\
&= \Big(a_1(1+\sigma_{n-1,1}) + \ldots + a_{n-1}(1+\sigma_{n-1,n-1})\Big)(1+\epsilon_n) + a_n(1+\delta_n)(1+\epsilon_n)\\
&= a_1(1+\sigma_{n1}) + a_2(1+\sigma_{n2}) + \ldots + a_n(1+\sigma_{nn}),
\end{aligned}
$$

woraus sich induktiv $(1+\sigma_{ni}) = (1+\delta_i)(1+\epsilon_i)\cdots(1+\epsilon_n)$ ergibt. Aus (3.1.8) folgt daher

$$|\sigma_{ni}| \le 1.06(n-i+2)\mathcal{E}, \quad \text{wenn } (n-i+1)\mathcal{E} \le 0.05\ . \tag{3.1.10}$$

Bemerkung: Bei Produktbildung ist die Reihenfolge der Operanden offensichtlich unwichtig. Bei Summation liefern aber die ersten Summanden den größten relativen Fehlerbeitrag. Wenn möglich, sollte man daher zuerst die *betragskleinsten* Koeffizienten a_i summieren, die Schranke

$$|\widetilde{w_n} - w_n| \le 1.06\mathcal{E}\ \sum_{i=1}^{n}(n-i+2)|a_i|$$

wird dann minimiert. Bei Summation (monoton) konvergenter Reihen heißt das, dass man mit den letzten, d.h. kleinsten, Koeffizienten beginnt (*Rückwärts-Summation*).

Beispiel 3.1.6 Die Schranke für den Summationsfehler liefert ein weiteres Argument für Quadraturverfahren höherer Ordnung, vgl. §2.2. Dazu sei $Q_n(f) = \sum_{i=0}^{n}\alpha_i f(x_i)$ (z.B. $T_n(f)$ oder $S_n(f)$) eine Integralnäherung der Ordnung m für $I(f) = \int_a^b f(x)dx$ mit dem Fehler $|Q_n(f) - I_n(f)| \le c_f h^m$ mit $h = (b-a)/n$. Selbst mit exakten Funktionswerten berechnen Computer aber nur einen verfälschten Wert $\tilde{Q}_n(f)$, der Gesamtfehler hat daher die Form

$$
\begin{aligned}
|I_n(f) - \tilde{Q}_n(f)| &\le |I_n(f) - Q_n(f)| + |Q_n(f) - \tilde{Q}_n(f)|\\
&\le c_f h^m + 1.06n\mathcal{E}(\sum_{i=0}^{n}|\alpha_i|)\|f\|_\infty = c_f h^m + c_Q\mathcal{E}h^{-1}.
\end{aligned}
$$

Dieser Gesamtfehler wird daher nicht beliebig klein, sondern steigt wieder an für $h \to 0$ bzw. $n \to \infty$. Das Minimum in der Schranke wird erreicht in $h_{\min}^{m+1} = \mathcal{E}c_Q/(mc_f)$ mit einem Minimalwert

$$C\mathcal{E}^{\frac{m}{m+1}} \le c_f h^m + c_Q h^{-1}\mathcal{E} \quad \forall h > 0.$$

Bei einer Maschinengenauigkeit von $\mathcal{E} \approx 10^{-15}$ ist also bei der Trapezregel ($m=2$) kein Fehler unter $\approx 10^{-15*2/3} = 10^{-10}$ und bei der Simpsonregel ($m=4$) nicht unter $\approx 10^{-15*4/5} = 10^{-12}$ zu erwarten, die höhere Ordnung ermöglicht dabei also einen kleineren Gesamtfehler.

3.2 Konditionszahlen für Algorithmen

In der Abschätzung (3.1.10) erkennt man, wie sich einzelne Summanden (Eingabegrößen) auf den Fehler auswirken. Allerdings betrifft diese Schranke den absoluten Fehler im Ergebnis, für die Gleitpunktdarstellung ist dagegen der relative Fehler entscheidend. Für allgemeine Rechenvorschriften werden im folgenden Kenngrößen (Konditionszahlen) hergeleitet, welche die relative

Empfindlichkeit eines Ergebnisses gegenüber Störungen der Eingangsgrößen beschreiben. Wegen der geringen Größe der Maschinengenauigkeit $\mathcal{E}$ vernachlässigt man jetzt Produkte von Maschinenfehlern $\mathcal{O}(\mathcal{E}^2)$ und schreibt $\mathcal{E} + O(\mathcal{E}^2) \doteq \mathcal{E}$, ähnlich zu (3.1.8), wo diese Anteile durch die Konstante 1.06 aufgefangen wurden: $\mathcal{E} + O(\mathcal{E}^2) \leq 1.06\mathcal{E}$.

Einen numerischen Algorithmus kann man als eine Folge von Operationen bzw. Abbildungen $F = \phi^{[N]} \circ \cdots \circ \phi^{[2]} \circ \phi^{[1]}$ modellieren, die auf einem Vektor von Eingabedaten $x = (x_1, \ldots, x_n)^\mathsf{T} \in \mathbb{R}^n$ operieren. Betrachtet man nur einen Ausgabewert, gilt also $F : \mathbb{R}^n \to \mathbb{R}$. Als ersten Schritt sollte man sich die Empfindlichkeit der Gesamtabbildung F bei Auswertung an einem (etwa durch Rundungsfehler) leicht veränderten Punkt $\tilde{x} \in \mathbb{R}^n$ ansehen. Wegen der Aussage von Satz 3.1.3 wird dabei von einem kleinen relativen Fehler ausgegangen in der Form

$$\tilde{x}_i = (1 + \epsilon_i)x_i,\ |\epsilon_i| \ll 1,\ i = 1, \ldots, n. \tag{3.2.1}$$

Für $x_i \neq 0$ ist diese Darstellung äquivalent mit $\epsilon_i = (\tilde{x}_i - x_i)/x_i$. Auch im Ergebnis betrachtet man relative Fehler $\epsilon_F = (F(\tilde{x}) - F(x))/F(x)$. Vernachlässigt man Produkte von Fehlertermen, läßt sich die Empfindlichkeit des Ergebnisses F gegenüber Störungen einzelner Eingabedaten x_i durch folgende Zahlen charakterisieren, $e^{(i)} \in \mathbb{R}^n$ sind dabei die Einheitsvektoren.

Definition 3.2.1 *Für die Funktion $F : \mathbb{R}^n \to \mathbb{R}$ werden die* Konditionszahlen *$\kappa_i \in \mathbb{R}_+$ an der Stelle $x \in \mathbb{R}^n$ mit $F(x) \neq 0$ definiert durch*

$$\kappa_i := \limsup_{h \to 0} \Big| \frac{F(x + hx_i e^{(i)}) - F(x)}{hF(x)} \Big|,\ i = 1, \ldots, n. \tag{3.2.2}$$

Beispiel 3.2.2 Für die arithmetischen Grundoperationen gelten folgende Zusammenhänge:

$$\begin{array}{lll}
F(x_1, x_2) := x_1 x_2, & \epsilon_F \doteq \epsilon_1 + \epsilon_2, & \kappa_1 = \kappa_2 = 1, \\
F(x_1, x_2) := x_1 / x_2, & \epsilon_F \doteq \epsilon_1 - \epsilon_2, & \kappa_1 = \kappa_2 = 1, \\
F(x_1, x_2) := x_1 + x_2, & \epsilon_F \doteq \frac{x_1}{x_1+x_2}\,\epsilon_1 + \frac{x_2}{x_1+x_2}\,\epsilon_2, & \kappa_1 = \frac{|x_1|}{|x_1+x_2|},\ \kappa_2 = \frac{|x_2|}{|x_1+x_2|}.
\end{array}$$

Nach (3.2.2) liest man die Konditionszahlen an den Vorfaktoren der ϵ_i ab.

Für andere, glatte Funktionen F kann man die Konditionszahlen über Ableitungen bestimmen. Die zu einem Algorithmus gehörige Abbildung F ist in einem Bereich $D \in \mathbb{R}^n$ normalerweise aber nur dann glatt, wenn die auftretenden Fallunterscheidungen für alle $x \in D$ gleich ausfallen.

Satz 3.2.3 *Die Funktion $F : \mathbb{R}^n \to \mathbb{R}$ sei stetig differenzierbar in einer Umgebung der Stelle $x \in \mathbb{R}^n$ mit $F(x) \neq 0$. Dann haben die in (3.2.2) definierten Konditionszahlen die Werte*

$$\kappa_i = \Big| \frac{x_i}{F(x)} \frac{\partial F}{\partial x_i}(x) \Big|,\ i = 1, \ldots, n.$$

Für den relativen Fehler $\epsilon_F = (F(\tilde{x}) - F(x))/F(x)$ bei Auswertung in einem gestörten Argument $\tilde{x}$, (3.2.1) mit $|\epsilon_i| \leq h$, $i = 1, \ldots, n$, gilt dann

$$|\epsilon_F| \leq \sum_{i=1}^{n} \kappa_i |\epsilon_i| + o(h),\ h \to 0. \tag{3.2.3}$$

Anmerkung: Die Ungleichung (3.2.3) ist scharf, mit Gleichheitbei bei geeigneter Vorzeichenverteilung der ϵ_i.

Beweis Wegen der stetigen Differenzierbarkeit von F gilt mit $\tilde{x}_i - x_i = x_i\epsilon_i$ die Darstellung

$$F(\tilde{x}) - F(x) = \sum_{i=1}^{n} \frac{\partial F}{\partial x_i}(x) x_i\, \epsilon_i + o(h).$$

Daraus folgt wegen $o(h)/h \to 0\,(h \to 0)$ durch die spezielle Wahl $\epsilon_j = h\delta_{ij}$, $j = 1, \ldots, n$, zunächst die Darstellung der Konditionszahl κ_i und anschließend aus der Dreieckungleichung die Formel (3.2.3). ∎

Das obige Beispiel zeigt, dass Gleitpunktdarstellung die Punktoperationen begünstigt. Bei Subtraktion ungefähr gleich großer Zahlen tritt aber der schon erwähnte Fall der *Auslöschung,* auf, da fur $|x_1 + x_2| \ll \max\{|x_1|, |x_2|\}$, die Konditionszahlen sehr groß werden.

Die in (3.2.2) definierten Konditionszahlen beschreiben die Störungsempfindlichkeit von F komponentenweise. Zur Vereinfachung kann man durch Verwendung von Normen eine einheitliche Konditionszahl definieren. Dies ist insbesondere bei linearen Abbildungen F angebracht und wird im nächsten Abschnitt über Lineare Gleichungssysteme eingeführt.

Bei der fehlerbehafteten Ausführung von Operationen in einem Algorithmus reicht die Betrachtung der Gesamtabbildung F natürlich nicht aus, auch die Einzelschritte müssen überprüft werden. Nicht nur die Assoziativität von Summe und Produkt geht durch Rundungsfehler verloren, auch andere Gesetze gelten im allgemeinen nicht mehr (exakt). Daher können verschiedene, analytisch äquivalente Ausdrücke, wie z.B. die der binomischen Formel $a^2 - b^2 = (a + b)(a - b)$, sehr unterschiedliche numerische Ergebnisse liefern. Als wichtiges Beispiel werde die Funktion

$$F(x) := \sqrt{1 + x^2} - x \tag{3.2.4}$$

betrachtet. Für $x \geq 0$ ist $0 \leq F(x) \leq 1$. Da $|F'(x)| = |\frac{x}{\sqrt{1+x^2}} - 1| \leq 1$ ist für $x \geq 0$, führt eine Störung im Argument x bei exakter Rechnung nur zu einer geringen absoluten Verfälschung im Funktionswert F. Auch für die Konditionszahl ergibt sich nach Satz 3.2.3 die harmlose Schranke

$$\kappa_x = \frac{|xF'(x)|}{|F(x)|} = \frac{|x|}{\sqrt{1 + x^2}} \leq 1.$$

Werden die Teilschritte in (3.2.4) aber nur mit endlicher Genauigkeit ausgeführt, sieht das Ergebnis schlechter aus. Selbst wenn nur bei der Addition in (3.2.4) ein relativer Fehler ϵ, $|\epsilon| \leq \mathcal{E}$, entsteht, 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 $|F| \leq 1$ noch größer. In der Formel (3.2.4) 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

$$F(x) = \sqrt{1 + x^2} - x = \frac{1 + x^2 - x^2}{\sqrt{1 + x^2} + x} = \frac{1}{x + \sqrt{1 + x^2}}, \quad x \geq 0. \tag{3.2.5}$$

Für $x < 0$ ist dagegen (3.2.4) die günstigere Form, eine Implementierung von F für $x \in \mathbb{R}$ sollte daher beide Formen mit Fallunterscheidung verwenden.

Beispiel 3.2.4 Formel (3.2.4) wird mit 4-ziffriger Dezimalrechnung in $x = 9.999$ ausgewertet, bei exakter Rechnung ist $F(9.999) = 0.04988\ldots$ Dagegen ist $x\tilde{\cdot}x = 99.98$, $1\tilde{+}99.98 = 101.0$, $\widetilde{\sqrt{101}} = 10.05$ und daher $\widetilde{F}(x) = 0.05100$. Dies entspricht einem relativen Fehler von $0.022 \cong 44 \cdot \mathcal{E}$. Die letzte, äquivalente Form in (3.2.5) ergibt dagegen den korrekten 4-stelligen Wert 0.04988. Für $x \geq 100$ schließlich liefert 4-stellige Rechnung in (3.2.4) immer den Wert $\widetilde{F}(x) = 0$.

Das Vorgehen beim Ausdruck (3.2.4) wird nun verallgemeinert. Bei einer zusammengesetzten Abbildung $F = \phi^{[N]} \circ \cdots \circ \phi^{[2]} \circ \phi^{[1]}$ muss man nun Fehler bei jeder einzelnen Auswertung berücksichtigen. Zunächst wird nur der Fall $N = 2$ mit

$$F = \psi \circ \phi \quad \text{ und } \ \phi : \mathbb{R}^n \to \mathbb{R}^m,\ \psi : \mathbb{R}^m \to \mathbb{R}$$

betrachtet, also die Vorschrift $z := \psi(y)$, $y := \phi(x)$ für $z = F(x)$ mit glatten Funktionen ϕ, ψ. Indizes bei den Funktionen bezeichnen die Komponenten, $\phi = (\phi_1, \ldots, \phi_m)^\mathsf{T}$. Das Ergebnis kann dann induktiv auf eine beliebige Zahl N von Teilschritten erweitert werden. Statt der exakten Berechnung werden bei Gleitpunktrechnung Werte

$$\tilde{z} = \tilde{\psi}(\tilde{y}), \quad \tilde{y} = \tilde{\phi}(\tilde{x})$$

geliefert. Für die Maschinen-Implementierung $\tilde{\phi}$ wird wie früher die Fehleraussage

$$\tilde{\phi}_i(x) = (1 + \epsilon_i^{(1)})\phi_i(x),\ i = 1, \ldots, m \iff \tilde{\phi}(x) = (I + E_1)\phi(x)$$

angenommen für alle x. bei der Abbildung $\phi : \mathbb{R}^n \to \mathbb{R}^m$ entspricht die Verfälschung also der Multiplikation mit einer Diagonalmatrix $I + E_1 = \mathrm{diag}(1 + \epsilon_1^{(1)}, \ldots, 1 + \epsilon_m^{(1)})$. Bei $\tilde{\psi}$ gelte $\tilde{\psi}(y) = (1 + \epsilon^{(2)})\psi(y)$. Alle Einzelfehler $\epsilon_i^{(j)}$ seien so klein, dass ihre Produkte vernachlässigbar sind (z.B. $|\epsilon_i^{(j)}| \leq c\mathcal{E}$). Man erhält so mit $\epsilon^{(2)}\psi(\tilde{y}) \doteq \epsilon^{(2)}\psi(y) = \epsilon^{(2)}z$ die Beziehungen

$$\begin{aligned}
\tilde{z} - z &= \tilde{\psi}(\tilde{y}) - \psi(y) = \tilde{\psi}(\tilde{y}) - \psi(\tilde{y}) + \psi(\tilde{y}) - \psi(y) \\
&\doteq \epsilon^{(2)}z + \psi'(y)(\tilde{y} - y), \\
\tilde{y} - y &= \tilde{\phi}(\tilde{x}) - \phi(\tilde{x}) + \phi(\tilde{x}) - \phi(x) \doteq E_1 y + \phi'(x)(\tilde{x} - x).
\end{aligned}$$

Die Ableitung in der letzten Gleichung ist die Funktionalmatrix $\phi'(x) \in \mathbb{R}^{m \times n}$. Auch für den Startfehler hat man mit (3.2.1) die Darstellung $\tilde{x} - x = E_0 x$ mit einer entsprechenden Diagonalmatrix E_0. Setzt man die Fehlerdarstellungen ineinander ein, tritt u.a. das Produkt $\psi'(y)\phi'(x)$ auf, das nach der Kettenregel wegen $y = \phi(x)$ gerade die Ableitung

$$F'(x) = \mathrm{grad}(\psi \circ \phi)(x) = \psi'(\phi(x))\phi'(x)$$

darstellt. Für den Gesamtfehler bekommt man so die Darstellung

$$\tilde{z} - z \doteq F'(x)E_0 x + \psi'(y)E_1 y + \epsilon^{(2)}z. \tag{3.2.6}$$

Diese läßt sich so interpretieren, dass der in einem Zwischenschritt des Algorithmus auftretende Fehler mit der Ableitung der *Restabbildung* multipliziert wird. Für den Startfehler E_0 ist dieses die Gesamtabbildung $F = \psi \circ \phi$, für den Fehler E_1 aber nur noch die letzte Abbildung ψ. Für den relativen Fehler bekommt man mit $z = \psi(y) = F(x)$ somit die Formel

$$\epsilon_z = \frac{\tilde{z} - z}{z} = \frac{F'(x)}{F(x)} E_0 x + \frac{\psi'(y)}{\psi(y)} E_1 y + \epsilon^{(2)}.$$

Diese Aussage kann als Verallgemeinerung von (3.2.3) geschrieben werden, wenn man die Konditionszahlen aus Satz 3.2.3 auf die Variablen y_i, x_j bezieht:

$$\kappa_{y_i} = \Big| \frac{y_i}{\psi(y)} \frac{\partial \psi}{\partial y_i}(y) \Big|, \quad \kappa_{x_j} = \Big| \frac{x_j}{F(x)} \frac{\partial F}{\partial x_j}(x) \Big|.$$

Aus (3.2.6) folgt dann insgesamt

$$|\epsilon_z| \dot{\leq} \sum_{j=1}^{n} \kappa_{x_j} |\epsilon_j^{(0)}| + \sum_{i=1}^{m} \kappa_{y_i} |\epsilon_i^{(1)}| + |\epsilon^{(2)}|. \tag{3.2.7}$$

Auch hier tritt bei einer geeigneten Vorzeichenverteilung Gleichheit auf. Merkregel:

Der im k-ten Schritt eines Algorithmus auftretende Rundungsfehler wirkt sich proportional zur Konditionszahl der *Restabbildung* auf den relativen Endfehler aus

In einem guten numerischen Algorithmus darf also keine einzige Restabbildung eine große Konditionszahl besitzen.

