Ludwigs-Maximilians-Universität München

— Sektion Physik —

Joachim Puls · Sigmund Stintzing

Numerik für Physiker

zweite, überarbeitete Ausgabe (2004)

Sebastian Bauer, Stephan Maciej, Benjamin Rieff

Wintersemester 2002/2003

Inhaltsverzeichnis

Kapitel 1

Nichtlineare Gleichungen

Als Teilaufgabe vieler numerischer Probleme müssen nichtlineare Gleichungen gelöst werden. Hierbei gibt es keine direkten Verfahren, wie bei linearen Gleichungssystemen, die nach endlich vielen Schritten eine Lösung liefern. Deshalb ist man auf iterative Verfahren angewiesen, die eine Lösung als Grenzwert einer Folge von Näherungslösungen liefern. Beispiele für nichtlineare Gleichungen sind Polynome, deren Grad höher als zwei ist, wie

$$x^5 - 5x^3 + 4x + 1 = 0$$

oder auch transzendente Gleichungen, wie

$$x = \tanh(2x).$$

Wir betrachten das Problem, eine Lösung $\boldsymbol{x} \in \mathbb{R}^n$ der Gleichung

$$F(\boldsymbol{x}) = \boldsymbol{y} \tag{1.1}$$

mit einer nichtlinearen Funktion $F : \mathbb{R}^n \supset \mathcal{G} \to \mathbb{R}^n$ und $\boldsymbol{y} \in \mathbb{R}^n$ zu finden. Dabei können die folgenden Äquivalenzumformungen hilfreich sein

$$F(\boldsymbol{x}) = \boldsymbol{y} \underset{f(\boldsymbol{x}) := F(\boldsymbol{x}) - \boldsymbol{y}}{\Longleftrightarrow} f(\boldsymbol{x}) = 0 \underset{g(\boldsymbol{x}) := f(\boldsymbol{x}) + \boldsymbol{x}}{\Longleftrightarrow} \boldsymbol{x} = g(\boldsymbol{x}). \tag{1.2}$$

Es wird hier nacheinander das Problem in ein Nullstellenproblem und dann in ein Fixpunktproblem übergeführt. Welche spezielle Form einen guten Ansatz für einen Algorithmus bietet, hängt vom konkreten Problem ab.

1.1 Beispiel (Geeignete Fixpunktform). Gesucht ist eine Lösung $x_* > 0$ der Gleichung $x^2 = 2$. Die Gleichung kann durch Äquivalenzumformungen in die Form $x = \frac{1}{x} + \frac{x}{2} =: g(x)$ gebracht werden. Mit der Iterationsvorschrift $x_{i+1} = g(x_i)$, $i \in \mathbb{N}_0$ erhalten wir z.B. bei 5 startend die Werte

x_0	x_1	x_2	x_3	x_4	$\cdots$
5.0	2.7	1.72	1.441	1.415	$\cdots$

Man sieht, dass die Folge gegen den tatsächlichen Wert von $x_* = \sqrt{2} = 1.414213\ldots$ konvergiert. Eine alternative Umformung ergibt $x = \frac{2}{x} =: g(x)$. Dieselbe Iterationsvorschrift $x_{i+1} = g(x_i)$ für $i \in \mathbb{N}_0$ liefert hier mit dem Startwert 5 allerdings

x_0	x_1	x_2	x_3	x_4	$\cdots$
5.0	0.4	5.0	0.4	5.0	$\cdots$

Es ergibt sich eine alternierende Reihe, die Folge konvergiert also nicht. Die Form $x = \frac{2}{x}$ ist damit ungeeignet zur Lösung dieses Problems.

1.1 Fixpunktiteration

Sei $g : \mathbb{R}^n \supset \mathcal{G} \to \mathcal{G}$. Eine Lösung $\boldsymbol{x}_* \in \mathcal{G}$ der Gleichung

$$\boldsymbol{x} = g(\boldsymbol{x}) \tag{1.3}$$

heißt *Fixpunkt*, d.h. $\boldsymbol{x}_* = g(\boldsymbol{x}_*)$. Um solche Gleichungen zu lösen, verwendet man die Fixpunktiteration (Methode der sukzessiven Approximation)

$$\boldsymbol{x}_{i+1} = g(\boldsymbol{x}_i) \qquad (i = 0, 1, \dots) \tag{1.4}$$

mit einem Startwert $\boldsymbol{x}_0 \in \mathcal{G}$.

Sei $\mathcal{G} \subset \mathbb{R}^n$. g heißt LIPSCHITZ-stetig mit der LIPSCHITZ-Konstante $L \geq 0$, wenn

$$\|g(\boldsymbol{x}) - g(\boldsymbol{x}')\| \leq L\|\boldsymbol{x} - \boldsymbol{x}'\| \qquad (\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{G}) \tag{1.5}$$

gilt. Ist sogar $L < 1$, so heißt g *kontrahierend*. Die Kontraktionseigenschaft hängt von der verwendeten Vektornorm ab. Offensichtlich ist die LIPSCHITZ-Stetigkeit von g hinreichend für die Stetigkeit.

Kriterium für LIPSCHITZ-Stetigkeit. Sei $g \in \mathcal{C}^1(\mathcal{G})$ mit $\mathcal{G} \subset \mathbb{R}^n$ kompakt und konvex. Dann ist g LIPSCHITZ-stetig mit der LIPSCHITZ-Konstanten $L = \max_{x\in\mathcal{G}} \|g'(\boldsymbol{x})\|$, wobei $g'(\boldsymbol{x}) := \left(\frac{\partial g_i}{\partial x_j}(\boldsymbol{x})\right)_{1\leq i,j\leq n}$ die Funktionalmatrix und die Matrixnorm die von der Vektornorm induzierte Matrixnorm ist. Ist zusätzlich $\max_{x\in\mathcal{G}} \|g'(\boldsymbol{x})\| < 1$, so ist g kontrahiernd. Speziell gilt für $n = 1$

$$\max_{x\in\mathcal{G}} |g'(x)| < 1.$$

$\mathcal{G}$ ist hier ein abgeschlossenes Intervall.
Die LIPSCHITZ-Stetigkeit folgt unmittelbar aus dem Mittelwertsatz der Differentialrechnung.

1.2 Satz (BANACH'scher Fixpunktsatz). *Sei $\mathcal{G} \subset \mathbb{R}^n$ abgeschlossen und $g : \mathcal{G} \to \mathcal{G}$ kontrahierend. Dann besitzt g genau einen Fixpunkt $\boldsymbol{x}_* \in \mathcal{G}$. Für jeden Startwert $\boldsymbol{x}_0 \in \mathcal{G}$ konvergiert die durch $\boldsymbol{x}_{i+1} := g(\boldsymbol{x}_i)$, $(i = 0, 1, \dots)$ definierte Folge gegen den Fixpunkt $\boldsymbol{x}_*$. Es gilt die Abschätzung*

$$\|\boldsymbol{x}_* - \boldsymbol{x}_i\| \leq \frac{L^{i-j}}{1-L}\|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\| \qquad (0 \leq j \leq i). \tag{1.6}$$

1.3 Beweis. *Die LIPSCHITZ-Stetigkeit von g führt zu*

$$\begin{aligned} \|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\| &= \|g(\boldsymbol{x}_i) - g(\boldsymbol{x}_{i-1})\| \leq L\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| \\ &= L\|g(\boldsymbol{x}_{i-1}) - g(\boldsymbol{x}_{i-2})\| \leq L^2\|\boldsymbol{x}_{i-1} - \boldsymbol{x}_{i-2}\| = \ldots \leq L^i\|\boldsymbol{x}_1 - \boldsymbol{x}_0\| \end{aligned}$$

für $i \geq 0$. Damit folgt für $j, k \geq 0$

$$\begin{aligned} \|\boldsymbol{x}_{j+k} - \boldsymbol{x}_j\| &= \|\boldsymbol{x}_{j+k} - \boldsymbol{x}_{j+k-1} + \boldsymbol{x}_{j+k-1} - \ldots + \boldsymbol{x}_{j+1} - \boldsymbol{x}_j\| \\ &\overset{\Delta-Ungl.}{\leq} \sum_{i=j}^{j+k-1} \|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\| \leq \left(\sum_{i=j}^{j+k-1} L^i\right) \|\boldsymbol{x}_1 - \boldsymbol{x}_0\| \\ &= L^j \frac{1-L^k}{1-L}\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|. \end{aligned}$$

Dabei ist $L < 1$ und daher $(\boldsymbol{x}_i)_{i\geq 0}$ eine CHAUCHY-Folge. Aufgrund der Vollständigkeit von $\mathbb{R}^n$ mit $\|\cdot\|$ konvergiert die Folge gegen ein $\boldsymbol{x}_* \in \mathcal{G}$. Wegen der Stetigkeit von g gilt $g(\boldsymbol{x}_*) = g(\lim_{i\to\infty} \boldsymbol{x}_i) = \lim_{i\to\infty} g(\boldsymbol{x}_i) = \lim_{i\to\infty} \boldsymbol{x}_{i+1} = \boldsymbol{x}_*$, d.h. $\boldsymbol{x}_*$ ist ein Fixpunkt. Die Existenz eines weiteren Fixpunktes $\tilde{\boldsymbol{x}_*} \neq \boldsymbol{x}_*$ wäre ein Widerspruch zu

$$\|\boldsymbol{x}_* - \tilde{\boldsymbol{x}_*}\| = \|g(\boldsymbol{x}_*) - g(\tilde{\boldsymbol{x}_*})\| \leq L\|\boldsymbol{x}_* - \tilde{\boldsymbol{x}_*}\|$$

mit $L < 1$. Wir haben damit auch gezeigt, dass für jedes $\boldsymbol{x}_0 \in \mathcal{G}$ $(\boldsymbol{x}_i)_{i\geq 0}$ gegen $\boldsymbol{x}_*$ konvergiert. □

Die Voraussetzungen des Satzes sind oft nur für eine "kleine" Menge $\mathcal{G}$ erfüllt (nur lokale Aussagen). Die Angabe dieser Menge ist häufig schwierig.

1.4 Beispiel (Nichtlineare Gleichung $x = e^{-x} =: g(x)$, $x \in \mathbb{R}$). Um den Fixpunktsatz anwenden zu können, benötigen wir ein abgeschlossenes Intervall, für welches die Voraussetzungen zutreffen. Dies ist zum Beispiel für $\mathcal{G} := [0.5, 0.69]$ erfüllt: $g : \mathcal{G} \to \mathcal{G}$, LIPSCHITZ-stetig mit $L = \max_{x\in\mathcal{G}} |-e^{-x}| = e^{-0.5} < 1$.

Für die Praxis sind die Fehlerabschätzungen

$$j = 0 \text{ in (1.6)}: \|\boldsymbol{x}_* - \boldsymbol{x}_i\| \leq \frac{L^i}{1-L}\|\boldsymbol{x}_1 - \boldsymbol{x}_0\| \quad (i > 0), \qquad \text{a priori,} \tag{1.7a}$$

$$j = i-1 \text{ in (1.6)}: \|\boldsymbol{x}_* - \boldsymbol{x}_i\| \leq \frac{L}{1-L}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| \quad (i > 0), \qquad \text{a posteriori (genauer)} \tag{1.7b}$$

nützlich. Offenbar ist die Abschätzung 1.7b im Allgemeinen genauer als 1.7a.
Sei $\varepsilon > 0$ eine vorgegebene Fehlerschranke. Die Iteration wird im i-ten Schritt abgebrochen, falls erstmalig $\frac{L}{1-L}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| \leq \varepsilon$ gilt. Nach (1.7b) ist dann $\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| \leq \varepsilon$. Nach (1.7a) ist die maximal notwendige Anzahl von Iterationschritten

$$\leq \left[\frac{\ln \frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|}{(1-L)\varepsilon}}{\ln \frac{1}{L}}\right] + 1 \,.$$

Dabei bezeichnet $[\ldots]$ den ganzzahligen Anteil.

1.5 Beispiel (Fortsetzung von Beispiel1.4).

i	x_i	i	x_i	i	x_i
		$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	0.55000000	10	0.56708394	20	0.56714309
1	0.57694981	11	0.56717695	21	0.56714340
2	0.56160877	12	0.56712420	22	0.56714323
3	0.57029086	13	0.56715412	23	0.56714332
4	0.56536097	14	0.56713715	24	0.56714327
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Im 12. Schritt ($i = 12$) ist nach (1.7b)[1] $\|x_* - x_{12}\| \leq 8.13 \cdot 10^{-5}$, tatsächlich ist $x_* = 0.56714329\ldots$ und damit $\|x_* - x_{12}\| = 1,91 \cdot 10^{-5}$. Um $\varepsilon = 10^{-6}$ zu erreichen sind maximal $\left[(\ln \frac{|x_1 - x_0|}{(1-L)\varepsilon})/\ln \frac{1}{L}\right] + 1 = 23$ Iterationsschritte nötig (Überschätzung).

Konvergenzverhalten. Sei $g : \mathbb{R}^n \supset \mathcal{G} \to \mathcal{G}$ eine Iterationsfunktion mit einem Fixpunkt $\boldsymbol{x}_* \in \mathcal{G}$. Für jeden Startwert $\boldsymbol{x}_0$ aus einer Umgebung $\mathcal{U}$ von $\boldsymbol{x}_*$ konvergiere die durch

$$\boldsymbol{x}_{i+1} := g(\boldsymbol{x}_i) \qquad (i = 0, 1, \ldots)$$

definierte Iterationfolge $(\boldsymbol{x}_i)_{i\geq 0}$ gegen den Fixpunkt $\boldsymbol{x}_*$. Dann heißt das Iterationsverfahren zur Bestimmung des Fixpunktes *lokal konvergent* (anziehender Fixpunkt). Ist $\mathcal{U} = \mathcal{G}$, so spricht man von *globaler Konvergenz*.
Für den praktischen Einsatz eines Iterationsverfahrens ist die Geschwindigkeit der Konvergenz wichtig:

> Für alle Startwerte $\boldsymbol{x}_0$ aus einer Umgebung $\mathcal{U} \subset \mathcal{G}$ erfülle die durch $\boldsymbol{x}_{i+1} := g(\boldsymbol{x}_i)$, $(i = 0, 1, \ldots)$ definierte Iterationsfolge $(\boldsymbol{x}_i)_{i\geq 0}$ die Ungleichung
>
> $$\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_*\| \leq C\|\boldsymbol{x}_i - \boldsymbol{x}_*\|^p \quad (i = 0, 1, \ldots) \tag{1.8}$$
>
> mit $C > 0$ und $p \geq 1$, wobei im Fall $p = 1$ noch $C < 1$ sei. Dann heißt das Iterationsverfahren zur Bestimmung des Fixpunktes *lokal konvergent von mindestens p-ter Ordnung*.

[1] nach (1.7a) $\|x_* - x_{12}\| \leq 1.70 \cdot 10^{-4}$

Bemerkungen.

(i) Ein Verfahren von mindestens p-ter Ordnung mit $p > 1$ ist auch von mindestens q-ter Ordnung für alle $1 \leq q \leq p$.

(ii) Ein Verfahren von mindestens p-ter Ordnung ist lokal konvergent (im Sinne der Anziehungseigenschaft).

(iii) C und wesentlich stärker p bestimmen die Konvergenzgeschwindigkeit einer Iterationsfolge. Je kleiner C und vor allem je größer p ist, desto schneller ist die Konvergenz.

Kriterium im eindimensionalen Fall ($n = 1$).

1.6 Satz. *Sei $g : \mathbb{R} \supset \mathcal{G} \to \mathcal{G}$ eine Iterationsfunktion mit einem Fixpunkt $x_* \in \mathcal{G}$ und p-mal stetig differenzierbar in einer Umgebung $\mathcal{U} \subset \mathcal{G}$ von x_*. Gilt dann*

$$0 < |g'(x_*)|, \text{ falls } p = 1 \tag{1.9a}$$

beziehungsweise

$$g^{(q)}(x_*) = 0 \text{ für } q = 1, 2, \ldots, p-1 \text{ und } g^{(p)}(x_*) = 0, \text{ falls } p = 2, 3, \ldots, \tag{1.9b}$$

so ist das Iterationsverfahren lokal konvergent von genau p-ter Ordnung.

1.7 Beweis. *Über die* TAYLOR*-Entwicklung*

$$g(x) = g(x_*) + \sum_p \frac{g^{(p)}(x_*)}{p!}(x - x_*)^p + \emptyset(|x - x_*|^{p+1})$$

sieht man

$$\frac{g(x) - x_*}{(x - x_*)^p} \to \frac{g^{(p)}(x_*)}{p!} \text{ für } x \to x_*.$$

Daraus folgt die Behauptung. □

Exkurs: Vektor- und Matrizennormen. Es sei eine Vektornorm $\|\boldsymbol{x}\|$ für $\boldsymbol{x} \in \mathbb{R}^n$ ($\mathbb{C}^n$) gegeben. Dann ist $\|A\| := \sup_{\boldsymbol{x} \neq 0} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$ für $A \in \mathbb{R}^{n,n}$ $(\mathbb{C})^{n,n}$ die induzierte Matrixnorm dieser Vektornorm. Die induzierte Matrixnorm ist mit der Vektornorm verträglich, das heißt $\|A\boldsymbol{x}\| \leq \|A\| \cdot \|\boldsymbol{x}\|$ $\forall \boldsymbol{x}, A$.

1.8 Beispiel. Einige häufig verwendete Vektornormen und ihre induzierten Matrixnormen:

(a) *Maximale Spaltensumme*

$$\|\boldsymbol{x}\|_1 := \sum_i |x_i| \qquad \to \qquad \|A\|_1 := \max_j \sum_i |A_{ij}|$$

(b) *Spektralnorm*

$$\|\boldsymbol{x}\|_2 := \left(\sum_i |x_i|^2 \right)^{\frac{1}{2}} \qquad \to \qquad \|A\|_2 = (\rho(A^\intercal A))^{\frac{1}{2}}$$

mit dem Spektralradius $\rho(A) := max_i|\lambda_i|$, wobei λ_i die Eigenwerte von A sind. Dies ist sehr aufwändig zu berechnen, daher wird häufig die FROBENIUS-Norm

$$\|A\|_F := \left(\sum_{i,j} |A_{ij}|^2 \right)^{\frac{1}{2}}$$

verwendet. Diese ist verträglich mit $\|\boldsymbol{x}\|_2$, $\|A\|_F \geq \|A\|_2$.

(c) *maximale Zeilensumme*

$$\|\boldsymbol{x}\|_\infty := \max_i |x_i| \qquad \to \qquad \|A\|_\infty := \max_i \sum_j |A_{ij}|$$

Eine Vektornorm $\|\cdot\|_a$ heißt äquivalent zu einer Vektornorm $\|\cdot\|_b$, wenn Zahlen m, $M > 0$ existieren mit $m\|\boldsymbol{x}\|_a \leq \|\boldsymbol{x}\|_b \leq M\|\boldsymbol{x}\|_a$ für alle $\boldsymbol{x}$. In einem endlichdimensionalen Vektorraum sind alle Normen äquivalent.

1.2 Bestimmung von Nullstellen

Sei $f : [a, b] \to \mathbb{R}$ stetig. Wir suchen eine Nullstelle $x_* \in [a, b]$ von f, d.h. eine Lösung der Gleichung

$$f(x) = 0. \tag{1.10}$$

1.2.1 Bisektionsverfahren

Beim Bisektionsverfahren wird eine Nullstelle sukzessive durch immer kleiner werdende Intervalle eingeschlossen (Intervallschachtelung). Wir gehen davon aus, dass ein Intervall $[a_0, b_0] \subset [a, b]$ bekannt ist, mit $f(a_0)f(b_0) < 0$. Dann existiert nach dem Satz von ROLLE ein $x_* \in (a_0, b_0)$ mit $f(x_*) = 0$, dies wird in Abbildung 1.1 deutlich. Es können auch mehrere Nullstellen zwischen a_0 und b_0 liegen.

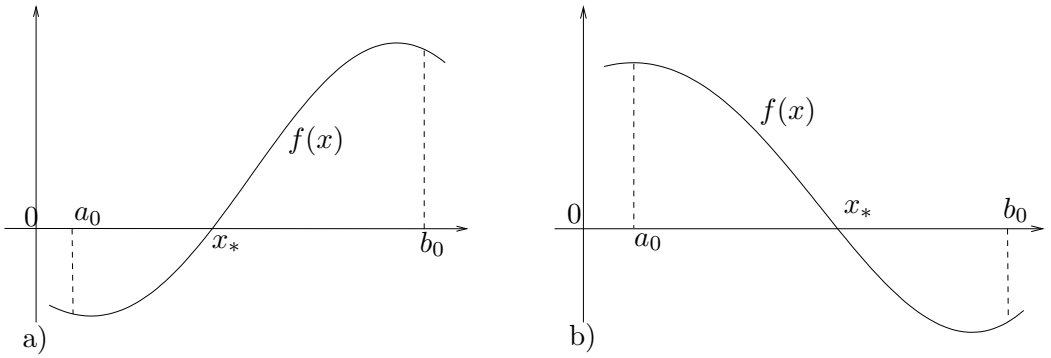


Abbildung 1.1: Satz von ROLLE

Für den Mittelpunkt $x_0 := \frac{1}{2}(a_0 + b_0)$ wird der Funktionswert berechnet. Ist $f(x_0) \neq 0$, so entscheidet das Vorzeichen von $f(x_0)$, in welchem der beiden Teilintervalle $[a_0, x_0]$ oder $[x_0, b_0]$ die gesuchte Nullstelle liegt. Ihre Lage ist damit auf ein gegenüber dem ursprünglichen Intervall halb so langen Intervall eingeschränkt. Auf diese Weise kann man, wie in Abbildung 1.2 gezeigt fortfahren, bis die gesuchte Lösung in einem hinreichend kleinen Intervall liegt.

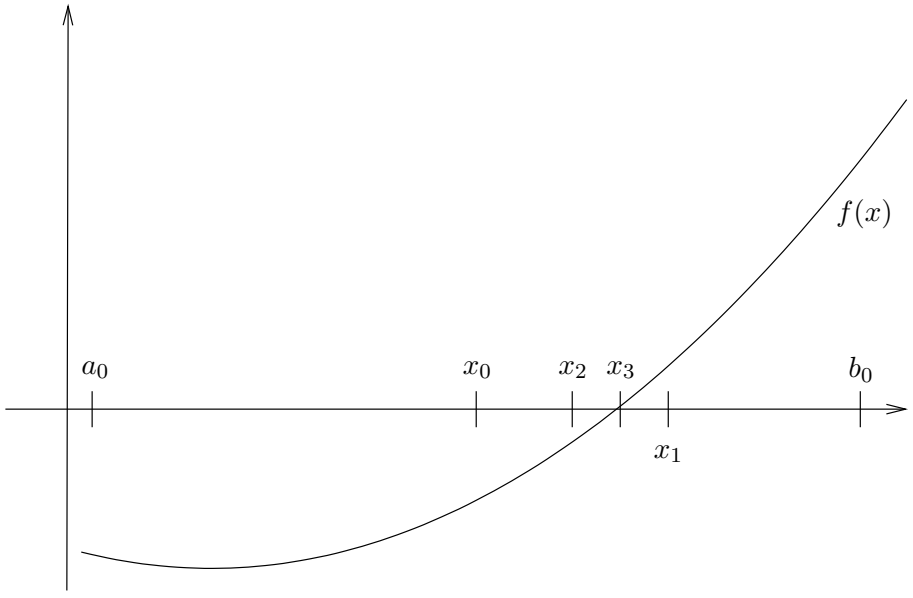


Abbildung 1.2: Bisektionsverfahren

Der Mittelpunkt x_i des Intervalls nach i Intervallhalbierungen ist eine Näherung für die Nullstelle x_* mit der Fehlerabschätzung

$$|x_i - x_*| \le \frac{b_0 - a_0}{2^{i+1}} \qquad (i = 0, 1, \dots). \tag{1.11}$$

Offensichtlich ist das Verfahren genau linear konvergent mit dem Konvergenzfaktor 1. Das Bisektionsverfahren liefert mit Sicherheit eine Nullstelle samt unterer und oberer Schranke.

Algorithmus:

(i) Suche Startintervall $[a_0, b_0]$ mit $f(a_0)f(b_0) < 0$ und setze $i = 0$.

(ii) Berechne Intervallmittelpunkt $x_i = \frac{1}{2}(a_i + b_i)$ und $f(x_i)$.

(iii) Stoppe, falls $f(x_i) = 0$.
Ist $f(a_i)f(x_i) < 0$, setze $a_{i+1} = a_i$, $b_{i+1} = x_i$.
Sonst setze $a_{i+1} = x_i$, $b_{i+1} = b_i$.

(iv) Ist $|b_{i+1} - a_{i+1}| <$ EPS, gehe zu v
Sonst erhöhe i um 1 und gehe zu ii.

(v) Setze $x_* = \frac{1}{2}(a_{i+1} + b_{i+1})$.

1.9 Beispiel. Mit Hilfe des Bisektionsverfahren wird die Nullstelle von

$$f(x) = x \tan x$$

gesucht. Als Startwerte werden $a_0 = 2$ und $b_0 = 4.6$ mit $f(a_0) = 4.18$ und $f(b_0) = -4.26$ verwendet.

i	a_i	b_i	x_i	$f(x_i)$
0	2	4.6	3.3	$+3.14 \cdot 10^{0}$
1	3.3	4.6	3.95	$+2.90 \cdot 10^{0}$
2	3.95	4.6	4.275	$+2.13 \cdot 10^{0}$
3	4.275	4.6	4.4375	$+8.91 \cdot 10^{-1}$
4	4.4375	4.6	4.51875	$-5.80 \cdot 10^{-1}$
5	4.4375	4.51875	4.47812	$+2.87 \cdot 10^{-1}$
10	4.493359	4.49589	4.49462	$-2.47 \cdot 10^{-2}$
20	4.493408	4.493411	4.493410	$-1.51 \cdot 10^{-5}$

1.2.2 Sekantenverfahren

Im Gegensatz zum Bisektionsverfahren wird beim Sekantenverfahren darauf verzichtet, die Nullstelle stets durch zwei Werte einzuschließen. Zu zwei Startwerten x_0 und x_1 mit $f(x_0) \neq f(x_1)$, welche die Nullstelle nicht einzuschließen brauchen, bestimmt man die Abszisse x_2 des Schnittpunktes der Sekante durch die Punkte $(x_0, f(x_0))$ und $(x_1, f(x_1))$ mit der x-Achse. Ist $f(x_1) \neq f(x_2)$, so wird mit der Sekante durch $(x_1, f(x_1))$ und $(x_2, f(x_2))$ der nächste Näherungswert x_3 als Nullstelle der Sekante berechnet, usw. (siehe Abbildung 1.3) Die Nullstelle der Sekante im i-ten Iterationschritt ergibt sich aus der Sekantengleichung

$$y(x) = f(x_i) + \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}(x - x_i)\,, \tag{1.12}$$

Mit $y(x_{i+1}) = 0$ folgt daraus

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \qquad (i = 1, 2, \dots). \tag{1.13}$$

Dabei wird $f(x_i) \neq f(x_{i-1})$ vorausgesetzt.

Die Berechnung des Wertes x_{i+1} erfolgt nicht nur aus x_i, sondern auch aus x_{i-1} (zweistufiges Verfahren). Daher können wir hier die Kriterien aus dem Abschnitt über das Konvergenzverhalten von einstufigen Iterationsverfahren nicht anwenden.

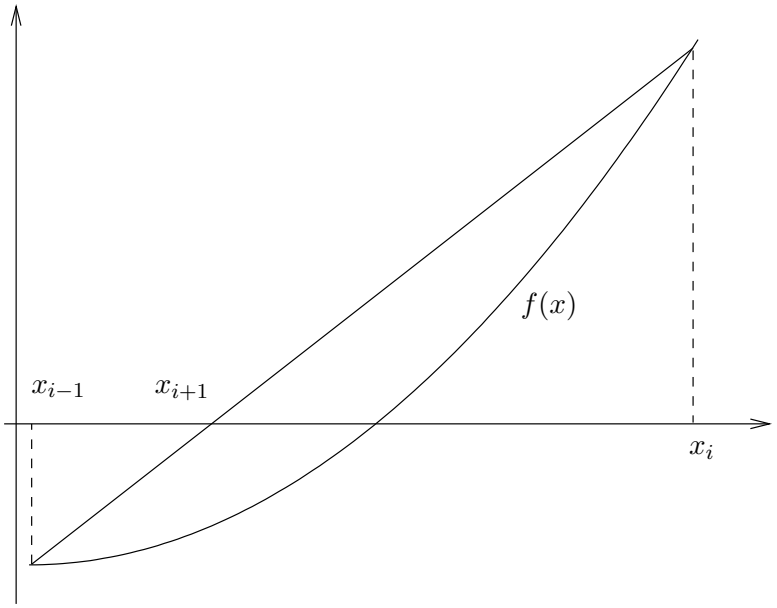


Abbildung 1.3: Sekantenverfahren

Konvergenzverhalten. Ist $f \in \mathcal{C}^{(2)}([a,b])$, so liefert eine Taylorentwicklung um x_* für den Fehler $\varepsilon_i := x_i - x_*$

$$|\varepsilon_{i+1}| \approx \left|\frac{f''(x_*)}{2f'(x_*)}\right| |\varepsilon_i||\varepsilon_{i-1}| \tag{1.14}$$

für kleine Fehler. Der Ansatz

$$|\varepsilon_i| = C|\varepsilon_{i-1}|^p \tag{1.15}$$

in (1.14) ergibt

$$C|\varepsilon_i|^p \approx \left|\frac{f''(x_*)}{2f'(x_*)}\right| C|\varepsilon_{i-1}|^p|\varepsilon_{i-1}|,$$

mithin

$$|\varepsilon_i| \approx \left|\frac{f''(x_*)}{2f'(x_*)}\right|^{\frac{1}{p}} |\varepsilon_{i-1}|^{\frac{p+1}{p}}. \tag{1.16}$$

Ein Vergleich mit (1.15) führt zu $p+1 = p^2$, also der nicht ganzzahligen genauen Konvergenzordnung

$$p = \frac{1}{2}(1+\sqrt{5}) = 1.618\ldots \qquad > 1. \tag{1.17}$$

Auch wenn $f(x_0)f(x_1) < 0$ gilt, muss das Verfahren nicht gegen eine Nullstelle zwischen x_0 und x_1 konvergieren.

1.10 Beispiel (Fortsetzung von Beispiel 1.9). Wie in Beispiel 1.9 wird die Nullstelle von

$$f(x) = x - \tan x$$

gesucht, diesmal allerdings mit Hilfe des Sekantenverfahrens

i	x_i	x_{i-1}	$f(x_i)$
1	4.0	4.6	$-4.26 \cdot 10^{0}$
2	4.6	4.24	$+2.28 \cdot 10^{0}$
3	4.24	4.3656	$+1.59 \cdot 10^{0}$
4	4.3656	4.6587	$-1.39 \cdot 10^{1}$
5	4.6587	4.3957	$+1.34 \cdot 10^{0}$
6	4.3957	4.4187	$+1.11 \cdot 10^{0}$
7	4.4187	4.5288	$-8.58 \cdot 10^{-1}$
8	4.5288	4.4808	$+2.38 \cdot 10^{-1}$
9	4.4808	4.491323	$+4.17 \cdot 10^{-2}$
10	4.491323	4.4935329	$-2.49 \cdot 10^{-3}$
11	4.4935329	4.49340824	$+2.45 \cdot 10^{-5}$
12	4.49340824	4.49340945	$+1.07 \cdot 10^{-8}$

1.2.3 Verfahren von NEWTON

Sei f zusätzlich stetig differenzierbar. Beim NEWTON-Verfahren wird die Funktion im Startwert x_0 mit $f'(x_0) \neq 0$ linearisiert und die Abszisse des Schnittpunktes der Tangente durch den Punkt $(x_0, f(x_0))$ mit der x-Achse bestimmt. Ist $f'(x_1) \neq 0$, so liefert der Abszissenschnittpunkt der Tangente durch $(x_1, f(x_1))$ den nächsten Näherungswert für die Nullstelle x_* von $f(x)$, usw. Aus der Tangentengleichung

$$y(x) = f(x_i) + (x - x_i) f'(x_i) \tag{1.18}$$

ergibt sich mit $y(x_{i+1}) = 0$ die Iterationsvorschrift

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (i = 0, 1, \dots). \tag{1.19}$$

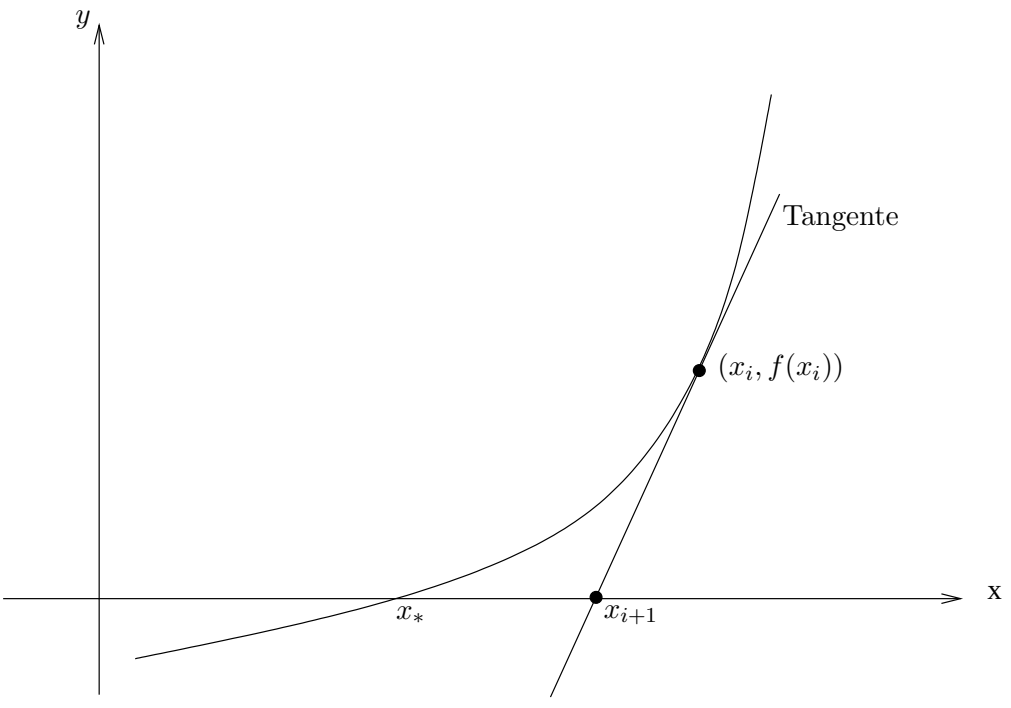


Abbildung 1.4: Verfahren von NEWTON

Im Vergleich zum Sekantenverfahren wird beim NEWTON-Verfahren die Sekante durch die Tangente ersetzt. Der Rechenaufwand beträgt zwei Funktionsauswertungen pro Iterationsschritt. Dies ist doppelt soviel wie beim Sekantenverfahren. Zudem ist die Berechnung der ersten Ableitung aufwendig, wenn sie nicht explizit bekannt ist. Das NEWTON-Verfahren besitzt aber bei einfachen Nullstellen eine höhere Konvergenzordnung als das Sekantenverfahren.

Konvergenzverhalten. Für $f \in \mathcal{C}^3([a,b])$ erhalten wir für die erste und zweite Ableitung der Iterationsfunktion

$$g(x) := x - \frac{f(x)}{f'(x)} : \tag{1.20}$$

$$g'(x) = \frac{f(x)f''(x)}{f'(x)^2}, \tag{1.21}$$

$$g''(x) = \frac{f'(x)^2 f''(x) + f(x)f'(x)f'''(x) - 2f(x)f''(x)^2}{f'(x)^3} \tag{1.22}$$

$$\tag{1.23}$$

Ist $f'(x_*) \neq 0$ (einfache Nullstelle) und $f''(x_*) \neq 0$, so gilt $g'(x_*) = 0$ und $g''(x_*) = \frac{f''(x_*)}{f'(x_*)} \neq 0$. Nach Satz 1.6 ist damit das Verfahren lokal konvergent von genau *zweiter Ordnung*. Ist sogar $f''(x_*) = 0$, so ist die Konvergenzordnung noch größer. Ist hingegen $f'(x_*) = 0$ (mehrfach Nullstelle), so ist das Verfahren nur linear lokal konvergent. Denn aus der Darstellung

$$f(x) = (x - x_*)^m \tilde{f}(x)$$

mit $m \geq 2$, $\tilde{f}(x_*) \neq 0$ und damit

$$g(x) = x - \frac{(x - x_*)\tilde{f}(x)}{m\tilde{f}(x) + (x - x_*)\tilde{f}'(x)}$$

folgt $g'(x_*) = 1 - \frac{1}{m}$, also

$$0 < g'(x_*) < 1.$$

1.11 Beispiel (Fortsetzung von Beipsiel 1.9). Die Nullstellen der Funktion

$$f(x) = x - \tan x$$

werden mit dem NEWTON-Verfahren berechnet ($x_* = 4.49340945$).

i	x_i	$f(x_i)$	$f'(x_i)$
0	4.5	-1.37 $\cdot 10^{-1}$	$-2.15 \cdot 10^{1}$
1	4.4936	-4.13 $\cdot 10^{-3}$	$-2.02 \cdot 10^{1}$
2	4.49340965	-3.97 $\cdot 10^{-6}$	$-2.01 \cdot 10^{1}$
3	4.49340945	+1.07 $\cdot 10^{-8}$	$-2.01 \cdot 10^{1}$

Das NEWTON-Verfahren konvergiert gegen eine Nullstelle im Allgemeinen nur dann, wenn der Startwert nahe der Nullstelle liegt (lokale Konvergenz). Dieser Bereich kann in manchen Fällen sehr klein sein. Das NEWTON-Verfahren ist nur in Ausnahmefällen global konvergent. Es kann sogar, wie Abbildung 1.5 zeigt, divergent sein.

Bemerkung: Gedämpftes NEWTON-Verfahren.

$$x_{i+1} = x_i - \frac{1}{2^{m_i}} \frac{f(x_i)}{f'(x_i)}$$

mit

$$m_i := \min \left\{ 0 \leq m \leq m_{\max} \middle| |f(x_i - \frac{1}{2^m} \frac{f(x_i)}{f'(x_i)})| < |f(x_i)| \right\}$$

oder $m_i := 0$, falls obige Ungleichung nicht erfüllbar ist. Mit dieser Methode erreichen wir eine Globalisierung des NEWTON-Verfahrens.

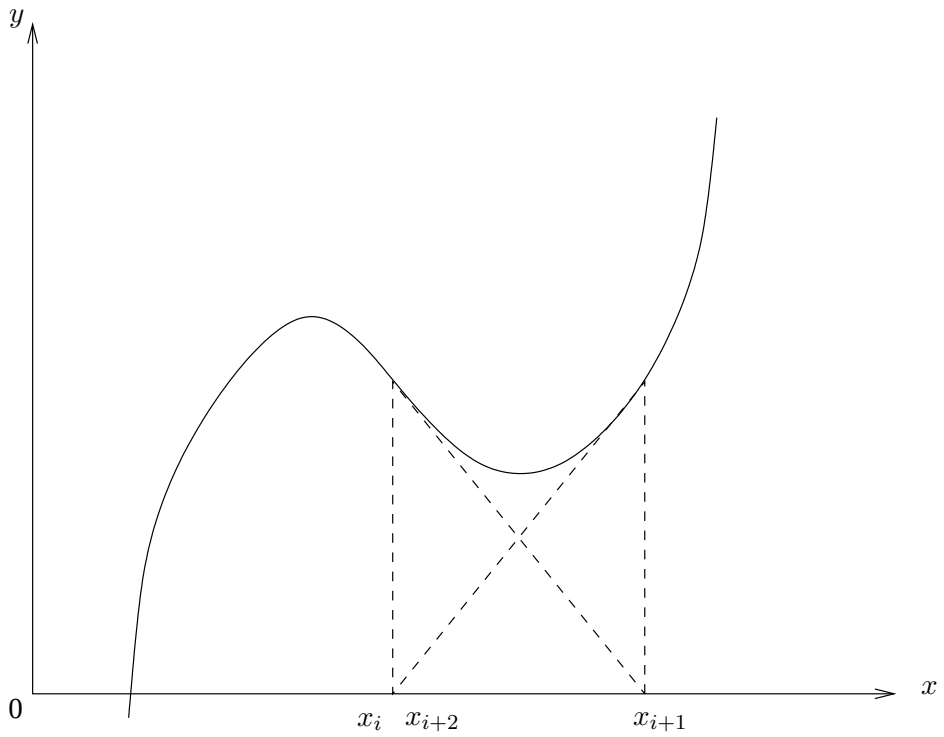


Abbildung 1.5: Divergenz des NEWTON-Verfahrens

Verallgemeinerung auf n Dimensionen. Sei $f : \mathbb{R}^n \supset \mathcal{G} \to \mathcal{G}$ stetig differenzierbar. Die NEWTON-Iterationsvorschrift lautet dann

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - f'(\boldsymbol{x}_i)^{-1} f(\boldsymbol{x}_i) \qquad (i = 0, 1, \dots) \tag{1.24}$$

mit der Funktionalmatrix $f'(\boldsymbol{x}_i)$. In der Praxis wird nicht $f'(\boldsymbol{x}_i)^{-1}$ berechnet, da dies meist zu aufwändig ist, sondern das lineare Gleichungssystem

$$f'(\boldsymbol{x}_i)\boldsymbol{\Delta}_i = -f(\boldsymbol{x}_i) \tag{1.25}$$

gelöst und dann

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{\Delta}_i \tag{1.26}$$

berechnet. Ist die Funktionalmatrix regulär, so ist das Verfahren lokal konvergent von mindestens 2-ter Ordnung. Die Berechnung der Funktionalmatrix kann allerdings sehr aufwändig sein.

Bemerkung: Vereinfachtes NEWTON-Verfahren. Für einen guten Startwert $\boldsymbol{x}_0$ wird $f'(\boldsymbol{x}_0)$ berechnet und $\boldsymbol{\Delta}_i$ aus

$$f'(\boldsymbol{x}_0)\boldsymbol{\Delta}_i = -f(\boldsymbol{x}_i)$$

bestimmt. $\Longrightarrow$ nur lineare Konvergenz

1.2.4 Schwierigkeiten bei der Nullstellenbestimmung

Bei der Bestimmung von Nullstellen können folgende Situationen auftreten:

(i) *Zwei deutlich voneinander getrennte Nullstellen*
gut konditioniert

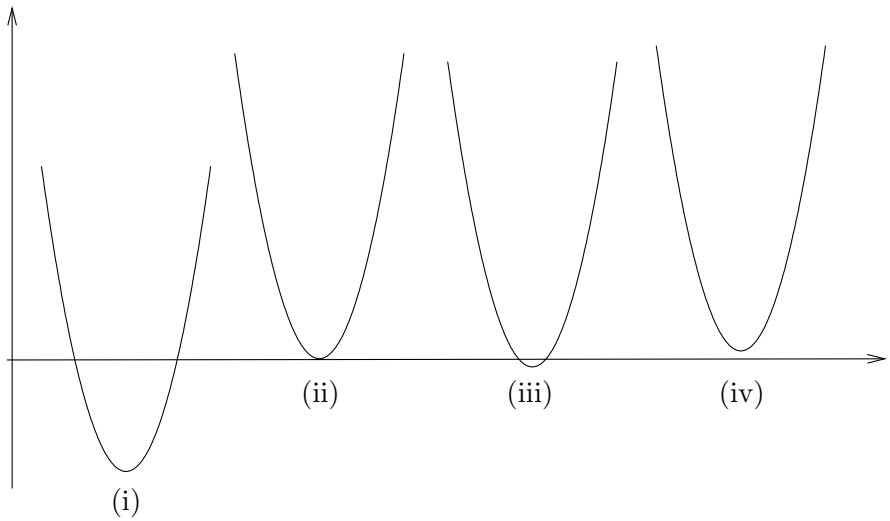


Abbildung 1.6: Unterschiedliche Situationen bei der Nullstellenbestimmung

(ii) *Eine doppelte Nullstelle*
Eine leichte vertikale Verschiebung (z.B. durch Rechenfehler bei der Funktionsauswertung der Funktion) führt zu keiner oder zwei reellen Nullstellen.

(iii) *Zwei dicht beieinanderliegende Nullstellen*
Schwierig, wenn nicht irgendeine Nullstelle, sondern beide Nullstellen gesucht werden.

(iv) *Zwei nichtreelle Nullstellen nahe einer doppelten reellen Nullstelle*
Eine Rechnung im Reellen versagt.

Die Nullstellenprobleme (ii), (iii) und (iv) sind schlecht konditioniert.

1.2.5 Nullstellen von Polynomen

In der Praxis müssen häufig Nullstellen von Polynomen berechnet werden (z.B. bei Eigenwertproblemen). Dazu ist das NEWTON-Verfahren gut geeignet (effizient, einfach zu programmieren). Bei diesem Verfahren müssen das Polynom und seine erste Ableitung in jedem Iterationsschritt an einer anderen Stelle berechnet werden. Dies sollte möglichst effizient geschehen.

Effiziente Auswertung eines Polynoms

Wir betrachten ein Polynom n-ten Grades

$$p_n(x) := a_0 x^n + a_1 x^{n-1} + \cdots + a_n \text{ mit } a_i \in \mathbb{R}\ (\mathbb{C}),\ a_0 \neq 0 \tag{1.27}$$

Die effiziente Berechnung des Polynomwertes an einer Stelle beruht auf der Division mit Rest des Polynoms durch einen linearen Faktor

$$p_n(x) = (x - \tilde{x}) p_{n-1}(x) + r. \tag{1.28}$$

Mit der Darstellung

$$p_{n-1}(x) = b_0 x^{n-1} + b_1 x^{n-2} + \cdots + b_{n-1} \tag{1.29}$$

($b_0 \neq 0$) des Quotientenpolynoms folgt aus (1.28) durch Koeffizientenvergleich

$$a_0 = b_0, \quad a_1 = b_1 - \tilde{x} b_0, \quad a_2 = b_2 - \tilde{x} b_1, \quad \ldots, a_n = b_n - \tilde{x} b_{n-1} \tag{1.30}$$

mit

$$b_n := r \tag{1.31}$$

Somit können die Koeffizienten des Quotientenpolynoms rekursiv nach

$$b_0 = a_0, \quad b_j = a_j + \tilde{x} b_{j-1} \qquad (j = 1, 2, \ldots, n) \tag{1.32}$$

berechnet werden. Offensichtlich ist nach (1.28)

$$p_n(\tilde{x}) = r \tag{1.33}$$

d.h. der Wert des Polynoms $p_n(x)$ an der Stelle $\tilde{x}$ ist gleich dem Rest r bei der Division von $p_n(x)$ durch $x - \tilde{x}$. Zur Berechnung von

$$p_n(\tilde{x}) = b_n \tag{1.34}$$

sind jeweils nur n Additionen und Multiplikationen notwendig. Die bei der Polynomdivision auftretenden Größen können im sogenannten HORNER-Schema zusammengefasst werden:

a_0	a_1	a_2	$\ldots$	a_{n-1}	a_n
	$\tilde{x}b_0$	$\tilde{x}b_1$	$\ldots$	$\tilde{x}b_{n-2}$	$\tilde{x}b_{n-1}$
	b_1	b_2	$\ldots$	b_{n-1}	b_n

Der Wert des Quotientenpolynoms $p_{n-1}(x)$ an der Stelle $\tilde{x}$ liefert die erste Ableitung $p_n'(x)$ des ursprünglichen Polynoms an dieser Stelle. Denn aus

$$p_n'(x) = p_{n-1}(x) + (x - \tilde{x}) p_{n-1}'(x) \tag{1.35}$$

folgt

$$p_n'(\tilde{x}) = p_{n-1}(\tilde{x}). \tag{1.36}$$

$p_{n-1}(\tilde{x})$ kann auf die gleiche Weise wie $p_n(\tilde{x})$ berechnet werden. Setzen wir

$$p_{n-1}(x) = (x - \tilde{x}) p_{n-2}(x) + c_{n-1} \tag{1.37}$$

mit

$$p_{n-2}(x) = c_0 x^{n-2} + c_1 x^{n-3} + \cdots + c_{n-2}, \tag{1.38}$$

dann gilt

$$c_0 = b_0, \quad c_j = b_j + \tilde{x} c_{j-1} \qquad (j = 1, 2, \ldots, n-1) \tag{1.39}$$

Es gilt

$$p_n'(\tilde{x}) = c_{n-1} \tag{1.40}$$

Im Rechner kann man ohne indizierte Variablen arbeiten.

Algorithmus:

- $b = a_0$, $c = b$, $b = a_1 + \tilde{x} b$
- für $j = 2, 3, \ldots, n$:

  $c = b + \tilde{x} c$

  $b = a_j + \tilde{x} b$

  Zuerst wird also c_{j-1} aus b_{j-1} und dann b_j aus b_{j-1} berechnet
- $p_n(\tilde{x}) = b$, $p_n'(\tilde{x}) = c$

Abspaltung von Nullstellen

Ist $x^{(1)}$ eine Nullstelle des Polynoms $p_n(x)$, so liefert die Division durch den Linearfaktor $x - x^{(1)}$ ein Quotientenpolynom $p_{n-1}(x)$, dessen Nullstellen die restlichen Nullstellen des Polynoms $p_n(x)$ sind, dabei tritt kein Rest auf. Für eine Nullstelle $x^{(2)}$ von $p_{n-1}(x)$ führt die Abspaltung von $x - x^{(2)}$ zu einem Quotientenpolynom $p_{n-2}(x)$, dessen Nullstellen die restlichen Nullstellen von $p_{n-1}(x)$ sind, usw.

Die sukzessive Bestimmung der Nullstellen eines Polynoms mit dem NEWTON-Verfahren und expliziter Abspaltung der Nullstellen kann in der Praxis ungünstig sein:
Der berechnete Wert einer Nullstelle des Polynoms ist nur eine Näherung für den exakten Wert. Bei der Berechnung der Koeffizienten des Quotientenpolynoms entstehen Rundungsfehler. Eine Nullstelle des verfälschten Quotientenpolynoms kann dann deutlich von derjenigen des exakten Quotientenpolynoms abweichen, wenn eine große Empfindlichkeit der Nullstelle gegenüber kleiner Änderungen der Koeffizienten besteht. Dieser Effekt kann sich im Laufe der Bestimmung der einzelnen Nullstellen so verstärken, dass die zuletzt berechneten Nullstellen nicht einmal eine grobe Näherung für die exakten Nullstellen darstellen.

Diese Schwierigkeiten kann man vermeiden, wenn die bereits bekannten Nullstellen nur implizit abgespalten werden, d.h. es wird immer nur mit den Koeffizienten des ursprünglichen Polynoms gearbeitet. Seien $x^{(1)}$, $x^{(2)}$, …, $x^{(n)}$ die Nullstellen von $p_n(x)$. Dann folgt aus

$$p_n(x) = a_0 \prod_{j=1}^{n} (x - x^{(j)}) \tag{1.41}$$

und

$$p_n'(x) = a_0 \sum_{j=1}^{n} \prod_{k=1, k\neq j}^{n} (x - x^{(k)}), \tag{1.42}$$

also

$$\frac{p_n'(x)}{p_n(x)} = \sum_{j=1}^{n} \frac{1}{x - x^{(j)}}. \tag{1.43}$$

Wir nehmen an, dass die ersten $1 \leq m < n$ Nullstellen $x^{(1)}$, $x^{(2)}$, …, $x^{(m)}$ bereits bekannt seien. Nach ihrer Abspaltung bleibt das Quotientenpolynom

$$p_{n-m}(x) = \frac{p_n(x)}{\prod_{j=1}^{m}(x - x^{(j)})} = a_0 \prod_{j=m+1}^{n} (x - x^{(j)}) \tag{1.44}$$

übrig. Seine erste Ableitung ist

$$p_{n-m}'(x) = a_0 \sum_{j=m+1}^{n} \prod_{k=m+1, k\neq j}^{n} (x - x^{(k)}), \tag{1.45}$$

mithin

$$\frac{p_{n-m}'(x)}{p_{n-m}(x)} = \sum_{j=m+1}^{n} \frac{1}{x - x^{(j)}}. \tag{1.46}$$

Somit gilt

$$\frac{p_{n-m}'(x)}{p_{n-m}(x)} = \frac{p_n'(x)}{p_n(x)} - \sum_{j=1}^{m} \frac{1}{x - x^{(j)}}. \tag{1.47}$$

Demnach lautet die Iterationsvorschrift zur Bestimmung der $(m+1)$-ten Nullstelle

$$x_{i+1} = x_i - \frac{1}{\frac{p_n'(x_i)}{p_n(x_i)} - \sum_{j=1}^{m} \frac{1}{x_i - x^{(j)}}} \qquad (i = 0, 1, \dots). \tag{1.48}$$

Der Rechenaufwand beträgt $2n + m + 1$ wesentliche Operationen ($\uparrow$ für $m \uparrow$).

Das beschriebene Verfahren funktioniert nicht nur für reelle Nullstellen von Polynomen mit reellen Koeffizienten, sondern auch für nichtreelle Nullstellen oder Nullstellen von Polynomen mit komplexen Koeffizienten. Im ersten Fall muss der Startwert nichtreell gewählt werden.

Bemerkung. Das Rechnen mit komplexen Zahlen kann vermieden werden, wenn alle Koeffizienten des Polynoms reell sind. Besitzt es eine nichtreelle Nullstelle $\tilde{x} = u + \mathrm{i}v$ mit $u, v \in \mathbb{R}$, so ist der konjungiert komplexe Wert $\tilde{x} = u - \mathrm{i}v$ auch eine Nullstelle. u und v werden als Nullstelle der Koeffizienten des linearen Restpolynoms bei der Division des Polynoms durch den Quadratfaktor

$$(x - \tilde{x})(x - \tilde{x}^*) = x^2 - 2ux + u^2 + v^2$$

bestimmt.

1.3 Minimieren einer Funktion

Das Nullstellenproblem lässt sich auch als Minimierungsproblem formulieren:

Sei $f : \mathbb{R}^n \supset \mathcal{G} \to \mathbb{R}$ und $h := f^2$. f hat genau dann eine Nullstelle bei $\boldsymbol{x}_* \in \mathcal{G}$, wenn h eine Minimalstelle bei $\boldsymbol{x}_*$ mit Minimalwert 0 hat.

Zur Minimierung einer Funktion $h \in \mathcal{C}^1(\mathcal{G})$ wird häufig das *Gradientenverfahren* angewendet. Dabei wird ausgenützt, dass der negative Gradient einer Funktion in Richtung der stärksten lokalen Abnahme der Funktion zeigt.

Algorithmus

(i) Wähle Startvektor $\boldsymbol{x}_0 \in \mathbb{R}^n$ und setze $i = 0$.

(ii) Berechne Suchvektor $\boldsymbol{s}_i := \nabla h(\boldsymbol{x}_i)$.

(iii) Bestimme $0 < \alpha_i \leq \alpha_{\max}$, so dass $h_i(\alpha) := h(\boldsymbol{x}_i + \alpha \boldsymbol{s}_i)$ ein Minimum bei α_i annimmt.

(iv) Berechne $\boldsymbol{x}_{i+1} := \boldsymbol{x}_i + \alpha \boldsymbol{s}_i$

(v) Erhöhe i um 1 und gehe nach ii).

Abbruch, wenn $|\boldsymbol{s}_i|$ oder α_i nahe 0 ist oder i zu groß wird.

Kapitel 2

Interpolation

Problem: Gegeben seien $n+1$ Datenpaare (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) mit $x_i \neq x_j$ $(i \neq j)$. Es wird angenommen, dass die Punkte (x_i, y_i) auf dem Graphen einer glatten Funktion $y(x)$ liegen. Sie sei nur an den diskreten Stellen x_i bekannt (oder kompliziert zu berechnen). Gesucht ist ein Näherungswert $\tilde{y}$ der Funktion $y(x)$ an einer Zwischenstelle $\tilde{x}$.

Ansatz: Verbinden der Punkte (x_i, y_i) durch eine einfache glatte Funktion $p(x)$ mit $p(x_i) = y_i \ \forall i$ und Näherung von $y(\tilde{x})$ durch $p(\tilde{x})$.

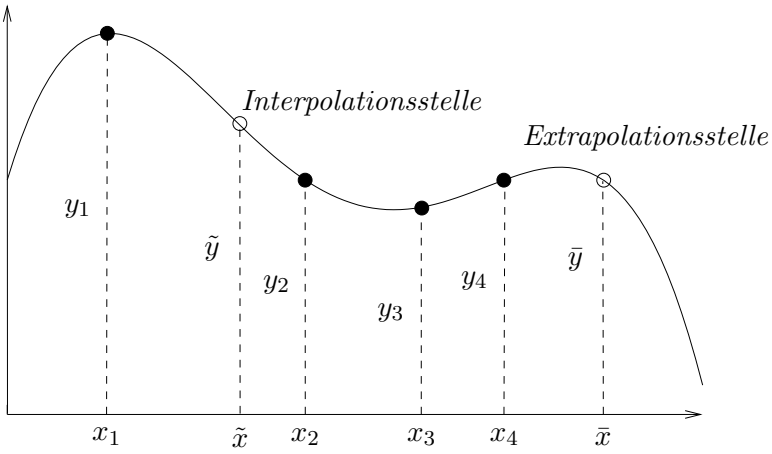


Abbildung 2.1: Interpolationsfunktion mit Stützstellen x_i und Stützwerten y_i

Anwendungen:

- Interpolation von numerischen Daten
- Interpolation von Messdaten (bei kleinen Messfehlern)
  Bemerkung: Ausgleichsrechnung bei Messdaten mit relevanten Fehlern
- numerische Differentiation
- Gewinnung von Quadraturformeln

2.1 Beispiel (Temperaturverlauf an einem Tag). Abbildung 2.2 zeigt den interpolierten Temperaturverlauf eines Tages, der aus den Temperaturwerten von verschiedenen Zeiten durch Polynominterpolation berechnet wurde.

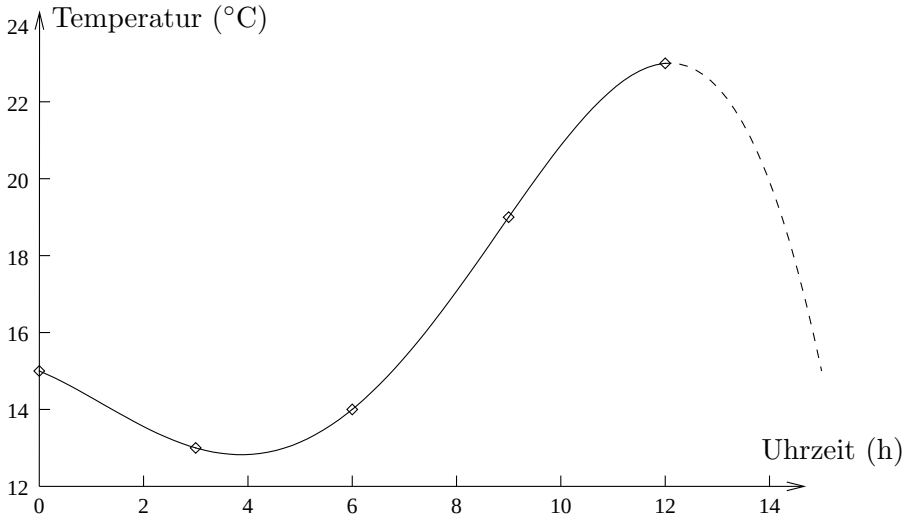


Abbildung 2.2: Interpolationsfunktion zu den Daten $\{(0,15),(3,13),(6,14),(9,19),(12,23)\}$

2.1 Polynominterpolation

Polynom (höchstens) n-ten Grades

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \text{ mit } a_i \in \mathbb{R}\,(\mathbb{C}) \tag{2.1}$$

2.1.1 Existenz und Eindeutigkeit

Gegeben seien $n+1$ paarweise verschiedene Stützstellen x_0, x_1, …, x_n mit zugehörigen beliebigen Stützwerten y_0, y_1, …, y_n. Zu diesen Stützstellen betrachten wir die sogenannten LAGRANGE-*Polynome*

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \qquad (i = 0, 1, \ldots, n) \tag{2.2}$$

vom Grad n. Sie haben offensichtlich die Eigenschaft

$$L_i(x_k) = \delta_{ik} \qquad (i, k = 0, 1, \ldots, n) \tag{2.3}$$

Daher erfüllt das Polynom höchstens n-ten Grades (LAGRANGE'sche-Interpolationsformel)

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x) \tag{2.4}$$

die Interpolationsbedingungen

$$p_n(x_k) = y_k \qquad (k = 0, 1, \ldots, n). \tag{2.5}$$

Sei $q_n(x)$ auch ein Interpolationspolynom höchstens n-ten Grades. Das Polynom $p_n(x) - q_n(x)$ von höchstens n-tem Grad hat dann $n+1$ Nullstellen und ist daher identisch Null nach dem Fundamentalsatz der Algebra. Wir haben somit den folgenden Satz bewiesen:

2.2 Satz. *Zu beliebigen $n+1$ Stützstellen (x_0, y_0), (x_1, y_1), …, (x_n, y_n) mit $x_i \neq x_j$ für $i \neq j$ $(i, j = 0, 1, \ldots, n)$ existiert genau ein Polynom höchstens n-ten Grades $p_n(x)$ mit der Eigenschaft $p_n(x_i) = y_i$ $(i = 0, 1, \ldots, n)$.*

2.1.2 LAGRANGE'sche Interpolationsformel

Die LAGRANGE'sche Interpolationsformel (2.4)

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

mit den Lagrangeschen-Polynomen (2.2)

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \qquad (i = 0, 1, \ldots, n).$$

löst explizit die Interpolationsaufgabe.

2.3 Beispiel. Aus den Stützpunkten $(-1,-1), (0,-1), (2,2)$ erhalten wir mit $x_0 = -1$, $x_1 = 0$, $x_2 = 2$ und $n = 2$:

$$\begin{aligned} L_0(x) &= \frac{x(x-2)}{(-1)(-3)} = \frac{1}{3}x^2 - \frac{2}{3}x, \\ L_1(x) &= \frac{(x+1)(x-2)}{1(-2)} = -\frac{1}{2}x^2 + \frac{1}{2}x + 1, \\ L_2(x) &= \frac{(x+1)x}{3 \cdot 2} = \frac{1}{6}x^2 + \frac{1}{6}x, \end{aligned}$$

also

$$p_2(x) = \frac{1}{2}x^2 + \frac{1}{2}x - 1, \qquad \text{z.B. } \tilde{x} = 1: \quad p_2(1) = 0.$$

Die Lagrangesche Interpolationsformel ist zwar wichtig für theoretische Überlegungen (z.B. zur Herleitung von Quadraturformeln in Kapitel 7), kann aber für praktische Zwecke ungeeignet sein. Die Lagrange-Polynome müssen nämlich neu berechnet werden, wenn ein weiterer Stützpunkt hinzukommt. Die beiden folgenden, rekursiven Methoden erfordern in dieser Hinsicht keinen solchen großen Aufwand.

2.1.3 Rekursionsformel von NEVILLE

Sei $p_{i,i+1,\ldots,j}(x)$ mit $j \geq i$ das eindeutig bestimmte Polynom höchstens $(j-i)$-ten Grades mit $p_{i,i+1,\ldots,j}(x_k) = y_k$ für alle $i \leq k \leq j$. Diese Polynome erfüllen die sogenannte *Rekursionsformel von* NEVILLE:

$$p_{i,\ldots,j}(x) = \frac{(x - x_i)p_{i+1,\ldots,j}(x) - (x - x_j)p_{i,\ldots,j-1}(x)}{x_j - x_i} \qquad (j > i) \tag{2.6}$$

2.4 Beweis. *Sei $p(x)$ das Polynom auf der rechten Seite von (2.6). Offensichtlich ist sein Grad höchstens $j - i$. Es gilt $p(x_i) = p_{i,\ldots,j-1}(x_i) = y_i$ und $p(x_j) = p_{i+1,\ldots,j}(x_j) = y_j$. Auch für $i < k < j$ gilt $p(x_k) = \frac{(x_k - x_i)y_k - (x_k - x_j)y_k}{(x_j - x_i)} = y_k$. Mithin ist $p(x) = p_{i,\ldots,j}(x)$ (Eindeutigkeit).* □

Diese Rekursionsformel erlaubt die Berechnung einer Interpolationsstelle $\tilde{x}$ nach dem Schema in Abbildung 2.3. Die Stützstellen müssen dabei nicht monoton angeordnet sein.

2.5 Beispiel (Fortsetzung von Beispiel 2.3). Das Rekursionsschema von Neville für das Beispiel 2.3 wird in Abbildung 2.4 gezeigt.

Algorithmus.

- Definiere ein Feld der Länge $n + 1$:

  j-te Spalte $(j = 0, 1, \ldots, n)$:

  $p[i] = p_{i-j,i-j+1,\ldots,i}(\tilde{x})$ $(i = j, j+1, \ldots, n)$

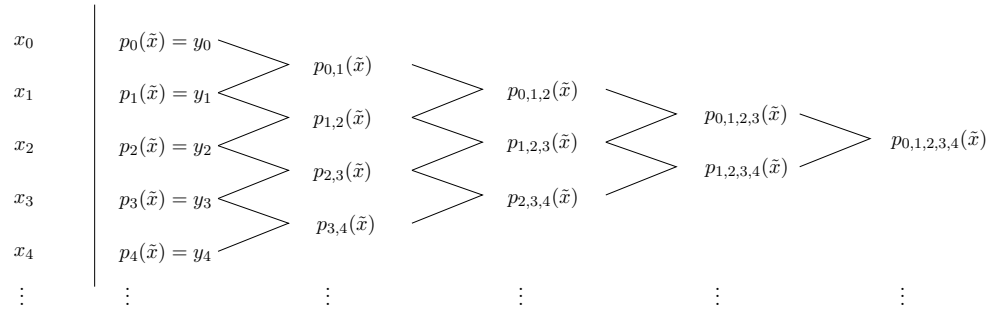


Abbildung 2.3: Rekursionsschema nach NEVILLE

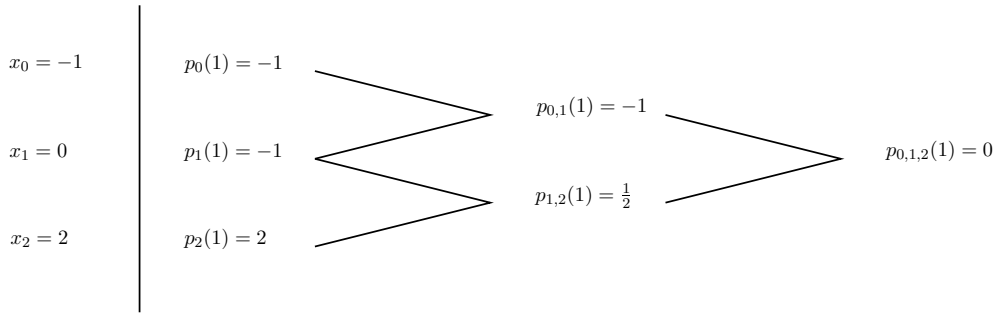


Abbildung 2.4: Berechnung eines Polynomwertes (Beispiel 2.3)

- für $i = 0, 1, \ldots, n$

  $p[i] = y_i$

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

  $p[i] = p[i] + (\tilde{x} - x_i)(p[i] - p[i-1])/(x_i - x_{i-j})$

  $(= ((\tilde{x} - x_{i-j})p[i] - (\tilde{x} - x_i)p[i-1])/(x_i - x_{i-j}))$

Es werden also $2n + 2(n-1) + \cdots + 2 \cdot 1 = 2\sum_{i=0}^{n} i = n(n+1)$ wesentliche Operationen benötigt.

Die NEVILLE-Rekursion eignet sich zur Auswertung des Interpolationspolynoms an einer bestimmten Stelle, aber nicht zur Bestimmung des Polynoms selbst. Dazu ist die NEWTON-Rekursion geeignet.

2.1.4 NEWTON'sche Interpolationsformel

Ein weiterer Ansatz zur expliziten Darstellung des Interpolationspolynoms ist

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})\,. \qquad (2.7)$$

Die Koeffizienten a_0, a_1, a_2, …, a_n sind durch die Interpolationsbedingungen eindeutig bestimmt:

$$\begin{aligned}
p_n(x_0) &= a_0 &&= y_0\\
p_n(x_1) &= a_0 + a_1(x_1 - x_0) &&= y_1\\
p_n(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) &&= y_2\\
&\vdots
\end{aligned}$$

Also:

$$\begin{aligned} a_0 &= y_0 \\ a_1 &= \frac{y_1 - y_0}{x_1 - x_0} \\ a_2 &= \frac{\frac{y_2-y_1}{x_2-x_1} - \frac{y_1-y_0}{x_1-x_0}}{x_2 - x_0} \\ &\vdots \end{aligned}$$

Für die Berechnung der Koeffizienten ist es zweckmäßig, die sogenannten *dividierten Differenzen* einzuführen. Es gilt

$$f_i := y_i \qquad (i = 0, 1, ..., n) \tag{2.8a}$$

$$f_{i,...,j} := \frac{f_{i+1,...,j} - f_{i,...,j-1}}{x_j - x_i} \qquad (j > i) \tag{2.8b}$$

Berechnungsschema : Um die *dividierten Koeffizienten* zu berechnen, geht man nach dem Schema in Abbildung 2.5 vor. Für die Koeffizienten des Interpolationspolynoms 2.7 gilt $a_i = f_{0,1,...,i}$.

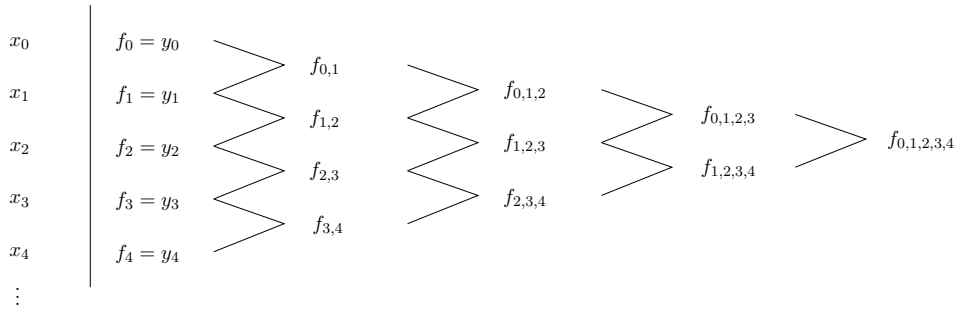


Abbildung 2.5: Berechnungsschema für die NEWTONSCHE-Interpolationsformel

2.6 Beweis.

$$\begin{aligned} f_0 &= y_0 = a_0 \\ f_{0,1} &= \frac{f_1 - f_0}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0} = a_1 \\ f_{0,1,2} &= \frac{f_{1,2} - f_{0,1}}{x_2 - x_0} = \frac{\frac{f_2-f_1}{x_2-x_1} - \frac{f_1-f_0}{x_1-x_0}}{x_2 - x_0} = \frac{\frac{y_2-y_1}{x_2-x_1} - \frac{y_1-y_0}{x_1-x_0}}{x_2 - x_0} = a_2 \\ &\vdots \end{aligned}$$

(Vollständige Induktion) □

2.7 Beispiel (Fortsetzung von Beispiel 2.3). Die Abbildung 2.6 zeigt schematisch die Berechnung des Interpolationspolinoms mit Hilfe der NEWTON'schen Interpolationsformel. Man erhält als Interpolationspolynom

$$p_3(x) = -1 + 0 \cdot (x + 1) + \frac{1}{2}(x + 1)x = \frac{1}{2}x^2 + \frac{1}{2}x - 1$$

Algorithmus.

- Definiere ein Feld der Länge $n + 1$:
  j-te Spalte $(j = 0, 1, ..., n)$:
  $f[i] = f_{i-j,i-j+1,...,i}$ $(i = j, j + 1, ..., n)$

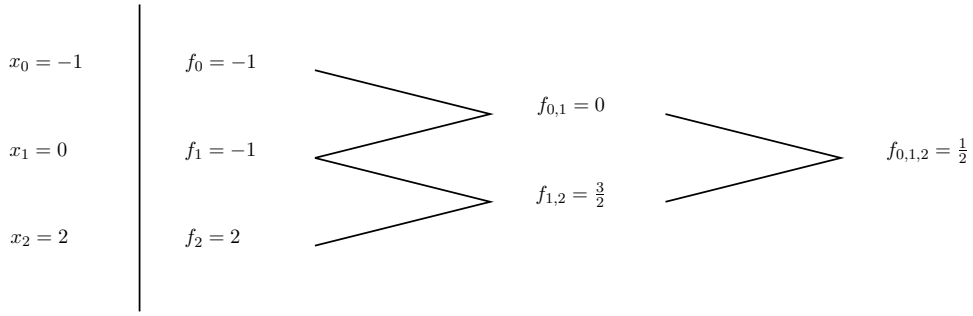


Abbildung 2.6: Berechnung eines Interpolationspolynoms (Beispiel 2.7)

- für $i = 0, 1, ..., n$:
  $f[i] = y_i$

  für $j = 1, 2, ..., n$
  für $i = n, n-1, ..., j$:
  $f[i] = (f[i] - f[i-1])/(x_i - x_{i-j})$
  (Zu beachten ist, dass kein noch benötigter Zahlenwert verloren geht.)