Beispiel 3.2.5 Die in (3.2.4) betrachtete Funktion $F(x)$ ist die positive Lösung der quadratischen Gleichung $F^2 + 2xF - 1 = 0$, in der bei ungünstiger Auswertung also erhebliche Fehler auftreten können. Die Funktion F läßt sich vereinfachend aus den beiden Abbildungen

$$\phi(x) = \begin{pmatrix} 1 + x^2 \\ x \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \ \psi(y) = \sqrt{y_1} - y_2$$

zusammensetzen. Die Konditionszahl κ_x für den Eingangsfehler wurde schon bestimmt, sie ist harmlos, $|\kappa_x| \leq 1$. Die Ableitung von ψ ist $\mathrm{grad}\psi(y) = (\frac{1}{2\sqrt{y_1}}, -1)$. Für die κ_{y_i} ergibt sich daher mit $y = \phi(x)$:

$$\kappa_{y_1} = \frac{|y_1|}{2\psi(y)\sqrt{y_1}} = \frac{\sqrt{1+x^2}}{2F(x)} = \frac{\sqrt{1+x^2}}{2}(\sqrt{1+x^2} + x), \quad \kappa_{y_2} = \frac{|y_2|}{\psi(y)} = |x|(\sqrt{1+x^2} + x).$$

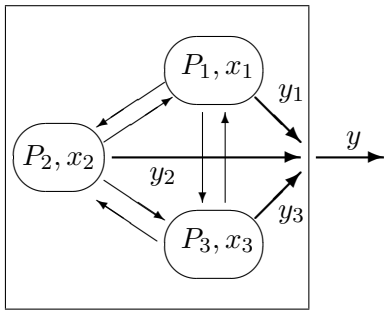
Im beiden Ausdrücken wurde der Kehrwert $1/F$ mit Hilfe von (3.2.5) ersetzt und zeigt das starke Anwachsen $\kappa_{y_i} \to \infty, \ x \to +\infty$. Für negative x gilt dagegen $F(x) > \sqrt{1+x^2} > x$, dort sind alle Konditionszahlen durch eins beschränkt und die ursprüngliche Formel (3.2.4) gut konditioniert.

Eine entsprechende, aufwändigere Analyse zeigt das umgekehrte Bild für die Darstellung (3.2.4): diese hat für $x > 0$ eine gute, für $x < 0$ dagegen eine schlechte Kondition. Bei einer Implementierung dieser speziellen Funktion F muss daher durch Fallunterscheidung die jeweils günstigere Form ausgewählt werden.

4 Lineare Gleichungssysteme

Viele Näherungsverfahren der Numerik bei Differentialgleichungen Integralgleichungen, etc., führen auf *Lineare Gleichungssysteme* (LGSe). In einigen Anwendungen können Modelle direkt als LGSe formuliert werden, die aber oft erst für große Modelldimensionen realistisch sind.

4.1 Beispiele



Beispiel 4.1.1 *Input-Output-Analyse*: Zur Analyse der Wechselwirkungen in einer Volkswirtschaft und der Auswirkungen geänderter Nachfrage nimmt man eine Unterteilung in Produktionsbereiche $P_j,\ j = 1, \ldots, n$, vor. Mit y_j wird die Endnachfrage der Verbraucher nach Produkten aus P_j bezeichnet, zusammengefaßt zu $y \in \mathbb{R}^n$. 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:

$$b_{ij} : \begin{cases} \text{Bedarf an Produkten aus } P_i \text{ zur Produktion einer} \\ \text{Einheit (Geldwert) in } P_j,\ b_{ij} \geq 0. \end{cases} \tag{4.1.1}$$

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

$$\begin{aligned} x = & \underbrace{y}_{\text{End-Nachfrage}} + \underbrace{Bx}_{\text{interne Nachfrage}} \\ \Leftrightarrow \quad & x - Bx = y \ \Leftrightarrow\ (I - B)x = y. \end{aligned} \tag{4.1.2}$$

Somit beschreibt die Matrix $(I - B)^{-1}$ die Auswirkungen einer sich ändernden Endnachfrage y. Die Volkswirtschaft arbeitet aber nur sinnvoll, wenn alle Elemente von $(I - B)^{-1}$ nichtnegativ sind (Kriterien?).

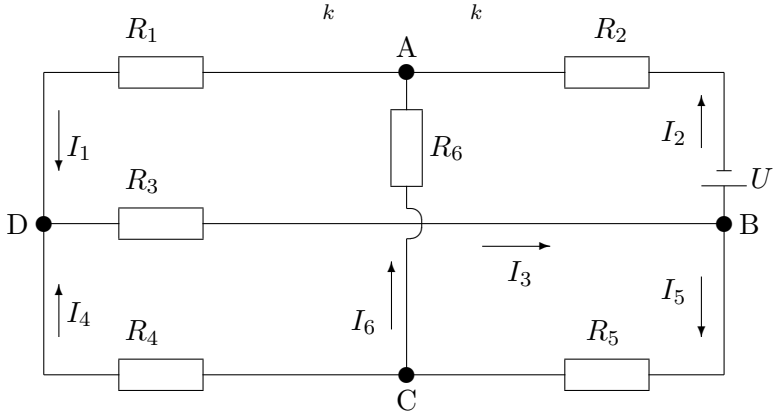
Bemerkung: Auch andere Probleme führen auf Bilanzgleichungen wie in Beispiel 1. In der Computergrafik, z.B., ergibt sich die absolute Helligkeit und Farbe eines Flächenelements aus der direkt und indirekt empfangenen Lichtmenge (Radiosity-Verfahren).

Beispiel 4.1.2 *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:

$$\text{Knoten}\quad \begin{array}{lrrrrrrcl} A: & -I_1 & +I_2 & & & & +I_6 & = & 0 \\ B: & & -I_2 & +I_3 & & -I_5 & & = & 0 \\ C: & & & & -I_4 & +I_5 & -I_6 & = & 0 \\ D: & I_1 & & -I_3 & +I_4 & & & = & 0 \end{array} \tag{4.1.3}$$

$$\text{Maschen}\quad \begin{array}{lrrrrrrcl} ACB: & & I_2R_2 & & & & -I_5R_5 & -I_6R_6 & = & U \\ ADB: & I_1R_1 & +I_2R_2 & +I_3R_3 & & & & & = & U \\ ADC: & I_1R_1 & & & -I_4R_4 & & & +I_6R_6 & = & 0 \\ BDC: & & & -I_3R_3 & -I_4R_4 & -I_5R_5 & & & = & 0 \end{array} \tag{4.1.4}$$

Die Gleichungen sind offensichtlich *linear abhängig*, überflüssige sind zu eliminieren. Die Summe der Gleichungen in (4.1.3) verschwindet, auch in (4.1.4) ist eine überflüssig. Damit bleiben sechs Gleichungen für sechs Unbekannte. In Matrixform schreibt sich (4.1.3,4.1.4) also kompakt als

$$\begin{array}{c} A \\ B \\ C \\ ACB \\ ADB \\ ADC \end{array} \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & R_2 & 0 & 0 & -R_5 & -R_6 \\ R_1 & R_2 & R_3 & 0 & 0 & 0 \\ R_1 & 0 & 0 & -R_4 & 0 & R_6 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ U \\ U \\ 0 \end{pmatrix} \tag{4.1.5}$$

Bemerkung: Ähnliche Strukturen tauchen auch bei anderen Problemen auf, etwa Transportproblemen in Graphen. Die Matrix des Teilsystems (4.1.3) ist die sog. *Inzidenzmatrix* des zum Netzwerk gehörigen Graphen.

Die Numerik befaßt sich mit der effizienten und numerisch stabilen Auflösung solcher Systeme, die bei heute üblichen Anwendungen insbesondere sehr große Dimensionen n aufweisen. Ein Ansatzpunkt ist dabei die Identifikation von Klassen von Matrizen, deren spezielle Struktur effizientere Verfahren ermöglicht (z.B., Definitheit, dünne Besetzung, s.u.).

4.2 Matrizen und Normen

Vektoren im $\mathbb{R}^n$ bzw. $\mathbb{C}^n$ (Sammelbegriff $\mathbb{K}^n$) werden als Spaltenvektoren betrachtet

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

Für quantitative Aussagen verwendet man Normen, etwa die Hölder-Normen

$$\|x\|_p := \Big(\sum_{j=1}^{n} |x_j|^p \Big)^{1/p}, \quad p \in \mathbb{R},\ p \geq 1. \tag{4.2.1}$$

Die wichtigsten Vertreter sind

$$\begin{array}{rcll} \|x\|_1 & = & \sum\limits_{j=1}^{n} |x_j| & \text{Summennorm} \\ \|x\|_2 & = & \sqrt{\sum\limits_{j=1}^{n} |x_j|^2} & \text{Euklidnorm} \\ \|x\|_\infty & = & \max\limits_{j=1}^{n} |x_j|, & \text{Maximumnorm} \end{array} \tag{4.2.2}$$

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

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

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

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

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

$$\|A\|_p := \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\|x\|_p = 1} \|Ax\|_p. \tag{4.2.3}$$

In (4.2.3) wird das Supremum als Maximum angenommen (Grund?). Die durch (4.2.2) *induzierten Normen* lassen sich explizit angeben:

$$\begin{array}{rcll} \|A\|_1 & = & \max\limits_{j=1}^{n} \sum\limits_{i=1}^{m} |a_{ij}| & \text{Spaltensummennorm,} \\ \|A\|_2 & = & \varrho(A^* A)^{1/2} & \text{Spektralnorm.} \\ \|A\|_\infty & = & \max\limits_{i=1}^{m} \sum\limits_{j=1}^{n} |a_{ij}| & \text{Zeilensummennorm,} \end{array} \tag{4.2.4}$$

Dabei ist der Spektralradius $\varrho(A^* A) = \max_j \lambda_j(A^* A)$ der maximale Eigenwert von $A^* A$, wobei $A^* := \bar{A}^{\mathsf{T}}$ ist. Für induzierte Matrixnormen überträgt sich die Dreieckungleichung von der

Vektornorm: $\|A+B\|_p \le \|A\|_p + \|B\|_p$. Induzierte Matrix-Normen sind *verträglich* mit der ursprünglichen Vektornorm durch die Produktformel

$$\|Ax\| \le \|A\|\,\|x\| \quad \forall x \in \mathbb{K}^n,\ A \in \mathbb{K}^{m\times n}. \tag{4.2.5}$$

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

$$\|A\|_F := \Big(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\Big)^{1/2} \qquad \text{Frobeniusnorm.} \tag{4.2.6}$$

Auch diese ist verträglich mit der Euklidnorm, denn aus der Cauchy-Schwarz-Ungleichung folgt

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

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

$$\|AB\| \le \|A\|\,\|B\|. \tag{4.2.7}$$

In dieser Vorlesung werden nur Normen benutzt, die (4.2.7) erfüllen. Wenn es also unterschiedliche Normen gibt, stellt sich die Frage nach einer besten (minimalen) Matrix-Norm. Für quadratische Matrizen gibt es leider nur eine größte untere Schranke (Infimum), die allerdings selbst keine Norm ist. Diese Größe ist für $A \in \mathbb{K}^{n\times n}$ der schon erwähnte *Spektralradius*

$$\varrho(A) := \max\{|\lambda_j(A)| :\ \lambda_j(A) \text{ EW zu } A\}. \tag{4.2.8}$$

Wegen der fehlenden Definitheit, etwa bei $\varrho\left(\begin{smallmatrix}0 & 1\\ 0 & 0\end{smallmatrix}\right) = 0$, ist $\varrho(A)$ aber sicherlich keine Norm. Der Zusammenhang mit Normen wird im nächsten Satz behandelt. Da die Aussagen von Eigenschaften der Matrix abhängt, werden einige der auch später verwendeten Begriffe und ihr Bezug zur Jordan-Normalform hier zusammengestellt:

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

Satz 4.2.1 *a) Für jede Matrixnorm $\|\cdot\|$ gilt* $\varrho(A) \le \|A\|$.

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

$$\varrho(A) \le \|A\|_M \le \varrho(A) + \varepsilon. \tag{4.2.9}$$

c) Für diagonalisierbare Matrizen A kann in (4.2.9) $\varepsilon = 0$ gewählt werden, für hermitesche, reell-symmetrische und normale Matrizen gilt insbesondere $\|A\|_2 = \varrho(A)$.

Beweis a) x sei Eigenvektor von A, $\Rightarrow$ $|\lambda|\|x\| = \|\lambda x\| = \|Ax\| \le \|A\|\|x\|$.

b) Die Jordan-Normalform von A sei

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

mit $\delta_i \in \{0,1\}, i = 1,\dots,n-1$. Eine weitere Ähnlichkeitstransformation mit der Diagonalmatrix $P = \mathrm{diag}(1,\varepsilon,\dots,\varepsilon^{n-1})$ führt Λ über in

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

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

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

Für die durch die Vektornorm $\|x\|_M := \|M^{-1}x\|_\infty$ induzierte Matrixnorm

$$\|B\|_M := \|M^{-1}BM\|_\infty, \quad M := XP,$$

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, denn unitäre Matrizen $X^{-1} = X^*$ sind *isometrisch*, $\|Xy\|_2 = \|y\|_2\,\forall y$. Für diese ist $\|X\Lambda X^{-1}\|_2 = \|\Lambda\|_2$. ∎

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

Satz 4.2.2 *(Neumannreihe) Gegeben sei die Matrix $B \in \mathbb{R}^{n\times n}$ und es gelte*

$$\varrho(B) < 1. \tag{4.2.10}$$

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

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

c) Dann gibt es eine Matrixnorm so, dass $\|B\| < 1$ ist. In dieser Norm gilt die Schranke

$$\|(I-B)^{-1}\| \le \frac{1}{1-\|B\|}. \tag{4.2.11}$$

Bemerkung: Dieser Satz ist die Basis vieler Störungsaussagen, (4.2.11) gilt natürlich in jeder Norm, für die $\|B\| < 1$ ist.

Beweis a) 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 (4.2.10).

b) Mit den Teilsummen $\sum_{k=0}^{m} B^k$ gilt die Identität

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

Nach Voraussetzung und Satz 4.2.1 existiert eine Norm mit $\|B\| < 1$. 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 b). Aus der Reihe b) ergibt sich mit Dreieck- und Produkt-Ungleichung die Schranke

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

4.3 Der Gauß-Algorithmus

Zur Lösung von linearen Gleichungssystemen

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \ i = 1, \ldots, n \quad \Leftrightarrow \quad Ax = b, \tag{4.3.1}$$

$A \in \mathbb{K}^{n \times n}$, gibt es verschiedene *direkte*, aber 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. In der Vorlesung werden zwei direkte Verfahren behandelt, die mit verschiedenen Umformungen arbeiten. Im Vordergrund steht dabei die Interpretation als *Faktorisierung* der Matrix A. Bei iterativen Verfahren konstruiert man dagegen eine Folge von Näherungen, die gegen die Lösung konvergiert. Dabei wird die Matrix nicht verändert, was in vielen Fällen ("dünnbesetzte" Matrizen) vorteilhaft sein kann.

Der Gauß-Algorithmus ist das Standardverfahren zur direkten Auflösung und nutzt aus, dass sich die Lösung x von (4.3.1) nicht ändert, wenn Gleichungen linear kombiniert werden. Ziel ist die Konstruktion eines Systems in Dreieckform,

$$\left.\begin{array}{rrrrcl} r_{11}x_1 & +r_{12}x_2 & +\ldots & +r_{1n}x_n & = & b'_1 \\ & r_{22}x_2 & +\ldots & +r_{2n}x_n & = & b'_2 \\ & & \ddots & \vdots & & \vdots \\ & & & r_{nn}x_n & = & b'_n \end{array}\right\} \quad \Leftrightarrow \quad Rx = b', \tag{4.3.2}$$

das man beginnend mit x_n direkt auflösen kann, falls alle $r_{ii} \neq 0 \, \forall i$ sind. Die Matrix A sei im folgenden regulär. Man erzeugt ein Dreiecksystem (4.3.2) (im ursprünglichen Speicherplatz der Matrix A) in $n - 1$ Durchgängen. Im ersten wird

1. für $a_{11} = 0$ eine Zeile mit $a_{i1} \neq 0$ gesucht und diese dann mit der ersten vertauscht (*Pivot-Schritt*, i-te Zeile = Pivot-Zeile);

2. jeweils dasjenige Vielfache der ersten Zeile zu den restlichen Zeilen $i = 2, \ldots, n$ addiert, das dort den Koeffizienten bei x_1 zu Null macht (*Eliminationsschritt*). Bei Behandlung der i-ten Zeile ist dabei $-a_{i1}/a_{11}$ der Vorfaktor der 1. Zeile.

Danach stehen in der ersten Spalte der Matrix Nullen unter $a_{11} \neq 0$ und das Verfahren kann mit den nächsten Spalten wiederholt werden. Nach $k-1$ Durchgängen hat das LGS also die Form

$$A^{(k)}x = b^{(k)} \quad \text{mit} \quad A^{(k)} = \left(\begin{array}{ccc|ccc} r_{11} & \cdots & r_{1,k-1} & r_{1k} & \cdots & r_{1n} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & r_{k-1,k-1} & r_{k-1,k} & \cdots & r_{k-1,n} \\ \hline & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & 0\,(\ell_{ij}) & & \vdots & & \vdots \\ & & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{array}\right), \tag{4.3.3}$$

die Schritte 1 und 2 sind dann nur auf die Restmatrix unten mit dem Pivotelement $a_{kk}^{(k)}$ anzuwenden, der Teil darüber bleibt unverändert. Die verwendeten Eliminationskoeffizienten

$$\ell_{ik} := \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

haben eine besondere Bedeutung, die später erläutert wird. Man speichert sie bei Durchführung des Algorithmus an die freiwerdenden Plätze der $a_{ik}^{(k+1)} = 0$ $(i > k)$. Nach $n-1$ Durchgängen mit Matrizen $A^{(1)} = A, A^{(2)}, \ldots, A^{(n-1)}$ ergibt sich ein Dreiecksystem (4.3.2) mit $R = A^{(n)}$, $b' = b^{(n)}$, das sich einfach auflösen läßt. Bei der folgenden Formulierung werden die oberen Versionsnummern weggelassen, das Zeichen ":=" bedeutet eine Zuweisung.

Algorithmus 4.3.1 Gauß-Algorithmus (kij-Variante) ohne Pivotisierung, durchführbar im Speicherplatz der Originalmatrix A (d.h. $l_{ij} \mapsto a_{ij}$, $j < i$),

für $k := 1, \ldots, n-1$: {	Elimination
für $i := k+1, \ldots, n$: {	
$\ell_{ik} := a_{ik}/a_{kk}$; $\underline{b_i := b_i - \ell_{ik} b_k;}$	
für $j := k+1, \ldots, n$: {	
$a_{ij} := a_{ij} - \ell_{ik} a_{kj}$; }}}	
$x_n := b_n / a_{nn}$;	
für $i := n-1, n-2, \ldots, 1$: {	Auflösung
$x_i := \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j\right) / a_{ii}$; }	

(4.3.4)

Beim Rechenaufwand werden alle arithmetischen Operationen gezählt (kurz FLOP), aber wieder nur der am schnellsten wachsenden Anteil berücksichtigt. Der Algorithmus enthält drei gestaffelte Schleifen (k, i, j), der Hauptaufwand entsteht daher in der 5. Zeile (innerste Schleife):

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

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

Eliminationsteil des Algorithmus müssen dann nur die die rechte Seite b betreffenden Schritte (unterstrichene Anweisung in der 3. Zeile) und die Auflösung wiederholt werden. Dazu wurden die Größen ℓ_{ik} gespeichert. Der Aufwand ist dann

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

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 reine Umformung beim System $A^{(k)}x = b^{(k)}$ in einem Eliminationsschritt ohne Pivotisierung,

```
für i = k+1, ..., n: {
    b_i^(k+1) := b_i^(k) − ℓ_ik b_k^(k);
    für j := k+1, ..., n: {
        a_ij^(k+1) := a_ij^(k) − ℓ_ik a_kj^(k);}}
```

entspricht der Multiplikation dieses Systems mit einer speziellen Matrix L_k^{-1},

$$A^{(k)}x = b^{(k)} \quad \rightarrow \quad A^{(k+1)}x = L_k^{-1}A^{(k)}x = L_k^{-1}b^{(k)} = b^{(k+1)},$$

die sich nur durch eine Rang-1-Matrix von der Einheitsmatrix unterscheidet,

$$L_k^{-1} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -\ell_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -\ell_{nk} & & 1 \end{pmatrix} = I - \ell^{(k)} e^{(k)\mathsf{T}}, \quad \ell^{(k)} := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \ell_{k+1,k} \\ \vdots \\ \ell_{nk} \end{pmatrix}. \tag{4.3.5}$$

Dabei ist $e^{(k)}$ der k-te Einheitsvektor und wegen $e^{(k)\mathsf{T}}\ell^{(k)} = 0$ gilt $(I - \ell^{(k)}e^{(k)\mathsf{T}})(I + \ell^{(k)}e^{(k)\mathsf{T}}) = I$, also $L_k = I + \ell^{(k)}e^{(k)\mathsf{T}}$. Somit wird die obere Dreieckmatrix $R = A^{(n)}$ erzeugt durch Multiplikation mit L-Matrizen:

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

$$A = L_1 L_2 \cdots L_{n-1}\, R =: LR. \tag{4.3.6}$$

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

$$A = LR = \begin{pmatrix} 1 & 0 & \dots & 0 \\ * & 1 & & \\ \vdots & & \ddots & \\ * & * & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} * & * & \dots & * \\ 0 & * & & * \\ & & \ddots & \vdots \\ 0 & & & * \end{pmatrix} \tag{4.3.7}$$

Die Faktoren L (ohne Hauptdiagonale) und R können bei (4.3.4) wieder in A gespeichert werden.

Satz 4.3.2 *Der Gaußalgorithmus (4.3.4) (ohne Pivotisierung) erzeugt eine* LR-Zerlegung *(4.3.7) der Matrix A. Der Rechenaufwand dafür ist* $\frac{2}{3}n^3$ *Operationen. Anschließend reduziert sich jede Lösung eines LGS* $Ax = b$ *auf*

$$b = Ax = L\underbrace{(Rx)}_{b^{(n)}} \quad \Leftrightarrow \quad Lb^{(n)} = b,\ Rx = b^{(n)}, \tag{4.3.8}$$

also auf die Auflösung von zwei LGSen mit Dreieckmatrizen mit insgesamt $2n^2$ *Operationen.*

Bemerkung: Der Gauß-Algorithmus (4.3.4) besteht aus 3 geschachtelten Schleifen mit der Basisoperation $a_{ij} := a_{ij} - a_{ik}a_{kj}$ und den Laufindizes i, j, k. Verblüffenderweise kann für jede Permutation dieser Schleifen eine korrekte Variante des Verfahrens angegeben werden. Praktische Bedeutung hat diese Erkenntnis, da die Effizienz dieser Varianten stark sprach- und maschinenabhängig sein kann (Matrizen zeilen-/spaltenweise gespeichert? Parallel-Verarbeitung?).

Zum Beispiel läßt sich die LR-Zerlegung direkt erzeugen ohne die Zwischenmatrizen $A^{(k)}$. Die Identität $A = L \cdot R$ wird dabei als Gleichungssystem für L und R aufgefaßt:

$$a_{ij} = \sum_{k=1}^{\min\{i,j\}} \ell_{ik}r_{kj} \quad \Longleftrightarrow \quad \begin{cases} a_{ij} = \sum\limits_{k=1}^{j-1} \ell_{ik}r_{kj} + \ell_{ij}r_{jj}, & i > j, \\ a_{ij} = \sum\limits_{k=1}^{i-1} \ell_{ik}r_{kj} + r_{ij}, & i \le j. \end{cases} \tag{4.3.9}$$

Diese Gleichungen kann man direkt nach den ℓ_{ij} (obere Gleichungen) bzw. r_{ij} (untere Gleichungen) auflösen. Auch hier sind noch unterschiedliche Reihenfolgen möglich und führen auf die ijk- und jik-Variante des Gauß-Algorithmus. Ein Vorteil dieser Varianten ist eine höhere erreichbare Genauigkeit, wenn die Summen in (4.3.9) mit erhöhter Genauigkeit berechnet werden.

Pivotisierung

Im allgemeinen muß man beim Gauß-Algorithmus mit Zeilenvertauschungen (*Pivotisierung*) zu kleine Hauptdiagonalelemente verhindern. Mögliche Pivotstrategien sind unter Effizienzgesichtspunkten zu analysieren. Andererseits kann man den Zusatzaufwand für Vertauschungen einsparen bei Klassen von Matrizen, wo Pivotisierung unnötig ist. Interessant ist hier, dass bestimmte Strukturen bzw. Eigenschaften der Matrix A beim Gauß-Algorithmus unverändert bleiben.

Definition 4.3.3 *Es sei A eine $n \times n$-Matrix.*

a) A heißt streng diagonaldominant*, wenn gilt*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1, \ldots, n. \tag{4.3.10}$$

b) A heißt $(2m+1)$-Bandmatrix*, wenn $a_{ij} = 0$ für $|i-j| > m$. Spezialfall $m = 1$:* Tridiagonalmatrix*.*

Bei definiten und streng diagonaldominanten Matrizen sind die Hauptdiagonalelemente offensichtlich von Null verschieden. Band-Struktur bleibt erhalten und erlaubt eine Reduktion des Aufwands beim Gauß-Algorithmus (ohne Pivotisierung).

Satz 4.3.4 *a) Die Pivotelemente $a_{ii}^{(i)}$, $i = 1, \ldots, k-1$, im Gauß-Algorithmus (4.3.4) seien von null verschieden. Wenn die Ausgangsmatrix A eine der Eigenschaften hermitesch, positiv (negativ) definit, streng diagonaldominant oder $2m+1$-Bandmatrix hatte, dann gilt dies auch für die Restmatrix $\left(a_{ij}^{(k)}\right)_{i,j \geq k}$ in (4.3.3).*

b) Bei definiten oder streng diagonaldominanten Matrizen benötigt der Gauß-Algorithmus keine Pivotisierung.

Beweis a) Für die Strukturerhaltung ist nur der erste Eliminationsschritt mit $a_{11} \neq 0$ zu untersuchen. Er erzeugt folgende Umformung. Aus

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

entsteht eine entsprechende Teilmatrix $B^{(2)}$ von $A^{(2)}$ durch eine Rang-1-Änderung:

$$B^{(2)} := \left(a_{ik} - \frac{1}{a_{11}} a_{i1} a_{1k}\right)_{i,k=2}^{n} = B^{(1)} - \frac{1}{a_{11}}\, s\, z^{\mathsf{T}}.$$

Daraus können die Aussagen abgeleitet werden. Ist A etwa hermitesch, dann gilt $a_{11} \in \mathbb{R}$, $s = \bar{z}$ und $B^{(1)} = {B^{(1)}}^*$. Daher ist auch $B^{(2)}$ hermitesch. Etwas schwieriger ist die Definitheit. Bei positiv definiter Matrix A gilt $x^* A x > 0$ für alle $x \in \mathbb{K}^n, x \neq 0$. Für $x^{\mathsf{T}} = e^{(1)} = (1, 0, \ldots, 0)$ folgt $a_{11} > 0$, im ersten Schritt war also keine Zeilenvertauschung notwendig. Nun sei mit $y \in \mathbb{K}^{n-1}, y \neq 0$, speziell

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

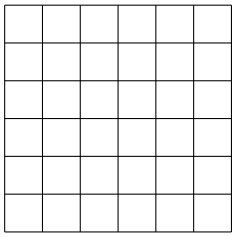
Also ist $0 < \hat{x}^* A \hat{x} = y^* B^{(1)} y - \frac{1}{a_{11}} |s^* y|^2 = y^* B^{(2)} y$ nach Voraussetzung, d.h., $B^{(2)}$ wieder positiv definit. Bei den anderen Eigenschaften geht man ähnlich vor.

b) folgt aus Teil a), da jedes Diagonalelement der Restmatrix positiven Betrag hat nach Definition. ∎

Bemerkung: Bei 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 Nummerierung der Gleichungen/Variablen ab und kann so beeinflusst werden. Bei geeigneter Implementierung bestimmt v.a. die Zahl der nichttrivialen Elemente der LR-Zerlegung den Rechenaufwand (z.B. Bandstruktur: $O(m^2 n)$). Ohne die Ausnutzung solcher Struktureigenschaften wären sehr große Systeme auch auf modernen Supercomputern nicht lösbar.

Beispiel 4.3.5 Poisson-Gleichung auf dem Einheitsquadrat mit einem $m \times m$-Gitter, $n = m^2$. Die Unbekannten x_i entsprechen Werten auf den Kreuzungspunkten im skizzierten Gitter, die mit ihren vier Nachbarn verknüpft sind. Daher besteht das LGS i.w. aus Gleichungen der Form $4x_i - x_{i-1} - x_{i+1} - x_{i-m} - x_{i+m} = b_i$, wobei Werte, die außerhalb des Gitters liegen (etwa mit Indizes außerhalb $\{1, \ldots, m^2\}$) bekannt sind.



In der angegebenen Nummerierung hat die Matrix Bandstruktur mit Bandbreite $2m + 1$, allerdings sind höchstens 5 Elemente pro Zeile nichttrivial. Beim Gauß-Algorithmus wird das Band vollständig aufgefüllt, aus $5m^2$ ursprünglichen Elementen entstehen ca. $2m^3$ in der LR-Zerlegung. Mit anderen Nummerierungen, die allerdings eine dynamische Speicherung der Matrixelemente erfordern, entsteht wesentlich weniger "fill-in".

Anzahl Matrixelemente bei LR-Zerlegung

$m, n =$	7, 49	15, 225	31, 961	63, 3969	127, 16129	255, 65025
Matrix A:	217	1065	4681	19593	~ 80000	~ 324000
Band-LR:	649	6553	58681	496249	$\sim 4 \cdot 10^6$	$\sim 33 \cdot 10^6$
LR, dünn:	527	4375	29319	172327	~ 930000	$\sim 4.7 \cdot 10^6$

Die aktuelle Top-500-Liste der Supercomputer (Quelle: *www.top500.org*, 11/2016, u.a. HRZ Mannheim) weist folgende gemessenen Leistungen aus (TFLOPS=10^{12} Operationen pro Sekunde). Diese Leistungen sind nur durch Parallelisierung erreichbar, der schnellste Rechner *Sunway TaihuLight* leistet real 93014 TeraFLOPS ($< 10^{17}$) mit 10 Millionen Prozessoren. Der Rechner Juqueen in Jülich leistet 5000 TFLOPS. Die in der Tabelle genannte Dimension n gibt die Größe eines vollbesetzten LGS an, das in 1 sec Rechenzeit gelöst werden könnte. Bei einer Rechenzeit von einer Stunde vergrößert sich die Dimensionen (nur) um den Faktor $\sqrt[3]{3600} \cong 15$.

Platz	Name/Typ	Ort	Proz.	TFLOPS	n
1	Sunway TaihuLight	Wuxi/China	10^7	93014	518656
3	Titan/Cray	Oak-Ridge/USA	560640	17590	297704
19	Hazel Hen/Cray	Jülich/De	458752	5008	195847

Zeilenvertauschungen sind beim Gauß-Algorithmus theoretisch nur für $a_{kk}^{(k)} = 0$ erforderlich, also wenn das Pivotelement exakt verschwindet. Bei der Rechnung mit *endlichen Zahldarstellungen* muß man aber auch kleine Pivotelemente vermeiden, damit keine unnötigen Fehler auftreten. Dies zeigt das folgende Beispiel.

Beispiel 4.3.6 Alle (Zwischen-)Ergebnisse werden in der Gleitkomma-Darstellung auf zwei Stellen gerundet. Erste Elimination, aus

$$\left.\begin{array}{rcrcl} 0.01\,x_1 & + & 2\,x_2 & = & 1 \\ x_1 & + & x_2 & = & 3 \end{array}\right\} \quad \text{wird} \quad \left\{\begin{array}{rcrcl} 0.01\,x_1 & + & 2x_2 & = & 1 \\ 0\cdot x_1 & - & 200\,x_2 & = & -97 \end{array}\right. ,$$

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

$$\begin{array}{rcl} a_{22}^{(2)} & = & a_{22} - l_{21}a_{12} = 1 - 100 \cdot 2 = -199 \quad \text{Rundung: } \widetilde{a_{22}^{(2)}} = -2 \cdot 10^2 \\ b_2^{(2)} & = & b_2 - l_{21}b_1 = 3 - 100 = -97 \quad \text{(ist exakt).} \end{array} \tag{4.3.11}$$

Damit erhält man $\widetilde{x_2} = gl(-97/(-200)) = 0.49$, $\widetilde{x_1} = gl((1 - 2 \cdot 0.49)/0.01) = 2.0$. Die exakte Lösung ist jedoch $(2.512.., 0.487..)$. Vertauscht man die beiden Gleichungen im Ausgangssystem, so ergibt die Elimination dagegen

$$\left.\begin{array}{rrl} x_1 & +x_2 & = 3 \\ 0.01x_1 & +2x_2 & = 1 \end{array}\right\} \quad \rightarrow \quad \left.\begin{array}{rrl} x_1 & +x_2 & = 3 \\ & 2x_2 & = 0.97 \end{array}\right\} \quad \rightarrow \quad \left\{\begin{array}{rl} \tilde{x}_1 & = 2.5 \\ \tilde{x}_2 & = 0.49 \end{array}\right.$$

Ursache für die Fehler in der ersten Variante ist der große Größenunterschied bei den Subtraktionen (4.3.11) und für $\tilde{x}_1$, verursacht durch Division mit dem kleinen Pivotelement 0.01.