Bei diesem Algorithmus werden $n+(n-1)+\ldots+1 = \frac{1}{2}n(n+1)$ wesentliche Operationen durchgeführt.

Berechnung eines Polynomwertes. Auf der Schreibweise

$$p_n(x) = a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + (x - x_2)(\ldots(a_{n-1} + (x - x_{n-1})a_n)))) \tag{2.9}$$

des Interpolationspolynoms beruht eine effiziente Berechnung eines Polynomwertes.

Algorithmus:

- $p = a_n$
- für $i = n-1, n-2, \ldots, 0$:
  $p = a_i + (x - x_i)p$

Es ist dann $p_n(x) = p$. Hierfür werden n wesentliche Operationen benötigt.

2.1.5 Fehlerabschätzung

Wir nehmen an, dass die Stützwerte y_i von einer Funktion $y(x)$ herrühren ($y(x_i) = y_i, i = 0, 1, ..., n$) und fragen uns, wie gut das Interpolationspolynom $p_n(x)$ die Funktion $y(x)$ an den Stellen wiedergibt, die keine Stützstellen x_i sind.

2.8 Satz. *Sei $y \in \mathcal{C}^{n+1}([a,b]), x_i \in [a,b] (i = 0, 1, ..., n)$ mit $x_i \neq x_j$ für $i \neq j$ und $\min_i x_i = a$, $\max_i x_i = b$ und $y_i := y(x_i) (i = 0, 1, \ldots, n)$. Sei ferner $p_n(x)$ das eindeutig bestimmte Polynom höchstens n-ten Grades mit $p_n(x_i) = y_i (i = 0, 1, ..., n)$. Dann gilt die Abschätzung*

$$|y(\tilde{x}) - p_n(\tilde{x})| \leq \left|\prod_{i=0}^{n}(\tilde{x} - x_i)\right| \max_{x\in[a,b]} \frac{|y^{n+1}(x)|}{(n+1)!} \qquad (\tilde{x} \in [a,b]). \tag{2.10}$$

2.9 Beweis. *Sei $\tilde{x} \in [a,b]$. Für $\tilde{x} = x_i$ ist die Ungleichung offenbar erfüllt. Sei nun $\tilde{x} \neq x_i (i = 0,1,...,n)$. Dann hat die $(n+1)$-mal stetig differenzierbare Funktion*

$$Y(x) := y(x) - p_n(x) - \frac{y(\tilde{x}) - p_n(\tilde{x})}{\prod_{i=0}^{n}(\tilde{x} - x_i)} \prod_{i=0}^{n}(x - x_i)$$

mindestens $n+2$ Nullstellen (x_0, x_1, ..., x_n, $\tilde{x}$). Aus dem Satz von Rolle *folgt sukzessive, dass $Y'(x)$ mindestens $n+1$ Nullstellen, $Y''(x)$ mindestens n Nullstellen, ... und $Y^{(n+1)}(x)$ mind. eine Nullstelle ξ besitzt. Es gilt also:*

$$0 = Y^{(n+1)}(\xi) = y^{(n+1)}(\xi) - \frac{y(\tilde{x}) - p_n(\tilde{x})}{\prod_{i=0}^{n}(\tilde{x} - x_i)}(n+1)!$$

und damit

$$|y(\tilde{x}) - p_n(\tilde{x})| = \left|\prod_{i=0}^{n}(\tilde{x} - x_i)\right| \frac{|y^{(n+1)}(\xi)|}{(n+1)!} \leq \left|\prod_{i=0}^{n}(\tilde{x} - x_i)\right| \max_{x\in[a,b]} \frac{|y^{(n+1)}(x)|}{(n+1)!} \quad \square.$$

Konvergiert die Polynominterpolation?

2.10 Beispiel (Runge-Funktion).

$$y(x) := \frac{1}{1 + (5x)^2}, \qquad x \in [-1,1] \tag{2.11}$$

Wir betrachten die Stützstellen $x_i = -1 + 2i/n$, $i = 0,1,...,n$. Wie man in Abbildung 2.7 sieht, approximiert p_{16} die Funktion $y(x)$ in der Mitte des Intervalls gut, zum Rand hin jedoch nur sehr schlecht (starke Oszillationen von p_{16} nahe dem Rand). Für den maximalen Interpolationsfehler ergibt sich in Abhängigkeit der Stützstellenanzahl:

n	1	5	13	19
$\max_{x\in[-1,1]} \lvert f(x) - p_n(x)\rvert$	0.96	0.43	1.07	8.57

Man sieht, dass der maximale Interpolationsfehler mit steigender Anzahl von Stützstellen zunimmt ($\max_{x\in[-1,1]} |f(x) - p_n(x)| \to \infty$). Die Polynominterpolation ist nicht in jedem Fall konvergent.

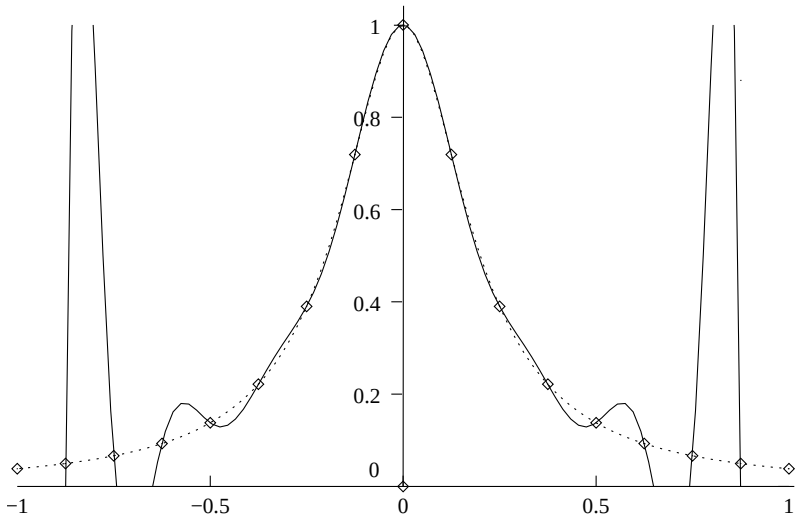


Abbildung 2.7: Polynominterpolation der Rungefunktion mit $n = 16$

Die Methode der Interpolation durch Polynome sollte nicht bei vielen Stützstellen angewandt werden. Sonst würde das Interpolationspolynom starke Oszillationen aufweisen und damit könnten die interpolierten Werte unbrauchbar werden. Bei vielen Stützstellen sollten stückweise Polynome, sogenannte Splines, die sich aus Polynomen niedrigen Grades zusammensetzen, bei der Interpolation verwendet werden (siehe Abschnitt 2.3).

2.2 Numerische Differentiation

Über das Interpolationspolynom $p_n(x)$ zu einer tabellarisch gegebenen Funktion $y(x)$ kann ihre m-te Ableitung ($m \leq n$) näherungsweise berechnet werden:

$$y^{(m)}(x) \approx p_n^{(m)}(x) \qquad \text{für } m \leq n \tag{2.12}$$

Im Folgenden setzen wir äquidistante Stützstellen

$$x_i := x_0 + i\,h (i = 0, 1, ..., n) \qquad \text{mit “kleinem” } h > 0 \tag{2.13}$$

voraus.

Spezialfälle.

(a) $n = 1$

$$p_1(x) = \sum_{i=0}^{1} y_i \prod_{j=0, j\neq i}^{1} \frac{x - x_j}{x_i - x_j} = \frac{1}{h}(-y_0(x - x_1) + y_1(x - x_0)) \tag{2.14a}$$

$$p_1'(x) = \frac{1}{h}(y_1 - y_0) \tag{2.14b}$$

$m = 1$:

$$y'(x_0) \approx \frac{1}{h}(y_1 - y_0) \qquad \text{(Fehler } \mathcal{O}(h)) \tag{2.15}$$

(b) $n = 2$

$$p_2(x) = \sum_{i=0}^{2} y_i \prod_{j=0, j\neq i}^{2} \frac{x - x_j}{x_i - x_j} = \tag{2.16a}$$

$$\frac{1}{2h^2}\left(y_0(x - x_1)(x - x_2) - 2y_1(x - x_0)(x - x_2) + y_2(x - x_0)(x - x_1)\right)$$

$$p_2'(x) = \frac{1}{2h^2}\left(y_0(2x - x_1 - x_2) - 2y_1(2x - x_0 - x_2) + y_2(2x - x_0 - x_1)\right) \tag{2.16b}$$

$$p_2''(x) = \frac{1}{h^2}(y_0 - 2y_1 + y_2) \tag{2.16c}$$

$m = 1$:

$$y'(x_1) \approx \frac{1}{2h}(y_2 - y_0) \qquad \text{Fehler } (\mathcal{O}(h^2)) \tag{2.17a}$$

$m = 2$:

$$y''(x_1) \approx \frac{1}{h^2}(y_0 - 2y_1 + y_2) \qquad \text{Fehler } (\mathcal{O}(h^2)) \tag{2.17b}$$

Bei sehr kleiner Schrittweite h treten wegen der Stellenauslöschung bei der Differenzbildung etwa gleich großer Zahlen große relative Fehler bei der numerischen Differentiation einer vorgegebenen Funktion $y(x)$ auf. Daher wird bei einer höheren Genauigkeitsanforderung die Ableitung für verschiedene nicht zu kleine Schrittweiten berechnet und anschließend eine Extrapolation nach $h = 0$ durchgeführt.

2.11 Beispiel (Zentraler Differenzenquotient).

$$y_2 = y(x_2) = y(x_1 + h) = y(x_1) + hy'(x_1) + \frac{h^2}{2}y''(x_1) + \frac{h^3}{6}y^{(3)}(x_1) + \frac{h^4}{24}y^{(4)}(x_1) + \ldots \tag{2.18a}$$

$$y_0 = y(x_0) = y(x_1 - h) = y(x_1) - hy'(x_1) + \frac{h^2}{2}y''(x_1) - \frac{h^3}{6}y^{(3)}(x_1) + \frac{h^4}{24}y^{(4)}(x_1) - \ldots \tag{2.18b}$$

Also

$$\frac{1}{2h}(y_2 - y_0) = y'(x_1) + \frac{h^2}{6}y^{(3)}(x_1) + \frac{h^4}{120}y^{(5)}(x_1) + \ldots \tag{2.19}$$

Berechnung von $\frac{1}{2h}(y_2 - y_0)$ sukzessive für $h_0 > h_1 > \ldots > h_k > \ldots > 0$ und anschließend jeweils Extrapolation nach $h = 0$ (über Interpolationspolynome). Abbruch, wenn extrapolierte Werte bei $k + 1$ und k gut übereinstimmen.

2.3 Spline-Interpolation

Um Polynome hohen Grades bei der Interpolation mit vielen Stützstellen zu vermeiden, kann man folgendermaßen vorgehen:

Unter Verzicht auf ein für das ganze Intervall $[x_0, x_n]$ einheitliches Polynom werden über jedem Teilintervall $[x_{i-1}, x_i]$, $i \in 1, 2, \ldots, n$ Polynome mit vorgegebener kleiner Maximalordnung konstruiert, die dann in geeigneter Weise (Stetigkeitsbedingungen) zusammengesetzt werden. Die Interpolation erfolgt hier über ein stückweises Polynom, den *Spline*. Im Folgenden werden nur die häufig verwendeten kubischen Splines betrachtet, die sich aus Polynomen höchstens dritten Grades zusammensetzen.

Gegeben seien $n+1$ Stützstellen $x_0 < x_1 < \ldots < x_n$ mit zugehörigen Stützwerten $y_0, y_1, \ldots, y_n$, $y_n \in \mathbb{R}$ ($\mathbb{C}$). Eine Funktion $s \in \mathcal{C}^{(2)}([x_0, x_n])$ mit $s \restriction [x_{i-1}, x_i]$ ein Polynom höchstens dritten Grades $(i = 1, 2, \ldots, n)$ heißt *kubischer Spline* (oder Spline dritten Grades). Die Interpolationsbedingungen sind

$$s(x_i) = y_i \qquad (i = 0, 1, \ldots, n). \tag{2.20}$$

Konstruktion eines kubischen Splines. Sei $h_i := x_i - x_{i-1}$ $(i = 1, 2, \ldots, n)$ und

$$s_i^{(1,2)} := s^{(1,2)}(x_i) \quad (i = 0, 1, \ldots, n).$$

Weil s'' stetig und $s'' \restriction [x_{i-1}, x_i]$ linear sein soll, gilt

$$s''(x) = -s''_{i-1}\frac{x - x_i}{h_i} + s''_i\frac{x - x_{i-1}}{h_i} \text{ für } \; x \in [x_{i-1}, x_i] \tag{2.21}$$

mit den Unbekannten s''_{i-1} und s''_i. Nach zweimaliger Integration ergibt sich

$$s(x) = -s''_{i-1}\frac{(x - x_i)^3}{6h_i} + s''_i\frac{(x - x_{i-1})^3}{6h_i} + a_i x + b_i \tag{2.22}$$

mit noch zu bestimmenden Konstanten a_i und b_i. Mit den Interpolationbedingungen

$$s_{i-1} = y_{i-1}: \qquad s''_{i-1}\frac{h_i^2}{6} + a_i x_{i-1} + b_i = y_{i-1} \tag{2.23a}$$

$$s_i = y_i: \qquad s''_i\frac{h_i^2}{6} + a_i x_i + b_i = y_i \tag{2.23b}$$

(Stetigkeit von $s(x)$) können a_i und b_i aus Gleichung (2.22) eliminiert werden:

$$s(x) = -s''_{i-1}\frac{(x - x_i)^3}{6h_i} + s''_i\frac{(x - x_{i-1})^3}{6h_i} + \left(\frac{y_i}{h_i} - s''_i\frac{h_i}{6}\right)(x - x_{i-1}) - \left(\frac{y_{i-1}}{h_i} - s''_{i-1}\frac{h_i}{6}\right)(x - x_i) \tag{2.24}$$

für $x \in [x_{i-1}, x_i]$. Nach Differentiation ergibt sich

$$s'(x) = -s''_{i-1}\frac{(x - x_i)^2}{2h_i} + s''_i\frac{(x - x_{i-1})^2}{2h_i} + \frac{y_i - y_{i-1}}{h_i} - (s''_i - s''_{i-1})\frac{h_i}{6}, \qquad x \in [x_{i-1}, x_i]. \tag{2.25}$$

Die Forderung der Stetigkeit von $s'(x)$ an der Stützstelle x_i führt zu

$$s''_i\frac{h_i}{2} + \frac{y_i - y_{i-1}}{h_i} - (s''_i - s''_{i-1})\frac{h_i}{6} = -s''_i\frac{h_{i+1}}{2} + \frac{y_{i+1} - y_i}{h_{i+1}} - (s''_{i+1} - s''_i)\frac{h_{i+1}}{6}$$

also

$$s''_{i-1}\frac{h_i}{6} + s''_i\frac{h_i + h_{i+1}}{3} + s''_{i+1}\frac{h_{i+1}}{6} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \qquad (i = 1, 2, \ldots, n-1). \tag{2.26}$$

Dies sind $n-1$ Gleichungen für die $n+1$ Unbekannten s''_0, s''_1, …, s''_n. Zur Vervollständigung gibt es mehrere Möglichkeiten (Randbedingungen):

i) periodische Randbedingungen (falls $y(x + (x_n - x_0)) = y(x)$)

$$s_0' = s_n', \tag{2.27a}$$
$$s_0'' = s_n'' \tag{2.27b}$$

ii) natürliche Randbedingungen

$$s_0'' = 0, \tag{2.28a}$$
$$s_n'' = 0 \tag{2.28b}$$

iii) vollständige Randbedingungen

$$s_0' = y_0', \tag{2.29a}$$
$$s_n' = y_n' \tag{2.29b}$$

zu (i) Gleichung (2.26) für $i = 1$ mit $s_0'' = s_n''$:

$$s_1'' \frac{h_1 + h_2}{3} + s_2'' \frac{h_2}{6} + s_n'' \frac{h_1}{6} = \frac{y_2 - y_1}{h_2} - \frac{y_1 - y_0}{h_1} \tag{2.30}$$

Zusätzliche Gleichung ((2.26) für $i = n$ mit $s_{n+1}'' = s_1''$, $h_{n+1} = h_1$, $y_{n+1} = y_1$):

$$s_1'' \frac{h_1}{6} + s_{n-1}'' \frac{h_n}{6} + s_n'' \frac{h_n + h_1}{3} = \frac{y_1 - y_n}{h_1} - \frac{y_n - y_{n-1}}{h_n} \tag{2.31}$$

Damit liegen insgesamt n Gleichungen für die n Unbekannten s_1'', s_2'', …, s_n'' vor. Wir erhalten damit das lineare Gleichungssystem in Matrixform

$$\begin{pmatrix} 2 & \lambda_1 & 0 & \dots & 0 & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & & 0 \\ 0 & \mu_3 & 2 & \lambda_3 & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & 0 & \dots & 0 & \mu_n & 2 \end{pmatrix} \begin{pmatrix} s_1'' \\ s_2'' \\ \\ \vdots \\ \\ s_n'' \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \\ \vdots \\ \\ d_n \end{pmatrix} \tag{2.32}$$

mit

$$\lambda_i := \frac{h_{i+1}}{h_i + h_{i+1}}, \tag{2.33a}$$
$$\mu_i := 1 - \lambda_i = \frac{h_i}{h_i + h_{i+1}} \tag{2.33b}$$

und

$$d_i := \frac{6}{h_i + h_{i+1}} \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right) \qquad (i = 1, 2, \dots, n-1), \tag{2.33c}$$

sowie

$$\lambda_n := \frac{h_1}{h_n + h_1}, \tag{2.34a}$$
$$\mu_n := 1 - \lambda_n = \frac{h_n}{h_n + h_1}, \tag{2.34b}$$
$$d_n := \frac{6}{h_n + h_1} \left(\frac{y_1 - y_n}{h_1} - \frac{y_n - y_{n-1}}{h_n} \right). \tag{2.34c}$$

zu (ii) Mit den zusätzlichen Gleichungen (2.28) $s_0'' = 0$, $s_n'' = 0$ ergibt sich folgendes lineares Gleichungssystem

$$\begin{pmatrix} 2 & \lambda_0 & 0 & \dots & 0 & 0 \\ \mu_1 & 2 & \lambda_1 & & & 0 \\ 0 & \mu_2 & 2 & \lambda_2 & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ 0 & 0 & \dots & 0 & \mu_n & 2 \end{pmatrix} \begin{pmatrix} s_0'' \\ s_1'' \\ \\ \vdots \\ \\ s_n'' \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \\ \vdots \\ \\ d_n \end{pmatrix} \tag{2.35}$$

Die λ_i, μ_i und d_i können gemäß (2.33) und

$$\lambda_0 := 0\,, \tag{2.36a}$$
$$\mu_0 := 1 - \lambda_0 = 1\,, \tag{2.36b}$$
$$d_0 := 0, \tag{2.36c}$$

sowie

$$\lambda_n := 1\,, \tag{2.37a}$$
$$\mu_n := 1 - \lambda_n = 0\,, \tag{2.37b}$$
$$d_n := 0 \tag{2.37c}$$

berechnet werden.

zu (iii) Aus Gleichung (2.25) ergibt sich für $i = 1$ mit $s_0' = f_0'$

$$s_0'' \frac{h_i}{3} + s_1'' \frac{h_1}{6} = \frac{y_1 - y_0}{h_1} - y_0' \tag{2.38}$$

und für $i = n$ mit $s_n' = f_n'$

$$s_{n-1}'' \frac{h_n}{6} + s_n'' \frac{h_n}{3} = -\frac{y_n - y_{n-1}}{h_n} + y_n' \tag{2.39}$$

Diese beiden Gleichungen vervollständigen das System von Gleichungen (2.26). In Matrixform ergibt sich ein Gleichungssystem analog zu ii) mit λ_i, μ_i, d_i für $i = 1, 2, \dots, n-1$ gemäß (2.33) und zusätzlich

$$\lambda_0 := 1\,, \tag{2.40a}$$
$$\mu_0 := 1 - \lambda_0 = 0\,, \tag{2.40b}$$
$$d_0 := \frac{6}{h_n}\left(\frac{y_1 - y_0}{h_1} + y_0'\right), \tag{2.40c}$$

sowie

$$\lambda_n := 0\,, \tag{2.41a}$$
$$\mu_n := 1 - \lambda_n = 1\,, \tag{2.41b}$$
$$d_n := \frac{6}{h_n}\left(y_n' - \frac{y_n - y_{n-1}}{h_n}\right). \tag{2.41c}$$

Bei allen drei Randbedingungen haben die Elemente der Koeffizientenmatrix für alle $i = (0), 1, 2, \dots, n$ die Eigenschaften

$$\lambda_i \geq 0\,, \tag{2.42a}$$
$$\mu_i \geq 0\,, \tag{2.42b}$$
$$\lambda_i + \mu_i = 1\,. \tag{2.42c}$$

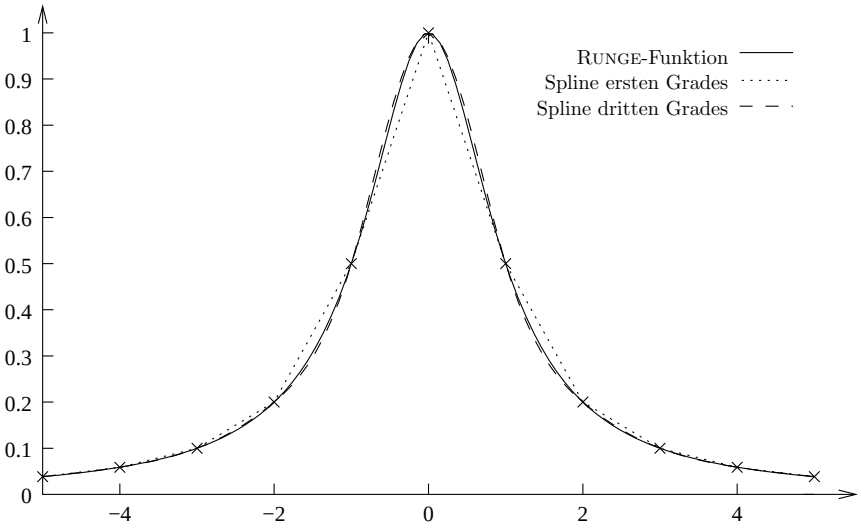


Abbildung 2.8: Interpolierender kubischer Spline mit vollständigen Randbedingungen

Aus diesen Eigenschaften folgt, dass die Koeffizientenmatrix (M_{ij}) strikt diagonaldominant, das heißt

$$\sum_{j,j\neq i} |M_{ij}| < |M_{ii}| \qquad \forall i\,,$$

und daher regulär ist. Somit existiert *genau ein* interpolierender Spline (jeweils für jede der drei Randbedingungen).

2.12 Beispiel (Fortsetzung zu Beispiel 2.10). Abbildung 2.8 zeigt die RUNGE-Funktion

$$y(x) := \frac{1}{1+x^2} \tag{2.43}$$

und die interpolierenden Splines ersten und dritten Grades dieser Funktion zu den Stützstellen $x_i = -5,\, 4,\, \ldots,\, 5$.

Bemerkungen.

a) Fehlerabschätzung für interpolierende Splines mit periodischen, natürlichen oder vollständigen Randbedingungen

$$\|y-s\|_\infty \leq \sqrt{2}h^{\frac{2}{3}}\|y''\|_2 \text{ mit } \; h = \max_{i=1,2,\ldots,n} |h_i|\,; \tag{2.44}$$

$$\|f\|_\infty := \max_{x\in[x_0,x_n]} |f(x)|, \qquad \|f\|_2 := \left(\int_{x_0}^{x_n} f^2(x)\mathrm{d}x\right)^{\frac{1}{2}}$$

b) Optimaleigenschaften interpolierender kubischer Splines mit periodischen, natürlichen oder vollständigen Randbedingungen.

Für jede interpolierende Funktion $f \in \mathcal{C}^2([x_0, x_n])$ gilt

$$\|s''\|_2 \leq \|f''\|_2. \tag{2.45}$$

Kubische Splines sind am "glattesten".

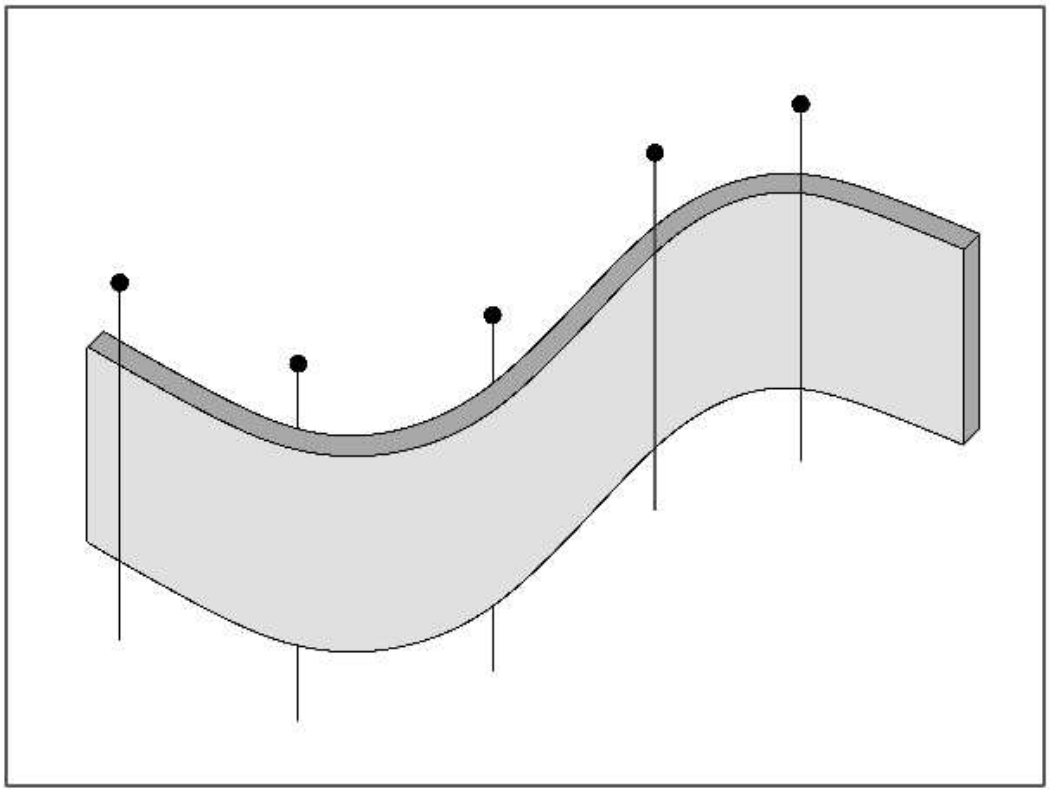


Abbildung 2.9: Straklatte (auf Zeichenbrett fixiert)

Hinweis. Der Begriff Spline kommt ursprünglich aus dem Schiffbau. Ein Spline (engl.) ist eine sogenannte Straklatte (dt.), also eine dünne biegsame Latte, die auf das Zeichenbrett aufgelegt wurde, um glatte Kurven zu erzeugen (Abbildung 2.9). Wenn $f(x)$ die Lattenform bezeichnet, dann ergibt sich für die Biegeenergie

$$E_b \sim \frac{1}{2}\int \frac{(f''(x))^2}{[1+(f'(x))^2]^{3/2}}\mathrm{d}x \approx \frac{1}{2}\int (f''(x))^2\mathrm{d}x$$

mit der Krümmung der Latte

$$\frac{f''(x)}{\sqrt{1+(f'(x))^2}} \approx f''(x)$$

für $|f'(x)| \ll 1$ (schwache Biegung).

Kapitel 3

Lösung linearer Gleichungssysteme, Matrixinversion

- Die numerische Bestimmung der Unbekannten in linearen Gleichungssystemen spielt eine zentrale Rolle in vielen Bereichen der Numerik.
- Viele Aufgaben der Physik führen auf dieses Problem, und es stellt sich auch oftmals im Rahmen anderer numerischer Aufgabenstellungen (d.h. die entsprechenden Algorithmen werden als *Hilfsmittel* verwendet).
- Das Hauptproblem ist die *endliche Genauigkeit* der Rechnung, die unter Umständen zu erhebliche Rundungsfehlern führen kann.

3.1 Vorbemerkungen und Einschränkungen

Lineare Gleichungssysteme haben folgende Struktur

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2N}x_N &= b_2 \\ \vdots \qquad\qquad & \vdots \\ a_{M1}x_1 + a_{M2}x_2 + a_{M3}x_3 + \cdots + a_{MN}x_N &= b_M. \end{aligned} \tag{3.1}$$

Damit haben wir also M Gleichungen mit M Konstanten $b_1, \ldots, b_M$ für die N Unbekannten $x_1, \ldots, x_N$. In Matrixschreibweise ergibt sich

$$A\boldsymbol{x} = \boldsymbol{b} \tag{3.2}$$

mit der Matrix $A \in \mathbb{R}^{M\times N}$, dem Unbekanntenvektor (Lösungsvektor) $\boldsymbol{x} \in \mathbb{R}^N$ und dem Konstantenvektor $\boldsymbol{b} \in \mathbb{R}^M$.

In diesem Kapitel werden wir uns ausschließlich mit dem Fall $M = N$ (quadratische Matrizen) beschäftigen, d.h. es gibt genauso viele Gleichungen wie Unbekannte. Des Weiteren setzen wir die Regularität der Matrix A voraus, d.h.

$$\det(A) \neq 0$$

Es liegt also weder eine Zeilenentartung noch eine Spaltenentartung vor.

Zeilenentartung: Eine oder mehrere Gleichungen $\in M$ sind Linearkombinationen der anderen (vgl. Beispiel 3.1).

Spaltenentartung: *Alle* Gleichungen enthalten bestimmte Variablen in der gleichen Linearkombination (vgl. Beispiel 3.2).

3.1 Beispiel (Zeilenentartung). Für die Matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 5 & 10 & 15 \end{pmatrix}$$

gilt für die Zeilen I, II, III

$$\begin{aligned} \text{II} &= 2 \cdot \text{I}, \\ \text{III} &= \text{I} + 2 \cdot \text{II} = 5 \cdot \text{I}. \end{aligned}$$

3.2 Beispiel (Spaltenentartung). Das Gleichungssystem enthalte überall $x_2 = 2 \cdot x_1$.

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 10 & 7 \\ 3 & 6 & 8 \end{pmatrix}.$$

Für die Spalten I und II gilt also

$$\text{II} = 2 \cdot \text{I}.$$

Für quadratische Matrizen impliziert Zeilenentartung Spaltenentartung und umgekehrt. Die Matrix aus Beispiel 3.2 ist spaltenentartet, ist sie also auch zeilenentartet?

Durch das GAUSSsche Eliminationsverfahren (→ Kapitel 3.4) formen wir die Matrix folgendermaßen um

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 10 & 7 \\ 3 & 6 & 8 \end{pmatrix} \xrightarrow{\text{II}'=\text{II}-5\cdot\text{I},\text{III}'=\text{III}-3\cdot\text{I}} \begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & -8 \\ 0 & 0 & -1 \end{pmatrix}$$

Es gilt also

$$\begin{aligned} \text{II}' &= 8 \cdot \text{III}' \\ \text{II} - 5 \cdot \text{I} &= 8(\text{III} - 3 \cdot \text{I}) \\ \text{II} &= 8 \cdot \text{III} - 19 \cdot \text{I} \qquad \square. \end{aligned}$$

Zeile II ist damit eine Linearkombination von III und I, d.h. es liegt auch Zeilenentartung vor.

Numerisches Problem.

- Auf Grund von Rundungsfehlern können Gleichungen, die "fast" Linearkombinationen anderer sind, zu "echten" Linearkombinationen werden.
- Akkumulierte Rundungsfehler können zu *völlig falschen* Lösungen führen.

  Daumenregel: Für die bei der Lösung von Gleichungssystemen mit dem Computer benötige Genauigkeit kann man folgende Regel angeben:

  N $\leq$ 50: "single precision" Es muss mit 4-`byte REAL` Zahlen, also 6-stelliger Genauigkeit gerechnet werden.

  N einige 100: "double precision" Es muss mit 8-`byte REAL` Zahlen, also 15-stelliger Genauigkeit gerechnet werden.

  Es existieren aber auch Gleichungssysteme mit $N \lesssim 10$, die stark problematisch sind!
- Für die Behandlung von nichtquadratischen Matrizen (mehr oder weniger Gleichungen als Unbekannte, $M \neq N$) bzw. singulären Matrizen wird die sogenannte *singular value decomposition* (SVD, Singulärwertzerlegung) verwendet[1].

[1] Ein schneller Einstieg findet sich z.B. in "Numerical Recipes", Kap. 2.6

3.2 Zusammenhang mit Matrixinversion

Gegeben sei eine reguläre Matrix $A \in \mathbb{R}^{N\times N}$. Gesucht wird A^{-1}. Wir nehmen an, dass wir ein Verfahren zur Lösung des linearen Gleichungssystems

$$A \cdot \boldsymbol{x} = \boldsymbol{b}$$

für beliebige $\boldsymbol{b}$ kennen. Damit können wir dann auch die Matrix A invertieren.

Schritt 1. Löse

$$A \cdot \boldsymbol{x}_1 = \boldsymbol{e}_1 \tag{3.3}$$

mit dem Einheitsvektor $\boldsymbol{e}_1 = (1, 0, 0, \ldots, 0)^\intercal$ der Länge N. $\boldsymbol{x}_1$ lässt sich auch schreiben als

$$\boldsymbol{x}_1 = A^{-1}\boldsymbol{e}_1 = \begin{pmatrix} (a^{-1})_{11} & (a^{-1})_{12} & \ldots & (a^{-1})_{1N} \\ (a^{-1})_{21} & (a^{-1})_{22} & & \\ \vdots & & \ddots & \vdots \\ (a^{-1})_{N1} & & \ldots & (a^{-1})_{NN} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} (a^{-1})_{11} \\ (a^{-1})_{21} \\ \vdots \\ (a^{-1})_{N1} \end{pmatrix},$$

d.h. unsere Lösung des Gleichungssystems (3.3), $\boldsymbol{x}_1$, ist gerade die *erste Spalte* von A^{-1}.

Schritt 2. Löse

$$A \cdot \boldsymbol{x}_2 = \boldsymbol{e}_2 \tag{3.4}$$

mit dem Einheitsvektor $\boldsymbol{e}_2 = (0, 1, 0, \ldots, 0)^\intercal$ der Länge N.

$$\Longrightarrow \boldsymbol{x}_2 = A^{-1}\boldsymbol{e}_2 = \begin{pmatrix} & & \\ & (a^{-1})_{ij} & \\ & & \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} (a^{-1})_{12} \\ (a^{-1})_{22} \\ \vdots \\ (a^{-1})_{N2} \end{pmatrix}.$$

Also ist $\boldsymbol{x}_2$ die *zweite Spalte* von A^{-1}.

Schritt 3 ... N. Löse

$$A \cdot \boldsymbol{x}_i = \boldsymbol{e}_i \qquad (i = 3, \ldots N)$$

Die Lösung $\boldsymbol{x}_i$ ist die i-te Spalte von A^{-1}.

Zusammenfassung. Die N-malige Lösung der Gleichungssysteme

$$A \cdot \boldsymbol{x}_i = \boldsymbol{e}_i \qquad (i = 1, 2, \ldots, N).$$

mit dem i-ten Einheitsvektor $\boldsymbol{e}_i$ ergibt zusammengefasst die Matrix A^{-1}, wobei die Spalten von A^{-1} durch die Lösungen $\boldsymbol{x}_i$ gegeben sind:

$$A^{-1} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N). \tag{3.5}$$

3.3 Überblick und Einteilung diverser Matrizen, dazugehörige Lösungsverfahren

Matrizen lassen sich insbesondere durch die Struktur ihrer Besetzung unterscheiden, also dadurch, an welcher Stelle sich von Null verschiedene Matrixelemente befinden. Man unterscheidet unter anderem folgende Arten von Matrizen:

"Volle" Matrizen sind, wie der Name schon sagt, bis auf geringe Ausnahmen voll besetzt. Zur Lösung von entsprechenden linearen Gleichungssystemen verwendet man das Gausssche Eliminationsverfahren (Kapitel 3.4), das Gauss-Jordan-Verfahren und vor allen das sog. LU Verfahren (Kapitel 3.5).

Gering besetzte ("sparse") Matrizen enthalten überwiegend Matrixelemente, die gleich Null sind. In dieser Klasse unterscheidet man entsprechend ihrer Struktur:

Bandmatrizen sind entlang der Hauptdiagonalen und eventuell einer oder mehrerer Nebendiagonalen besetzt. Unterschieden wird hier häufig noch zwischen *bidiagonalen* und *tridiagonalen* Matrizen, die in Abschnitt 3.9 behandelt werden (vgl. Abbildung 3.1).

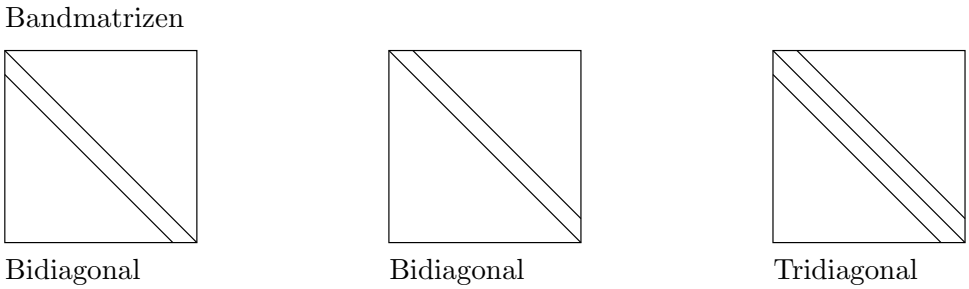


Abbildung 3.1: Mögliche Besetzungen von Bandmatrizen

Block-Matrizen werden in *block-diagonale* und *block-tridiagonale* Matrizen unterteilt (Abb. 3.2),

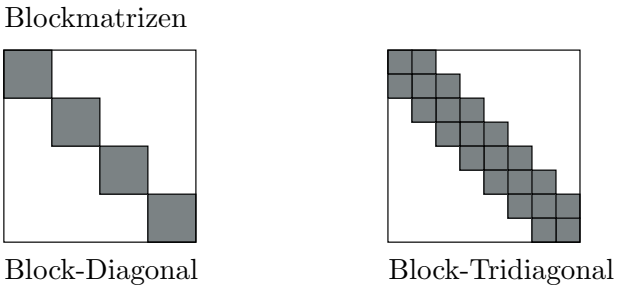


Abbildung 3.2: Mögliche Besetzungen für Blockmatrizen

wobei

Geränderte Matrizen *zusätzlich* an einem oder mehreren Rändern besetzt sind.

Zur numerischen Behandlung von gering besetzen Matrizen bestimmter Struktur verweisen wir auf das auf den meisten Rechnern implementierte Programmpaket Lapack (linear algebra package) bzw. für allgemeine, gering besetzte Matrizen die sog. *bikonjugierte Gradientenmethode* (Literatur).

Spezielle Matrizen sind u.a.

symmetrische, positv definite Matrizen, deren zugehörige Gleichungssysteme mit dem CHOLESKY-Verfahren (Kapitel 3.10) gelöst werden können oder

Vandermonde, Toeplitz-Matrizen, ... auf die wir hier nicht näher eingehen werden.

3.4 Die GAUSS-Elimination

Die Idee, die hinter der GAUSS-Elimination steht, beruht darauf, dass die Lösung eines Gleichungssystems unverändert bleibt, wenn man

a) Gleichungen und entsprechende Konstanten vertauscht (zeilenweise)

b) Linearkombinationen mit anderen Zeilen bildet (incl. des analogen Vorgehens bzgl. der Konstanten).

$\Longrightarrow$ Bringe das Gleichungssystem

$$A \cdot \boldsymbol{x} = \boldsymbol{b}$$

über die obigen Operationen a) und b) auf *obere Dreiecksform*. Danach lassen sich die Lösungen $\boldsymbol{x}_N$, $\boldsymbol{x}_{N-1}$, ..., $\boldsymbol{x}_1$ sukzessive, beginnend mit der letzten Zeile, berechnen!

3.3 Beispiel ($N = 3$).

$$\begin{pmatrix} 1 & 5 & 7 \\ 3 & 0 & 4 \\ 7 & 5 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\left(\begin{array}{ccc|c} 1 & 5 & 7 & 1 \\ 3 & 0 & 4 & 2 \\ 7 & 5 & 5 & 3 \end{array}\right) \xrightarrow{\text{II}-3\cdot\text{I},\text{III}-7\cdot\text{I}} \left(\begin{array}{ccc|c} 1 & 5 & 7 & 1 \\ 0 & -15 & -17 & -1 \\ 0 & -30 & -44 & -4 \end{array}\right)$$

$$\xrightarrow{\text{III}'-2\cdot\text{II}'} \left(\begin{array}{ccc|c} 1 & 5 & 7 & 1 \\ 0 & -15 & -17 & -1 \\ 0 & 0 & -10 & -2 \end{array}\right)$$

Damit haben wir eine obere Dreiecksmatrix erhalten, und in Komponentenschreibweise lautet das entsprechende Gleichungssystem

$$\begin{aligned} x_1 + 5x_2 + 7x_3 &= 1 \\ -15x_2 - 17x_3 &= -1 \\ -10x_3 &= -2. \end{aligned}$$

Wir erhalten damit die gleichen Lösungen wie mit dem Ausgangssystem (sukzessive Bildung von Linearkombinationen), die sich leicht berechnen lassen.

$\Rightarrow$ Aus III$''$

$$x_3 = \frac{2}{10} = \frac{1}{5}$$

in Zeile II$'$

$$-15x_2 - \frac{17}{5} = -1$$

$\Rightarrow$ Aus II$'$

$$x_2 = -\frac{1}{15}\left(\frac{12}{5}\right) = -\frac{4}{25}$$

in Zeile I

$$x_1 - \frac{4}{5} + \frac{7}{5} = 1$$

$\Rightarrow$ Aus I

$$x_1 = \frac{2}{5}$$

$\Rightarrow$

$$\boldsymbol{x} = \left(\frac{2}{5}, -\frac{4}{25}, \frac{2}{5}\right)^{\intercal}$$

Daraus lässt sich folgender simpler *Algorithmus* konstruieren (im Weiteren anhand eines Beispiels mit $N = 4$)

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & b_2 \\ a_{31} & a_{32} & a_{33} & a_{34} & b_3 \\ a_{41} & a_{42} & a_{43} & a_{44} & b_4 \end{array}\right)$$

Schritt 1. Setze alle Elemente a_{i1} durch Bildung von Linearkombinationen mit Zeile I auf 0, $\forall i = 2, \ldots N$. In unserem Beispiel müssen also a_{21}, a_{31} und a_{41} auf 0 gebracht werden.

$$\left.\begin{array}{rcl} \mathrm{II}^{(1)} & = & \mathrm{II} - a_{21}\frac{1}{a_{11}}\mathrm{I} \\ \mathrm{III}^{(1)} & = & \mathrm{III} - a_{31}\frac{1}{a_{11}}\mathrm{I} \\ \mathrm{IV}^{(1)} & = & \mathrm{IV} - a_{41}\frac{1}{a_{11}}\mathrm{I} \end{array}\right\} \tag{3.6}$$

Hier wird die Annahme gemacht, dass $a_{11} \neq 0$ gilt. Ansonsten wird Zeile I mit einer Zeile vertauscht, in der $a_{i1} \neq 0$ ist. Es muss eine solche Zeile geben, da A sonst nicht regulär wäre! Gleichung (3.6) lautet in Komponentenschreibweise

$$\begin{aligned} a_{ik}^{(1)} &= a_{ik} - \frac{a_{i1}}{a_{11}} \cdot a_{1k} && i = 2, \ldots N \quad k = 1, \ldots, N \\ b_i^{(1)} &= b_i - \frac{a_{i1}}{a_{11}} \cdot b_1 && i = 2, \ldots N \end{aligned} \tag{3.7}$$

Gleichung (3.7) impliziert für $k = 1$ das erwünschte Nullsetzen von $a_{ik}^{(1)}$, $i = 2, \ldots, N$, denn

$$a_{i1}^{(1)} = a_{i1} - \frac{a_{i1}}{a_{11}} \cdot a_{11} = 0$$

Mit Schritt 1 haben wir also Folgendes erreicht

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} & b_3^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} & b_4^{(1)} \end{array}\right),$$

und die Operation (3.7) muss nur für $k = 2, \ldots, N$ explizit durchgeführt werden.

$$\begin{aligned} a_{ik}^{(1)} &= a_{ik} - l_{i1}a_{1k} && i, k = 2, \ldots N \\ b_i^{(1)} &= b_i - l_{i1}b_1 && i = 2, \ldots N \end{aligned} \tag{3.8}$$

mit $l_{i1} = \dfrac{a_{i1}}{a_{11}}$

Schritt 2. Es werden analog dem ersten Schritt alle $a_{i2}^{(1)}$, $i = 3, \ldots, N$ auf 0 gebracht. Wir nehmen wieder an, dass für die erzeugten $a_{22}^{(1)} \neq 0$ gilt. Die Form

$$\Rightarrow \left(\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} & b_3^{(2)} \\ 0 & 0 & a_{43}^{(2)} & a_{44}^{(2)} & b_4^{(2)} \end{array}\right)$$

erhalten wir mit

$$\begin{aligned} a_{ik}^{(2)} &= a_{ik}^{(1)} - l_{i2} \cdot a_{2k}^{(1)} && i, k = 3, \ldots N \\ b_i^{(2)} &= b_i^{(1)} - l_{i2} \cdot b_2^{(1)} && i = 3, \ldots, N \end{aligned} \tag{3.9}$$

und $l_{i2} = \dfrac{a_{i2}^{(1)}}{a_{22}^{(1)}}$.

Man beachte, dass Schritt 2 nur für die Spalten $k = 3, \ldots, N$ explizit durchgeführt werden muss, da laut Konstruktion $a_{i2}^{(2)} = 0 \; \forall i = 3, \ldots, N$ gilt.

Schritt 3. Bei $N = 4$ ist dies der letzte Schritt. Wir bringen das Gleichungssystem auf die (hier finale) Gestalt

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} & b_3^{(2)} \\ 0 & 0 & 0 & a_{44}^{(3)} & b_4^{(3)} \end{array}\right),$$

indem wir die Umformungen

$$\begin{aligned} a_{ik}^{(3)} &= a_{ik}^{(2)} - l_{i3} \cdot a_{3k}^{(2)} && i,k = 4, \ldots N \\ b_i^{(3)} &= b_i^{(2)} - l_{i3} \cdot b_3^{(2)} && i = 4, \ldots, N \end{aligned} \tag{3.10}$$

mit $l_{i3} = \dfrac{a_{i3}^{(2)}}{a_{33}^{(2)}}$ verwenden. Wieder wird vorausgesetzt, dass $a_{33}^{(2)} \neq 0$ gilt. Mit der

Umbenennung

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} & b_3^{(2)} \\ 0 & 0 & 0 & a_{44}^{(3)} & b_4^{(3)} \end{array}\right) \longrightarrow \left(\begin{array}{cccc|c} u_{11} & u_{12} & u_{13} & u_{14} & y_1 \\ 0 & u_{22} & u_{23} & u_{24} & y_2 \\ 0 & 0 & u_{33} & u_{34} & y_3 \\ 0 & 0 & 0 & u_{44} & y_4 \end{array}\right) \tag{3.11}$$

erhalten wir den Gauss-Algorithmus in allgemeiner Form

$$a_{ik}^{(0)} = a_{ik}, \qquad i,k = 1, \ldots N \tag{3.12a}$$

$$b_i^{(0)} = b_i, \qquad i = 1, \ldots, N \tag{3.12b}$$

$$a_{ik}^{(n)} = a_{ik}^{(n-1)} - l_{in} a_{nk}^{(n-1)}, \qquad i,k = n+1, \ldots, N \tag{3.12c}$$

$$b_i^{(n)} = b_i^{(n-1)} - l_{in} b_n^{(n-1)}, \qquad i = n+1, \ldots, N \tag{3.12d}$$

$$l_{in} = \frac{a_{in}^{(n-1)}}{a_{nn}^{(n-1)}}. \tag{3.12e}$$

In jedem Schritt wird vorausgesetzt, dass $a_{nn}^{(n-1)} \neq 0$ gilt! Die Schritte, die in diesem Schema ausgeführt werden müssen, sind $n = 1, \ldots, N-1$. Es ergeben sich dann die *Endgleichungen*

$$u_{ik} = a_{ik}^{(i-1)}, \qquad i = 1, \ldots, N \quad k = i, \ldots, N \tag{3.13a}$$

$$y_i = b_i^{(i-1)}, \qquad i = 1, \ldots, N \tag{3.13b}$$

Bei dem vorliegenden Algorithmus ist zu beachten:

- Zeile i von u_{ik} hängt Schritt $i-1$ ab
- die erste Zeile bleibt unverändert
- die letzte Zeile enthält nur ein Element

Mit dem GAUSS-Algorithmus haben wir also das Gleichungssystem

$$A \cdot \boldsymbol{x} = \boldsymbol{b}$$

in das dazu äquivalente System

$$U \cdot \boldsymbol{x} = \boldsymbol{y}$$

mit oberer Δ-Matrix U umgeformt. Daraus lassen sich dann die Unbekannten x_N, x_{N-1}, ..., x_1 durch *Rückwärtseinsetzung* mühelos berechnen.

$$\begin{aligned} x_N &= \frac{y_N}{u_{NN}} \\ x_{N-1} &= \frac{y_{N-1} - u_{N-1,N}x_N}{u_{N-1,N-1}} \\ &\vdots \end{aligned}$$

Allgemein finden wir die Lösungen x_i mit

$$x_i = \frac{y_i - \sum_{k=i+1}^{N} u_{ik}x_k}{u_{ii}} \qquad i = N, \ldots, 1. \tag{3.14}$$

In unserem Fall $N = 4$ lauten die Lösungen damit

$$\begin{aligned} x_4 &= \frac{y_4}{u_{44}} \\ x_3 &= \frac{y_3 - u_{34}x_4}{u_{33}} \\ x_2 &= \frac{y_2 - u_{24}x_4 - u_{23}x_3}{u_{22}} \\ x_1 &= \frac{y_1 - u_{14}x_4 - u_{13}x_3 - u_{12}x_2}{u_{11}} \end{aligned}$$

Die Divisionen sind allesamt möglich, da laut Voraussetzung $u_{ii} = a_{ii}^{(i-1)} \neq 0$ gilt!

Benötigte Rechenzeit. Wir berücksichtigen hier nur die sog. *wesentliche* Operationen, d.h. Multiplikation und Division (Addition oder Subtraktion fallen demgegenüber nicht ins Gewicht).

In jedem Schritt n werden

- $N - n$ Quotienten l_{in} (einer pro Zeile) berechnet (jeweils eine Division) und
- $(N - n)^2$ Berechnungen der $a_{ik}^{(n)}$ (jeweils eine Multiplikation)
- $N - n$ Berechnungen der $b_i^{(n)}$ (jeweils eine Multiplikation)

durchgeführt. Insgesamt sind dann für die Schritte $n = 1, \ldots, N-1$

$$2 \cdot \sum_{n=1}^{N-1} (N-n) + \sum_{n=1}^{N-1} (N-n)^2$$

wesentliche Operationen notwendig. Umformung des oberen Ausdrucks liefert

$$\begin{aligned} &= 2\,(N-1+N-2+\cdots+1) + \left((N-1)^2 + (N-2)^2 + \cdots + 1^2\right) \\ &= 2\,\frac{(N-1)N}{2} + \frac{(N-1)N(2N-1)}{6} \\ &= 2\,\frac{1}{2}(N^2 - N) + \frac{1}{6}\left(2N^3 - 3N^2 + N\right) \\ &= \underbrace{\frac{1}{2}\left(N^2 - N\right)}_{\text{Transformation von } \boldsymbol{b}} + \underbrace{\frac{1}{3}\left(N^3 - N\right)}_{\text{Transformation der Matrix}} \end{aligned}$$

Für die *Rückwärtseinsetzung* werden dann für jedes i noch $N-i$ Multiplikationen und eine Division durchgeführt, insgesamt also

$$\sum_{i=1}^{N}(N+1-i) = \frac{1}{2}\left(N^2+N\right)$$

wesentliche Operationen.

Zur Durchführung des ganzen GAUSS-Algorithmus für *ein* $\boldsymbol{b}$ werden somit

$$Z_{\text{GAUSS}} = \frac{1}{3}\left(N^3-N\right) + \frac{1}{2}\left(N^2-N\right) + \frac{1}{2}\left(N^2+N\right) = \frac{1}{3}\left(N^3-N\right) + N^2$$

wesentliche Operationen ausgeführt.
Falls man das Gleichungssystem für mehrere $\boldsymbol{b}$ gleichzeitig lösen will (z.B. im Rahmen einer Inversionsaufgabe), müssen alle $\boldsymbol{b}$ *im Vorhinein* bekannt sein, da die Transformation $\boldsymbol{b} \to \boldsymbol{y}$ die jeweiligen Quotienten l_{ij} benötigt. Damit resultiert für k gegebene $\boldsymbol{b}$ eine Rechenzeit von

$$Z_{\text{GAUSS}} = \frac{1}{3}\left(N^3-N\right) + kN^2. \tag{3.15}$$

Problem: Falls man für ein $\boldsymbol{b}_1$ das Gleichungssystem gelöst hat und *danach* die Lösung für ein anderes $\boldsymbol{b}_2$ sucht, muss der gesamte Algorithmus von vorne "gefahren" werden und man braucht wiederum $\mathcal{O}(\frac{1}{3}N^3)$ Operationen!

Frage: Lässt sich eine Möglichkeit finden, das Verfahren so zu modifizieren, dass man nach erfolgter Lösung des Gleichungssystems für $\boldsymbol{b}_1$ fast ohne zusätzlichen Aufwand die Lösung für ein anderes $\boldsymbol{b}_2$ erzielen kann?

Antwort: Ja, mit der sogenannten LU-Zerlegung, die wir im Folgenden behandeln werden.

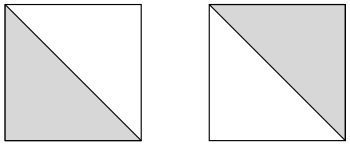
3.5 Von der GAUSS-Elimination zur LU-Zerlegung

3.5.1 Das Prinzip

Bisher hatten wir aus der ursprünglichen Matrix A eine obere Δ-Matrix erzeugt (inkl. der analogen Operationen bzgl. des $\boldsymbol{b}$-Vektors).
Jetzt wollen wir (über den zu besprechenden Algorithmus) die Matrix A,

$$A = L \cdot U \tag{3.16}$$

dergestalt faktorisieren, dass U eine obere (engl. upper) und L eine untere (engl. lower) Δ-Matrix ist. Man beachte, dass bei beiden Matrizen die Diagonale besetzt ist! Dann kann man schreiben



$L \qquad \cdot \qquad U$

$$\begin{aligned} A \cdot \boldsymbol{x} = (L \cdot U) \cdot \boldsymbol{x} &= \boldsymbol{b} \\ L(U \cdot \boldsymbol{x}) &= \boldsymbol{b} \qquad (3.17a) \\ U \cdot \boldsymbol{x} &= \boldsymbol{y}. \qquad (3.17b) \end{aligned}$$

Die gesuchte Lösung $\boldsymbol{x}$ ergibt sich damit in *zwei* Schritten.

Schritt 1. Löse

$$L \cdot \boldsymbol{y} = \boldsymbol{b}$$

mit der unteren Δ-Matrix L. Das lässt sich ohne Weiteres durch *Vorwärtseinsetzen* erzielen.

$$\begin{aligned} l_{11}y_1 &= b_1 \\ l_{21}y_1 + l_{22}y_2 &= b_2 \\ l_{31}y_1 + l_{32}y_2 + l_{33}y_3 &= b_3 \\ \vdots & \quad \vdots \end{aligned}$$

(Vorgehen analog der Rückwärtseinsetzung, aber diesmal von oben nach unten)

$\Rightarrow \boldsymbol{y}$

Schritt 2. Löse

$$U \cdot \boldsymbol{x} = \boldsymbol{y}$$

mit $\boldsymbol{y}$ aus Schritt 1 durch *Rückwärtseinsetzen*

$\Rightarrow \boldsymbol{x}$

Falls $A = L \cdot U$ zerlegt wurde, lässt sich dieses Verfahren für beliebige $\boldsymbol{b}$ durchführen.

Rechenzeit pro $\boldsymbol{b}$ (ohne Zerlegung).

Da die Vorwärtseinsetzung in etwa die gleiche Rechenzeit wie die Rückwärtseinsetzung in Anspruch nimmt, sind ca.

$$2 \cdot \frac{1}{2}\left(N(N+1)\right) = N^2 + N$$

wesentliche Operationen durchzuführen.

3.5.2 Wie lässt sich die LU-Zerlegung bewerkstelligen?

Wir hatten mit der GAUSS-Elimination Folgendes erreicht

$$A \cdot \boldsymbol{x} = \boldsymbol{b} \quad \longrightarrow \quad U \cdot \boldsymbol{x} = \boldsymbol{y},$$

wobei U eine obere Δ-Matrix war.

Beachte: Im Laufe des Algorithmus ($N-1$ Schritte) wird die Ausgangsmatrix A sukzessive durch U ersetzt. Wie es sich bald herausstellen wird, ist es zweckmäßig, die "Nullen" unterhalb der Diagonalen durch die jeweiligen Quotienten l_{ik} zu ersetzen, d.h. final ist A ersetzt durch (für $N = 4$)

$$\left(\begin{array}{cccc|c} u_{11} & u_{12} & u_{13} & u_{14} & y_1 \\ l_{21} & u_{22} & u_{23} & u_{24} & y_2 \\ l_{31} & l_{32} & u_{33} & u_{34} & y_3 \\ l_{41} & l_{42} & l_{43} & u_{44} & y_4 \end{array}\right). \tag{3.18}$$

Die l_{21}, l_{31}, l_{41} kommen jeweils aus dem ersten Schritt, die l_{32}, l_{42} aus dem zweiten und l_{43} aus dem dritten. Wir hatten bereits in (3.12a) und (3.13a)

$$\begin{aligned} u_{ik} = a_{ik}^{(i-1)} &= a_{ik}^{(i-2)} - l_{i,i-1}a_{i-1,k}^{(i-2)} \\ &= a_{ik}^{(i-3)} - l_{i,i-2}a_{i-2,k}^{(i-3)} - l_{i,i-1}a_{i-1,k}^{(i-2)} \\ &= \dots \\ &= a_{ik}^{(0)} - l_{i1}a_{1k}^{(0)} - l_{i2}a_{2k}^{(1)} - l_{i3}a_{3k}^{(2)} - \dots - l_{i,i-1}a_{i-1,k}^{(i-1)} \end{aligned}$$

$$\Rightarrow u_{ik} = a_{ik} - l_{i1}u_{1k} - l_{i2}u_{2k} - \dots - l_{i,i-1}u_{i-1,k} \qquad i \geq 2, \quad k \geq i \qquad \text{"oben"} \tag{3.19}$$

Der Vergleich mit dem Schema (3.18) zeigt also, dass die Matrixelemente u_{ik} gebildet werden aus

- dem Originalelement a_{ik}
- den Quotienten l_{ij}, $j = 1, \ldots, i-1$ der *Zeile* i
- den (schon berechneten) u_{jk}, $j = 1, \ldots, i-1$ der *Spalte* k

3.4 Beispiel (Berechnung zweier u_{jk}).

$$\begin{aligned} u_{44} &= a_{44} - (l_{41} \quad l_{42} \quad l_{43}) \cdot \begin{pmatrix} u_{14} \\ u_{24} \\ u_{34} \end{pmatrix} \\ u_{24} &= a_{24} - (l_{21}) \cdot (u_{14}) \end{aligned}$$

Nach Umformung von (3.19) ergibt sich

$$\text{I}: \qquad a_{ik} = \sum_{j=1}^{i-1} l_{ij} u_{jk} + u_{ik}, \qquad k \geq i \geq 1 \tag{3.20}$$

wobei diese Gleichung auch für $i = 1$ gültig ist!

Wie setzen sich nun die Quotienten l_{ik}, $k \geq 1$, $i > k$ zusammen? l_{ik} wird aus $a_{ik}^{(k-1)}$ im k-tem Schritt ermittelt durch

$$\begin{aligned} l_{ik} &= \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} = \qquad \text{vgl. (3.12a)} \\ &= \frac{a_{ik} - l_{i1} a_{1k}^{(0)} - l_{i2} a_{2k}^{(1)} - \ldots - l_{i,k-1} a_{k-1,k}^{(k-2)}}{a_{kk}^{(k-1)}} \\ &= \frac{a_{ik} - l_{i1} u_{1k} - l_{i2} u_{2k} - \ldots - l_{i,k-1} u_{k-1,k}}{u_{kk}}. \end{aligned} \tag{3.21}$$

Der Vergleich mit dem Schema (3.18) ergibt, dass die Quotienten l_{ik} gebildet werden aus

- den Originalelement a_{ik}
- den Quotienten l_{ij}, $j = 1, \ldots, k-1$ der *Zeile* i (vorhergehende Spalten)
- den (schon berechneten) u_{jk}, $j = 1, \ldots, k-1$ der *Spalte* k (vorhergehende Zeilen)
- dem Diagonalelement u_{kk}

3.5 Beispiel (Berechnung dreier l_{ij}).

$$\begin{aligned} l_{21} &= \frac{a_{21}}{a_{11}} \\ l_{41} &= \frac{a_{41}}{a_{11}} \\ l_{43} &= \frac{a_{43} - (l_{41} \quad l_{42}) \cdot \begin{pmatrix} u_{13} \\ u_{23} \end{pmatrix}}{u_{33}} \end{aligned}$$

Die Umformung von Gleichung (3.21) ergibt

$$\text{II}: \qquad a_{ik} = \sum_{j=1}^{k} l_{ij} u_{jk}, \qquad i \geq k \geq 1 \tag{3.22}$$

Man sieht, dass in Gleichung I (3.20) die a_{ik} eine obere Δ-Matrix (mit Diagonale), in Gleichung II (3.22) eine untere Δ-Matrix (ohne Diagonale) bilden. Beide Gleichungen erinnern an eine Matrixmultiplikation, so dass wir folgendermassen fortfahren. Wenn wir die Matrix L als

$$L := \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix},$$

also mit zusätzlichen "1" auf der Diagonalen, definieren, dann gilt zusammen mit den Gleichungen I, II (3.20, 3.22)

$$A = L \cdot U,$$

d.h. L und U bilden die gewünschte Zerlegung in Δ-Matrizen.

3.6 Beispiel ("Beweis" für $N = 4$).

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} =$$

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} & l_{21}u_{13} + u_{23} & l_{21}u_{14} + u_{24} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + u_{33} & l_{31}u_{14} + l_{32}u_{24} + u_{34} \\ l_{41}u_{11} & l_{41}u_{12} + l_{42}u_{22} & l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{44} \end{pmatrix}$$

Für den unteren Teil der Matrix gilt damit

$$\sum_{j=1}^{k} l_{ij}u_{jk} \overset{(3.22)}{=} a_{ik} \qquad (i > k) \qquad \text{(Gl. II)}$$

und für den oberen Teil der Matrix (incl. der Diagonalen)

$$\sum_{j=1}^{i-1} l_{ij}u_{jk} + u_{ik} \overset{(3.20)}{=} a_{ik} \qquad (k \geq i) \qquad \text{(Gl. I)}.$$

Fazit. Die GAUSSsche Elimination (mit $a_{ii}^{(i-1)} \neq 0$) ergibt die gewünschte LU-Zerlegung "automatisch", d.h. ohne zusätzlichen Rechenaufwand.

Vorgehensweise.

1. $A \rightarrow$ GAUSS-Elimination, speichere die Quotienten in unterer Δ-Matrix ab

$$\rightarrow A = L \cdot U$$

2. Durch Vorwärtseinsetzen $L \cdot y = b$ lösen (beachte dass $l_{ii} = 1$)

$$\rightarrow y_i = b_i - \sum_{j=1}^{i-1} l_{ij}y_j$$

3. Durch Rückwärtseinsetzen $U \cdot x = y$ lösen

$$\rightarrow x_i = \frac{y_i - \sum_{j=i+1}^{N} u_{ij}x_j}{u_{ii}}$$

Bisherige Voraussetzung war:

$$a_{ii}^{(i-1)} \neq 0$$

Diese anscheinend äußerst strikte Voraussetzung stellt jedoch, wie sich herausstellt, kein Problem dar, da sich Folgendes zeigen lässt:

- Für jede reguläre Matrix A existiert *vor* dem k-ten Eliminationsschritt stets eine Zeilenpermutation derart, dass das k-te Diagonalelement von Null verschieden ist.

- dies gilt z.B. für a_{11} a priori, da zumindest ein $a_{i1} \neq 0$ sein muss, damit die Matrix A regulär ist.

Wir denken uns jetzt den notwendigen Zeilenaustausch schon vor der Zerlegung durchgeführt. Die Buchführung über den Zeilenaustausch ist dabei unerlässlich, da bei der Vorwärtseinsetzung $\boldsymbol{b}$ analog behandelt werden muss. Dazu benutzen wir die Permutationsmatrix P. Die resultierende Matrix mit vertauschten Zeilen lautet dann

$$P \cdot A$$

3.7 Beispiel (Permutation einer Matrix).

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$$

Man beachte, dass die Permutationsmatrix P in jeder Zeile *und* in jeder Spalte genau ein Element mit dem Wert 1 hat, die anderen Einträge sind Null.

$$\Rightarrow \det(P) = \pm 1$$

da P durch Zeilenvertauschung aus der Einheitsmatrix gebildet wird.

3.8 Satz. *Zu jeder regulären Matrix A existiert eine Permutationsmatrix P, so dass gilt*

$$P \cdot A = L \cdot U \tag{3.23}$$

Für die Lösung des Problems $A \cdot \boldsymbol{x} = \boldsymbol{b}$ ergibt sich also folgendes Vorgehen:

1. unabhängig von $\boldsymbol{b}$
$$P \cdot A = L \cdot U$$

2. Dieser Schritt wird geändert, wir haben $(P \cdot A) \cdot \boldsymbol{x} = P \cdot \boldsymbol{b}$ und damit
$$L \cdot \boldsymbol{y} = P \cdot \boldsymbol{b}$$

3. wie vorher
$$U \cdot \boldsymbol{x} = \boldsymbol{y}$$

Als "Abfallprodukt" dieser Rechnung ergibt sich außerdem noch die Determinante von A,

$$\begin{aligned} |A| &= (-1)^V \cdot |L| \cdot |U| \\ &= (-1)^V \cdot 1 \cdot \prod_{i=1}^{N} u_{ii} \end{aligned} \tag{3.24}$$

wenn V die Anzahl der notwendigen Vertauschungen war.
Man erinnere sich, dass die insgesamt benötigte Rechenzeit für das GAUSSsche Eliminatonsverfahren (bei k Vektoren $\boldsymbol{b}$) mit

$$Z_{\text{GAUSS}} = \frac{1}{3}\left(N^3 - N\right) + kN^2 \stackrel{!}{=} Z_{\text{LU}}$$

skalierte. Dies entspricht damit genau der Rechenzeit, die zur Lösung des gleichen Problemes mittels LU-Zerlegung erforderlich ist, da für das Vorwärtseinsetzen $\frac{1}{2}(N^2 - N)$ Operationen benötigt werden. Damit gilt also:
Durch die LU-Zerlegung wird kein Zeitgewinn, wohl aber ein Strategiegewinn erzielt!

3.6 Pivotisierung

Das Matrixelement, durch das geteilt wird ($a_{kk}^{(k-1)}$), wird Pivotelement (Drehpunkt, Türangel) genannt

- Der obige Satz bezüglich $P \cdot A = L \cdot U$ gewährleistet die Existenz eines von Null verschiedenen Pivotelements.
- Die konkrete Wahl (d.h. welche Zeilen werden vertauscht) bleibt aber offen.
- Diese Wahl ist jedoch *entscheidend* für die *Genauigkeit*!

3.6.1 Kolonnenmaximumsstrategie

Zunächst wollen wir die sog. Kolonnenmaximumsstrategie zur Wahl des "besten" Pivotelements betrachen. Unter den zur Verfügung stehenden Pivotelementen wird das betragsmäßig größte gewählt, da das Teilen durch kleine Zahlen zu Problemen führen kann.

a) Welche Zahlen stehen zur Verfügung?

Berechnung von $l_{ik} = \dfrac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$, $i > k$.

Vor dem k-ten Eliminationsschritt stehen alle Zeilen $i \geq k$ zur Verfügung, um $|a_{kk}^{(k-1)}|$ zu maximieren (vgl. Beispiel 3.9).

3.9 Beispiel ($N = 4$, Matrix vor dem zweiten Eliminationsschritt).

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & \to a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & \to a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & \to a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{pmatrix}$$

Die Zeilen werden so vertauscht, dass $a_{22}^{(1)} = \max_{i \geq 2} |a_{i2}^{(1)}|$.

b) Allgemein wird der Index p so bestimmt, dass vor dem Schritt k gilt

$$\max_{i \geq k} \left| a_{ik}^{(k-1)} \right| = \left| a_{pk}^{(k-1)} \right|, \tag{3.25}$$

falls $p \neq k$, wird die Zeile p mit Zeile k vertauscht. Mit dieser Wahl ist gewährleistet, dass

$$|l_{ik}| = \left| \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \right| \leq 1 \qquad i > k$$

$\Rightarrow l_{ik}$ ist der Faktor, mit dem die k-te Zeile multipliziert wird, und ist damit ≤ 1,

$\Rightarrow$ die Fortpflanzung von Rundungsfehlern wird so verringert.

c) Zu beachten ist aber auch (hier für den ersten Eliminationsschritt), dass $a_{i1}^{(0)}$ und $a_{1k}^{(0)}$ symmetrisch eingehen, da

$$\begin{aligned} a_{ik}^{(1)} &= a_{ik}^{(0)} - l_{i1} \cdot a_{1k}^{(0)} \\ &= a_{ik}^{(0)} - \frac{a_{i1}^{(0)} \cdot a_{1k}^{(0)}}{a_{11}^{(0)}}. \end{aligned}$$

D.h., falls zwar l_{i1} klein, $a_{1k}^{(0)}$ aber groß ist, ergibt sich trotz Pivotmaximierung immer noch eine große Fehlerfortpflanzung von Rundungsfehlern (vgl. Beispiel 3.10).

3.10 Beispiel (Kolonnenmaximumsstrategie).

$$A = \begin{pmatrix} \boxed{0.1} & 1000 & 2000 \\ 0.01 & 2 & 3 \\ 0.04 & 5 & 6 \end{pmatrix}$$

a_{11} ist optimal bezüglich der Kolonnenmaximumsstrategie, a_{12} und a_{13} sind aber $\gg a_{11}$

$$\begin{aligned} \Rightarrow a_{22}^{(1)} &= 2 - \frac{0.01}{0.1} \cdot 1000 \\ a_{23}^{(1)} &= 3 - \frac{0.01}{0.1} \cdot 2000 \\ a_{32}^{(1)} &= 5 - \frac{0.04}{0.1} \cdot 1000 \\ a_{33}^{(1)} &= 6 - \frac{0.04}{0.1} \cdot 2000 \end{aligned}$$

Hier ergeben sich aus kleinen Werten für l und sehr großen Werten für die a_{1k} Subtraktionen von der Größenordnung

$$\mathcal{O}(1) - \mathcal{O}(100)$$

Deshalb ist es sinnvoller, die sog. *relative* Kolonnenmaximumsstrategie zu verwenden.

3.6.2 Relative Kolonnenmaximumsstrategie

Bei der relativen Kolonnenmaximumsstrategie wird diejenige Zeile gewählt, in der das potentielle Pivotelement betragsmäßig am *relativ größten* ist. Demzufolge wird statt (3.25) der Index p so bestimmt, dass

$$\max_{i \geq k} \left(\frac{\left| a_{ik}^{(k-1)} \right|}{\sum_{j=k}^{n} \left| a_{ij}^{(k-1)} \right|} \right) = \frac{\left| a_{pk}^{(k-1)} \right|}{\sum_{j=k}^{n} \left| a_{pj}^{(k-1)} \right|} \tag{3.26}$$

Falls $p \neq k$, werden die Zeile p und k vertauscht. Es ist zu beachten, dass die Werte für l natürlich nicht auf ≤ 1 beschränkt bleiben.