Vermeidbare Ungenauigkeiten begrenzt man durch *Pivot-Strategien*, die kleine Nenner a_{kk} beim Basisschritt $a_{ij} := a_{ij} - l_{ik}a_{kj}$, $l_{ik} = a_{ik}/a_{kk}$ verhindern. Dabei wird im Schritt k zur Vertauschung mit Zeile k die Pivotzeile $p \in \{k, \ldots, n\}$ nach folgender Regel gewählt:

1. Spaltenpivotisierung, absolutes Maximum: $|a_{pk}^{(k)}| = \max\limits_{i=k}^{n} |a_{ik}^{(k)}|$. $\Big(\Rightarrow |l_{ik}| \leq 1 \Big)$.
2. Spaltenpivotisierung, relatives Maximum: $\dfrac{|a_{pk}^{(k)}|}{\sum_{j=k}^{n} |a_{pj}^{(k)}|} = \max\limits_{i=k}^{n} \dfrac{|a_{ik}^{(k)}|}{\sum_{j=k}^{n} |a_{ij}^{(k)}|}$.
3. Vollständige Pivotisierung, zusätzlich sind Spaltenvertauschungen (Umnumerierung der Variablen) vorzunehmen $|a_{pq}^{(k)}| = \max\limits_{i,j=k}^{n} |a_{ij}^{(k)}|$.

Da pro Eliminationsschritt bei Strategie 1) nur die k-te Spalte und die k-te und p-te Zeile bearbeitet werden, ist deren Aufwand mit $O(n^2)$ geringer als der zur Elimination. Die aufwändigen Strategien 2) und 3) werden nur in Ausnahmefällen benutzt. Strategie 2) unterdrückt den Einfluß der Multiplikation von Zeilen mit einer Konstanten (Skalierung). Ein ähnlicher Effekt ist auch bei der Teilpivotisierung 1) erzielbar durch vorherige *Äquilibrierung* der Matrix A, bei der zu Beginn alle Zeilen auf Summennorm 1 skaliert werden.

Bei Pivotisierung wird natürlich keine direkte LR-Zerlegung von A berechnet. Zeilenvertauschungen lassen sich aber als Multiplikation mit Permutationsmatrizen $P = (\delta_{i,\pi(j)})$, $P^{\mathsf{T}}P = I$, schreiben mit dem Kroneckersymbol δ. Gespeichert wird natürlich nicht die Matrix P sondern die Permutation $\pi(\cdot)$ in einem Indexfeld. Damit lassen sich wieder verschiedene LGSe mit der Matrix A lösen. Abschließend gilt also:

Der Gauß-Algorithmus mit absoluter Spaltenpivotisierung erzeugt eine Zerlegung

$$A = PLR = (PLP^{\mathsf{T}})(PR). \tag{4.3.12}$$

der Aufwand dafür ist $\frac{2}{3}n^3 + O(n^2)$ *Operationen.*

Inverse Matrix A^{-1}:

Für die praktische Rechnung lohnt es sich **nicht**, die Inverse A^{-1} für die Darstellung $x = A^{-1}b$ explizit zu berechnen, auch nicht bei mehreren LGSen $Ax^{(j)} = b^{(j)},\ j = 1, 2, \ldots$. Denn der Rechenaufwand für das Produkt $x = A^{-1}b$ ist der *gleiche*, wie der zur Lösung der beiden Dreiecksysteme bei $LRx = b$, jeweils $2n^2$ (vgl. Berechnung nach (4.3.4)), die Berechnung von A^{-1} amortisiert sich daher nie. Wenn A^{-1} doch explizit berechnet werden soll, ergeben sich bei vorhandener LR-Zerlegung die Spalten von

$$A^{-1} = C = \left(c^{(1)}, \ldots, c^{(n)}\right) \quad \text{aus} \quad LRc^{(j)} = e^{(j)},\ j = 1, \ldots, n. \tag{4.3.13}$$

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.

4.4 Kondition, Rundungsfehleranalyse

Bei realistischen Größenordnungen erfordern LGSe sehr viele (inexakte!) Rechenoperationen. Daher muß die erreichbare Genauigkeit untersucht werden. Nicht nur die Rundungsfehler im Gauß-Algorithmus, sondern schon die Rundung der Eingangsdaten A und b kann sich bei fast-singulären LGSen katastrophal auf die Lösung auswirken. Diese Fehleranfälligkeit eines LGS läßt sich bei geeigneter Formulierung durch eine einzige Größe, die *Konditionszahl* der Matrix, charakterisieren. Im Unterschied zu §3 betrachtet man dabei Fehler nicht komponentenweise, sondern ganzheitlich mit Normen. Eingabefehlern berücksichtigt man dadurch, dass man statt des ursprünglichen Problems $Ax = b$ mit regulärer Matrix A das Problem

$$(A + A')\tilde{x} = b + b', \quad \tilde{x} = x + x' \tag{4.4.1}$$

mit gestörter Matrix $\tilde{A} = A + A'$ und gestörter rechter Seite $\tilde{b} = b + b'$ exakt löst. Durch eine geeignete Betrachtungsweise wird die Rundungsfehleranalyse des Gauß-Algorithmus übrigens auf diesen Fall zurückgeführt.

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

$$\begin{aligned} (I + A^{-1}A')(x + x') &= A^{-1}(b + b') = x + A^{-1}b' \\ \iff (I + A^{-1}A')x' &= A^{-1}b' - A^{-1}A'x \end{aligned} \tag{4.4.2}$$

Nach Satz 4.2.2 (Neumannreihe) ist dieses LGS lösbar, wenn in einer Norm gilt

$$\|A^{-1}A'\| \le \|A^{-1}\|\,\|A'\| =: q < 1. \tag{4.4.3}$$

Dann folgt die Schranke

$$\|x'\| \leq \frac{\|A^{-1}b'\| + q\|x\|}{1-q}. \tag{4.4.4}$$

Besonders aussagekräftig sind Abschätzungen, die *skalierungsinvariant* sind und sich bei Multiplikation des LGS mit Konstanten nicht ändern. Daher betrachtet man wie bei der Analyse von Gleitpunktrechnung (§3.1) relative Fehler. Mit $\|b\| = \|Ax\| \leq \|A\|\|x\|$ folgt so aus (4.4.4):

$$\frac{\|x'\|}{\|x\|} \leq \frac{1}{1-q}\Big(\underbrace{\|A^{-1}\|\,\|A\|}\ \frac{\|b'\|}{\|b\|} + q\Big).$$

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

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

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

Definition 4.4.1 *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., $\kappa_p(A) = \|A\|_p\|A^{-1}\|_p,\ p \geq 1$.*

Diese Konditionszahl beschreibt die Empfindlichkeit des LGS gegenüber Störungen. Sie ist skalierungsinvariant, es ist $\kappa(tA) = \kappa(A)\,\forall t \in \mathbb{R}\setminus\{0\}$. Für verträgliche Matrixnormen gilt $\kappa(A) \geq 1$, denn $1 \leq \|I\| = \|AA^{-1}\| \leq \|A\|\|A^{-1}\|$. Die gerade durchgeführte Diskussion hat folgende Fehlerabschätzung bewiesen:

Satz 4.4.2 *Im gestörten System (4.4.1) werden die folgenden relativen Fehler betrachtet:*

$$\epsilon_x := \frac{\|\tilde{x}-x\|}{\|x\|} = \frac{\|x'\|}{\|x\|}, \quad \epsilon_b := \frac{\|\tilde{b}-b\|}{\|b\|} = \frac{\|b'\|}{\|b\|}, \quad \epsilon_A := \frac{\|\tilde{A}-A\|}{\|A\|} = \frac{\|A'\|}{\|A\|}.$$

Unter der Voraussetzung

$$\kappa(A)\cdot\epsilon_A < 1, \tag{4.4.5}$$

ist das gestörte LGS (4.4.1) eindeutig lösbar und der relative Fehler in der Lösung ist beschränkt durch

$$\epsilon_x \leq \frac{\kappa(A)}{1-\kappa(A)\epsilon_A}\ \Big(\epsilon_A + \epsilon_b\Big). \tag{4.4.6}$$

Für $\kappa(A)\epsilon_A \ll 1$ gilt i.w. ein linearer Zusamenhang $\epsilon_x \leq \kappa(A)\,(\epsilon_A + \epsilon_b) + O(\epsilon^2)$. Für große $\kappa(A) \gg 1$ ist das Problem aber *schlecht konditioniert*. Dann können schon kleine Störungen der Daten zu erheblichen Änderungen der Lösung führen. Wegen dieser zentralen 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 zu bestimmen ist als, z.B., $\|\cdot\|_2$, werden in der Praxis gerne Abschätzungen von $\|A^{-1}\|_\infty$ verwendet.

a) Für streng diagonaldominante Matrizen (4.3.10) ist $A = D(I - D^{-1}B)$ mit $D = \mathrm{diag}(a_{ii})$ und $\|D^{-1}B\|_\infty < 1$. Aus Satz 4.2.2 folgt daher

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

b) Wenn die LR-Zerlegung $A = LR$ für die Auflösung berechnet ist, kann mit der folgenden Methode mit geringem Zusatzaufwand $O(n^2)$ auch eine Schranke für $\|A^{-1}\|_\infty \le \|L^{-1}\|_\infty \|R^{-1}\|_\infty$ bestimmt 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.

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

$$\|B\|_\infty = \|B\mathbb{1}\|_\infty, \quad \mathbb{1} = (1, 1, \ldots, 1)^\mathsf{T}.$$

2. Für $C^{-1} \ge 0$ ist $\|C^{-1}\|_\infty = \|C^{-1}\mathbb{1}\|_\infty = \|z\|_\infty$, wobei z das LGS $Cz = \mathbb{1}$ löst. Damit kann die Norm $\|C^{-1}\|_\infty$ berechnet werden, ohne C^{-1} explizit zu kennen!

3. Die Norm $\|\cdot\|_\infty$ ist monoton, d.h. aus

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

4. Zerlegt man $R = D(I - B)$ in die Diagonale $D = \mathrm{diag}(r_{ii})$ und den Rest, dann ist B strikt obere Dreieckmatrix mit $\rho(B) = 0 \Rightarrow B^n = 0$. Aus dem Satz 4.2.2 (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 1. - 4. führen auf $\|R^{-1}\|_\infty \le \|(|D| - |DB|)^{-1}\mathbb{1}\|_\infty$, d.h., auf

Satz 4.4.3 *$R \in \mathbb{K}^{n\times n}$ sei eine reguläre obere Dreieckmatrix. Der (positive) Vektor $z \in \mathbb{R}^n$ sei definiert durch das LGS*

$$|r_{ii}|z_i - \sum_{j=i+1}^{n} |r_{ij}|z_j = 1, \quad i = n, n-1, \ldots, 1. \tag{4.4.7}$$

Dann gilt

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

Beispiel 4.4.4

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

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 $P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = P^{-1}$.

Rückwärts-Analyse der Fehler

Der letzte Abschnitt hat gezeigt, dass für sehr schlecht konditionierte Probleme auch der beste numerische Algorithmus keine genaue Lösung berechnen kann. Der grundlegende Fehler wurde in diesem Fall schon vom *Problemsteller bei der Formulierung* gemacht. Um die Verantwortlichkeiten zu trennen, entwickelt man in der Numerik Algorithmen so, dass deren Ergebnisse die *exakte* Lösung eines nur wenig gestörten Originalproblems sind. In Bezug auf den Gauß-Algorithmus (u.a.) bedeutet diese Sichtweise, dass man die berechnete Lösung $\tilde{x}$ als Lösung von

$$(A + A')\tilde{x} = b, \qquad \|A'\| \le c\,\mathcal{E}\|A\|, \tag{4.4.8}$$

vgl. (4.4.1), ansehen kann mit einer moderaten Konstanten c (hängt nur leicht von n ab). Diese Sichtweise nennt man *Rückwärts-Analyse*. Bei einem gutartigen Algorithmus, der (4.4.8) erfüllt,

werden große Fehler in der Lösung $\tilde{x}$ dann durch die ungünstige Problemstellung mit einer großen Konditionszahl $\kappa(A)$ verursacht.

Die Rückwärtsanalyse der LR-Zerlegung kann mit Satz 4.3.4 und den Techniken aus §3 geführt werden und liefert folgende Abschätzung, in der die tatsächlich berechneten Faktoren $\tilde{L}, \tilde{R}$ eingehen.

$$\tilde{L}\tilde{R} = A + A', \quad |A'| \leq \mathcal{E} \cdot 3(n-1)(|A| + |\widetilde{L}||\widetilde{R}|) + O(\mathcal{E}^2). \tag{4.4.9}$$

In der Schranke ist als problematischster Anteil das Betragsprodukt $|\widetilde{L}||\widetilde{R}|$ zu erkennen, dessen Elemente möglicherweise viel größer als die von $|A|$ $(= |LR| \leq |L||R|)$ sein können. Dies ist insbesondere dann der Fall, wenn kleine Pivotelemente auftreten und dann $|l_{ik}| = |a_{ik}^{(k)}/a_{kk}^{(k)}| \gg 1$ ist, vgl. (4.3.4). Der Satz ist auch bei Pivotisierung anwendbar, da der Gauß-Algorithmus bei Spaltenpivotisierung nach (4.3.12) die LR-Zerlegung eines zeilenpermutierten Systems $(P^{\mathsf{T}}A)x = P^{\mathsf{T}}b$ berechnet. Bei dieser Pivotisierungsstrategie gilt $|\widetilde{l_{ik}}| \leq 1$ für den berechneten L-Faktor in $A = PLR$. Auch bei den Elementen von $\tilde{R}$ kann man dann von kleineren Beträgen ausgehen.

Die Hauptaufgabe der Pivotisierung ist es also, das Anwachsen der Elemente in den L- und R-Faktoren zu begrenzen, damit das Ausgangsproblem $Ax = b$ gemäß der Schranke (4.4.9) möglichst wenig gestört wird. Bei einer moderaten Kondition $\kappa(A)$ ist dann nach Satz 4.4.2 der Fehler der berechneten Lösung $\tilde{x}$ tatsächlich klein.

4.5 Iterationsverfahren für lineare Gleichungssysteme

Auch mit exakten Lösungsverfahren wird bei numerischer Durchführung nur eine fehlerbehaftete Lösung berechnet. Daher kann man auch von vorneherein Verfahren betrachten, welche mit Hilfe konvergenter Folgen nur Approximationen der Lösung bestimmen. Für lineare Gleichungssysteme der Form $(I - B)x = r$ mit "kleiner" Matrix B, d.h., mit Spektralradius $\rho(B) < 1$, kennt man eine explizite Darstellung der Lösung z über die Neumannreihe (Satz 4.2.2),

$$z = (I - B)^{-1}r = \sum_{j=0}^{\infty} B^j r = r + B\sum_{j=0}^{\infty} B^j r. \tag{4.5.1}$$

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

$$x^{(k+1)} := Bx^{(k)} + r, \quad k = 0, 1, \ldots, \tag{4.5.2}$$

berechnet werden mit den Näherungen $x^{(1)} = r,\ x^{(2)} = r + Br,\ x^{(3)} = r + Br + B^2r, \ldots$. Die Konvergenzaussage läßt sich erheblich verallgemeinern. Die Iteration (4.5.2) dient nämlich zur Bestimmung eines *Fixpunkts* der affin-linearen Abbildung $x \mapsto g(x) := Bx + r$ mit

$$z = g(z), \qquad g : \mathbb{R}^n \to \mathbb{R}^n. \tag{4.5.3}$$

Allgemein gilt für *kontrahierende* Abbildungen, die eine *Lipschitz-Bedingung* (4.5.5) erfüllen, der

Satz 4.5.1 (Banachscher Fixpunksatz) *Gegeben sei ein Banachraum H und eine Abbildung $g: H \to H$. Wenn auf einer abgeschlossenen Teilmenge $\Omega \subset H$ mit $q \in \mathbb{R}, 0 < q < 1$, die Aussagen*

$$g(\Omega) \subseteq \Omega \qquad (\text{“}g \text{ bildet } \Omega \text{ auf sich ab”}) \tag{4.5.4}$$

$$\|g(x) - g(y)\| \leq q\|x - y\| \ \forall x, y \in \Omega, \quad (\text{“}g \text{ ist kontrahierend”}) \tag{4.5.5}$$

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

$$x^{(k+1)} := g(x^{(k)}), \quad k = 0, 1, \ldots, \tag{4.5.6}$$

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

$$\|x^{(k)} - z\| \leq \frac{q}{1-q}\|x^{(k)} - x^{(k-1)}\| \qquad (\text{“}a\text{-}posteriori\text{”}) \tag{4.5.7}$$

$$\leq \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\| \qquad (\text{“}a\text{-}priori\text{”}). \tag{4.5.8}$$

Bemerkung: Die beiden Fehlerschranken besitzen unterschiedliche Anwendungen,

(4.5.7): Verwendung als Abbruchkriterium in der Iteration, d.h., wenn $x^{(k)}$ berechnet ist,

(4.5.8): Abschätzung der zum Erreichen einer bestimmten Genauigkeit erforderlichen Anzahl von Iterationen, d.h., des Aufwands.

Beweis Wegen (4.5.5) gilt induktiv

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

Aus der Dreieckungleichung folgt daraus für beliebiges $m > 0$ die Schranke

$$\begin{aligned}\|x^{(k+m)} - x^{(k)}\| &\leq \sum_{j=0}^{m-1} \|x^{(k+j+1)} - x^{(k+j)}\| \leq \sum_{j=1}^{m} q^j\|x^{(k)} - x^{(k-1)}\| \\ &\leq \frac{q}{1-q}\|x^{(k)} - x^{(k-1)}\| \leq \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\| \ \to 0, \ k \to \infty.\end{aligned} \tag{4.5.9}$$

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

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

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

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

Nach (4.5.9) folgen aus $x^{(k)} - z = \lim_{m\to\infty} \left(x^{(k)} - x^{(k+m)}\right)$ die Schranken wie

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

Dieser Satz ist die Grundlage für viele Konvergenzaussagen bei Iterationsverfahren, auch bei nichtlinearen Problemen (vgl. §5). Die Lipschitzbedingung (4.5.5) bei der affin-linearen Funktion $g(x) = Bx + r$ lautet $\|g(x) - g(y)\| = \|B(x-y)\| \le \|B\|\|x-y\|$. Die Kontraktivität, $\|B\| < 1$, läßt sich wegen Satz 4.2.1 durch die Forderung $\rho(B) < 1$ an den Spektralradius ersetzen.

Die Matrix A des LGSs liegt in der Regel natürlich nicht in der Form $A = I - B$ vor. Aber durch Abspaltung eines regulären *Haupt*-Teils M bei $A = M - N$, ist

$$Ax = Mx - Nx = b \iff M^{-1}Ax = x - \underbrace{M^{-1}N}_{=:B}x = M^{-1}b =: r \iff x = Bx + r.$$

Unter der Voraussetzung $\rho(B) = \rho(M^{-1}N) < 1$ konvergiert die Iteration (4.5.2). Für den praktischen Einsatz muß man natürlich $M^{-1}y$ einfach berechnen können. Einfache Fälle sind

Gesamt- und Einzelschrittverfahren

Beispiel 4.5.2 Betrachtet werde das folgende 2×2-Gleichungsystem mit Lösung $(2,1)^\mathsf{T}$.

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

“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 $\left(\begin{smallmatrix} 1 & 0.01 \\ -0.02 & 4 \end{smallmatrix}\right)$ als regulärer *Hauptteil* M benutzt. Als bessere (?!) Variante kann man zur Berechnung von $x_2^{(k+1)}$ statt $x_1^{(k)}$ den dann schon bekannten Wert $x_1^{(k+1)}$ verwenden. Tatsächlich basieren beide Verfahren auf der 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,

$$A = \begin{pmatrix} & & R \\ & D & \\ L & & \end{pmatrix}, \quad \text{d.h., } L = \begin{pmatrix} 0 & \cdots & 0 \\ * & \ddots & \vdots \\ * & * & 0 \end{pmatrix}, \quad R = \begin{pmatrix} 0 & * & * \\ \vdots & \ddots & * \\ 0 & \cdots & 0 \end{pmatrix}. \tag{4.5.10}$$

Gesamtschrittverfahren, Jacobi-Iteration: Zerlegung $M := D,\ N := -L - R$, die Iterationsmatrix ist $B_G = -D^{-1}(L+R)$. Der Iterationsschritt lautet für $k = 0, 1, 2, \ldots$:

$$\begin{array}{rcll} Dx^{(k+1)} & = & b - (L+R)x^{(k)}, & \iff \\ x_i^{(k+1)} & = & \frac{1}{a_{ii}}\Big(b_i - \sum\limits_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum\limits_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big), & i = 1, \ldots, n. \end{array} \tag{4.5.11}$$

Einzelschrittverfahren, Gauß-Seidel-Iteration: Zerlegung $M := D + L,\ N := -R$, die Iterationsmatrix ist $B_E = -(D+L)^{-1}R$ und der Iterationsschritt für $k = 0, 1, 2, \ldots$:

$$\begin{aligned} (D+L)x^{(k+1)} &= b - Rx^{(k)}, \qquad \Longleftrightarrow \\ x_i^{(k+1)} &= \tfrac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big), \quad i = 1, \ldots, n. \end{aligned} \tag{4.5.12}$$

Die Iterationsschritte (4.5.11) und (4.5.12) besitzen den gleichen Rechenaufwand, in (4.5.12) werden in der Zeile i allerdings für $j < i$ 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 (4.5.11), (4.5.12) = 2·(Anzahl der nichttrivialen Matrixelemente)

Diese Tatsache ist ein entscheidender Vorteil von Iterationsverfahren, da auch eine unregelmäßige *dünne Besetzung* von A direkt ausgenutzt werden kann, anders als bei Eliminationsverfahren.

Hinreichend für die Konvergenz beider Iterationen ist die *Diagonaldominanz* von A.

Satz 4.5.3 *Die Matrix A sei streng diagonaldominant, (4.3.10). Dann konvergieren Gesamt- und Einzelschrittverfahren. Genauer gilt*

$$\|B_E\|_\infty = \|(D+L)^{-1}R\|_\infty \le \|D^{-1}(L+R)\|_\infty = \|B_G\|_\infty < 1. \tag{4.5.13}$$

Somit gelten die Abschätzungen (4.5.8) in der Maximumnorm mit $q = \|B_G\|_\infty$ bzw. $q = \|B_E\|_\infty$.

Beweis Durch Übergang zu den Matrizen $D^{-1}A$, $D^{-1}L, D^{-1}R$ kann oBdA $D = I$ angenommen werden, die betrachtete Matrixnorm ist $\|.\|_\infty$.

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

b) Es werden analog zu Satz 4.4.3 Betragsabschätzungen benutzt und die stärkere Aussage

$$\|C\| \le \|B_G\|, \quad C := (I - |L|)^{-1}|R| \tag{4.5.14}$$

bewiesen. Damit ist dann die Behauptung (4.5.13) durch $\|B_E\| \le \|C\| \le \|B_G\|$ bewiesen, da im Satz 4.4.3 gezeigt wurde (man beachte $\varrho(L) = \varrho(|L|) = 0$)

$$0 \le |B_E| = |(I+L)^{-1}R| \le (I - |L|)^{-1}|R| = C.$$

Zum Nachweis von (4.5.14): Aus $C \ge 0$ folgt $\|C\| = \|z\|_\infty =: q$, wobei für $z := C\mathbb{1},\ \mathbb{1} = (1, \ldots, 1)^\mathsf{T}$ gilt

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

Für die Lösung dieses Gleichungssystems folgt

$$\begin{aligned} q &= \|z\|_\infty = \max_i \Big\{ \sum_{j<i} |a_{ij}|z_j + \sum_{j>i} |a_{ij}| \Big\} \\ &\le \max_i \Big\{ q \sum_{j<i} |a_{ij}| + \sum_{j>i} |a_{ij}| \Big\} = \|qL + R\|. \end{aligned} \tag{4.5.15}$$

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

$$1 \le \max_i \{ \sum_{j<i} |a_{ij}| + \frac{1}{q} \sum_{j>i} |a_{ij}| \} = \|L + \frac{1}{q} R\|,$$

die wegen Teil a) aber nur für $q < 1$ erfüllt sein kann. Denn wegen der Monotonie der Zeilensummennorm gilt $\|C\| = q \le \|qL + R\| \le \|L + R\| = \|B_G\| < 1$, also gilt (4.5.15) $\Rightarrow$ (4.5.14) $\Rightarrow$ (4.5.13). ∎

Nach diesem Satz konvergiert das Einzelschrittverfahren bezüglich der Zeilensummennorm nicht schlechter als das Gesamtschrittverfahren. Für bestimmte Matrixstrukturen kann sogar mit der entscheidenden Größe, dem Spektralradius eine bessere Konvergenz des ESVs gezeigt werden. Andererseits kann aber nur Iterationsschritt (4.5.11) des GSV parallel ausgeführt werden.

Hinreichende Kriterien für Konvergenz von Gesamt- und Einzelschritt-Verfahren sind die Diagonaldominanz von A oder A^{T}, sowie die positive Definitheit bei hermiteschen Matrizen.

Relaxationsverfahren

Die gerade beschriebenen Verfahren können oft durch Einführung eines Beschleunigungsparameters ohne höheren Rechenaufwand verbessert werden. Ohne Einschränkung wird schon von einer Matrix der Form $A = I - B$ ausgegangen. Mit $0 < \omega \in \mathbb{R}$ wird das Iterationsverfahrens (4.5.2) folgendermaßen modifiziert:

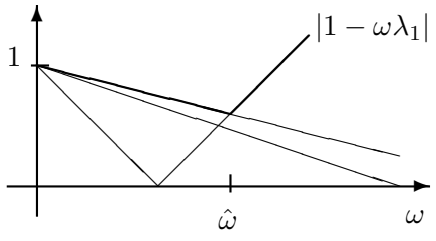
$$x^{(k+1)} := (1 - \omega)x^{(k)} + \omega(Bx^{(k)} + r) = x^{(k)} + \omega(r - Ax^{(k)}), \tag{4.5.16}$$

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

$$B(\omega) := (1 - \omega)I + \omega B = I - \omega A, \tag{4.5.17}$$

mit $B(1) = B$. Beim Ansatz (4.5.16),(4.5.17) will man natürlich ω so bestimmen, dass der Konvergenzfaktor $\rho(B(\omega))$ möglichst klein wird. Dabei gilt $\rho(B(0)) = 1$.

Die Eigenwerte von $B(\omega)$ sind $1 - \omega\lambda_i$ mit den Eigenwerten λ_i von A. $B = B(1)$ hat die Eigenwerte $\mu_i = 1 - \lambda_i$. Eine Optimierung des Parameters ω ist einfach, wenn alle Eigenwerte reell sind, etwa $\lambda_n \le \ldots \le \lambda_1$. Dann existiert für $\lambda_n > 0$ ein $\omega > 0$ mit $\rho(B(\omega)) = \max_i |1 - \omega\lambda_i| < 1$. Zur Minimierung sind nur die flachste und steilste Gerade zu betrachten,



$$\rho(B(\omega)) = \max\{|1 - \omega\lambda_1|, |1 - \omega\lambda_n|\}. \tag{4.5.18}$$

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

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

Satz 4.5.4 *Die Eigenwerte λ_i von A seien reell und positiv, $0 < \lambda_n \le \ldots \le \lambda_2 \le \lambda_1$. Dann ist der optimale Relaxationsparameter $\hat\omega$ bei (4.5.16) gegeben durch*

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

Bemerkung: 1) Die Konvergenz des einfachen Verfahrens ($\omega = 1$) war nicht vorausgesetzt, insbesondere ist $\mu_1 < -1$, $\lambda_1 > 2$, 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) Der Konvergenzfaktor $\rho(B(\hat\omega))$ hängt nur vom Verhältnis $\lambda_1/\lambda_n \le \kappa(A)$ der extremalen Eigenwerte ab, das in der Größenordnung der Kondition von A liegt.

4) 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 (4.5.16)) wird in der Vorlesung *Numerik IIA* behandelt.

Das Verfahren (4.5.16) 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 4.5.5 *SOR-Verfahren (successive overrelaxation)*

```
für i := 1, ..., n: {
   s := b_i;
   für j := 1, ..., n: {s := s − a_ij x_j;}
   x_i := x_i + ω s/a_ii;}
```

(4.5.19)

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

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

In der Matrix-Schreibweise

$$(D + \omega L)x^{(k+1)} := \Big((1-\omega)D - \omega R\Big)x^{(k)} + \omega b \tag{4.5.20}$$

wird allerdings erkennbar, dass die Iterationsmatrix

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

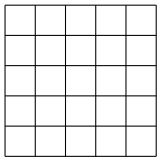
nichtlinear vom Parameter ω abhängt. Eine Analyse des Verfahrens ist für diese Vorlesung zu aufwendig. Als Beispiel einer Konvergenzaussage sei zitiert

Satz 4.5.6 *Für symmetrische und positiv definite Matrizen A gilt*

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

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 (vgl. Beisp. 4.3.5): Poisson-Gleichung auf dem Einheitsquadrat mit einem $m \times m$-Gitter, $n = m^2$. Das LGS besteht i.w. aus Gleichungen der Form $4x_i - x_{i-1} - x_{i+1} - x_{i-m} - x_{i+m} = b_i$, wobei für Randpunkte feste Werte eingesetzt werden.



Die Spektralradien sind explizit berechenbar, für $m = 10$ 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.

Allerdings wird die Konvergenz langsamer bei größeren Problemen, denn allgemein gilt für große $n = m^2$ in diesem Beispiel $\varrho(B_E) \cong 1 - \pi^2/m^2$. Daher sind ca. $-1/\log_{10} \varrho(B_E) \cong 0.23m^2 = 0.23n$ Iterationsschritte des ESV, sowie bei optimalem Parameter $\varrho(B_S) \cong 1 - 2\pi/m$, d.h., $-1/\log_{10} \varrho(B_S) \cong 0.37m = 0.37\sqrt{n}$ SOR-Schritte erforderlich. Da die Iterationsverfahren ca. $10n$ Operationen pro Schritt benötigen, führt die Annahme, dass der Anfangsfehler um 10^{-8} verkleinert werden soll, auf folgende Aufwandsschätzungen. Zum Vergleich ist der Aufwand beim Gauß-Algorithmus für Bandmatrizen ($2m^2n$) angegeben.

Verfahren	ESV	SOR	Band-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.

4.6 Ausgleichsprobleme

Nicht immer stimmt die Zahl m der Bedingungen in einem Problem mit der der Freiheitsgrade n überein. Will man etwa den Verlauf einer abhängigen Größe $y = f(t)$ durch Linearkombination spezieller Funktionen ϕ_j beschreiben, $f(t) = \sum_{j=1}^{n} x_j \phi_j(t)$, ist es bei praktischer Messung wegen der unvermeidlichen Meß- oder Modell-Fehler sinnvoll, die Freiheitsgrade x_j durch mehr als n Messungen zu bestimmen:

$$\sum_{j=1}^{n} \phi_j(t_i) x_j = y_i,\ i = 1, \ldots, m, \quad \Longleftrightarrow\ Ax = b, \tag{4.6.1}$$

mit $x = (x_j) \in \mathbb{K}^n$, $A = (a_{ij}) \in \mathbb{K}^{m \times n}$, $a_{ij} = \phi_j(t_i)$, $b = (b_i) = (y_i) \in \mathbb{K}^m$. Da ein solches System (4.6.1) für $m > n$ in der Regel überbestimmt ist, begnügt man sich damit, das *Residuum* bzw. den *Defekt* $r := Ax - b$ zu minimeren, $\min_x \|Ax - b\|$. Mit der Euklidnorm wird die Methode

besonders einfach, sie heißt *Methode der kleinsten Quadrate* nach Gauß (*least-squares-method*):

$$\|A\hat{x} - b\|_2^2 = \min_{x\in\mathbb{K}^n} \|Ax - b\|_2^2 = \min_{x\in\mathbb{K}^n} \sum_{i=1}^{m} \Big| \sum_{j=1}^{n} a_{ij}x_j - b_i \Big|^2. \tag{4.6.2}$$

Um in allen Fällen eine eindeutig definierte Lösung zu erhalten, ergänzt man diese Bedingung im Fall von mehreren Lösungen $\hat{x}$ durch eine weitere.

Definition 4.6.1 *Ein Vektor $\hat{x} \in \mathbb{K}^n$ heißt* Kleinste-Quadrate-Lösung *des Systems $Ax = b$, $A \in \mathbb{K}^{m\times n}$, $b \in \mathbb{K}^m$, wenn er (4.6.2) erfüllt. Unter diesen Minimalvektoren wird mit x^+ diejenige mit kleinster Norm bezeichnet,*

$$\|x^+\|_2 = \min\{\, \|\hat{x}\|_2 :\ \hat{x} \in \mathbb{K}^n \text{ erfüllt (4.6.2)}\}. \tag{4.6.3}$$

Für allgemeine Matrizen $A \in \mathbb{K}^{m\times n}$ benötigt man folgende Begriffe.

$$\begin{array}{lcll} N(A) & := & \{x \in \mathbb{K}^n :\ Ax = 0\} & \text{Nullraum bzw. Kern von } A, \\ R(A) & := & A\mathbb{K}^n = \{y \in \mathbb{K}^m :\ \exists x \in \mathbb{K}^n \text{ mit } Ax = y\} & \text{Bildraum von } A. \end{array} \tag{4.6.4}$$

Aus der Linearen Algebra kennt man den Zusammenhang

$$R(A)^\perp = N(A^*) \tag{4.6.5}$$

für das orthogonale Komplement von $R(A)$. Mit diesen Begriffen können die Lösungen $\hat{x}$ bzw. x^+ genauer charakterisiert werden.