3.11 Beispiel (3.10 mit *relativer* Kolonnenmaximumsstrategie). In der folgenden Tabelle wird das Verfahren zur Suche des "besten" Pivotelements gezeigt

a_{1j}	a_{2j}	a_{3j}	$\lvert a_{i1} \rvert \Big/ \sum_{1}^{3} \lvert a_{ij} \rvert$
0.1	1000	2000	$3.333 \cdot 10^{-5}$
0.01	2	3	$1.996 \cdot 10^{-3}$
0.04	5	6	$3.622 \cdot 10^{-3}$ ←

→ Vertauschung $1 \leftrightarrow 3$

$$\begin{pmatrix} 0.04 & 5 & 6 \\ 0.01 & 2 & 3 \\ 0.1 & 1000 & 2000 \end{pmatrix}$$

→ Berechnung der a_{ij} mit Subtraktionen der Ordnung $\mathcal{O}(1) - \mathcal{O}(1)$ bzw. $\mathcal{O}(1000) - \mathcal{O}(10)$

$$\begin{aligned} a_{22}^{(1)} &= 2 - \frac{0.01}{0.04} \cdot 5 \\ a_{23}^{(1)} &= 3 - \frac{0.01}{0.04} \cdot 6 \\ a_{32}^{(1)} &= 1000 - \frac{0.1}{0.04} \cdot 5 \\ a_{33}^{(1)} &= 2000 - \frac{0.1}{0.04} \cdot 6 \end{aligned}$$

3.12 Definition (Maschinengenauigkeit). *Die Maschinengenauigkeit ist die Anzahl der Mantissenstellen (dezimal), auf die nach jeder Operation gerundet wird. Im weiteren Verlauf wird* fl.op. *für "floating point operation" (also Operationen mit Gleitkommazahlen, "Gleitkommaarithmetik") verwendet.*

3.13 Beispiel (Zerlegung von A mit dreistelliger Maschinengenauigkeit).

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

a) Zerlegung ohne Pivotisierung

$$\begin{aligned} l_{21} &= \frac{a_{21}}{a_{11}} = \text{ fl.op. } \left(\frac{1}{10^{-4}}\right) = 10^4 \\ u_{22} &= a_{22}^{(1)} = a_{22} - l_{21}a_{12} = \text{ fl.op. } (1 - 10^4 \cdot 1) \underset{\text{Rundung}}{=} -10^4 \\ \Rightarrow LU &= \begin{pmatrix} 1 & 0 \\ 10^4 & 1 \end{pmatrix} \begin{pmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{pmatrix} = \\ &= \begin{pmatrix} 10^{-4} & 1 \\ 1 & 0 \end{pmatrix} \end{aligned}$$

Die Matrix A, die sich aus der LU-Zerlegung ergibt, entspricht also nicht mehr der ursprünglichen Matrix A, da $a_{22} = 0$ (statt vorher 1)!!!

Man beachte, dass sich für ALLE Matrizen, die ein a_{22} haben, so dass fl.op. $(a_{22} - 10^4) = -10^4$ gilt, die gleiche LU-Zerlegung ergibt!!!

b) Jetzt mit Pivotisierung (hier nur Zeilenvertauschung)

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

dann gilt

$$\begin{aligned} l_{21} &= \frac{a_{21}}{a_{11}} = \text{ fl.op. } \left(\frac{10^{-4}}{1}\right) = 10^{-4} \\ u_{22} &= a_{22} - l_{21}a_{12} = \text{ fl.op. } (1 - 10^{-4} \cdot 1) = 1 \end{aligned}$$

und damit

$$L \cdot U = \begin{pmatrix} 1 & 0 \\ 10^{-4} & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 10^{-4} & 1.0001 \end{pmatrix}$$

Da bei dreistelliger Maschinengenauigkeit $1.0001 = 1$ gilt, haben wir also wieder die korrekte Matrix und die LU-Zerlegung war erfolgreich!

3.14 Beispiel ($N = 3$, fünfstellige Genauigkeit). Die Matrix A habe vor dem ersten Schritt die Form

$$A = \begin{pmatrix} 2.1 & 2512 & -2516 \\ -1.3 & 8.8 & -7.6 \\ 0.9 & -6.2 & 4.6 \end{pmatrix}$$

Wir suchen mit Hilfe der relativen Kolonnenmaximumsstrategie nach dem "besten" Pivotelement, zunächst vor dem Schritt $k = 1$

$A^{(k-1)}$	$s_i = \sum_{j=k}^{N} \lvert a_{ij}^{(k-1)} \rvert$	$q_i = \frac{\lvert a_{ik}^{(k-1)} \rvert}{s_i}$
$\begin{pmatrix} 2.1 & 2512 & -2516 \\ -1.3 & 8.8 & -7.6 \\ 0.9 & -6.2 & 4.6 \end{pmatrix}$	5030.1 17.7 11.7	0.00041 0.0734 0.0769 $\rightarrow$ max

Es werden also die Zeilen I und III vertauscht, und dann der Schritt $k = 1$ durchgeführt. Wir suchen wiederum nach dem besten Pivotelement (nun in den Zeilen II und III).

$A^{(k-1)}$	s_i	q_i
$\begin{pmatrix} 0.9 & -6.2 & 4.6 \\ l_{21} = -1.4444 & -0.15530 & -0.95580 \\ l_{31} = 2.3333 & 2526.5 & -2526.7 \end{pmatrix}$	$-$ 1.1111 5053.2	$-$ 0.139.. 0.499 $\rightarrow$ max

Die Zeilen II und III werden vertauscht und Schritt 2 wird durchgeführt. Man erhält

$$\begin{pmatrix} 0.9 & -6.2 & 4.6 \\ 2.3333 & 2526.5 & -2526.7 \\ -1.4444 & l_{32} = -6.1468 \cdot 10^{-5} & -1.1111 \end{pmatrix}$$

und damit haben wir die LU-Zerlegung abgeschlossen. Für die Determinante von A erhalten wir

$$\det(A) = \underbrace{(-1)^2}_{\text{2 Vertauschungen}} \cdot 0.9 \cdot 2526.5 \cdot (-1.1111) = -2526.5$$

Wir wollen nun das Gleichungssystem $A \cdot \boldsymbol{x} = \boldsymbol{b}$ für $b = (6.5, -5.3, 2.9)^\intercal$ lösen. Dazu wird zunächst $\boldsymbol{y}$ mittels

$$L \cdot \boldsymbol{y} = P \cdot \boldsymbol{b}, \qquad P \cdot \boldsymbol{b} = \begin{pmatrix} 2.9 \\ 6.5 \\ -5.3 \end{pmatrix}$$

bestimmt. Vorwärtseinsetzen liefert

$$\begin{aligned} y_1 &= 2.9 \\ y_2 &= 6.5 - 2.3333 \cdot 2.9 = -0.26660 \\ y_3 &= -5.3 - \left[(-1.4444 \cdot 2.9) + (-6.1428 \cdot 10^{-5}) \cdot (-0.2666)\right] = -1.1112 \end{aligned}$$

Dann löst man

$$U \cdot \boldsymbol{x} = \boldsymbol{y}$$

durch Rückwärtseinsetzen und erhält schließlich

$$\begin{aligned} x_3 &= 1.0001 \\ x_2 &= 1.0001 \\ x_1 &= 5.0002. \end{aligned}$$

Die Inversion der Matrix A erfolgt durch N-malige Lösung mit $\boldsymbol{e}_i$, $i = 1, \ldots, N$ oder direkt mittels sogenannter "Austauschschritte"[2]

3.7 Iterative Verbesserung der Lösung

Sei $\boldsymbol{x}$ die exakte Lösung von $A \cdot \boldsymbol{x} = \boldsymbol{b}$ und $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta x}$ die numerische Lösung, d.h.

$$A \cdot \hat{\boldsymbol{x}} = A \cdot (\boldsymbol{x} + \boldsymbol{\delta x}) = \boldsymbol{b} + \boldsymbol{\delta b}$$

mit $\boldsymbol{\delta b} = A\boldsymbol{\delta x}$. Damit lässt sich folgende Korrektur $\boldsymbol{\delta x}$ berechnen

$$A \cdot \boldsymbol{\delta x} = \boldsymbol{\delta b} = A \cdot \hat{\boldsymbol{x}} - \boldsymbol{b} \tag{3.27}$$

$\boldsymbol{r} = A \cdot \hat{\boldsymbol{x}} - \boldsymbol{b}$ heißt Residuum und gibt die Abweichung $A \cdot \hat{\boldsymbol{x}}$ (der numerischen Lösung) vom Konstantenvektor an. $\boldsymbol{r}$ lässt sich in $\mathcal{O}(N^2)$ Operationen berechnen. Dann lässt sich

$$A \cdot \boldsymbol{\delta x} = \boldsymbol{r} \tag{3.28}$$

sofort lösen, falls A LU zerlegt wurde (es sind wiederum N^2 Operationen für das Vorwärts- und Rückwärtseinsetzen nötig). Die verbesserte Lösung ist dann

$$x_{\text{neu}} = \hat{\boldsymbol{x}} - \boldsymbol{\delta x}. \tag{3.29}$$

Das neue Residuum sollte kleiner geworden sein!
Es ist zu beachten, dass

- $A \cdot \hat{\boldsymbol{x}} - \boldsymbol{b}$ in doppelter Genauigkeit (bzgl. der LU-Zerlegung) berechnet werden muss (sowohl die Matrixmultiplikation als auch die Subtraktion), um Auslöschungseffekte zu vermeiden. Diese Vorgehensweise ist ESSENTIELL für den Erfolg des Verfahrens.
- vor der LU-Zerlegung die Matrix A abgespeichert werden muss, da sie zur Berechnung der Residuums benötigt wird.
- auch der Vektor $\boldsymbol{b}$ vor dem ersten Vorwärts- bzw. Rückwärtseinsetzen abgespeichert werden muss, da dieser ebenfalls zur Berechnung von $\boldsymbol{r}$ benötigt wird.
- eine einmalige Verbesserung meist ausreichend ist.
- eine zweimalige Verbesserung verwendet werden kann, um die Konvergenz des Verfahrens zu überprüfen.
- die zusätzliche Rechenzeit (einige N^2, aufgrund der doppeltgenauen Multiplikation $A \cdot \hat{\boldsymbol{x}}$) kaum ins Gewicht fällt.

[2] z.B. Schwarz, "Numerische Mathematik", Kap. 1.4

3.8 Fehlerbetrachtung

Aussagen über die erzielte Genauigkeit bezüglich der numerischen Lösung $\hat{\boldsymbol{x}}$ und den Fehler $\boldsymbol{x} - \hat{\boldsymbol{x}}$ sind insbesondere bei linearen Gleichungssystemen und Inversionsverfahren äußerst wichtig. Da, wie sich herausstellen wird, die erzielte Genauigkeit von der Matrix A selbst abhängt, benötigen wir bestimmte *Maßzahen* für vektorielle und Matrixgrößen.

3.8.1 Normen

Sei im Weiteren $\boldsymbol{x} \in \mathbb{R}^N$ und $A \in \mathbb{R}^{N\times N}$

3.15 Definition (Vektornorm). *Die Vektornorm $\|x\|$ ist eine reellwertige Funktion der Komponenten eines Vektors mit folgenden Eigenschaften*

$$\|\boldsymbol{x}\| \geq 0 \qquad \forall\ \boldsymbol{x},\ \textit{und}\ \|\boldsymbol{x}\| = 0\ \textit{nur für}\ \boldsymbol{x} = 0 \tag{3.30a}$$

$$\|c \cdot \boldsymbol{x}\| = |c| \cdot \|\boldsymbol{x}\| \qquad \forall\ c \in \mathbb{R} \tag{3.30b}$$

$$\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\| \qquad \forall\ \boldsymbol{x}, \boldsymbol{y} \quad \textit{Dreiecksungleichung} \tag{3.30c}$$

3.16 Beispiel (Verschieden Vektornormen).

$$\|\boldsymbol{x}\|_\infty := \max_i |x_i| \qquad \text{Maximumsnorm}$$

$$\|\boldsymbol{x}\|_2 := \left(\sum_i x_i^2\right)^{\frac{1}{2}} \qquad \text{Euklidische Norm}$$

$$\|\boldsymbol{x}\|_1 := \sum_i |x_i| \qquad L_1\text{-Norm}$$

Verschiedene Vektornormen sind äquivalent in dem Sinne, dass zwischen ihnen $\forall \boldsymbol{x} \in \mathbb{R}^N$ bestimmte Ungleichungen gelten, z.B.

$$\frac{1}{\sqrt{N}}\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{N}\|\boldsymbol{x}\|_\infty.$$

Im weiteren Verlauf des Skripts wird hauptsächlich die Maximumsnorm und (später) die Euklidische Norm verwendet.

3.17 Definition (Matrixnorm). *Die Matrixnorm $\|A\|$ ist eine reellwertige Funktion der Komponenten einer Matrix mit folgenden Eigenschaften*

$$\|A\| \geq 0 \qquad \forall A,\ \textit{und}\ \|A\| = 0\ \textit{nur für}\ A = 0 \tag{3.31a}$$

$$\|cA\| = |c| \cdot \|A\| \qquad \forall c \in \mathbb{R} \tag{3.31b}$$

$$\|A + B\| \leq \|A\| + \|B\| \qquad \forall A, B\ \textit{Dreiecksungleichung} \tag{3.31c}$$

$$\|A \cdot B\| \leq \|A\| \cdot \|B\| \qquad \forall A, B\ \textit{Submultiplikativität} \tag{3.31d}$$

Aus (3.31d) folgt unter anderem auch

$$\|\mathbb{1}\| = \|\mathbb{1} \cdot \mathbb{1}\| \leq \|\mathbb{1}\| \cdot \|\mathbb{1}\|$$

und damit

$$\boxed{1 \leq \|\mathbb{1}\|}$$

3.18 Beispiel (Verschieden Matrixnormen).

$$\|A\|_Z := \max_i \sum_k |a_{ik}| \qquad \text{Zeilensummennorm}$$

$$\|A\|_F := \left(\sum_i \sum_k a_{ik}^2\right)^{\frac{1}{2}} \qquad \text{Frobenius-Norm}$$

$$\|A\|_G := N \cdot max_{i,k}|a_{ik}| \qquad \text{Gesamtnorm}$$

Verschieden Matrixnormen sind äquivalent, z.B. gilt

$$\frac{1}{N}\|A\|_G \leq \|A\|_Z \leq \|A\|_G \leq N\|A\|_Z.$$

3.19 Definition. *Eine Matrixnorm $\|A\|$ und eine Vektornorm $\|\boldsymbol{x}\|$ sind kompatibel (verträglich), wenn*

$$\underbrace{\|A \cdot \boldsymbol{x}\|}_{\text{Vektornorm}} \leq \underbrace{\|A\|}_{\text{Matrixnorm}} \cdot \underbrace{\|\boldsymbol{x}\|}_{\text{Vektornorm}} \qquad \forall x \in \mathbb{R}^N, A \in \mathbb{R}^{N\times N} \tag{3.32}$$

3.20 Beispiel (Verschiedene kompatible Normen).

- $\|A\|_G$, $\|A\|_Z$ sind kompatibel mit $\|\boldsymbol{x}\|_\infty$
- $\|A\|_G$, $\|A\|_F$ sind kompatibel mit $\|\boldsymbol{x}\|_2$

Für beliebige kompatible Normen gilt normalerweise, dass

$$\|A \cdot \boldsymbol{x}\| < \|A\| \cdot \|\boldsymbol{x}\| \qquad \forall \boldsymbol{x} \neq \mathbf{0},$$

d.h. dass die linke Seite *echt kleiner* ist. Um spätere Abschätzungen besser einschränken zu können und $\|A\|$ so gut wie möglich zu minimieren, definiert man folgende Matrixnorm, um zumindest für einen Vektor Gleichheit zu erzwingen.

3.21 Definition. *Die natürliche (zugeordnete) Norm oder Grenznorm ist der zu einer gegebenen Vektornorm $\|\boldsymbol{x}\|$ definierte Zahlenwert*

$$\|A\|_{\text{nat}} :\overset{\text{I}}{=} \max_{\boldsymbol{x}\neq 0} \frac{\|A \cdot \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \overset{\text{II}}{=} \max_{\|\boldsymbol{x}\|=1} \|A \cdot \boldsymbol{x}\| \tag{3.33}$$

Man beachte, dass die natürliche *Matrix*norm über *Vektor*normen eingeführt wird.

Aus I folgt:
Sei $\boldsymbol{x}_1$ derjenige Vektor, für den obiges Maximum erreicht wird (bzgl. einer *vorgegebenen* Vektornorm), d.h.

$$\|A\|_{\text{nat}} = \frac{\|A \cdot \boldsymbol{x}_1\|}{\|\boldsymbol{x}_1\|}.$$

Dann gilt für diesen Vektor per Definition

$$\|A \cdot \boldsymbol{x}_1\| = \|A\|_{\text{nat}} \cdot \|\boldsymbol{x}_1\|,$$

d.h. (Kompatibilität vorausgesetzt) Gleichheit bezüglich Ungleichung (3.32) wird erzielt! Für beliebige (kompatible) Matrixnormen gilt aber

$$\|A \cdot \boldsymbol{x}_1\| \leq \|A\|_{\text{bel}} \cdot \|\boldsymbol{x}_1\|.$$

Damit erhalten wir sofort

$$\|A\|_{\text{nat}} \leq \|A\|_{\text{bel}}$$

Die natürliche Norm ist also die kleinste kompatible Matrixnorm!

Die Identität II folgt aus

$$\max_{\boldsymbol{x}\neq 0} \frac{\|A \cdot \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \overset{(3.31\text{b})}{=} \max_{\boldsymbol{x}\neq 0} \left\| A \cdot \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} \right\| = \max_{\|y=1\|} \|A \cdot \boldsymbol{y}\|.$$

Aus diesen und weiteren Überlegungen läßt sich folgender Satz beweisen

3.22 Satz (Die natürliche Matrixnorm).

1. *Die natürliche Matrixnorm ist eine Matrixnorm im Sinne der Definition.*
2. *Sie ist mit der zugrunde liegenden Vektornorm kompatibel.*
3. *Sie ist unter allen kompatiblen Normen die kleinste.*

3.23 Beispiel (wichtig für folgende Betrachtungen). $\|A\|_\infty$ sei die der Maximumsnorm $\|\boldsymbol{x}\|_\infty$ zugeordnete Matrixnorm

$$\|A\|_\infty \overset{\text{Def.}}{=} \max_{\|\boldsymbol{x}\|_\infty=1} \|A\cdot\boldsymbol{x}\|_\infty = \max_{\|\boldsymbol{x}\|_\infty=1} \underbrace{\left\{ \max_i \left| \underbrace{\sum_{k=1}^{N} a_{ik}x_k}_{A\cdot\boldsymbol{x}} \right| \right\}}_{\|A\cdot\boldsymbol{x}\|_\infty} = \max_i \left\{ \max_{\|\boldsymbol{x}\|_\infty=1} \left| \sum_k a_{ik}x_k \right| \right\} =$$

(alle Vektoren der Art $(1,-1,-1,1,1,-1)^\intercal$ etc haben $\|\boldsymbol{x}\|_\infty = 1$)

$$= \max_i \sum_{k=1}^{N} |a_{ik}| = \|A\|_Z \qquad \text{Zeilensummennorm}$$

Das heißt, die Zeilensummennorm ist die natürliche Matrixnorm bezüglich der Maximumsnorm, also

$$\|A\|_{\text{nat}} = \|A\|_Z \qquad \text{bzgl.} \quad \|\boldsymbol{x}\|_\infty \tag{3.34}$$

3.8.2 Fehlerabschätzung, Konditionszahl

Es stellt sich die Frage, in wie weit sich aus der Größe des (berechenbaren) Residuenvektors

$$\boldsymbol{r} = A\cdot\hat{\boldsymbol{x}} - \boldsymbol{b}$$

auf die Größe des Fehlers

$$\boldsymbol{x} - \hat{\boldsymbol{x}}$$

schließen lässt (vgl. auch Kapitel 3.7).

$$\begin{aligned} A\cdot\boldsymbol{x} &= \boldsymbol{b} \\ A\cdot\hat{\boldsymbol{x}} &= \boldsymbol{b}+\boldsymbol{r} \\ \Rightarrow A\cdot(\boldsymbol{x}-\hat{\boldsymbol{x}}) &= -\boldsymbol{r} \\ \Rightarrow \boldsymbol{x}-\hat{\boldsymbol{x}} &= -A^{-1}\cdot\boldsymbol{r} \\ \Rightarrow \|\boldsymbol{x}-\hat{\boldsymbol{x}}\| &\le \|A^{-1}\|\cdot\|\boldsymbol{r}\|, \end{aligned}$$

falls die Matrixnorm kompatibel mit der Vektornorm ist.
Mit $\boldsymbol{\delta x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$ folgt dann

$$\frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \le \frac{1}{\|\boldsymbol{x}\|}\|A^{-1}\|\cdot\|\boldsymbol{r}\|. \qquad \text{I}$$

Andererseits gilt

$$\begin{aligned} \|A\cdot\boldsymbol{x}\| &= \|\boldsymbol{b}\| \\ \|A\|\cdot\|\boldsymbol{x}\| &\ge \|\boldsymbol{b}\| \\ \Rightarrow \frac{1}{\|\boldsymbol{x}\|} &\le \frac{\|A\|}{\|\boldsymbol{b}\|}. \qquad \text{II} \end{aligned}$$

Aus I und II erhält man

$$\frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \le \|A^{-1}\|\cdot\|A\|\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|} = \kappa(A)\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}. \tag{3.35}$$

Hier ist $\kappa(A) = \|A^{-1}\| \cdot \|A\|$ die *Konditionszahl* der Matrix. Diese hängt nur von A und der verwendeten Norm ab. Es ist zu beachten, dass

$$1 \leq \|\mathbb{1}\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\|$$

gilt. Das heißt, für alle Normen und alle A ist

$$\kappa(A) \geq 1. \tag{3.36}$$

Typische Werte für $\kappa(A)$ sind 1000 - 10000.

Wir definieren nun den relativen Fehler des Ergebnisses als

$$\varepsilon = \frac{\|\boldsymbol{\delta x}\|_\infty}{\|\boldsymbol{x}\|_\infty}.$$

Falls $\varepsilon = 10^{-p}$, so stimmen die *größten* Komponenten von $\boldsymbol{x}$ und $\hat{\boldsymbol{x}}$ auf p Dezimalstellen überein, d.h. ε ist eine geeignete Größe zur Beschreibung der *erzielten* Rechengenauigkeit. Aus Gleichung (3.35) folgt damit, dass aus einem kleinen Residuum nur dann auf einen kleinen relativen Fehler ε geschlossen werden darf, wenn die *Konditionszahl klein* ist!

3.8.3 Numerische Stabilität

Wir wollen eigentlich

$$\text{I}: \qquad A \cdot \boldsymbol{x} = \boldsymbol{b}$$

berechnen. Aufgrund der numerischen Abarbeitung (z.B. unter Verwendung der LU-Zerlegung, $LU = A + E$, siehe Beispiel 3.13) lösen wir aber tatsächlich das gestörte Gleichungssystem

$$\text{II}: \qquad (A + E)\hat{\boldsymbol{x}} = \boldsymbol{b}.$$

Um eine relevante Lösung zu erzielen, sollte die Störung E natürlich klein gegen A sein. (Wir subsummieren hier, dass der hauptsächliche Fehler in $\boldsymbol{x}$ bei der LU-Zerlegung und nicht beim Vorwärts- bzw. Rückwärtseinsetzen gemacht wird). Für den resultieren Fehler ergibt sich dann aus I und II

$$\begin{aligned} A \cdot (\boldsymbol{x} - \hat{\boldsymbol{x}}) &= -E \cdot \boldsymbol{x} \\ \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| &= \|A^{-1}E\hat{\boldsymbol{x}}\| \leq \|A^{-1}\| \cdot \|E\| \cdot \|\hat{\boldsymbol{x}}\| \end{aligned}$$

und damit

$$\varepsilon = \frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leq \kappa(A)\frac{\|E\|}{\|A\|} \tag{3.37}$$

Damit ε klein ist, muss also sowohl die Konditionszahl als auch $\frac{\|E\|}{\|A\|}$ klein sein, wobei letzteres natürlich vom verwendeten Algorithmus abhängt.

Bei numerisch stabilen Algorithmen (Gauss, LU-Zerlegung mit Pivotisierung) läßt sich zeigen, daß typischerweise $\frac{\|E\|}{\|A\|} \lesssim N \cdot \mathcal{O}(\delta)$, wenn δ die Maschinengenauigkeit ist.

$$\Rightarrow \varepsilon \leq \kappa(A) \cdot N \cdot \delta$$

3.24 Beispiel (vgl. Beispiel 3.13).

$$A = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}, \qquad \delta = 10^{-3}$$

Mit der (exakten) inversen Matrix

$$A^{-1} = \begin{pmatrix} -1.0001 & 1.0001 \\ 1.0001 & -1.0001 \cdot 10^{-4} \end{pmatrix}$$

folgt dann

$$\kappa(A) = \|A\|_\infty \|A^{-1}\|_\infty = 2 \cdot 2.0002$$

und wir sehen, dass die Matrix gut konditioniert ist. Ohne Pivotisierung war

$$\begin{aligned} LU &= \begin{pmatrix} 10^{-4} & 1 \\ 1 & 0 \end{pmatrix} \\ \Rightarrow \|E\|_\infty &= \|A - LU\|_\infty = 1 \\ \varepsilon &\leq \kappa(A)\frac{\|E\|_\infty}{\|A\|_\infty} \approx 4 \cdot \frac{1}{2} = 2!! \end{aligned}$$

d.h. die LU-Zerlegung ohne Pivotisierung ist numerisch instabil. Mit Pivotisierung hatten wir

$$\begin{aligned} LU &= \begin{pmatrix} 1 & 1 \\ 10^{-4} & 1.0001 \end{pmatrix} \\ \|E\|_\infty &= 10^{-4} \\ \frac{\|E\|_\infty}{\|A\|_\infty} &= \mathcal{O}(10^{-4}) \end{aligned}$$

Berücksichtigen wir noch die letzte Rundung ($1.0001 \rightarrow 1.00$) bei einer Maschinengenauigkeit von $\delta = 10^{-3}$, so ergibt sich in diesem Fall sogar ein verschwindender Fehler, $\|E\|_\infty = 0$!

Problem: Zur Berechnung von κ benötigen wir die Norm der inversen Matrix, $\|A^{-1}\|$. Diese lässt sich z.B. mit LAPACK-Routinen (Endung "x") in $\mathcal{O}(N^2)$ Operationen simultan mit der LU-Zerlegung berechnen.

3.8.4 Störung in den Eingangsdaten

A und $\boldsymbol{b}$ können meist nicht exakt dargestellt werden, sondern unterliegen Rundungsfehlern, zum einen bei der Initialisierung der entsprechenen `Real` Variablen und insbesondere, falls sie selbst Ergebnisse von vorherigen Rechnungen sind.

Es stellt sich die Frage, wie groß der resultierenden Fehler $\boldsymbol{\delta x}$ *allein* aufgrund dieser Fehler in den Eingangsdaten ist, d.h. wenn wir einen numerisch stabilen Algorithmus mit vernachlässigbarem $\|E\|$ verwenden könnten. (Dieser Fehler ist der kleinstmögliche und auf jeden Fall vorhanden!). Es gilt also, das Problem

$$(A + \Delta A)(\boldsymbol{x} + \boldsymbol{\delta x}) = (\boldsymbol{b} + \boldsymbol{\Delta b}) \tag{3.38}$$

mit Eingangsfehlern $\Delta A < A$, $\boldsymbol{\Delta b} < \boldsymbol{b}$ zu lösen, wobei wir die gestörte Matrix $(A + \Delta A)$ als regulär annehmen. Ausmultiplizieren ergibt

$$\begin{aligned} \boldsymbol{\delta x} &= A^{-1}(\boldsymbol{\Delta b} - \Delta A \cdot \boldsymbol{x} - \Delta A \cdot \boldsymbol{\delta x}) \\ \|\boldsymbol{\delta x}\| &\leq \|A^{-1}\|\|\boldsymbol{\Delta b} - \Delta A \cdot \boldsymbol{x} - \Delta A \boldsymbol{\delta x}\| \\ &\leq \|A^{-1}\| \left\{\|\boldsymbol{\Delta b}\| + \|\Delta A\| \cdot \|\boldsymbol{x}\| + \|\Delta A\| \cdot \|\boldsymbol{\delta x}\|\right\} \\ \\ &\Rightarrow \|\boldsymbol{\delta x}\| \underbrace{\left\{1 - \|A^{-1}\|\|\Delta A\|\right\}}_{>0 \text{ wegen kleiner Störung}} \leq \|A^{-1}\| \left\{\|\boldsymbol{\Delta b}\| + \|\Delta A\| \cdot \|\boldsymbol{x}\|\right\} \\ \|\boldsymbol{\delta x}\| &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\|\Delta A\|} \left(\|\boldsymbol{\Delta b}\| + \|\Delta A\| \cdot \|x\|\right) \qquad \left(\frac{1}{\|\boldsymbol{x}\|} \leq \frac{\|A\|}{\|\boldsymbol{b}\|}\right) \\ \frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} &\leq \frac{\|A^{-1}\|\|A\|}{1 - \|A^{-1}\|\|\Delta A\|} \left(\frac{\|\boldsymbol{\Delta b}\|}{\|\boldsymbol{b}\|} + \frac{\|\Delta A\|}{\|A\|}\right) \end{aligned}$$

Ergebnis:

$$\varepsilon = \frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\boldsymbol{\|\Delta b\|}}{\|\boldsymbol{b}\|}\right) \approx \kappa(A) \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\boldsymbol{\|\Delta b\|}}{\|\boldsymbol{b}\|}\right), \tag{3.39}$$

d.h. der relative Fehler in den Eingangsdaten wird mit κ verstärkt.

Wir nehmen jetzt an, dass mit d-stelligen fl.op. gerechnet wird, d.h. für die Eingangsdaten gilt $\frac{\|\Delta A\|}{\|A\|} \approx \frac{\|\boldsymbol{\Delta b}\|}{\|\boldsymbol{b}\|} \approx 5 \cdot 10^{-d}$. (Der Fehler in der Darstellung wirkt sich nur auf die letzte Stelle aus).
Sei nun $\kappa(A) = 10^a$ und $5 \cdot 10^{a-d} \ll 1$. Dann gilt mit (3.39)

$$\varepsilon = \frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leq 10^{-(d-a-1)} \tag{3.40}$$

Daumenregeln.

(i) Bei der Lösung von $A \cdot \boldsymbol{x} = \boldsymbol{b}$ sind in der berechneten Lösung allein aufgrund der Eingangsfehler nur $d - a - 1$ Dezimalstellen, bezogen auf die betragsmäßig größte Komponente, sicher. Der Fehler in den kleineren Komponenten kann sehr viel größer werden (d-stellige fl.op., $\kappa(A) = 10^a$)

(ii) Damit das Nachiterationsschema (Kapitel 3.7) eine Verbesserung ergibt, muss

$$a < d - 1 \tag{3.41}$$

gelten.

3.9 Tridiagonalsysteme

Tridiagonalsysteme treten oftmals bei Randwertproblemen auf ($\Rightarrow$ gewöhnliche Differentialgleichungen), aber auch in anderem Zusammenhang, wie z.B. bei der Berechnung interpolierender Splines. Ein solches System hat die Form

$$\begin{pmatrix} b_1 & c_1 & 0 & & \dots & \\ a_2 & b_2 & c_2 & & \dots & \\ 0 & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & c_{N-1} \\ & & & & a_N & b_N \end{pmatrix} \begin{pmatrix} x_1 \\ \\ \vdots \\ \\ x_N \end{pmatrix} = \begin{pmatrix} y_1 \\ \\ \vdots \\ \\ y_N \end{pmatrix}$$

Die Koeffizienten des Systems werden sinnvollerweise in 4 Vektoren (oder Feldern) $a(N)$, $b(N)$, $c(N)$ und $y(N)$, abgespeichert. Die nicht benötigten Werte $a(1)$ und $c(N)$ werden auf Null gesetzt.

Das Standardlösungsschema für Tridiagonalsysteme, falls keine Pivotisierung erforderlich ist, ist die "normale" Gauss-Elimination.

1. Zeile (I/b_1)

$$\longrightarrow \quad 1 \quad d_1 \quad 0 \quad \dots \;\Big|\; y_1'$$

mit

$$\begin{aligned} d_1 &= c_1/b_1 \\ y_1' &= y_1/b_1 \end{aligned}$$

2. Zeile (II $- a_2 \cdot$ I)

$$\begin{array}{cccc|c} 1 & d_1 & 0 & \dots & y_1' \\ a_2 & b_2 & c_2 & \dots & y_2 \end{array}$$

$$\longrightarrow \quad \begin{array}{cccc|c} 1 & d_1 & 0 & \dots & y_1' \\ 0 & b_2' & c_2' & \dots & \tilde{y}_2 \end{array}$$

mit

$$\begin{aligned} b_2' &= b_2 - a_2 \cdot d_1 \\ \tilde{y}_2 &= y_2 - a_2 \cdot y_1' \\ c_2' &= c_2 - a_2 \cdot 0 = c_2 \end{aligned}$$

$$\begin{array}{rcccc|c} & 1 & d_1 & 0 & \dots & y_1' \\ \longrightarrow & 0 & 1 & d_2 & \dots & y_2' \end{array}$$

mit

$$\begin{aligned} d_2 &= c_2/b_2' \\ y_2' &= \tilde{y}_2/b_2'. \end{aligned}$$

Analog verfährt man bei den Schritten $i = 3, \dots, N$ (d_N existiert nicht). Das Ergebnis ist dann

$$\begin{pmatrix} 1 & d_1 & & & \\ 0 & 1 & d_2 & & \\ & & \ddots & \ddots & \\ & & 0 & 1 & d_{N-1} \\ & & & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \\ \vdots \\ \\ x_N \end{pmatrix} = \begin{pmatrix} y_1' \\ \\ \vdots \\ \\ y_N' \end{pmatrix}.$$

Die Koeffizienten ergeben sich aus

$$\begin{aligned} b_i' &= b_i - d_{i-1} a_i \\ d_i &= c_i / b_i' \qquad \text{und} \\ y_i' &= (y_i - y_{i-1}' a_i)/b_i', \qquad i = 2, \dots, N \end{aligned}$$

Den Lösungsvektor $\boldsymbol{x}$ erhält man durch Rückwärtseinsetzen

$$\begin{aligned} x_N &= y_N' \\ x_i &= y_i' - d_i x_{i+1}, \qquad i = N-1, \dots, 1 \end{aligned}$$

Für die Berechnung der d_i und y_i' wird eine Schleife von $1, \dots, N$ benötigt (falls die c_i und y_i nicht zerstört werden sollen, müssen sie extra abgespeichert werden). Für diese Vorwärtselimination sind

$$Z_{\text{Tri., Vorw.}} = 2 + 4(N-1)$$

wesentliche Operationen nötig. Die x_i werden mit einer Schleife von $N-1, \dots, 1$ berechnet, d.h. für die Rückwärtseinsetzung werden

$$Z_{\text{Tri., Rückw.}} = N - 1$$

Operationen benötigt. Die gesamte Rechenzeit skaliert also mit

$$Z_{\text{Tri.}} = 5N - 3.$$

Inversion. Zur Inversion wird das System N-mal mit $\boldsymbol{y} = \boldsymbol{e}_i$, also durch $\mathcal{O}(N^2)$ Operationen gelöst.

Pivotisierung. In der Regel ist Pivotisierung bei Tridiagonalsystemen nicht nötig, da sie aufgrund ihrer Herkunft meist diagonal dominiert sind (oder oftmals symmetrisch und positiv definit, siehe Kap. 3.10).

3.25 Definition (Diagonle Dominanz). *Diagonale Dominanz liegt vor, falls in jeder Zeile gilt*

$$|a_{ii}| > \sum_{k \neq i} |a_{ik}| \tag{3.42}$$

3.26 Satz.

Falls diagonale Dominanz vorliegt, ist keine Pivotisierung (d.h. Zeilenaustausch) erforderlich (3.43)

Der Beweis erfolgt dadurch, dass man zeigen kann, dass sich diagonale Dominanz auf reduzierte Gleichungssysteme überträgt!

Falls keine diagonale Dominanz vorliegt, ist unter Umständen Pivotisierung erforderlich. Bzgl. der entsprechenden Verfahren verweisen wir auf die Literatur [3]. Die Anzahl der wesentlichen Operationen ist dann $9(N-1)$.

Anmerkung.

- Es gibt Probleme, bei denen *nur die Diagonale* der Inversen benötigt wird (A^{-1} ist voll besetzt, wenn A Tridiagonalgestalt hat). Diese Größe lässt sich unter Verwendung eines äußerst geschickten Algorithmus in $\mathcal{O}(N)$ Operationen anstatt in $\mathcal{O}(N^2)$ Operationen berechnen [4].

3.10 Symmetrische, positiv definite Systeme: Das CHOLESKY Verfahren

In bestimmten Fällen ist die Koeffizientenmatrix symmetrisch *und* positiv definit (z.B. bei linearen Ausgleichsverfahren, siehe Kap. 5)

3.27 Definition. *Eine symmetrische Matrix $A \in \mathbb{R}^{N\times N}$ heißt* positiv definit*, wenn die dazugehörige quadratische Form*

$$Q(\boldsymbol{x}) = \boldsymbol{x}^\intercal A\boldsymbol{x} = \sum_i \sum_k a_{ik}x_i x_k > 0 \qquad \forall \boldsymbol{x} \in \mathbb{R}^N \text{ und } \boldsymbol{x} \neq 0 \tag{3.44}$$

3.28 Satz. *Positiv definite symmetrische Matrizen haben folgende Eigenschaften:*

- *Alle Diagonalelemente sind echt größer Null,* $a_{ii} > 0, \quad i = 1, \ldots, N.$ (3.45a)
- $a_{ik}^2 < a_{ii}a_{kk} \qquad \forall i,k, \quad i \neq k.$ (3.45b)
- *Das betragsmäßig größte Element liegt auf der Diagonalen.* (3.45c)

3.29 Beweis.

- *zu (3.45a) Wähle* $\boldsymbol{x} = \boldsymbol{e}_i \Rightarrow Q(\boldsymbol{x}) = a_{ii}$*, da* $Q(\boldsymbol{x}) > 0\ \forall \boldsymbol{x}$ *per Definition*
- *zu (3.45b) Wähle* $\boldsymbol{x} = \xi\boldsymbol{e}_i + \boldsymbol{e}_k$*,* $\xi \in \mathbb{R} \Rightarrow$

$$\begin{aligned} Q(\xi\boldsymbol{e}_i + \boldsymbol{e}_k) &= \sum_j \sum_l a_{jl} \underbrace{(\xi\delta_{ij} + \delta_{kj})}_{\delta_{ij} \text{ ist das KRONECKER-}\delta} (\xi\delta_{il} + \delta_{kl}) \\ &= \sum_j (\xi a_{ij} + a_{jk})(\xi\delta_{ij} + \delta_{kj}) \\ &= \xi^2 a_{ii} + \xi a_{ik} + \xi a_{ki} + a_{kk} \\ &= \xi^2 a_{ii} + 2\xi a_{ik} + a_{kk} \overset{\text{per Definition}}{>} 0 \end{aligned}$$

$\Rightarrow \xi^2 a_{ii} + 2\xi a_{ik} + a_{kk} = 0$ *darf keine Lösung für* $\xi \in \mathbb{R}$ *haben. Für die Diskriminante ("*$(b^2 - 4ac)$*") muss also* $4a_{ik}^2 - 4a_{ii}a_{kk} < 0$ *gelten.*

$$\Rightarrow a_{ik}^2 < a_{ii}a_{kk} \square$$

[3] z.B. Schwarz, "Numerische Mathematik", Kap. 1.3.3

[4] Rybicki & Hummer, 1991, Astronomy & Astrophysics 245, 171, Appendix A

- *zu (3.45c) Wähle* $\max|a_{ik}|$ *so, dass es nicht auf der Diagonalen liegt,*

$$\Rightarrow a_{ik}^2 > a_{ii}a_{kk}.$$

Dies steht aber im Widerspruch zu (3.45b)

- Es ist zu beachten, dass die Eigenschaften (3.45) notwendige, aber *nicht* hinreichende Bedingungen für positive Definitheit sind.
- Eine notwendige und hinreichende Bedingung ergibt sich aus der Reduktion der quadratischen Form auf eine Summe von Quadraten (s.u.).
- U.a. gilt allerdings, dass eine symmetrische Matrix positiv definit ist, wenn *alle* Eigenwerte λ_i echt positiv sind.

Jetzt werden wir eine erste Zerlegung von Q durchführen. Dabei ist zu beachten, dass per Definition $a_{11} > 0$. Wir bilden einen quadratischen Ausdruck bezüglich x_1 und nutzen dabei die Symmetrie.

$$\begin{aligned}
Q(\boldsymbol{x}) =& \sum_i \sum_k a_{ik}x_i x_k && (\text{“}x_1\text{” Terme heraus ziehen})\\
=& \sum_{i=1}^N a_{i1}x_i x_1 + \sum_{i=1}^N \sum_{k=2}^N a_{ik}x_i x_k\\
=& \sum_{i=1}^N a_{i1}x_i x_1 + \sum_{k=2}^N a_{1k}x_1 x_k + \sum_{i=2}^N \sum_{k=2}^N a_{ik}x_i x_k = && (a_{1k} = a_{k1})\\
=& a_{11}x_1^2 + 2\sum_{i=2}^N a_{i1}x_i x_1 + \sum_{i=2}^N \sum_{k=2}^N a_{ik}x_i x_k\\
=& \left((\sqrt{a_{11}}x_1)^2 + 2\sum_{i=2}^N a_{i1}x_i x_1 + \left(\sum_{i=2}^N \frac{a_{i1}}{\sqrt{a_{11}}}\cdot x_i\right)^2\right) && a_{11} > 0\\
&+ \sum_{i=2}^N \sum_{k=2}^N a_{ik}x_i x_k - \sum_{i=2}^N \sum_{k=2}^N \frac{a_{i1}a_{k1}}{a_{11}}x_i x_k && \text{Abziehen der quadr. Ergänzung}\\
=& \left(\sqrt{a_{11}}x_1 + \sum_{i=2}^N \frac{a_{i1}}{\sqrt{a_{11}}}x_i\right)^2 + \sum_{i=2}^N \sum_{k=2}^N \left(a_{ik} - \frac{a_{i1}a_{k1}}{a_{11}}\right)x_i x_k\\
=& \left(\sum_{i=1}^N l_{i1}x_i\right)^2 + \underbrace{\sum_{i=2}^N \sum_{k=2}^N a_{ik}^{(1)}x_i x_k}_{Q^{(1)}(x^{(1)})} && (3.46)
\end{aligned}$$

Hier gilt

$$\begin{aligned}
l_{11} =& \sqrt{a_{11}}\\
l_{i1} =& \frac{a_{i1}}{\sqrt{a_{11}}} = \frac{a_{i1}}{l_{11}}, && i = 2,\ldots,N\\
a_{ik}^{(1)} =& \underbrace{a_{ik} - \frac{a_{i1}a_{k1}}{a_{11}}}_{\hat{=}\text{ 1. Schritt der Gauss-Elimination}} = a_{ik} - l_{i1}l_{k1} && i,k = 2,\ldots,N
\end{aligned}$$

$Q^{(1)}$ ist eine quadratische Form bezüglich derjenigen Matrix, die aus dem ersten Gaussschen Eliminationschritt folgt (siehe Gleichung (3.7)).

3.30 Satz. *Eine symmetrische Matrix $A \in \mathbb{R}^{N\times N}$ mit $a_{11} > 0$ ist genau dann positiv definit, falls die reduzierte Matrix $A^{(1)}$ mit den Elementen $a_{ik}^{(1)}$ wie in Gleichung (3.46) positiv definit ist.*

3.31 Beweis (Notwendigkeit). *Sei A positiv definit. Dann lässt sich zu jedem Vektor $\boldsymbol{x}^{(1)} = (x_2, \ldots, x_N)^\intercal \neq 0$ aufgrund von $a_{11} > 0$ (und damit $l_{11} \neq 0$) ein solches x_1 finden, so dass*

$$\sum_{i=1}^{N} l_{i1}x_i = 0 \qquad \left(\text{explizit: } x_1 = -\frac{\sum_{i=2}^{N} l_{i1}x_i}{l_{11}}\right)$$

$$\Rightarrow Q(\boldsymbol{x}) = \sum_{i=1}^{N} l_{i1}x_i + Q^{(1)}(\boldsymbol{x}^{(1)})$$

mit $\boldsymbol{x} = (x_1, \boldsymbol{x}^{(1)})^\intercal$ und $\boldsymbol{x}^{(1)}$ beliebig. Es ist aber auch

$$0 < Q(\boldsymbol{x}) = 0 + Q^{(1)}(\boldsymbol{x}^{(1)})$$

und damit

$$\Rightarrow Q^{(1)}(\boldsymbol{x}^{(1)}) > 0 \qquad \forall \boldsymbol{x}^{(1)},$$

das heißt auch $Q^{(1)}$ ist positiv definit.

Damit ergibt sich ein einfacher Test bezüglich positiver Definitheit einer symmetrischen Matrix:

3.32 Satz. *Eine symmetrische Matrix $A \in \mathbb{R}^{N\times N}$ ist genau dann positiv definit, wenn der* Gausssche *Eliminationsprozess ohne Zeilenaustausch mit N positiven Pivotelementen ausführbar ist.*

Dies folgt aus einer sukzessiven Zerlegung: Wenn $a_{11} > 0$ gilt, dann ist $Q^{(1)}$ positiv definit; damit ist dann $a_{22}^{(1)} > 0$, und $Q^{(2)}$ ebenfalls positiv definit, usw. Damit kann die gesamte Matrix auf positive Definitheit überprüft werden. Falls man also ein *nicht-positives Diagonalelement* $a_{ii}^{(i-1)}$ findet, ist die Matrix *nicht* positiv definit.

Aufgrund von

$$Q(\boldsymbol{x}) = \left(\sum_{i=1}^{N} l_{i1}x_i\right)^2 + Q^{(1)}(\boldsymbol{x}^{(1)})$$

lässt sich die quadratische Form immer weiter zerlegen

$$Q^{(1)}(\boldsymbol{x}^{(1)}) = \left(\sum_{i=2}^{N} l_{i2}x_i\right)^2 + Q^{(2)}(\boldsymbol{x}^{(2)}),$$

etc. Letztendlich zerfällt die quadratische Form in eine Summe aus N Quadraten.

$$Q(\boldsymbol{x}) = \sum_i \sum_k a_{ik}x_i x_k = \sum_{k=1}^{N} \left(\sum_{i=k}^{N} l_{ik}x_i\right)^2, \tag{3.47}$$

wobei analog zum ersten Reduktionsschritt

$$\begin{aligned} l_{kk} &= \sqrt{a_{kk}^{(k-1)}}, \qquad l_{ik} = \frac{a_{ik}^{(k-1)}}{l_{kk}} && i = k+1, \ldots N \\ a_{ij}^{(k)} &= a_{ij}^{(k-1)} - l_{ik}l_{jk}, && i,j = k+1, \ldots, N. \end{aligned}$$

Damit lässt sich folgende Linksmatrix L definieren

$$L = \begin{pmatrix} l_{11} & 0 & 0 & \ldots & 0 \\ l_{21} & l_{22} & 0 & \ldots & 0 \\ l_{31} & l_{32} & l_{33} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{N1} & l_{N2} & l_{N3} & \ldots & l_{NN} \end{pmatrix}. \tag{3.48}$$

Für $Q(\boldsymbol{x})$ hatten wir die beiden Darstellungen

i)
$$Q(\boldsymbol{x}) = \sum_i \sum_k a_{ik} x_i x_k = \boldsymbol{x}^\intercal A \boldsymbol{x}$$

ii)
$$Q(\boldsymbol{x}) = \sum_k \left(\sum_{i=k}^{N} l_{ik} x_i \right)^2 \stackrel{!}{=} (L^\intercal \boldsymbol{x})^\intercal (L^\intercal \boldsymbol{x})$$

3.33 Beispiel (Darstellung ii) mit $N = 3$).

$$\begin{aligned} Q(\boldsymbol{x}) = (l_{11}x_1 + l_{21}x_2 + l_{31}x_3)^2 \quad &+ \\ (l_{22}x_2 + l_{32}x_3)^2 \quad &+ \\ (l_{33}x_3)^2. & \end{aligned}$$

Andererseits gilt

$$L^\intercal \cdot \boldsymbol{x} = \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} l_{11}x_1 + l_{21}x_2 + l_{31}x_3 \\ l_{22}x_2 + l_{32}x_3 \\ l_{33}x_3 \end{pmatrix},$$

das heißt, es ist

$$\underbrace{(L^\intercal \cdot \boldsymbol{x})^\intercal}_{\text{Zeilenvektor}} \cdot (L^\intercal \cdot \boldsymbol{x}) = (l_{11}x_1 + \cdots)^2 + (l_{22}x_2 + \cdots)^2 + (l_{33}x_3)^2 \stackrel{!}{=} Q(\boldsymbol{x})$$

Insgesamt finden wir also

$$\begin{aligned} Q(\boldsymbol{x}) \quad &= \quad \boldsymbol{x}^\intercal A \boldsymbol{x} = (L^\intercal \boldsymbol{x})^\intercal \cdot (L^\intercal \boldsymbol{x}) = \boldsymbol{x}^\intercal (LL^\intercal) \boldsymbol{x} \\ \text{mit} \quad &\quad A = LL^\intercal \end{aligned} \tag{3.49}$$

Eine positiv definite, symmetrische Matrix lässt sich in das Produkt $LL^\intercal$ zerlegen. Diese Zerlegung heißt CHOLESKY-*Zerlegung*.

Vorgehensweise. Es gilt, das Gleichungssystem

$$A \cdot \boldsymbol{x} = \boldsymbol{b}$$

zu lösen, wobei A eine symmetrische, positiv definite Matrix ist. Durch die Zerlegung $A = L \cdot L^\intercal$ von A erhalten wir das System $L\,(L^\intercal \cdot \boldsymbol{x}) = \boldsymbol{b}$. Das heißt, wir können

$$L \cdot \boldsymbol{y} = \boldsymbol{b}$$

durch Vorwärtseinsetzen und dann

$$L^\intercal \cdot \boldsymbol{x} = \boldsymbol{y}$$

durch Rückwärtseinsetzen für $\boldsymbol{x}$ lösen. Es ist keine Pivotisierung notwendig, da alle Pivotelemente ungleich Null sind.[5] Eine Pivotisierung wäre bei vorliegendem Algorithmus auch nicht möglich, da jeder Zeilenaustausch die Symmetrie der Matrix zerstören würde.

Die erforderliche Rechenzeit für dieses Verfahren setzt sich aus

- N Quadratwurzeln
- $\frac{1}{6}(N^3 + 3N^2 - 4N)$ Operationen zur Zerlegung

[5] Deshalb ist auch keine Pivotisierung nötig, falls man tridiagonale Gleichungssysteme (Abschnitt 3.9) löst, deren darstellende Matrix positiv definit symmetrisch ist!

- $N^2 + N$ Operationen für das Vorwärts- bzw Rückwärtseinsetzen

zusammen. Es sind also insgesamt

$$Z_{\text{CHOLESKY}} \approx \frac{1}{6}N^3 + \frac{3}{2}N^2 + \frac{1}{3}N \overset{\text{für große } N}{\approx} \frac{1}{6}N^3$$

(wesentliche) Operationen nötig. Damit gilt also für große N

$$Z_{\text{CHOLESKY}} \approx \frac{1}{2}Z_{\text{LU}}.$$

Da beim CHOLESKY-Verfahren aufgrund der Symmetrie nur die "Hälfte" der Matrix benötigt wird (inklusive der Diagonalelemente), lässt sich durch Abspeichern in einem Vektor auch entsprechend die entsprechende Hälfte an Speicherplatz einsparen.
Man beachte insbesondere, dass die Zerlegung die Symmetrie erhält!

3.34 Beispiel.

$$A = \begin{pmatrix} 5 & 7 & 3 \\ 7 & 11 & 2 \\ 3 & 2 & 6 \end{pmatrix}$$

Schritt 1

$$\begin{aligned} l_{kk} &= \sqrt{a_{kk}^{(k-1)}} && i = k+1, \ldots, N \\ l_{ik} &= \frac{a_{ik}^{(k-1)}}{l_{kk}} && k = 1 \end{aligned}$$

Daraus ergibt sich

$$\begin{aligned} l_{11} &= \sqrt{5} \\ l_{21} &= \tfrac{7}{\sqrt{5}} \\ l_{31} &= \tfrac{3}{\sqrt{5}} \end{aligned}$$

Mit $a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik}l_{jk}$, $i, j = k+1, \ldots, N$ ergibt sich für die Koeffizienten der Matrix $A^{(1)}$

$$\begin{aligned} a_{22}^{(1)} &= a_{22} - l_{21}^2 = 11 - \tfrac{49}{5} &&= 1.2 > 0! \\ a_{32}^{(1)} &= a_{32} - l_{31}l_{21} = 2 - \tfrac{21}{5} &&= -2.2 \\ a_{33}^{(1)} &= a_{33} - l_{31}^2 = 6 - \tfrac{9}{5} &&= 4.2. \end{aligned}$$

Die Matrix $A^{(1)}$ hat also die folgende Form

$$A^{(1)} = \begin{pmatrix} 1.2 & -2.2 \\ -2.2 & 4.2 \end{pmatrix}$$

Schritt 2

$$\begin{aligned} l_{22} &= \sqrt{1.2} \\ l_{32} &= \tfrac{-2.2}{\sqrt{1.2}} \end{aligned}$$

und wir erhalten

$$a_{33}^{(2)} = 4.2 - \frac{2.2^2}{1.2} = 0.1\bar{6} > 0!$$

$\Rightarrow$ die Matrix ist positv definit

$$l_{33} = \sqrt{0.1\bar{6}} \qquad \text{Fertig!}$$

$$\Rightarrow L = \begin{pmatrix} \sqrt{5} & 0 & 0 \\ \frac{7}{\sqrt{5}} & \sqrt{1.2} & 0 \\ \frac{3}{\sqrt{5}} & \frac{-2.2}{\sqrt{1.2}} & \sqrt{0.1\bar{6}} \end{pmatrix}$$

Das Ergebnis kann durch Ausmultiplizieren überprüft werden, es ist $L \cdot L^\intercal = A$ und damit ist das Ergebnis korrekt.

3.11 Ergänzung: Die Spektralnorm

Wir suchen die natürliche Matrixnorm zur EUKLIDschen Vektornorm $\|\boldsymbol{x}\|_2$.

$$\begin{aligned}\|A\|_2 &= \max_{\|\boldsymbol{x}\|_2=1} \|A\cdot\boldsymbol{x}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} [(A\cdot\boldsymbol{x})^\mathsf{T}(A\cdot\boldsymbol{x})]^{\frac{1}{2}} \\ &= \max_{\|\boldsymbol{x}\|_2=1} [\boldsymbol{x}^\mathsf{T}A^\mathsf{T}A\boldsymbol{x}]^{\frac{1}{2}}\end{aligned}$$

- $\boldsymbol{x}^\mathsf{T}(A^\mathsf{T}\cdot A)\boldsymbol{x}$ ist eine quadratische Form
- es ist aber ebenso eine Norm, das heißt ≥ 0 für alle $A\cdot\boldsymbol{x}$ und damit auch für alle $\boldsymbol{x}$.
- $A^\mathsf{T}\cdot A$ ist symmetrisch.

$\Longrightarrow (A^\mathsf{T}\cdot A)$ ist positiv *semidefinit* (bzw. positiv definit, falls A regulär, da dann $A\cdot\boldsymbol{x}\neq 0\ \forall\boldsymbol{x}$ außer $\boldsymbol{x}=0$.)
$\Longrightarrow$ positiv semidefinite bzw. positiv definite symmetrische Matrizen $(A^\mathsf{T}\cdot A)$ haben nichtnegative bzw. positive Eigenwerte und N Eigenvektoren, die eine vollständige, orthonormierte Basis im $\mathbb{R}^N$ bilden. Das heißt

$$\begin{aligned}(A^\mathsf{T}\cdot A)\boldsymbol{x}_i &= \mu_i\boldsymbol{x}_i, \qquad \mu_i\geq 0\in\mathbb{R} \quad (\mu_i>0) \\ \boldsymbol{x}_i^\mathsf{T}\boldsymbol{x}_j &= \delta_{ij}\end{aligned}$$

$\Rightarrow$ beliebige Vektoren $\in\mathbb{R}^N$ lassen sich darstellen als

$$\boldsymbol{x}=\sum_{i=1}^N c_i x_i$$

$$\begin{aligned}\Rightarrow \boldsymbol{x}^\mathsf{T}(A^\mathsf{T}\cdot A)\boldsymbol{x} &= \left(\sum_{i=1}^N c_i x_i\right)^\mathsf{T}(A^\mathsf{T}\cdot A)\left(\sum_{j=1}^N c_j x_j\right) \\ &= \left(\sum_{i=1}^N c_i x_i\right)^\mathsf{T}\sum_{j=1}^N c_j\mu_j x_j \\ &= \sum_{i=1}^N c_i^2\mu_i\end{aligned}$$

Also gilt

$$\begin{aligned}\|A\|_2 &= \max_{\|\boldsymbol{x}\|_2=1}\left(\sum_{i=1}^N c_i^2\mu_i\right)^{\frac{1}{2}} \leq \\ &\leq \max_{\|\boldsymbol{x}\|_2=1}\left(\mu_{\max}\sum_{i=1}^N c_i^2\right)^{\frac{1}{2}} = \\ &= \sqrt{\mu_{\max}}, \qquad \text{da } \sum_{i=1}^N c_i^2=1 \text{ für } \|\boldsymbol{x}\|_2=1\end{aligned}$$

Der Maximalwert (Gleichheit) wird für den zu $\mu_{\max}$ gehörigen Eigenvektor $\boldsymbol{x}=\boldsymbol{x}_{\max}$ mit $c^\mathsf{T}=(0,\ldots,\underbrace{1}_{\text{zu }\mu_{\max}},\ldots,0)$ tatsächlich angenommen.

$$\Rightarrow \|A\|_2=\max_{\|\boldsymbol{x}\|_2=1}\|A\cdot\boldsymbol{x}\|_2=\sqrt{\mu_{\max}} \tag{3.50}$$

Hier ist $\mu_{\max}$ der maximale Eigenwert von $(A^\mathsf{T}\cdot A)$. $\|A\|_2$ heißt *Spektralnorm* und ist im Sinne der Definition die kleinste mit der EUKLIDschen Norm verträgliche Matrixnorm.

3.35 Beispiel. Sei nun A selbst symmetrisch. Dann gilt $A^\intercal \cdot A = A^2$, A^2 hat Eigenwerte $\mu_i = \lambda_i^2 \geq 0$, falls λ_i die reellen Eigenwerte von A sind. Damit gilt dann

$$\begin{aligned} \|A\|_2 &= |\lambda_{\max}|, \\ |\lambda_{\max}| &= \max_i |\lambda_i| \end{aligned}$$

Die *Spektralnorm* einer symmetrischen Matrix ist ihr *betragsmäßig größter Eigenwert.*

Es bleibt die Frage zu klären, wie sich $\|A^{-1}\|_2$ berechnen lässt. (Dies wird zur Berechnung der Konditionszahl benötigt).

- A muss regulär sein, sonst ist die Frage sinnlos.
- Mit Gleichung (3.50) gilt $\|A^{-1}\|_2 = \sqrt{\xi_{\max}}$, wenn $\xi_{\max}$ der größte Eigenwert von $(A^{-1})^\intercal \cdot A^{-1} = (A \cdot A^\intercal)^{-1}$ ist.

  NR.: $AA^{-1} = 1 = (AA^{-1})^\intercal = (A^{-1})^\intercal A^\intercal \Rightarrow (A^\intercal)^{-1} = (A^{-1})^\intercal$
- Die Eigenwerte von $(A \cdot A^\intercal)^{-1}$ sind die reziproken Eigenwerte von $(A \cdot A^\intercal)$. Man beachte, dass $A \cdot A^\intercal$ hier positiv definit ist, da A regulär sein muss. Damit sind alle Eigenwerte *echt größer* 0.

  NR.: $B \cdot x_i = b_i x_i \Rightarrow x_i = B^{-1} \cdot b_i x_i$

  $\Rightarrow B^{-1} x_i = \dfrac{1}{b_i} x_i$ für $b_i \neq 0$

Die Matrix $AA^\intercal$ ist eine Ähnlichkeitstransformation von $A^\intercal A$ (siehe Kapitel 4), da $A^{-1}(AA^\intercal)A = A^\intercal \cdot A$, und hat damit die gleichen Eigenwerte. Insgesamt gilt also

$$\begin{aligned} \|A^{-1}\|_2 &= \sqrt{\xi_{\max}} = \max(\text{Eigenwerte von } (A^\intercal A)^{-1}) \\ &= \frac{1}{\sqrt{\mu_{\min}}}, \qquad \text{wenn } \mu_{\min} \text{ der kleinste Eigenwert von } A^\intercal A \text{ ist} \end{aligned} \tag{3.51}$$

3.36 Beispiel. Sei nun A selbst symmetrisch

$$\|A^{-1}\|_2 = \frac{1}{|\lambda_{\min}|}, \qquad |\lambda_{\min}| = \min_i |\lambda_i|$$

Insgesamt ist also für die Konditionszahl einer regulären Matrix A bezüglich der Spektralnorm

$$\kappa_2(A) = \sqrt{\frac{\mu_{\max}}{\mu_{\min}}}, \tag{3.52}$$

wenn μ_i die Eigenwerte von $(A^\intercal \cdot A)$ sind. Falls A symmetrisch ist, gilt

$$\kappa_2(A) = \frac{|\lambda_i|_{\max}}{|\lambda_i|_{\min}}, \tag{3.53}$$

wenn λ_i die Eigenwerte von A sind.

3.37 Beispiel (Konditionszahl $\kappa_2(A)$ der Matrix aus Beispiel 3.13).

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

A ist symmetrisch, d.h. die Eigenwerte von A bestimmen die Koordinationszahl. Die Eigenwerte werden aus dem charakteristischen Polynom berechnet.

$$\begin{aligned} \det(A - \lambda_i \mathbb{1}) &= 0, \qquad \text{d.h.} \\ (10^{-4} - \lambda_{1,2})(1 - \lambda_{1,2}) - 1 &= 0 \\ \Rightarrow \lambda_{1,2}^2 - \lambda_{1,2}(1 + 10^{-4}) - (1 - 10^{-4}) &= 0 \\ \lambda_1 &= 1.6180616 \\ \lambda_2 &= -0.617962 \\ \Rightarrow \kappa_2(A) = \frac{|\lambda_1|}{|\lambda_2|} &= 2.6183044, \qquad \text{zu vergleichen mit} \\ \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty &= 4.0004 \end{aligned}$$

Die Ergebnisse sind natürlich verschieden, haben aber die gleiche Größenordnung!

Kapitel 4

Eigenwertprobleme

4.1 Motivation, Grundbegriffe und Lösungsstrategien

In diesem Kapitel wenden wir uns der numerischen Behandlung von Eigenwertproblemen zu, insbesondere der Berechnung von Eigenwerten *reeller* Matrizen. Bzgl. der Berechnung der zugehörigen Eigenvektoren werden wir uns relativ kurz fassen. Vorweg sei auf die Bibliotheksroutinen des Programmpaketes EISPACK [1] und anderer Implementierungen (hauptsächlich IMSL und NAG) verwiesen, die für praktisch alle denkbaren Probleme die entsprechenden Algorithmen bereitstellen.

4.1.1 Das charakteristische Polynom: Problematik der numerischen Lösung

Zunächst wollen wir kurz motivieren, warum die auf der Hand liegende Methode zur Berechnung von Eigenwerten, d.h. die Berechnung der Nullstellen des charakteristischen Polynoms, bei ihrer numerischen Umsetzung mit großen Problemen behaftet ist.

Wir betrachten dazu die Eigenwertgleichung für eine Matrix A $\in \mathbb{R}^{N\times N}$ oder $\mathbb{C}^{N\times N}$,

$$A \cdot \boldsymbol{x}_k = \lambda_k \cdot \boldsymbol{x}_k \tag{4.1}$$

λ_k ist der Eigenwert zum Eigenvektor $\boldsymbol{x}_k$, und obige Gleichung ist äquivalent zum Gleichungssystem

$$(A - \lambda_k \mathbb{1})\, \boldsymbol{x}_k = \mathbf{0}$$

Da $\boldsymbol{x}_k$ nicht der Nullvektor sein soll, muss die Matrix $(A-\lambda_k\mathbb{1})$ singulär sein, d.h. ihre Determinante verschwinden! Der Eigenwert λ_k ergibt sich also als Nullstelle des charakteristischen Polynoms

$$P(\lambda)_A = \det(A - \lambda\mathbb{1}) = p_N\lambda^N + p_{N-1}\lambda^{N-1} + \cdots + p_1\lambda^1 + p_0 \tag{4.2}$$

Das Polynom $P(\lambda)$ ist von *echtem Grad* N, wobei für seine Koeffizienten Folgendes gilt

$$p_N = (-1)^N \tag{4.2a}$$
$$p_{N-1} = (-1)^{N-1}(a_{11} + a_{22} + \cdots + a_{NN}) = Spur(A)\cdot(-1)^{N-1} \tag{4.2b}$$
$$p_0 = \det(A) \tag{4.2c}$$

Die Nullstellen des charakteristischen Polynoms $P(\lambda)_A =: 0$ sind die gesuchten N Eigenwerte, die nicht notwendigerweise verschieden sein müssen. Gleiche Eigenwerte (von Nullstellen mit Vielfachheit > 1) heißen entartet.

Die zugehörigen Eigenvektoren ergeben sich durch Einsetzen der λ_k in Gl. (4.1). Die Matrix $(A-\lambda_k\mathbb{1})$ ist dann singulär und hat zumindest einen von Null verschiedenen Vektor als Lösung der homogenen Gleichung.

[1] basierend auf Wilkinson & Reinsch, 1971, Linear Algebra, vol. II of Handbook for Automatic Computation (New York, Springer)

- In der Physik treten Eigenwertprobleme hauptsächlich bei quantenmechanischen Fragestellungen auf, und als
- *Einfache Lösungsidee* liegt sofort auf der Hand, zunächst das charakteristische Polynom und dann seine Nullstellen entsprechend Kapitel 2.2.5 zu berechnen.
- Wie allerdings schon in Kapitel 2.2.4 angesprochen, kann es dabei zu einer Vielzahl von Problemen kommen, wobei dasjenige von eng benachbarten Eigenwerten (= Nullstellen) im Folgenden näher beleuchtet werden soll.

Problematik: Zunächst ist festzuhalten, dass schon die Koeffizienten des Polynoms Rundungsfehlern unterliegen (aufgrund der schon erwähnten Eingangsfehler und insbesondere der Tatsache, dass auch sie *berechnet* werden müssen). Deshalb löst man nicht bzgl. der Nullstellen des eigentlichen Polynoms, sondern bzgl. eines leicht unterschiedlichen:

$$P^*(\lambda) = (-1)^N \lambda^N + p^*_{N-1}\lambda^{N-1} + p^*_{N-2}\lambda^{N-2} + \cdots + p^*_1\lambda + p^*_0$$

wobei

$$p^*_j = p_j + \epsilon q_j \quad , \quad j = 0, \ldots, N-1, \qquad \epsilon \text{ klein.}$$

Frage: Wie hängen die Nullstellen λ^* von den Fehlern in p_j ab? (Man vergleiche mit dem analogen Problem, dass sich im Rahmen der Lösung von linearen Gleichungssystemen stellt, vgl. Kap. 3.8.4.

Sei nun $Q(\lambda) = q_{N-1}\lambda^{N-1} + \cdots + q_1\lambda + q_0$ das sog. "Störpolynom", und damit

$$P^*(\lambda) = P(\lambda) + \epsilon Q(\lambda). \tag{4.3}$$

Annahme: Des weiteren nehmen wir an, dass λ_k eine einfache Nullstelle von $P(\lambda)$ sei und die Nullstellen von P^* stetig von ϵ abhängen mögen.

Dann ergibt sich mit Hilfe des Newton-Raphson-Verfahrens (Kap. 2.2.3), dass sich die gesuchten Nullstellen aufgrund der fehlerhaften Darstellung der Polynomkoeffizienten wie folgt ändern:

$$\Delta\lambda_k = -\frac{P^*(\lambda_k)}{P^{*\prime}(\lambda_k)} = \frac{-\epsilon Q(\lambda_k)}{P'(\lambda_k) + \epsilon Q'(\lambda_k)}, \qquad \text{da } P(\lambda_k) = 0.$$

Zusätzlich sei $|\epsilon Q'(\lambda_k)| \ll |P'(\lambda_k)|$, d.h. ϵ sehr klein (und $P'(\lambda_k) \neq 0$, da es sich um eine einfache Nullstelle handelt)

$$\Rightarrow \quad \lambda^*_k \approx \lambda_k - \epsilon\frac{Q(\lambda_k)}{P'(\lambda_k)}. \tag{4.4}$$

Der obige Quotient wird als "Konditionszahl" der Nullstelle bezeichnet und wird groß, wenn die Ableitung des Polynoms bei λ_k, $P'(\lambda_k)$, klein wird. Dies gilt insbesondere, wenn mehrere Nullstellen eng benachbart sind.

4.1 Beispiel (Charakteristisches Polynom im Falle $N = 3$).

$$N = 3, \qquad A = \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix}$$

$$\begin{aligned} P(\lambda)_A &= \det(A - \lambda\mathbb{1}) = (a_1 - \lambda)(a_2 - \lambda)(a_3 - \lambda) = \\ &= \underbrace{-\lambda^3}_{(4.2a)} + \lambda^2 \underbrace{(a_1 + a_2 + a_3)}_{(4.2b)} - \lambda(a_1a_2 + a_2a_3 + a_1a_3) + \underbrace{a_1a_2a_3}_{(4.2c)} \end{aligned}$$

Man beachte, dass sich das gleiches Polynom für alle diejenigen Matrizen ergibt, die die gleichen Eigenwerte haben, d.h. die "ähnlich" zu A sind (siehe Kap. 4.1.3). Für die Ableitung ergibt sich

$$P'(\lambda) = -3\lambda^2 + 2\lambda(a_1 + a_2 + a_3) - (a_1a_2 + a_2a_3 + a_1a_3),$$

und ausgewerten an den Nullstellen/Eigenwerten, $P'(\lambda = \lambda_k)$, finden wir

$$\begin{aligned} P'(a_1) &= -a_1^2 + a_1a_2 + a_1a_3 - a_2a_3 \\ P'(a_2) &= -a_2^2 + a_1a_2 + a_2a_3 - a_1a_3 \\ P'(a_3) &= -a_3^2 + a_1a_3 + a_2a_3 - a_1q_2. \end{aligned}$$

Die Nullstellen selbst seien eng benachbart, und ohne Einschränkung der Allgemeinheit sei dabei

$$a_1 = a_2 - \delta, \qquad a_3 = a_2 + \delta$$

$$\begin{aligned} \Rightarrow \quad P'(a_1) &= -2\delta^2 \\ P'(a_2) &= \delta^2 \\ P'(a_3) &= -2\delta^2 \end{aligned}$$

d.h. $\lambda_k^* = \lambda_k - \epsilon Q(\lambda_k)/(c_k\delta^2)$ mit $c_k = (-2, 1, -2)$ für $k = 1, 2, 3$.

Sei nun $\lambda_2 = 1$, d.h. $\lambda_1 = 1 - \delta$ und $\lambda_3 = 1 + \delta$.
$\Rightarrow \quad P(\lambda) = -\lambda^3 + 3\lambda^2 - (3 - \delta^2)\lambda + (1 - \delta^2)$.

Es werde nur ein Koeffizient gestört, in unserem Beispiel der zweite ($\rightarrow Q(\lambda) = p_2\lambda^2$), wobei die relative Störung $\epsilon = 10^{-4}$ betrage; des weiteren sei $\delta = 0.1$.

$$\begin{aligned} \Rightarrow \quad P(\lambda) &= -\lambda^3 + 3\lambda^2 - 2.99\lambda + 0.99 \\ P^*(\lambda) &= -\lambda^3 + 3.0003\lambda^2 - 2.99\lambda + 0.99 \end{aligned}$$

Es ergibt sich damit folgende Vorhersage für die numerisch berechneten Eigenwerte, die sich aufgrund der Eingangsstörung von p_2 ergeben

$$\begin{aligned} \lambda_1^* &\approx \lambda_1 - \epsilon\frac{Q(\lambda_1)}{(-2\delta^2)} \quad \rightarrow \quad 0.9 - 10^{-4}\,\frac{3 \cdot 0.9^2}{-2 \cdot 0.01} \\ &= 0.9 + 10^{-4} \cdot 121.5 = 0.912 \\ \lambda_2^* &= \lambda_2 - \epsilon\frac{3 \cdot 1^2}{0.01} = 1 - 10^{-4} \cdot 300 = 0.97 \\ \lambda_3^* &= \lambda_3 - \epsilon\frac{3 \cdot 1.1^2}{-2 \cdot 0.01} = 1.1 + 10^{-4} \cdot 181.5 = 1.118 \end{aligned}$$

Zum Vergleich: Die numerische Berechnung der Nullstellen des gestörten Polynoms ergibt

λ_1^*	=	0.916	Vorhersage:	0.912
λ_2^*	=	0.969		0.970
λ_3^*	=	1.115		1.118

d.h. unsere Vorhersage trifft tatsächlich zu. Obwohl obiges Beispiel ein auf den ersten Blick noch "gutartiges" Problem darstellt, finden wir aufgrund eines relativen Fehlers von 10^{-4} in nur einem Koeffizienten einen Fehler von mehreren Prozent in allen resultierenden Nullstellen!

- *Man beachte*: Die Empfindlichkeit auf Störungen wächst mit dem Grad des Polynoms, N!
- *Konsequenz*: Man vermeide die Berechnung der Eigenwerte mit Hilfe des charakteristischen Polynoms.

4.1.2 Einge Grundbegriffe

In diesem Kapitel wollen wir einige für das weitere Vorgehen wichtige Begriffe und Zusammenhänge aus der linearen Algebra rekapitulieren. Falls

$$\begin{aligned} A = A^\intercal, &\qquad \text{d.h. } a_{ij} = a_{ji}, &&\qquad \text{ist } A \text{ symmetrisch} \\ A = A^\dagger, &\qquad \text{d.h. } a_{ij} = a^*_{ji}, &&\qquad \text{ist } A \text{ HERMITEsch (selbstadjungiert)} \end{aligned}$$

(Der Stern "*" bedeutet hier und im Folgenden die komplexe Konjugation.) Falls

$$\begin{aligned} A^\intercal \cdot A = A \cdot A^\intercal = \mathbb{1}, &\qquad \text{ist } A \text{ orthogonal} \\ A^\dagger \cdot A = A \cdot A^\dagger = \mathbb{1}, &\qquad \text{ist } A \text{ unitär} \\ A^\dagger \cdot A = A \cdot A^\dagger, &\qquad \text{ist } A \text{ normal} \end{aligned}$$

Demzufolge sind symmetrisch reelle und hermitesche Matrizen normal! Für *reellwertige* Matrizen A sind die Eigenschaften

- *hermitesch* und *symmetrisch* bzw.
- *unitär* und *orthogonal*

gleichbedeutend. Obige Eigenschaften (hermitesch, normal) der Matrizen haben weitreichende Konsequenzen für ihre Eigenwerte und -vektoren:

- *reell symmetrische* und *hermitesche* Matrizen besitzen ausschließlich reelle Eigenwerte.
- *normale* Matrizen besitzen einen vollständigen und orthogonal Satz von Eigenvektoren bezüglich eines Vektorraumes der Dimension N (falls $A \in \mathbb{C}^{N\times N}, \mathbb{R}^{N\times N}$) (ggf. lassen sich die Eigenvektoren mit dem GRAM-SCHMIDT Verfahren orthogonalisieren).
- Eine Matrix, die spaltenweise aus solch einem Satz von orthonormalen Eigenvektoren gebildet ist, ist unitär:

$$\begin{gathered} ((e_1)\cdots(e_N)) = B \\ B^\dagger = \begin{pmatrix} (\tilde{e}_1) \\ \vdots \\ (\tilde{e}_N) \end{pmatrix} \qquad (\tilde{e}_1) = e_1^\dagger \quad \text{(Zeilenvektor)} \\ \Rightarrow \quad \left(B^\dagger \cdot B\right)_{ij} = \underbrace{e_i^\dagger \cdot e_j}_{(*)} \overset{\text{Orthonormalität}}{=} \delta_{ij} \end{gathered}$$

  $(*)$: Skalarprodukt bezüglich komplexer Zahlen

- Eine Matrix, die spaltenweise aus den (normalisierten) Eigenvektoren einer reellen, symmetrischen Matrix gebildet ist, ist orthogonal, da die entsprechenden Eigenvektoren reell sind.

Im Weiteren werden wir uns nur mit *reelle* Matrizen befassen. Falls diese symmetrisch sind, haben sie ausschließlich reelle Eigenwerte und Eigenvektoren; fall sie nicht symmetrisch sind, können sowohl reelle als auch komplexe Eigenwerte und Eigenvektoren vorliegen.

4.1.3 Strategie zur Eigenwert-Bestimmung

Die wichtigsten Algorithmen zur Bestimmung von Eigenwerten reellwertiger Matrizen sind durch das Jacobi-Verfahren (Kap. 4.2, für symmetrische Matrizen) und den QR- (oder QL-) Algorithmus (Kap. 4.4, für allgemeine reelle Matrizen) gegeben. Beide beruhen auf dem wichtigen Begriff der *Ähnlichkeitsabbildung*. Im Folgenden geben wir einen kurzen Überblick über die prinzipielle Strategien beider Algorithmen.

Ähnlichkeitsabbildungen

4.2 Definition. *Zwei Matrizen A, B heißen ähnlich, falls eine reguläre Matrix C existiert, so dass*

$$C^{-1} \cdot A \cdot C = B \tag{4.5}$$

4.3 Lemma. *Ähnlichkeitstransformationen erhalten die Eigenwerte!*

4.4 Beweis. *Sei Z regulär und A' ähnlich zu A:*

$$A' = Z^{-1}AZ$$

Dann lautet das dazugehörige charakteristische Polynom

$$\begin{aligned} P(\lambda)_{A'} &= \det\left(Z^{-1}AZ - \lambda\mathbb{1}\right) \\ &= \det\left(Z^{-1}\left(A - \lambda\mathbb{1}\right)Z\right) \\ &= \det\left(Z^{-1}\right)\det\left(A - \lambda\mathbb{1}\right)\det\left(Z\right). \end{aligned}$$

Mit $\det\left(Z^{-1}\right) = \det^{-1}\left(Z\right)$ *finden wir damit*

$$P(\lambda)_{A'} = \det\left(A - \lambda\mathbb{1}\right) = P(\lambda)_A \qquad \square.$$

4.5 Satz (Hauptachsentheorem). *Zu jeder reellen symmetrischen Matrix A existiert eine orthogonale Matrix U, so dass sich A über eine Ähnlichkeitsabbildung bezüglich U auf Diagonalgestalt transformieren lässt, d.h.*

$$U^{-1} \cdot A \cdot U = D \tag{4.6}$$

wobei die Spalten von U die Eigenvektoren von A und die Elemente der Diagonalmartix D die dazugehörigen Eigenwerte sind.

Dieser Satz beruht darauf, dass reelle symmetrischen Matrizen einen vollständigen Satz von Eigenvektoren haben.
→ Multiplikation von (4.6) mit U

$$A \cdot U = U \cdot D,$$

ausgeschrieben für die Spalten $\boldsymbol{u}_k$, $k = 1, \ldots N$ ergibt

$$A \cdot \boldsymbol{u}_k = \boldsymbol{u}_k \cdot \lambda_k \qquad \text{mit} \quad \lambda_k = D_{kk}$$

Dies ist aber nichts anders als die Eigenwertgleichung für den Eigenwert/-vektor k!

Prinzipielle Strategie des Jacobi-Verfahrens. Um die Eigenwerte einer symmetrischen Matrix zu bestimmen, bildet man eine Folge von elementaren und *orthogonalen* Ähnlichkeitsabbildungen P_i (den sog. JAKOBI-Rotationen)

$$A \quad \to \quad P_1^{-1}AP_1 \quad \to \quad P_2^{-1}(P_1^{-1}AP_1)P_2 \quad \to \quad \ldots \quad \text{etc.}$$

bis Diagonalgestalt erreicht ist. Die (wiederum orthogonale) Matrix

$$U = P_1 \cdot P_2 \ldots P_{\text{last}}$$

ist dann die gesuchte Matrix, deren Spalten die Eigenvektoren von A sind.

QR-Transformation

Für allgemeine (nicht symmetrische) Matrizen gilt der folgende

4.6 Satz (Satz von Schur für reelle Matrizen). *Zu jeder reellen Matrix A existiert eine orthogonale Matrix U, so dass die zu A ähnliche Matrix*

$$\tilde{R} = U^{-1}AU$$

die Quasidreicksgestalt

$$\tilde{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & \dots & R_{1m} \\ 0 & R_{22} & R_{23} & \dots & R_{2m} \\ 0 & 0 & R_{33} & \dots & R_{3m} \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & 0 & R_{mm} \end{pmatrix} \quad , \quad m \leq N \tag{4.7}$$

hat und die Matrizen R_{ii}, $i = 1, \dots m$ entweder die Ordnung 1 oder 2 haben und im letzten Fall ein Paar von konjugiert komplexen Eigenwerten besitzen.

Die Eigenwerte sind dann aus den Untermatrizen R_{ii} direkt ablesbar oder leicht zu berechnen. (Man erinnere sich z.B., dass die Eigenwerte einer Δ-Matrix durch ihre Diagonalelemente definiert sind.)

Strategie des QR-Algorithmus. Man erzeuge auf geschickte Weise die Quasidreiecksmatrix $\tilde{R}$. Dies wird durch sogenannte QR-Transformationen *iterativ* (und schnell konvergent) erzielt:

(1.) Zerlege $A_k = Q_k \cdot R_k$, wobei Q orthogonal und R eine (echte) Rechts-Δ-Matrix ist.

(2.) QR-Transformation:

$$A_{k+1} = R_k Q_k = \left(Q_k^{-1} A_k\right) Q_k, \qquad \text{Ähnlichkeitsabbildung!}$$

(3.) Zerlege wiederum, iteriere

$$A_{k+1} = Q_{k+1} R_{k+1}, \qquad A_{k+2} = R_{k+1} Q_{k+1} \tag{4.8}$$

$\Rightarrow A \xrightarrow{k\to\infty} \tilde{R}$

4.2 Reelle symmetrische Matrizen: Das JACOBI-Verfahren

Das im Folgenden vorgestellte Verfahren zur Bestimmung der Eigenwerte/-vektoren von reellen symmetrischen Matrizen wurde erstmalig von JACOBI (1846) angewendet und hat folgende Eigenschaften:

- Es ist einfach und stabil, aber
- langsamer als das entsprechende QR-Verfahren.
- Es besteht aus einer Abfolge von *orthogonalen Ähnlichkeitstransformationen*, den sog. "JACOBI-Rotationen", bis Diagonalstruktur erreicht ist.
- Diese Struktur wird allerding nicht in einer endlicher Zahl von Rotationen erreicht, sondern nur bis auf vorgegebene Toleranz ϵ bzgl. der verbleibenden Größe der Nebendiagonalelemente.

4.2.1 Die JACOBI-Rotation

Die sog. "Rotationsmatrix" $U(p,q,\varphi)$ ist wie folgt definiert:

$$U(p,q,\varphi) = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & c & \cdots & s & & & \leftarrow p \\ & & & 1 & & & & \\ & & \vdots & \ddots & \vdots & & & \\ & & & & 1 & & & \\ & & -s & \cdots & c & & & \leftarrow q \\ & & & & & \ddots & & \\ & & & & & & 1 & \end{pmatrix} \quad \text{mit } 1 \le p < q \le N, \qquad (4.9)$$

$$\begin{matrix} & & \uparrow & & \uparrow & & & \\ & & p & & q & & & \end{matrix}$$

d.h. sie ist eine Einheitsmatrix bis auf die Elemente $u_{pp} = u_{qq} = c$, $u_{pq} = s$, $u_{qp} = -s$, mit $s = \sin\varphi$ und $c = \cos\varphi$. U ist orthogonal (d.h., $U^\intercal U = \mathbb{1}$), da $c^2 + s^2 = 1$.

Wir wollen nun untersuchen, wie sich *eine* Ähnlichkeitstransformation unter Verwendung obiger Rotationsmatrix auf die Ausgangsmatrix A auswirkt,

$$A'' = U^\intercal A U = A' U \qquad \text{"JACOBI-Rotation".}$$

- Dazu betrachen wir zunächst die Linksmultiplikation mit $U^\intercal$,

$$A' = U^\intercal A$$

.

$$\underbrace{\begin{pmatrix} 1 & & & & \\ & c & & -s & \\ & & 1 & & \\ & s & & c & \\ & & & & 1 \end{pmatrix}}_{U^\intercal} \cdot \begin{pmatrix} - & - & - \\ & A & \\ - & - & - \end{pmatrix} \begin{matrix} \\ \leftarrow p \\ \\ \leftarrow q \\ \\ \end{matrix}$$

$$\begin{matrix} & \uparrow & \uparrow & & \\ & p & q & & \end{matrix}$$

$a'_{ij} = a_{ij}$ für $i \neq p, q$, und für die Zeilen p, q gilt

$$\left.\begin{aligned} a'_{pj} &= a_{pj} \cdot c - a_{qj} \cdot s \\ a'_{qj} &= a_{pj} \cdot s + a_{qj} \cdot c \end{aligned}\right\} \quad j = 1, \ldots N. \qquad (4.10)$$

$\Rightarrow$ Die *Zeilen* p, q werden geändert, die "neuen" Zeilen sind Linearkombinationen der "alten".

- Die anschließende Rechtsmultiplikation von A' mit U,

$$A'' = A' U$$

wirkt sich folgendermaßen aus:

$$\begin{pmatrix} | & & | \\ | & & | \\ | & A' & | \\ | & & | \\ | & & | \end{pmatrix} \cdot \underbrace{\begin{pmatrix} 1 & & & & \\ & c & & s & \\ & & 1 & & \\ & -s & & c & \\ & & & & 1 \end{pmatrix}}_{U} \begin{matrix} \\ \\ \leftarrow p \\ \\ \leftarrow q \end{matrix}$$

$$\begin{matrix} \uparrow & \uparrow \\ p & q \end{matrix}$$

$a''_{ij} = a'_{ij}$ für $j \neq p, q$, und für die Spalten p, q gilt

$$\left.\begin{aligned} a''_{ip} &= a'_{ip} \cdot c - a'_{iq} \cdot s \\ a''_{iq} &= a'_{ip} \cdot s + a'_{iq} \cdot c \end{aligned}\right\} \quad i = 1, \ldots N \tag{4.11}$$

$\Rightarrow$ Die *Spalten* p, q werden geändert, die "neuen" Spalten sind Linearkombinationen der "alten".

- Insgesamt gilt also, dass die Zeilen p und q durch $U^\intercal \cdot A$ und die Spalten p und q durch $A' \cdot U$ verändert werden. Die Kreuzungspunkte werden durch beide Teilprozesse affektiert (siehe Abbildung 4.1).

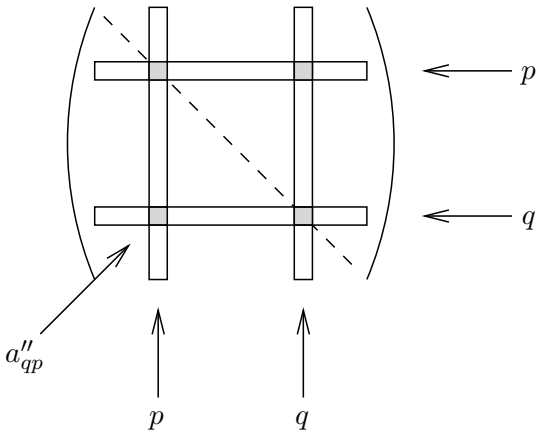


Abbildung 4.1: Auswirkung *einer* JACOBI-Rotation auf die Zeilen und Spalten einer Matrix.

Man beachte, dass A'' symmetrisch bleibt, denn

$$(A'')^\intercal = \left(U^{-1}AU\right)^\intercal = (U^\intercal AU)^\intercal = U^\intercal A^\intercal U = U^\intercal AU = A'' \qquad \square.$$

An den Kreuzungspunkten gilt (Beweis durch Einsetzen)

$$\begin{aligned} a''_{pp} &= a_{pp}c^2 + a_{qq}s^2 - 2csa_{pq} \\ a''_{qq} &= a_{pp}s^2 + a_{qq}c^2 + 2csa_{pq} \\ a''_{pq} &= a''_{qp} = cs\left(a_{pp} - a_{qq}\right) + a_{pq}\left(c^2 - s^2\right) \end{aligned} \tag{4.12}$$

$$\Rightarrow \quad a''_{pp} + a''_{qq} = a_{pp} + a_{qq}, \tag{4.13}$$

d.h. die Summe der Elemente auf den *diagonalen* Kreuzungspunkten bleibt erhalten (Erhaltung der Spur unter Ähnlichkeitstransformationen)!

- Auf Grund der erhaltenen Symmetrie muss die Transformation nur unterhalb und auf der Diagonalen durchgeführt werden. Damit sind für eine Rotation ca. $4N$ wesentliche Operationen notwendig.
- Die JACOBI-Rotation wird auch (p, q)-Rotation genannt.

4.2.2 Nullsetzen eines Nebendiagonalelements

Mit Hilfe einer Serie obiger JACOBI-Rotationen sollen nun bestimmte Nebendiagonalelemente a''_{qp} $(= a''_{pq})$ sukzessive auf Null gesetzt werden, um letztendlich die Matrix zu diagonalisieren.

- Die Auswahl der entsprechenden Indizes (p, q) wird dabei später diskutiert werden.
- Ebenfall muss natürlich (später) gezeigt werden, dass das Nullsetzen eines Elementes nicht zu einer (extremen) Vergrößerung von schon zuvor auf Null gesetzten Elementen führen darf.
- *Wichtig* ist insbesondere eine numerisch stabile Durchführung der entsprechenden Rotation, die wir in diesem Abschnitt besprechen werden.

Im Weiteren wollen wir also diejenige Transformation ableiten (d.h. die entsprechende Rotationsmatrix bestimmen), die das Element a''_{qp} auf Null setzt!

$$a''_{qp} =: 0 = cs\,(a_{pp} - a_{qq}) + a_{qp}\left(c^2 - s^2\right)$$

$$\Rightarrow \quad \frac{c^2 - s^2}{cs} = \frac{a_{qq} - a_{pp}}{a_{qp}}$$

Wir definieren nun

$$\frac{a_{qq} - a_{pp}}{2a_{qp}} = \frac{c^2 - s^2}{2cs} = \Theta \quad (\hat{=} \cot 2\varphi) \tag{4.14}$$

und des weiteren

$$y = \frac{c}{s} \quad (\hat{=} \cot \varphi) \tag{4.15}$$

$$\Rightarrow \quad \Theta = \frac{\frac{c^2}{s^2} - 1}{2\frac{c}{s}} = \frac{y^2 - 1}{2y}$$

$$\Rightarrow \quad y^2 - 1 - 2y\Theta = 0$$

$$y_{12} = \frac{2\Theta \pm \sqrt{4\Theta^2 + 4}}{2} = \Theta \pm \sqrt{\Theta^2 + 1}$$

Für den Tangens des gesuchten Drehwinkels, der a''_{qp} auf Null setzt, ergibt sich also

$$t = \tan\varphi = \frac{1}{y} = \frac{1}{\Theta \pm \sqrt{\Theta^2 + 1}}.$$

Um t numerisch stabil zu berechnen, wählen wir das Vorzeichen so, dass keine numerische *Auslöschung* auftritt:

$$t = \frac{1}{\underbrace{\Theta + \operatorname{sign}(\Theta)}_{(*)}\sqrt{\Theta^2 + 1}} \qquad \text{für } \Theta \neq 0 \tag{4.16a}$$

($(*)$: $\Theta + \sqrt{\ldots}$ für $\Theta > 0$ bzw. $\Theta - \sqrt{\ldots}$ für $\Theta < 0$)

Mit den folgenden Bedingungen fangen wir die "Sonderfälle" ab

$$t = 1 \qquad \text{für } \Theta = 0 \tag{4.16b}$$

$$t = \frac{1}{2\Theta} \qquad \text{für } \Theta^2 > \text{größte darstellbare Zahl auf dem Computer} \tag{4.16c}$$

Die letztere Bedingung tritt dann auf, wenn a_{qp} selbst schon sehr nahe bei Null liegt, d.h. die Ausgangsmatrix kaum gedreht werden muss. Mit diesen Vereinbarungen liegen t und φ im Bereich

$$\Rightarrow \quad -1 < t \leq 1 \quad \text{, d.h.} \quad -\frac{\pi}{4} < \varphi \leq \frac{\pi}{4}$$

was sich in Kap. 4.2.3b als wichtig herausstellen wird.

Wenn t berechnet ist, ergeben sich die Werte von c, s (mit $t = \frac{s}{c}$, $t^2 = \frac{s^2}{c^2} = \frac{1-c^2}{c^2} = \frac{1}{c^2} - 1$ folgendermaßen

$$c = \frac{1}{\sqrt{1+t^2}} \tag{4.17a}$$

$$s = t \cdot c \tag{4.17b}$$

und

$$\boxed{a''_{qp} = 0} \tag{4.18a}$$

wird explizit gesetzt. Um nun die vollständige Transformation von A unter Verwendung der gefundenen Rotationsmatrix (wiederum numerisch stabil!) durchzuführen, müssen wir die transformierten Zeilen und Spalten inklusive der diagonalen Kreuzungspunkte ($a''_{qp} = a''_{pq} = 0$ aufgrund Konstruktion) berechnen. Mit Gl. (4.12) hatten wir

$$\begin{aligned} a''_{pp} &= c^2 a_{pp} + s^2 a_{qq} - 2csa_{qp} \\ a''_{qp} &= 0 = (c^2 - s^2)a_{qp} + sc(a_{pp} - a_{qq}) \end{aligned}$$

$$\Rightarrow \quad a_{qq} = \frac{c^2 - s^2}{sc} a_{qp} + a_{pp}$$

$$\begin{aligned} \Rightarrow \quad a''_{pp} &= c^2 a_{pp} + s^2 \left(\frac{c^2 - s^2}{sc} a_{qp} + a_{pp} \right) - 2csa_{qp} \\ &= a_{pp} + a_{qp} \left(\underbrace{\frac{\left(c^2 - s^2\right) s}{c} - 2cs}_{*} \right) \end{aligned}$$

Nebenrechnung für *:

$$\begin{aligned} * = t(c^2 - s^2) - 2cs &= tc^2 - ts^2 - 2cs \\ \text{mit } (s = tc) \qquad &= tc^2 - t^3 c^2 - 2tc^2 \\ &= -tc^2 - t^3 c^2 = -t(c^2 + t^2 c^2) \\ &= -t(c^2 + s^2) = -t! \end{aligned}$$

Also: Der "obere" Kreuzungspunkt transformiert sich mittels

$$\boxed{a''_{pp} = a_{pp} - t \cdot a_{qp}} \tag{4.18b}$$

und der "untere"

$$a''_{qq} \overset{(4.13)}{=} a_{pp} + a_{qq} - a''_{pp}, \quad \text{d.h.}$$
$$\boxed{a''_{qq} = a_{qq} + t \cdot a_{qp}} \tag{4.18c}$$

Für die *Zeilen* p, q (geändert über $U^\intercal A$, vgl. Gl. (4.10)) finden wir

$$a'_{pj} = a_{pj}c - a_{qj}s$$
$$a'_{qj} = a_{pj}s + a_{qj}c$$

und unter Verwendung von

$$r = \frac{s}{1+c} \qquad (\hat{=} \tan\frac{\varphi}{2}) \tag{4.19}$$

ergibt sich

$$\begin{aligned} a'_{pj} &= a_{pj}c + a_{pj}(1-c) - a_{qj}s - a_{pj}(1-c) \\ &= a_{pj} - s\left(a_{qj} + a_{pj}\left(\frac{1-c}{s}\right)\right) \end{aligned}$$

Nebenrechnung:

$$\frac{1-c}{s} = \frac{(1-c)(1+c)}{s(1+c)} = \frac{1-c^2}{s(1+c)} = \frac{s^2}{s(1+c)} = r$$

$$\boxed{a'_{pj} = a_{pj} - s\left(a_{qj} + r a_{pj}\right)} \tag{4.20a}$$

Diese Formulierung ist auch dann numerisch stabil, wenn φ klein ist. Analog ergibt sich

$$\boxed{a'_{qj} = a_{qj} + s\left(a_{pj} - r a_{qj}\right)} \tag{4.20b}$$

Letztendlich transformieren sich die *Spalten* p, q (geändert über $A'U$, vgl. (4.11))

$$\boxed{a''_{ip} = a'_{ip} - s\left(a'_{iq} + r a'_{ip}\right)} \tag{4.21a}$$

$$\boxed{a''_{iq} = a'_{iq} + s\left(a'_{ip} - r a'_{iq}\right)} \tag{4.21b}$$

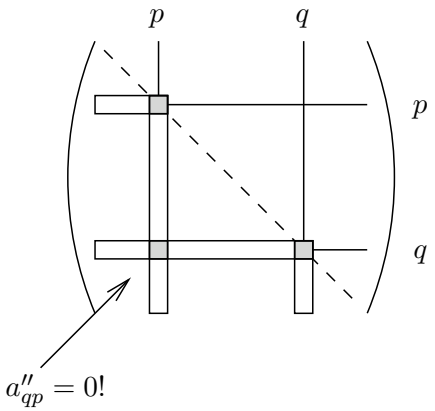
wobei die Indizes in (4.20, 4.21) im Prinzip über $j = 1, \ldots N$ und $i = 1, \ldots N$ laufen. Da wir in Gl. 4.18 schon die Kreuzungspunkte explizit berechnet haben, müssen wir aufgrund der erhaltenen Symmetrie nur noch die in Abb. 4.2 gekennzeichneten Bereiche für die Zeilen und Spalten p, q unterhalb der Diagonale transformieren.

Zusammenfassung. Wähle p, q mit $a_{qp} \neq 0$

$$\Rightarrow \quad \Theta = \frac{a_{qq} - a_{pp}}{2a_{qp}} \quad \overset{(4.16)}{\rightarrow} t \quad \overset{(4.17)}{\rightarrow} c, s \quad \overset{(4.19)}{\rightarrow} r$$

- $a''_{qp} = 0$
- a''_{pp}, a''_{qq} (4.19) Diagonalelemente von A''
- a'_{pj}, a'_{qj}, a''_{ip}, a''_{iq} (4.20, 4.21) Zeilen und Spalten von A''

Damit ist eine JACOBI*-Rotation, die* a''_{qp} *auf Null setzt, abgearbeitet.*



$$
\begin{array}{lll}
\rightarrow \text{Zeilen} & a'_{pj} & j = 1, \ldots p-1 \\
& a'_{qj} & j = 1, \ldots p-1 \\
& & j = p+1, \ldots q-1 \\
\rightarrow \text{Spalten} & a''_{ip} & i = p+1, \ldots q-1 \\
& & i = q+1, \ldots N \\
& a''_{iq} & i = q+1, \ldots N
\end{array}
$$

Abbildung 4.2: Tatsächlich zu transformierende Elemente (Symmetrieerhaltung)

4.2.3 Auswahl des Nebendiagonalelements, Konvergenz

Bisher ist noch nicht klar, welche Strategie anzuwenden ist, um die Indizes p, q für das auf Null zu setzende Nebendiagonalelement auszuwählen, und in wie weit man mit einer Serie von Transformationen tatsächlich *alle* Nebendiagonalelemente zum Verschwinden bringen kann (bis auf eine vorgegebene Toleranz ϵ). In diesem Abschnitt werden wir nun beide Fragen klären.

a) "Klassisches" Verfahren

Das sog. "klassische" JACOBI-Verfahren wurde vom "Erfinder" eingeführt und eignet sich insbesondere dazu, eine Matrix "von Hand" zu diagonalisieren [2]. Es besteht aus einer Folge von Transformationen mit

$$A^{(k)} = U_k^{\mathsf{T}} A^{(k-1)} U_k, \quad k = 1, 2 \ldots \qquad \text{und} \quad A^{(0)} = A \tag{4.22}$$

Die Auswahl des auf Null zu setzenden Elementes erfolgt dabei dergestalt, dass man im k-ten Schritt das betragsmäßig größte Element der Matrix $A^{(k-1)}$ verwendet,

$$|a_{qp}^{(k-1)}| = \max_{i>j} |a_{ij}^{(k-1)}|. \tag{4.23}$$

Dann garantiert der folgende Satz, dass man solch ein Verfahren tatsächlich zur Diagonalisierung verwenden kann.

4.7 Satz. *Die Folge (4.22) mit $a_{qp}^{(k-1)}$ aus (4.23) konvergiert gegen eine Diagonalmatrix D!*

4.8 Beweis. *Zum Beweis dieses Satzen betrachten wir zunächst die Summe der Quadrate der Nebendiagonalelemente,*

$$S\left(A^{(k)}\right) = \sum_i \sum_{j \neq i} \left(a_{ij}^{(k)}\right)^2 \quad k = 1, 2, \ldots \tag{4.24}$$

Damit lautet die Kurzfassung des obigen Satzes, dass

$$S\left(A^{(k)}\right) \quad \rightarrow \quad 0 \quad \textit{für} \quad k \rightarrow \infty \tag{4.25}$$

Schritt (i): In einem ersten Schritt spalten wir diese Summe (bzgl. einer schon rotierten Matrix A'') in mehrere Teilsummen auf, entsprechend der in Abb. 4.3 skizzierten Struktur.

[2] Die entsprechende computergestützte Vorgehensweise werder wir im nächsten Abschnitt diskutieren

$$
\begin{aligned}
S\left(A''\right) &= \sum_i \sum_{j\neq i} a_{ij}''^2 \qquad (4.26)\\
&= \underbrace{\sum_{i\neq p,q} \sum_{j\neq i,p,q} a_{ij}''^2}_{\text{unverändert}} + \underbrace{\sum_{i\neq p,q} \left(a_{ip}''^2 + a_{iq}''^2\right)}_{\text{Spalte p,q}}\\
&+ \underbrace{\sum_{j\neq p,q} a_{pj}''^2 + a_{qj}''^2}_{\text{Zeile p,q}} + \underbrace{2a_{qp}''^2}_{\text{Kreuzungspunkte p,q 2mal}}
\end{aligned}
$$

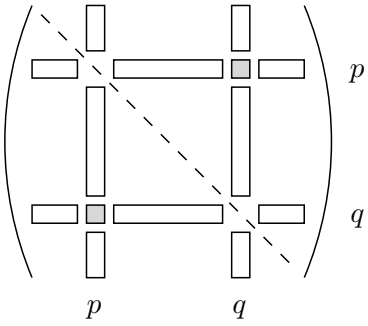


Abbildung 4.3: Struktur der Matrix A''; Man vergleiche mit Gl. 4.26 bzgl. der Teilsummen für die Spalten p, q, Zeilen p, q und Kreuzungspunkte $(p, q) = (q, p)$.

Schritt (ii) Um das Verhalten dieser Teilsummen untersuchen zu können, verwenden wir im Weiteren die Tatsache, dass orthogonale Transformationen längenerhaltend sind ($\|.\|^2$, EUKLID*ische Norm).*

- *Insbesondere erhält die erste Transformation, $A' = U^\intercal \cdot A$, die Spaltenlänge*

$$\left(U^\intercal\right)\left(A \quad (*) \quad\right) \qquad * \hat{=} \text{ k-te Spalte.}$$

  Dazu betrachten wir die k-te Spalte von $U^\intercal A \hat{=} U^\intercal \boldsymbol{a}_k$

$$\|U^\intercal \boldsymbol{a}_k\|^2 = (U^\intercal \boldsymbol{a}_k)^\intercal \cdot (U^\intercal \boldsymbol{a}_k) = \boldsymbol{a}_k^\intercal U U^\intercal \boldsymbol{a}_k = \|\boldsymbol{a}_k\|^2$$

- *Die zweite Transformation, $A'' = A' \cdot U$, erhält die Zeilenlänge*

$$\begin{pmatrix} (*) \\ A' \end{pmatrix}\begin{pmatrix} U \end{pmatrix} \qquad * \hat{=} \text{ k-te Zeile.}$$

  Dazu betrachten wir die k-te Zeile von $A'U \hat{=} (\boldsymbol{a'})_k\, U$, wobei $(\boldsymbol{a'})_k$ ein Zeilenvektor ist.

$$\|(\boldsymbol{a'}_k)U\|^2 = (\boldsymbol{a'}_k)U \cdot \underbrace{\left((\boldsymbol{a'}_k)U\right)^\intercal}_{\text{Spaltenvektor}} = (\boldsymbol{a'}_k)UU^\intercal(\boldsymbol{a'}_k)^\intercal = \|(\boldsymbol{a'}_k)\|^2$$

Für die erste Transformation $A' = U^\intercal \cdot A$ finden wir damit Folgendes ...

- $\sum_i a_{ij}'^2 = \sum_i a_{ij}^2 \quad \forall j$ *Erhaltung der Spaltenlänge*
- *Aber: in dieser Transformation werden nur die Zeilen p, q verändert,*

$$\Rightarrow \quad a_{pj}'^2 + a_{qj}'^2 = a_{pj}^2 + a_{qj}^2 \quad \forall j$$

- *Die anschließende Transformation 2 $(A'U)$ verändert die Spalten $j = p, q$ und erhält* **nicht** *die Spaltenlänge der veränderten Spalten.*

$$
\begin{aligned}
&\Rightarrow \quad a_{pj}''^2 + a_{qj}''^2 = a_{pj}'^2 + a_{qj}'^2 \quad \forall j \text{ außer } p, q\\
&\Rightarrow \quad a_{pj}''^2 + a_{qj}''^2 = a_{pj}^2 + a_{qj}^2 \quad \forall\ j \neq p, q \qquad (4.27)
\end{aligned}
$$

... und für die zweite Transformation $A'' = A' \cdot U$ gilt

- $\sum_j a_{ij}''^2 = \sum_j a_{ij}'^2 \quad \forall i$ *Erhaltung der Zeilenenlänge*

- *Aber: in dieser Tansformation werden nur die Spalten p, q verändert,*

$$\Rightarrow \quad a_{ip}''^2 + a_{iq}''^2 = a_{ip}'^2 + a_{iq}'^2 \quad \forall i$$

- *Die vorhergehende Transformation 1 ($A \to A' = U^\intercal A$) hat die Zeilen $i = p, q$ verändert und dort* **nicht** *die Zeilenlänge erhalten.*

$$\begin{aligned} \Rightarrow \quad & a_{ip}'^2 + a_{iq}'^2 = a_{ip}^2 + a_{iq}^2 \quad \forall\ i \text{ außer } p, q \\ \Rightarrow \quad & a_{ip}''^2 + a_{iq}''^2 = a_{ip}^2 + a_{iq}^2 \quad \forall\ i \neq p, q \end{aligned} \tag{4.28}$$

Zusammengefasst finden wir also:
$\forall$ Spalten außer p und q gilt, dass die Spaltenlänge unverändert ist, insbesondere $(a_{pj}''^2 + a_{qj}''^2)$ erhalten bleibt.
$\forall$ Zeilen außer p und q gilt also, dass die Zeilenlänge unverändert ist, insbesondere $(a_{ip}''^2 + a_{iq}''^2)$ erhalten bleibt.

Im Hinblick auf die Zerlegung $S(A'')$ (4.26) verändert sich also nur der (zweifache) Anteil des Kreuzungspunktes auf der Nebendiagonale, und wir erhalten für die Veränderung der Summe der Quadrate der Nebendiagonalelemente

$$S(A'') = \underbrace{S(A) - 2a_{qp}^2}_{\text{unveränderter Anteil}} + \underbrace{2a_{qp}''^2}_{=0} \tag{4.29}$$

bzw. generell (in Abhängigkeit vom Index k)

$$S\left(A^{(k)}\right) = S\left(A^{(k-1)}\right) - 2a_{qp}^{(k-1)^2} \quad , \quad k = 1, 2, \ldots \tag{4.30}$$

Als finale Frage stellt sich nun, ob S eine Nullfolge ist, wie es der Satz behauptet. Auf Grund der Wahl ("klassisches Verfahren", Gl. 4.23)

$$|a_{qp}^{(k-1)}| = \max_{j>i} |a_{ij}^{(k-1)}|$$

gilt sicherlich

$$S\left(A^{(k-1)}\right) \leq (N^2 - N)\left(a_{qp}^{(k-1)}\right)^2 \tag{4.31}$$

(der negative Anteil "$-N$" entspricht der Korrektur für die Diagonale), d.h.

$$S\left(A^{(k)}\right) = S\left(A^{(k-1)}\right) - 2\left(a_{qp}^{(k-1)}\right)^2 \leq S\left(A^{(k-1)}\right) \cdot \left(1 - \frac{2}{N^2 - N}\right). \tag{4.32}$$

Da diese Abschätzung unabhängig von $a_{qp}^{(k-1)}$ ist und $\forall k = 1, 2, \ldots$ gilt, folgt sofort

$$S\left(A^{(k-1)}\right) \leq \left(1 - \frac{2}{N^2 - N}\right)^k S\left(A^{(0)}\right). \tag{4.33}$$

Im Falle von

- $N = 2$: $S\left(A^{(1)}\right) = 0$
  genügt also eine Rotation, um die Matrix "exakt" zu diagonalisieren, während für

- $N > 2$: $\quad 1 - \frac{2}{N^2 - N} < 1$
  $S\left(A^{(k)}\right)$ *tatsächlich eine Nullfolge für $k \to \infty$ ist.*

□.

Anmerkungen.

- Die Konvergenz des "klassischen" JACOBI-Verfahrens ist mindestens linear (vgl. Kap. 1).
- Sobald die Nebendiagonalelemente hinreichend klein sind, konvergiert das Verfahren sogar quadratisch.
- Die Fehler der Eigenwerte sinken mit $\sqrt{S\left(A^{(k)}\right)}$.
- Die Gesamtrechenzeit lässt sich nach oben mit $\approx 4N^3 \ln(1/\epsilon)$ abschätzen, wenn $\dfrac{S\left(A^{(k)}\right)}{S\left(A^{(0)}\right)} \leq \epsilon^2$.

  Bezeichnen wir mit $N_N = \frac{N^2-N}{2}$ die Zahl der unteren Nebendiagonalelemente, folgt aus der Konvergenzabschätzung (4.33)

  $$\left(1 - \frac{1}{N_N}\right)^k \leq \epsilon^2,$$

  d.h. die Zahl der notwendigen Schritte, um $S(A^{(k)})$ um einen Faktor ϵ^2 gegenüber dem Startwert zu reduzieren, lässt sich mit

  $$k \gtrsim N_N \cdot 2 \ln \frac{1}{\epsilon} \approx N^2 \ln \frac{1}{\epsilon} \quad \text{für } N^2 \gg N \tag{4.34}$$

  nähern. Damit ergibt sich (bei $4N$ Operationen pro Schritt) die oben angegebene Rechenzeit. Diese Abschätzung ist allerdings sehr pessimistisch (aufgrund der Annahme einer anhaltenden linearen Konvergenz), wie das nachfolgende Beispiel 4.9 zeigt.

4.9 Beispiel (Klassisches JACOBI-Verfahren). Bestimmung der Eigenwerte einer 3×3 Matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 5 & 6 \end{pmatrix}.$$

Numerische Rechnung in `single precision`, geforderte Genauigkeit $\epsilon = 10^{-6}$. Das Verfahren erreicht diese Genauigkeit nach 7 Rotationen, während die Abschätzung (4.34) einen (viel zu pessimistischen) Wert $k < 69$ voraussagt.

```
 0.10000E+01
 0.20000E+01   0.30000E+01
 0.30000E+01   0.40000E+01   0.50000E+01

  rotation no  0  resulted in sum =   58.0000000

 ----------------------------------------------------
  new rotation with p,q =  2 3   4.0000000

 0.10000E+01
-0.26983E+00  -0.12311E+00
 0.35954E+01   0.00000E+00   0.81231E+01

  rotation no  1  resulted in sum =   25.9999981

 ----------------------------------------------------
  new rotation with p,q =  1 3   3.5954406

-0.49926E+00
-0.24904E+00  -0.12311E+00
 0.00000E+00  -0.10385E+00   0.96224E+01

  rotation no  2  resulted in sum =    0.1456120

 ----------------------------------------------------
  new rotation with p,q =  1 2  -0.2490415

-0.62327E+00
 0.00000E+00   0.89782E-03
-0.46288E-01  -0.92961E-01   0.96224E+01
```

```
  rotation no  3  resulted in sum =    2.1568717E-02

 ----------------------------------------------------
  new rotation with p,q =  2 3  -9.2961356E-02

-0.62327E+00
-0.44716E-03  -0.27774E-06
-0.46285E-01   0.00000E+00   0.96233E+01

  rotation no  4  resulted in sum =    4.2850906E-03

 ----------------------------------------------------
  new rotation with p,q =  1 3  -4.6285477E-02

-0.62348E+00
-0.44716E-03  -0.27774E-06
 0.00000E+00   0.20199E-05   0.96235E+01

  rotation no  5  resulted in sum =    3.9990820E-07

 ----------------------------------------------------
  new rotation with p,q =  1 2  -4.4715771E-04

-0.62348E+00
 0.00000E+00   0.42962E-07
 0.14486E-08   0.20199E-05   0.96235E+01

  rotation no  6  resulted in sum =    8.1596084E-12

 ----------------------------------------------------
  new rotation with p,q =  2 3   2.0198520E-06

-0.62348E+00
-0.30405E-15   0.42962E-07
 0.14486E-08   0.00000E+00   0.96235E+01

  rotation no  7  resulted in sum =    4.1971277E-18

 ----------------------------------------------------
 converged with S =    4.1971277E-18 , eigenvalues are

-0.6234754   4.2961577E-08   9.6234760
```

b) Zyklisches Verfahren

- Die Abnahme der Summe $S(A^{(k)})$ ist maximal beim klassischen Verfahren, und
- bei Rechnung von "Hand" ($\rightarrow$ JACOBI, 1846) ist dieses Verfahren mit Sicherheit das zweckmäßigste.
- Allerdings erfordert es $N_N = \frac{1}{2}(N^2 - N)$ Vergleichsoperationen, um die jeweiligen Maxima $|a_{qp}|$ zu finden. Während diese Operationen bei Rechnung von "Hand" durch bloßes "Hinsehen" und ohne zusätzlichen Rechenaufwand "im Kopf" durchgeführt werden können, sind sie bei einer Computerrechnung im Vergleich zu den $4N$ Multiplikationen pro Drehung viel zu "teuer"!

Deshalb findet in diesem Fall das sog. "zyklische" Verfahren Anwendung, mit einem fest vorgegeben Zyklus, d.h. ohne zusätzliche Abfragen. Ein zweckmäßiger Zyklus ist dabei durch eine Abfolge von Rotationen mit Nullsetzung der folgenden Indexpaare p, q gegeben:

$$\begin{aligned}(p,q) \;=\; & (1,2),\ (1,3),\ (1,4),\ldots(1,N) \\ & (2,3),\ (2,4),\ldots(2,N) \\ & \vdots \\ & (N-1,N).\end{aligned} \tag{4.35}$$

Falls während eines Zyklus zufälligerweise ein bestimmter Wert $a_{qp} = 0$ sein sollte, verwendet man einfach das nächste Indexpaar.

4.10 Satz. *Das zyklische Verfahren konvergiert gegen eine Diagonalmatrix, wenn der Drehwinkel im Bereich*

$$-\frac{\pi}{4} < \varphi \leq \frac{\pi}{4}$$

liegt.

Letztere Bedingung ist auf Grund Gl. (4.16, Berechnung von t) gewährleistet.

- Nach einem vollen Zyklus mit N_N Rotationen gilt dann (für nichtentartete Eigenwerte)

$$S\left(A^{(k+N_N)}\right) \leq \frac{S\left(A^{(k)}\right)^2}{2\delta^2}, \tag{4.36}$$

  d.h. das Verfahren konvergiert *quadratisch*, *wenn* der minimale Abstand zweier Eigenwerte durch $\min_{i\neq j}|\lambda_i - \lambda_j| = 2\delta > 0$ gegeben ist und $S\left(A^{(k)}\right) < \frac{1}{4}\delta^2$ im k-ten Schritt erziehlt wurde.

- Diese quadratische Konvergenz setzt schon nach wenigen Zyklen ein, und normalerweise sind 6-8 Zyklen zur Diagonalisierung ausreichend.

- Man beachte, dass das Verfahren um so schneller konvergiert, je größer die Separation der Eigenwerte ist! (Grund: früheres Einsetzen der quadratischen Konvergenz und schnellere Konvergenz $\sim \delta^{-2}$)

4.11 Beispiel (Zyklisches JACOBI-Verfahren). Bestimmung der Eigenwerte der 3×3 Matrix aus Bsp. 4.9

Man beachte, dass insgesamt 3 Zyklen mit jeweils $N_N = 3$ Rotationen benötigt werden, um S (die Summe der Quadrate der Nebendiagonalelemente) auf ca. 10^{-23} zu bringen. Man beachte auch das Einsetzen der quadratischen Konvergenz nach Zyklus 2.

```
 0.10000E+01
 0.20000E+01   0.30000E+01
 0.30000E+01   0.40000E+01   0.50000E+01

  cycle no  0  resulted in sum =   58.0000000
 ---------------------------------------------------
-0.27430E+00
-0.31050E+00  -0.34088E+00
-0.28642E+00   0.00000E+00   0.96152E+01

  cycle no  1  resulted in sum =    0.3568898
 ---------------------------------------------------
-0.28953E-04
-0.42397E-02  -0.62345E+00
 0.79212E-04   0.00000E+00   0.96235E+01

  cycle no  2  resulted in sum =    3.5963367E-05
 ---------------------------------------------------
-0.12103E-06
-0.44338E-11  -0.62348E+00
-0.23308E-18   0.00000E+00   0.96235E+01

  cycle no  3  resulted in sum =    3.9316506E-23
 ---------------------------------------------------
 converged with S =    3.9316506E-23 , eigenvalues are
  -1.2102767E-07  -0.6234751   9.6234751
```

4.2.4 Berechnung der Eigenvektoren

Nach Konvergenz der Verfahrens (klassisch oder zyklisch) haben wir also die Ausgangsmatrix diagonalisiert,

$$D = V^\intercal \cdot A \cdot V$$

wobei $V = U_1 \cdot U_2 \cdot U_3 \ldots$

- Da umgekehrt $A \cdot V = V \cdot D$ gilt, (vergleiche Gl. (4.6 ff)), stellen die Spalten von V die Eigenvektoren von A dar. Da des weiteren V orthogonal ist (per Konstruktion), d.h. $V^\intercal \cdot V = \mathbb{1}$ und die Eigenvektoren reell sind (A symmetrisch), sind die Eigenvektoren *orthonomiert*.
- Die Qualität der Eigenvektoren ist allerdings schlechter als diejenige der Eigenwerte (d.h. die Eigenwertgleichung wird ggf. nicht numerisch "exakt" erfüllt).
- Man konstruktiert die Eigenvektoren mittels

$$V_0 = \mathbb{1} \quad , \quad V_k = V_{k-1} \cdot U_k \quad , \quad k = 1, 2, \ldots \tag{4.37}$$

- und die Gesamtrechnenzeit zur Bestimmung der Eigenwerte und Eigenvektoren skaliert (für das zyklische Verfahren) mit Z $\approx 32\ N^3$ *pro Zyklus*.

4.3 GIVENS-Rotationen

Wie sich im Weiteren herausstellen wird (insbesondere im Rahmen des QR-Algorithmus), lässt sich die Eigenwertaufgabe numerisch einfacher lösen, wenn die zu diagonalisierende Matrix zuvor (durch orthogonale Ähnlichkeitsabbildungen!) auf eine "geeignete" Form gebracht wurde.

Unsymmetrische Matrizen sollen dabei auf sog. obere HESSENBERG-Form H transformiert werden,

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \ldots & h_{1(N-1)} & h_{1N} \\ h_{21} & h_{22} & h_{23} & & h_{2(N-1)} & h_{2N} \\ 0 & h_{32} & h_{33} & & & \vdots \\ \vdots & & h_{43} & & & \\ \vdots & & \vdots & & & \vdots \\ 0 & & 0 & 0 & h_{N(N-1)} & h_{NN} \end{pmatrix}, \tag{4.38}$$

also eine obere Dreicksmatrix inklusive der ersten Unterdiagonalen.

Symmetrische Matrizen werden auf *Tridiagonal*gestalt bebracht. Da orthogonale Ähnlichkeitsabbildungen die Symmetrie erhalten, ist die resultierende Tridiagonalmatrix ebenfalls symmetrisch.

- Im Vergleich zur *allgemeinen* Tridiagonalmatrix (siehe auch Kap. 3.9),

$$\begin{pmatrix} b_1 & c_1 & 0 & & & 0 \\ a_2 & b_2 & c_2 & & & \\ & & & \ddots & & \\ 0 & & & & a_N & b_N \end{pmatrix}$$

- lautet die Nomenklatur für die Elemente einer *symmetrischen* Tridiagonalmatrix J,

$$J = \begin{pmatrix} \alpha_1 & \beta_1 & & & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & & & \\ & \beta_2 & \alpha_3 & \beta_3 & & & \\ & & & \ddots & & & \\ & & & & \beta_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ 0 & & & & & \beta_{N-1} & \alpha_N \end{pmatrix}. \tag{4.39}$$

Die gewünschte Transformation auf H bzw. J kann durch

- GIVENS-Rotationen ($\rightarrow$ hier behandelt) und
- HOUSEHOLDER-Transformationen[3]

erziehlt werden.

4.3.1 Transformation auf HESSENBERG-Form

Die Idee des hier vorgestellten Algorithmus ist es, eine Folge von Rotationen druchzuführen, durch die jeweils ein Element *unterhalb der ersten Subdiagonale* auf Null gebracht wird. *Nachfolgende Rotationen sollen diese Null(en) erhalten!*

- Die Transformation auf HESSENBERG-Form lässt sich in einer *endlichen* Zahl von Operationen durchführen.
- Im Gegensatz zur JACOBI-Rotation darf das auf Null zu setzende Element *nicht* im außendiagonalen Kreuzungspunkt liegen.

Das Vorgehen besteht darin, eine Sequenz von Rotationen,

$$P_{23}P_{24}\dots P_{2N};\quad P_{34}P_{35}\dots P_{3N};\quad P_{45}\dots P_{4N};\quad \dots\quad ;\quad P_{(N-1)N} \tag{4.40}$$

durchzuführen, wobei der Drehwinkel durch Nullsetzen der Matrixelemente unterhalb der Nebendiagonalen bestimmt wird:

$$a_{j(i-1)} \rightarrow 0 \quad \text{bei Rotation } P_{ij} \tag{4.41}$$

Wenn alle Drehungen ausgeführt sind, ist Transformation auf HESSENBERG-Form erreicht.

Im Folgenden werden wir das Verfahren etwas näher beleuchten. Die **erste Sequenz** besteht darin, die betreffenden Elemente $a_{i1},\quad i = 3,\dots N$ der ersten Spalte auf Null zu setzen.

$$\begin{array}{cc} & \\ & \\ P_{24} & P_{23} \\ \rightarrow & \rightarrow \\ & \rightarrow \\ \rightarrow & \end{array}
\begin{pmatrix}
 & \downarrow & & \downarrow & \\
 & \downarrow & \downarrow & & \\
a_{11} & a_{12} & a_{13} & a_{14} & \dots \\
a_{21} & a_{22} & a_{23} & a_{24} & \\
\boxed{0} & a_{32} & a_{33} & a_{34} & \\
\boxed{0} & a_{42} & a_{43} & a_{44} & \dots \\
\vdots & & & \vdots &
\end{pmatrix}
\begin{array}{l} \\ \\ P_{24} \rightarrow a_{41} \rightarrow 0 \\ P_{23} \rightarrow a_{31} \rightarrow 0 \\ \\ \\ \end{array}$$

Die Rotationen $P_{2i},\ i = 3,\dots N$ modifizieren die Elemente a_{i1} der ersten Spalte nur über die Zeilenoperationen in Zeile 2 und Zeile i!

Drehung P_{23}

$$\begin{aligned}
Z_2' &= cZ_2 - sZ_3 && \text{Zeile Z, vergleiche (4.10)} \\
Z_3' &= sZ_2 + cZ_3 && \\
S_2'' &= cS_2' - sS_3' && \text{Spalte S, vergleiche (4.11)} \\
S_3'' &= sS_2' + cS_3' &&
\end{aligned}$$

Der Drehwinkel $c, s = \cos\varphi, \sin\varphi$ wird dabei durch das Nullsetzen von a_{31}

$$a_{31}' = sa_{21} + ca_{31} := 0$$

[3] siehe z.B. "Numerical Recipes", Kap. 11.2

mit $c^2 + s^2 = 1$ bestimmt.

$$c = \frac{|a_{21}|}{\sqrt{a_{21}^2 + a_{31}^2}} \quad , \quad s = \frac{-\text{sign}(a_{21})a_{31}}{\sqrt{a_{21}^2 + a_{31}^2}} \tag{4.42}$$

$(\Rightarrow \quad -\frac{\pi}{2} \leq \varphi \leq \frac{\pi}{2})$

Es gilt, dass $a''_{31} = a'_{31} = 0$, da die Spaltenoperationen nur die Spalten (2,3) modifizieren.

DrehungP_{24}

$$\left.\begin{array}{rcl} Z_2 & \rightarrow & cZ_2 - sZ_4 \\ Z_4 & \rightarrow & sZ_2 + cZ_4 \end{array}\right\} Z_2 \text{ (und teilweise } Z_4\text{) durch Drehung } P_{23} \text{ modifiziert}$$

$$\left.\begin{array}{rcl} S_2 & \rightarrow & cS_2 - sS_4 \\ S_4 & \rightarrow & sS_2 + cS_4 \end{array}\right\} S_2 \text{ (und teilweise } S_4\text{) durch Drehung } P_{23} \text{ modifiziert}$$

c und s ergeben sich durch Nullsetzen von a_{41},

$$a''_{41} = sa_{21} + ca_{41} := 0$$

(man beachte, dass a_{21} durch die vorhergende Drehung P_{23}) modifiziert wurde,

... und so weiter, bis die Drehung P_{2N} abgeschlossen ist. Insgesamt gilt für die Drehwinkel zum Abarbeiten der ersten Spalte

$$c = \frac{|a_{21}|}{\sqrt{a_{21}^2 + a_{j1}^2}} \quad , \quad s = \frac{-\text{sign}(a_{21})a_{j1}}{\sqrt{a_{21}^2 + a_{j1}^2}} \quad j = 3, \ldots N. \tag{4.43}$$

Man beachte, dass z.B. a_{21} durch jede Drehung verändert wird!

Mit der **zweiten Sequenz** setzten wir jetzt die Elemente $a_{i2}, \quad i = 4, \ldots N$ der zweiten Spalte sukzessive auf Null.

$$\begin{array}{cc} & \\ & \\ P_{35} & P_{34} \\ & \\ \rightarrow & \rightarrow \\ & \rightarrow \\ \rightarrow & \end{array} \quad \begin{array}{c} \\ \\ \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \ldots \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} & \\ 0 & \boxed{0} & a_{43} & a_{44} & a_{45} & \\ \vdots & \boxed{0} & a_{53} & a_{54} & a_{55} & \ldots \\ & \vdots & \vdots & & \vdots & \end{pmatrix} \end{array} \quad \begin{array}{l} \\ \\ P_{35} \rightarrow a_{52} \rightarrow 0 \\ P_{34} \rightarrow a_{42} \rightarrow 0 \end{array}$$

DrehungP_{34}

$$\begin{array}{rcl} Z_3 & \rightarrow & cZ_3 - sZ_4 \\ Z_4 & \rightarrow & sZ_3 + cZ_4 \end{array}$$

(analog für Spalte (3, 4)). Der entscheidene Punkt ist nun, dass durch diese Operation *die zuvor (erste Sequenz) auf Null gesetzten Elemente a_{31}, a_{41} diesen Wert behalten!*

$$a'_{42} = sa_{32} + ca_{42} := 0$$

usw.

Für die **dritte Spalte** bilden wir mittels der Rotationen $P_{4j}, \quad j = 5, \ldots N$ Linearkombinationen von Zeile 4 und Zeile j (und für Spalte 4 und j), und die Nullen von Spalte (1,2) (ab Zeile 3 bzw. 4) bleiben Null.

Insgesamt gilt also für den Drehwinkel, um das Element a_{ij} auf Null zu setzen ($i = j+2, \dots N$)

$$c = \frac{|a_{j+1,j}|}{\sqrt{a_{j+1,j}^2 + a_{ij}^2}} \quad , \quad s = \frac{-\text{sign}(a_{j+1,j})a_{ij}}{\sqrt{a_{j+1,j}^2 + a_{ij}^2}} \tag{4.44}$$

wobei das jeweils aktuelle $a_{j+1,j}$ verwendet wird. Dieses wird durch die entsprechende Drehung natürlich wiederum modifiziert.

Man beachte, dass für die Elimination des Elementes a_{ij} der Spalte j *in dieser Spalte* "nur" das Element $a_{j+1,j}$ verändert wird (immer wieder, man beachte die horizontalen Pfeile in den beiden vorhergehenden Skizzen).

Rechenaufwand. Insgesamt sind $N^* = \frac{1}{2}(N-1)(N-2)$ dieser sog. GIVENS-Rotationen durchzuführen. Dabei werden, pro Rotation, folgende (wesentliche) Operationen benötigt:

- Um das Element a_{ij} der j-ten Spalte zu eliminieren, müssen für die Neuberechnung von $a_{j+1,j}$ 4 Multiplikationen druchgeführt und 1 Quadratwurzel gezogen werden.
- Danach sind $4(N-j)$ Operationen zur Bildung der entsprechenden Linearkombinationen der Zeilen erforderlich: Die Spalten $(1, \dots j-1)$ bleiben unverändert (Linearkombinationen von "0") und die aktuelle Spalte ist durch die Neuberechnung von $a_{j+1,j}$ schon abgedeckt.
- Schließlich sind noch (in Analogie zur JACOBI-Rotation, hier gibt es nichts zu "sparen") $4N$ Multiplikationen für die Spalten $j+1$ und i durchzuführen.

Aufsummation über alle Spalten ergibt insgesamt folgende Zahl von Operationen, die notwendig sind, um eine allgemeine Matrix auf HESSENBERG-Form zu bringen.

$$\begin{aligned} Z_{\text{HESSENBERG}} &= \frac{10}{3}N^3 - 8N^2 + \frac{2}{3}N + 4 \quad \text{Multiplikationen} \\ N^* &= \frac{1}{2}(N-1)(N-2) \qquad \text{Quadratwurzeln} \end{aligned}$$

Eigenvektoren. Nach Transformation auf HESSENBERG-Form kann mit dem noch zu besprechenden QR-Algorithmus die Eigenwertaufgabe für H gelöst werden, und wir finden die Eigenwerte von H (diese sind natürlich diejenigen von A) und die Eigenvektoren von H. Die zugehörigen Eigenvektoren von A ergeben sich wie folgt:

$$H = U_{N^*}^\intercal U_{N^*-1}^\intercal \dots U_2^\intercal U_1^\intercal A U_1 U_2 \dots U_{N^*} = Q^\intercal A Q, \quad Q = U_1 U_2 \dots U_{N^*} \tag{4.45}$$

Aus der originalen Eigenwertaufgabe folgt

$$A \cdot \boldsymbol{x} = \lambda \cdot \boldsymbol{x} \quad \Rightarrow \quad Q^\intercal A \boldsymbol{x} = Q^\intercal \lambda \boldsymbol{x},$$

und da Q aufgrund Konstruktion orthogonal ist, ergibt sich

$$Q^\intercal A \underbrace{(Q Q^\intercal)}_{\mathbb{1}} \boldsymbol{x} = Q^\intercal \lambda \boldsymbol{x} = \lambda Q^\intercal \boldsymbol{x}$$

d.h.

$$H \cdot (Q^\intercal \cdot \boldsymbol{x}) = \lambda \, (Q^\intercal \cdot \boldsymbol{x}) \,. \tag{4.46}$$

Wenn $\boldsymbol{x}_j$ die Eigenvektoren von A sind, dann sind $\boldsymbol{y}_j = (Q^\intercal \cdot \boldsymbol{x}_j)$ die entsprechenden Eigenvektoren von H. Falls also die Eigenwertaufgabe bezüglich H gelöst ist (z.B. mit dem QR-Algorithmus), berechnen sich die Eigenvektoren von A zu

$$\boldsymbol{y}_j = Q^\intercal \cdot \boldsymbol{x}_j \quad \Rightarrow \quad \boldsymbol{x}_j = Q \cdot \boldsymbol{y}_j = U_1 U_2 \dots U_{N^*} \cdot \boldsymbol{y}_j$$

Für die Berechnung der Eigenvektoren ist das *Abspeichern der Transformationsmatrix* (Q oder $Q^\intercal$ erforderlich.

4.3.2 Transformation auf Tridiagonalgestalt

Im Falle einer symmetrischen Matrix A wird die gleiche Folge von Ähnlichkeitstransformationen (4.40) auf eine symmetrische Matrix angewandt. Auf Grund der Symmetrieerhaltung muss die resultierende Matrix wiederum symmetrisch sein. Eine

$$\text{Symmetrische HESSENBERG-Form} = \text{Tridiagonalmatrix (symmetrisch)}$$

Die unterliegende Symmetrie erspart natürlich einen entsprechenden Rechenaufwand, da nur unterhalb und auf der Diagonalen "gearbeitet" werden muss.

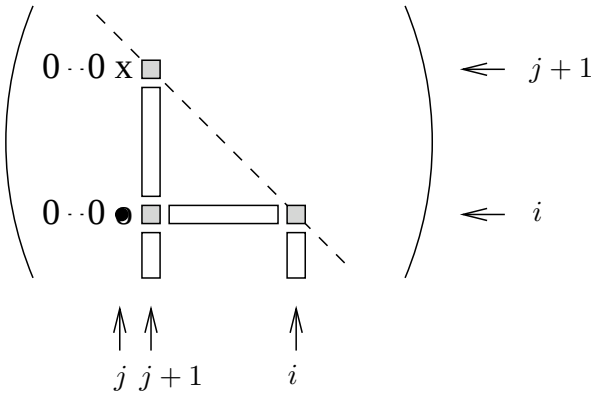


Abbildung 4.4: Zur Transformation von symmetrischen Matrizen auf Tridiagonalgestalt.

Anhand der Skizze 4.4 geben wir kurz die Operationen an, die notwendig sind, um ein Element a_{ij} ($\bullet$) in Spalte j, Zeile i, $i \geq j+2$ zu eliminieren.

1) Man berechne zunächst den erforderlichen Drehwinkel (wie in (4.44)),

2) und modifiziere dann die Zeile $j+1$. Hier sind nur das Element $a_{j+1,j}$ ($\times$) und der diagonale Kreuzungspunkte $a_{j+1,j+1}$ betroffen (im allgemeinen Fall hatten wir hier auch alle restlichen Spalten $j, \ldots, N$ zu betrachten).

3) In Zeile i werden die Elemente $a_{i,k}, \quad k = j+1, \ldots i$ (bis zur Diagonalen) modifiziert, und dann

4) in Spalte $j+1$ die Elemente von der Diagonalen bis zur Zeile N.

5) Schließlich ist noch die Spalte i, wiederum von der Diagonalen bis zur Zeile N zu bearbeiten.

Insgesamt haben wir also (vgl. Abb. 4.4) das Element $a_{j+1,j}(\times)$, 3 Kreuzungspunkte und die Zeilen und Spalten "dazwischen" (weiße Balken) zu modifizieren.

Rechenaufwand.

$$Z_{\text{Tri}} = \frac{4}{3}N^3 - \frac{7}{3}N^2 + \frac{7}{6}N + 1 \quad \text{Multiplikationen}$$
$$\frac{1}{2}(N-1)(N-2) \quad \text{Quadratwurzeln}$$

zum Vergleich:

$$Z_{\text{HESSENBERG}} \sim \frac{10}{3}N^3$$

Anmerkung. Es existiert auch ein "schneller" GIVENS-Algorithmus, der um einen Faktor zwei schneller als der hier beschriebene, aber auch wesentlich komplexer ist [4]

[4] z.B. Schwarz,"Numerische Mathematik", Kap. 6.3.3

4.12 Beispiel (Algorithmus). Erzeugung einer oberen HESSENBERG-Matrix unter Verwendung von GIVENS-Rotationen.

```
Algorithm for generating upper Hessenberg matrix with Givens-Rotations

original matrix A = a(i,j), i,j=1,n
delta = "machine accuracy"

do j=1,n-2                    !Alle Givens-Rotationen
  do i=j+2,n

    if (a(i,j) ne 0.) then    !nur falls nicht schon ``Null''

      if (abs(a(j+1,j)) < delta*abs(a(i,j))) then
        w=-a(i,j)             !Spezialfall 90 deg Drehung
        c=0.
        s=1.
      else
        w=sign(a(j+1,j)) * sqrt( a(j+1,j)^2 + a(i,j)^2 )  !Standard
        c=a(j+1,j)/w
        s=-a(i,j)/w
      endif

      a(j+1,j)=w              !Drehung für a_j+1,j
      a(i,j)=0.               !a_ij explizit eliminiert

      do k=j+1,n              !Zeilenoperationen, ab Spalte j+1
        h=c*a(j+1,k) - s*a(i,k)
        a(i,k)=s*a(j+1,k) + c*a(i,k)  !Zeile i
        a(j+1,k)=h                    !Zeile j+1
      enddo

      do k=1,n                !Spaltenoperationen, alle Zeilen
        h=c*a(k,j+1) - s*a(k,i)
        a(k,i)=s*a(k,j+1) + c*a(k,i)  !Spalte i
        a(k,j+1)=h                    !Spalte j+1
      enddo
    endif
  enddo
enddo
```

Der folgende `output` eines Programmes, das obigen Algorithmus verwendet, zeigt die Transformation einer *allgemeinen* und einer *symmetrischen* 6×6 Ausgangsmatrix. Man beachte, dass sich die symmetrische Tridiagonalmatrix "automatisch" ergibt, obwohl der "allgemeine" Algorithmus (Kap. 4.3.1) verwendet wurde.

```
  original matrix A (asymmetric)

  7.000000    3.000000    4.000000  -11.000000   -9.000000   -2.000000

 -6.000000    4.000000   -5.000000    7.000000    1.000000   12.000000

 -1.000000   -9.000000    2.000000    2.000000    9.000000    1.000000

 -8.000000    0.000000   -1.000000    5.000000    0.000000    8.000000

 -4.000000    3.000000   -5.000000    7.000000    2.000000   10.000000

  6.000000    1.000000    4.000000  -11.000000   -7.000000   -1.000000

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

  Givens transformed matrix A^h (upper hessenberg)

  7.000000   -7.276069   -5.812049    0.139701    9.015201   -7.936343

-12.369317    4.130719   18.968509   -1.207073  -10.683309    2.415951
```

```
  0.000000   -7.160342    2.447765   -0.565594    4.181396   -3.250955

  0.000000    0.000000   -8.598771    2.915100    3.416858    5.722969

  0.000000    0.000000    0.000000   -1.046436   -2.835101   10.979178

  0.000000    0.000000    0.000000    0.000000   -1.414293    5.341517

======================================================================

  original matrix B (symmetric)

  1.000000    5.000000    3.000000    8.000000    5.000000    1.000000

  5.000000    3.000000    4.000000    5.000000    6.000000   -2.000000

  3.000000    4.000000    5.000000    6.000000    7.000000    8.000000

  8.000000    5.000000    6.000000    7.000000   -1.000000    9.000000

  5.000000    6.000000    7.000000   -1.000000    9.000000   10.000000

  1.000000   -2.000000    8.000000    9.000000   10.000000   11.000000

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

  Givens transformed matrix B^h (upper hessenberg)

  1.000000   11.135529   -0.000000    0.000000    0.000000    0.000000

 11.135529   18.661290   13.778423    0.000000   -0.000000   -0.000000

  0.000000   13.778423    9.996039    9.538522    0.000000   -0.000000

  0.000000    0.000000    9.538522   -0.472736    2.810696    0.000000

  0.000000    0.000000    0.000000    2.810696   -1.423393   -2.574040

  0.000000    0.000000    0.000000    0.000000   -2.574040    8.238799
```

4.4 Eigenwerte und –vektoren von symmetrischen Tridiagonalmatrizen und HESSENBERG-Matrizen: Der QR-Algorithmus

4.4.1 QR-Zerlegung und Transformation

Anmerkung: Alle im weiteren vorgestellten Überlegungen lassen sich nicht nur für eine Rechtsdreiecksmatrix R, sondern in analoger Weise auch für eine Linksdreiecksmatrix L ("QL-Algorithmus") durchführen.

4.13 Satz. *Jede quadratische Matrix A lässt sich in das Produkt*

$$A = Q \cdot R \tag{4.47}$$

zerlegen, wobei Q eine orthogonale und R eine Rechtsdreiecksmatrix ist. Diese Faktorisierung nennt man QR-Zerlegung.

4.14 Beweis. *Der Beweis wird durch Konstruktion der Zerlegung geführt, indem wir spaltenweise alle subdiagonalen Elemente der Ausgangsmatrix A,*

$$a_{21}, a_{31}, \ldots, a_{N1}; \quad a_{32}, a_{42}, \ldots, a_{N2}; \quad \ldots; \quad a_{N,N-1}$$

über Linksmultiplikationen mit JACOBI*-Rotationsmatrizen $U(p,q)$ bzgl. der Indexpaare*

$$(1,2),(1,3),\ldots,(1,N); \quad (2,3),(2,4),\ldots,(2,N); \quad \ldots; \quad (N-1,N) \tag{4.48}$$

$$A^{(k)} = U_k^\intercal A^{(k-1)}, \qquad A^{(0)} = A, \qquad k = 1, \dots N^* = \frac{1}{2}(N-1)N \tag{4.49}$$

eliminieren. Man beachte, dass dieser Eliminationsprozess keine *Ähnlichkeitsabbildung, sondern "nur" eine (orthogonale) Transformation darstellt.*

Die entsprechenden Drehwinkel werden (wie immer) durch Nullsetzen von

$$a_{ij}^{(k)} = a_{jj}^{(k-1)} \cdot s + a_{ij}^{(k-1)} \cdot c := 0 \tag{4.50}$$

(mit $c^2 + s^2 = 1$) ermittelt. Falls zufälligerweise ein Element schon Null ist, $a_{ij}^{(k-1)} = 0$, dann wird $U_k = \mathbb{1}$ gesetzt. Für eine numerisch stabile Formulierung der Transformation (die analog zur Givenstransformation durchgeführt wird) verweisen wir auf die Literatur [5] *und auf die Beispielalgorithmen (4.12, 4.22)*

Analog zur Givens*-Rotation (Kap. 4.3) wird hier sukzessive eine Rechtsdreiecksmatrix aufgebaut, wobei schon erzeugte Nullen erhalten bleiben. Die* Unterschiede *zwischen* QR*-Zerlegung (links) und* Givens*-Rotation (rechts) werden im nächsten Diagramm für die Spalte "2" (bzgl. des zuerst zu eliminierenden Elementes) veranschaulicht.*

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots \\ 0 & a_{22} & a_{23} & \dots \\ 0 & \boxed{a_{32}} & a_{33} & \dots \\ 0 & a_{42} & a_{43} & \dots \\ 0 & \vdots & \vdots & \end{pmatrix} \begin{matrix} \\ \\ \leftarrow \\ \leftarrow \\ \\ \end{matrix} \qquad \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \dots \\ a_{21} & a_{22} & a_{23} & a_{24} & \dots \\ 0 & a_{32} & a_{33} & a_{34} & \dots \\ 0 & \boxed{a_{42}} & a_{43} & a_{44} & \dots \\ 0 & a_{52} & \vdots & \vdots & \\ & & \uparrow & \uparrow & \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \\ \leftarrow \\ \\ \end{matrix}$$

Für die QR*-Zerlegung werden nur Zeilenoperationen $A' = U^\intercal A$, für die* Givens*-Rotation jedoch Zeilen-* **und** *Spaltenoperationen $A'' = U^\intercal A U$ durchgeführt.*

Insgesamt hat man nach N^* Transformationen,

$$\underbrace{U_{N^*}^\intercal U_{N^*-1}^\intercal \dots U_2^\intercal U_1^\intercal A^{(0)}}_{Q^\intercal} = R \tag{4.51}$$

sowohl die Rechtsdreiecksmatrix R als auch die orthogonale Matrix Q (als Produkt der elementaren orthogonalen Transformationen U_k) aufgebaut und damit die Ausgangsmatrix wie gewünscht faktorisiert. $\square$.