Satz 4.6.2 *Es sei $P : \mathbb{K}^m \mapsto R(A)$ der Orthogonal-Projektor auf $R(A)$, er hat die Eigenschaften $P^2 = P,\ P^* = P$. Dann gilt:*

a) Es gibt Minimallösungen $\hat{x}$, die Bedingung (4.6.2) ist äquivalent mit

$$A\hat{x} = Pb, \qquad \text{sowie mit} \tag{4.6.6}$$

$$A^*A\hat{x} = A^*b, \quad \text{bzw.}, \quad A^*(A\hat{x} - b) = 0. \tag{4.6.7}$$

b) Die allgemeine Lösung von (4.6.2) hat die Form $\hat{x} + N(A)$. Die Lösung x^+ mit kleinster Norm (4.6.3) ist eindeutig, es gilt $x^+ \in N(A)^\perp$.

Bemerkung: Das System (4.6.7) nennt man "Normalengleichung". Bei überbestimmten Systemen mit $N(A) = \{0\}$ kann die Lösung $\hat{x} = x^+$ daraus mit Standardverfahren berechnet werden. Auf die gravierenden Nachteile dabei wird im nächsten Abschnitt eingegangen.

Beweis a) Äquivalenz mit (4.6.6): Wegen $Ax = PAx\,\forall x$ gilt

$$\begin{aligned} \|Ax - b\|_2^2 &= \|Ax - Pb - (I - P)b\|^2 \\ &= \|Ax - Pb\|_2^2 - 2\mathrm{Re}\Big(b^* \underbrace{(I - P^*)P}_{=0}(Ax - b)\Big) + \|(I - P)b\|_2^2. \end{aligned}$$

Da der letzte Summand nicht von x abhängt, bekommt man das Minimum für $\|Ax - Pb\|_2 = 0$. Dieses Minimum wird angenommen, da das System (4.6.6) wegen $Pb \in R(A)$ lösbar ist. Dieses System ist auch äquivalent mit (4.6.7), denn

$$\left.\begin{array}{lll} (4.6.6) & \Longleftrightarrow & \text{Defekt } r := A\hat{x} - b = (P-I)b \quad \in R(A)^\perp \\ (4.6.7) & \Longleftrightarrow & \text{Defekt } r := A\hat{x} - b \qquad\qquad\quad \in N(A^*) \end{array}\right\} \text{äquivalent wegen (4.6.5).}$$

b) Die Lösungsmenge des Systems (4.6.6) ist der affine Unterraum $\hat{x} + N(A)$. In diesem existiert genau ein Element kleinster Norm, $x^+ \in N(A)^\perp$, die Projektion des Nullpunkts auf diesen Unterraum. ■

Die Lösung wird also eindeutig bei Einschränkung der zu A gehörigen linearen Abbildung auf $N(A)^\perp$. Dies ist wohlbekannt, die eingeschränkte Abbildung

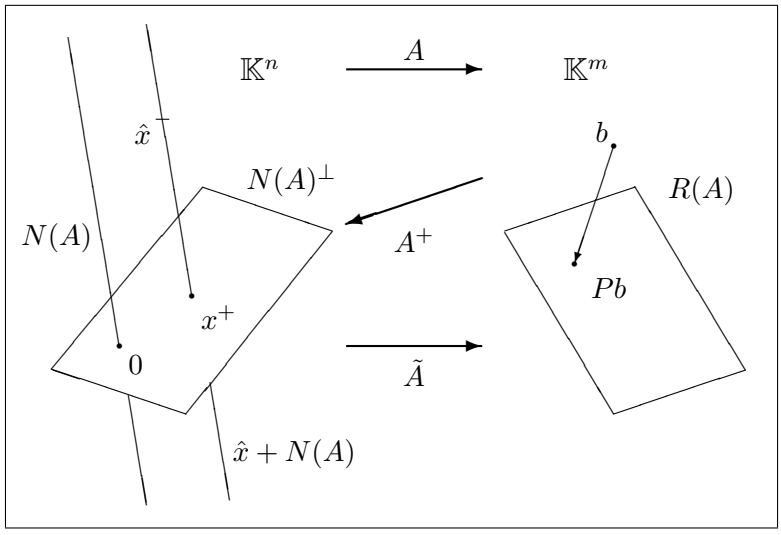
$$\tilde{A} = A|_{N(A)^\perp} : N(A)^\perp \to R(A)$$

ist sogar bijektiv und daher invertierbar. Daher ist die Zuordnung $b \mapsto x^+$ eine lineare Abbildung

$$A^+ : \mathbb{K}^m \to \mathbb{K}^n, \quad b \mapsto A^+ b := \tilde{A}^{-1} P\, b = x^+,$$

die man *Verallgemeinerte Inverse* oder *Pseudo-Inverse* nennt.

Skizze:



Diese Pseudoinverse A^+ existiert *für jede* Matrix A und stimmt bei regulärem A mit A^{-1} überein. Der Begriff erlaubt bei theoretischen Überlegungen eine kompakte Schreibweise für die Kleinste-Quadrate-Lösung $x^+ = A^+ b$, für numerische Zwecke wird sie aber **nicht** berechnet, genausowenig wie die übliche Inverse A^{-1}, vgl. die Schlußbemerkung zu §4.3.

Die QR-Zerlegung

Für Rang$(A) = n$ kann man das Ausgleichsproblems (4.6.2) im Prinzip durch Auflösung der Normalengleichung (4.6.7), $A^*Ax = A^*b$, lösen. Dies führt aber zu unnötig großen Fehlern. Im letzten Abschnitt wurde gerade gezeigt, dass die Konditionszahl der System-Matrix die Genauigkeit bestimmt. Die in der Normalengleichung auftretende Matrix ist aber $B = A^*A$ und ihre

Konditionszahl κ_2 im regulären Fall $m = n$ gerade

$$\kappa_2(B) = \kappa_2(A^*A) = \varrho(A^*A)\varrho(A^{-1}(A^*)^{-1}) = \kappa_2(A)^2.$$

Beim Übergang zur Normalengleichung wird die Konditionszahl des Ausgangssystems $Ax = b$ quadriert und für $\kappa(A) \gg 1$ also extrem vergrößert. Als Alternative sucht man Umformungen mit geringer Änderung der Kondition. Bei Betrachtung von κ_2 dienen dazu unitäre Umformungen, da diese *isometrisch* sind und insbesondere die Norm in (4.6.2) sogar unverändert lassen.

Als Alternative zur LR-Zerlegung betrachtet man daher die Zerlegung der Matrix in eine unitäre Matrix Q und eine obere Dreieckmatrix R, die sogenannte *QR-Zerlegung*,

$$A = QR, \quad Q \in \mathbb{K}^{m\times m},\ Q^*Q = I, \quad R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \end{pmatrix} \in \mathbb{K}^{m\times n}. \tag{4.6.8}$$

Mit dieser Zerlegung kann das Ausgleichsproblem direkt gelöst werden (s.u.). Auch quadratische Systeme $Ax = b$ können mit dieser Zerlegung gelöst werden, es gilt $x = R^{-1}Q^*b$. Diese Methode ist oft sogar numerisch stabiler als die übliche LR-Zerlegung, $A = L\tilde{R}$ (4.3.8), da $\kappa_2(R) = \kappa_2(QR) = \kappa_2(A)$ gilt. Dagegen kann $\kappa_2(\tilde{R}) \gg \kappa_2(L\tilde{R}) = \kappa_2(A)$ sein. Die Berechnung der QR-Zerlegung ist allerdings doppelt so teuer wie die der LR-Zerlegung.

Eine QR-Zerlegung (4.6.8) läßt sich auch mit dem Orthogonalisierungsverfahren von Gram-Schmidt (bei den Spalten von A) berechnen. Die dabei verwendeten Subtraktionen können aber zu großen Rundungsfehlern führen, wenn Spaltenvektoren von A beinahe linear abhängig sind (Auslöschung). Günstiger ist die Umformung auf Dreieckgestalt mit elementaren Spiegelungen (Householder-Transformationen).

Satz 4.6.3 *Mit einem Vektor $u \in \mathbb{K}^m$, $\|u\|_2 = 1$, ist die Matrix*

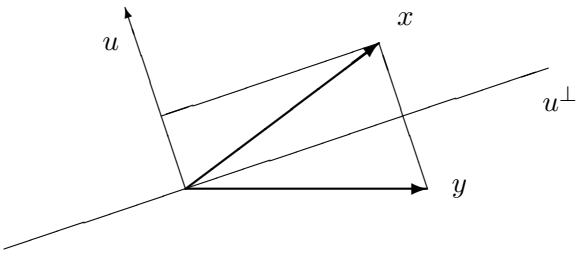
$$H := I - 2uu^*. \tag{4.6.9}$$

hermitesch ($H^* = H$), unitär ($H^*H = I$) und involutorisch ($H^2 = I$).

Beweis $H^* = I - 2uu^* = H \Rightarrow H^*H = H^2$, $H^2 = I - 4uu^* + 4(u^*u)uu^* = I$. ∎

Geometrische Bedeutung:

$y = Hx = x - 2(u^*x)u$
Spiegelung an $u^\perp$.



Diese Spiegelungen ersetzen die Zeilenoperation mit Matrizen L_j, (4.3.5), im Gauß-Algorithmus.

Im ersten Schritt ist also der Spiegelvektor u so zu bestimmen, dass ein vorgegebener Vektor x (die erste Spalte von A) auf die $e^{(1)}$-Achse gespiegelt wird:

$$Hx = x - 2(u^*x)u \stackrel{!}{=} \beta e^{(1)} \quad (= y). \tag{4.6.10}$$

Die Eigenschaften von H führen dabei auf folgende Nebenbedingungen:

$$\left.\begin{array}{lcl} H \text{ unitär} & \Rightarrow & |\beta| = \|x\|_2 \\ H \text{ hermitesch} & \Rightarrow & \mathbb{R} \ni x^*Hx = \beta x^* e^{(1)} = \beta \bar{x}_1 \end{array}\right\} \Rightarrow \ \beta = \pm\sigma\|x\|_2,$$

wobei $x_1 = \sigma|x_1|$, $|\sigma| = 1$ das Vorzeichen ist. Mit (4.6.10) und $\|u\|_2 = 1$ folgt die Darstellung

$$u = \frac{x - \beta e^{(1)}}{\|x - \beta e^{(1)}\|}.$$

Um bei der Differenzbildung $x_1 - \beta$ in der ersten Komponente von u eine Auslöschung zu vermeiden, wählt man das Vorzeichen von β so, dass eine Addition der Beträge stattfindet:

$$x_1 - \beta = \sigma|x_1| \mp \sigma\|x\|_2 \stackrel{!}{=} \sigma(|x_1| + \|x\|_2),$$

also $\beta = -\sigma\|x\|_2$. Damit wird

$$\|x - \beta e^{(1)}\|_2^2 = (|x_1| + \|x\|_2)^2 + |x_2|^2 + \ldots + |x_m|^2 = 2\|x\|(|x_1| + \|x\|).$$

Der Satz fasst diese Konstruktion zusammen.

Satz 4.6.4 *Der Vektor $0 \neq x \in \mathbb{K}^m$ mit $x_1 = \sigma|x_1|$, $|\sigma| = 1$, wird durch die Matrix*

$$H = I - \frac{vv^*}{\|x\|_2(|x_1| + \|x\|_2)}, \quad v := x + \sigma\|x\|_2 e^{(1)},$$

auf $-\sigma\|x\|_2 e^{(1)}$ abgebildet.

Die Elimination in den folgenden Spalten kann analog fortgeführt werden, da Produkte von Spiegelungen unitär sind. Dabei darf aber die erste Spalte nicht wieder verändert werden. Ausgehend von $A = A^{(1)}$ betrachtet man analog zu (4.3.3) die Struktur

$$A^{(k)} = \left(\begin{array}{ccc|ccc} r_{11} & \cdots & r_{1,k-1} & r_{1k} & \cdots & r_{1n} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & r_{k-1,k-1} & r_{k-1,k} & \cdots & r_{k-1,n} \\ \hline & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & 0 & & \vdots & & \vdots \\ & & & a_{mk}^{(k)} & \cdots & a_{mn}^{(k)} \end{array}\right) =: \begin{pmatrix} R^{(k)} & * \\ 0 & B^{(k)} \end{pmatrix} \tag{4.6.11}$$

mit $R^{(k)} \in \mathbb{K}^{(k-1)\times(k-1)}$, $B^{(k)} \in \mathbb{K}^{(m-k+1)\times(n-k+1)}$. Mit der Wahl

$$H^{(k)} = \left(\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & \tilde{H}^{(k)} \end{array}\right) \quad \Longleftrightarrow \quad u^{(k)\mathsf{T}} = (\underbrace{0, \ldots, 0}_{k-1}, \underbrace{*, \ldots, *}_{m-k+1}).$$

für die nächste Transformation bleiben beim Produkt $H^{(k)}A^{(k)} =: A^{(k+1)}$ die ersten $k-1$ Zeilen unverändert. Daher wird Satz 4.6.4 auf den Teilvektor

$$\tilde{H}^{(k)} \begin{pmatrix} a_{kk}^{(k)} \\ \vdots \\ a_{mk}^{(k)} \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} * \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{4.6.12}$$

angewendet. In $\ell := \min\{m-1, n\}$ Schritten bekommt man so die Zerlegung

$$A^{(\ell+1)} = R^{(\ell+1)} =: R = H^{(\ell)} \cdots H^{(2)} H^{(1)} A \quad \Longleftrightarrow \quad A = \underbrace{H^{(1)} H^{(2)} \cdots H^{(\ell)}}_{Q} R =: Q\,R.$$

in eine unitäre Matrix Q und die Dreieckmatrix R.

Algorithmus 4.6.5 QR-Zerlegung einer Matrix $A \in \mathbb{K}^{m\times n}$ in das Produkt einer unitären Matrix $Q \in \mathbb{K}^{m\times m}$ und einer oberen Dreieckmatrix $R \in \mathbb{K}^{m\times n}$ in $\ell = \min\{m-1, n\}$ Schritten:

$$\boxed{\begin{array}{l} \text{für } k := 1, \ldots, \ell\text{: } \{ \\ \quad c_k^2 := \sum_{i=k}^{m} |a_{ik}^{(k)}|^2,\ c_k > 0;\ \ d_k := c_k^2 + |a_{kk}^{(k)}| c_k; \\ \quad v^{(k)} := \left(0, \ldots, 0, \sigma_k c_k + a_{kk}^{(k)}, a_{k+1,k}^{(k)}, \ldots, a_{mk}^{(k)}\right)^{\mathsf{T}}, \text{ wo } \sigma_k |a_{kk}^{(k)}| = a_{kk}^{(k)}; \\ \quad \text{Anwendung der Transformation } H^{(k)} := I - \frac{1}{d_k} v^{(k)} v^{(k)*} \text{ in der Form} \\ \qquad A^{(k+1)} := A^{(k)} - v^{(k)} \Big(\underbrace{\frac{1}{d_k} v^{(k)*} A^{(k)}}_{z^{(k)*}} \Big); \\ \} \end{array}} \tag{4.6.13}$$

Die Zerlegung $A = QR$ liegt dann vor in der Form

$$Q = H^{(1)} \cdots H^{(\ell)}, \quad R = A^{(\ell+1)}, \tag{4.6.14}$$

$$\text{wobei} \quad R = \begin{pmatrix} r_{11} & \cdots & r_{1m} & \cdots & r_{1n} \\ & \ddots & \vdots & & \vdots \\ 0 & & r_{mm} & \cdots & r_{mn} \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \\ & 0 & \end{pmatrix}$$

$$\text{wenn} \qquad m \le n,\ \ell = m-1, \qquad\qquad m > n,\ \ell = n.$$

Praktische Durchführung, Rechenaufwand

Bei den Umformungen werden keine teuren Produkte von Matrizen berechnet, der Übergang $A^{(k+1)} = H^{(k)} A^{(k)}$ wird effizienter entsprechend der Klammerung im letzten Schritt von (4.6.13) durchgeführt. Auch Q wird nicht explizit berechnet. Bei der Auflösung mehrerer Gleichungssysteme mit der Matrix A kann das Produkt $Q^* y$ mit gleichem Aufwand in der Form $H^{(\ell)} \cdots H^{(1)} y$ ausgewertet werden. Dazu läßt man die Komponenten der Vektoren $v^{(k)}$ im freien Platz unter und einschließlich der Hauptdiagonale von R stehen. Nur die Elemente r_{kk} werden zur Vereinfachung in einem Zusatzfeld gespeichert. So erfordert die Durchführung folgenden *Rechenaufwand*:

Der Aufwand für c_k, d_k ist geringfügig und wird vernachlässigt. In (4.6.13) werden berechnet:

$$z^{(k)^*} := \frac{1}{d_k} v^{(k)^*} A^{(k)} = (\underbrace{0 \dots 0}_{k-1} \underbrace{* \dots *}_{m-k+1}) \left(\begin{array}{cc} \ddots & \\ \underbrace{0}_{k-1} & \underbrace{B^{(k)}}_{n-k+1} \end{array} \right) \Big\}_{m-k+1} = (\underbrace{0 \dots 0}_{k-1} \bullet \underbrace{* \dots *}_{n-k}).$$

Da das Ergebnis in der k-ten Spalte ('$\bullet$') nach Konstruktion (4.6.12) bekannt ist, sind für die Produktbildung $2(m-k+1)(n-k)$ Operationen erforderlich. Der Vektor $v^{(k)}$ besitzt höchstens $m-k+1$ und $z^{(k)}$ nicht mehr als $n-k$ nichttriviale Elemente. Daher benötigt die Bildung von $A^{(k+1)} = A^{(k)} - v^{(k)} z^{(k)^*}$ ebenfalls $2(m-k+1)(n-k)$ Operationen (Spalte k ist bekannt). Im Fall $n < m$ erhält man so den Gesamtaufwand

$$4 \sum_{k=1}^{n} (m-k+1)(n-k) = 4 \sum_{i=0}^{n-1} (m-n+i+1)i = \frac{4}{3} n^3 + 2(m-n)n^2 + \dots$$

Ein analoges Ergebnis gilt für $n \geq m$ und führt auf die Aussage

Rechenaufwand für die QR-Zerlegung einer $m \times n$-Matrix:

$$\frac{4}{3} \min\{m^3, n^3\} + 2|m-n| \min\{m^2, n^2\} + \dots \text{ Operationen.}$$

Bemerkung: Im regulären Fall mit $m = n$ ist die QR-Zerlegung also ungefähr doppelt so teuer wie eine LR-Zerlegung (mit oder ohne Pivotisierung). Dafür wird die Konditionszahl κ_2 von A bei der QR-Zerlegung nicht vergrößert: $\kappa_2(R) = \kappa_2(A)$.