4.15 Satz. *Tridiagonalmatrizen oder* Hessenberg*-Matrizen der Ordnung N lassen sich in $(N-1)$ Drehungen* QR*-zerlegen, wobei*

$$A = QR \quad mit \quad Q = U_1 U_2 \dots U_{N-1} \tag{4.52}$$

und $U_i = U(i, i+1)$ ist, d.h. man die Transformation bzgl. der Indexpaare $(1,2), (2,3), \dots (N-1, N)$ durchführt.

Dieser Satz (im Zusammenhang mit dem nächsten) impliziert eine enorme Zeitersparnis ($N-1$ gegenüber $\frac{1}{2}N(N-1)$ Drehungen) und ist der Grund, warum man *allgemeine* Matrizen vor Durchführung des QR-Algorithmus (der iterativer Natur ist und eine oftmalige QR-Zerlegung erfordert) auf Hessenberg-Form bzw. *symmetrische* Matrizen auf Tridiagonalgestalt transformiert! Der Beweis wird analog zum Beweis des allgemeinen Satzes durchgeführt (Beweis 4.14) und beruht darauf, dass nur die Elimination *eines* von Null verschiedenen Elementes auf der ersten Subdiagonalen notwendig ist.

[5] z.B. Schwarz, "Numerische Mathematik", Kap. 6.3.1

4.16 Definition. *Unter* QR*-Transformationen versteht man folgende Transformation*

$$A' = RQ = Q^\intercal AQ \tag{4.53}$$

wenn $A = QR$ gilt. Die QR*-Transformierte A' ist per Definition orthogonal-ähnlich zu A.*

4.17 Satz. *Die* QR*-Transformierte einer* HESSENBERG*-Matrix ist wiederum eine* HESSENBERG*-Matrix.*

Dies läßt sich wiederum durch Konstruktion zeigen; die Anzahl der wesentlichen Operationen für Zerlegung *und* Transformation skaliert hierbei mit $\approx 4N^2$ für $N \gg 1$. Damit gilt auch der folgende

4.18 Satz. *Die* QR*-Transformierte einer symmetrischen Trigiagonalmatrix ist wiederum symmetrisch und tridiagonal!*

4.19 Beweis.

(i) Sei A symmetrisch und tridiagonal, dann folgt

$$A' = Q^\intercal AQ \qquad \Rightarrow \qquad (A')^\intercal = Q^\intercal A^\intercal Q = A',$$

d.h. A' ist symmetrisch

(ii) eine tridiagonale Matrix ist ein Spezialfall einer HESSENBERG*-Matrix, d.h. nach obigem Satz (4.17) hat A'* HESSENBERG*-Form.*

(iii) eine symmetrische HESSENBERG*-Matrix ist aber eine symmetrische Tridiagonalmatrix,* □.

MERKE: Hat man eine allgemeine Matrix über GIVENS-Rotationen auf HESSENBERG-Form orthogonal-ähnlich transformiert, lassen sich alle weiteren orthogonal-ähnlichen QR-Transformationen in ca. $4N^2$ Operationen durchführen, und die HESSENBERG-Form bleibt dabei erhalten.

Den eigentlichen Zusammenhang zwischen QR-Transformation und Eigenwertproblem zeigt nun das nächste Kapitel.

4.4.2 Der QR-Algorithmus

Basis für diesen Zusammenhang, der letztendlich im QR-Algorithmus mündet, ist der folgende Satz von SCHUR (1908), der hier in der Formulierung für reelle Matrizen wiedergegeben wird und schon in der Einführung zur Gesamtproblematik zitiert wurde.

4.20 Satz (Satz von SCHUR). *Zu jeder reellen Matrix A der Ordnung N existiert eine orthogonale Matrix U der Ordnung N, so dass die zu A ähnliche Matrix $\tilde{R} = U^\intercal AU$ die Quasidreiecksgestalt*

$$\tilde{R} = U^\intercal AU = \begin{pmatrix} R_{11} & \dots & & \dots & R_{1m} \\ 0 & R_{22} & & & R_{2m} \\ \vdots & & R_{33} & & \vdots \\ & & & \ddots & \\ 0 & \dots & & 0 & R_{mm} \end{pmatrix} \tag{4.54}$$

besitzt. Die Matrizen $R_{ii} (i = 1, \dots, m)$ besitzen entweder die Ordnung eins oder die Ordnung zwei und haben im letzteren Fall ein Paar von konjugiert komplexen Eigenwerten (ohne Beweis).

Auf diesem Satz beruht nun die

Idee des QR-Algorithmus: Man konstruiere obige Quasidreiecksmatrix $\tilde{R}$ (deren Existenz aufgrund des Satzes gesichert ist) auf möglichst effektive Weise. Die Eigenwerte sind dann entweder direkt ablesbar oder aus den 2×2 Untermatrizen einfach zu berechnen. Diese Konstruktion versucht man, über eine Folge von QR-Transformationen (bezüglich HESSENBERG-Matrizen) H,

$$H_k = Q_k R_k, \qquad H_{k+1} = R_k Q_k, \qquad k = 1, 2, \ldots \tag{4.55}$$

zu erreichen. Dann gilt folgende Konvergenzaussage

4.21 Satz (Francis, 1961).

(i) Die Matrix H habe Eigenwerte λ_i mit der Eigenschaft, dass

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_N|,$$

d.h. die Eigenwerte sind reell und nicht entartet. Die dazugehörigen Eigenvektoren $\boldsymbol{x}_i$ seien als Spalten in der regulären Matrix $X \in R^{N \times N}$ (der Reihe nach) zusammengefasst.
Falls nun für X^{-1} die LU-Zerlegung existiert [6], dann konvergiert die Folge der QR*-Transformationen H_k für $k \to \infty$ gegen eine Rechtsdreicksmatrix und es gilt*

$$\lim_{k \to \infty} h_{ii}^{(k)} = \lambda_i \quad , \quad i = 1, \ldots N, \tag{4.56}$$

d.h. in der erzeugten Matrix sind die Eigenwerte betragsmäßig geordnet*, mit dem absolut größten Eigenwert "links oben".*

(ii) Hat die Matrix H Paare von konjugiert komplexen Eigenwerten derart, dass ihre Beträge und die Beträge der reellen Eigenwerte paarweise verschieden sind, und existiert die komplexe LU-Zerlegung von X^{-1}, dann konvergiert die Folge H_k, $k \to \infty$ gegen die Quasidreiecksmatrix $\tilde{R}$ (4.54)

(...länglicher Beweis...)

Problem. Obwohl aufgrund dieses Satzes die Konvergenz (und damit die Lösung des Eigenwertproblemes) gesichert ist (zumindest unter der gegebenen Einschränkung, die allerdings fast immer erfüllt ist), kann diese Konvergenz sehr langsam sein. Es lässt sich nämlich zeigen, dass für große k die Subdiagonalelemente wie

$$|h_{i+1,i}^{(k)}| \approx |\frac{\lambda_{i+1}}{\lambda_i}|^k, \qquad i = 1, \ldots N-1 \tag{4.57}$$

konvergieren. Damit gilt, dass

(a) im *reellen* Fall, $|\lambda_{i+1}/\lambda_i| < 1$ die Konvergenz nur *linear* ist und vom Verhältnis $(\lambda_{i+1}/\lambda_i)$ abhängt, wobei dieses Verhältnis nahe bei "1" liegen kann und somit die Konvergenz "schleichend" ist.

(b) Für *konjugiert komplexe* Paare trit keine Konvergenz ein, da $|\lambda_{i+1}/\lambda_i| = 1$ (wie es auch der Satz behauptet; man beachte das Auftreten von entsprechenden 2×2 Untermatrizen).

Aufgrund dieser Problematik ist der "reine" QR-Algorithmus nur schlecht geeignet, das Eigenwertproblem zu lösen, da die erforderliche Rechenzeit teilweise unannehmbar wäre. Es gibt allerdings eine Möglichkeit, die Konvergenz drastisch zu beschleunigen, die auf der sog. "Spektralverschiebung" beruht und auf zwei Arten in den Algorithmus "eingebaut" werden kann. Im weiteren werden wir uns auf die sog. "explizite" Spektralverschiebung konzentrieren und die entsprechenden Algorithmen (sowohl für rein reelle als auch für reelle und komplexe Eigenwerte) vorstellen. Bezüglich der zweiten Methode, der sog. "impliziten" Spektralverschiebung, sei auf die Literatur verwiesen [7].

[6] d.h. alle Pivotelemente von X^{-1} schon *ohne* Permutation $\neq 0$ sind!, vgl. Kap. 3.5.2

[7] z.B. Schwarz, "Numerische Mathematik, Kap. 6.4.3 und "Numerical Recipes", Kap. 11.3/4

4.4.3 Der QR-Algorithmus mit expliziter Spektralverschiebung für reelle Eigenwerte

Die oben erwähnte *Spektralverschiebung* beruht darauf, die Eigenwerte einer Matrix durch Subtraktion einer Diagonalmatrix mit identischen Elementen $\sigma \in \mathbb{R}$ zu modifizieren. Die modifizierte Matrix

$$\tilde{H} = H - \sigma \mathbb{1}$$

hat dann die Eigenwerte $\lambda_i - \sigma$, $i = 1, \dots N$. Diese seien jetzt so indiziert, daß

$$|\lambda_1 - \sigma| > |\lambda_2 - \sigma| > \cdots > |\lambda_N - \sigma|.$$

Damit konvergieren die Subdiagonalelemente der Folge

$$\tilde{H}_k \quad \text{mit} \quad \tilde{H}_1 = H - \sigma \mathbb{1}$$

mit (vgl. 4.57)

$$|\tilde{h}^{(k)}_{i+1,i}| \approx |\frac{\lambda_{i+1} - \sigma}{\lambda_i - \sigma}|^k < 1, \qquad i = 1, \dots N-1. \tag{4.58}$$

Sei nun σ eine gute Näherung für λ_N, so dass

$$|\lambda_N - \sigma| \ll |\lambda_i - \sigma|, \qquad i = 1, \dots N-1$$

gilt. Dann konvergiert das "letzte" Subdiagonalelement $\tilde{h}^{(k)}_{N,N-1}$ sehr schnell gegen Null (man beachte, dass der Quotient in Gl. (4.58) mit dieser Wahl sehr klein ist), und die Matrix zerfällt nach wenigen Iterationsschritten in die folgende Struktur (nachdem wir die Spektralverschiebung rückgängig gemacht haben, s.u.)

$$\begin{pmatrix} \times & \times & \times & \times & | & \times \\ \times & \times & \times & \times & | & \times \\ 0 & \times & \times & \times & | & \times \\ 0 & 0 & \times & \times & | & \times \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \underbrace{\tilde{h}_{N,N-1}}_{\to 0} & | & \to \lambda_N \end{pmatrix} \quad \Rightarrow \begin{pmatrix} \hat{H} & | & \hat{\boldsymbol{h}} \\ \cdots & \cdots & \cdots \\ \mathbf{0}^\intercal & | & \lambda_N \end{pmatrix}$$

Nachdem die Matrix zerfallen ist und wir somit den ersten Eigenwert λ_N bestimmt haben, sind die restlichen Eigenwerte diejenigen der Matrix $\hat{H} \in \mathbb{R}^{(N-1)\times(N-1)}$, die wir genauso wie eben beschrieben behandeln.

Die Spektralverschiebung wird nicht nur einmal, sondern vor jedem QR- Schritt angewandt, wobei σ den Veränderungen angepasst wird. Am Ende des Schrittes wird die jeweilige Verschiebung wieder rückgängig gemacht. Damit lautet der QR-Algorithmus mit expliziter Spektralverschiebung

$$\begin{aligned} H_k - \sigma_k \mathbb{1} &= Q'_k R'_k && \textit{(Zerlegung)} \\ \Rightarrow R'_k &= Q'^{\intercal}_k \left(H_k - \sigma_k \mathbb{1}\right) \\ &\Rightarrow \\ H_{k+1} &= R'_k Q'_k + \sigma_k \mathbb{1} && \textit{(Transformation, Spektralversch. rückgängig gemacht)} \\ &= Q'^{\intercal}_k \left(H_k - \sigma_k \mathbb{1}\right) Q'_k + \sigma_k \mathbb{1} \\ &= Q'^{\intercal}_k H_k Q'_k \end{aligned} \tag{4.59}$$

Unter Verwendung der Spektralverschiebung haben wir also wiederum ein Matrix H_{k+1} erzeugt, die orthogonal ähnlich zu H_k ist (wie im originalen Algorithmus, allerdings jetzt mit einer veränderten Transformationsmatrix Q'), aber die Konvergenz beschleunigt. Man beachte allerdings, dass die Eigenwerte jetzt nicht mehr bezüglich ihrer (absoluten) Größe $|\lambda_i|$ angeordnet sind.

Nachzutragen bleibt nun noch, welchen Wert man für das jeweilige σ_k annimmt. Optimal wäre natürlich $\sigma = \lambda_N$, aber λ_N wird ja gerade gesucht (da beisst sich die Katze in den Schwanz...)! In Praxis werden

Zwei Möglichkeiten zur Wahl von σ betrachtet.

(a) Man verwendet einfach das aktuelle "letzte" Diagonalelement

$$\sigma_k = h_{NN}^{(k)}, \qquad k = 1, 2, \ldots \tag{4.60}$$

(b) Eine bessere Konvergenz (und eine Verallgemeinerung auf den komplexen Fall, siehe nächstes Kapitel) ergibt sich, wenn man σ über die Eigenwerte der aktuellen, letzten 2×2-Untermatrix nähert. Diese berechnen sich über

$$C_k = \begin{pmatrix} h_{N-1,N-1} & h_{N-1,N} \\ h_{N,N-1} & h_{NN} \end{pmatrix} \hat{=} \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$(a-\lambda)(d-\lambda) - bc = 0$$

$$ad - \lambda(a+d) + \lambda^2 - bc = 0$$

$$\Rightarrow \quad \lambda^2 - \lambda(a+d) + (ad - bc) = 0$$

$$\lambda_{1,2} = \frac{a+d}{2} \pm \sqrt{\left(\frac{a+d}{2}\right)^2 - (ad-bc)} \tag{4.61}$$

Falls die Diskriminante < 0 ist, liegen (höchstwahrscheinlich) konjugiert komplexe Lösungen vor, und man wendet den im nächsten Abschnitt diskutierten QR-Doppelschritt (Kap. 4.4.4) an. Ansonsten wählt man für σ_k denjenigen Eigenwert, der näher an h_{NN} liegt!

$$\sigma_k = \lambda_i, \quad \text{so dass} \quad |\lambda_i - h_{NN}| = \text{Min!} \tag{4.62}$$

Mit beiden Vorgehensweisen erzielt man letztendlich eine *quadratische* Konvergenz (gegenüber der linearen ohne Spektralverschiebung) für das Verschwinden des (jeweils letzten) Subdiagonalelementes, d.h. $h_{N,N-1} \to 0$ nach nur (sehr) wenigen Iterationen. Als vernünftiges Konvergenzkriterium fordert man, dass

$$|h_{i+1,i}^{(k)}| < \delta \cdot \max\left(|h_{ii}^{(k)}|, |h_{i+1,i+1}^{(k)}|\right) \tag{4.63}$$

wenn δ die Maschinengenauigkeit ist. Dieses Kriterium erweist sich auch für betragsmäßig kleine Eigenwerte als geeignet.

Nach Konvergenz, d.h. wenn $h_{N,N-1} \to 0$, gilt

$$\Rightarrow \quad \sigma^{(k)} = \lambda_N = h_{NN}^{(k)} \qquad (\text{da "c"} = 0),$$

und die Matrix entkoppelt. Die restlichen $(N-1)$ Eigenwerte der Untermatrix $\hat{H} \in \mathbb{R}^{(N-1)\times(N-1)}$ werden, wie schon oben erwähnt, analog ermittelt, wobei sich σ_k wiederum aus den Eigenwerten der letzten 2×2 Untermatrix (jetzt bzgl. der Indizes $(N-2, N-1)$) ergibt.

Aus dem bisher Ausgeführten ist der wesentlicher Vorteil des QR-Algorithmus sofort offensichtlich. *Mit jedem berechneten Eigenwert reduziert sich die Ordnung der noch zu bearbeitenden Matrix*, und man gewinnt einen zusätzlichen Zeitvorteil. (Man erinnere sich, dass beim JACOBI-Algorithmus jede Iteration (=Rotation) die gleiche Anzahl an Operationen erforderte.)

Nachdem wir nun σ_k spezifiziert haben, lautet der QR-Algorithmus mit expliziter Spektralverschiebung pro Schritt *im Prinzip* folgendermaßen

$$H_{k+1} = \underbrace{U_{N-1}^\intercal \ldots U_2^\intercal U_1^\intercal}_{Q_k'^T} (H_k - \sigma_k \mathbb{1}) \underbrace{U_1 U_2 \ldots U_{N-1}}_{Q_k'} + \sigma_k \mathbb{1} \tag{4.64}$$

wobei U_i die entsprechenden Rotationsmatrizen $U_i = U(i, i+1)$ sind (vergleiche 4.48).

Ein dieser Gleichung immanentes Problem *scheint* es nun zu sein, dass pro Schritt die Drehwinkel aller Einzelrotationen U_i abgespeichert werden müssen, was einen zusätzlichen Zeit- und Verwaltungsaufwand erfordern würde. Auf Grund der Assoziativität von Matrixmultiplikationen lässt sich die Reihenfolge der Multiplikationen allerdings so wählen, dass dieses Problem erst gar nicht auftritt.

Wir nehmen dazu an, dass für $A \in \mathbb{R}^{6\times 6}$ die Operationen

$$U_2^\intercal U_1^\intercal \, (H_k - \sigma_k \mathbb{1})$$

schon durchgeführt wurden. Daraus resultiert folgende Matrixstruktur

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{pmatrix}$$

Die ersten beiden Zeilen wurden dabei durch die Transformationen $U_1^\intercal = U^\intercal(1,2)$ und $U_2^\intercal = U^\intercal(2,3)$ modifiziert. Die folgenden Rotationen $U_3^\intercal, U_4^\intercal, \ldots$ betreffen nur noch die dritte und folgende Zeilen, d.h. Zeile 1 und 2 *und insbesondere Spalte 1 und 2 dieser Zeilen* werden durch die weitergehende Zerlegung nicht mehr verändert. Deshalb kann nun die Operation $\cdot U_1$ (Rechtsmultiplikation) durchgeführt werden, da sie "nur" Linearkombinationen der Spalten 1 und 2 bildet. Den nächsten Schritt bildet dann die Linksmultiplikation von $U_3^\intercal$, danach die Rechtmultiplikation $\cdot U_2$ usw. Mit dieser Vorgehensweise müssen die jeweiligen Drehwinkel *nicht* abgespeichert werden, und das gesamte *gestaffelte* Verfahren lässt sich wie folgt beschreiben.

- *Schritt 1*: $U_1^\intercal$ von links, merke Drehwinkel (alt)
- *Schritt i*: $i \leq 2 \leq N-1$
  a) $U_i^\intercal$ von links, neuer Drehwinkel: QR-Zerlegung in i, (i+1) ter Zeile
  b) U_{i-1} von rechts mit altem Drehwinkel: Aufbau der H-Matrix in i, (i-1) ter Spalte
  c) alter Drehwinkel = neuer Drehwinkel
  d) fortlaufend Subtraktion von σ_k und finale Addition für Diagonale
- *Schritt N*: U_{N-1} von rechts, Addition von σ_k zu Diagonalelement $(N-1, N-1)$ und (N, N)

Der QR-Algorithmus für Tridiagonalmatrizen (aus symmetrischen Originalmatrizen) konvergiert sogar *kubisch*!

4.22 Beispiel (Algorithmus). QR-Algorithmus mit expliziter Spektralverschiebung für allgemeine reelle Matrizen; nur reelle Eigenwerte

```
QR algorithm with explicit spectral shift, step n
Hessenberg matrix h(n,n)
machine precision delta
                      !Zuerst wird sigma_k ermittelt

a=h(n-1,n-1)          !Matrix C_k
b=h(n-1,n)
c=h(n,n-1)
d=h(n,n)

aux=.5*(a+d)
```

```
disc=aux^2-(a*d-b*c)  !Diskriminante

if(disc < 0.) then    !konj. komplexe Eigenwerte
  print*,' imaginary eigenvalues found, matrix not transformed'
  print*
  imag=.true.
  return
endif

disc=sqrt(disc)
sig1=aux+disc         !Eigenwerte von C_k
sig2=aux-disc

if( abs(sig1-d) <= abs(sig2-d) ) then  !Wähle sigma_k
  sigma=sig1
else
  sigma=sig2
endif

                      !QR Transformation, Schritt k, gestaffelte Ausführung
h(1,1)=h(1,1)-sigma   !explizite Spektralverschiebung

do i=1,n              !für alle Spalten

  if(i < n) then      !bis auf letzten Schritt

    if( abs(h(i,i)) < delta * abs(h(i+1,i)) ) then !Spezialfall, 90 deg Drehung
      w=abs(h(i+1,i))
      c=0.
      s=sign(1,h(i+1,i))
    else              !Standard
      w=sqrt(h(i,i)^2 + h(i+1,i)^2)
      c=h(i,i)/w      !neuer Drehwinkel
      s=-h(i+1,i)/w
    endif

    h(i,i)=w          !h_ii geändert
    h(i+1,i)=0.       !Subdiagonalelement eliminiert
    h(i+1,i+1)=h(i+1,i+1)-sigma  !Spektralverschiebung

    do j=i+1,n        !Zeilenoperationen mit neuem Drehwinkel
      g=c*h(i,j) - s*h(i+1,j)
      h(i+1,j)=s*h(i,j) + c*h(i+1,j)
      h(i,j)=g
    enddo
  endif

  if(i > 1) then      !ab Schritt 2

    do j=1,i          !Spaltenoperatione mit altem Drehwinkel
      g=ct*h(j,i-1) - st*h(j,i)
      h(j,i)=st*h(j,i-1) + ct*h(j,i)
      h(j,i-1)=g
    enddo
    h(i-1,i-1)=h(i-1,i-1)+sigma  !finale Korrektur
  endif

  ct=c                !alter Drehwinkel = neuer Drehwinkel
  st=s

enddo
h(n,n)=h(n,n)+sigma   !letzte Korrektur
```

Der folgende `output` eines Programmes, das obigen Algorithmus verwendet, zeigt eine Sequenz von QR-Transformationen einer *allgemeinen* 6×6 Ausgangsmatrix. Zunächst wird über GIVENS eine obere HESSENBERG-Form gebildet und diese dann mittels QR mit expliziter Spektralverschiebung weiterverarbeitet. Zur Bestimmung des ersten Eigenwertes sind dabei 7 Schritte notwendig. Der zweite Eigenwert bestimmt sich (nun aus der verbleibenden 5×5 Matrix) in 3 Iterationenen. Für die verbleibende 4×4 Matrix stoppt der Algorithmus, weil eine negative Diskriminante für C_k ermittelt wird.
Das jeweilige Konvergenzverhalten für das letzte Subdiagonalelement wird ebenfalls protokolliert, man findet größtenteils eine *quadratische* Konvergenz

```
  original matrix a (asymmetric)

  7.000000    3.000000    4.000000  -11.000000   -9.000000   -2.000000

 -6.000000    4.000000   -5.000000    7.000000    1.000000   12.000000

 -1.000000   -9.000000    2.000000    2.000000    9.000000    1.000000

 -8.000000    0.000000   -1.000000    5.000000    0.000000    8.000000

 -4.000000    3.000000   -5.000000    7.000000    2.000000   10.000000

  6.000000    1.000000    4.000000  -11.000000   -7.000000   -1.000000

------------------------------------------------------------------------
  Givens transformed matrix a^h (upper hessenberg)

  7.000000   -7.276069   -5.812049    0.139701    9.015201   -7.936343

-12.369317    4.130719   18.968509   -1.207073  -10.683309    2.415951

  0.000000   -7.160342    2.447765   -0.565594    4.181396   -3.250955

  0.000000    0.000000   -8.598771    2.915100    3.416858    5.722969
                                                 ----------------------
  0.000000    0.000000    0.000000   -1.046436  | -2.835101   10.979178 |

  0.000000    0.000000    0.000000    0.000000  | -1.414293    5.341517 |
                                                 ----------------------
                                                   1. Untermatrix C_1

------------------------------------------------------------------------
  qr transformation, dim =  6  it =  1             1. Transformation

  sigma =    2.3424691828527697

 10.965709    7.884201    0.967311   -8.886877    7.276220   22.712401

 -8.851271    0.841162   -0.303428    6.706005   -6.542602    2.549021

  0.000000   -6.543402    2.323660    2.173139    5.357753   -6.749913

  0.000000    0.000000   -1.060600   -2.840623   11.264555    0.034875

  0.000000    0.000000    0.000000   -1.394725    5.351617   -0.018167

  0.000000    0.000000    0.000000    0.000000   -0.165619    2.358475
                                                       ^
                                                  h(n,n-1) wird kleiner
------------------------------------------------------------------------
  qr transformation, dim =  6  it =  2

  sigma =    2.3574703994163020

  6.245879    6.994592   14.211420  -11.604268   -4.715402   13.648974
```

```
 -5.731853    5.311709    4.174269   -0.438905    3.930892   16.306351

  0.000000   -0.920617   -2.004107    3.858957    8.525548    4.359026

  0.000000    0.000000   -2.194370    6.247790    7.183280   -7.629419

  0.000000    0.000000    0.000000   -1.140015    0.304290    6.000472

  0.000000    0.000000    0.000000    0.000000   -0.046925    2.894439
                                                       ^
                                                  und kleiner
-------------------------------------------------------------------------
  qr transformation, dim =  6  it =  3

  sigma =    2.7807384120871612

  5.002682    5.959130   -0.306328   -1.468529    9.412667   -6.544733

 -6.292543    4.700951   -5.878361  -10.936873   13.288174   19.925014

  0.000000   -0.460740    2.306565    3.234193   11.767113   -0.503055

  0.000000    0.000000   -1.012597    2.241030   -2.189677  -11.682394

  0.000000    0.000000    0.000000   -0.839729    1.764372    2.361489

  0.000000    0.000000    0.000000    0.000000   -0.007646    2.984399
                                                       ^
                                                  und kleiner
-------------------------------------------------------------------------
  qr transformation, dim =  6  it =  4

  sigma =    2.9694159749196212

  4.827024    5.924491  -11.487652    1.361448   -9.715866  -21.005982

 -5.918631    5.028020    5.072473   -2.033129   12.077297   -0.021853

  0.000000   -0.096355    3.117470    3.570375   -5.484933    9.394808

  0.000000    0.000000   -2.196275   -2.027953   10.058579    5.954444

  0.000000    0.000000    0.000000   -0.655216    5.056039    4.326270

  0.000000    0.000000    0.000000    0.000000   -0.000104    2.999400
                                                       ^
                                                  und kleiner
-------------------------------------------------------------------------
  qr transformation, dim =  6  it =  5

  sigma =    2.9996187080813965

  5.008875    5.988555   -2.239970   -5.495918  -15.624389   -6.177763

 -5.981347    4.992098   -0.525135   -8.312230   -7.217940  -20.220489

  0.000000   -0.033808   -2.009435    3.865648   -9.451051   -6.071070

  0.000000    0.000000   -3.702849    4.010098   -5.889334    8.099599

  0.000000    0.000000    0.000000   -0.179653    3.998364    5.851904
```

```
  0.000000    0.000000    0.000000    0.000000   -0.000000    3.000000

                                                  und kleiner
------------------------------------------------------------------------

  qr transformation, dim =  6  it =  6

  sigma =    3.0000000258923145

  4.991623    5.988209   -3.490458   -5.233181    1.236387   17.201407

 -5.982847    5.021476    6.488094    7.053759  -16.317615  -12.261388

  0.000000   -0.033423    0.160555    5.243584   11.907163    0.132287

  0.000000    0.000000   -0.870774    1.955150   -1.175364  -10.801113

  0.000000    0.000000    0.000000   -0.107481    3.871195    4.563720

  0.000000    0.000000    0.000000    0.000000    0.000000    3.000000
                                                       ^
                                                  und kleiner
------------------------------------------------------------------------
  qr transformation, dim =  6  it =  7

  sigma =    3.0000000000000155

  5.016891    6.012736    9.352165    5.186822   16.107590   17.066758

 -5.996704    5.002805    1.991770    2.451839   -3.943152   12.447252

  0.000000   -0.015739    1.527433    5.823072  -10.784469    2.970617
                                     ----------------------
  0.000000    0.000000   -0.740567  | 0.388041    4.453352 | 10.183320

  0.000000    0.000000    0.000000  | 0.045255    4.064829 |  5.000627
                                    ----------------------
  0.000000    0.000000    0.000000    0.000000    0.000000    3.000000  <---
                                dies ist die nächste Untermatrix

                                                 und die Matrix zerfält
------------------------------------------------------------------------

  real eigenvalue found    3.0000000000000164    Dies ist der erste Eigenwert

  Iteration history for h^(k+1), n =  6
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1    2.342469    2.358475  -.165619E+00    5.351617
  2    2.357470    2.894439  -.469253E-01    0.304290
  3    2.780738    2.984399  -.764577E-02    1.764372
  4    2.969416    2.999400  -.104138E-03    5.056039
  5    2.999619    3.000000  -.390852E-07    3.998364     ab hier quadr.
  6    3.000000    3.000000  0.115289E-14    3.871195     Konvergenz
  7    3.000000    3.000000  0.873993E-30    4.064829

------------------------------------------------------------------------
  qr transformation, dim =  5  it =  1            Weiter mit der reduzierten
                                                  5 x 5 Matrix
  sigma =    4.0091738937123900

  5.000572    6.000497    0.831623    1.319300    6.569846

 -6.010013    4.994436  -10.744109   -2.646671   15.236538
```

```
  0.000000   -0.006628    2.863426    5.011943    9.047176

  0.000000    0.000000   -1.486712   -0.858568   -7.407964

  0.000000    0.000000    0.000000    0.000079    4.000134
                                          ^
                                      Die Subdiagonale ist kleiner geworden
-------------------------------------------------------------------------
  qr transformation, dim =  5  it =  2

  sigma =    4.0000128592600745

  4.996143    5.999050   -8.772877    6.812145  -13.950845

 -5.999763    4.999892   -0.111712   -1.271211    8.972972

  0.000000   -0.002040    2.217321    0.993110    0.379351

  0.000000    0.000000   -5.506490   -0.213357   11.694599

  0.000000    0.000000    0.000000   -0.000000    4.000000
                                          ^
                                         und wird noch kleiner
-------------------------------------------------------------------------

  qr transformation, dim =  5  it =  3

  sigma =    3.9999999998042930

  4.999907    5.999583   -1.845630   -1.991066  -11.136782

 -5.999675    4.999037   -3.519216  -10.257471  -12.292905

  0.000000   -0.001942   -1.306024    4.364919  -11.240090

  0.000000    0.000000   -2.136865    3.307080   -3.250358

  0.000000    0.000000    0.000000   -0.000000    4.000000  <---
                                          ^
                                         und schon zerfällt die Matrix
-------------------------------------------------------------------------

  real eigenvalue found    3.9999999999999787  Dies ist der 2. Eigenwert

  Iteration history for h^(k+1), n =  5
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1    4.009174    4.000134  0.794124E-04   -0.858568    quadratische
  2    4.000013    4.000000  -.147278E-09   -0.213357    Konvergenz von
  3    4.000000    4.000000  -.128295E-19    3.307080    Anfang an

-------------------------------------------------------------------------
  qr transformation, dim =  4  it =  1

  imaginary eigenvalues found, matrix not transformed

  4.999907    5.999583   -1.845630   -1.991066

 -5.999675    4.999037   -3.519216  -10.257471
                         ----------------------
  0.000000   -0.001942  |-1.306024    4.364919 |  Diskriminante negativ,

  0.000000    0.000000  |-2.136865    3.307080 |  Stop!
```

Das folgende Beispiel zeigt die QR-Zerlegung einer *symmetrischen* 6×6 Ausgangsmatrix, die mit dem "allgemeinen" Algorithmus behandelt wird. Hier werden (wie es sein muss) alle Eigenwerte (reell!) gefunden. Man beachte die kubische Konvergenz des Verfahrens in diesem Fall!

```
  original matrix a (symmetric)

  1.000000    5.000000    3.000000    8.000000    5.000000    1.000000

  5.000000    3.000000    4.000000    5.000000    6.000000   -2.000000

  3.000000    4.000000    5.000000    6.000000    7.000000    8.000000

  8.000000    5.000000    6.000000    7.000000   -1.000000    9.000000

  5.000000    6.000000    7.000000   -1.000000    9.000000   10.000000

  1.000000   -2.000000    8.000000    9.000000   10.000000   11.000000

------------------------------------------------------------------------
  Givens transformed matrix a^h (upper hessenberg) symmetrisch, Tridiag!

  1.000000   11.135529   -0.000000    0.000000    0.000000    0.000000

 11.135529   18.661290   13.778423    0.000000   -0.000000   -0.000000

  0.000000   13.778423    9.996039    9.538522    0.000000   -0.000000

  0.000000    0.000000    9.538522   -0.472736    2.810696    0.000000

  0.000000    0.000000    0.000000    2.810696   -1.423393   -2.574040

  0.000000    0.000000    0.000000    0.000000   -2.574040    8.238799

------------------------------------------------------------------------
  qr transformation, dim =  6  it =  1

  sigma =    8.8817484447060178

  2.264367   16.468518    0.000000   -0.000000   -0.000000    0.000000

 16.468518   21.887647    7.238300   -0.000000   -0.000000    0.000000

  0.000000    7.238300   -4.804783    3.207344    0.000000   -0.000000

  0.000000    0.000000    3.207344    4.022637    6.106560    0.000000
                                                 ----------------------
  0.000000    0.000000    0.000000    6.106560  | 3.824078   -0.026679 |

  0.000000    0.000000    0.000000    0.000000  |-0.026679    8.806055 |
                                                 ----------------------
------------------------------------------------------------------------
  qr transformation, dim =  6  it =  2

  sigma =    8.8061981307427359

  7.912711   19.884662    0.000000   -0.000000   -0.000000    0.000000

 19.884662   15.532262    4.764137    0.000000   -0.000000   -0.000000

  0.000000    4.764137   -5.015218    1.658918   -0.000000   -0.000000

  0.000000    0.000000    1.658918   -1.278980    1.902604    0.000000

  0.000000    0.000000    0.000000    1.902604   10.043341   -0.000004
```

```
  0.000000    0.000000    0.000000    0.000000   -0.000004    8.805884

------------------------------------------------------------------------
  qr transformation, dim =  6  it =  3

  sigma =    8.8058843287474318

 13.734169   20.700691   -0.000000   -0.000000   -0.000000    0.000000

 20.700691    9.028060    3.127160    0.000000   -0.000000    0.000000

  0.000000    3.127160   -4.677932    1.219094    0.000000   -0.000000
                                     ----------------------
  0.000000    0.000000    1.219094  |-1.241566    0.299661 |  0.000000

  0.000000    0.000000    0.000000  |  0.299661   10.351385|  0.000000
                                      ----------------------
  0.000000    0.000000    0.000000    0.000000    0.000000    8.805884 <---

  eigenvalue found    8.8058843287499435

  Iteration history for h^(k+1), n =  6
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1    8.881748    8.806055  -.266792E-01    3.824078   Kubische
  2    8.806198    8.805884  -.368932E-05   10.043341   Konvergenz!
  3    8.805884    8.805884  0.588542E-17   10.351385

------------------------------------------------------------------------
  qr transformation, dim =  5  it =  1

  sigma =   10.3591260517710104

 15.725219   20.608496    0.000000    0.000000    0.000000

 20.608496    6.625228    2.222833    0.000000    0.000000

  0.000000    2.222833   -4.443082    0.941813    0.000000

  0.000000    0.000000    0.941813   -1.072442   -0.000002

  0.000000    0.000000    0.000000   -0.000002   10.359193

------------------------------------------------------------------------
  qr transformation, dim =  5  it =  2

  sigma =   10.3591928631745098

 17.253656   20.324677   -0.000000   -0.000000   -0.000000

 20.324677    4.872035    1.554758   -0.000000   -0.000000
                         ----------------------
  0.000000    1.554758  |-4.325234    0.727550 |  0.000000

  0.000000    0.000000  | 0.727550   -0.965535 |  0.000000
                         ----------------------
  0.000000    0.000000    0.000000    0.000000   10.359193  <---

  eigenvalue found   10.3591928631745116

  Iteration history for h^(k+1), n =  5
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1   10.359126   10.359193  -.174664E-05   -1.072442
```

```
  2   10.359193   10.359193  0.281120E-21   -0.965535

------------------------------------------------------------------------
  qr transformation, dim =  4  it =  1

  sigma =   -0.8147495813065653

 30.522484    8.607633    0.000000   -0.000000

  8.607633   -8.598937    0.461310   -0.000000

  0.000000    0.461310   -4.279886    0.001310

  0.000000    0.000000    0.001310   -0.808738

------------------------------------------------------------------------
  qr transformation, dim =  4  it =  2

  sigma =   -0.8087370630888746

 32.174587    2.596963   -0.000000   -0.000000

  2.596963  -10.279636    0.162179   -0.000000

  0.000000    0.162179   -4.251291    0.000000

  0.000000    0.000000    0.000000   -0.808737

------------------------------------------------------------------------
  qr transformation, dim =  4  it =  3

  sigma =   -0.8087370600859152

 32.319449    0.757210    0.000000   -0.000000
            ------------------------
  0.757210 |-10.428189    0.057821 |  0.000000

  0.000000 |  0.057821   -4.247600 | -0.000000
            ------------------------
  0.000000    0.000000   -0.000000   -0.808737  <---

  eigenvalue found   -0.8087370600859153

  Iteration history for h^(k+1), n =  4
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1   -0.814750   -0.808738  0.131002E-02   -4.279886
  2   -0.808737   -0.808737  0.114148E-11   -4.251291
  3   -0.808737   -0.808737  -.368469E-28   -4.247600

------------------------------------------------------------------------
  qr transformation, dim =  3  it =  1

  sigma =   -4.2470591231683636

 32.332473    0.128273   -0.000000

  0.128273  -10.441753    0.000000

  0.000000    0.000000   -4.247060

------------------------------------------------------------------------
  qr transformation, dim =  3  it =  2
```

```
  sigma =   -4.2470604916666002
 ----------------------
|32.332847    0.021724 |  0.000000

| 0.021724  -10.442127 | -0.000000
 ----------------------
  0.000000   -0.000000   -4.247060  <---

  eigenvalue found   -4.2470604916665993

  Iteration history for h^(k+1), n =  3
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1   -4.247059   -4.247060  0.127718E-07  -10.441753
  2   -4.247060   -4.247060  -.188535E-23  -10.442127

------------------------------------------------------------------------
  qr transformation, dim =  2  it =  1

  sigma =  -10.4421376520426463

 32.332858 <---   0.000000                      Fertig!

 -0.000000      -10.442138 <---

  eigenvalue found  -10.4421376520426463

  Iteration history for h^(k+1), n =  2
  k    sigma_k      h(n,n)     h(n-1,n)     h(n-1,n-1)
  1  -10.442138  -10.442138  -.358393E-18   32.332858


   6  real eigenvalues found

  Eigenvalue  6  =    8.8058843287499435   Man beachte!
  Eigenvalue  5  =   10.3591928631745116   Eigenwerte nicht
  Eigenvalue  4  =   -0.8087370600859153   geordnet!
  Eigenvalue  3  =   -4.2470604916665993
  Eigenvalue  2  =  -10.4421376520426463
  Eigenvalue  1  =   32.3328580118706910
```

4.4.4 Komplexe Eigenwerte: Der QR-Doppelschritt

Wie wir im vorangegangenen Abschnitt (und im obigen Beispiel) gesehen haben, können bei der "Diagonalisierung" von reellen Matrizen natürlich auch konjugiert komplexe Eigenwerte auftreten, die wir mit unserer bisherigen Vorgehensweise nicht erfassen konnten: Einerseits ergibt die Wahl von $\sigma_k = h_{NN}^{(k)} \in \mathbb{R}$ entsprechend Gl. (4.60) keine vernünftige Näherung für einen komplexen Eigenwert, andererseits ergibt ein komplexer Wert von σ_k entsprechend Gl. (4.61) eine Matrix mit komplexer Diagonalen, die wir bislang nicht bearbeiten können. Der sog. "QR-Doppelschritt" löst diese Problematik. Als Vorbereitung auf diesen betrachten wir noch einmal die quadratische 2×2 Untermatrix in der "linken unteren" Ecke,

$$C_k = \begin{pmatrix} h_{N-1,N-1}^{(k)} & h_{N-1,N}^{(k)} \\ h_{N,N-1}^{(k)} & h_{NN}^{(k)} \end{pmatrix} \hat{=} \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

jetzt aber nicht nur mit den dazugehörigen reellen, sondern ggf. auch komplexen Eigenwerten und definieren

$$\begin{aligned} \lambda_1 &= \alpha + \sqrt{\beta} \qquad & \lambda_1 &= \alpha + i\sqrt{(-\beta)} \\ \lambda_2 &= \alpha - \sqrt{\beta} \qquad & \lambda_2 &= \alpha - i\sqrt{(-\beta)}, \end{aligned} \tag{4.65}$$

wobei

$$\alpha = \frac{a+d}{2} \qquad \text{und} \qquad \beta = \left(\frac{a+d}{2}\right)^2 - (ad - bc)$$

die Diskriminante des Problemes ist.

Falls die Eigenwerte nun komplex sind, definieren wir den QR-Doppelschritt als zweifache UR-Transformation, einmal mit $\sigma_k = \lambda_1$ und danach mit $\sigma_{k+1} = \lambda_2$, wobei die UR-Transformation das komplexe Analogon der QR-Transformation ist, und die zugehörige UR-Zerlegung durch folgenden Satz spezifiziert ist.

4.23 Satz. *Für jede komplexe Matrix $A \in \mathbb{C}^{N\times N}$ existiert die unitäre* UR*-Zerlegung*

$$A = U \cdot R, \tag{4.66}$$

wobei U eine unitäre Matrix ($U^\dagger = U^{-1}$, vgl. Kap. 4.1.2) und R eine komplexe Rechtsdreiecksmatrix mit reeller *Diagonalen ist (ohne Beweis).*

Der QR-Doppelschritt als zweifache UR-Transformation lässt sich nun folgendermaßen darstellen:

a) erste Zerlegung

$$H_k - \sigma_k \mathbb{1} = U_k R_k \tag{4.67a}$$

wobei $(H_k - \sigma_k \mathbb{1})$ eine komplexe Diagonale hat (s.o.).

b) und zugehörige Transformation

$$H_{k+1} = R_k U_k + \sigma_k \mathbb{1} = U_k^\dagger H_k U_k \tag{4.67b}$$

c) zweite Zerlegung

$$H_{k+1} - \sigma_{k+1} \mathbb{1} = U_{k+1} R_{k+1} \tag{4.67c}$$

d) und zugehörige Transformation

$$H_{k+2} = R_{k+1} U_{k+1} + \sigma_{k+1} \mathbb{1} = U_{k+1}^\dagger H_{k+1} U_{k+1} \tag{4.67d}$$

Mit (4.67b) finden wir also zunächst

$$H_{k+2} = U_{k+1}^\dagger U_k^\dagger H_k U_k U_{k+1} = \underbrace{(U_k U_{k+1})}_{\text{unitär}}{}^\dagger H_k \left(U_k U_{k+1}\right),$$

dass H_{k+2} eine unitär-ähnliche Transformation von H_k ist!
Andererseits gilt mit (4.67c)

$$\underbrace{H_{k+1}}_{U_k^\dagger H_k U_k} - \sigma_{k+1}\mathbb{1} = U_{k+1} R_{k+1}$$

$$U_k^\dagger H_k U_k - \sigma_{k+1}\mathbb{1} = U_{k+1} R_{k+1}$$

(von links mit U_k, von rechts mit $U_k^\dagger$ multipliziert) $\Rightarrow$

$$H_k - \sigma_{k+1} I = U_k U_{k+1} R_{k+1} U_k^\dagger$$

(von rechts mit $U_k R_k \overset{(4.67a)}{=} H_k - \sigma_k \mathbb{1}$ multipliziert) $\Rightarrow$

$$\Rightarrow \quad \underbrace{\left(H_k - \sigma_{k+1} I\right)\left(H_k - \sigma_k I\right)}_{:=X} = U_k U_{k+1} R_{k+1} R_k \tag{4.68}$$

Die Matrix X ist ist nun reell, wenn H_k reell ist, da

$$\begin{aligned} X &= H_k^2 - (\sigma_k + \sigma_{k+1})\, H_k + \sigma_k \sigma_{k+1} \mathbb{1} \\ &= H_k^2 - sH_k + t\mathbb{1}, \end{aligned} \tag{4.69}$$

und $s, t \in \mathbb{R}$ sind:

$$\begin{aligned} s &= 2\alpha = h_{N-1,N-1}^{(k)} + h_{NN}^{(k)} \\ t &= \alpha^2 - \beta = (ad - bc) = h_{N-1,N-1}^{(k)} h_{NN}^{(k)} - h_{N-1,N}^{(k)} h_{N,N-1}^{(k)} \end{aligned} \tag{4.70}$$

Also lässt sich Gl. (4.68) als eine einzige *unitäre* UR- Zerlegung der *reellen* Matrix X

$$X = \underbrace{U_k U_{k+1}}_{\text{unitär}} \cdot \underbrace{R_k R_{k+1}}_{\text{Rechsdreiecksmatrix mit reeller Diagonale}}$$

interpretieren. Da X reell ist, existiert natürlich auch eine entsprechende orthogonale QR Zerlegung, nämlich

$$X = Q \cdot R$$

Demzufolge können also U_k, U_{k+1} so gewählt werden, dass ihr Produkt

$$U_k U_{k+1} =: Q \tag{4.71}$$

eine orthogonale Matrix ist. Mit dieser Wahl gilt dann

$$H_{k+2} = (U_k U_{k+1})^\dagger\, H_k\, (U_k U_{k+1}) = Q^\intercal H_k Q \tag{4.72}$$

und H_{k+2} ist orthogonal ähnlich zu H_k, und wiederum eine HESSENBERG-Matrix (UR-Transformationen erhalten die HESSENBERG-Form).
Damit ergibt sich folgender Algorithmus für die Lösung des Eigenwertproblemes einer reellen Matrix mit komplexen Eigenwerten

i) Man berechne $X = H_k^2 - sH_k + t\mathbb{1}$ mit s und t aus σ_k, $\sigma_{k+1} = \lambda_{1,2}$ (4.65, 4.70),

ii) zerlege dann die reelle Matrix $X = Q \cdot R$ und

iii) führe die Transformation mit diesem Q durch: $H_{k+2} = Q^\intercal H_k Q$.

iv) Die explizite Subtraktion und (finale) Addition der σ, die im Falle von reellen Eigenwerten notwendig war, entfällt hier, da die Spektralverschiebung schon bei der Berechnung von X berücksichtigt wird und bei der *Transfomation* (d.h. der Rechtmultiplikation mit $Q = U_k U_{k+1}$) wieder rückgängig gemacht wird.

Damit ist ein QR-Doppelschritt fertig, und das Verfahren wird wie im reellen Falle iterativ zur Konvergenz gebracht (Wenn die Matrix zerfallen ist, erniedrigt sich ihre Ordnung um zwei!). So wie hier beschrieben, ist das Verfahren äußerst einfach zu programmieren, aber auch sehr rechenintensiv, da zur Berechnung von H_k^2 $\mathcal{O}(N^3)$ Operationen notwending sind. Abhilfe schafft hier der sog. QR-Doppelschritt mit *impliziter Spektralverschiebung*, auf die schon früher verwiesen wurde.

4.24 Beispiel (QR-Doppelschritt). Im Folgenden zeigen wir den `output` eines Programmes, das den QR-Doppelschritt mit expliziter Spektralverschiebung zur Berechnung von komplexen Eigenwerten verwendet. Die Berechnung setzt das vorhergehende Beispiel 4.22 fort (allgemeine Matrix), das nach Auffinden von komplexen Lösungen für die Untermatrix C_k stoppte. Für die restlichen zwei Paare von konjugiert komplexen Eigenwerten wird nur noch *eine* Sequenz von Doppelschritten benötigt, da nach Konvergenz die reduzierte Matrix die Ordnung 2×2 hat, deren Eigenwerte sich direkt berechnen lassen.
Das Konvergenzverhalten (hier Subdiagonalelement $h_{N-1,N-2}$!) wird wiederum protokolliert, die Konvergenz ist hier nur linear.

```
  Fortsetzung des Beispiels QR Zerlegung einer asymmetrischen Matrix
  Simpler QR Doppelschritt
  .
  .
  .
--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  1

  imaginary eigenvalues found, qr double step
  sigma =    1.0005278725193647  +/-    2.0017645113381772 i

  5.000000   6.000000  -6.209540  -9.174597

 -6.000000   5.000000   0.306369  -1.464549

  0.000000   0.000000  -0.560085   5.281168
                ^ Dies ist das entscheidende Subdiagonalelement
  0.000000   0.000000  -1.218265   2.560085

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  2

  imaginary eigenvalues found, qr double step
  sigma =    0.9999999459025395  +/-    1.9999997250419024 i

  5.000000   6.000000  -2.107436  -4.094790

 -6.000000   5.000000   6.807328   7.577735

  0.000000  -0.000000  -0.109134   5.558500

  0.000000   0.000000  -0.940933   2.109134

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  3

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777189  +/-    1.9999998996675394 i

  5.000000   6.000000   7.998802   2.073761

 -6.000000   5.000000   6.792018   3.249238

  0.000000  -0.000000   2.515501   5.314642

  0.000000   0.000000  -1.184791  -0.515501

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  4

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777189  +/-    1.9999998996675392 i

  5.000000   6.000000   6.214435   7.474542

 -6.000000   5.000000  -4.383515  -3.356664

  0.000000  -0.000000   0.165761   5.671432

  0.000000   0.000000  -0.828002   1.834240

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  5

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777185  +/-    1.9999998996675388 i

  5.000000   6.000000  -2.223165  -0.736298

 -6.000000   5.000000  -7.588071  -7.868098

  0.000000  -0.000000   0.353232   5.728093
```

```
  0.000000   0.000000  -0.771340   1.646768

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  6

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777185  +/-    1.9999998996675390 i

  5.000000   6.000000 -10.233922  -4.117060

 -6.000000   5.000000  -0.948748  -1.544964

  0.000000  -0.000000   2.282323   5.466992

  0.000000   0.000000  -1.032442  -0.282323

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  7

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777185  +/-    1.9999998996675392 i

  5.000000   6.000000  -1.748923  -4.659400

 -6.000000   5.000000   5.655283   8.259504

  0.000000  -0.000000  -0.688402   5.175842

  0.000000   0.000000  -1.323591   2.688402

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  8

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777187  +/-    1.9999998996675394 i

  5.000000   6.000000   7.221369   3.388980

 -6.000000   5.000000   6.325924   4.617327

  0.000000  -0.000000   1.631712   5.731974

  0.000000   0.000000  -0.767460   0.368288

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  9

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777187  +/-    1.9999998996675399 i

  5.000000   6.000000   0.340175   9.912056

 -6.000000   5.000000  -1.371709  -4.972345

  0.000000  -0.000000  -1.551258   3.022249

  0.000000   0.000000  -3.477184   3.551258

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  10

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777187  +/-    1.9999998996675394 i

  5.000000   6.000000  -1.588392   1.186177

 -6.000000   5.000000 -10.979281   0.704098

  0.000000  -0.000000   3.548971   3.501527

  0.000000   0.000000  -2.997907  -1.548971

--------------------------------------------------------------------------------
```

```
  qr transformation, dim =  4  it =  11

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777187  +/-    1.9999998996675397 i

  5.000000   6.000000 -10.261641  -3.872055

 -6.000000   5.000000  -1.366691  -1.675842

  0.000000  -0.000000   2.361016   5.419577

  0.000000   0.000000  -1.079856  -0.361016

--------------------------------------------------------------------------------
  qr transformation, dim =  4  it =  12

  imaginary eigenvalues found, qr double step
  sigma =    1.0000000371777187  +/-    1.9999998996675394 i

  5.000000   6.000000  -1.938838  -4.978659

 -6.000000   5.000000   5.546738   8.102958
                      ---------------------
  0.000000  -0.000000 |-0.706471   5.159852 |      Die Matrix zerfällt
                                                   nach 12 Iterationen
  0.000000   0.000000 |-1.339582   2.706471 |
                      ---------------------
       Aus dieser Untermatrix berechnen sich die Eigenwerte

  imaginary eigenvalues found    1.0000000371777187 +/-   1.9999998996675397 i

  Iteration history for h^(k+1), n =  4
  k    h(n-1,n-2)     h(n-2,n-2)
  1  0.374739E-06    5.000000
  2  -.451217E-07    5.000000
  3  -.122348E-07    5.000000
  4  -.342740E-08    5.000000
  5  -.108305E-08    5.000000   lineare
  6  -.316798E-09    5.000000   Konvergenz
  7  -.814162E-10    5.000000
  8  -.227833E-10    5.000000
  9  -.416469E-11    5.000000
 10  -.628569E-14    5.000000
 11  -.178455E-14    5.000000
 12  -.451063E-15    5.000000

--------------------------------------------------------------------------------
  qr transformation, dim =  2  it =  1

  imaginary eigenvalues found, qr double step
  sigma =    4.9999999628222831  +/-    6.0000000193572891 i

  5.000000   6.000000     Reduzierte Matrix,
                          Eigenwerte können sofort bestimmt werden
 -6.000000   5.000000

  imaginary eigenvalues found    4.9999999628222831 +/-   6.0000000193572891 i

  Zusammenfassung der Eigenwerte (reell und komplex)

  Imag eigenvalue  1  =    4.9999999628222831 +   6.0000000193572891 i
  Imag eigenvalue  2  =    4.9999999628222831 -   6.0000000193572891 i
  Imag eigenvalue  3  =    1.0000000371777187 +   1.9999998996675397 i
  Imag eigenvalue  4  =    1.0000000371777187 -   1.9999998996675397 i
  Real eigenvalue  5  =    3.9999999999999787
  Real eigenvalue  6  =    3.0000000000000164
```

4.5 Ergänzungen

4.5.1 Balancing

Wie sich zeigen lässt, skalieren die Fehler der Eigenwerte mit der EUKLIDischen oder FROBENIUS Matrixnorm, $\left(\sum_{i,k} a_{ik}^2\right)^{\frac{1}{2}}$.

Mit dem hier kurz vorgestellten "Balancing" lässt sich diese Norm und damit die Fehler der Eigenwerte reduzieren. Dies wird durch eine Folge von Ähnlichkeitstransformationen

$$A_{i+1} = D_i^{-1} A_i D_i \quad , \quad i = 1, 2, \ldots$$

mit geeignet gewählten Diagonalmatrizen D_i erzielt. Für die "ausbalancierte" Matrix gilt dann

$$\sum_{j=1,N} a_{ij}^2 \approx \sum_{j=1,N} a_{ji}^2 \quad \forall i = 1, \ldots N \tag{4.73}$$

d.h. die Norm der Zeilen und zugehörigen Spalten ist ungefähr gleich groß. Man beachte, dass symmetrische Matrizen per Definition "ausbalanciert" sind. Bzgl. des Verfahrens verweisen wir auf "Numerical Recipes", Kap. 11.5.

4.25 Beispiel (Balancing einer Matrix).

$$\begin{pmatrix} 1 & 2 & 300 & 4 & 5 \\ 2 & 3 & 400 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 600 & 7 & 8 \\ 5 & 6 & 700 & 8 & 9 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} 1 & 2 & 37.5 & 4 & 5 \\ 2 & 3 & 50 & 5 & 6 \\ 24 & 32 & 5 & 48 & 56 \\ 4 & 5 & 75 & 7 & 8 \\ 5 & 6 & 87.5 & 8 & 9 \end{pmatrix}$$

$$\left(\sum_{i,k} a_{ik}^2\right)^{\frac{1}{2}} \approx 1050 \qquad \left(\sum_{i,k} a_{ik}^2\right)^{\frac{1}{2}} \approx 157$$

4.5.2 Bestimmung der Eigenvektoren nach Durchführung des QR-Algorithmus

Im Folgenden wollen wir kurz diskutieren, wie man nach einer Bestimmung der Eigenwerte mittels QR-Algorithmus die dazugehörigen Eigenvektoren ermittelt. Aufgrund der in Praxis auftretenden Schwierigkeiten, mit dem dieses Problem behaftet ist, raten wir jedoch dazu, wann irgendmöglich entsprechende *Bibliotheksroutinen*, z.B. EISPACK[8] zu verwenden.

Direkte Berechnung

Die direkte Berechnung ist sehr aufwendig, da sie eine konsequente Buchhalten über alle orthogonalen Transformationen erfordert, die letztendlich zu der die Eigenwerte darstellenden Quasidreiecksmatrix R (4.54) geführt haben. Dies sind die Transformationen

$$Q = Q_1 Q_2 \ldots Q_M,$$

wenn die Ausgangsmatrix A über GIVENS in eine HESSENBERG-Matrix H und diese auf R QR-transformiert wurde, so dass final

$$R = Q^\intercal A Q$$

gilt. Mit der originalen Eigenwertaufgabe

$$A \cdot \boldsymbol{x} = \lambda \cdot \boldsymbol{x}$$

[8] basierend auf Wilkinson & Reinsch, 1971, Linear Algebra, vol. II of Handbook for Automatic Computation (New York, Springer)

folgt dannn

$$\underbrace{Q^\intercal A Q}_{R} \cdot \underbrace{Q^\intercal \boldsymbol{x}}_{\boldsymbol{y}} = \lambda \underbrace{Q^\intercal \boldsymbol{x}}_{\boldsymbol{y}},$$

d.h. die Eigenwertaufgabe reduziert sich zunächst auf die Lösung von

$$R \cdot \boldsymbol{y} = \lambda \boldsymbol{y} \qquad \text{mit} \quad \boldsymbol{y} := Q^\intercal \boldsymbol{x},$$

was sich aufgrund der Quasidreieckstruktur von R bei bekanntem λ_i einfach durchführen lässt. Zur Bestimmung der tatsächlich gesuchten Eigenvektoren ist dann allerding die Kenntnis von Q unerlässlich,

$$\boldsymbol{x}_i = Q \cdot \boldsymbol{y}_i.$$

Inverse Iteration

Das zweite Verfahren ist allgemeinerer Natur und setzt nur voraus, dass man die Eigenwerte einer Matrix zuvor schon bestimmt hat (wie auch immer). Aufgrund der iterativen Natur dieses Verfahrens eignet es sich auch zu einer Verbesserung der Qualität von schon zuvor bestimmten Eigenvektoren, z.B. mittels des JACOBI-Verfahrens (vgl. Kap. 4.2.4).

Wir nehmen dazu an, dass für die Originalmatrix A ein numerischer Näherungswert $\bar{\lambda}$ für den entsprechenden "exakten" Eigenwert λ_i bekannt ist. Definiert man nun die Folge

$$\left(A - \bar{\lambda}\mathbb{1}\right) \boldsymbol{z}^{(k)} = \boldsymbol{z}^{(k-1)} \tag{4.74}$$

mit einem geeignet definierten Startwert $\boldsymbol{z}^{(0)}$, zum Beispiel $(1, 1, 1, \ldots, 1)^\intercal$ oder einem Schätzwert für den gesuchten Eigenvektor, dann konvergiert die Folge $\boldsymbol{z}^{(k)}$, $k = 1, 2, \ldots$ (unter einigen Einschränkungen) gegen den zu λ_i gehörigen Eigenvektor.

Der Beweis zu dieser Aussage wird im Folgenden skizziert. Unter der (fast immer gerechtfertigten) Annahme, dass A einen vollständigen Satz von Eigenvektoren besitzt ($\boldsymbol{x}_j$, $j = 1, \ldots N$), lassen sich die iterierten Werte $\boldsymbol{z}$ nach diesen Eigenvektoren entwickeln, d.h.

$$\begin{aligned} \boldsymbol{z}^{(k-1)} &= \sum \beta_j \boldsymbol{x}_j \\ \boldsymbol{z}^{(k)} &= \sum \alpha_j \boldsymbol{x}_j \end{aligned}$$

Einsetzen in (4.74) liefert

$$\begin{aligned} \left(A - \bar{\lambda}\mathbb{1}\right) \sum \alpha_j \boldsymbol{x}_j &= \sum \beta_j \boldsymbol{x}_j \\ \sum \alpha_j \left(\lambda_j - \bar{\lambda}\right) \boldsymbol{x}_j &= \sum \beta_j \boldsymbol{x}_j \end{aligned}$$

$$\Rightarrow \quad \alpha_j = \frac{\beta_j}{\lambda_j - \bar{\lambda}}.$$

Demzufolge lässt sich also $\boldsymbol{z}^{(k)}$ durch die Koeffizienten von $\boldsymbol{z}^{(k-1)}$ ausdrücken,

$$\boldsymbol{z}^{(k)} = \sum \frac{\beta_j}{\left(\lambda_j - \bar{\lambda}\right)} \boldsymbol{x}_j.$$

Sei nun der Startvektor durch die Entwicklung

$$\boldsymbol{z}^{(0)} = \sum c_j \boldsymbol{x}_j$$

gegeben, dann folgt sofort, dass

$$\boldsymbol{z}^{(k)} = \sum \frac{c_j}{\left(\lambda_j - \bar{\lambda}\right)^k} \boldsymbol{x}_j \tag{4.75}$$

ist. (Man beachte die Potenz k im Nenner!). Falls nun der Eigenwert nicht entartet und $\bar{\lambda}$ eine guter Schätzwert für λ_i ist,

$$c_i \neq 0 \quad \text{und} \quad 0 < |\lambda_i - \bar{\lambda}| \ll \min_{j \neq i} |\lambda_j - \bar{\lambda}| \tag{4.76}$$

dann gilt mit 4.75

$$\boldsymbol{z}^{(k)} = \frac{1}{(\lambda_i - \bar{\lambda})^k} \Big(c_i \boldsymbol{x}_i + \underbrace{\sum_{j \neq i}^{N} c_j \frac{(\lambda_i - \bar{\lambda})^k}{(\lambda_j - \bar{\lambda})^k} \boldsymbol{x}_j}_{\to 0} \Big) \tag{4.77}$$

dass $\boldsymbol{z}^{(k)}$ rasch gegen den gesuchten Eigenvektor $\boldsymbol{x}_i$ konvergiert, falls nach jeder Iteration $\boldsymbol{z}^{(k)}$ normiert wird!

Das Problem dieser Vorgehensweise liegt natürlich darin, dass wir de facto das lineare Gleichungssystem

$$\left(A - \bar{\lambda}\mathbb{1}\right) \boldsymbol{y} = \boldsymbol{b}$$

mit $\boldsymbol{y} = \boldsymbol{z}^{(k)}$ und $\boldsymbol{b} = \boldsymbol{z}^{(k-1)}$ lösen müssen, zum Beispiel über LU-Zerlegung (pro Eigenwert $\bar{\lambda}$ ist dann nur eine einmalige Zerlegung notwendig). Die (zu zerlegende) Matrix $(A - \bar{\lambda}\mathbb{1})$ ist aber nahezu *singulär*, da $\bar{\lambda} \approx \lambda_i$, d.h. das Gleichungssystem ist äußerst schlecht konditioniert, was zu großen Fehlern in der Lösung führt (vgl. Kap. 3), egal wie "gut" der Lösungsalgorithmus ist.

Dass die inverse Iteration überhaupt vernünftige Resultate liefert, liegt nun daran, dass im allgemeinen dieser Fehler, der bei der Lösung von linearen Gleichungssystemen gemacht wird, hier:

$$\bar{\boldsymbol{z}}^{(k)} - \boldsymbol{z}^{(k)}$$

(mit $\bar{\boldsymbol{z}}^{(k)}$ der numerische und $\boldsymbol{z}^{(k)}$ der "exakten" Lösung), eine dominante Komponente in Richtung des Eigenvektors des betragskleinsten Eigenwertes der das Gleichungssytem darstellenden Matrix (hier: $(A - \bar{\lambda}\mathbb{1})$) hat.

Auf Grund der Voraussetzung (4.76) ist der betragskleinste Eigenwert aber $|(\lambda_i - \bar{\lambda})|$, d.h. der resultierende numerische Fehler liegt hauptsächlich in Richtung des gesuchten Eigenvektors $\boldsymbol{x}_i$, d.h. wirkt sich kaum aus! *Das ist der eigentliche Grund, warum die inverse Iteration funktioniert.*

Kapitel 5

Ausgleichsprobleme

Die Bestimmung der Parameter eines (vorgegebenen) funktionalen Zusammenhanges anhand von experimentellen Daten ist eine Aufgabe, die sich im Rahmen physikalischer Untersuchungen (und darüber hinaus) äusserst häufig stellt.

In diesem Kapitel werden wir die entsprechenden Methoden zur Gewinnung dieser Parameter kennenlernen, sowohl bezüglich linearer als auch nichtlinearer Probleme [1]. Wir werden aber nicht nur die (numerische) *Bestimmung* dieser Parameter diskutieren, sondern insbesondere auch die *"Qualität"* der abgeleiteten Parameter betrachten, d.h. mit welchem Fehler sie behaftet sind und ob der vorgegebene funktionale Zusammenhang überhaupt mit den Daten kompatibel ist.

Um all diese Fragen zu beantworten, wird an einigen Stellen ein kleiner Exkurs in die Behandlung statistischer Probleme unumgänglich sein.

5.1 Motivation

Als einführendes Beispiel betrachten wir die Ausgleichung für eine lineare, zweiparametrige Funktion, bekannt unter dem Begriff *Methode der kleinsten Quadrate* oder *least square fitting*.
Wir nehmen dazu an, dass wir in einem bestimmten Experiment folgende Werte y als Funktion der Zeit t zu $N = 7$ Zeitpunkten gemessen haben:

t	1	2	3	4	5	6	7
y	1.2	1.9	3.1	4.2	2.0	6.5	6.8

Diese Messwerte sind in Abb. 5.1 visualisiert. Betrachten wir die Daten, so liegt es nahe, einen *linearen* Zusammenhang zwischen y und t anzunehmen,

$$y = a + bt.$$

Die Aufgabe der Ausgleichung ist es nun, die entsprechenden Parameter a, b aus der Messreihe abzuleiten. (Eine analoge Aufgabe wäre folgende: Theoretische Überlegungen haben ergeben, dass ein bestimmter physikalischer Prozess durch obige Gleichung beschrieben werden kann. Zur Überprüfung dieser Überlegung werde ein Experiment durchgeführt, wobei die Parameter durch die Ausgleichung zu bestimmen sind und ggf. mit der theoretischen Vorhersage verglichen werden sollen.)
Um diese Aufgabe zu lösen, definiert man zunächst die sog. Residuen,

$$r_i = y_i - y(t_i) \qquad \text{mit} \qquad y(t_i) = a + bt_i, \qquad i = 1, \ldots, N,$$

d.h. die Abweichungen der experimentellen von den "theoretischen" Werten und bestimmt die gesuchten Parameter a, b dadurch, dass die euklidische Norm des Residuen*vektors* $\boldsymbol{r}$ minimiert werden

[1] Der Begriff der (Nicht-)Linearität bezieht sich in diesem Zusammenhang darauf, wie die zu bestimmenden Parameter in den funktionalen Zusammenhang eingehen.

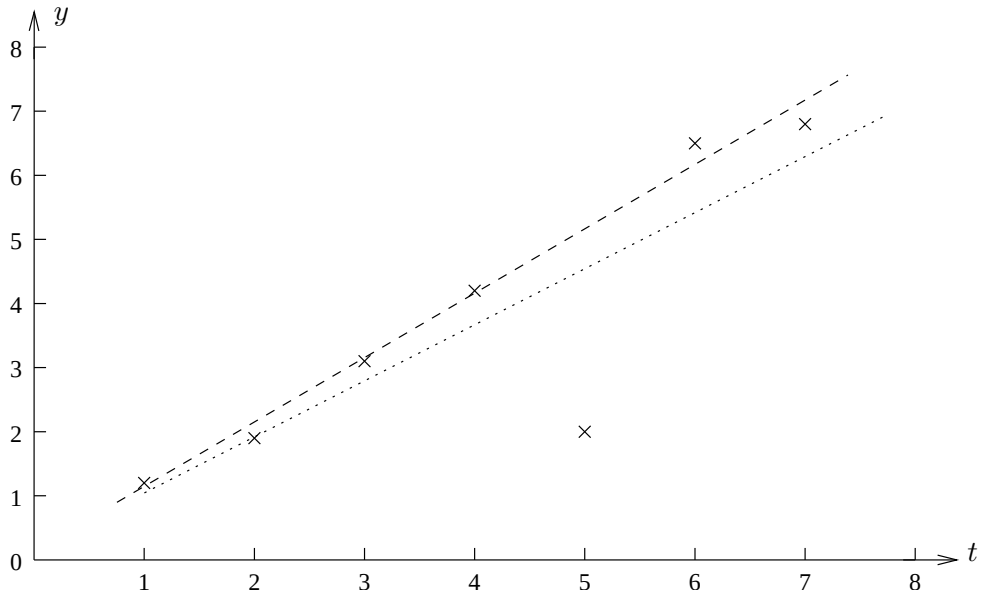


Abbildung 5.1: Messreihe (Datenpunkte) und mögliche Ausgleichsgeraden. Vgl. mit Beispiel 5.1.

soll. Mit dieser Minimierung der *quadratischen* Abweichung soll also die "Länge" des Fehlervektors minimiert werden,

$$\|\boldsymbol{r}\|_2^2 = \sum_{i=1}^{N} r_i^2 = \sum_{i=1}^{N} \left(y_i - y(t_i)\right)^2 = \min! \tag{5.1}$$

beziehungsweise

$$\sum_{i=1}^{N} \left(y_i - (a + b\, t_i)\right)^2 = \min!$$

Das Minimum ergibt sich durch Nullsetzen der partiellen Ableitungen bzgl. a, b

$$\begin{aligned} \frac{\partial}{\partial a} &= \sum \left[2\left(y_i - (a + bt_i)\right)(-1)\right] = 0 \\ \frac{\partial}{\partial b} &= \sum \left[2\left(y_i - (a + bt_i)\right)(-t_i)\right] = 0. \end{aligned}$$

Nach Umordnung (alle Summen erstrecken sich von $i = 1, \ldots, N$)

$$\begin{aligned} aN + b\sum t_i &= \sum y_i \\ a\sum t_i + b\sum t_i^2 &= \sum t_i y_i \end{aligned}$$

und Einführung der Abkürzungen

$$\sum y_i = S_y, \qquad \sum t_i = S_t, \qquad \sum t_i y_i = S_{ty}, \qquad \sum t_i^2 = S_{tt}$$

lässt sich das Problem als lineares Gleichungssystem für die zwei Unbekannten a, b formulieren,

$$\begin{pmatrix} N & S_t \\ S_t & S_{tt} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} S_y \\ S_{ty} \end{pmatrix}, \tag{5.2}$$

dessen Lösung durch

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} S_{tt}S_y - S_t S_{ty} \\ N S_{ty} - S_t S_y \end{pmatrix} \qquad \text{mit} \qquad \Delta = N S_{tt} - S_t^2 \tag{5.3}$$

gegeben ist.

5.1 Beispiel (Messreihe Abb. 5.1). Für unser erstes Beispiel lauten die entsprechenden Größen $N = 7$, $S_y = 25.7$, $S_t = 28$, $S_{ty} = 127.7$, $S_{tt} = 140$ und $\Delta = 196$. Damit erhält man als gesuchte Parameter der Ausgleichung $a = 0.114286$ und $b = 0.889286$, die zur gestrichelten *Ausgleichsgeraden* in Abbildung 5.1 führen.

Es stellt sich nun die Frage, ob der "Ausreißer" y_5 ein Messfehler ist. Unter dieser Hypothese ergeben sich durch "Weglassen" des Punktes "5" (mit $N = 6$) die Parameter $a_1 = 0.114286$ (dieser Wert bleibt zufällig unverändert) und $b_1 = 1.00062$, entsprechend der gepunkteten Ausgleichsgeraden in Abb. 5.1.

Schon dieses einfache Beispiel führt zu mehreren offensichtlichen Fragen:

- Wie lässt sich Einfluss von Messfehlern *quantitativ* berücksichtigen?
- Wie groß sind die *Unsicherheiten* in den Parametern der Ausgleichung $a \pm \Delta a$, $b \pm \Delta b$?
- Die Parameter a, b lassen sich immer ausrechnen, unabhängig davon, ob die Ausgleichsfunktion $y = a + bt$ den "tatsächlichen" Gegebenheiten entspricht oder nicht!

  Wie lässt sich also die "Güte des Fits" quantifizieren, d.h. die Wahrscheinlichkeit, dass die gewählte Ausgleichsfunktion mit dem Verlauf der Messpunkte kompatibel ist?

  Falls in unserem Beispiel alle Messpunkte einen Fehler von $\Delta y_i = \pm 0.2$ hätten, dann wäre die Ausgleichung mit ziemlicher Sicherheit falsch! Diese Aussage sollte ebenfalls quantifiziert werden können.
- Schließlich sollte das Vorgehen nicht nur auf Geraden, sondern auf unterschiedliche Ausgleichsfunktionen anwendbar sein, z. B. auf Parabeln,

  $$y = a + bt + ct^2.$$

  Wie lautet das entsprechende Vorgehen?

5.2 Die χ^2-Minimierung

Um diese Fragen beantworten zu können, muss das Problem von einem etwas allgemeineren Standpunkt aus betrachtet werden. Dazu ist eine kleiner Exkurs in die Statistik (Modellierung von Daten) unabdingbar.

Problem. Es seien N Datenpunkte $(x_i,\ y_i)$, $i = 1, \ldots, N$ für ein Modell gegeben, welches die funktionale Abhängigkeit

$$y(x) = y(x, a_1, \ldots, a_M)$$

vorhersagt, und $a_1, \ldots, a_M$ seien adjustierbare Parameter des Modelles mit $N > M$ (im vorhergehenden Beispiel entspricht dies $M = 2$, $a_1 = a$ und $a_2 = b$).

- Berechenbar ist die Wahrscheinlichkeit, dass bei *gegebenen* Parametern $a_1, \ldots, a_M$ die (Mess-) Werte y_i innerhalb gewisser Fehlergrenzen $\pm\Delta y$ auftreten.
- Diese Wahrscheinlichkeit wird mit der gesuchten Wahrscheinlichkeit für die Parameter $a_1, \ldots, a_M$ identifiziert, wenn die *Daten* $y_i \pm \Delta y$ *gegeben* sind. Man beachte, dass diese Identifikation rein *intuitiv* und keine formale Ableitung dieser Identifikation möglich ist.

Unter der **Annahme**, dass jeder Datenpunkt y_i einen Messfehler habe, der

- zufällig,
- unabhängig und
- normalverteilt (Gaussverteilt) mit Standardabweichung σ_i

um das "wahre" Modell $y(x)$ ist, ergibt sich die Wahrscheinlichkeit, dass bei einem solchen Modell der (Mess-)Wert

$$y_i \pm \Delta y$$

angenommen wird, als

$$P_i \propto \exp\left(-\frac{1}{2}\frac{(y_i - y(x_i))^2}{\sigma_i^2}\right)\Delta y.$$

Die Gesamtwahrscheinlichkeit, dass der gesamte Datensatz $y_i \pm \Delta y$, $i = 1, \ldots, N$ auftritt, ist das Produkt der Einzelwahrscheinlichkeiten.

Unter der weiteren Voraussetzung, dass die Standardabweichungen der Normalverteilung für alle Datenpunkte gleich sind (diese Einschränkung wird später fallengelassen), ergibt sich

$$P \propto \prod_{i=1}^{N}\left[\exp\left(-\frac{1}{2}\frac{(y_i - y(x_i))^2}{\sigma^2}\right)\Delta y\right]. \tag{5.4}$$

Falls nun die Wahrscheinlichkeiten für den Datensatz (sehr) klein sind, gelangen wir zu der Schlussfolgerung, dass der Parametersatz "höchstwahrscheinlich falsch" ist.

Aufgrund obiger Identifikation ist andererseits die größte Wahrscheinlichkeit (*"maximum likelihood"*) für den Parametersatz $a_1, \ldots, a_M$ dann gegeben, wenn das Maximum der Wahrscheinlichkeit für den Datensatz y_i realisiert wird! Die

$$\begin{aligned} & \text{Maximierung der Wahrscheinlichkeit } P \\ \hat{=}\quad & \text{Maximierung des (natürlichen) Logarithmus der Wahrscheinlichkeit, } \ln P \\ \hat{=}\quad & \text{Minimierung von } (-\ln P) \end{aligned}$$

Der Parametersatz $a_1, \ldots, a_M$ ist also dann am wahrscheinlichsten, wenn der Ausdruck

$$\left[\sum_{i=1}^{N}\frac{(y_i - (x_i))^2}{2\sigma^2}\right] - N\ln\Delta y$$

sein Minimum annimmt! Da $N, \Delta y$ und σ Konstanten sind, ist diese Forderung äquivalent mit der Forderung

$$\sum_{i=1}^{N}(y_i - y(x_i))^2 \quad = \quad \min!$$

und ein Vergleich mit Gl. (5.1) zeigt, dass die Methode der kleinsten Quadrate die größte Wahrscheinlichkeit für die gesuchten Parameter ergibt, *wenn die Messfehler unabhängig normalverteilt mit gleicher Standardabweichung* sind.

Um die Vorgehensweise zu verallgemeinern, habe jetzt jeder Datenpunkt (x_i, y_i) eine *individuelle* Standardabweichung σ_i. Damit ergibt sich für die entsprechende Gesamtwahrscheinlichkeit (5.4) (unabhängige, normalverteilte Fehler vorausgesetzt)

$$P \propto \prod_{i=1}^{N}\left[\exp\left(-\frac{1}{2}\frac{(y_i - y(x_i))^2}{\sigma_i^2}\right)\Delta y\right], \tag{5.5}$$

und es gilt, das Minimum der Funktion

$$\sum_{i=1}^{N}\frac{(y_i - y(x_i, a_1, \ldots, a_M))^2}{\sigma_i^2} =: \chi^2 \tag{5.6}$$

zu finden.

Zusammenfassend stellen wir also fest: Wenn die oben eingeführte Größe χ^2 ihr Minimum annimmt, dann ist die Messung des Datensatzes y_i, $i = 1, \ldots, N$ bei einem "wahren" Modell $y(x, a_1, \ldots, a_M)$ am wahrscheinlichsten (sofern die Messfehler den o.g. Einschränkungen genügen). Der entsprechende"Rückschluss" lautet dann, dass bei Messung dieses Datensatzes die Parameter $a_1, \ldots, a_M$ am wahrscheinlichsten sind. Diese sog. χ^2-*Minimierung* ist *das* Standardverfahren zur Modellierung von Daten.

5.3 Die Güte des Fits

Da die Messfehler zufällig verteilt sind (bzw. dies laut Voraussetzung sein sollen!), werden bei verschiedenen Messreihen natürlich auch unterschiedliche Datensätze realisiert. Pro Messreihe lässt sich jeweils ein minimiertes χ^2 (mit den dazugehörigen Parametern $a_1, \ldots, a_M$) berechnen.

Diese verschiedenen χ^2 folgen wiederum einer Verteilung, der sog. $\chi^2 - Verteilung$

$$p(\chi^2, f),$$

wobei "p" eine sog. Wahrscheinlichkeitsdichteverteilung" ist, mit der Eigenschaft, dass $p(x)\mathrm{d}x$ die Wahrscheinlichkeit dafür ist, dass ein Wert im Intervall $[x, x + dx]$ realisiert wird (vgl. Kap. 8).

Diese $\chi^2 - Verteilung$ hängt nun nicht nur von χ^2, sondern auch von der Zahl der sog.

Freiheitsgrade f

ab. Die Zahl der Freiheitsgrade ist normalerweise die Zahl der Terme in der Summe (5.6), d.h. die Zahl der Datenpunkte.

Wenn χ^2 minimiert ist, sind allerdings nicht mehr alle Terme der Summe statistisch unabhängig, und zumindest für *lineare Modelle* $\left(y = \sum_{k=1}^{M} a_k f_k(x), \text{ s.u.}\right)$ gilt

$$f = N - M \tag{5.7}$$

5.3.1 Die χ^2-Verteilung

Zunächst wollen wir die eigentliche Bedeutung der χ^2-Verteilung in einfacher Weise interpretieren und wichtige Eigenschaften angeben. Laut Definition war

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y}{\sigma_i} \right)^2 .$$

(a) Für ein "perfektes" Modell resultiert der Unterschied $y_i - y$ allein aus Messfehlern, und diese sollen laut Annahme normalverteilt mit Standardabweichung σ_i sein.

Damit ist die Größe $x_i = \dfrac{y_i - y}{\sigma_i}$ eine normalverteilte "Zufallsvariable" (siehe Kap. 8) mit Erwartungswert ("Mittelwert") 0 und Varianz (Quadrat der Standardabweichung) 1.

(b) Damit lässt sich

$$\chi^2 = \sum_{i=1}^{N} x_i^2 \tag{5.8}$$

als Summe von N Quadraten von normalverteilten Zufallsvariablen mit Mittelwert 0 und Varianz 1 deuten. Aufgrund der Normalverteilung liegen dabei

$\sim 67\%$ dieser Zufallsvariablen x_i im Intervall ± 1,

$\sim 95\%$ dieser Zufallsvariablen im Intervall ± 2 und

$\sim 99,7\%$ dieser Zufallsvariablen im Intervall ± 3.

Nach Quadrierung und Aufsummation sollte die Größe χ^2 $(0 \leq \chi^2 < \infty)$ also in etwa in der Größenordnung von N liegen.

5.2 Beispiel.

- Man erzeuge mit einem *Zufallszahlengenerator* (→ Kap. 8) und entsprechenden Methoden N normalverteilte Zufallszahlen x_i,
- quadriere diese Zahlen und summiere sie auf.
- Diese Größe ("χ^2") sollte für $N \gg 1$ in der Größenordnung von N liegen. Diese Methode ist ein schneller Test, ob die Zufallszahlen normalverteilt sind.

(c) Die Wahrscheinlichkeit, dass diese Summe den Wert $[\chi^2, \chi^2 + \mathrm{d}\chi^2]$ annimmt, lässt sich analytisch berechnen:

$$p(\chi^2, N)\,\mathrm{d}\chi^2 = \frac{1}{2^{N/2}\Gamma(N/2)} \mathrm{e}^{-\chi^2/2} \left(\chi^2\right)^{N/2-1} \mathrm{d}\chi^2. \tag{5.9}$$

$p(\chi^2, N)$ ist die "Wahrscheinlichkeitsdichte" der χ^2-Funktion bei N Freiheitsgraden, mit Gammafunktion $\Gamma(x)$,

$$\Gamma(x) = \int_0^\infty t^{x-1} \mathrm{e}^{-t} \mathrm{d}t, \qquad x > 0, \tag{5.10}$$

Für die Gammafunktion gilt $\Gamma(x+1) = x\Gamma(x)$ und $\Gamma(n+1) = n!\ \forall n \in \mathbb{N}_0$.

Der Erwartungswert von χ^2 bei N Freiheitsgraden berechnet sich zu

$$\overline{\chi^2} = N \tag{5.11}$$

(vgl. obige Argumentation) und die Standardabweichung ist

$$\sigma = \sqrt{2N}. \tag{5.12}$$

Man beachte, dass für $N \gg 1$ die χ^2-Verteilung in die Normalverteilung (Gaussverteilung) übergeht. Abb. 5.2 zeigt die Wahrscheinlichkeitsdichte der χ^2-Verteilung für $N = 10$ und $N = 30$ Freiheitsgrade.

Freiheitsgrade und lineare Modelle. *Falls χ^2 minimiert wurde*, sind nicht alle Terme der Summe (5.8) statistisch unabhängig. Die Verteilung (5.9) gilt jedoch nur für statistisch unabhängige Zufallsvariablen!

Wenn das zugrunde liegende Modell allerdings *linear in den Parameteren $a_1, \ldots, a_M$ ist*, lässt sich zeigen, dass die minimierten χ^2 wiederum der Verteilung (5.9) folgen, allerdings aufgrund der M Zwangsbedingungen bzgl.

$$p(\chi^2, f) \qquad \text{mit} \qquad f = N - M$$

Freiheitsgraden.

Lineare Modelle sind dabei durch

$$y = \sum_{k=1}^{M} a_k f_k(x) \qquad ,$$

definiert, wobei f_k vorgegebene Funktionen sind.
Nochmals: Die Linearität bezieht sich hier auf die *Parameter*!

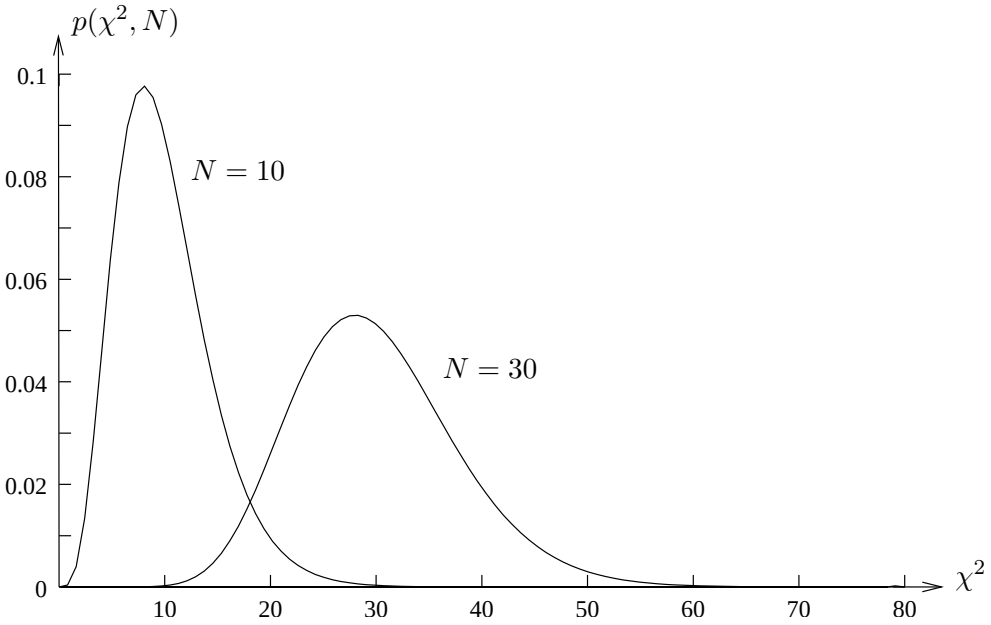


Abbildung 5.2: Wahrscheinlichkeitsdichte der χ^2-Verteilung für $N = 10$ und $N = 30$ Freiheitsgrade. Für $N \gg 1$ geht $p(\chi^2, N)$ in die Normalverteilung über.

5.3 Beispiel (Lineare/Nichtlineare Modelle).

(i) Beispiele für lineare Modelle sind

$$\begin{aligned} y &= a_1 + a_2 x & f_1 &= 1, f_2 = x \\ y &= a_1 + a_2 x + a_3 x^2 & f_3 &= x^2 \\ y &= a_1 + a_2 \cos(x) & f_2 &= \cos(x) \\ y &= a_1 x + a_2 \exp(x) & f_1 &= x, f_2 = \exp(x) \end{aligned}$$

(ii) und für nichtlineare Modelle (vgl. Kap. 5.6)

$$y = x \cdot \sin(a^x), \qquad y = x^{a_1} + x^{a_2}$$

5.4 Beispiel (Lineare Regression bei 10 Messwerten und zwei Parametern). Verteilung der minimierten χ^2.

- Man betrachte die lineare Regression bei 10 Messwerten und zwei Parametern a, b,

  $$y_i = a + bx_i + \text{NOISE (GAUSS)} \qquad (i = 1, \ldots, 10),$$

  Aufgrund des GAUSSschen Rauschens erfüllen die auf diese Weise herbeigeführten künstlichen "Messfehler" die o.g. Einschränkungen "exakt".
- Man führe 10000 solcher "Messreihen" durch und berechne das jeweilige $\chi^2_{\min}$, das sich nach der entsprechenden Minimierung ergibt. Die resultierende Verteilung wird in Abb. 5.3 wiedergegeben; wie man sieht, wird die theoretische χ^2 - Verteilung mit $f = 10 - 2 = 8$ Freiheitsgraden sehr gut wiedergegeben. Auch der berechnete Mittelwert und die Standardabweichung der Verteilung, $\overline{\chi}^2 = 8.07 \pm 4.03$, entspricht sehr gut den theoretischer Werten (Gln. 5.11, 5.12), $\overline{\chi}^2 = 8 \pm 4$.

Beachte: Viele auf den ersten Blick als nichtlinear erscheinende Funktionen lassen sich durch Logarithmieren auf lineare Probleme zurückführen.

5.5 Beispiel.

$$\begin{aligned} y = x^a \quad &\rightarrow \quad \ln y = a \cdot \ln x \qquad (x > 0) \\ y = a_1 x^{a_2} \quad &\rightarrow \quad \ln y = \ln a_1 + a_2 \ln x \end{aligned}$$

Die lineare Regression bzgl. $\ln x$, $\ln y$, (mit $x_i, y_i > 0$) liefert die Parameter a_2 und $\ln a_1$, also a_2 und a_1 selbst.

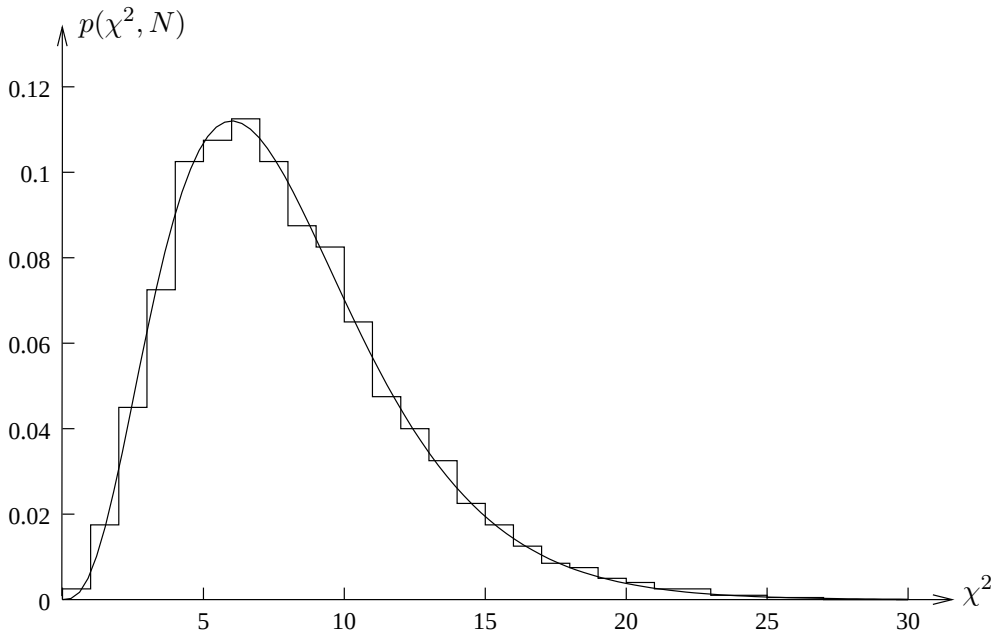


Abbildung 5.3: Zum Beispiel 5.4; Histogramm: "gemessene" Verteilung der minimierten χ^2 aus 10000 Messreihen; Kurve: Wahrscheinlichkeitsdichte $p(\chi^2, f)$ für $f = 8$.

Bei dieser Vorgehensweise gilt es allerdings, eine konistente Fehlerbetrachtung durchzuführen, denn

- eine Ausgleichung bzgl. $\ln y$, $\ln x$ verwendet implizit auch den entsprechenden Fehler bzgl. $\ln y_i \pm \sigma_i$!
- Falls die (linearen) Fehler Δy_i gegeben sind, ist eine Transformation der Fehler notwendig:
- In diesem Fall lassen sich die Fehler durch

$$\sigma_i \approx \ln(y_i \pm \Delta y_i) - \ln y_i$$

  approximieren, d.h.

$$\sigma_i \approx \ln\left[y_i\left(1 \pm \frac{\Delta y_i}{y_i}\right)\right] - \ln y_i = \ln\left(1 \pm \frac{\Delta y_i}{y_i}\right)$$

- Aufgrund der Asymmetrie bzgl. "±" muss man ggf. mitteln. Für kleine relative Fehler ergibt sich

$$\sigma_i \approx \ln\left(1 \pm \frac{\Delta y_i}{y_i}\right) \approx \pm\frac{\Delta y_i}{y_i},$$

  d.h. der lineare Fehler Δy_i des Ausgangsproblemes wird in der "logarithmierten" Ausgleichung durch den entsprechenden relativen Fehler ersetzt.

5.3.2 Die Fitgüte Q

Die obige χ^2-Verteilung (5.9) ergibt sich nur dann, wenn die unterliegenden Annahmen tatsächlich erfüllt sind, d.h. das Modell zutrifft und die Standardabweichungen der Messfehler normalverteilt sind, mit entsprechenden Werten für die jeweiligen σ_i.

Auf dieser Tatsache basiert nun folgende **einfache Idee**. Um zu testen, ob das unterliegende Modell tatsächlich zutrifft und die Fehler richtig angenommen wurden, führt man, wie im vorhergehenden

Beispiel 5.4 vorexerziert, ein große Anzahl von Messreihen durch und vergleicht die erziehlte Verteilung der minimierten χ^2 mit der theoretisch zu erwartenden;
Falls man Abweichungen findet, ist entweder das Modell selbst oder der angenommene Fehler "falsch".

Leider lässt sich diese einfache Idee kaum in die Praxis umsetzen, da das Verfahren entweder zu teuer oder zu zeitaufwendig ist. Folgende Überlegung bringt uns jedoch weiter.

Falls das Modell "wahr" ist, müsste das minimierte χ^2 (im Weiteren mit χ^2_0 bezeichnet) einen "vernünftigen" Wert innerhalb der zufälligen Verteilung *aller möglichen* χ^2-Werte *für dieses Modell* aufweisen. Man erinnere sich, dass das gleiche Modell ja verschiedene Messwerte aufgrund unterschiedlicher Messfehler und damit unterschiedliche χ^2_0 produzieren kann, so wie es in Beispiel 5.4 der Fall war.

Es stellt sich also die Frage, wie groß ist die Wahrscheinlichkeit dafür ist, dass bei der entsprechenden Messreihe *zufällig* ein anderer Wert für χ^2_0 auftreten könnte.

Antwort: Diese Wahrscheinlichkeit lässt sich durch Aufsummieren der "Einzelwahrscheinlichkeiten" $p\,(\chi^2, f)\,\mathrm{d}\chi^2$ bestimmen, d.h.

$$W(\chi^2 \leq \chi^2_0, f) = \int_0^{\chi^2_0} p\,(\chi^2, f)\,\mathrm{d}\chi^2 \tag{5.13}$$

ist die (kumulative) Wahrscheinlichkeit dafür, dass man bei Durchführung des Experimentes einen kleineren Wert als den tatsächlich "gemessenen", χ^2_0, hätte finden können. Einsetzen ergibt

$$W = \frac{1}{2^{f/2}\Gamma(f/2)} \int_0^{\chi^2_0} \mathrm{e}^{-\chi^2/2} (\chi^2)^{f/2-1}\,\mathrm{d}\chi^2$$

und mit der Substitution $\dfrac{\chi^2}{2} = u$, d.h. $\mathrm{d}\chi^2 = 2\mathrm{d}u$ und $\chi^2 = 2u$ folgt

$$\begin{aligned} W &= \frac{1}{2^{f/2}\Gamma(f/2)} \int_0^{\chi^2_0/2} \mathrm{e}^{-u} u^{f/2-1} \cdot 2\mathrm{d}u \cdot 2^{f/2-1} \\ &= \frac{1}{\Gamma(f/2)} \int_0^{\chi^2_0/2} \mathrm{e}^{-u} u^{f/2-1} \mathrm{d}u \\ &= \frac{\gamma(f/2, \chi^2_0/2)}{\Gamma(f/2)} =: P\,(f/2, \chi^2_0/2). \end{aligned} \tag{5.14}$$

$\gamma(a, x)$ ist hierbei die *unvollständige* Gammafunktion

$$\gamma(a, x) = \int_0^x t^{a-1} \mathrm{e}^{-t} \mathrm{d}t, \tag{5.15}$$

im Vergleich zur *vollständigen* Gammafunktion

$$\Gamma(a) = \int_0^\infty t^{a-1} \mathrm{e}^{-t} \mathrm{d}t$$

so dass die oben eingeführte Funktion P im Allgemeinen durch

$$P(a,x) = \frac{\gamma(a,x)}{\Gamma(a)}$$

definiert ist.

Die Wahrscheinlichkeit, dass bei f Freiheitsgraden und einem "wahren" Modell auch andere minimierte χ^2 mit $\chi^2 \leq \chi_0^2$ hätten auftreten können, ist also durch die P-Funktion mit den Argumenten $f/2$ und $\chi_0^2/2$ gegeben!

Üblicherweise arbeitet man nicht mit dieser Funktion, sondern mit der dazu komplementären sog. *"Fitgüte"* Q,

$$Q(\chi_0^2, f) = 1 - P\left(\frac{f}{2}, \frac{\chi_0^2}{2}\right). \tag{5.16}$$

Man beachte wiederum die Argumente!

Da die Wahrscheinlichkeit, dass *irgendein* Wert $0 \leq \chi^2 < \infty$ auftreten kann, gleich "1" ist (wie es natürlich auch sein sollte; dies ergibt sich aus $P\,(f/2, \infty) = 1$), gibt die "Fitgüte" Q die Wahrscheinlichkeit an, dass χ^2 *nicht* im Bereich $[0, \chi_0^2]$ liegt, d.h. im Bereich $\chi_0^2 \ldots \infty$ liegen muss!
Die Fitgüte Q gibt damit die Wahrscheinlichkeit an, dass andere $\chi^2 \geq \chi_0^2$ hätten auftreten können.

Falls man also einen Wert von $Q \approx 1$ berechnet, ergibt sich der Verdacht, dass χ_0^2 viel zu klein, mit anderen Worten der Fit "zu gut" ist.

Falls andererseits $Q \approx 0$, ist kein größeres χ^2 möglich. Das sich durch die Ausgleichung ergebende χ_0^2 ist also viel zu groß, und der Fit ist inakzeptabel. (Auf weitere Implikationen werden wir im Folgenden eingehen.)

> Die Berechnung der Fitgüte erlaubt es uns festzustellen, ob das Modell selbst und/ oder die angenommenen Fehler *im Prinzip* mit der Messung kompatibel sind.

Basierend auf diesen Ausführungen stellen wir nun das "Kochrezept" zur Durchführung einer linearen Regression zusammen.

Erste Vorgehensweise (empfohlen)

(i) Man bestimme die Parameter des (gegebenen) Modelles, $a_1, \ldots, a_M$ durch Minimierung von χ^2, d.h.

$$0 = \sum_{i=1}^{N} \left(\frac{(y_i - y(x_i, a_1, \ldots, a_M))}{\sigma_i^2} \cdot \frac{\partial y(x_i, a_1, \ldots, a_M)}{\partial a_k} \right), \qquad k = 1, \ldots, M$$

d.h. löse die M linearen Gleichungen für $a_1, \ldots, a_M$.

(ii) Aus diesen Parametern ergibt sich sofort das minimierte χ_0^2.

(iii) Man überprüfe nun die Wahrscheinlichkeit dieses Wertes, d.h. die "Güte des Fits", durch Berechnung von

$$Q(\chi_0^2, f) = 1 - P(\frac{f}{2}, \frac{\chi_0^2}{2})$$

In Abhängigkeit vom resultierenden Wert lassen sich dann folgende Aussagen treffen

- Falls $0.05 \leq Q \leq 0.95$, ist der Fit (im Prinzip) in Ordnung.

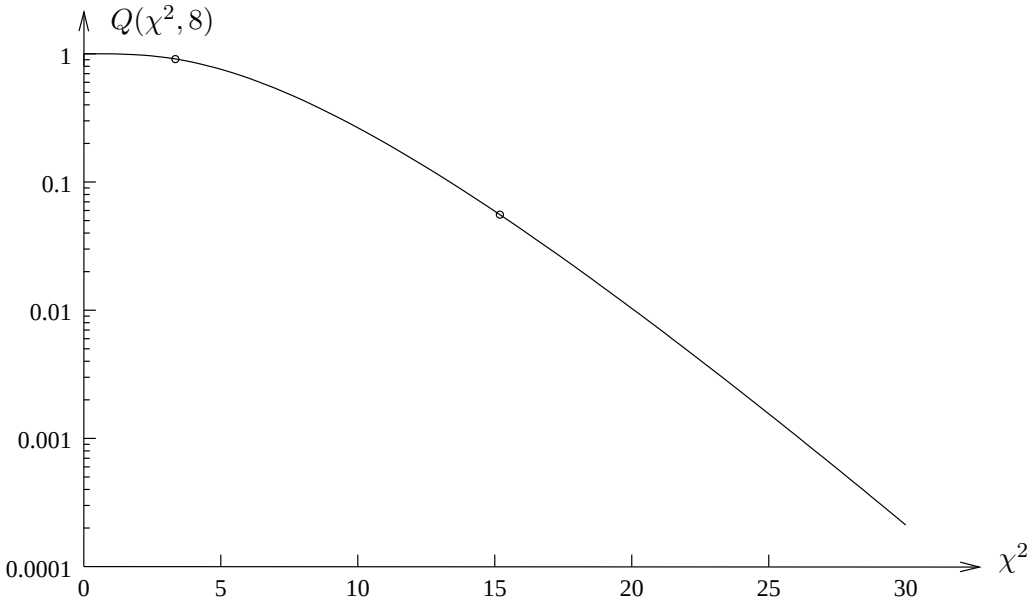


Abbildung 5.4: $Q(\chi^2, 8)$. Für $f = 8$ Freiheitsgrade sind alle χ_0^2 im (markierten) Bereich $2.73 \ldots 15.5$ akzeptabel ($0.05 \leq Q \leq 0.95$, der Wert $Q = 10^{-3}$ entspricht einem $\chi_0^2 \approx 26$).

- Falls $Q \gtrsim 10^{-3}$, *kann* der Fit in Ordnung sein, man sollte jedoch eine zusätzliche Messreihe durchführen.
- Falls $Q \lesssim 10^{-3}$, ist χ_0^2 zu groß! Entweder das Modell ist "falsch" oder bestimmte σ_i wurden zu klein angenommen oder die Messfehler sind nicht normalverteilt.
- Falls Q sehr nahe bei 1 liegt, ist χ_0^2 zu klein. Entweder wurden die Messfehler zu groß angenommen, oder es handelt sich um *Datenfälschung*!

Die Abbildungen 5.4 und 5.5 veranschaulichen den Verlauf der Fitgüte Q also Funktion von χ^2 für $f = 8$ bzw. $f = 98$ Freiheitsgrade (entsprechend einer linearen Regression mit z.B. zwei Parametern und 10 bzw. 100 Datenpunkten[2]) und zeigen den Bereich der "erlaubten" χ^2- Werte bezüglich $0.05 \leq Q \leq 0.95$.

Falls keine Aussagen über die generellen bzw. individuellen Messfehler möglich sind, muss unser "Kochrezept" modifiziert werden.

Zweite Vorgehensweise (falls keine Fehleraussagen möglich)

(i) Zunächst nehmen wir an, dass alle Fehler gleich sind, $\sigma_i =: \sigma \quad \forall i$. Dann folgen aus Minimierung von

$$\sum_{i=1}^{N} \left(y_i - y(x_i, a_1, \ldots, a_M)\right)^2 = \text{min!}$$

($\rightarrow$ Methode der kleinsten Quadrate) die gesuchten Parameter $a_1, \ldots, a_M$

(ii) *Wir fordern jetzt, dass der Fit in Ordnung ist*, und dass damit insbesondere das minimierte χ_0^2 seinem theoretischen Erwartungswert $\overline{\chi^2}(f)$ (vgl. 5.11) entspricht!

$$\chi_0^2 = \sum_{i=1}^{N} \frac{\left(y_i - y(x_i, a_1, \ldots, a_M)\right)^2}{\sigma^2} \quad \stackrel{!}{=} \quad \overline{\chi^2}(f) = N - M.$$

[2] natürlich auch für alle anderen Kombinationen $f = N - M = 8$ bzw. 98

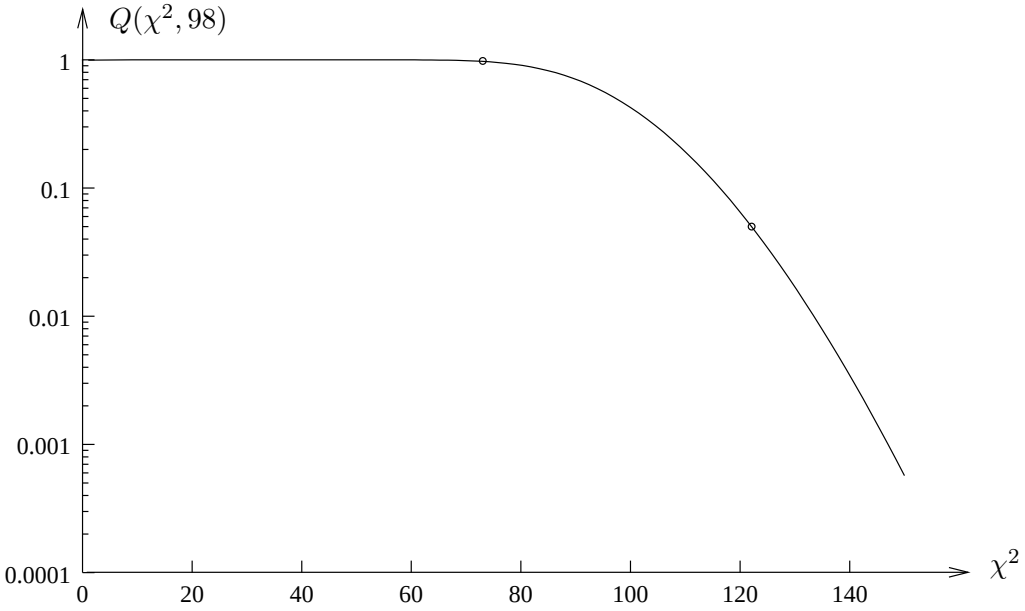


Abbildung 5.5: $Q(\chi^2, 98)$. Für $f = 98$ Freiheitsgrade sind alle χ_0^2 im markierten Bereich $76.1 \ldots 122.1$ akzeptabel ($0.05 \leq Q \leq 0.95$, der Wert $Q = 10^{-3}$ entspricht einem $\chi_0^2 \approx 145$).

Damit ergibt sich sofort die (bis dato unbekannte) Standardabweichung σ zu

$$\sigma^2 = \frac{\sum_{i=1}^{N} (y_i - y(x_i, a_1, \ldots, a_M))^2}{N - M} \tag{5.17}$$

Nochmals: Bei dieser zweiten Methode wird explizit (und ohne dass wir dafür normalerweise einen Anhaltspunkt hätten) angenommen, dass alle Messfehler gleich sind, $\sigma_i = \sigma$, und dass der Fit in Ordnung ist! Beide Annahmen sind wesentlicher Bestandteil der Methode der kleinsten Quadrate.

Als letzte der ganz zu Anfang (Kap. 5.1) gestellten Fragen bleibt nun zu klären, wie wir die *Fehler in den abgeleiteten Parametern* $a_1, \ldots, a_M$ abschätzen können. Prinzipiell (die tatsächlichen Gleichungen werden später angegeben) gehen wir dabei davon aus, dass *jeder* Datenpunkt zum Fehler in den Parametern beiträgt, und aus der üblichen *Fehlerfortpflanzung*[3] folgt dann

$$\sigma_{a_k}^2 = \sum_{i=1}^{N} \sigma_i^2 \cdot \left(\frac{\partial a_k}{\partial y_i} \right)^2 . \tag{5.18}$$

Identifikation der Standardabweichungen mit den gesuchten bzw. gegebenen Fehlern ergibt damit

$$\Delta a_k^2 = \sum_{i=1}^{N} \Delta y_i^2 \cdot \left(\frac{\partial a_k}{\partial y_i} \right)^2 .$$

In den nächsten (beiden) Kapiteln werden wir nun alle bislang abgeleiteten Ergebnisse auf die Regression für *lineare* Modelle anwenden, zunächst für den Fall $M = 2$, d.h. den Fit an eine Gerade, und dann für den Fall beliebiger M.

5.4 Lineare Regression: Fit an eine Gerade

In diesem (schon zuvor) betrachteten Fall lautet das Modell, das es zu fitten gilt,

$$y(x, a, b) = a + bx,$$

[3] auch hier geht wieder die Annahme von *normalverteilten* Fehlern ein

wobei hier

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - (a + bx_i)}{\sigma_i} \right)^2$$

zu minimieren ist. Im Vergleich zur Methode der kleinsten Quadrate (Gln. 5.1 bis 5.3) ergeben sich nur marginale Unterschiede, die aus dem Faktor $\left(\frac{1}{\sigma_i}\right)^2$, d.h. der Angabe der individuellen Messfehler resultieren. Insbesondere verändern sich bezüglich o.g. Gleichungen die Werte

$$N \to \sum \frac{1}{\sigma_i^2} =: S, \qquad S_x \to \sum \frac{x_i}{\sigma_i^2}, \qquad S_y \to \sum \frac{y_i}{\sigma_i^2} \qquad \text{etc.}$$

Mit $\Delta = S \cdot S_{xx} - S_x^2$ folgt dann

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} S_{xx}S_y & - & S_x S_{xy} \\ S S_{xy} & - & S_x S_y \end{pmatrix} \tag{5.19}$$

die zu (5.3) analoge Lösung. (Der Unterschied ist nur durch die jeweiligen Messfehler σ_i^2 gegeben, die in *allen* Größen im Nenner auftreten.)

Falls alle Fehler gleich sind, $\sigma_i = \sigma$, kürzt sich der Faktor $\frac{1}{\sigma^2}$ aus den Gleichungen heraus, und Gleichung (5.19) ist identisch mit (5.3).

Merke: Falls alle Messungen mit dem gleichen Fehler behaftet sind (unabhängig von seiner aktuellen Größe), $\sigma_i = \sigma$, ergibt die χ^2 Minimierung und die Methode der kleinsten Quadrate identische Ergebnisse *für die Parameter des Modelles*. Die Fehler der Parameter und die Fitgüte hängen allerdings von dem tatsächlichen Wert σ ab (s.u.)

5.4.1 Fehler der abgeleiteten Parameter

Die Fehler der abgeleiteten Parameter a, b ergeben sich aus der Fehlerfortpflanzung (5.18)

$$\sigma_a^2 = \sum_i \sigma_i^2 \cdot \left(\frac{\partial a}{\partial y_i} \right)^2, \qquad \sigma_b^2 = \sum_i \sigma_i^2 \cdot \left(\frac{\partial b}{\partial y_i} \right)^2.$$

Unter Berücksichtigung, dass Δ unabhängig von y_i ist,

$$\frac{\partial a}{\partial y_i} = \frac{S_{xx} - x_i S_x}{\Delta \cdot \sigma_i^2}, \qquad \frac{\partial b}{\partial y_i} = \frac{x_i S - S_x}{\Delta \cdot \sigma_i^2}$$

folgt

$$\begin{aligned} \sum \sigma_i^2 \cdot \left(\frac{\partial a}{\partial y_i} \right)^2 &= \sum \sigma_i^2 \cdot \left(\frac{S_{xx} - x_i S_x}{\Delta \cdot \sigma_i^2} \right)^2 = \\ &= \frac{1}{\Delta^2} \left(S_{xx}^2 S - 2 S_x^2 S_{xx} + S_x^2 S_{xx} \right) \\ &= \frac{1}{\Delta^2} \cdot S_{xx} \underbrace{\left(S S_{xx} - S_x^2 \right)}_{\Delta} = \frac{S_{xx}}{\Delta}. \end{aligned}$$

Der Fehler von b ergibt sich in analoger Weise, und wir erhalten als Resultat

$$\sigma_a^2 = \frac{S_{xx}}{\Delta}, \qquad \sigma_b^2 = \frac{S}{\Delta} \tag{5.20}$$

Falls die Messfehler gleich *und bekannt* sind, $\sigma_i = \sigma$, skalieren damit die Fehler der abgeleiteten Parameter mit ($S \propto \frac{1}{\sigma^2}$, $\Delta \propto \frac{1}{\sigma^4}$) $\sigma_a^2, \sigma_b^2 \propto \sigma^2$, d.h.

$$\sigma_a, \sigma_b \propto \sigma. \tag{5.21}$$

5.4.2 Unbekannte Messfehler

Falls die individuellen (oder globalen) Messfehler andererseits *unbekannt* sind, wenden wir das "zweite" Verfahren an:

(i) Man setze im gesamten obigen Algorithmus explizit $\sigma_i = 1$ (zur Erinnerung: $S(\sigma_i = 1) = N$)

(ii) und berechne a, b mit (5.19).

(iii) Danach schätze man den globalen Fehler durch

$$\sigma^2 = \frac{\sum (y_i - (a + bx_i))^2}{N - M}$$

ab, vgl. (5.17).

(iv) Die Fehler der Parameter resultieren dann aufgrund obiger Skalierungen aus einer einfachen Muliplikation der mit $\sigma_i = 1$ erzielten Fehler (5.20) mit dem abgeschätzen Fehler aus (iii),
$\sigma_a^2 = \sigma_a^2(\sigma_i = 1) \cdot \sigma^2$
$\sigma_b^2 = \sigma_b^2(\sigma_i = 1) \cdot \sigma^2$.

Die unterschiedlichen Vorgehensweisen und die daraus resultierenden unterschiedlichen Ergebnisse wollen wir nun anhand des Datensatzes aus Beispiel 5.1 verdeutlichen.

5.6 Beispiel (Fit an Gerade mit unterschiedlichen Voraussetzungen).

(1) Zunächst soll keine Angabe über die Messfehler vorliegen, d.h. das "zweite Verfahren" (entsprechend der Methode der kleinsten Quadrate) wird angewandt. Für $N = 7$ (alle Punkte) hatten wir schon die Parameter der Ausgleichung berechnet,

$$a = 0.114286, \qquad b = 0.889286$$

Unter der Annahme, dass der Fit in Ordnung ist, und mit $N - M = 5$ Freiheitsgraden ergibt sich

$$\sum (y - (a + bx_i))^2 = 8.2911$$

und wir finden einen globalen Fehler von

$$\sigma^2 = \frac{8.2911}{5}, \qquad \text{d.h.} \quad \sigma = 1.288$$

Mit diesem Wert hätten wir also einen akzeptablen Fit mit einem seinem Erwartungswert entsprechenden $\chi_0^2 = \overline{\chi^2}$ erzielt. Die Fehler der Parameter werden dann durch

$$\sigma_a(\sigma_i = 1) = 0.8451 \qquad \sigma_b(\sigma_i = 1) = 0.1889$$
$$\Rightarrow \sigma_a(\sigma) = 1.0885 \qquad \sigma_b(\sigma) = 0.2433$$

abgeschätzt, insgesamt ergibt sich also

$$a = 0.114286 \pm 1.0885 \leftarrow$$

(damit ist der Achsenabschnitt *sehr* unsicher) und eine etwas besser definierte Steigung

$$b = 0.889286 \pm 0.2433$$

Die großen Unsicherheiten resultieren hier natürlich aus der Tatsache, dass es einen "Ausreisser" gibt (Messwert "5"), wir aber trotzdem davon ausgegangen sind, dass der Fit in Ordnung ist!

(2) Wir nehmen nun an, dass die Messfehler wiederum gleich, aber bekannt sind, und zwar

$$\sigma_i = 0.2 \quad \forall i.$$

Mit Hilfe der Skalierung ergeben sich

$$a = 0.114286 \pm 0.169 \qquad b = 0.889286 \pm 0.0378,$$

im Vergleich zu vorher anscheinend recht gut definierte Werte. Betrachtet man aber das zugehörige

$$\chi_0^2 = 207.27$$

so findet man einen für $f = 5$ Freiheitsgrade sehr großen Wert, der einer Fitgüte

$$Q \approx 10^{-42}$$

entspricht. Wir schließen also, dass entweder das Modell falsch ist oder die Fehler als zu klein angenommen wurden!

(3) Wir betrachten jetzt einen ähnlichen Fall wie zuvor, $\sigma_i = 0.2, i \neq 5$. Der Messfehler des "Ausreissers" sei jetzt aber erheblich größer als bei den anderen Datenpunkten, $\sigma_5 = 2$. Damit finden wir (mit Gl. 5.19)

$$a = 0.114285 \pm 0.16903 \qquad b = 0.99927 \pm 0.0386.$$

Im Vergleich zu (2) ändern sich die Parameter, während ihre Fehler ähnlich (gering) sind. Das zugehörige

$$\chi_0^2 = 10.1323$$

scheint jetzt aber in Ordnung zu sein, und Berechnung von

$$Q = 0.0715 > 0.05$$

bestätigt diesen Eindruck. Modell und Fehler sind kompatibel mit den zur Verfügung stehenden Daten!

5.4.3 Regression zwischen zwei Messgrößen

Unter bestimmten Umständen wird eine Regression zwischen *zwei* Messgrößen angestrebt, d.h. auch die Werte der Abszisse x_i seien mit Messfehlern behaftet! Wie üblich gewinnt man die Parameter der Regression durch χ^2-Minimierung, allerdings jetzt mit

$$\chi^2(a,b) = \sum \frac{[y_i - (a + bx_i)]^2}{\sigma_{\text{tot},i}^2} \tag{5.22}$$

wobei $\sigma_{\text{tot},i}$ die *insgesamt* resultierende Standardabweichung ($\hat{=}$ totaler Messfehler) ist, die sich aus den jeweiligen Komponenten in "x"- und "y"- Richtung ergibt.

Um diesen Fehler zu berechen, betrachten wir ein bestimmtes Residuum bzgl. der zu modellierenden Geraden, l_i (vgl. Abb. 5.6),

$$l_i = y_i - (a + bx_i)$$

Der in die Ausgleichung eingehende Fehler ist derjenige dieses Residuums, und ist im bislang diskutierten"Standardfall" nur durch den Fehler Δy_i gegeben.

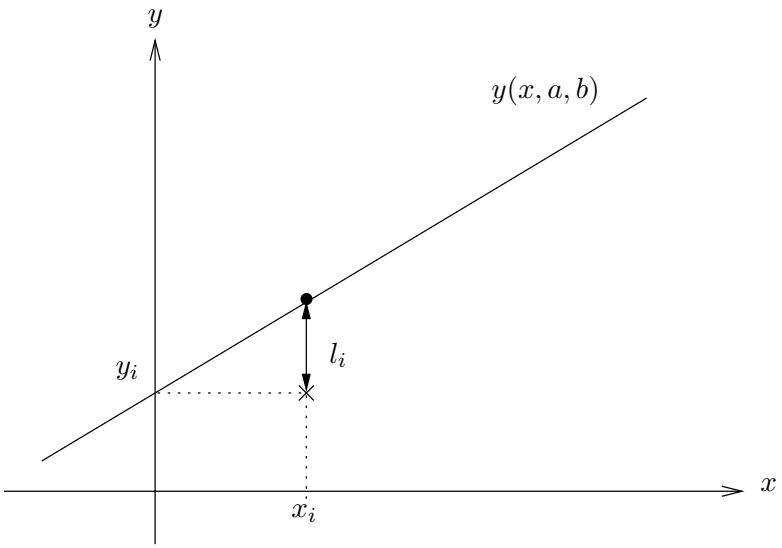


Abbildung 5.6: Zur Regression zwischen zwei Messgrößen.

Wir gehen im Weiteren davon aus, dass die Messfehler in x- und y-Richtung *unabhängig sind*. Dann finden wir aus der üblichen Fehlerfortpflanzung, dass der Fehler des Residuums, Δl_i, über

$$\Delta l_i^2 = \left[\left(\frac{\partial l_i}{\partial y_i} \right)^2 \Delta y_i^2 + \left(\frac{\partial l_i}{\partial x_i} \right)^2 \Delta x_i^2 \right] = \Delta y_i^2 + b^2 \Delta x_i^2$$

mit den entsprechenden Fehlern Δx_i und Δy_i zusammenhängt.

Wir finden also für den in die Ausgleichung eingehenden totalen Messfehler

$$\begin{aligned}\sigma_{\text{tot},i}^2 &= \sigma_{y_i}^2 + b^2\sigma_{x_i}^2 \\ &= \sigma_{y_i}^2\left(1 + b^2 \cdot \frac{\sigma_{x_i}^2}{\sigma_{y_i}^2}\right),\end{aligned} \tag{5.23}$$

und es ist ersichtlich, dass sich der Messfehler in x-Richtung um so drastischer auswirkt, je steiler die Relation ist.

Bei der χ^2-Minimierung ergibt sich nun allerdings folgendes *Problem.* Die Gleichung für $\dfrac{\partial\chi^2}{\partial b}$ wird nichtlinear, da b^2 im Nenner von χ^2 steht. Eine mögliche Lösung wird z.B. in "Numerical Recipes", Kapitel 15.3 angegeben. Ebenso lässt sich das Problem *iterativ* wie im Rahmen der "nichtlinearen Probleme" (Kap. 5.6) bearbeiten. Für die im totalen Fehler (5.23) auftretende Steigung b verwendet man dabei den aus der vorhergehenden Iteration erzielten Wert.

5.5 Der allgemeine lineare Fall - Normalgleichungen

Bisher hatten wir uns ausschließlich mit der Ausgleichung bezüglich zwei-parametriger linearer Modelle beschäftigt, dem Fit an eine Gerade. Oftmals werden auch kompliziertere Modelle benötigt, deren Parameter durch Regression bestimmt werden sollen. Auch im Folgenden seien diese Modelle *linear in den Parametern* a_k,

$$y(x) = \sum_{k=1}^{M} a_k X_k(x), \tag{5.24}$$

wenn $X_k(x)$ beliebige, an allen Messpunkten x_i berechenbare Funktionen sind, die sog. *Basisfunktionen.*

Beispiele für solche Basisfunktionen sind

$$\begin{aligned} X_k(x) &= x^{k-1} \quad \text{Polynome} \\ X_k(x) &= \cos(kx) \quad \text{trigonometrische Funktionen} \end{aligned}$$

etc. Wiederum gilt es, das entsprechende χ^2,

$$\chi^2 = \sum_{i=1}^{N}\left(\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i}\right)^2 \tag{5.25}$$

zu minimieren (ggf. mit $\sigma_i = 1\,\forall i$), wobei mehr Messpunkte als Parameter ($N > M$) vorhanden sein sollen (müssen!).

Wir betrachten zunächst folgendes lineares Gleichungssystem bzgl. aller Messpunkte, das sog. "Fehlergleichungssystem"

$$\begin{aligned} a_1\frac{X_1(x_1)}{\sigma_1} + a_2\frac{X_2(x_1)}{\sigma_1} + \cdots + a_M\frac{X_M(x_1)}{\sigma_1} - \frac{y_1}{\sigma_1} &= r_1 \\ a_1\frac{X_1(x_2)}{\sigma_2} + a_2\frac{X_2(x_2)}{\sigma_2} + \cdots + a_M\frac{X_M(x_2)}{\sigma_2} - \frac{y_2}{\sigma_2} &= r_2 \\ &\vdots \\ a_1\frac{X_1(x_N)}{\sigma_N} + a_2\frac{X_2(x_N)}{\sigma_N} + \cdots + a_M\frac{X_M(x_N)}{\sigma_N} - \frac{y_N}{\sigma_N} &= r_N \end{aligned} \tag{5.26}$$

$r_i, i = 1, \ldots, N$ sind die Residuen der Ausgleichung, in Einheiten der jeweiligen Messfehler σ_i. Dieses Gleichungssystem, dass sich auch kürzer als

$$\sum_{k=1}^{M} C_{ik} a_k + d_i = r_i$$

bzw.

$$C\boldsymbol{a} + \boldsymbol{d} = \boldsymbol{r} \tag{5.27}$$

formulieren lässt, mit $C \in \mathbb{R}^{N\times M}$, $\boldsymbol{d}, \boldsymbol{r} \in \mathbb{R}^N$ und $\boldsymbol{a} \in \mathbb{R}^M$, hat für $N > M$ mehr Zeilen als Spalten und ist damit ein überbestimmtes Gleichungssystem. Wir nehmen nun an, dass die Spalten dieses Systems linear unabhängig seien, was immer dann der Fall ist, wenn die entsprechenden Basisfunktionen linear unabhängig sind (ein Modell mit linear abhängigen Funktionen wäre ziemlicher Unfug).

Vergleicht man (5.25) mit (5.26) bzw. (5.27), so ist sofort ersichtlich, dass die χ^2-Minimierung der Minimierung der euklidischen Norm des Residuenvektors, $\sum r_i^2$, entspricht, wie wir es auch schon für den Fit an eine Gerade gefunden hatten. Aus dem Vergleich ergeben sich die Zusammenhänge

$$C_{ik} = \frac{X_k(x_i)}{\sigma_i}, \qquad d_i = -\frac{y_i}{\sigma_i}, \qquad \chi^2 = \|\boldsymbol{r}\|_2^2; \tag{5.28}$$

Die Matrix C wird auch *"Designmatrix"* genannt. Quadrieren von $\boldsymbol{r}$ (bzw. Berechnung von χ^2) ergibt dann

$$\begin{aligned}
\|\boldsymbol{r}\|^2 &= (C\boldsymbol{a} + \boldsymbol{d})^\intercal (C\boldsymbol{a} + \boldsymbol{d}) \\
&= \boldsymbol{a}^\intercal C^\intercal C\boldsymbol{a} + \underbrace{\boldsymbol{a}^\intercal (C^\intercal \boldsymbol{d})}_{(C^\intercal \boldsymbol{d})^\intercal \boldsymbol{a}} + \underbrace{(\boldsymbol{d}^\intercal C)\boldsymbol{a}}_{(C^\intercal \boldsymbol{d})^\intercal \boldsymbol{a}} + \boldsymbol{d}^\intercal \boldsymbol{d} \\
&= \boldsymbol{a}^\intercal C^\intercal C\boldsymbol{a} + 2 \cdot (C^\intercal \boldsymbol{d})^\intercal \boldsymbol{a} + \boldsymbol{d}^\intercal \boldsymbol{d}.
\end{aligned}$$

Definiert man nun eine Matrix $A = C^\intercal C \in \mathbb{R}^{M\times M}$ und einen Vektor $\boldsymbol{b} = C^\intercal \boldsymbol{d} \in \mathbb{R}^M$, dann gilt für eine mit A gebildete quadratische Form $Q(\boldsymbol{a})$,

(i) $Q(\boldsymbol{a}) = \boldsymbol{a}^\intercal A\boldsymbol{a} = \boldsymbol{a}^\intercal C^\intercal C\boldsymbol{a} = (C\boldsymbol{a})^\intercal (C\boldsymbol{a}) = \|C\boldsymbol{a}\|^2 \geq 0 \quad \forall \boldsymbol{a}$

(ii) $Q(\boldsymbol{a}) = 0 \Leftrightarrow (C\boldsymbol{a}) = 0 \Leftrightarrow \boldsymbol{a} = 0$, da die Spalten von C linear unabhängig sind.

Also ist A positiv definit symmetrisch! (Vgl. Kap. 3.10)

Wir definieren jetzt das Funktional $F(\boldsymbol{a})$, welches identisch zu χ^2 ist,

$$F(\boldsymbol{a}) = \boldsymbol{r}^\intercal \boldsymbol{r} = \boldsymbol{a}^\intercal A\boldsymbol{a} + 2\boldsymbol{b}^\intercal \boldsymbol{a} + \boldsymbol{d}^\intercal \boldsymbol{d}$$

so dass die χ^2-Minimierung durch

$$\boldsymbol{\nabla} F(\boldsymbol{a}) = \boldsymbol{0}, \tag{5.29a}$$

oder in Komponentenschreibweise:

$$\frac{\partial F(\boldsymbol{a})}{\partial a_i} = 2\sum_{k=1}^{M} A_{ik} a_k + 2b_i = 0, \quad i = 1, \ldots, M, \tag{5.29b}$$

also

$$A \cdot \boldsymbol{a} + \boldsymbol{b} = \boldsymbol{0} \tag{5.29c}$$

gegeben ist. Das letzte Gleichungssystem nennt man auch das *"Normalgleichungssystem"*.

Zusammenfassend lässt sich also feststellen, dass die χ^2-Minimierung für lineare Modelle der Minimierung der euklidischen Norm des Residuenvektors entspricht, und äquivalent zur Lösung des Normalgleichungssystems ist. Mit der alternativen Schreibweise

$$A \cdot \boldsymbol{a} = \boldsymbol{\beta}, \qquad \text{wobei} \qquad \boldsymbol{\beta} = C^\intercal(-\boldsymbol{d}) = C^\intercal \begin{pmatrix} y_1/\sigma_1 \\ \vdots \\ y_N/\sigma_N \end{pmatrix} \tag{5.30}$$

ergibt sich folgender einfache

Algorithmus zur χ^2-Minimierung für lineare Modelle.

(i) Man berechne die Designmatrix C mit Elementen $C_{ik} = \dfrac{X_k(x_i)}{\sigma_i}$.

(ii) Daraus folgt die Matrix $A = C^\intercal C$ mit Elementen $a_{kj} = \displaystyle\sum_{i=1}^{N} \frac{X_k(x_i)X_j(x_i)}{\sigma_i^2}$.

(iii) Schließlich wird noch der Vektor $\boldsymbol{\beta} = C^\intercal(-\boldsymbol{d})$, mit Elementen $\beta_k = \displaystyle\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2}$ berechnet.

(iv) Nun zerlegt man die positiv definite symmetrische Matrix A mit dem CHOLESKY-Verfahren (vgl. Kap 3.10), $A = LL^\intercal$

(v) und löst das Normalgleichungssystem für den gesuchten Satz der Parameter, $\boldsymbol{a}$,
$A \cdot \boldsymbol{a} = \boldsymbol{\beta}$, d.h. $LL^\intercal \cdot \boldsymbol{a} = \boldsymbol{\beta}$
durch Vorwärts- und Rückwärtseinsetzen,
$L \cdot \boldsymbol{z} = \boldsymbol{\beta}$ und $L^\intercal \cdot \boldsymbol{a} = \boldsymbol{z}$.

(vi) Das erzielte (minimierte) $\chi_0^2 = \|\boldsymbol{r}\|^2$ ergibt sich über
$\boldsymbol{r} = C \cdot \boldsymbol{a} + \boldsymbol{d}$,
wenn die Komponenten von $\boldsymbol{d}$ durch $d_i = -\dfrac{y_i}{\sigma_i}$ gegeben sind,

(vii) und die Güte des Fits kann dann aus dem entsprechenden Wert für $Q\left(\chi_0^2, N - M\right)$ ermittelt werden.

Schließlich bleiben noch die Fehler der Parameter a_k zu berechnen.

5.5.1 Fehler der abgeleiteten Parameter

Aus der allgemeinen Gleichung für die entsprechende Fehlerfortpflanzung (vgl. 5.18) hatten wir

$$\sigma^2(a_j) = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial a_j}{\partial y_i}\right)^2$$

Mit $A\boldsymbol{a} = \boldsymbol{\beta}$, d.h. $\boldsymbol{a} = A^{-1}\boldsymbol{\beta}$ definieren wir die Matrix $B = A^{-1}$ als Inverse von A,

$$B_{ik} = \left(A^{-1}\right)_{ik}.$$

Daraus folgt für einen Parameter a_j

$$a_j = \sum_{k=1}^{M} \left(A^{-1}\right)_{jk} \beta_k = \sum_{k=1}^{M} B_{jk} \left(\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2}\right),$$

und die partielle Ableitung nach y_i ergibt sich zu

$$\frac{\partial a_j}{\partial y_i} = \sum_{k=1}^{M} B_{jk} \frac{X_k(x_i)}{\sigma_i^2},$$

da A und B unabhängig von y_i sind. Insgesamt finden wir also

$$\begin{aligned}
\sigma^2(a_j) &= \sum_{i=1}^{N} \sigma_i^2 \left(\sum_{k=1}^{M} B_{jk} \frac{X_k(x_i)}{\sigma_i^2} \right)^2 \\
&= \sum_{i=1}^{N} \sigma_i^2 \left(\sum_{k=1}^{M} B_{jk} \frac{X_k(x_i)}{\sigma_i^2} \right) \left(\sum_{l=1}^{M} B_{jl} \frac{X_l(x_i)}{\sigma_i^2} \right) \\
&= \sum_{k=1}^{M} \sum_{l=1}^{M} B_{jk} B_{jl} \underbrace{\sum_{i=1}^{N} \frac{X_k(x_i) X_l(x_i)}{\sigma_i^2}}_{A_{kl}}.
\end{aligned}$$

Mit $\sum_k B_{jk} A_{kl} = \delta_{jl}$ (wobei δ das KRONECKER-Symbol ist), finden wir einen erstaunlich einfachen Ausdruck für $\sigma^2(a_j)$, nämlich

$$\sigma^2(a_j) = \sum_{l=1}^{M} B_{jl} \delta_{jl} = B_{jj}$$

Die Varianz von a_j, d.h. das gesuchte Quadrat der Standardabweichung, ist nichts anderes als das Diagonalelement der Inversen von A! Da wir $A = LL^\intercal$ schon zerlegt hatten, lässt sich diese Inverse (schnell) berechnen und wir haben alle das Problem kennzeichnenden Größen gefunden.

5.5.2 Unbekannte Messfehler

Falls die Messfehler σ_i unbekannt sind, wenden wir das "zweite" Verfahren wie früher an.

- Mittels Normalgleichungen (jetzt jedoch mit $\sigma_i = 1 \,\forall i$) berechnen wir in gewohnter Weise die Parameter $a_1, \ldots, a_M$.
- Daraus ergibt sich das minimierte $\chi_0^2 = \|\boldsymbol{r}\|^2$,
- und unter Annahme eines perfekten Fits folgt für den globalen Fehler $\sigma^2 = \frac{\chi_0^2}{N-M}$
- Schließlich schätzen wir die Fehler der Parameter mit $\sigma_{a_j} = \sqrt{\frac{\chi_0^2}{N-M}} \cdot B_{jj}$ unter Verwendung von $B_{jj} = \left[A^{-1}(\sigma_i = 1)\right]_{jj}$ ab.

5.7 Beispiel (χ^2-Minimierung mit Normalengleichungen). Das folgende Beispiel ist in wesentlichen Teilen dem Lehrbuch "Numerische Mathematik" von H.R. Schwarz entnommen (Seite 344).

"Zur analytischen Beschreibung der Kennlinie eines nichtlinearen Übertragungselementes y=f(x) sind für exakte Eingangsgrößen x_i die Ausgangsgrößen y_i beobachtet worden.

x	0.2	0.5	1.0	1.5	2.0	3.0
y	0.3	0.5	0.8	1.0	1.2	1.3

Das Übertragungselement verhält sich für kleine x linear, und die Kennlinie besitzt für große x eine horizontale Asymptote. Um diesem Verhalten Rechnung zu tragen, soll für $f(x)$ der Ansatz

$$f(x) = \alpha_1 \frac{x}{1+x} + \alpha_2(1 - \mathrm{e}^{-x})$$

mit den beiden Parametern α_1 und α_2 verwendet werden."

Messfehler seien nicht bekannt, d.h. es wird $\sigma_i = 1, i = 1, \ldots, 6$ gesetzt. Bei sechsstelliger Rechnung lauten die Designmatrix C und der Vektor $\boldsymbol{d}$

$$C = \begin{pmatrix} 0.166667 & 0.181269 \\ 0.333333 & 0.393469 \\ 0.500000 & 0.632121 \\ 0.600000 & 0.776870 \\ 0.666667 & 0.844665 \\ 0.750000 & 0.950213 \end{pmatrix}, \qquad \boldsymbol{d} = \begin{pmatrix} -0.3 \\ -0.5 \\ -0.8 \\ -1.0 \\ -1.2 \\ -1.3 \end{pmatrix},$$

wobei die erste Spalte von C der Basisfunktion $X_1(x_i) = x_i/(1 + x_i)$ und die zweite Spalte $X_2(x_i) = 1 - \exp(-x_i)$ entspricht. Die Koeffizienten des Normalgleichungssystems $A\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{0}$ lauten dann

$$A = \begin{pmatrix} 1.75583 & 2.32266 \\ 2.23266 & 2.84123 \end{pmatrix}, \qquad \boldsymbol{b} = -\boldsymbol{\beta} = \begin{pmatrix} -2.99167 \\ -3.80656 \end{pmatrix},$$

was nach CHOLESKY-Zerlegung zur Linksdreiecksmatrix L und der Lösung $\boldsymbol{a}$ für die gesuchten Parameter führt:

$$L = \begin{pmatrix} 1.32508 & \\ 1.68492 & 0.0487852 \end{pmatrix}, \qquad \boldsymbol{a} = \begin{pmatrix} 0.384196 \\ 1.03782 \end{pmatrix}.$$

Die Datenpunkte und die resultierende Ausgleichung werden in Abb. 5.7 wiedergegeben.

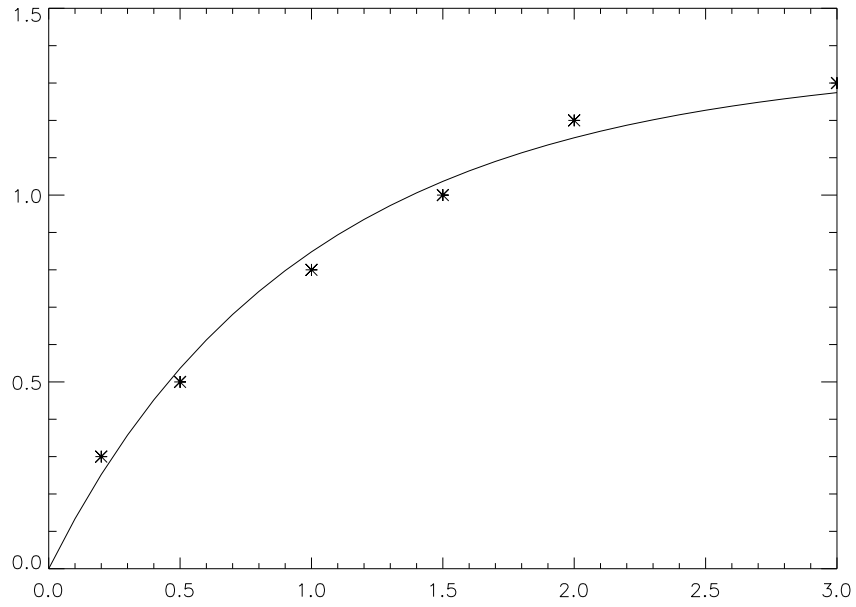


Abbildung 5.7: Datenpunkte und Ausgleichung bzgl. $y = f(x)$ für das Beispiel 5.7.

Aus dem aus der Ausgleichung folgenden Residuenvektor $\boldsymbol{r} = C\boldsymbol{a} + \boldsymbol{d}$ ergibt sich nach Quadrieren ein minimiertes $\chi_0^2 = 0.0101$ und daraus mit $f = 4$ Freiheitsgraden eine globale Standardabweichung $\sigma \approx 0.05$. Die Diagonalelemente der Inversen von A berechnen sich zu $B_{11} \approx 680$ und $B_{22} \approx 420$, so dass sich letztendlich die Parameter und ihre Fehler (auf vier Stellen) zu

$$\alpha_1 = 0.3842 \pm 1.310 \qquad \text{und} \qquad \alpha_2 = 1.038 \pm 1.032$$

ergeben. Man beachte, dass, obwohl ein "schöner" Fit erzielt wurde, die Parameter mit sehr großen Unsicherheiten behaftet sind. Die Ursache dafür ist eine recht schlechte Konditionierung (vgl. Kap. 3.8) der Matrix, $\kappa(A) \approx 5100$ (bzgl. Spektralnorm, Kap. 3.11), die daran liegt, dass die Basisfunktionen relativ ähnlich, d.h. fast *linear abhängig* sind.

Die im vorhergehenden Beispiel aufgetretene schlechte Konditionierung der Matrix A ist kein Einzelfall, sondern eher der Normalfall. Dies macht die Lösung der Normalgleichungen anfällig für numerische Fehler; diese können aufgrund der erforderlichen Berechnung von Skalarprodukten und den damit einhergehenden Rundungsfehlern nochmals verstärkt werden. In dem ergänzenden Kapitel 5.7 wird ein alternatives Lösungverfahren beschrieben, das diese Problematik mit Hilfe orthogonaler Transformationen erheblich verringert.

5.6 Nichtlineare Probleme

In diesem Kapitel wollen wir uns nun mit der Ausgleichung bezüglich *nichtlinearer* Modelle beschäftigen. Exemplarische Fälle wurden schon im Beispiel 5.3 angegeben. Die Schwierigkeit, die hierbei auftritt und die es zu bewältigen gilt, ist diejenige, dass die das χ^2 minimierenden Gleichungen *kein* lineares Gleichungssystem mehr bilden und deshalb *iterativ* gelöst werden müssen.

Wiederum gilt es, bei N Datenpunkten die Länge des Residuenvektors mit Komponenten (Fehlergleichungen)

$$\frac{1}{\sigma_i}\left[f_i(a_1,\ldots,a_M)-y_i\right]=r_i \qquad (i=1,\ldots,N) \tag{5.31}$$

zu minimieren, wobei das Modell $f_i=f(x_i,a_1,\ldots,a_M)$, wie eben ausgeführt, nun *nichtlinear* in den M Parametern $a_1,\ldots,a_M$ sein soll. Wir fordern also, dass

$$\chi^2=\boldsymbol{r}^\intercal\boldsymbol{r}=\sum_{i=1}^{N}\frac{\left[f_i(a_1,\ldots,a_M)-y_i\right]^2}{\sigma_i^2}=\min!,$$

d.h. $a_1,\ldots,a_M$ ergeben sich (im Prinzip und wie immer) aus

$$\boldsymbol{\nabla}\chi^2(\boldsymbol{a})=\boldsymbol{0},$$

und wir erhalten die M gekoppelten und bzgl. der a_j nichtlinearen Gleichungen

$$\frac{1}{2}\frac{\partial\chi^2(\boldsymbol{a})}{\partial a_j}=\sum_{i=1}^{N}\frac{\left(f_i(a_1,\ldots,a_M)-y_i\right)}{\sigma_i^2}\frac{\partial f_i(a_1,\ldots,a_M)}{\partial a_j}=0 \qquad (j=1,\ldots,M). \tag{5.32}$$

Man beachte, dass das minimierte χ_0^2 jetzt nicht mehr $Q(\chi^2,f)$-verteilt ist, da die Freiheitsgrade nichtlinear in χ^2 eingehen (vgl. Kap. 5.3.1). Mit anderen Worten: Bei nichtlinearen Problemen ist *kein Gütetest möglich*!

5.6.1 Minimierung mit dem Gauss-Newton-Verfahren

Wie schon angesprochen, erfolgt die Minimierung nichtlinearer Probleme iterativ. Das einfachste derartige Verfahren ist dabei die Methode von Gauss-Newton, das auf folgendem prinzipiellen Vorgehen beruht.

- Zunächst linearisiere man die Funtionswerte f_i bzgl. der jeweilig aktuelle Parameterwerte $\boldsymbol{a}^{(k)}$,
- setze diese (Funktionswerte) dann in die Fehlergleichungen ein und
- iteriere das Verfahren bzgl. der sich jeweils ergebenden Parameterwerte $\boldsymbol{a}^{(k+1)}$.

Bei einem bekannten Startwert $a_1^{(0)},\ldots,a_M^{(0)}$ (z.B. durch "Raten") lautet die o.g. Linearisierung

$$f_i(a_1,\ldots,a_M)\approx f_i\left(a_1^{(0)},\ldots,a_M^{(0)}\right)+\sum_{j=1}^{M}\frac{\partial f_i(\boldsymbol{a})}{\partial a_j}\Big|_{\boldsymbol{a}^{(0)}}\left(a_j-a_j^{(0)}\right), \qquad (i=1,\ldots,N) \tag{5.33}$$

und durch Einsetzen in die Fehlergleichung (5.36) gewinnt man

$$\frac{1}{\sigma_i}\left(f_i\left(a_1^{(0)},\ldots,a_M^{(0)}\right)+\sum_{j=1}^{M}\frac{\partial f_i(\boldsymbol{a})}{\partial a_j}\Big|_{\boldsymbol{a}^{(0)}}\cdot\delta_j-y_i\right)=\rho_i^{(0)},\qquad(i=1,\ldots,N),\tag{5.34}$$

wobei $\delta_j = a_j - a_j^{(0)}$ die Differenz zwischen neuem und altem Parameter und ρ_i das Residuum der *linearisierten* Fehlergleichung ist, wobei $\rho_i \neq r_i$ außer für lineare Modelle gilt. Durch Umschreiben von (5.39) erhält man

$$\frac{f_i\left(\boldsymbol{a}^{(0)}\right)-y_i}{\sigma_i}+\sum_{j=1}^{M}C_{ij}^{(0)}\delta_j=\rho_i^{(0)},\qquad(i=1,\ldots,N)$$

mit der (durch $(\sigma_i)^{-1}$ modifizierten) "JACOBI-Matrix"

$$C_{ij}^{(0)}=\frac{\partial f_i(a_1,\ldots,a_M)}{\sigma_i\partial a_j}\Big|_{\boldsymbol{a}^{(0)}}$$

und das gesamte linearisierte Gleichungssytem lautet in kompakter Schreibweise

$$C^{(0)}\boldsymbol{\delta}+\boldsymbol{d}^{(0)}=\boldsymbol{\rho}^{(0)},\tag{5.35}$$

mit $C\in\mathbb{R}^{N\times M}$, $\boldsymbol{d},\boldsymbol{\rho}\in\mathbb{R}^N$ und $\boldsymbol{\delta}\in\mathbb{R}^M$. Man beachte, dass in dieser Formulierung unser "neues" $\boldsymbol{d}$ dem ursprünglichen Residuum $\boldsymbol{r}$ entspricht.