Mit der QR-Zerlegung läßt sich das Kleinste-Quadrate-Problem (4.6.2) bei Matrizen von vollem Rang, $\text{Rang}(A) = \min\{m, n\}$, direkt lösen. Denn für $m \geq n$, $\text{Rang}(A) = n$, gilt

$$A = QR, \quad \text{mit } R = \begin{pmatrix} R_0 \\ 0 \end{pmatrix}, \ R_0 \in \mathbb{K}^{n \times n} \text{ regulär.} \tag{4.6.15}$$

Die Norm der transformierten rechte Seite $Q^* b = z = \begin{pmatrix} \hat{z} \\ \tilde{z} \end{pmatrix}$, $\hat{z} = (z_1, \dots, z_n)^\mathsf{T}$ wird jetzt aufgespalten:

$$\|Ax - b\|_2^2 = \|Q(Rx - Q^* b)\|_2^2 = \|Rx - z\|_2^2 = \|R_0 x - \hat{z}\|_2^2 + \|\tilde{z}\|_2^2. \tag{4.6.16}$$

Da die zweite Norm unabhängig von x ist, wird das Minimum in (4.6.16) im Minimum der ersten Norm angenommen, die Minimalstelle $\hat{x} = x^+$ ist Lösung des regulären Dreiecksystems

$$R_0 x^+ = \hat{z}.$$

Die Pseudoinverse zur linearen Abbildung $b \mapsto \hat{x} = x^+$ hat daher die Darstellung

$$A^+ = R_0^{-1} E_{nm} Q^*, \quad E_{nm} := \begin{pmatrix} 1 & & 0 \cdots \\ & \ddots & \vdots \\ & 1 & 0 \cdots \end{pmatrix} = (I_n, 0) \in \mathbb{R}^{n \times m}. \tag{4.6.17}$$

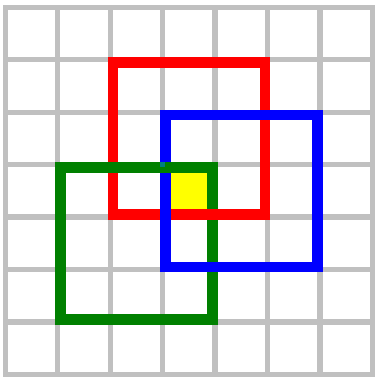
Die Matrix E_{nm} wählt dabei die ersten n Komponenten eines m-Vektors aus. Die Darstellung (4.6.17) ist äquivalent zu $A^+ = (A^*A)^{-1}A^*$, numerisch aber besser konditioniert (vgl. die Diskussion am Anfang des Abschnitts)! Der Projektor $P : \mathbb{K}^m \to R(A)$ ist übrigens

$$P = AA^+ = QE_{nm}^{\mathsf{T}} R_0 R_0^{-1} E_{nm} Q^* = Q \begin{pmatrix} I_n & 0 \\ 0 & 0 \end{pmatrix} Q^*. \tag{4.6.18}$$

Insbesondere bilden die ersten n Spalten QE_{nm}^{T} von Q eine Orthonormalbasis von $R(A) = N(A^*)^\perp$, und die letzten $m-n$ Spalten eine von $R(A)^\perp = N(A^*)$.

Aus dieser Beobachtung folgt auch sofort, dass die Lösung $x^+ \in N(A)^\perp = R(A^*)$ eines *unterbestimmten* Gleichungssystems, $m < n$, $\mathrm{Rang}(A) = m$, mit der QR-Zerlegung von $A^* = QR$ berechnet werden kann. Hier gilt $A^+ = QE_{mn}^{\mathsf{T}}(R_0^*)^{-1} : \mathbb{K}^m \mapsto \mathbb{K}^n$. Der allgemeine Fall $\mathrm{Rang}(A) < \min\{m,n\}$ läßt sich aus den beiden Basisfällen zusammensetzen. Eine weiteres Lösungsverfahren mit Hilfe einer aufwendigeren *Singulärwert-Zerlegung*, die allerdings auch zusätzliche Optionen bietet, wird in der Numerik IIA bei den Eigenwertproblemen behandelt.

Beispiel 4.6.6 *Bildrekonstruktion mit erhöhter Auflösung*
Bei Vergrößerung eines einzelnen Digital-Bildes können natürlich keine zusätzlichen Details erkennbar gemacht werden. Anders sieht es aus, wenn von einem Objekt (Schriftzug) mehrere Bilder niedriger Auflösung vorhanden sind, etwa aus einer Videoaufnahme.



Zur Vereinfachung wird folgende, künstliche Situation betrachtet. Von einem Rasterbild (grau) werden Aufnahmen mit einem Drittel der Auflösung gemacht, wobei in jedem groben Pixel (fett) eine Mittelwertbildung der Helligkeiten der feinen Pixel stattfindet. Bei einer langsamen Bewegung der Kamera oder des Objekts liefert jedes feine Pixel (gefüllt) bei verschiedenen Aufnahmen evtl. in unterschiedlichen Positionen der Grobpixel einen Beitrag. Daraus läßt sich ein Lineares Gleichungssystem für die unbekannten Helligkeitswerte x_{ij} des feinen Bildes aufstellen.
Wegen der Problemgröße (im Beispiel $n = 229441 = 479 \times 479$ Feinpixel) wurde nicht die QR-Zerlegung, sondern ein Iterationsverfahren eingesetzt. Das Bild zeigt eine der Kleinkopien (159×159), nach Rekonstruktion sind die beiden Schriftzüge lesbar.

5 Nichtlineare Gleichungssysteme

Viele praktische Probleme führen auf nichtlineare Gleichungen. Dies ist, z.B., auch bei den Problemen aus §4.1 der Fall, wenn man komplexere Wechselwirkungen betrachtet. Die unbekannten Größen $x \in \mathbb{R}^n$ sind dann implizit definiert durch Bedingungen in einer der beiden Formen

$$f(x) = 0, \qquad f : \mathbb{R}^n \mapsto \mathbb{R}^n, \tag{5.0.1}$$

$$x = g(x), \qquad g : \mathbb{R}^n \mapsto \mathbb{R}^n, \tag{5.0.2}$$

als Lösung eines *nichtlinearen Gleichungssystems*. Ausführlich geschrieben bedeutet das

$$\begin{cases} f_1(x_1, \ldots, x_n) &= 0 \\ \qquad\vdots & \\ f_n(x_1, \ldots, x_n) &= 0 \end{cases} \quad \text{bzw.} \quad \begin{cases} x_1 = & g_1(x_1, \ldots, x_n) \\ \quad\vdots & \\ x_n = & g_n(x_1, \ldots, x_n) \end{cases}.$$

Dabei wird im folgenden mindestens einmalige stetige Differenzierbarkeit der Funktionen f bzw. g angenommen. Die Gleichungen (5.0.1) und (5.0.2) sind Standardformen solcher Probleme, (5.0.1) heißt *Nullstellen-*, (5.0.2) *Fixpunktproblem*.

Für das Fixpunktproblem (5.0.2) kennt man im Banachschen Fixpunktsatz (Satz 4.5.1) ein hinreichendes Existenzkriterium *und* ein Lösungsverfahren (die Iteration). Das Nullstellenproblem (5.0.1) wird meist durch Umformungen auf eine Gestalt (5.0.2) gebracht, für die der Fixpunktsatz anwendbar ist. Zunächst wird der einfachste Spezialfall $n = 1$ betrachtet.

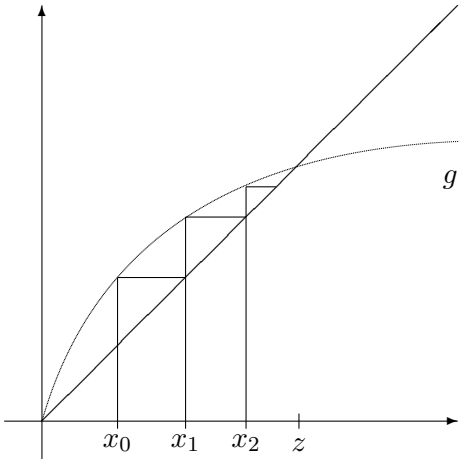
5.1 Nullstellen einer skalaren Funktion

Für reelle Funktionen g können die Aussagen des Fixpunktsatzes durch Monotoniebetrachtungen verfeinert werden. Es sei $g \in C^1[a, b]$.

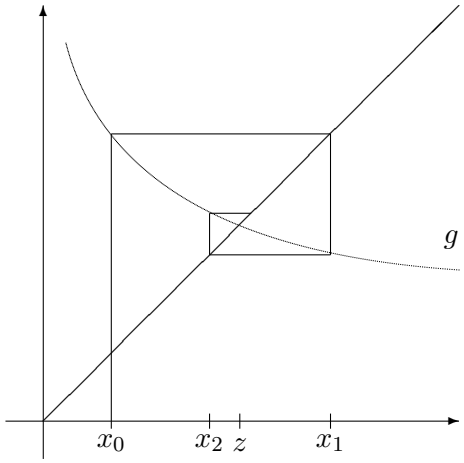
- Die Existenz eines Fixpunkts (oder einer Nullstelle) läßt sich schon aus dem Zwischenwertsatz folgern:
$$a \le g(a),\ g(b) \le b \quad \Rightarrow \quad \exists\, z = g(z) \in [a, b].$$
- Zur Überprüfung der Kontraktion im Banachschen Fixpunktsatz kann der Mittelwertsatz herangezogen werden, zur Lipschitzbedingung gilt
$$|g(x) - g(y)| = |g'(\xi)||x - y|.$$
Die Forderung $|g'| \overset{!}{<} 1$ kann man zur Vereinfachung nur im Fixpunkt z selbst stellen. Denn wenn $|g'(z)| < 1$ gilt, folgt aus der Stetigkeit von g' die Existenz einer ganzen Umgebung von z, in der $|g'(\xi)| \le q < 1$ ist. Dort ist dann die Konvergenz gesichert. Zusätzlich wirkt sich das Vorzeichen von $g'(z)$ auf die Konvergenz so aus, dass bei wachsendem g die Konvergenz monoton ist, bei fallendem g alternierend. Die Umgebung von z, in der tatsächlich Konvergenz gegen z eintritt, der sogenannte *Einzugsbereich* von z, kann

wesentlich größer sein als der Bereich $\{\xi : |g'(\xi)| < 1\}$ aus dem Fixpunktsatz. Im folgenden linken Beispiel hat die Funktion g den weiteren Fixpunkt 0 mit $g'(0) > 1$ ("abstoßender Fixpunkt"). Offensichtlich gilt aber Konvergenz gegen z für alle Startwerte $x_0 \in (0, z]$.

$0 < g'(z) < 1$: monotone Konvergenz



$-1 < g'(z) < 0$: alternierende Konvergenz



- Im Fall $g'(z) = 0$ tritt ein weiterer Effekt auf:

  Beispiel 5.1.1 $f(x) = x^2 - a = 0\,(a > 0) \iff x = \frac{a}{x} \iff 2x = x + \frac{a}{x}$.

  Es sei $g(x) := \frac{1}{2}(x + \frac{a}{x}) \Rightarrow$ 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

$$x_{k+1} - z = g(x_k) - g(z) = \underbrace{g'(z)}_{=0}(x_k - z) + \frac{1}{2}g''(\xi_k)(x_k - z)^2. \tag{5.1.1}$$

  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 Iterationsfolgen bzw. Iterationsverfahren ein.

Definition 5.1.2 *Eine Folge* $(x_k)_{k\geq 0}$ *mit* $\lim_{k\to\infty} x_k = z$, *heißt konvergent von der* Ordnung $p \geq 1$, *wenn* $k_0 \in \mathbb{N}$, $q \in \mathbb{R}_+$ *(wobei* $q < 1$ *bei Ordnung* $p = 1$ *sei) existieren mit*

$$|x_{k+1} - z| \leq q\,|x_k - z|^p \quad \forall k \geq k_0. \tag{5.1.2}$$

Mit der Ordnung eines Verfahrens meint man das maximal mögliche p. In den Fällen $p = 1, 2, 3$ spricht man speziell von linearer, quadratischer bzw. kubischer Konvergenz. Wichtig für die Konstruktion von Iterationsverfahren ist, dass sich die Ordnung an der Taylor-Entwicklung der Iterationsfunktion g im Fixpunkt z ablesen läßt (wie in (5.1.1)).

Satz 5.1.3 *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 außer dem letzten verschwinden ($g^{(p)}(\xi) \neq 0$ wegen der Stetigkeit). Daher gilt (5.1.2). Die Konvergenz folgt unter der präzisierten Voraussetzung $q\varepsilon^{p-1} < 1$ aus

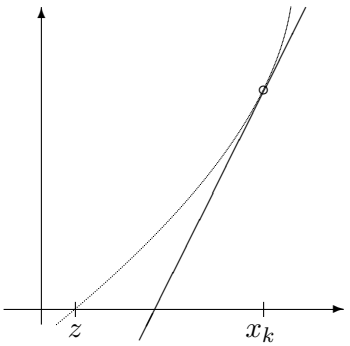
$$|x_{k+1} - z| \leq q|x_k - z|^p \leq (q\varepsilon^{p-1})|x_k - z|. \qquad \blacksquare$$

Bei der Konstruktion von Iterationsverfahren zur Nullstellenbestimmung benutzt man ein generelles Prinzip: Die Funktion f wird durch ein einfaches Modell approximiert (z.B. ein Polynom), dessen Nullstelle einfach berechnet werden kann. Dazu gibt es verschiedene Ansätze.

Newton-Verfahren

Approximation der Funktion f bei der Nullstelle z durch Abschnitte des Taylorpolynoms:

$$f(z) = \underbrace{\underbrace{0 \stackrel{!}{=} f(x_k) + f'(x_k)(z - x_k)}_{\to \text{ Newton-Verfahren}} + \frac{1}{2}f''(x_k)(z - x_k)^2 + \ldots}_{\to \text{ Newton-Raphson-Verfahren}}$$



Das lineare Modell führt auf das *Newtonverfahren*

$$x_{k+1} := g_N(x_k), \quad g_N(x) := x - \frac{f(x)}{f'(x)}\ , \tag{5.1.3}$$

das quadratische Modell auf das *Newton-Raphson-Verfahren*

$$x_{k+1} := g_R(x_k), \quad g_R(x) := x - \frac{2f(x)}{f'(x) + \mathrm{sign} f'(x)\sqrt{f'(x)^2 - 2f(x)f''(x)}} \ . \tag{5.1.4}$$

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 5.1.3 sind für $f'(z) \neq 0$ die Ableitungen von g_N zu berechnen:

$$\begin{aligned} g_N'(x) &= 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = f(x)\frac{f''(x)}{f'(x)^2} &&= 0 && \text{in } x = z, \\ g_N''(x) &= \frac{f'(x)^2 f''(x) + f(x)[\cdots]}{f'(x)^3} &&= \frac{f''(z)}{f'(z)} && \text{in } x = z, \end{aligned} \tag{5.1.5}$$

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 (5.1.4) konvergiert i.a. kubisch.

Bemerkung: 1) Verfahren höherer Ordnung konvergieren i.a. nur *lokal*, d.h., für genügend kleine Startfehler $|x_0 - z|$. Eine konkrete Aussage über den zulässigen Startfehler folgt im nächsten Abschnitt für den $\mathbb{R}^n$. Der *Einzugsbereich* des Newtonverfahrens kann in ungünstigen Fällen sehr klein sein. Dann ist es sehr schwer, einen geeigneten Startwert x_0 zu finden.

2) Auch bei mehrfachen Nullstellen, d.h., für $f'(z) = 0$, ist das Newtonverfahren noch einsetzbar. Bei $f(x) = (x-z)^m r(x),\ r(z) \neq 0$, gilt in der Nähe von z nämlich

$$g_N(x) = x - \frac{(x-z)^m r(x)}{m(x-z)^{m-1}r(x) + (x-z)^m r'(x)} = x - \frac{(x-z)r(x)}{mr(x) + (x-z)r'(x)} \quad \Rightarrow$$

$$x_{k+1} - z = g_N(x_k) - z = (x_k - z)\Big(1 - \frac{1}{m + (x_k - z)r'(x_k)/r(x_k)}\Big) \cong (1 - \frac{1}{m})(x_k - z).$$

Daher konvergiert das Newtonverfahren noch linear mit dem Kontraktionsfaktor $1 - \frac{1}{m} < 1$.