Dieses Gleichungssystem *für die Korrekturen* $\boldsymbol{\delta}$ *und die modifizierten Residuen* $\boldsymbol{\rho}$ entspricht dem "üblichen" Fehlergleichungssystem (für die Parameter selbst) und kann mit den gleichen Methoden (Normalgleichungen, orthogonale Transformationen) gelöst werden; Nach Minimierung von $\boldsymbol{\rho}^\intercal\boldsymbol{\rho}$ findet man also $\boldsymbol{\delta}$ und daraus

$$\boldsymbol{a}^{(1)}=\boldsymbol{a}^{(0)}+\boldsymbol{\delta}\tag{5.36}$$

Nochmals:

> In den üblichen Fehlergleichungen (5.27) wird die Designmatrix (5.28) durch die (um $(\sigma_i)^{-1}$ modifizierte) JACOBI-Matrix und der Vektor $\boldsymbol{d}\left(d_i=-\frac{y_i}{\sigma_i}\right)$ durch das aktuelle Residuum mit Komponenten $\left(\frac{f_i-y_i}{\sigma_i}\right)$ ersetzt, und dieses System bzgl. der Länge des modifizierten Residuumvektors minimiert.

Damit finden wir aus den Startwerten $\boldsymbol{a}^{(0)}$ eine erste Korrektur $\boldsymbol{\delta}^{(1)}$ und damit einen neuen Näherungswert $\boldsymbol{a}^{(1)}$ (Gl. 5.41). Daraus werden wiederum die aktualisierten

$$C^{(1)},\quad \boldsymbol{d}^{(1)}\quad\text{und}\quad\boldsymbol{\rho}^{(1)}$$

ermittelt; Minimierung liefert $\boldsymbol{\delta}^{(2)}$ usw.; Insgesamt gilt also

$$\boldsymbol{a}^{(k)}=\boldsymbol{a}^{(k-1)}+\boldsymbol{\delta}^{(k)},\qquad(k=1,2,\ldots).$$

Wenn (hoffentlich!) die Korrekturen schließlich gegen Null konvergiert sind, $\boldsymbol{\delta}^{(k_{\max})}\equiv 0$, dann gilt $\boldsymbol{d}^{(k_{\max})}=\boldsymbol{\rho}^{(k_{\max})}$, und da unser "neuer" Vektor $\boldsymbol{d}\hat{=}\boldsymbol{r}$ dem originalen Residuum entspricht, finden wir *in dieser letzten Iteration* $k_{\max}$

$$\chi_0^2=\|\boldsymbol{r}^{(k_{\max})}\|^2=\|\boldsymbol{\rho}^{(k_{\max})}\|^2.$$

5.8 Beispiel (Nichtlineare Probleme: Ausgleichung mit dem GAUSS-NEWTON-Verfahren). Gegeben sei folgende Messreihe

x	0.1	1.0	2.0	3.0	4.0	5.0
y	0.05	0.25	0.37	0.38	0.55	0.70

aus der die Parameter a_1, a_2 des nichtlinearen Modelles

$$y = a_1 x^{a_2}$$

abgeleitet werden sollen. Im Prinzip lässt sich diese Aufgabe auch durch Logarithmieren lösen (vgl. Beispiel 5.5) wenn man die Fehlerproblematik konsistent berücksichtigt. Hier soll aber der nichtlineare Algorithmus getestet werden!

Es seien keine Fehler angegeben, so dass wir $\sigma_i = 1 \quad \forall i$ setzen. Die Elemente des benötigten Vektors $\boldsymbol{d}$ und der JACOBImatrix C sind dann folgendermaßen definiert $(i = 1, \ldots, 6)$

$$\begin{aligned} d_i &= a_1^{(k)} x_i^{a_2^{(k)}} - y_i \\ C_{i1}^{(k)} &= x_i^{a_2^{(k)}} \qquad \text{aus } \left[\frac{\partial}{\partial a_1}\right] \\ C_{i2}^{(k)} &= a_1^{(k)} x_i^{a_2^{(k)}} \cdot \ln(x_i) \quad \text{aus } \left[\frac{\partial}{\partial a_2}\right] \end{aligned}$$

Abbildung 5.8) visualisiert die Konvergenz des Verfahrens durch Abbildung der Ausgleichsfunktion $y(x, a_1, a_2)$ mit den jeweils aktuellen Parametern. Nach 5 Iterationen sind die Parameter soweit konvergiert, dass eine Änderung der Funktion nicht mehr sichtbar ist.

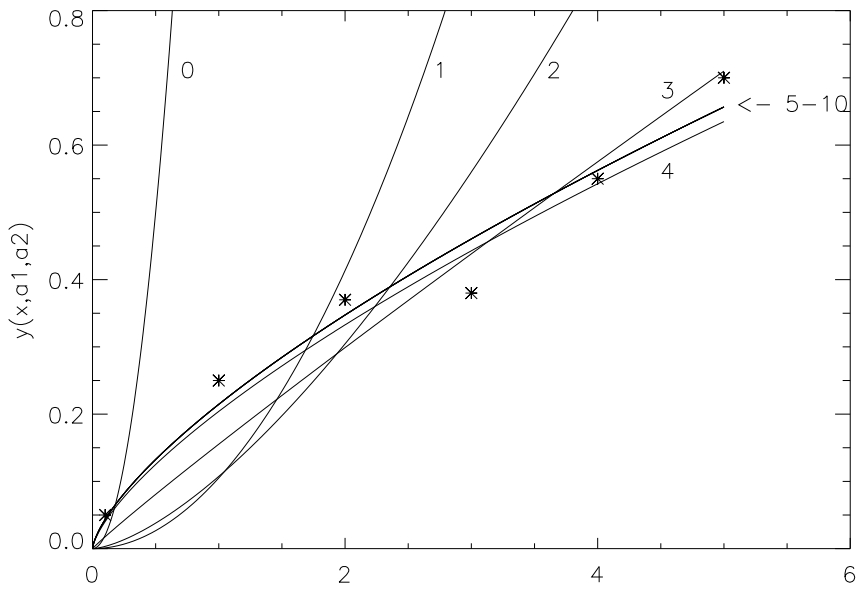


Abbildung 5.8: Konvergenz des nichtlinearen Ausgleichproblems von Beispiel 5.8. Die Ziffern geben den jeweiligen Iterationsschritt an (0: Startwerte), und die Sterne die zu fittenden Datenpunkte.

Folgender `output` gibt die entscheidenden Größen der Iteration wieder; Nach 10 Iterationen sind alle Größen auf 6 Stellen konvergiert, und für die letzte Iteration gilt tatsächlich $\chi^2 = \|\boldsymbol{\rho}\|^2$.

Iteration	a_1	a_2	χ^2	$\|\boldsymbol{\rho}\|^2$
Startwerte	2.0000000	2.0000000	3.7913372 E+03	
1	0.1051422	1.9755337	4.8144546	4.1137148 E-02
2	0.1074560	1.5027140	0.4137889	4.0095866 E-02
3	0.1553198	0.9443398	1.9202616 E-02	2.2767834 E-02
4	0.2040593	0.7053221	1.1833843 E-02	1.1495252 E-02
5	0.2143596	0.6955138	1.0311612 E-02	1.0313257 E-02
6	0.2146427	0.6948923	1.0311215 E-02	1.0311225 E-02
7	0.2146604	0.6948283	1.0311216 E-02	1.0311214 E-02
8	0.2146623	0.6948217	1.0311218 E-02	1.0311216 E-02
9	0.2146624	0.6948210	1.0311219 E-02	1.0311218 E-02
10	0.2146625	0.6948209	1.0311219 E-02	$\approx$ 1.0311220 E-02

5.6.2 Minimierungsmethoden

Bei ungeeigneten Startwerten $\boldsymbol{a}^{(0)}$ oder in problematischen Fällen *garantiert* die GAUSS-NEWTON Linearisierung leider *nicht* das Auffinden der gesuchten Lösung. Eigentlich muss nämlich die *Bedingung*

$$\chi^2\left(\boldsymbol{a}^{(k)}\right) < \chi^2\left(\boldsymbol{a}^{(k-1)}\right), \qquad (k = 1, 2, \dots) \tag{5.37}$$

mitberücksichtigt werden, wobei diese Bedingung in der GAUSS-NEWTON Iteration oftmals verletzt wird (vgl. Beispiel 5.9, erste Tabelle).

Um das Minimum von χ^2 zu finden (mit GAUSS-NEWTON minimieren wir "nur" die linearisierten Gleichungen), muss eine sog. "Abstiegsrichtung" $\boldsymbol{v}^{(k)}$ bekannt sein, so dass der jeweilige Parametersatz

$$\boldsymbol{a}^{(k)} = \boldsymbol{a}^{(k-1)} + t\boldsymbol{v}^{(k)}, \qquad 0 < t \in \mathbb{R} \tag{5.38}$$

die obige Bedingung erfüllt (vgl. Abb. 5.9).

Methode des stärksten Abstieges

Eine mögliche Abstiegsrichtung ist natürlich durch den (negativen) lokalen Gradienten (d.h. die *stärkste* Abstiegsrichtung)

$$-\boldsymbol{\nabla}\chi^2\Big|_{\boldsymbol{a}^{(k-1)}}$$

der vorhergehenden Iteration gegeben. Für die j-te Komponente des Gradienten hatten wir in Gl. 5.32

$$\boldsymbol{\nabla}_j\chi^2(\boldsymbol{a})\Big|_{\boldsymbol{a}^{(k-1)}} = 2\sum_{i=1}^{N} \underbrace{\frac{f_i - y_i}{\sigma_i}}_{r_i^{(k-1)}} \underbrace{\frac{1}{\sigma_i}\frac{\partial f_i}{\partial a_j}}_{C_{ij}^{(k-1)}}\Big|_{\boldsymbol{a}^{(k-1)}}, \qquad (j = 1, \dots, M),$$

gefunden, d.h.

$$\boldsymbol{\nabla}\chi^2\left(\boldsymbol{a}^{(k-1)}\right) = 2 \cdot C^{\intercal(k-1)} \cdot \boldsymbol{r}^{(k-1)}, \tag{5.39}$$

so dass die entsprechende Abstiegsrichtung

$$\boldsymbol{v} = -C^{\intercal(k-1)} \cdot \boldsymbol{r}^{(k-1)}$$

resultiert. Damit kann ein "neuer" Parametersatz dadurch ermittelt werden, dass man $0 < t \in \mathbb{R}$ so bestimmt, dass

$$\chi^2\left(\boldsymbol{a}^{(k)}\right) = \min_t\left(\chi^2\left(\boldsymbol{a}^{(k-1)} - t \cdot C^{\intercal(k-1)}\boldsymbol{r}^{(k-1)}\right)\right)$$

entlang dieser Richtung sein Minimum annimmt. Die folgende Abbildung 5.9 veranschaulicht diesen Gedankengang anhand der (logarithmischen) χ^2-Isokonturen bzgl. des Problemes aus Beispiel 5.8.

Die skizzierte Vorgehensweise erfordert eine (nichtlineare) Minimierung bzgl. des Parameters t, und das Verfahren selbst ist nicht sehr effizient. Eine "schnellere" Lösung ergibt sich mit der im nächsten Abschnitt diskutierten Methode.

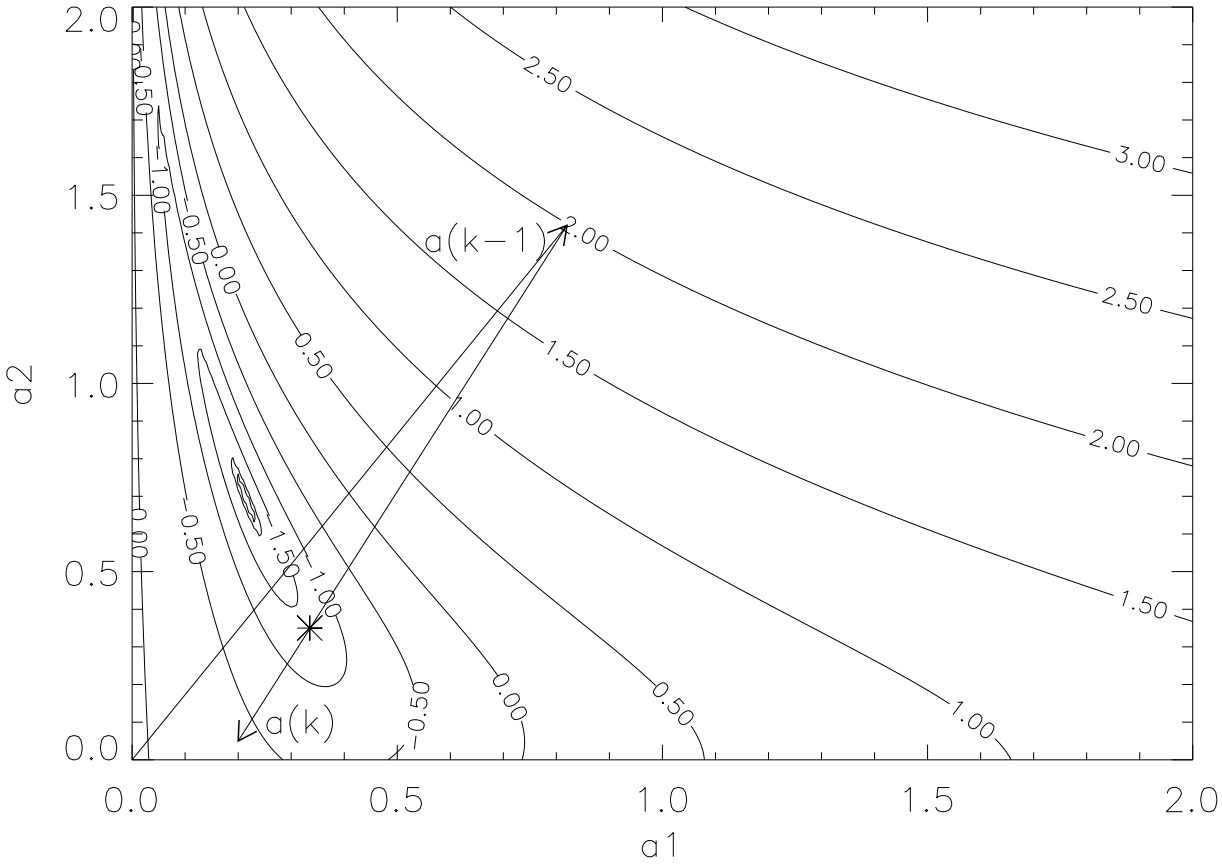
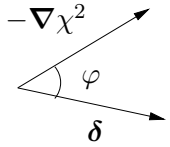


Abbildung 5.9: Isokontures von $\log_{10}(\chi^2(a_1, a_2))$ des nichtlinearen Ausgleichproblems von Beispiel 5.8. Die Methode des stärksten Abstieges, $\boldsymbol{a}^{(k)} = \boldsymbol{a}^{(k-1)} + t \cdot \left(-\boldsymbol{\nabla}\chi^2\Big|_{\boldsymbol{a}^{(k-1)}}\right)$ wird durch die Pfeile, der erste entsprechend $\boldsymbol{a}^{(k-1)}$, der zweite entsprechend $\boldsymbol{a}^{(k)}$, angedeutet. Der Wert von t wird so gewählt, dass das Minimum entlang von $\boldsymbol{a}^{(k)}$ (mit einem Stern gekennzeichnet) "getrofffen" wird. Man beachte die Lage des globablen Minimums von χ^2 entsprechend der in Beispiel 5.8 berechneten Parameter $a_1 = 0.2147$ und $a_2 = 0.6948$.

Die modifizierte Gauss-Newton Methode

Solange der lokale Gradient $\boldsymbol{\nabla}\chi^2\left(\boldsymbol{a}^{(k-1)}\right) \neq 0$ nicht verschwindet, stellt auch der Korrekturvektor $\boldsymbol{\delta}^{(k)}$ für die Gauss-Newton Näherung $\boldsymbol{a}^{(k-1)}$ (5.36) eine Abstiegsrichtung dar. Es lässt sich nämlich zeigen (hier ohne Beweis), dass $-\boldsymbol{\nabla}\chi^2$ und $\boldsymbol{\delta}$ einen spitzen Winkel ($\cos\varphi > 0$, vgl. folgende Skizze) bilden, bzw.



$$\left(\boldsymbol{\nabla}\chi^2\left(\boldsymbol{a}^{(k-1)}\right)\right)^{\mathsf{T}} \cdot \boldsymbol{\delta}^{(k)} < 0 \qquad \forall\ k, k = 1, \ldots \tag{5.40}$$

Mit dieser Erkenntnis lässt sich folgender einfache Algorithmus konstruieren, der einerseits das Gauss-Newton-Verfahren verwendet, andererseits aber auch der Bedingung (5.37) genügt.

(i) Man berechne $\boldsymbol{\delta}^{(k)}$ aus $\boldsymbol{a}^{(k-1)}$ wie im originalen GAUSS-NEWTON-Verfahren.

(ii) Danach bestimme man

$$\chi^2\left(\boldsymbol{y}(t)\right) \qquad \text{mit} \qquad \boldsymbol{y} = \boldsymbol{a}^{(k-1)} + t \cdot \boldsymbol{\delta}^{(k)}$$

wobei t, ausgehend von $t = 1$ (Korrektur entsprechend dem originalen Verfahren), sukzessive halbiert wird, $t = \frac{1}{2}, \frac{1}{4}, \ldots$, solange bis

(iii) $\chi^2\left(\boldsymbol{y}(t)\right) < \chi^2\left(\boldsymbol{a}^{(k-1)}\right)$ gilt. Dant wird $\boldsymbol{a}^{(k)} = \boldsymbol{y}$ gesetzt, und man beginnt

(iv) einen neuen Iterationsschritt $k + 1$

Das modifizierte GAUSS-NEWTON-Verfahren konvergiert auf jeden Fall, in der Nähe der Lösung sogar quadratisch; am Anfang kann die Konvergenz allerdings sehr langsam sein, insbesondere bei ungünstiger Wahl des Startvektors $\boldsymbol{a}^{(0)}$.

5.9 Beispiel (Nichtlineare Probleme: Ausgleichung mit dem modifizierten GAUSS-NEWTON-Verfahren für Beispiel 5.8). In Bespiel 5.8 wurde obige Bedingung $\chi^2\left(\boldsymbol{y}(t)\right) < \chi^2\left(\boldsymbol{a}^{(k-1)}\right)$ unter Verwendung des Startvektors $\boldsymbol{a}^{(0)} = (2, 2)^\intercal$ in allen Schritten schon für den Wert $t = 1$ erreicht (vgl. Konvergenz von χ^2 in der entsprechenden Tabelle). Verwendet man andere Startwerte, kann sich dies allerdings ändern. Im Folgenden zeigen wir zunächst das Konvergenzverhalten des *originalen* Verfahrens, wobei wir jetzt in der Nähe des Nullpunktes starten, $\boldsymbol{a}^{(0)} = (0.05, 0.05)^\intercal$.

Iteration	a_1	a_2	χ^2	$\|\rho\|^2$		
0	5.0000000 E-02	5.0000000 E-02	0.9118437			
1	0.3073723	2.7497268	**8.4046893 E+02**	3.6074547 E-02	$\leftarrow$	$> \chi^2(\boldsymbol{a}^{(0)})$
2	4.9700469 E-02	2.6653039	11.0061922	7.6535761 E-02		
3	5.4439206 E-02	2.0983009	1.0875301	7.2366618 E-02		
4	9.6185878 E-02	1.2576905	4.6056997 E-02	4.5428138 E-02		
5	0.1795839	0.6609693	**5.5052727 E-02**	1.6426157 E-02	$\leftarrow$	$> \chi^2(\boldsymbol{a}^{(4)})$
6	0.2156046	0.6982932	1.0402102 E-02	1.0332459 E-02		
7	0.2145631	0.6952044	1.0311240 E-02	1.0311434 E-02		
8	0.2146515	0.6948607	1.0311215 E-02	1.0311216 E-02		
9	0.2146613	0.6948251	1.0311221 E-02	1.0311214 E-02		
10	0.2146623	0.6948214	1.0311221 E-02	1.0311222 E-02		

In Iteration 1 und 5 wächst also das resultierende χ^2 an. Mit dem modifizierten GAUSS-NEWTON Verfahren erzielen wir andererseits folgenden Konvergenzzyklus

Iteration	a_1	a_2	χ^2	$\|\rho\|^2$	t
0	5.0000000 E-02	5.0000000 E-02	**0.9118437**		
			8.4046893 E+02	0.9118437	1.00
			1.7040532	0.9118437	0.50
			0.2351825	0.9118437	0.25
1	0.1143431	0.7249317	0.2351825	0.9118437	
			1.1277056 E-02	0.2351825	1.00 $\leftarrow$
2	0.2137730	0.6756312	1.1277056 E-02	0.2351825	
			1.0311885 E-02	1.1277056 E-02	1.00 $\leftarrow$
3	0.2152012	0.6931893	1.0311885 E-02	1.1277056 E-02	
			1.0311226 E-02	1.0311885 E-02	1.00 $\leftarrow$
4	0.2147089	0.6946501	1.0311226 E-02	1.0311885 E-02	
			1.0311213 E-02	1.0311226 E-02	1.00 $\leftarrow$
5	0.2146673	0.6948032	1.0311213 E-02	1.0311226 E-02	

Wie ersichtlich, ist nur bei der ersten Iteration eine Reduktion von t notwendig. Für die Iterationen 2-5 reicht $t = 1$ ($\rightarrow$ Standardverfahren) aus, um das χ^2 zu reduzieren. Man beachte, dass insgesamt nur fünf Iterationen notwendig sind, um die gleiche Präzision wie früher zu erreichen.

Man beachte, dass die *Abstiegsrichtung* bei Variation von t immer gleich bleibt ($= \boldsymbol{\delta}^{(k)}$). Dies wird sich im nächsten zu besprechenden Verfahren ändern.

Das LEVENBERG-MARQUARDT-Verfahren

ist die von allen bisher diskutierten Methoden die schnellste und heutiger Standard. Das Problem bei den vorhergehenden Algorithmen,

$$\boldsymbol{a}^{(k)} = \boldsymbol{a}^{(k-1)} + t \cdot \boldsymbol{v}^{(k)}$$

mit

$$\chi^2\left(\boldsymbol{a}^{(k)}\right) < \chi^2\left(\boldsymbol{a}^{(k-1)}\right)$$

war die Steuerung von t. MARQUADT (1963), basierend auf Vorarbeiten von LEVENBERG, schlug zu einer zweckmäßigen (und selbstregulierenden) Steuerung der "Länge" des Korrekturvektors vor, die Parameter $\boldsymbol{a}^{(k)}$ durch folgendes Vorgehen zu aktualisieren,

$$\boldsymbol{a}^{(k)} = \boldsymbol{a}^{(k-1)} + \boldsymbol{\delta}^{(k)} = \boldsymbol{a}^{(k-1)} + \|\boldsymbol{\delta}^{(k)}\|_2 \cdot \hat{\boldsymbol{\delta}}^{(k)},$$

wobei die Korrektur (d.h. ihre *Richtung* und ihre *Länge*) $\boldsymbol{\delta}^{(k)}$ aus Minimierung von

$$\|C^{(k-1)} \cdot \boldsymbol{\delta}^{(k)} + \boldsymbol{d}^{(k-1)}\|_2^2 + \lambda^2 \cdot \|\boldsymbol{\delta}^{(k)}\|_2^2 = \min \tag{5.41}$$

bei **vorgegebenem** $\lambda > 0$ berechnet werden soll. Auf diese Weise wird ein Kompromiss bzgl. der Minimierung von $\|\boldsymbol{\rho}^{(k-1)}\|^2$ (erster Term) und der Minimierung der Länge des Korrekturvektors $\|\boldsymbol{\delta}^{(k)}\|^2$ (zweiter Term) erzielt.

Es ist sofort ersichtlich, dass für $\lambda = 0$ das Verfahren der originalen GAUSS-NEWTON-Linearisierung entspricht. Der entscheidende *"Trick"* ist nun der folgende: Die Länge der Korrekturvektors, $\|\boldsymbol{\delta}\|$, lässt sich durch λ beeinflussen, und zwar dergestalt, dass λ und $\|\boldsymbol{\delta}\|$ antikorreliert sind, d.h.

$$\lambda \uparrow \quad \Rightarrow \quad \|\boldsymbol{\delta}\| \downarrow \quad \text{kleine Korrektur}$$

$$\lambda \downarrow \quad \Rightarrow \quad \|\boldsymbol{\delta}\| \uparrow \quad \text{große Korrektur,}$$

wenn $\boldsymbol{\delta}$ eine Abstiegsrichtung ist!

5.10 Beweis.

(i) Zunächt gilt es zu zeigen, dass $\boldsymbol{\delta}$ eine Abstiegsrichtung, d.h. dass

$$\left[\nabla\chi^2\left(\boldsymbol{a}^{(k-1)}\right)\right]^{\mathsf{T}} \cdot \boldsymbol{\delta}^{(k)} < 0,$$

falls $\nabla\chi^2 \neq 0$ (Gradient und Korrektur müssen einen spitzen Winkel bilden, vgl. Gl. 5.40 und die dazugehörige Skizze). Aus Gründen der Übersichtlichkeit wird im Weiteren der Iterationsindex k unterdrückt. Die LEVENBERG-MARQUARDT*sche Minimierungsbedingung*

$$\|C \cdot \boldsymbol{\delta} + \boldsymbol{d}\|^2 + \lambda^2\|\boldsymbol{\delta}\|^2 = min$$

entspricht nun einem GAUSS-NEWTON*schen Minimierungsproblem bzgl. der modifizierten Länge* $\tilde{\boldsymbol{\rho}}^{\mathsf{T}}\tilde{\boldsymbol{\rho}}$

$$\tilde{C}\boldsymbol{\delta} + \tilde{\boldsymbol{d}} = \tilde{\boldsymbol{\rho}}, \tag{5.42}$$

wenn $\tilde{C}$ und $\tilde{\boldsymbol{d}}$ durch

$$\tilde{C} = \begin{pmatrix} C \\ \dots \\ \lambda\mathbb{1} \end{pmatrix} \in \mathbb{R}^{(N+M)\times M} \qquad \textit{und} \qquad \tilde{\boldsymbol{d}} = \begin{pmatrix} \boldsymbol{d} \\ \dots \\ \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{N+M}$$

definiert sind. Die modifizierte Matrix $\tilde{C}$ hat, unabhängig von der ursprünglichen JAKOBI*matrix C, auf jeden Fall den Rang M (aufgrund der Einheitsmatrix im "unteren" Teil), wenn $\lambda > 0$ ist. Daraus folgt sofort, dass $\tilde{C}^\intercal\tilde{C}$ positiv definit symmetrisch ist (vgl. Gln. 5.28 ff.), mit anderen Worten, dass $\boldsymbol{x}^\intercal A\boldsymbol{x} > 0 \quad \forall \boldsymbol{x} \neq \boldsymbol{0}$ und $A = \tilde{C}^\intercal\tilde{C}$. Mit dieser Erkenntnis folgt ausserdem, dass auch A^{-1} positiv definit ist, denn $\boldsymbol{x}^\intercal A\boldsymbol{x} = \boldsymbol{x}^\intercal AA^{-1}A\boldsymbol{x} = \boldsymbol{x}'^\intercal A^{-1}\boldsymbol{x}' > 0$ (beachte: $A = A^\intercal$).*

Aufgrund der positiven Definitheit lassen sich also für A die Normalgleichungen zur Minimierung anwenden

$$\begin{aligned}
\boldsymbol{\delta} &= -\left(\tilde{C}^\intercal\tilde{C}\right)^{-1}\left(\tilde{C}^\intercal\tilde{\boldsymbol{d}}\right) && \text{(vgl. Gl. 5.29)}\\
&= -\left(\tilde{C}^\intercal\tilde{C}\right)^{-1}\left(C^\intercal\boldsymbol{d}\right)\\
&= -\left(\tilde{C}^\intercal\tilde{C}\right)^{-1}\left[\frac{1}{2}\boldsymbol{\nabla}\chi^2(\boldsymbol{a})\right]\\
\Rightarrow \qquad \left[\boldsymbol{\nabla}\chi^2\right]^\intercal\cdot\boldsymbol{\delta} &= -\frac{1}{2}\underbrace{\left[\boldsymbol{\nabla}\chi^2\right]^\intercal\cdot(\tilde{C}^\intercal\tilde{C})^{-1}\cdot\boldsymbol{\nabla}\chi^2}_{\text{quadr. Form, } (\tilde{C}^\intercal\tilde{C})^{-1} \text{ pos. def.}} < 0!
\end{aligned}$$

Damit ist also die über (5.41) definierte Korrektur $\boldsymbol{\delta}$ tatsächlich eine Abstiegsrichtung $\forall\,\lambda > 0$; diese Abstiegsrichtung ist im Allgemeinen günstiger als diejenige aus dem modifiziertem GAUSS-NEWTON *Verfahren.*

(ii) Als Zweites ist jetzt zu zeigen, dass $\|\boldsymbol{\delta}\|$ und λ antikorreliert sind. Zunächst gilt, dass

$$A = \tilde{C}^\intercal\tilde{C} = C^\intercal C + \lambda^2\cdot\mathbb{1} \qquad \in \mathbb{R}^{M\times M}.$$

Da $C^\intercal C$ reell symmetrisch ist, existieren orthogonale Matrizen $U \in \mathbb{R}^{M\times M}$, so dass

$$U^\intercal C^\intercal CU = D$$

(vgl. Kap. 4) und D eine Diagonalmatrix mit Elementen $d_i \geq 0$ (da $C^\intercal C$ positiv semi*definit) ist. Also gilt*

$$U^\intercal AU = D + \lambda^2\cdot\mathbb{1},$$

und der Korrekturvektor lässt sich durch

$$\begin{aligned}
\boldsymbol{\delta} &= -\left(\tilde{C}^\intercal\tilde{C}\right)^{-1}\left(C^\intercal\boldsymbol{d}\right)\\
&= -\left[U\left(D+\lambda^2\mathbb{1}\right)U^\intercal\right]^{-1}\left(C^\intercal\boldsymbol{d}\right) \qquad (5.43)
\end{aligned}$$

beschreiben. ($\boldsymbol{d}$ nicht zu verwechseln mit d_i bzgl. D). Die Länge von $\boldsymbol{\delta}$ (d.h. ihr Quadrat) ergibt sich damit als

$$\begin{aligned}
\boldsymbol{\delta}^\intercal\boldsymbol{\delta} &= \boldsymbol{d}^\intercal\cdot C\Big[U\underbrace{\left(D+\lambda^2\mathbb{1}\right)^{-1}}_{\text{Diagonalelemente}>0}U^\intercal\Big]\left[U\left(D+\lambda^2\mathbb{1}\right)^{-1}U^\intercal\right]\cdot\left(C^\intercal\boldsymbol{d}\right)\\
&= \left(\boldsymbol{d}^\intercal CU\right)\left(D+\lambda^2\mathbb{1}\right)^{-2}\left(U^\intercal C^\intercal\boldsymbol{d}\right)\\
&= \boldsymbol{h}^\intercal\left(D+\lambda^2\mathbb{1}\right)^{-2}\boldsymbol{h} = \sum_{i=1}^{M}\frac{h_i^2}{(d_i+\lambda^2)^2} \qquad (5.44)
\end{aligned}$$

Wenn man berücksichtig, dass h_i und d_i unabhängig von λ sind, folgt sofort dass $\|\boldsymbol{\delta}\|^2$ fällt, wenn λ anwächst und umgekehrt! $\square$.

Anzumerkung bleibt, dass, falls λ sehr groß wird,

$$\boldsymbol{\delta} \overset{\lambda \gg 1}{\approx} -\left(\lambda^2 \mathbb{1}\right)^{-1} \left(C^\intercal \boldsymbol{d}\right) \approx -\frac{1}{2\lambda^2} \boldsymbol{\nabla} \chi^2(\boldsymbol{a})$$

die LEVENBERG-MARQUARDT-Korrektur einer (sehr kleinen) Korrektur in *Richtung des stärksten Abstieges* entspricht.

5.11 Beispiel (LEVENBERG - MARQUARDT Korrekturen für den ersten Schritt aus Beispiel (5.9)).

λ	a_1	a_2	χ^2	$\|\boldsymbol{\delta}\|$
1.0000000 E-03	0.3073756	2.7496104	8.4017437 E+02	2.7118516
2.0000001 E-03	0.3073855	2.7492614	8.3930304 E+02	2.7115049
4.0000002 E-03	0.3074251	2.7478640	8.3581213 E+02	2.7101178
8.0000004 E-03	0.3075834	2.7422898	8.2202936 E+02	2.7045839
1.6000001 E-02	0.3082094	2.7202237	7.6964221 E+02	2.6826789
3.2000002 E-02	0.3106124	2.6354723	5.9739410 E+02	2.5985739
6.4000003 E-02	0.3188433	2.3443582	2.4851335 E+02	2.3100555
0.1280000	0.3387026	1.6330513	26.4827747	1.6091615
0.2560000	0.3612898	0.7603183	0.7921224	0.7755343
0.5120000	0.3653020	0.2770010	6.9263600 E-02	0.3885161
1.0240000	0.3355445	0.1162315	0.1616566	0.2931250
2.0480001	0.2507845	7.0527077 E-02	0.3002898	0.2018311
4.0960002	0.1414439	5.6336302 E-02	0.5745031	9.1663167 E-02
8.1920004	7.8762844 E-02	5.1755205 E-02	0.7949180	2.8816350 E-02
16.3840008	5.7686672 E-02	5.0453164 E-02	0.8796216	7.7000153 E-03
32.7680016	5.1955383 E-02	5.0114267 E-02	0.9035793	1.9587171 E-03
65.5360031	5.0490998 E-02	5.0028630 E-02	0.9097642	4.9183273 E-04
1.3107201 E+02	5.0122887 E-02	5.0007161 E-02	0.9113229	1.2309354 E-04
2.6214402 E+02	5.0030731 E-02	5.0001793 E-02	0.9117134	3.0781852 E-05
5.2428802 E+02	5.0007682 E-02	5.0000448 E-02	0.9118111	7.6959932 E-06

Wir verwenden wiederum den Startwert $\boldsymbol{a}^{(0)} = (0.05, 0.05)^\intercal$, vgl. Beispiel 5.9, und variieren hier λ von $10^{-3} \ldots 524$ (jeweils um eine Faktor 2). Wie vorhergesagt, wird $\|\boldsymbol{\delta}\|$ dadurch monoton verringert.

Zunächst nimmt $\chi^2(\boldsymbol{a}^{(1)})$ mit wachsendem λ ab (mit einem dem GAUSS-NEWTON-Verfahren entsprechenen Ausgangswert, vgl. Gl. 5.41) erreicht sein Minimum bei $\lambda = 512$ und wächst dann wieder an. Für sehr kleine λ werden die Startwerte der Parameter wieder erreicht, da die Korrekturen sehr klein geworden sind.

Man beachte

(i) Die zum erzielten Minimum von χ^2 (bei $\lambda = 512$) zugehörigen Parameter $\boldsymbol{a}^{(1)}$ liegen *erheblich* näher an der finalen Lösung, als es nach den ersten Iterationsschritten der vorhergehenden Verfahren der Fall war.

(ii) Die Abstiegsrichtung ändert sich mit λ, und zwar von $\boldsymbol{\delta}$(GAUSS-NEWTON) für $\lambda \approx 0$ auf $-\boldsymbol{\nabla}\chi^2(\boldsymbol{a})$ (stärkster Abstieg) für $\lambda \gg 1$.

Der LEVENBERG-MARQUARDT-Algorithmus kann wie folgt formuliert werden

(i) Die Startwerte der Parameter, $\boldsymbol{a}^{(0)}$, werden "geraten."

(ii) Man berechne $\chi^2\left(\boldsymbol{a}^{(0)}\right)$ und beginne den Algorithmus mit einem typischen Startwert $\lambda = 10^{-3}$.

(iii) Für die aktuellen Werte von $\boldsymbol{a}$ und λ berechne man mittels Gl. 5.41 die Korrektur $\boldsymbol{\delta}$. Effektive Verfahren dazu findet man in der Literatur.

(iv) Aus den sich ergebenden Größen resultiert das "neue" $\chi^2(\boldsymbol{a} + \boldsymbol{\delta})$.

`FALLS` $\chi^2(\boldsymbol{a} + \boldsymbol{\delta}) \geq \chi^2(\boldsymbol{a})$ `DANN`
vergrößere man λ (um einen Faktor 2 ... 10), d.h. *reduziere* $\|\boldsymbol{\delta}\|$ (Schrittweitenverkleinerung)
`GEHE NACH` (iii)

`FALLS` $\chi^2(\boldsymbol{a}+\boldsymbol{\delta}) < \chi^2(\boldsymbol{a})$ `DANN`
$\boldsymbol{a}^{neu} = \boldsymbol{a}+\boldsymbol{\delta}$
und man verkleinere λ für den nächsten Schritt (Faktor 2 ... 10) , d.h. *vergrößere* $\|\boldsymbol{\delta}\|$
`GEHE NACH` (iii)

(v) `STOP, FALLS` $\chi^2(\boldsymbol{a}+\boldsymbol{\delta}) < \chi^2(\boldsymbol{a})$ `UND` $\Delta\chi^2 = \mathcal{O}(10^{-3})$

5.12 Beispiel (Vollständige LEVENBERG-MARQUARDT **Iteration für das Beispiel 5.9).**

	λ	a_1	a_2	χ^2	χ^2 (alt)
0		0.05	0.05		0.9118437
1	0.001	0.3073756	2.7496104	840.17737	0.9118437
1	0.005	0.3074549	2.7468174	833.20685	0.9118437
1	0.025	0.3093837	2.6788180	680.10040	0.9118437
1	0.1250000	0.3378532	1.6639022	29.3314800	0.9118437
1	0.6250001	0.3610862	0.2092060	**0.092393279**	0.9118437
akzeptiert	0.6250001	0.3610862	0.2092060	0.092393279	
2	**0.1250000**	0.2636830	0.5008482	**0.019743597**	0.092393279
akzeptiert	0.1250000	0.2636830	0.5008482	0.019743597	
3	**0.025**	0.2213230	0.6618897	**0.010654236**	0.019743597
akzeptiert	0.025	0.2113230	0.6618897	0.010654236	
4	**0.005**	0.2155805	0.6913252	**0.010313449**	0.010654236
konvergiert	0.005	0.2155805	0.6913252	0.010313449	

Vergrößerung/Verkleinerung von λ jeweils um einen Faktor 5.

Iteration 0 : Startwert $\boldsymbol{a}^{(0)} = (0.05, 0.05)^\intercal$
Iteration 1 : Vergrößerung von λ, bis $\chi^2(\boldsymbol{a}+\boldsymbol{\delta}) < \chi^2(\boldsymbol{a}^{(0)})$.
Iteration 2 : Verkleinerung von λ, wird sofort akzeptiert.
Iteration 3 : weitere Verkleinerung von λ, wird sofort akzeptiert.
Iteration 4 : weitere Verkleinerung von λ, Konvergenz!

5.7 Ergänzung: Lineare Ausgleichungsprobleme – Orthogonale Transformation

Am Ende von Kap. 5.5 hatten wir darauf hingewiesen, dass die Ausgleichung mit Hilfe der *Normalgleichungen* aufgrund der Berechnung von Skalarprodukten und der typischerweise schlechten Konditionierung der Matrix A äusserst fehleranfällig ist. Im Folgenden diskutieren wir kurz eine erheblich stabilere Alternative.

Die Methode beruht im Wesentlichen darauf, dass orthogonale Transformationen die Vektorlänge erhalten. Fall also eine Matrix $Q \in \mathbb{R}^{N\times N}$ orthogonal ist, ist die Minimierung des Residuums $\boldsymbol{r}$ (vgl. Gl. 5.27),

$$C \cdot \boldsymbol{a} + \boldsymbol{d} = \boldsymbol{r}$$

äquivalent mit der Minimierung von $\hat{\boldsymbol{r}}$

$$Q^\intercal C\boldsymbol{a} + Q^\intercal \boldsymbol{d} = Q^\intercal \boldsymbol{r} = \hat{\boldsymbol{r}}. \tag{5.45}$$

Wir fordern nun, dass die Transformation eine Quasirechtsdreiecksmatrix (U-Matrix) erzeugen soll,

$$Q^\intercal C = \hat{R}, \qquad \hat{R} = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

wobei $R \in \mathbb{R}^{M\times M}$ eine "echte" Rechtsdreicksmatrix und die Nullmatrix $0 \in \mathbb{R}^{(N-M)\times M}$ ist.

5.13 Satz. *Zu jeder Matrix $C \in \mathbb{R}^{N\times M}$ mit M linear unabhängigen Spaltenvektoren $M < N$ existiert obige Transformation mit orthogonaler Matrix Q, so dass die Rechtsdreiecksmatrix R* regulär *ist.*

5.14 Beweis. *Der Beweis erfolgt durch Konstruktion der Matrix $\hat{R}$ über eine Folge von* Givens-*Rotationen (vgl. Kap. 4.3)*

$$\begin{array}{cccccc} (1,2), & (1,3), & (1,4) & \dots & (1,N) \\ & (2,3), & (2,4) & \dots & (2,N) \\ & & & & \vdots \\ & & & & (M,N) \end{array}$$

Da Q regulär und der $\operatorname{rang}(C) = M$ *ist, muss ebenfalls* $\operatorname{rang}(\hat{R}) = M$ *gelten und damit R regulär sein.* □

Nach erfolgter orthogonaler Transformation gilt also

$$\hat{R}\boldsymbol{a} + \hat{\boldsymbol{d}} = \hat{\boldsymbol{r}}$$

wobei diese Gleichung in Komponentenform folgendermaßen lautet

$$\begin{aligned} R_{11}a_1 + R_{12}a_2 + \cdots + R_{1M}a_M + \hat{d}_1 &= \hat{r}_1 \\ R_{22}a_2 + \cdots + R_{2M}a_M + \hat{d}_2 &= \hat{r}_2 \\ &\vdots \\ R_{MM}a_M + \hat{d}_M &= \hat{r}_M \\ \hat{d}_{M+1} &= \hat{r}_{M+1} \\ &\vdots \\ \hat{d}_N &= \hat{r}_N \end{aligned} \tag{5.46}$$

Unser Ziel ist es (wie immer), $\|\hat{\boldsymbol{r}}\|^2 \stackrel{!}{=} \|\boldsymbol{r}\|^2$ zu minimieren. *Wenn* nun für die ersten M Gleichungen $\hat{r}_1, \dots, \hat{r}_M = 0$ gilt, folgt daraus offensichtlich, dass

$$\chi^2 = \min \|\hat{\boldsymbol{r}}\|^2 = (\hat{r}_{M+1}, \dots, \hat{r}_N) \begin{pmatrix} \hat{r}_{M+1} \\ \vdots \\ \hat{r}_N \end{pmatrix},$$

da die letzten $N - N$ Komponenten unabhängig von den Unbekannten a_i sind.

Demzufolge lässt sich $\boldsymbol{a}$ durch Lösung des Gleichungssytems

$$R \cdot \boldsymbol{a} + \hat{\boldsymbol{d}}_{(1\dots M)} = \boldsymbol{0} \tag{5.47}$$

sofort berechnen, und die tatsächlichen Residuen ergeben sich durch Rücktransformation

$$\boldsymbol{r} = Q\hat{\boldsymbol{r}}. \tag{5.48}$$

Aus diesen Überlegungen resultiert folgender einfacher

Algorithmus.

(i) Man zerlege $C = Q\hat{\boldsymbol{R}}$ (über Givens-Rotationen oder Householder-Transformationen, vgl. Kap. 4.3) und

(ii) berechne $\hat{\boldsymbol{d}} = Q^\intercal \boldsymbol{d}$.

(iii) Mit diesen Größen wird das Gleichungssystem $R\boldsymbol{a} + \hat{\boldsymbol{d}}_{(1\dots M)} = \boldsymbol{0}$ (durch simples Rückwärtseinsetzen) gelöst, und man erhält sofort den gesuchten Parametersatz $\boldsymbol{a}$.

(iv) Das minimierte χ_0^2 resultiert aus $\chi_0^2 = \|\hat{\boldsymbol{d}}_{M+1}, \dots, \hat{\boldsymbol{d}}_N\|^2$,

(v) und die Residuen (falls gewünscht) ergeben sich zu $\boldsymbol{r} = Q\hat{\boldsymbol{r}}$.

Anmerkungen. Wir wir zuvor schon festgestellt haben, ist dieser Algorithmus erheblich *stabiler* als die Lösung der Normalgleichungen (über das CHOLESKY-Verfahren). Es stellt sich die Frage, warum das so ist.

Mit Hilfe der orthogonalen Transformation würde sich für die (hier nicht benötigte) Matrix A ergeben,

$$A = C^\intercal C = \left(\hat{R}^\intercal Q^\intercal\right)\left(Q\hat{R}\right) = \hat{R}^\intercal\hat{R} \stackrel{!}{=} R^\intercal R$$

dass sie in das Produkt einer Links- und einer Rechtsdreicksmatrix zerfällt.

Andererseits ergab sich aus den Normalgleichungen die alternative Darstellung $A = LL^\intercal$, da A positiv definit symmetrisch ist. R ist also *im Prinzip* und im Wesentlichen (d.h. bis auf Vorzeichen) gleich $L^\intercal$, und wir erhalten (wiederum im Prinzip) das gleiche Ergebnis, wenn wir $\boldsymbol{a}$ einerseits aus den Normalgleichungen und andererseits über das hier vorgestellte Verfahren ermitteln.

Der springende Punkt ist hier, dass R und L auf unterschiedliche Weise berechnet werden. Während die orthogonalen Transformationen die Norm der jeweiligen Spaltenvektoren erhalten, wird diese beim CHOLESKY-Verfahren sukzessive reduziert, und es ist eine Stellenauslöschung möglich. Ebenfalls müssen zur Berechnung von R keine Skalarprodukte berechnet werden!

Kapitel 6

Funktionenapproximation – Fourier-Reihen und Tschebyscheff-Polynome

In diesem Kapitel stellen wir uns die Aufgabe, eine gegebene Funktion $f(x)$ durch eine *Ersatzfunktion* $g(x)$ zu approximieren, und zwar bzgl. eines bestimmten Intervalles und im quadratischen Mittel. Im Weiteren werden wir uns dabei auf Ersatzfunktionen beschränken, die Elemente eines *linearen* Funktionenraumes und *orthogonal* sind.

Diese Aufgabe gilt es immer dann zu lösen, wenn

- die Orginalfunktion äußerst schwierig oder zeitaufwendig zu berechnen ist, aber in einem numerischen Algorithmus häufig auftritt, oder
- eine (numerisch) einfach zu behandelnde "Ersatzfunktion" für eine *periodische* Orginalfunktion gesucht wird.

Beispiele für den ersten Punkt sind die Gamma- oder die Fehlerfunktion, für den zweiten Punkt die (2π)-periodische Dachfunktion.

6.1 Beispiel (Polynomapproximation für die Gammafunktion). In Kap. 5 benötigten wir zur Berechnung der χ^2-Verteilung und der "Fitgüte" die Gammafunktion,

$$\Gamma(x) = \int_0^\infty t^{x-1} \mathrm{e}^{-t} \mathrm{d}t,$$

wobei eine direkte und präzise numerische Quadratur (Kap. 7) relativ zeitaufwendig ist; falls man die Funktion (bei verschiedenen Argumenten) relativ häufig benötigt, verbietet sich solch eine direkte Lösung aus "ökonomischen" Gründen.

Schlägt man in geeigneten Büchern nach, die entsprechende Näherungen bereitstellen[1], so findet man für das Intervall $0 \le x \le 1$ (aufgrund der Rekursion $\Gamma(x) = x\Gamma(x)$ muss die Funktion nur in diesem Intervall bekannt sein) die Polynomnäherung

$$\Gamma(x+1) = 1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + \epsilon(x), \qquad |\epsilon(x)| \le 5 \cdot 10^5,$$

wenn

$$\begin{aligned} a_1 &= -0.57486\,46 \\ a_2 &= 0.95123\,63 \\ a_3 &= -0.69985\,88 \\ a_4 &= 0.42455\,49 \\ a_5 &= -0.10106\,78 \end{aligned}$$

Diese Näherung (inkl. der Fehlerabschätzung) ist ein direktes Ergebnis einer *Funktionenapproximation* und resultiert hier aus der sog. Tschebyscheff-Interpolation, die wir zum Schluss dieses Kapitels kennenlernen werden.

[1] z.B. "Handbook of Mathematical Functions", eds. M. Abramowitz & I.A. Stegun

6.1 FOURIER-Polynome und -Reihen

Es sei eine Funktion $f : \mathbb{R} \to \mathbb{R}$ gegeben, die stückweise stetig mit der Periode 2π ist, d.h.

$$f(x + 2\pi) = f(x) \qquad \forall x \in \mathbb{R}, \tag{6.1a}$$

wobei $f(x)$ Sprungstellen haben darf mit

$$\lim_{h\to 0} f(x_0 - h) = y_0^- \quad , \quad \lim_{h\to 0} f(x_0 + h) = y_0^+ . \tag{6.1b}$$

Diese Funktion soll mittels der Basisfunktionen $\cos(nx)$, $\sin(nx)$, $n = 0, 1, \ldots$ im **quadratischen Mittel** approximiert werden. (Die Wahl dieser Basisfunktionen liegt auf der Hand, wenn f (2π)-periodisch ist.) Mit anderen Worten, die Ersatzfunktion $g_n(x)$ soll die Bedingung

$$\|g_n(x) - f(x)\|_2^2 := \int_{-\pi}^{\pi} (g_n(x) - f(x))^2 \, \mathrm{d}x = \min!$$

erfüllen, wenn $g_n(x)$ durch

$$g_n(x) = \frac{1}{2} a_0 + \sum_{k=1}^{n} (a_k \cos(kx) + b_k \sin(kx)) \tag{6.2}$$

definiert ist. $g_n(x)$ heißt FOURIER-*Polynom* und die Koeffizienten a_k sind gesucht!

Zunächst ist festzuhalten, dass die angegebenen Basisfunktionen im Intervall $[-\pi + a, \pi + a]$ $\forall a \in \mathbb{R}$ paarweise orthogonal sind,

$$\begin{aligned}
&\int_{-\pi+a}^{\pi+a} \cos(jx)\cos(kx)\mathrm{d}x = \begin{cases} 0 & j \neq k \\ 2\pi & j = k = 0 \\ \pi & j = k > 0 \end{cases} \quad \to \text{Faktor } \tfrac{1}{2} \text{ bei } a_0 \\
&\int_{-\pi+a}^{\pi+a} \sin(jx)\sin(kx)\mathrm{d}x = \begin{cases} 0 & j \neq k \\ \pi & j = k \end{cases} \\
&\int_{-\pi+a}^{\pi+a} \cos(jx)\sin(kx)\mathrm{d}x = 0 \quad \text{immer!}
\end{aligned} \tag{6.3}$$

(Beweis durch Nachrechnen!)

Die Koeffizienten a_k, b_k des FOURIER-Polynoms ergeben sich durch Minimierung der quadratischen Funktion (im Weiteren sei o.E.d.A. $a = 0$)

$$\|g_n(x) - f(x)\|^2 = F\left(a_0, \ldots, a_n, b_1, \ldots, b_n\right),$$

und die dazugehörigen Koeffizienten aus

$$\frac{\partial F}{\partial a_0} = \cdots = \frac{\partial F}{\partial b_n} \stackrel{!}{=} 0.$$

Diese Vorgehensweise entspricht der Bestimmung der Parameter von Ausgleichsproblemen (vgl. Kap 5).

Zunächst berechenen wir die Funktion F durch Ausmultiplizieren,

$$\begin{aligned}
F &= \|g_n(x) - f(x)\|^2 \\
&= \int_{-\pi}^{\pi} \left[\frac{1}{2}a_0 + \sum_{k=1}^{n} (a_k \cos(kx) + b_k \sin(kx)) - f(x)\right]^2 \mathrm{d}x \\
&= \int_{-\pi}^{+\pi} \left(\frac{1}{2}a_0 - f(x)\right)^2 \mathrm{d}x + 2\sum_{k=1}^{n} \int_{-\pi}^{+\pi} \left(\frac{1}{2}a_0 - f(x)\right)(a_k \cos(kx) + b_k \sin(kx))\,\mathrm{d}x + \\
&\qquad + \sum_{k=1}^{n}\sum_{j=1}^{n} \int_{-\pi}^{\pi} (a_k \cos(kx) + b_k \sin(kx))\,(a_j \cos(jx) + b_j \sin(jx))\,\mathrm{d}x \\
&= \frac{\pi}{2}a_0^2 - a_0 \int_{-\pi}^{\pi} f(x)\mathrm{d}x + \int_{-\pi}^{\pi} (f(x))^2\,\mathrm{d}x - \\
&\qquad -2\sum_{k=1}^{n} a_k \int_{-\pi}^{\pi} f(x)\cos(kx)\mathrm{d}x - 2\sum_{k=1}^{n} b_k \int_{-\pi}^{\pi} f(x)\sin(kx)\mathrm{d}x + \pi \sum_{k=1}^{n} \left(a_k^2 + b_k^2\right),
\end{aligned}$$

und danach die partiellen Ableitungen

$$\begin{aligned}
\frac{\partial F}{\partial a_0} &= \pi a_0 - \int_{-\pi}^{\pi} f(x)\mathrm{d}x \stackrel{!}{=} 0 \\
\frac{\partial F}{\partial a_k} &= -2\int_{-\pi}^{\pi} f(x)\cos(kx)\mathrm{d}x + 2\pi a_k \stackrel{!}{=} 0 \qquad k = 1, \ldots, n \\
\frac{\partial F}{\partial b_k} &= -2\int_{-\pi}^{\pi} f(x)\sin(kx)\mathrm{d}x + 2\pi b_k \stackrel{!}{=} 0 \qquad k = 1, \ldots, n.
\end{aligned}$$

Insgesamt ergibt sich also für die gesuchten Koeffizienten

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)\mathrm{d}x \qquad k = \mathbf{0}, \ldots, n$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)\mathrm{d}x \qquad k = 1, \ldots, n \tag{6.4}$$

Mit diesen FOURIER-Koeffizienten definiert man dann die FOURIER-Reihe durch

$$\lim_{n\to\infty} g_n(x) =: g(x)$$

6.2 Satz. *Sei $f(x)$ 2π-periodisch, stückweise stetig mit stückweise stetiger erster Ableitung. Dann konvergiert die* FOURIER*-Reihe $g(x_0)$*

(i) gegen $f(x_0)$, falls $f(x)$ stetig bei x_0 ist,

(ii) gegen $\frac{1}{2}(y_o^+ + y_0^-)$, falls x_0 eine Sprungstelle ist und y_o^+, y_0^- wie in (6.1b) definiert sind.

6.3 Beispiel (2π-periodische "Dachfunktion").

$$f(x) = |x| \qquad -\pi \le x \le \pi$$

$f(x)$ ist eine stetige, "gerade" Funktion ($f(x) = f(-x)$), demzufolge sind alle Koeffizienten der `Sinus`-Terme, b_k, a priori gleich Null. Die restlichen Koeffizienten (der `Cosinus`-Terme) berechnen sich zu

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} |x|\mathrm{d}x = \frac{2}{\pi}\int_{0}^{\pi} x\mathrm{d}x = \pi$$

$$a_k = \frac{2}{\pi}\int_{0}^{\pi} x\cos(kx)\mathrm{d}x = \frac{2}{\pi k^2}\cos(kx)\Big|_0^{\pi} = \frac{2}{\pi k^2}\left[(-1)^k - 1\right],$$

und die FOURIER-Reihe lautet

$$g(x) = \frac{\pi}{2} - \frac{4}{\pi}\left(\frac{\cos(x)}{1^2} + \frac{\cos(3x)}{3^2} + \frac{\cos(5x)}{5^2} + \dots\right).$$

Ein Nebenprodukt dieser Reihe lässt sich durch Auswertung bei $x = 0$ angeben:

$$f(0) = 0 \overset{\text{Satz } 6.2}{=} g(0) = \frac{\pi}{2} - \frac{4}{\pi}\sum_{\substack{i=1,\\ i \text{ ungerade}}}^{n} \frac{1}{i^2},$$

d.h

$$\frac{1}{1} + \frac{1}{3^2} + \frac{1}{5^2} + \dots = \frac{\pi^2}{8}$$

Folgende Abbildung 6.1 zeigt das entsprechende FOURIER-Polynom zu $n = 4$ und $n = 16$. Man sieht die sehr gute Annäherung an die Ausgangsfunktion.

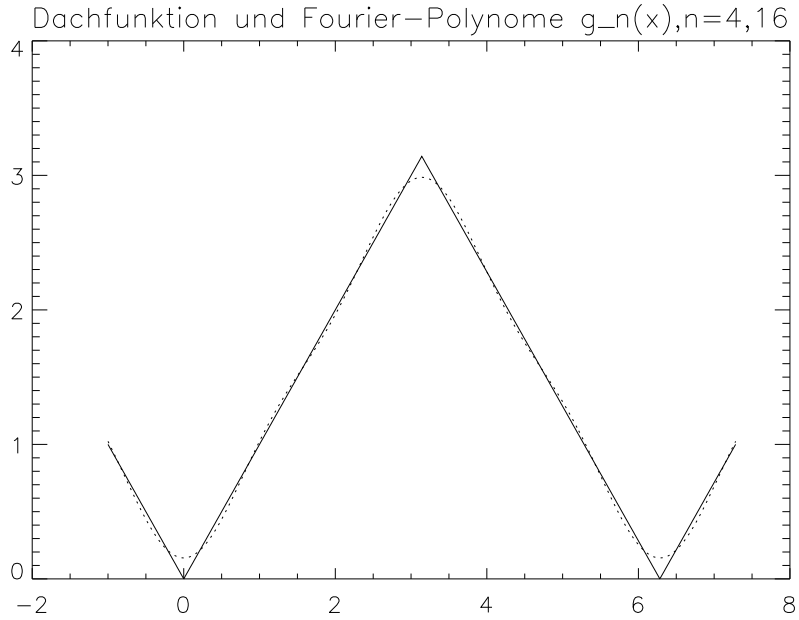


Abbildung 6.1: Die 2π-periodischen Dachfunktion $|x|$ im Vergleich mit den dazugehörigen FOURIER-Polynomen g_4 (gepunktet) und g_{16} (gestrichelt).

6.4 Beispiel (2π-periodische x^2-Funktion). Eine analoge Behandlung der 2π-periodischen Funktion

$$f(x) = x^2 \qquad x = 0 \dots 2\pi, \quad \text{periodisch},$$

die eine Sprungstelle bei 2π aufweist (d.h., die Reihe soll bei diesem Wert gegen $\frac{1}{2}(0 + (2\pi)^2) = 2\pi^2$ konvergieren), zeigt die Abbildung 6.2.

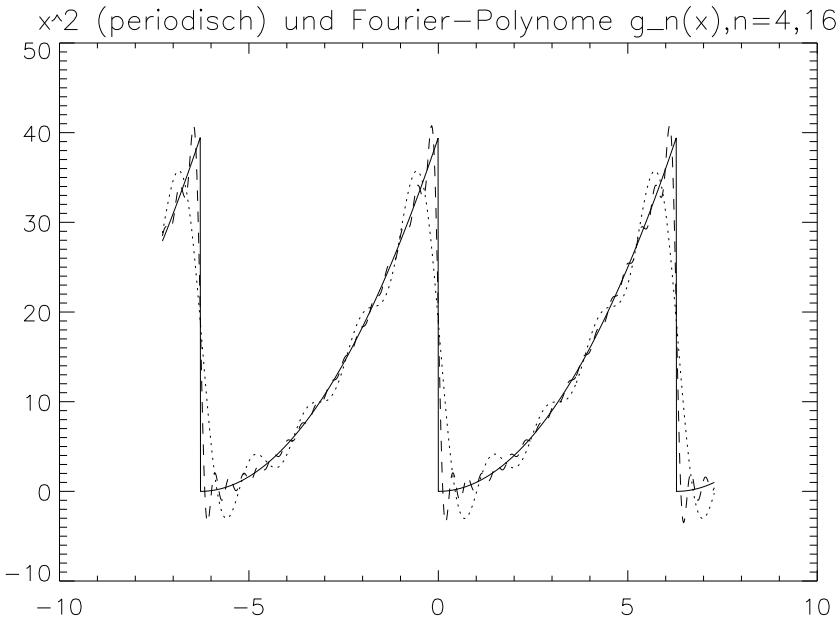


Abbildung 6.2: FOURIER-Approximation der 2π-periodischen Funktion x^2 mittels FOURIER-Polynomen g_4 (gepunktet) und g_{16} (gestrichelt). Man beachte die vorhergesagte Konvergenz gegen $2\pi^2$ bei $x = 2\pi$ (periodisch).

Die FOURIER-Koeffizienten entsprechend (6.4) müssen im Allgemeinen numerisch berechnet werden, da für die meisten in Praxis auftretenden Funktionen $f(x)$ keine analytische Lösung der Integrale möglich ist. Das Mittel der Wahl ist hier die sog. Traprezintegration (Kap. 7), da diese (abgesehen von der einfachen Formulierung) *bei periodischen Funktionen* ein äusserst günstiges Fehlerverhalten hat.

Nebenbemerkung: Kurz gesagt, lautet die entsprechende "Integrationsformel" für die Trapezintegration über ein Intervall $[a, b]$

$$\int_a^b f(x)\mathrm{d}x \approx \frac{1}{2}\left(f(a) + f(b)\right)(b-a),$$

d.h. die Fläche unter der Funktion ($\hat{=}$ Integral) wird durch eine Trapezfläche approximiert. Aufspaltung des Intervalls in N *äquidistante* Subintervalle der Breite h und Anwendung der Trapezintegration ergibt dann

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x &= \int_a^{a+h} f(x)\mathrm{d}x + \int_{a+h}^{a+2h} f(x)\mathrm{d}x + \cdots + \int_{b-h}^{b} f(x)\mathrm{d}x \\ &= \frac{1}{2}\left(f(a) + f(a+h)\right)h + \frac{1}{2}\left(f(a+h) + f(a+2h)\right)h + \cdots + \frac{1}{2}\left(f(b-h) + f(b)\right)h \\ &= \frac{h}{2}\left(f(a) + 2f(a+h) + \cdots + 2f(b-h) + f(b)\right).\end{aligned}$$

Die sog. "Integrationsgewichte" ($N+1$ bei N Intervallen) lauten also $(1, 2, 2, \ldots, 1)$.

Mit dieser "Formel" und unter tatsächlicher Verwendung von $N+1$ äquidistanten "Stützstellen" auf einem gesamten Intervall der Breite 2π, d.h. N Subintervallen der Breite h

$$h = \frac{2\pi}{N}, \quad x_j = hj = \frac{2\pi}{N}j, \quad j = 0, N \tag{6.5}$$

ergibt sich (man beachte, dass wir nun das Intervall $[0, 2\pi]$ statt $[-\pi, \pi]$ verwenden)

$$\Rightarrow \quad a_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos kx \mathrm{d}x \qquad (\rightarrow \text{Trapezregel})$$
$$\approx \frac{1}{\pi} \frac{h}{2} \left(f(x_0) \cos kx_0 + 2 \sum_{j=1}^{N-1} f(x_j) \cos kx_j + f(x_N) \cos kx_N \right).$$

Das Integral wird also durch eine entsprechende Summe ersetzt! Berücksichtigt man nun die Periodizität der Funktion,

$$f(x_0) = f(x_N)$$
$$\cos kx_0 = \cos kx_N,$$

dann haben der erste und der letzte Summand die gleichen Werte, und mit $h = \frac{2\pi}{N}$ folgt

$$a_k^* = \frac{2}{N} \sum_{j=1}^{N} f(x_j) \cos kx_j \qquad (k = 0, 1, \dots) \tag{6.6}$$

wobei sich aufgrund der Periodizität die Summe jetzt bis $j = N$ erstreckt und der Stern den *Näherungswert* bezeichnet. Für die genäherten Koeffizienten b_k^* folgt analog

$$b_k^* = \frac{2}{N} \sum_{j=1}^{N} f(x_j) \sin kx_j \qquad (k = 1, 2, \dots)$$

6.5 Satz. *Es sei $N = 2n$ mit $n \in \mathbb{N}$. Das* FOURIER*-Polynom*

$$g_n^*(x) = \frac{1}{2} a_0^* + \sum_{k=1}^{n-1} (a_k^* \cos kx + b_k^* \sin kx) + \frac{1}{2} a_n^* \cos nx \tag{6.7}$$

mit den genäherten Koeffizienten entsprechend Gln. 6.6 ist das eindeutige interpolierende[2] FOURIER*-Polynom zu den Stützstellen x_j, $j = 1, N$. (N Stützstellen, da $f(x_0) = f(x_N)$, und N Koeffizienten $a_0^* ... a_n^*, b_1^* ... b_{n-1}^*$). Insbesondere gilt*

$$g_n^*(x_j) = f(x_j).$$

Der Beweis dieses Satzes basiert auf den sog. "diskreten" Orthogonalitätsrelationen der Basisfunktionen auf einem äquidistantem Gitter und wird hier nicht vorgeführt.
Bemerkung: Falls die Funktion eine Sprungstelle bei x_j hat, gilt obiger Satz unter der Voraussetzung, dass man $f(x_j) = \frac{1}{2} \left(y_0^+ + y_0^- \right)$ setzt.

6.6 Satz. *Für $m < n$ approximiert $g_m^*(x)$ die Funktion $f(x)$ im* diskreten *quadratischen Mittel dergestalt, dass*

$$F = \sum_{j=1}^{N} (g_m^*(x_j) - f(x_j))^2$$

minimal wird. Dabei ist

$$g_m^*(x) = \frac{1}{2} a_0^* + \sum_{k=1}^{m} a_k^* \cos kx + b_k^* \sin kx \tag{6.8}$$

[2] vgl. Kap. 2

6.7 Beweis. *Der Beweis dieses Satzes erfolgt dadurch, dass wir durch Minimierung der Funktion F die entsprechenden Koeffizienten (im weiteren mit α_k, β_k bezeichnet) explizit berechnen und dadurch zeigen, dass sie mit den Werten a_k^*, b_k^* aus Gl. 6.6 übereinstimmen.*

$$F = \sum_{j=1}^{N} \left(\frac{1}{2}\alpha_0 + \left(\sum_{k=1}^{m} \alpha_k \cos kx_j + \beta_k \sin kx_j \right) - f(x_j) \right)^2 = min!$$

$$\begin{aligned} \Rightarrow \quad \frac{\partial F}{\partial \alpha_0} &= \sum_{j=1}^{N} \left(\frac{1}{2}\alpha_0 + \sum_{k=1}^{m} (\alpha_k \cos kx_j + \beta_k \sin kx_j) - f(x_j) \right) \\ &= \frac{N}{2}\alpha_0 + \sum_{k=1}^{m} \left(\alpha_k \sum_{j=1}^{N} \cos kx_j + \beta_k \sum_{j=1}^{N} \sin kx_j \right) - \sum_{j=1}^{N} f(x_j) \stackrel{!}{=} 0 \end{aligned}$$

Mit Hilfe einer Nebenrechnung zeigen wir zunächst, dass die Summen über die `Sinus/Cosinus`*-Werte ein einfaches Resultat zeitigen, nämlich*

$$\sum_{j=1}^{N} \cos kx_j = \begin{cases} 0 & falls\ \frac{k}{N} \notin \mathbb{Z} \\ N & falls\ \frac{k}{N} \in \mathbb{Z} \end{cases} \tag{6.9a}$$

$$\sum_{j=1}^{N} \sin kx_j = 0 \quad \forall k \in \mathbb{Z} \tag{6.9b}$$

Dazu berechnen wir eine komplexe Summe S, die wir mit Hilfe der Euler*schen Formel weiter bearbeiten.*

$$S := \sum_{j=1}^{N} (\cos kx_j + \mathrm{i} \sin kx_j) = \sum_{j=1}^{N} \mathrm{e}^{\mathrm{i}kx_j} = \sum_{j=1}^{N} \left(\mathrm{e}^{\mathrm{i}kh}\right)^j = \sum_{j=1}^{N} \left(\mathrm{e}^{2\pi \mathrm{i} \frac{k}{N}}\right)^j$$

wobei die letzten beiden Identitäten aus $x_j = h \cdot j = \frac{2\pi}{N} j$ resultieren.

(a) Sei zunächst $\frac{k}{N} = z \in \mathbb{Z}$. Wegen $e^{2\pi \mathrm{i} z} = 1$ (der `Cosinus` *ist 1, der* `Sinus` *ist 0) besteht die Summe aus N Summanden der Größe 1^j, d.h. die Summe ergibt sich zu $\sum_{j=1}^{N} \left(e^{\mathrm{i}kh}\right)^j = N$.*

(b) Nun sei $\frac{k}{N} \notin \mathbb{Z}$. Unter Verwendung der Summenregel für geometrische Reihen,

$$\sum_{j=1}^{N} aq^{j-1} = a\frac{q^N - 1}{q - 1}$$

finden wir in diesem Fall

$$\sum_{j=1}^{N} \left(\mathrm{e}^{\mathrm{i}hk}\right)^j = \sum_{j=1}^{N} e^{\mathrm{i}kh} \left(\mathrm{e}^{\mathrm{i}kh}\right)^{j-1} = \mathrm{e}^{\mathrm{i}kh} \frac{\mathrm{e}^{\mathrm{i}khN} - 1}{\mathrm{e}^{\mathrm{i}kh} - 1} = e^{\mathrm{i}kh} \frac{\overbrace{e^{2\pi \mathrm{i}k}}^{=1} - 1}{e^{\mathrm{i}kh} - 1} = 0$$

da der Nenner aufgrund $\frac{k}{N} \notin \mathbb{Z}$ ungleich Null ist.

Insgesamt ergibt sich also für die komplexe Summe S

$$\begin{aligned} S &= 0 + 0\mathrm{i} \quad \frac{k}{N} \notin \mathbb{Z} \\ S &= N + 0\mathrm{i} \quad \frac{k}{N} \in \mathbb{Z} \end{aligned}$$

und daraus mittels Koeffizientenvergleich die Behauptung der Nebenrechnung (6.9 a,b).

Da in unserem Fall laut Voraussetzung $k = 1, \dots m \leq \frac{N}{2}$ *ist, d.h.* $\frac{k}{N} \notin \mathbb{Z}$*, resultieren die Summen über die* `Sinus/Cosinus`*-Werte beide zu Null, und wir erhalten für die partielle Ableitung bzgl.* α_0

$$\frac{\partial F}{\partial \alpha_0} = \frac{N}{2}\alpha_0 - \sum_{j=1}^{N} f(x_j) \stackrel{!}{=} 0,$$

und der gesuchte Koeffizient ergibt sich zu

$$\alpha_0 = \frac{2}{N}\sum_{j=1}^{N} f(x_j) = a_0^*!,$$

d.h. entspricht genau dem numerischen Näherungswert. Auf analoge Weise finden wir die Entsprechungen für die übrigen Koeffizienten

$$\frac{\partial F}{\partial \alpha_l} = 0 \quad \rightarrow \qquad \alpha_l = \frac{2}{N}\sum_{j=1}^{N} f(x_j)\cos lj = a_l^*!$$

$$\frac{\partial F}{\partial \beta_l} = 0 \quad \rightarrow \qquad \beta_l = \frac{2}{N}\sum_{j=1}^{N} f(x_j)\sin lj = b_l^*!$$

und damit ist der Satz bewiesen. □

Beachte: Die Berechnung der N Koeffizienten (N Stützstellen) des interpolierenden FOURIER-Polynoms erfordert $\mathcal{O}(N^2)$ Operationen!

6.8 Beispiel (Interpolierendes und approximierendes FOURIER-Polynom). In den folgenden beiden Abbildungen vergleichen wir die (2π)-periodische Dachfunktion aus Beispiel 6.3 mit dem eindeutigen *interpolierenden* FOURIER-Polynom $g_4^*(x)$ für 8 Stützstellen und mit dem entsprechenden *approximierenden* Polynom $g_2^*(x)$ (bgzl. der gleichen Stützstellen).