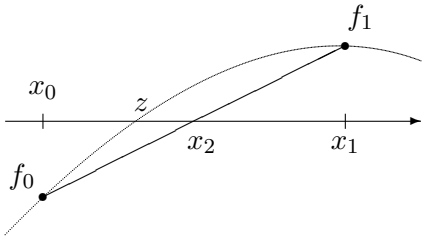
Mehrstufige Verfahren

Die oben behandelten Verfahren verbessern jeweils eine aktuelle Näherung x_k. Im Reellen ist es aber auch sinnvoll, weitere Größen mitzuführen, etwa um Einschließungen der Nullstelle z zu verwalten. Das folgende *Bisektionsverfahren* nutzt nur den Vorzeichenwechsel, um ausgehend von einer Starteinschließung $x_0 < y_0$ mit $f(x_0)f(y_0) < 0$ das Intervall jeweils zu halbieren,

$$m_k := \frac{x_k + y_k}{2}, \quad [x_{k+1}, y_{k+1}] := \begin{cases} [m_k, y_k] & \text{für} \quad f(m_k)f(y_k) < 0, \\ [x_k, m_k] & \text{für} \quad f(x_k)f(m_k) < 0. \end{cases} \tag{5.1.6}$$

Dieses einfache Verfahren konvergiert immerhin linear mit dem Faktor $q = 1/2$ ohne Differenzierbarkeitsvoraussetzung an f, nur $f \in C^0$ ist vorauszusetzen. Es kann dazu dienen, verläßliche Startwerte für das Newtonverfahren bereitzustellen.

Das Verfahren (5.1.6) läßt sich verbessern, wenn der "Testpunkt" m_k mit einem Modell der Funktion f bestimmt wird. Zur Approximation von $f \in C^1$ ist außer dem Taylorpolynom auch eine interpolierende Gerade einsetzbar. Durch zwei Funktions-Wertepaaren $(x_0, f_0), (x_1, f_1)$ legt man die *Sekante* und bestimmt daraus eine verbesserte Näherung:



$$0 \stackrel{!}{=} s(x) = f[x_1] + (x - x_1) f[x_1, x_0]$$

mit $f[x_1, x_0] = (f_1 - f_0)/(x_1 - x_0)$. Dies führt auf den neuen Wert

$$x_2 = x_1 - f_1 \frac{x_1 - x_0}{f_1 - f_0} = \frac{x_0 f_1 - x_1 f_0}{f_1 - f_0}. \tag{5.1.7}$$

Die erste Form zeigt die Ähnlichkeit mit dem Newtonverfahren, da $(x_1 - x_0)/(f_1 - f_0) \cong 1/f'(x_1)$ gilt. Es kommt aber ohne die Berechnung (und Implementierung) der Ableitung f' aus!

Verwendet man die Formel (5.1.7) mit den Werten x_k, y_k zur Bestimmung eines besseren Testpunkts m_k im Verfahren (5.1.6), ergibt sich oft keine Beschleunigung, da einer der Randpunkte x_k, y_k selten verändert wird (*Regula falsi*). Beim *Sekantenverfahren* verwendet man die Formel (5.1.7) als Vorschrift $x_{k+1} := G(x_k, x_{k-1})$, $k \geq 1$, eines 2-stufigen Iterationsverfahrens. Dieses weist eine Konvergenzordnung $p_S = (1 + \sqrt{5})/2 \doteq 1.618 > 1$ auf. Da es nur eine f-Auswertung pro Schritt verwendet, ist es im $\mathbb{R}^1$ sogar effizienter als das Newtonverfahren (2 Auswertungen f, f'). Eine Verallgemeinerung auf Systeme ist aber schwierig.

5.2 Newtonverfahren im $\mathbb{R}^n$

Bei der Lösung nichtlinearer Gleichungs*systeme* (5.0.1) mit $n > 1$ gibt es kaum eine Alternative zum Newtonverfahren. Nach Definition der Ableitung in einer Stelle $\bar{x} \in \mathbb{R}^n$,

$$f'(\bar{x}) := \frac{\partial f}{\partial x}(\bar{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\bar{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\bar{x}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\bar{x}) & \dots & \frac{\partial f_n}{\partial x_n}(\bar{x}) \end{pmatrix} = \begin{pmatrix} \nabla f_1 \\ \vdots \\ \nabla f_n \end{pmatrix} \quad \text{zu } f = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}, \tag{5.2.1}$$

gilt mit der Matrix $A := f'(\bar{x})$ die Aussage

$$f(x) = f(\bar{x}) + A(x - \bar{x}) + o(\|x - \bar{x}\|), \; x \to \bar{x}. \tag{5.2.2}$$

Falls der Abstand $\|z - \bar{x}\|$ zu einer Nullstelle z klein genug und A regulär ist, kann zur Approximation von z das *lineare Modell* benutzt werden:

$$f(z) = 0 \stackrel{!}{=} f(\bar{x}) + A(z - \bar{x}) + \dots \quad \Rightarrow \quad z \cong \bar{x} - A^{-1} f(\bar{x}).$$

Dies führt auf das folgende *Newtonverfahren* im $\mathbb{R}^n$, mit der Iteration

$$f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = -f(x^{(k)}), \quad k = 0, 1, \dots, \tag{5.2.3}$$

die mit einem Startvektor $x^{(0)} \in \mathbb{R}^n$ durchgeführt wird. Pro Schritt (5.2.3) ist dabei ein lineares Gleichungssystem zu lösen, mit dessen Lösung man die aktuelle Näherung korrigiert,

$$\begin{array}{rcll} A_k s^{(k)} & = & -f(x^{(k)}) & \text{mit Matrix } A_k = f'(x^{(k)}), \\ x^{(k+1)} & := & x^{(k)} + s^{(k)}. & \end{array} \tag{5.2.4}$$

Die Inverse A_k^{-1} wird dazu aber *nicht* berechnet (vgl. §4.3). Der Einzugsbereich der Iteration (5.2.3) sieht meist sehr kompliziert aus. Man kann aber eine Kugel um z konstruieren, in der Konvergenz $x^{(k)} \to z$ eintritt für jeden Startwert. Zuvor sei an den Mittelwertsatz im $\mathbb{R}^n$ erinnert:

Satz 5.2.1 *Es sei $D \subseteq \mathbb{R}^n$ offen, $f \in C^2(D)$ und $D_0 \subseteq D$ konvex und abgeschlossen. Dann gelten für $x, y \in D_0$ die Aussagen*

$$f(x) - f(y) = \int_0^1 f'\Big(y + t(x-y)\Big)\, dt \cdot (x-y), \tag{5.2.5}$$

$$\|f(x) - f(y)\| \le L_1 \|x-y\|, \quad L_1 := \max\{\|f'(x)\| : x \in D_0\}, \tag{5.2.6}$$

$$\Big(f'(x) - f'(y)\Big)v = \int_0^1 f''(y + t(x-y))\, dt\ [v, x-y] \quad \forall v \in \mathbb{R}^n, \tag{5.2.7}$$

$$\|f'(x) - f'(y)\| \le L_2 \|x-y\|, \quad L_2 := \max\{\|f''(x)\| : x \in D_0\}. \tag{5.2.8}$$

In (5.2.7) ist f'' eine bilineare Abbildung, $f''(x) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$. 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 $\varphi_i(t) := f_i\big(y + t(x-y)\big)$, $t \in [0,1]$, gilt nach der Kettenregel

$$\begin{array}{rcl} \dfrac{d\varphi_i}{dt}(t) & = & \mathrm{grad} f_i\big(y + t(x-y)\big) \cdot (x-y) \quad \Rightarrow \\ \varphi_i(1) - \varphi_i(0) & = & \displaystyle\int_0^1 \varphi_i'(t)\, dt = \int_0^1 \mathrm{grad} f_i\big(y + t(x-y)\big)\, dt \cdot (x-y), \end{array}$$

d.h., (5.2.5) nach (5.2.1). Die Schranke (5.2.6) ergibt sich daraus sofort, denn

$$\|f(x) - f(y)\| \le \| \int_0^1 f'\big(y + t(x-y)\big) dt \| \, \|x-y\| \le \max_{t\in[0,1]} \|f'\big(y + t(x-y)\big)\| \, \|x-y\|.$$

Der Beweis von (5.2.7),(5.2.8) geht analog. ∎

Das Newtonverfahren konvergiert lokal quadratisch:

Satz 5.2.2 *Es sei $D \subseteq \mathbb{R}^n$ 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*

$$\|f'(z)^{-1}\| \le \frac{1}{\lambda_1}, \tag{5.2.9}$$

$$\sup_{x \in D} \|f''(x)\| \le L_2. \tag{5.2.10}$$

Die Kugel $K_\delta := \{x \in \mathbb{R}^n : \ \|x - z\| \le \delta\}$ sei in D enthalten,

$$K_\delta \subseteq D \quad \textit{für} \quad \delta < \frac{2\lambda_1}{3L_2}.$$

Dann konvergiert das Newtonverfahren (5.2.3) für jeden Startvektor $x^{(0)} \in K_\delta$ gegen z. Die Konvergenz ist quadratisch,

$$\|x^{(k+1)} - z\| \le \frac{1}{2}\frac{L_2}{\lambda_1 - \delta L_2}\|x^{(k)} - z\|^2. \tag{5.2.11}$$

Beweis O.B.d.A. wird der erste Schritt mit $x^{(0)} =: x$ betrachtet. Es gilt nach (5.2.3) und (5.2.5)

$$\begin{aligned}\|x^{(1)} - z\| &= \| x - z - f'(x)^{-1}\big(f(x) - f(z)\big)\| \\ &= \|\Big(I - f'(x)^{-1}\int_0^1 f'\big(z + t(x-z)\big)dt\Big)(x-z)\| \\ &\le \underbrace{\|f'(x)^{-1}\|}_{=:q_1}\underbrace{\int_0^1 \|f'(x) - f'\big(z+t(x-z)\big)\|dt}_{=:q_2}\ \|x-z\|. \end{aligned}\tag{5.2.12}$$

Mit der Abkürzung $y = z + t(x-z)$, d.h., $x - y = (1-t)(x-z)$, läßt sich nach (5.2.8) in K_δ die Differenz der Ableitungen abschätzen durch

$$q_2 = \int_0^1 \|f'(x) - f'\big(z+t(x-z)\big)\|dt \le L_2\int_0^1 (1-t)dt\,\|x-z\| = \frac{1}{2}L_2\|x-z\|. \tag{5.2.13}$$

Als nächstes wird die Regularität der Ableitung $f'(x)$ gezeigt. Mit der Regularität von $f'(z)$ folgt $f'(z)^{-1}f'(x) = I + f'(z)^{-1}\big(f'(x) - f'(z)\big)$. Nach Voraussetzung (5.2.9) ist der zweite Summand klein,

$$\|f'(z)^{-1}\big(f'(x) - f'(z)\big)\| \le \frac{L_2}{\lambda_1}\|x-z\| \le \frac{L_2\delta}{\lambda_1} < \frac{2}{3} < 1.$$

Nach dem Satz von Neumann, Satz 4.2.2, existiert daher $f'(x)^{-1}$ und nach (4.2.11) ist

$$q_1 = \|f'(x)^{-1}\| \le \|f'(z)^{-1}\|\frac{1}{1 - L_2\delta/\lambda_1} \le \frac{1}{\lambda_1 - L_2\delta}.$$

Diese Schranke und (5.2.13) führen mit (5.2.12) auf die letzte Behauptung (5.2.11),

$$\|x^{(1)} - z\| \le q_1 q_2\|x-z\| \le \frac{1}{\lambda_1 - \delta L_2}\frac{1}{2}L_2\|x-z\|^2.$$

Damit ist die quadratische Konvergenz gesichert, wenn noch $\|x^{(1)} - z\| \le q_1q_2\|x-z\|,\ q_1q_2 \le q < 1$, gilt. Dies gilt tatschlich für $\delta < 2\lambda_1/(3L_2)$:

$$\|x^{(1)} - z\| \le \underbrace{\frac{1}{2}\frac{L_2\delta}{\lambda_1 - \delta L_2}}_{=:\,q<1}\|x-z\| < \frac{1}{3}\frac{\lambda_1}{\lambda_1 - 2\lambda_1/3}\|x-z\| = \|x-z\|. \qquad \blacksquare$$

Durchführung des Newtonverfahrens

1. Im $\mathbb{R}^n$ sind pro Schritt des Newtonverfahrens $n^2 + n$ Komponenten von f und f' zu berechnen und eine LR- oder QR-Zerlegung durchzuführen (Aufwand $O(n^3)$ Oper.). Wenn $x^{(0)}$ genügend nahe bei z liegt, kann evtl. die alte Ableitungsmatrix (und ihre Zerlegung!) übernommen werden. Dies führt zum *vereinfachten Newtonverfahren*

$$f'(x^{(0)})(x^{(k+1)} - x^{(k)}) = -f(x^{(k)}), \quad k = 0, 1, \ldots. \tag{5.2.14}$$

Hier ändert sich beim Gleichungssystemen jetzt nur die rechte Seite $-f(x^{(k)})$ und der Aufwand reduziert sich auf n Funktionskomponenten und $O(n^2)$ Operationen pro Schritt nach einmaliger Berechnung der LR-Zerlegung (vgl. §4.3). Die vereinfachte Iteration (5.2.14) konvergiert aber nur noch linear mit $\|I - f'(x^{(0)})^{-1} f'(z)\| =: q$ als asymptotischem Konvergenzfaktor. Wenn q genügend klein ist und keine extreme Endgenauigkeit gefordert wird, kann dies schneller zum Erfolg führen, da das volle Newtonverfahren n-mal so teuer ist pro Schritt.

2. Satz 5.2.2 garantiert nur *lokale Konvergenz* in einer (kleinen?) Kugel um die Nullstelle. Die globale Konvergenz der Iteration kann verbessert werden, indem man den Fortschritt anhand der Norm $\|f(x)\|_2$ überprüft. Dabei versucht man, die Funktion ("Zielfunktion")

$$\varphi(x) := \|f(x)\|_2^2 \qquad \text{zu } \textbf{minimieren}.$$

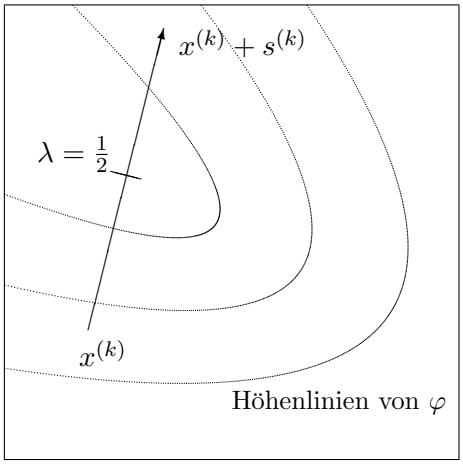
Der Vektor $s^{(k)} := -f'(x^{(k)})^{-1} f(x^{(k)})$ des Newtonschritts wird dazu nur als Such*richtung* verwendet, und der nächste Punkt $x^{(k+1)}$ so auf dem Strahl

$$x^{(k+1)} := x^{(k)} + \lambda_k s^{(k)}, \quad 0 < \lambda_k \leq 1,$$

bestimmt, dass $\varphi(x^{(k+1)}) < \varphi(x^{(k)})$ ist. Man führt somit eine *Liniensuche* durch. Da für die Richtungsableitung von ϕ gilt

$$\frac{d}{d\lambda} \varphi(x^{(k)} + \lambda s^{(k)})\Big|_{\lambda=0} = -2\varphi(x^{(k)}) < 0,$$

existiert ein solcher Punkt für genügend kleines λ.



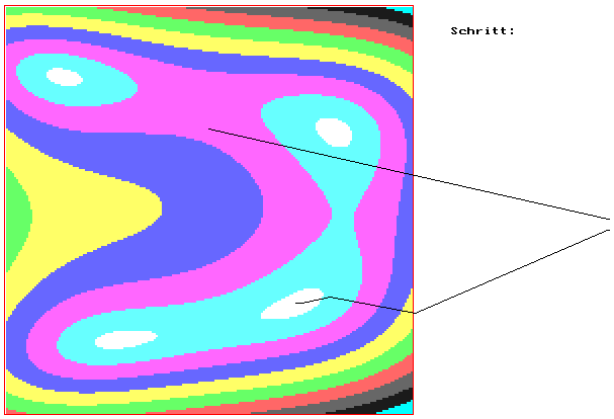
In der Praxis halbiert man dazu die Strecke $x^{(k)} \ldots x^{(k)} + s^{(k)}$ solange, bis φ einen kleineren Wert annimmt. Diese Wahl führt auf $\lambda_k \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$. Da dann $\lambda_k \leq 1$ gilt, spricht man vom *gedämpften Newtonverfahren*. Nahe bei der Nullstelle wählt diese Methode automatisch wieder $\lambda_k = 1$, sodass die quadratische Konvergenz nicht zerstört wird.

Demo-Beispiel Im Quadrat $[-1, 1] \times [-1, 1] \subseteq \mathbb{R}^2$ besitzt die Polynom-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 $(-0.44, -0.64)$. Bei der ungedämpften Newton-Iteration mit dem Startwert $(0, 0.4)$ führt der erste Schritt weit außerhalb des Quadrats mit einer starken Vergrößerung des Wertes φ. Bei einer schwierigeren Funktion könnte dies fatal für die Konvergenz sein. Hier haben die Iterierten (12-stellige Rechnung) folgende Werte

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



Ab Schritt 4 konvergiert die Iteration offensichtlich quadratisch gegen eine Lösung in der unteren Halbebene. Bei einer Liniensuche im ersten Schritt wird dagegen die nähergelegene Nullstelle bei $(0.62, 0.39)$ gefunden.

Index