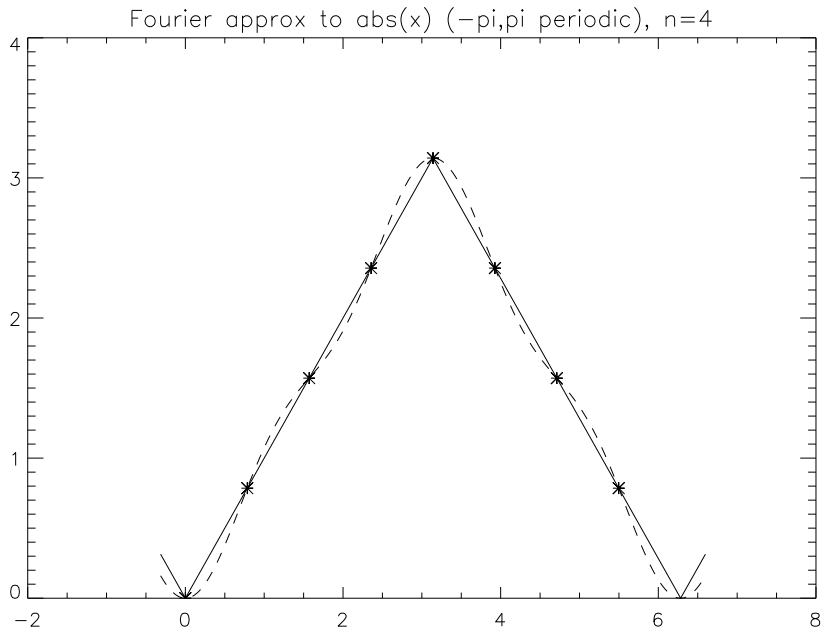


Abbildung 6.3: Dachfunktion und *interpolierendes* FOURIER-Polynom $g_4^*(x)$ bei acht Stützstellen. Man beachte, dass $g_4^*(x_j) = f(x_j)$ für alle Stützstellen (mit Sternen markiert). Man vergleiche mit dem entsprechenden Polynom $g_4(x)$ von Abb. 6.1, das aus einer *analytischen Berechnung der Koeffizienten* resultierte.

Die FOURIER-Koeffizienten, die aus der Trapezintegration folgen, lauten

$$b_i = 0, i = 1, \dots, 3 \quad \text{und} \quad a_0 = \pi, a_1 = -1.34076, a_2 = 0., a_3 = -0.230028, a_4 = 0.$$

Damit resultiert das interpolierende Polynom

$$g_4^*(x) = \frac{\pi}{2} + a_1 \cos(x) + a_3 \cos(3x)$$

und das approximierende Polynom

$$g_2^*(x) = \frac{\pi}{2} + a_1 \cos(x).$$

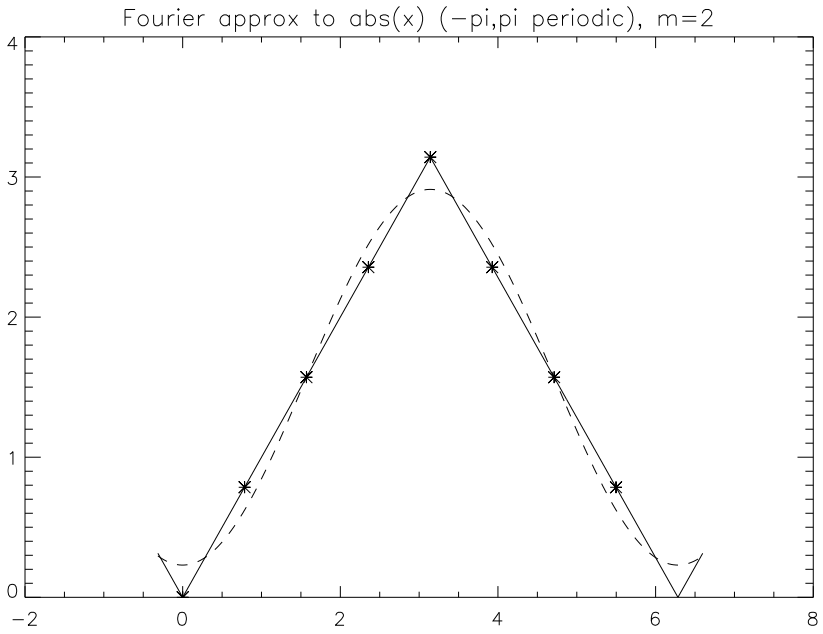


Abbildung 6.4: Dachfunktion und *approximierendes* FOURIER-Polynom $g_2^*(x)$ bei acht Stützstellen. Die quadratische Abweichung wird minimiert, aber $g_2^*(x_j) \neq f(x_j)$ für die meisten Stützstellen.

6.2 FOURIER-Transformation

Im vorhergehenden Kapitel hatten wir gesehen, dass die Berechnung der N FOURIER-Koeffizienten $a_k, k = 0, N/2$ und $b_k, k = 1, N/2-1$ $\mathcal{O}(N^2)$ Operationen erfordert (und noch dazu "teure", aufgrund der auftretenden `Cosinus`- und `Sinus`-Terme).

Es existiert allerdings ein Algorithmus, der diese Aufgabe in $\mathcal{O}(N \log_2(N))$ Operationen bewältigt, die sog. **Fast** FOURIER **Transformation** oder FFT. Dieser Algorithmus ist bei einer großen Anzahl von Stützstellen geradezu "lebenswichtig".

Die diesem Algorithmus zugrunde liegende FOURIER- Transformation (FT) ist nicht nur im Rahmen der Berechung von approximierenden/interpolierenden Polynomen relevant, sondern

- von generellem Interesse; einer der wichtigsten Bereiche ist die *Signalverarbeitung*. So sind z.B. digitales Audio und Video ohne (F)FT undenkbar.
- Auch die *Analyse* von (z.B. physikalischen) Daten erfordert oftmals diese Technik. Beispiele hierfür sind die Analyse von *zeitlichen Signalen*, wobei mit Hilfe der (F)FT die darin enthaltenen Frequenzen und deren Leistung (das sog. "Powerspektrum") sofort bestimmt werden können. Darüber hinaus erlaubt die FFT eine äussert schnelle Durchführung wichtiger Operationen, z.B. die *Faltung* (vgl. Kap. 6.5.2) zweier Funktionen

$$(x \circ y)(t) := \int_{-\infty}^{\infty} x(\tau) y(t-\tau) \mathrm{d}\tau,$$

(die u.a. dann zum Tragen kommt, wenn ein originales Signal x durch einen Detektor "verfälscht" wird), und die Korrelation zweier Funktionen

$$\mathrm{corr}(x,y) := \int_{-\infty}^{\infty} x(\tau+t)y(\tau)\mathrm{d}\tau.$$

- FOURIER-Transformationen bilden die Basis für alternative Lösungsmethoden von partiellen Differentialgleichungen, und schliesslich
- gehören sie zum Grundkonzept vieler (physikalischer) Disziplinen; als Beispiel sei hier der Übergang von Orts- nach Impulsraum (und umgekehrt) in der Quantenmechanik genannt.

6.2.1 Grundlagen

6.9 Definition (FOURIER-Transformation). *Für beliebige, komplexwertige Funktionen $\tilde{f}$ und $\tilde{F}$ definieren wir die Vorwärts- und Rückwärts-*FOURIER*-Transformation durch*

$$\tilde{F}(k') = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \tilde{f}(x)\mathrm{e}^{-\mathrm{i}k'x}\mathrm{d}x \qquad \textit{(Vorwärtstransformation)}$$

$$\tilde{f}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \tilde{F}(k')\mathrm{e}^{\mathrm{i}k'x}\mathrm{d}k' \qquad \textit{(Rückwärtstransformation)}$$

Auf Grund der EULERschen Formel $\mathrm{e}^{-\mathrm{i}k'x} = \cos k'x + \mathrm{i}\sin k'x$ entspricht diese Transformation im Wesentlichen der Definition der FOURIER-Koeffizienten (6.4), und zwar in der Kombination $a_k + \mathrm{i}b_k$. Damit lässt sich die (F)FT als Mittel zu ihrer Berechnung verwenden. Auf diesen Aspekt werden wir in Kapitel 6.3 zurückkommen.

Manchmal wird die FOURIER-Transformation (Def. 6.9) auch umgekehrt definiert, d.h. die Vorwärtstransformation über $\mathrm{e}^{\mathrm{i}k'x}$ und die Rückwärtstransformation über $\mathrm{e}^{-\mathrm{i}k'x}$. Solange man allerdings die jeweils verwendeten Definition konsequent "durchhält", ist dieser Unterschied irrelevant.

Zunächst wollen wir beweisen, dass mit obiger Definition der *Vorwärts-* und *Rückwärts*-Transformation das Gewünschte erzielt wird, dass nämlich eine Hintereinanderausführung von beiden Operationen die Identität ergibt.

6.10 Beweis. *Zu zeigen ist also, dass "Rückwärts(Vorwärts) = Identität"*

$$\begin{aligned}
\frac{1}{\sqrt{2\pi}} \int \mathrm{e}^{\mathrm{i}k'x}\tilde{F}(k')\mathrm{d}k' =& \frac{1}{2\pi} \int \mathrm{d}k'\mathrm{e}^{\mathrm{i}k'x} \int \mathrm{d}\xi \tilde{f}(\xi)\mathrm{e}^{-\mathrm{i}k'\xi} \\
& \textit{(Vertauschung der Integrationsreihenfolge)} \\
=& \frac{1}{2\pi} \int \mathrm{d}\xi \tilde{f}(\xi) \int \mathrm{d}k'\mathrm{e}^{\mathrm{i}k'(x-\xi)} \\
=& \int \mathrm{d}\xi \tilde{f}(\xi)\delta(x-\xi) \stackrel{!}{=} \tilde{f}(x)
\end{aligned}$$

wenn wir berücksichtigen, dass $\frac{1}{2\pi} \int \mathrm{d}k'\mathrm{e}^{\mathrm{i}k'(x-\xi)} = \delta(x-\xi)$ *die* DIRACsche *Delta-Funktion ist.*

Mit $k' = 2\pi k$ gilt dann auch

$$\int \mathrm{d}k e^{2\pi i k(x-\xi)} = \delta(x-\xi) \quad \leftarrow \quad \text{ohne "}2\pi\text{"}$$

und wir führen eine alternative Formulierung der FOURIER-Transformation ein;

6.11 Definition (FOURIER-Transformation, alternative Formulierung).

$$F(k) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}\mathrm{d}x, \qquad f(x) = \int_{-\infty}^{\infty} F(k)e^{2\pi ikx}\mathrm{d}k \tag{6.10}$$

Der Vorteil dieser Definition ist es, *dass alle Normierungsfaktoren verschwinden*!

6.12 Beispiel.

(a) $x \hat{=}$ Ort, $\Rightarrow\ k \hat{=} \dfrac{1}{\text{Ort}}$ = inverse Wellenlänge ($k' = 2\pi k \hat{=}$ Wellenzahl)

(b) $x \hat{=}$ Zeit $\Rightarrow\ k \hat{=} \dfrac{1}{\text{Zeit}}$ = Frequenz ($k' = 2\pi k \hat{=}$ Kreisfrequenz)

Im weiteren leiten wir zwei

Wichtige Eigenschaften der FOURIERtransformation ab. Real- und Imaginärteil der betreffendnen Funktionen seien hier mit den Indizes "R" und "I" gekennzeichnet.

$$\begin{aligned}
F(k) &= \int \mathrm{d}x f(x)e^{-2\pi \mathrm{i}kx} \\
&= \int \mathrm{d}x \left(f_R + \mathrm{i}f_I\right)\left(\cos(-2\pi kx) + \mathrm{i}\sin(-2\pi kx)\right) \\
&= \int \mathrm{d}x f_R \left(\cos(2\pi kx) - \mathrm{i}\sin(2\pi kx)\right) + \int \mathrm{d}x f_I \left(\mathrm{i}\cos(2\pi kx) + \sin(2\pi kx)\right) \\
F(-k) &= \int \mathrm{d}x f_R \left(\cos(2\pi kx) + \mathrm{i}\sin(2\pi kx)\right) + \int \mathrm{d}x f_I \left(\mathrm{i}\cos(2\pi kx) - \sin(2\pi kx)\right)
\end{aligned}$$

Falls also die Funktion (oder das "Signal") $f(x)$ reell ist, $f_I = 0$, dann gilt

$$F(-k) = \left(F(k)\right)^*$$

($*$: komplexe Konjugation). Falls $f(x)$ andererseits imaginär ist, $f_R = 0$, dann gilt

$$F(-k) = -\left(F(k)\right)^*.$$

Weitere Eigenschaften der FOURIER-Transformation werden in der Ergänzung (6.5.1) angeführt.

6.2.2 Diskrete Datenpunkte: Die NYQUIST-Frequenz

Ohne Einschränkung der Allgemeinheit identifizieren wir nun die (allgemeinen) Variablen x, k mit den physikalischen Variablen t, f, d.h.

$$\begin{aligned}
x &\to t \qquad \text{Zeit} \\
k &\to f \qquad \text{Frequenz}
\end{aligned}$$

und die Funktionen f, F werden im Weiteren als Signal h und Transformierte des Signals H,

$$\begin{aligned}
f(x) &\to h(t) \qquad \text{Signal} \\
F(k) &\to H(f) \qquad \text{FOURIER-Transformierte/-Komponente des Signals}
\end{aligned}$$

bezeichnet. Die physikalische Bedeutung von $H(f)$ wird später ersichtlich werden.

Sei nun das Signal $h(t)$ auf äquidistanten Stützstellen t_n vorgegeben (z.B. gemessen),

$$h_n = h(n \cdot \Delta) \qquad , \quad n = \ldots, -3, -2, -1, 0, 1, 2, 3 \ldots$$

$1/\Delta$ bezeichnet man als "Abtastrate" (oder auf *Denglisch* "Sampling-Rate"). Sie gibt die Anzahl der abgetasteten Datenpunkte pro Zeiteinheit an.

6.13 Beispiel. $\Delta = 5$s, d.h. das Signal werde alle 5 Sekunden abgetastet. Die Abtastrate ist damit $\frac{1}{\Delta} = 0.2\text{s}^{-1}$, es werden 0.2 Datenpunkte/Sekunde gemessen.

6.14 Definition. *Für jede Abtastrate existiert die sogenannte* NYQUIST*-Frequenz (kritische Frequenz)*

$$f_c = \frac{1}{2\Delta}. \tag{6.11}$$

6.15 Beispiel. Man betrachte eine sinusförmiges Signal mit Frequenz f und taste es mit der daraus folgenden kritischen Abtastrate $\Delta^{-1} = 2f$ ab. Damit hat das Signal gerade die kritische Frequenz $f = f_c$, und für *beliebige* Zeiten t lautet dieses

$$h(t) = \sin(2\pi f t) := \sin(2\pi f_c t) = \sin\left(\frac{\pi t}{\Delta}\right).$$

Zu den *tatsächlich gemessenen* Zeitpunkten $t_n = n \cdot \Delta$ ergibt sich

$$h(t_n) = h_n = \sin(\pi n)$$

bzw. bei willkürlicher Anfangsphase

$$h_n = \sin(\pi n + \varphi_0) \qquad n = 0, 1, 2, \ldots.$$

Falls nun
der erste gemessene Datenpunkt beim Maximum liegt (was $\varphi_0 = \frac{\pi}{2}$ impliziert), dann liegt
der zweite Datenpunkt beim Minimum ($\sin(\pi + \frac{\pi}{2})$),
der dritte Datenpunkt wieder beim Maximum ($\sin(2\pi + \frac{\pi}{2})$) und so fort.

> Das kritische Abtasten einer Sinuswelle ergibt also genau zwei Datenpunkte/Zyklus

Damit wird klar, warum man diese Frequenz (bzw. die Abtastrate) als "kritisch" bezeichnet. Falls einerseits $\varphi_0 = 0$, messen wir zu allen Zeiten die Nulldurchgänge, d.h. $h(n) = 0 \ \forall \ n$. Falls andererseits $\varphi_0 \neq 0$, messen wir immer die gleichen (absoluten) Amplituden. In beiden Fällen (da der Anfangszeitpunkt der Messung willkürlich ist und wir keine Information über φ_0 besitzen) ist bei dieser Abtastrate keine Aussage über die tatsächliche Amplitude des Signals möglich.

6.2.3 Das Sampling-Theorem und Aliasing

6.16 Satz (Sampling-Theorem). *Falls die Funktion $h(t)$, abgetastet mit Δ^{-1}, bandweitenlimitiert ist, so dass*

$$H(f) = 0 \qquad \forall \quad |f| \geq f_c = \frac{1}{2\Delta} \tag{6.12}$$

so ist die Funktion vollständig *durch die Datenpunkte h_n bestimmt, d.h. das Signal ist vollständig* rekonstruierbar*, und es gilt*

$$h(t) = \Delta \cdot \sum_{n=-\infty}^{\infty} h_n \frac{\sin\left(2\pi f_c (t - n\Delta)\right)}{\pi(t - n\Delta)} \tag{6.13}$$

Andersherum ausgedrückt: Falls man die Bandbreite f_{max} (sodass bei f_{max} gerade kein Signal mehr vorhanden ist) eines Signals kennt, kann man dieses Signal *ohne Informationsverlust* durch das Abtasten mit dem *Doppelten* dieser Frequenz, $\frac{1}{\Delta} = 2f_{max}$, aufnehmen!

Man beachte, dass für eine vollständige Rekonstruktion *sehr* lange zu messen ist!

6.17 Beispiel (Rekonstruktion eines Signales; unterschiedliche Abtastraten).
Wir betrachten ein Signal $h(t) = \sin(2\pi f t) \cdot \sin(2\pi\,(0.1f)\,t)$ mit $f = 1$, d.h. einen `SINUS` mit Frequenz f, der mit einer zehnfach kleineren Frequenz moduliert wird. O.E.d.A. sei die Phasenverschiebung gleich Null, d.h. zu Beginn der Messung ($t_0 = 0$) ereigne sich gerade der Nulldurchgang, $h(0) = 0$. Vor und nach der Messung sei kein Signal vorhanden, so dass wir für alle Zeitpunkte ausserhalb des Messbereiches $h_n = 0$ in obiger Summe setzen können.

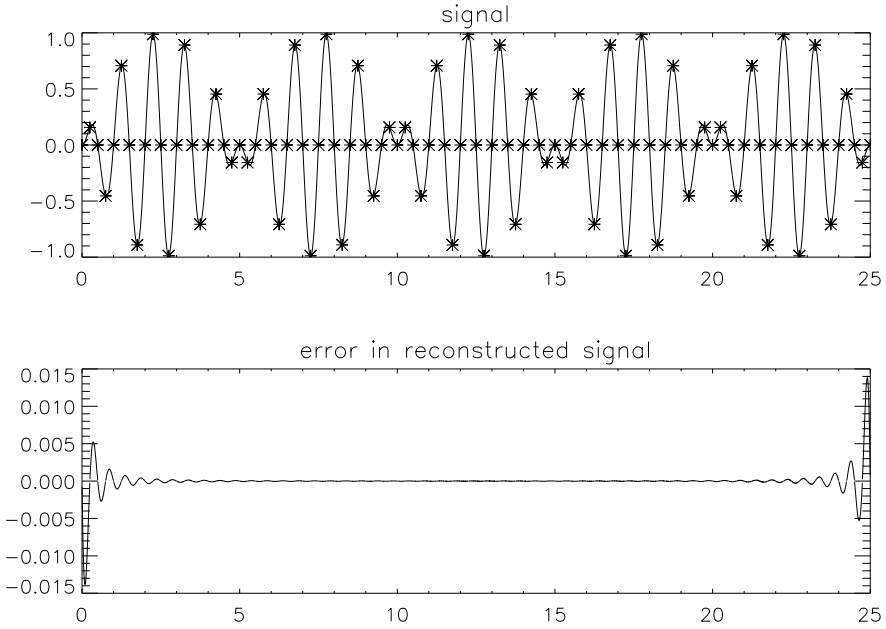


Abbildung 6.5: Oben: Tatsächliches (durchgezogene Linie) und abgetastetes Signal (Sterne) aus Beispiel 6.17, Fall (a). Abtastrate entsprechend einer Zeitkonstanten $\Delta = 0.25$, d.h. doppelt so schnell wie kritisch. Unten: Relativer Fehler zwischen rekonstruiertem und tatsächlichem Signal.

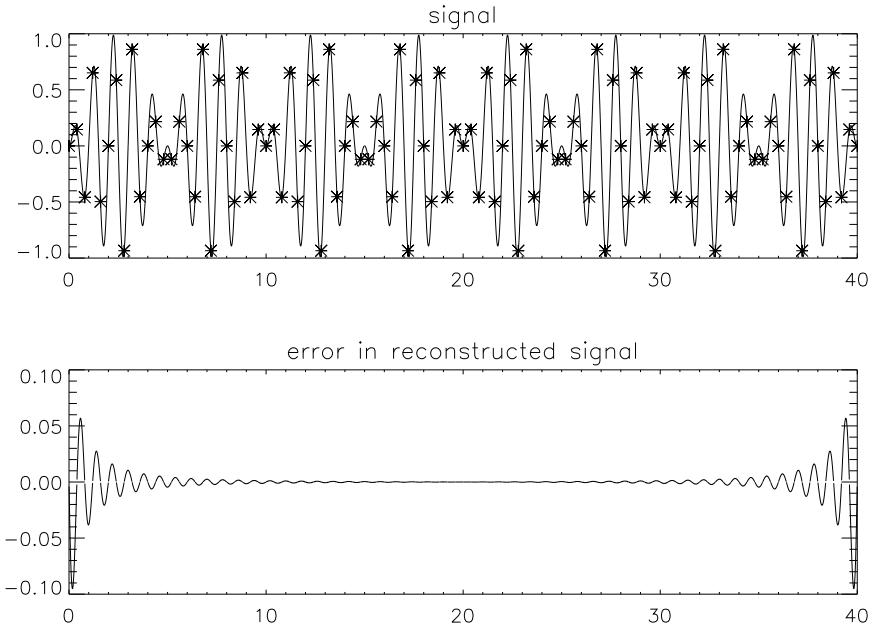


Abbildung 6.6: Wie Abb. 6.5, jedoch Fall (b). Zeitkonstante $\Delta = 0.4$ (20% schneller als kritisch).

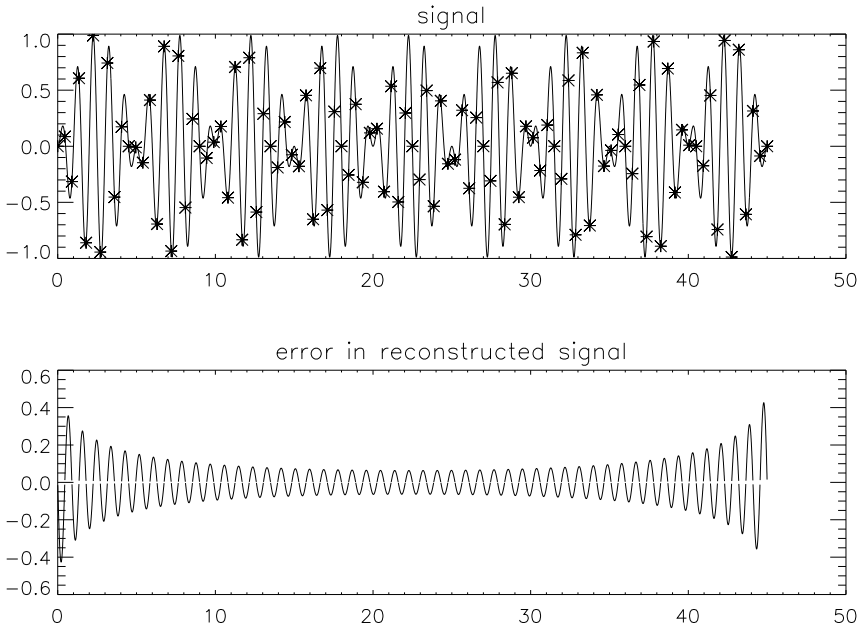


Abbildung 6.7: Wie Abb. 6.5, jedoch Fall (c). Zeitkonstante $\Delta = 0.45$ (10% schneller als kritisch).

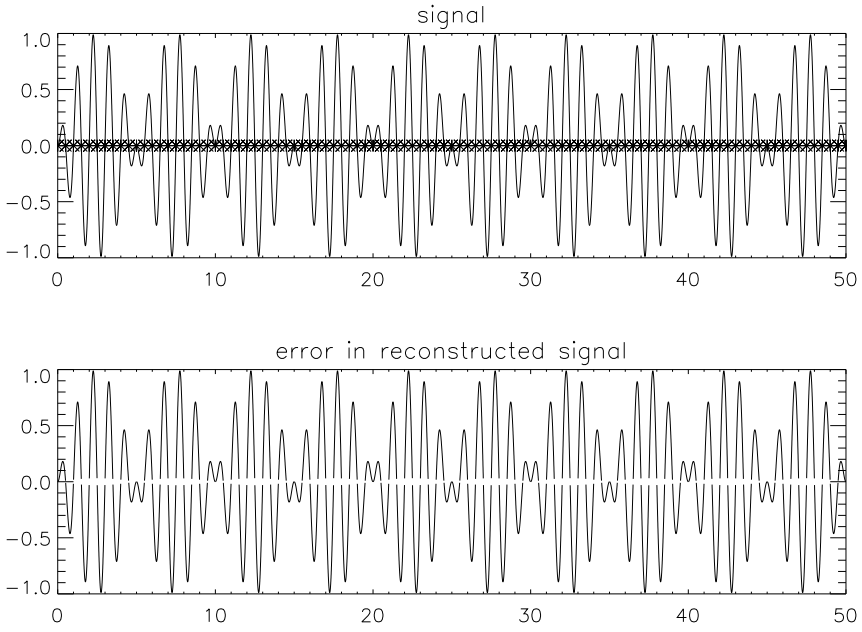


Abbildung 6.8: Wie Abb. 6.5, jedoch Fall (d). Zeitkonstante $\Delta = 0.5$ (*kritisches Abtasten*!)

Es werden jeweils 101 Messungen mit verschiedenen Abtastraten durchgeführt,

a) $\frac{1}{\Delta_1}$, $\Delta_1 = 0.25$ → doppelt so schnell wie kritisch
b) $\frac{1}{\Delta_2}$, $\Delta_2 = 0.40$
c) $\frac{1}{\Delta_3}$, $\Delta_3 = 0.45$
d) $\frac{1}{\Delta_4}$, $\Delta_4 = 0.50$ → kritisch

wobei die kritische Abtastrate einem Zeitintervall von $\Delta_c = \frac{1}{2f_{max}} = 0.5$ entspricht, zumindest falls das Signal "unendlich lang" wäre. Andernfalls (wie hier) kommt es beim Ein- und Ausschalten des Signals zu Beimischungen sehr hoher Frequenzen, die eine präzise Rekonstruktion des Signals zumindest an den (zeitlichen) Rändern unmöglich machen.

In den Abbildungen 6.5 bis 6.8 werden nun das theoretische Signal und die gemessenen Werte h_n (Sterne) im oberen Teil dargestellt. Aufgrund der unterschiedlichen Zeitkonstanten, aber gleichen Anzahl von Messungen ist die Gesamtdauer des Signales und der Messreihe unterschiedlich. Der relative Fehler des aus den Messwerten entsprechend Gl. 6.13 rekonstruierten Signales (für alle Zeitpunkte!) wird im unteren Teil der Abbildung angegeben. Man beachte insbesondere die unterschiedliche Skala dieses Fehlers in den jeweiligen Abbildungen.

Der erste Abbildung 6.5 (doppelt so schnell wie kritisch abgetastet) zeigt außer an den Rändern (Beimischung hoher Frequenzen, s.o.) eine sehr gute Rekonstruktion. Nähert sich die Abtastrate nun ihrem kritischen Wert (Abb. 6.6 und 6.7), wächst der Fehler; dieser kann jedoch durch längeres Messen (wenn das Experiment es gestattet) verkleinert werden. Die letzte Rekonstruktion (bei der kritischen Abtastrate, Abb. 6.7) weist bis zu 100% Fehler *zu allen Zeiten* auf. Aufgrund $\varphi_0 = 0$ werden hier nur die Nulldurchgänge gemessen, doch auch für beliebig andere φ_0 würde man das gleiche Ergebnis erhalten.

Einige Bemerkungen zu negativen Frequenzen. Normalerweise sollten reelle Signale nur positive Frequenzen aufweisen. Die FT ergibt aber nichtverschwindende Komponenten $H(f)$ auch für negative Frequenzen, und es gilt $H(f) = [H(-f)]^*$, vgl. Seite 6-11. Diese Komponenten haben keine physikalische Relevanz und sollten nur als "Zwischenergebnisse" betrachtet werden. *Physikalische Größen*, z.B. die "einseitige" Leistungsdichte (s.u.) sind meist Kombinationen von $H(f)$ und $H(-f)$!

Im Weiteren wollen wir nun zeigen, dass H geradezu nichtverschwindende Werte bei negativen Frequenzen aufweisen *muss*, da ansonsten *kein* reelles Signal (re-)konstruiert werden könnte.

Zunächst wollen wir nochmals festhalten, dass eine FOURIER-Komponente bei Frequenz f,

$$H(f) = \int_{-\infty}^{\infty} h(t)\mathrm{e}^{-2\pi \mathrm{i} f t}\mathrm{d}t$$

im Allgemeinen komplexwertig sein muss, d.h. $H(f) = H_R(f) + \mathrm{i}H_I(f)$, um die benötige Information bzgl. *Amplitude und Phase* zu beinhalten. Um aus diesen FOURIER-Komponenten ein *reelles Signal* zu erhalten,

$$\begin{aligned} h_R(t) &= \int_{-\infty}^{\infty} H(f)e^{2\pi \mathrm{i} f t}\mathrm{d}f \\ &= \int_{-\infty}^{\infty} (H_R(f) + \mathrm{i}H_I(f)) \cdot (\cos(2\pi f t) + \mathrm{i}\sin(2\pi f t))\,\mathrm{d}f \\ &= \int_{-\infty}^{\infty} (H_R(f)\cos(2\pi f t) - H_I(f)\sin(2\pi f t))\,\mathrm{d}f + \\ &\quad + \mathrm{i}\int_{-\infty}^{\infty} (H_I(f)\cos(2\pi f t) + H_R(f)\sin(2\pi f t))\,\mathrm{d}f, \end{aligned}$$

muss also das zweites Integral in obiger Gleichung verschwinden. Wir müssen also fordern, dass

$$
\begin{aligned}
0 &\stackrel{!}{=} \int_{-\infty}^{\infty} (H_I(f)\cos(2\pi ft) + H_R(f)\sin(2\pi ft))\,\mathrm{d}f \\
&= \int_{0}^{\infty} (H_I(f)\cos(2\pi ft) + H_R(f)\sin(2\pi ft))\,\mathrm{d}f + \int_{0}^{\infty} (H_I(-f)\cos(2\pi ft) - H_R(-f)\sin(2\pi ft))\,\mathrm{d}f \\
&\quad \text{(im zweiten Integral } f \to -f\text{, Integrationsgrenzen vertauscht; cos gerade, sin ungerade)} \\
&= \int_{0}^{\infty} ((H_I(f) + H_I(-f))\cos(2\pi ft) + (H_R(f) - H_R(-f))\sin(2\pi ft))\,\mathrm{d}f
\end{aligned}
$$

Dieses Integral verschwindet nun i.A. *nicht* für verschwindende Komponenten bei negativen Frequenzen $H_R(-f) = H_I(-f) \equiv 0$, sondern nur für

$$
\left.\begin{array}{rcl} H_I(f) & = & -H_I(-f) \\ H_R(f) & = & H_R(-f) \end{array}\right\} \quad H(-f) = [H(f)]^*,
$$

d.h. für die auf Seite 6-11 abgeleitete Bedingung Damit ist also gezeigt, dass ein reelles Signal FOURIER-Komponenten bei negativen Frequenzen, $\neq 0$, aufweisen *muss*!

Eine wichtige Größe im Rahmen der FT ist die

Leistung des Signals P. Sie ist eine *Erhalungsgröße* der FOURIER-Transformation, wie folgender Satz zeigt

6.18 Satz (PARSEVALs Theorem). *Für die insgesamte Leistung eines Signals P gilt*

$$
P := \int_{-\infty}^{\infty} |h(t)|^2 \mathrm{d}t = \int_{-\infty}^{\infty} |H(f)|^2 \mathrm{d}f \tag{6.14}
$$

(ohne Beweis).

Damit lässt sich eine spektrale Leistungsdichte im Bereich f, $f + \mathrm{d}f$ definieren,

$$
P(f) = |H(f)|^2,
$$

die machmal auch *zweiseitige* Leistungsdichte genannt wird. Die FOURIER-Komponenten (bzw. ihr Absolutquadrat) geben also die Leistungsdichte des Signals bei einer bestimmten Freuenz an, was ihre physikalische Bedeutung erklärt. Oftmals verwendet man auch die sog. *einseitige* Leistungsdichte

$$
P_e(f) = (P(f) + P(-f)) = |H(f)|^2 + |H(-f)|^2, \quad 0 < f < \infty.
$$

Man beachte, dass für reelle Signale $h(t)$ (mit $H(-f) = [H(f)]^*$)

$$
P(f) = H(f) \cdot H^*(f) = H^*(-f) \cdot H(-f) = P(-f)
$$

die Leistungsdichte symmetrisch ist. Für *beliebige* Signale gilt bei der kritischen Frequenz (aufgrund bestimmter Periodizitätsbedingungen, s.u.) $H(f_c) = H(-f_c)$. Falls nun das Signal reell ist, folgt daraus sofort, dass

$$
H(f_c) = [H(-f_c)]^* = [H(f_c)]^* \qquad \text{(reelles Signal)}
$$

$H(f_c)$ reell ist, unabhängig davon, ob das Signal gerade oder ungerade ist. Dieser Sachverhalt wird in Kürze wichtig werden.

Aliasing.

Im vorigen Abschnitt hatten wir gesehen, dass ein bandweitenlimitiertes Signal vollständig rekonstruiert werden kann, falls das Signal schneller als kritisch abgetastet wurde. Falls das Signal andererseits *nicht* bandweitenlimitiert ist, wird die Situation problematischer. In diesem Fall wird nämlich die gesamte spektrale Leistung, die *außerhalb* des Frequenzbereiches $-f_c < f < f_c$ liegt, fälschlicherweise in diesen Bereich hinein "transformiert", und das Signal kann *nicht* mehr rekonstruiert werden.[3] Dieses Phänomen heisst Aliasing (falsche Übersetzung), und resultiert auf Grund des *diskreten* Abtastens!

Motivation. Zwei unterschiedliche Signale $\exp(2\pi \mathrm{i} f_1 t)$ und $\exp(2\pi \mathrm{i} f_2 t)$, die mit einem Zeitintervall Δ abgetastet werden, ergeben das *gleiche* Signal zu allen (diskreten) Zeitpunkten $t = n\Delta$, wenn ihre Frequenzdifferenz

$$|f_2 - f_1| = \frac{k}{\Delta}, \qquad (k = 1, 2, 3, \dots) \tag{6.15}$$

ein Vielfaches der Abtastrate $\frac{1}{\Delta}$ ist.

Man beachte, dass $\frac{1}{\Delta}$ gerade der Frequenzdifferenz des Bereiches $[-f_c, f_c]$ entspricht. In Zusammenhang mit Gl. 6.15 werden deshalb alle Frequenzkomponenten, die außerhalb dieses Bereiches liegen, in diesen Bereich transformiert, und statt der FOURIER-Komponente $H(f_1)$ "misst" man[4] $H(f_1) + H(f_2)$! Mit anderen Worten: Aufgrund des diskreten Abtastens kann nicht zwischen den Komponenten bzgl. f_1 und f_2 unterscheiden, und beide addieren sich bei f_1.

6.19 Beweis. *O.E.d.A. sei $f_2 = f_1 + \Delta f > f_1$. Damit gilt für alle Zeiten t*

$$\exp(2\pi \mathrm{i} f_2 t) = \exp(2\pi \mathrm{i} f_1 t) \cdot \exp(2\pi \mathrm{i} \Delta f t).$$

Zu den abgetasteten Zeitpunkten t_n gilt zusätzlich

$$\exp(2\pi \mathrm{i} \Delta f t_n) = \exp(2\pi \mathrm{i} \Delta f \cdot n \cdot \Delta).$$

Dieser Faktor ist zu allen Zeitpunkten der Messung dann (und nur dann) gleich "Eins", wenn $\Delta f \cdot \Delta$ eine ganze Zahl ist, d.h., wenn

$$\Delta f = \frac{k}{\Delta} \qquad \text{für} \quad k = 1, 2, 3, \dots$$

Falls $f_2 < f_1$, gilt das analoge, womit der Absolutbetrag in (6.15) erkärt ist. □

Nochmals: Die FOURIER-Komponenten entsprechend $f_k = f_1 + \frac{k}{\Delta}$ des Signals werden bei der FT in den zu f_1 gehörigen Kanal gebracht (falls $-f_c < f_1 < f_c$), d.h. anstelle von $H(f_1)$ "messen" wir

$$H(f_1) + H(f_1 + \frac{1}{\Delta}) + H(f_1 + \frac{2}{\Delta}) + \dots,$$

sofern die entsprechenden Komponenten vorhanden sind.

[3] Zumindest nicht mehr vollständig: Falls bei den hohen Frequenzen nur wenig Leistung vorhanden ist, ist die Rekonstruktion zumindest näherungsweise "richtig", vgl. die "Randbereiche" in Beispiel 6.17.

[4] d.h. erhält man durch FOURIER-Transformation

Insbesondere gilt Folgendes ($\epsilon < \frac{1}{\Delta} : k = 1$ vorausgesetzt, vgl. Skizze):

- Komponenten mit $f_c + \epsilon$ werden in den "Kanal" $-f_c + \epsilon$ (da $\underbrace{-f_c + \epsilon}_{f_1} + \frac{1}{\Delta} = \underbrace{f_c + \epsilon}_{f_2}$)
- und Komponenten mit $-f_c - \epsilon$ in den "Kanal" $f_c - \epsilon$ (da $\underbrace{-f_c - \epsilon}_{f_2} + \frac{1}{\Delta} = \underbrace{f_c - \epsilon}_{f_1}$)

transformiert.

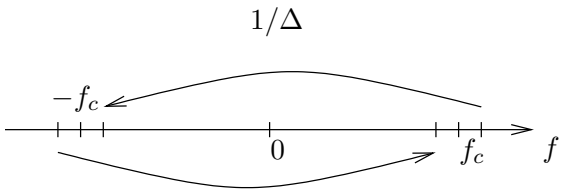


Abbildung 6.9: Aliasing zweier FOURIER-Komponenten mit $f_c + \epsilon$ und $-f_c - \epsilon$

Das Auftreten oder Nichtauftreten von Aliasing-Effekten ermöglicht allerdings eine relativ einfache Überprüfung, ob "richtig" abgetastet wurde:

> Bei $\pm f_c$ sollte die (spektrale) *Leistungsdichte* gegen Null gehen;
> falls nicht, wurde falsch abgetastet.

(siehe die Beispiele in Ergänzung 6.5.3.)

6.2.4 Diskrete FOURIER-Transformation

(Das im Weiteren beschriebene Vorgehen ist in vieler Hinsicht analog zur Berechnung der *Näherungswerte* für die FOURIER- Koeffizienten, a_k^*, b_k^*, vgl.(6.6).)

Es sei wiederum ein (beliebiges, d.h. komplexes) Signal gegeben, das zu N Zeitpunkten (startend mit $t = 0$) gemessen wurde

$$h_k = h(t_k), \quad t_k = k\Delta, \quad k = 0, \ldots, N-1$$

Wir unterstellen hierbei, dass das Zeitintervall Δ "klein genug" ist. Dies kann zumindest *a posteriori* (d.h. nach erfolgter FT) an den frequentiellen Rändern (s.o.) überprüft werden; gegebenenfalls muss das Intervall verringert und erneut gemessen werden.

Wir suchen nun die FOURIER-Transformierte $H(f)$. Man beachte, dass im Gegensatz zu den in Kap. 6.1 betrachteten FOURIER-Polynomen und -Reihen der Bereich *nicht* auf 2π(periodisch) eingeschränkt ist!

Da wir "nur" N Datenpunkte haben, können aus Eindeutigkeitsgründen auch "nur" N *diskrete* Frequenzen berechnet werden, die, wie oben erläutert, im Bereich $[-f_c, f_c]$ mit $f_c = \dfrac{1}{2\Delta}$ liegen müssen. Aufgrund $\Delta f = \frac{2f_c}{N} = \frac{1}{N\Delta}$ liegen diese Frequenzen dann bei

$$f_n = \frac{n}{N\Delta}, \qquad n = -N/2, \ldots, N/2.$$

Dies sind natürlich $N+1$ Frequenzen, die sich aber auf N reduzieren, wenn wir berücksichtigen, dass $H(-f_c) = H(f_c)$ ist (s.u.).

Benutzen wir zur numerischen Berechnung der FOURIER-Integrale (6.10) wie in Kap. 6.1 die Trapezintegration, so ergibt sich

$$H(f_n) = \int_{-\infty}^{\infty} h(t)\mathrm{e}^{-2\pi\mathrm{i}f_n t}\mathrm{d}t$$
$$\xrightarrow{\text{(Trapez)}} \sum_{k=0}^{N-1} h_k \mathrm{e}^{-2\pi\mathrm{i}f_n t_k}\Delta.$$

Man beachte, dass die Integrationsgewichte alle gleich sind (vgl. Seite 6-5), d.h. folgende *implizite* Annahme getroffen wurde:

(a) $h_N = h_0$ (vgl. Gl. (6.6)): das Signal ist periodisch bzgl. des Gesamtintervalles $N\Delta$ oder

(b) $h_N = h_0 \approx 0$: das Signal ist endlich und geht an den Rändern gegen 0.

Damit ergibt sich

$$\begin{aligned} H(f_n) =& \Delta \sum_{k=0}^{N-1} h_k \mathrm{e}^{-2\pi\mathrm{i}\frac{n}{N\Delta}k\Delta} \\ =& \Delta \sum_{k=0}^{N-1} h_k \mathrm{e}^{-2\pi\mathrm{i}kn/N} \\ :=& \Delta \cdot H_n \end{aligned}$$

Schon hier ist ersichtlich, dass die FOURIER-Komponenten selbst periodisch in $(k \cdot n)$ mit der Periode N sind, ein Faktum, dass sich für die *schnelle* Berechnung (FFT!) als essentiell erweisen wird.

Die diskrete FOURIER-Transformation hat also folgende Eigenschaften:

- Aus N (komplexen) Sigalen h_k ergeben sich N (komplexe) Komponenten H_n.
- Diese Komponenten hängen nicht von dimensionalen Parametern ab, sondern nur von der *Anzahl* der (Mess-)Punkte.
- Die notwendige Dimensionierung findet erst final durch

  $$H(t_n) = \Delta \cdot H_n$$

  statt.
- Die Komponenten

  $$H_n = \sum_{k=0}^{N-1} h_k \mathrm{e}^{-2\pi\mathrm{i}kn/N}$$

  sind (u.a.) periodisch in n mit der Periode N, z.B. gilt $H_{-n} = H_{N-n}$, $n = 1, 2, \ldots$
- Insbesondere ergibt sich damit $H_{-\frac{N}{2}} = H_{\frac{N}{2}}$, d.h. die schon öfters benutzte Identität

  $$H(-f_c) = H(f_c). \tag{6.16}$$

- Anstatt $n = -\frac{N}{2} \ldots \frac{N}{2}$ zu verwenden, indiziert man üblicherweise (um den folgenden FFT-Algorithmus so einfach wie möglich zu halten) $n = 0, \ldots, N-1$.

- Demzufolgen liegt die FOURIER-Komponente zur

Frequenz	0	bei	$n = 0$
Frequenz	$0 < f < f_c$	bei	$1 \le n \le N/2 - 1$
Frequenz	f_c ($\hat{=} -f_c$)	bei	$n = N/2$
Frequenz	$-f_c < f < 0$	bei	$N/2 + 1 \le n \le N - 1$

Diese Umindizierung ist "erlaubt", da die negative Frequenzen $n = -\frac{N}{2} \cdots -1$ nun bei den Indizes $n = N/2, \ldots N-1$ zu liegen kommen, d.h. die alten Indizes n durch die neuen $n + N$ ersetzt werden und die Fourierkomponenten periodisch in n mit Periode N sind (s.o.),

$$\exp\left(-2\pi \mathrm{i} k \frac{(n+N)}{N}\right) = \exp\left(-2\pi \mathrm{i} k \frac{n}{N}\right) \cdot \underbrace{\exp\left(-2\pi \mathrm{i} k\right)}_{=1 \ \forall \ k \in \mathbb{N}_0}.$$

Die **Rücktransformation**

$$h(t) = \int_{-\infty}^{\infty} H(f) \mathrm{e}^{2\pi \mathrm{i} f t} \mathrm{d}f$$

wird analog zur Vorwärtstransformation durchgeführt,

$$\begin{aligned} h_k = h(t_k) &= \sum_{n=0}^{N-1} (\Delta \cdot H_n)\, \mathrm{e}^{2\pi \mathrm{i} f_n t_k} \Delta f \\ &= \Delta \cdot \Delta f \sum_{n=0}^{N-1} H_n \ \mathrm{e}^{2\pi \mathrm{i} \frac{n}{N\Delta} k \Delta} \\ &= \Delta \frac{1}{\Delta N} \sum_{n=0}^{N-1} H_n \mathrm{e}^{2\pi \mathrm{i} k n/N}, \end{aligned}$$

Insgesamt ergibt sich also

$$\left.\begin{array}{lll} h(t_k) = h_k & = & \dfrac{1}{N} \displaystyle\sum_{n=0}^{N-1} H_n \mathrm{e}^{2\pi \mathrm{i} k n/N} \\ H(f_n) = \Delta \cdot H_n & = & \Delta \displaystyle\sum_{n=0}^{N-1} h_k \mathrm{e}^{-2\pi \mathrm{i} k n/N} \end{array}\right\} \begin{array}{ll} \text{gleicher} & \text{Vorzeichen umkehren,} \\ \text{Algorithmus!} & \text{andere "Normierung"} \end{array} \tag{6.17}$$

Diskrete PARSEVAL-Relation. Mit diesen Transformationen lässt sich dann auch die PARSEVAL-Relation in ihrer diskreten Formulierung angeben,

$$\sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2, \tag{6.18}$$

wobei diese Relation, abgesehen von ihrer eigentlichen Bedeutung, auch einen guten Test für die Programmierung des Algorithmus darstellt.

6.2.5 FFT – Fast-FOURIER-Transformation

Die Berechnung der FOURIER-Komponenten H_n lässt sich auch in Matrixschreibweise angeben,

$$\begin{aligned} H_n &= \sum_{k=0}^{N-1} h_k \mathrm{e}^{-2\pi \mathrm{i} nk/N} = \sum_{k=0}^{N-1} (W_N)^{nk} h_k, \qquad W_N = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}}, \qquad n = 0, \ldots, N-1 \\ \boldsymbol{H} &= W_{nk} \boldsymbol{h}, \end{aligned}$$

d.h. der Komponentenvektor $\boldsymbol{H}$ ergibt sich durch Multiplikation der Matrix W_{nk} mit dem Signalvektor $\boldsymbol{h}$.

Demzufolge sollte also die Berechnung aller Komponenten $\mathcal{O}(N^2)$ Operationen benötigen (vgl. auch Seite 6-8). Unter Verwendung der Fast-FOURIER-Transformation lässt sich diese Aufgabe aber in

$$\text{FFT:} \quad \mathcal{O}(N \log_2 N)$$

Operationen lösen. Diese Erkenntnis ist nicht neu, sondern war u.a schon Gauß (1805), Danielson & Lanczos (1942), Cooley & Tukey (den "Erfindern des sog. *Butterfly*-Algorithmus; Mitte der 60er) bekannt.[5]

Im Weiteren folgen wir hier der Ableitung von Danielson & Lanczos, wobei die Anzahl der Punkte eine Potenz von 2 sein muss, $N = 2^n$. Es existieren auch Algorithmen, die andere Basen verwenden, allerdings ist der hier vorgestellte Algorithmus einer der am häufigst verwendeten (und einfachsten)!

Einer der entscheidenden Punkt in allen Ableitungen ist der folgende: da $(W_N)^{nk}$ periodisch mit Periode N ist, muss diese (kostspielig zu ermittelnde) Größe nur N-mal (aufgrund der $\pm$ Symmetrie sogar nur $N/2$-mal) berechnet werden, obwohl tatsächlich $n \cdot k = N^2$ Elemente benötigt werden. Des Weiteren erlaubt die Periodizität auch eine geschickte Umformulierung des Problemes, die letztendlich die extrem schnelle Abarbeitung der Transformation ermöglicht:

Das DANIELSON-LANCZOS Lemma

6.20 Lemma. *Eine diskrete* FOURIER*-Transformation der "Länge" N kann als Summe zweier diskreter* FOURIER*-Transformationen der Länge $\frac{N}{2}$ geschrieben werden, wobei die eine aus den "geraden" Punkten und die andere aus den "ungeraden" Punkten des Orginalproblems gebildet wird.*

6.21 Beweis.

$$\begin{aligned} {}^{N}H_n &=: \sum_{k=0}^{N-1} \mathrm{e}^{-2\pi \mathrm{i} nk/N} h_k \\ &= \sum_{k=0}^{N/2-1} \mathrm{e}^{-2\pi \mathrm{i} n(2k)/N} h_{2k} + \sum_{k=0}^{N/2-1} \mathrm{e}^{-2\pi \mathrm{i} n(2k+1)/N} h_{2k+1} \\ &= \sum_{k=0}^{N/2-1} \mathrm{e}^{-2\pi \mathrm{i} nk/(N/2)} h_{2k} + W_N^n \sum_{k=0}^{N/2-1} \mathrm{e}^{-2\pi \mathrm{i} nk/(N/2)} h_{2k+1} \\ &= {}^{N/2}H_n^e + W_N^n \cdot {}^{N/2}H_n^o, \end{aligned} \tag{6.19}$$

wobei "e" gerade (even) und "o" ungerade (odd) bedeuted. □

Man beachte, dass $n = 0, \ldots, N-1$, aber ${}^{N/2}H_n^e$ und ${}^{N/2}H_n^o$ periodisch mit Länge $N/2$ sind.

[5] und, den Verfassern der ersten Version dieses Skriptes zu Folge, auch allen Studenten in der ersten Reihe, noch bevor es der Dozent (J.P.) wusste.

Obiges Lemma kann nun rekursiv angewendet werden (vgl. Abbildung 6.10), und im letzten Schritt erhalten wir

$$ {}^{\frac{N}{N}}H_n^{oooeeo\ldots eo} = \sum_{k=0}^{N/N-1} \mathrm{e}^{-2\pi \mathrm{i} n k/(N/N)} h_m \stackrel{!}{=} h_m, $$

wobei sich nun die Frage stellt, welcher Komponente von $\boldsymbol{h}$ der Index m entspricht.

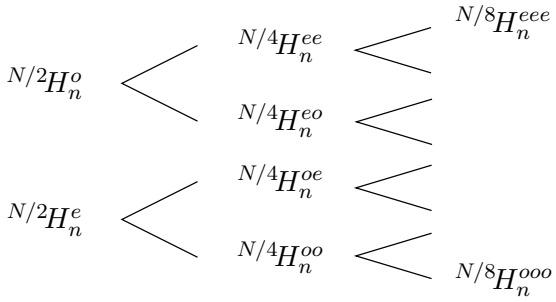


Abbildung 6.10: Schema zum DANIELSON-LANCZOS-Lemma am Beispiel $N = 8$

Die Länge des "Musters" beträgt offensichtlich $\log_2 N$, und für die "Berechnung" von m lässt sich folgende Vorschrift angeben:

(1) man nehme das Muster und drehe dieses von hinten nach vorne um, z.B.

$$ [oooeeo \ldots eo] \quad \rightarrow \quad [oe \ldots oeeooo] $$

(2) die "e" werden durch eine 0, die "o" durch eine 1 ersetzt, in unserem Beispiel

$$ [oe \ldots oeeooo] \quad \rightarrow \quad [10 \ldots 100111] $$

(3) und die entsprechende Binärzahl ist der gesuchte Index m!

Diese Vorgehensweise lässt sich aus der sukzessiven (und symmetrischen) Unterteilung der jeweiligen FT in "gerade" und "ungerade" Transformationen niedrigerer Ordnung ableiten.

Der *entscheidende Punkt* des DANIELSON-LANCZOS-Lemmas ist nun der folgende:

- **Für alle** Komponenten $n \in [0, \ldots, N-1]$ wird das gleiche finale Bitmuster erzeugt, und die Berechnung der unterschiedlichen H_n unterscheidet sich nur bezüglich der anzuwendenden Faktoren W_N^n, d.h. der Potenz n, mit der W_N potenziert wird.

6.22 Beispiel (DANIELSON-LANCZOS Rekursion für den Fall $N = 8$ und beliebige Komponenten $n = 0, \ldots, 7$). (Man vergleiche mit Kap. 6.5.4).

$$
\begin{aligned}
{}^8H_n &= {}^4H_n^e + W_8^n \cdot {}^4H_n^o \\
&= \left({}^2H_n^{ee} + W_4^n \cdot {}^2H_n^{eo}\right) + W_8^n \left({}^2H_n^{oe} + W_4^n \cdot {}^2H_n^{oo}\right) \\
&= \left(\left({}^1H_n^{eee} + W_2^n \cdot {}^1H_n^{eeo}\right) + W_4^n \left({}^1H_n^{eoe} + W_2^n \cdot {}^1H_n^{eoo}\right)\right) + W_8^n \left(\left({}^1H_n^{oee} + W_2^n \cdot {}^1H_n^{oeo}\right) + W_4^n \left({}^1H_n^{ooe} + W_2^n \cdot {}^1H_n^{ooo}\right)\right) \\
&= \left((h_{eee} + W_2^n h_{eeo}) + W_4^n (h_{eoe} + W_2^n h_{eoo})\right) + W_8^n \left((h_{oee} + W_2^n h_{oeo}) + W_4^n (h_{ooe} + W_2^n h_{ooo})\right) \\
&= \left((h_{000} + W_2^n h_{100}) + W_4^n (h_{010} + W_2^n h_{110})\right) + W_8^n \left((h_{001} + W_2^n h_{101}) + W_4^n (h_{011} + W_2^n h_{111})\right) \\
&= \left((h_0 + W_2^n h_4) + W_4^n (h_2 + W_2^n h_6)\right) + W_8^n \left((h_1 + W_2^n h_5) + W_4^n (h_3 + W_2^n h_7)\right). \qquad (6.20)
\end{aligned}
$$

Die ursprüngliche Ordnung $[0, 1, 2, 3, 4, 5, 6, 7]$ wird also (im Falle $N = 8$) in die Reihenfolge $[0, 4, 2, 6, 1, 5, 3, 7]$ umgeordnet.

Nochmals: Diese Umordnung ist für alle $n \in [0, N-1]$ gültig, die Berechnung der unterschiedlichen H_n unterscheidet sich nur in den Gewichten W_8^n, W_4^n und W_2^n!

Aufrund der Perodizität müssen zunächst nur $N = 8$ Gewichte explizit berechnet werden, alle anderen sind "Abfallprodukte". Für beliebige N sind dies die Gewichte bzgl. $n/N, n = 0, \ldots N-1$ (sofern N eine Potenz von 2 ist!)

$$
\begin{array}{lll}
W_8^n & = \mathrm{e}^{-2\pi \mathrm{i} n/8} & \text{für } n = 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7 \\
W_4^n & = \mathrm{e}^{-2\pi \mathrm{i} n/4} & \qquad\qquad\quad \updownarrow \quad\ \updownarrow \quad\ \updownarrow \quad\ \updownarrow \\
 & = \mathrm{e}^{-2\pi \mathrm{i}(2n)/8} & \qquad\qquad n = 0, \quad 1, \quad 2, \quad 3 \\
 & & \qquad\qquad \hat{=}\ 4, \quad 5, \quad 6, \quad 7 \\
W_2^n & = \mathrm{e}^{-2\pi \mathrm{i} n/2} & \qquad\qquad\quad \uparrow \quad\ \uparrow \\
 & = \mathrm{e}^{-2\pi \mathrm{i}(4n)/8} & \qquad\qquad\quad \downarrow \quad\ \downarrow \\
 & = \mathrm{e}^{-2\pi \mathrm{i}(2n)/4} & \qquad\qquad n = 0, \quad 1 \\
 & & \qquad\qquad \hat{=}\ 2, \quad 3 \\
 & & \qquad\qquad \hat{=}\ 4, \quad 5 \\
 & & \qquad\qquad \hat{=}\ 6, \quad 7
\end{array}
$$

Ein zweckmäßiges Rechenschema startet mit den niedrigsten "Frequenzen" W_2,

$$
\begin{aligned}
W_2^0 &= 1 \\
W_2^1 &= \mathrm{e}^{-\pi \mathrm{i}} = \cos(-\pi) + \mathrm{i}\sin(-\pi) = \cos(\pi) - \mathrm{i}\sin(\pi) = -1 \qquad (= -W_2^0),
\end{aligned}
$$

danach mit W_4

$$
\begin{aligned}
W_4^1 &= \mathrm{e}^{-\mathrm{i}\pi/2} = \cos(\tfrac{\pi}{2}) - \mathrm{i}\sin(\tfrac{\pi}{2}) = -\mathrm{i} \\
W_4^3 &= \mathrm{e}^{-3\mathrm{i}\pi/2} = \cos(\tfrac{3\pi}{2}) - \mathrm{i}\sin(\tfrac{3\pi}{2}) = \mathrm{i} \qquad (= -W_4^1),
\end{aligned}
$$

und (im Falle $N = 8$) final

$$
\begin{aligned}
W_8^1 &= \mathrm{e}^{-\mathrm{i}\pi/4} = \cos(\tfrac{\pi}{4}) - \mathrm{i}\sin(\tfrac{\pi}{4}) = \frac{1}{\sqrt{2}}(1 - \mathrm{i}) \\
W_8^3 &= \mathrm{e}^{-3\mathrm{i}\pi/4} = \cos(\tfrac{3\pi}{4}) - \mathrm{i}\sin(\tfrac{3\pi}{4}) = -\frac{1}{\sqrt{2}}(1 + \mathrm{i}) \\
W_8^5 &= -\frac{1}{\sqrt{2}}(1 - \mathrm{i}) \qquad (= -W_8^1) \\
W_8^7 &= \frac{1}{\sqrt{2}}(1 + \mathrm{i}) \qquad (= -W_8^3).
\end{aligned}
$$

Tatsachlich sind also sogar nur $N/2 = 4$ Werte zu berechnen, da die anderen 4 Werte das Negative davon sind.

Der numerische Algorithmus der Mustererzeugung ist ebenso einfach, er entspricht einer simplen "Bitumkehr"des Ausgangsmusters:

0	$\rightarrow$	000	$\rightarrow$	000	$\rightarrow$	0
1		001		100		4
2		010		010		2
3		011		110		6
4		100		001		1
5		101		101		5
6		110		011		3
7		111		111		7
		$\downarrow$		$\downarrow$		
		entspricht		Bitumkehr, entspricht		
		eee . . . ooo		Vertauschung d. Musters		

Das prinzipielles Rechenschema (vgl. Abbildung 6.11) [6] wird zusammengefasst mittels

$$
\begin{aligned}
{}^{2n}H_i &= {}^{n}H_i + W_k^i \, {}^{n}H_j \\
{}^{2n}H_j &= {}^{n}H_i + W_k^j \, {}^{n}H_j = {}^{n}H_i - W_k^i \, {}^{n}H_j.
\end{aligned}
$$

Diese Zusammenfassung, mit den Indexdifferenzen (j-i)=1,2,4,. . . , startend bei ${}^{1}H = h_k$ (umsortiert!), ergibt Gl. 6.20 für alle n! Am Beispiel von ${}^{8}H_5$ wollen wir demonstrieren, dass dies tatsächlich der

[6] An dieser Stelle sei nochmals auf das ergänzende Kapitel 6.5.4 hingewiesen, in dem die FFT (wiederum am Beispiel $N = 8$) unter einem etwas anderen Blickwinkel ableitet wird, um ein tieferes Verständnis zu ermöglichen.

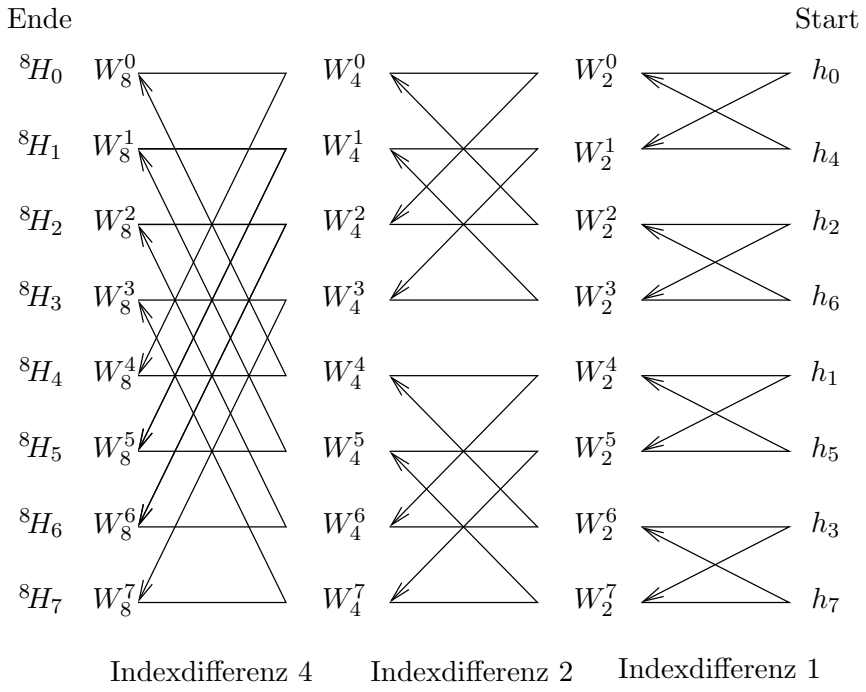


Abbildung 6.11: Prinzipielles Rechenschema für die FFT am Beispiel $N = 8$. Die Pfeile geben an, welche Werte jeweils zusammengefasst werden müssen; das entsprechende Gewicht (an der Spitze des Pfeiles) muss dabei mit der Komponente des *höheren* Index multipliziert werden (siehe Text).

Fall ist. Folgen wir den "Pfeilen" in Abb. 6.11, so ergibt sich

$$\begin{aligned} {}^8H_5 &= {}^4H_1 + W_8^5 \cdot {}^4H_5 \\ &= {}^2H_1 + W_4^1 \cdot {}^2H_3 + W_8^5 \cdot ({}^2H_5 + W_4^5 \cdot {}^2H_7) \\ &= h_0 + W_2^1 h_4 + W_4^1(h_2 + W_2^3 h_6) + W_8^5((h_1 + W_2^5 h_5) + W_4^5(h_3 + W_2^7 h_7)), \end{aligned}$$

und unter Verwendung der Periodizität $W_2^1 = W_2^5$, $W_4^1 = W_4^5$, $W_2^3 = W_2^5$ und $W_2^7 = W_2^5$ erhalten wir den zu (6.20) identischen Ausdruck für $n = 5$,

$${}^8H_5 = h_0 + W_2^5 h_4 + W_4^5(h_2 + W_2^5 h_6) + W_8^5((h_1 + W_2^5 h_5) + W_4^5(h_3 + W_2^5 h_7)).$$

Für unser Beispiel $N = 8$ lautet damit der prinzipielle FFT-Algorithmus

- Umsortierung der h_k, $k = 0, \ldots, N-1$, durch Bitumkehr
- 4 ^2H Transformationen
  2 ^4H Transformationen, mit Ordnung aus vorhergehendem Schritt
  1 ^8H Transformationen, mit Ordnung aus vorhergehendem Schritt

Es sind also insgesamt $\log_2 N$ Schritte durchzuführen. Pro Schritt werden N Operationen (eine Addition/eine Multiplikation pro Element) benötigt, vgl. (6.20), und die (einmalige) Berechnung der $N/2$ Gewichte ist vernachlässigbar, falls viele Schritte durchzuführen sind.

Insgesamt finden wir also tatsächlich, dass die FFT $\mathcal{O}(N \log_2 N)$ wesentliche Operationen erfordert, und der Algorithmus lässt sich (ohne Bitumkehr) mit ca. 20 Anweisungen formulieren. Für große N (was normalerweise der Fall ist) ist es entscheidend, die Gewichte numerisch stabil berechnen.[7]

[7] "Numerical Recipes", Kap. 5.5

6.23 Beispiel (Fast-FOURIER-Analyse eines einfachen Signals). Wir betrachten das einfache (reelle) Signal

$$h(t) = 3 + 2\sin(\frac{\pi}{4}t)$$

mit Frequenz $2\pi f = \frac{\pi}{4}$, d.h. $f = \frac{1}{8}$.

Dieses Signal werde mit einem Zeitintervall $\Delta t = 1$ abgetastet, entsprechend einer NYQUIST-Frequenz $f_c = \frac{1}{2} > f$; diese *Sampling*-Rate sollte somit für eine vollständige Rekonstruktion des Signales geeignet sein. Gemessen werde zu den 8 Zeitpunkten $t = 0, \ldots, 7$ (siehe Abbildung 6.12). Damit liegt folgende Messreihe vor:

$$\begin{array}{llll} h_0 = 3 & h_1 = 3+\sqrt{2} & h_2 = 5 & h_3 = 3+\sqrt{2} \\ h_4 = 3 & h_5 = 3-\sqrt{2} & h_6 = 1 & h_7 = 3-\sqrt{2} \end{array}$$

.

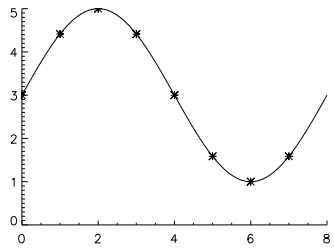


Abbildung 6.12: Sinus-Signal und 8 Messpunkte aus Beispiel 6.23

Unser FFT Rechenschema lautet für diesen Fall

$$\begin{array}{lcr} H_0 & = & 24 \\ H_1 & & -8\mathrm{i} \\ H_2 & & 0 \\ H_3 & & 0 \\ H_4 & & 0 \\ H_5 & & 0 \\ H_6 & & 0 \\ H_7 & & 8\mathrm{i} \end{array} = \left\{ \begin{array}{ccccc} 24 & 12 & 6 & 3 & h_0 \\ -4\mathrm{i} + \frac{1}{\sqrt{2}}(1-\mathrm{i})2\sqrt{2}(1-\mathrm{i}) & -4\mathrm{i} & 0 & 3 & h_4 \\ 0 & 0 & 6 & 5 & h_2 \\ 4\mathrm{i} - \frac{1}{\sqrt{2}}(1+\mathrm{i})2\sqrt{2}(1+\mathrm{i}) & 4\mathrm{i} & 4 & 1 & h_6 \\ 0 & 12 & 6 & 3+\sqrt{2} & h_1 \\ -4\mathrm{i} - \frac{1}{\sqrt{2}}(1-\mathrm{i})2\sqrt{2}(1-\mathrm{i}) & 2\sqrt{2}(1-\mathrm{i}) & 2\sqrt{2} & 3-\sqrt{2} & h_5 \\ 0 & 0 & 6 & 3+\sqrt{2} & h_3 \\ 4\mathrm{i} + \frac{1}{\sqrt{2}}(1+\mathrm{i})2\sqrt{2}(1+\mathrm{i}) & 2\sqrt{(}1+\mathrm{i}) & 2\sqrt{2} & 3-\sqrt{2} & h_7 \\ \text{Indexdifferenz 4} & \text{Indexdifferenz 2} & \text{Indexdifferenz 1} & & \text{umgeordnet} \end{array} \right.$$

Beachte:

- $\Delta f = \frac{2f_c}{N} = \frac{1}{8}$, d.h. die FOURIER-Transformierte weist nicht-verschwindende Komponenten bei den Indizes (f_0, f_1, f_7) entsprechend den Frequenzen $f = 0, \frac{1}{8}, -\frac{1}{8}$ auf.
- Das Verhalten der Komponenten entspricht der Voraussage für reelle Signale $H(-f) = [H(f)]^*$ (vgl. Seite 6-11).
- Die diskrete PARSEVAL-Relation (6.18) wird als Test überprüft:

  $$\sum |h(t_k)|^2 = 2 \cdot 3^2 + 5^2 + 1^2 + 2(3+\sqrt{2})^2 + 2(3-\sqrt{2})^2 = 88$$

  soll gleich

  $$\frac{1}{N}\sum |H_n|^2 = \frac{(24^2 + 64 + 64)}{8} = 88 \qquad \text{sein, stimmt!}$$

Rekonstruktion des Signals. Vorausgreifend auf den nächsten Abschnitt berechnen wir nun aus den FOURIER-komponenten die FOURIER-Koeffizienten des entsprechenden interpolierenden Polynoms. Da sich das Signal aufgrund $f < f_c$ (nur eine Frequenz vorhanden) rekonstruieren lassen sollte, sollte das Polynom mit dem Signal identisch sein!

In Gleichung (6.21) werden wir den Zusammenhang zwischen Komponenten und Koeffizienten ableiten,

$$\left.\begin{array}{rcll} a_k^* & = & \frac{2}{N}\,\Re(H_k) & k = 0, \ldots, N/2 \\ b_k^* & = & -\frac{2}{N}\,\Im(H_k) & k = 1, \ldots, N/2-1 \end{array}\right\} \quad \text{(nur für positive Frequenzen).}$$

Demzufolge ergibt sich

$$\begin{aligned} a_0^* &= \frac{1}{4}\cdot 24 = 6 \qquad \text{(einziger Realteil)}, \\ b_1^* &= \frac{1}{4}\cdot 8 = 2, \end{aligned}$$

und das (interpolierende) *Fourier*-Polynom zu den Signalen $h_k, k = 0, 7$ lautet

$$\begin{aligned} h(t) &= \frac{a_0^*}{2} + \sum_{i=1}^{N/2-1} (a_i^* \cos(2\pi f_i t) + b_i^* \sin(2\pi f_i t)) + \frac{a_{N/2}^*}{2}\cos(2\pi f_{N/2} t) \qquad \text{(siehe Gl. 6.7)} \\ &\rightarrow 3 + 2\sin(2\pi\frac{1}{8}t) = 3 + 2\sin(\frac{\pi}{4}t), \end{aligned}$$

d.h. entspricht tatsächlich unserem Ausgangssignal für beliebige Zeiten t!.

Geringere *Sampling*-**Rate.** Da das Signal nur *eine* Frequenz bei $f = 1/8$ aufweist, ist das Abtasten mit $\Delta t = 1$ eigentlich "zu viel des Guten" ("*oversampling*"). Es sollte im Prinzip ausreichen, das Signal mit $\Delta t = 2$ abzutasten, da dann immer noch $f_c = \frac{1}{4} > f$ gilt. Dies soll im Folgenden durchgeführt werden, um insbesondere die Unterschiede zur obigen Analyse aufzuzeigen. Wir nehmen dazu an, dass die "alten" Signale h_1, h_3, h_5, h_7 vorliegen, jetzt natürlich mit den Indizes $0 \ldots 3$. Mit entsprechender Bitumkehr lautet dann das FFT-Schema für $N = 4$

$$\begin{array}{lc} H_0 & \\ H_1 & \\ H_2 & = \\ H_3 & \end{array} \left\{ \begin{array}{cccc} 12 & 6 & 3+\sqrt{2} & h_0 \\ 2\sqrt{2}(1-\mathrm{i}) & 2\sqrt{2} & 3-\sqrt{2} & h_2 \\ 0 & 6 & 3+\sqrt{2} & h_1 \\ 2\sqrt{2}(1+\mathrm{i}) & 2\sqrt{2} & 3-\sqrt{2} & h_2 \\ \text{Indexdifferenz 2} & \text{Indexdifferenz 1} & & \text{umgeordnet} \end{array} \right.$$

Wiederum ist $\Delta f = \frac{2f_c}{N} = \frac{2}{4\cdot 4} = \frac{1}{8}$, d.h. die FOURIER-Transformierte weist nun nicht-verschwindende Komponenten bei den Indizes (f_0, f_1, f_3) entsprechend den Frequenzen $f = 0, \frac{1}{8}, -\frac{1}{8}$ auf, allerdings haben sich die Komponenten verändert (s.u.)

Test mittels diskreter PARSEVAL-Relation:

$$\sum |h(t_k)|^2 = 2\cdot(3+\sqrt{2})^2 + 2\cdot(3-\sqrt{2})^2 = 44 \stackrel{!}{=} \frac{1}{N}\sum |H_n|^2 = \frac{(12^2 + 8\cdot|1-\mathrm{i}|^2 + 8\cdot|1+\mathrm{i}|^2)}{8}.$$

Die **Rekonstruktion des Signals** ergibt

$$\begin{aligned} a_0^* &= \frac{2}{4}\cdot 12 = 6 \\ a_1^* &= \frac{2}{4}\cdot 2\sqrt{2} = \sqrt{2} \\ b_1^* &= \frac{2}{4}\cdot 2\sqrt{2} = \sqrt{2}, \end{aligned}$$

so dass das rekonstruierte Signal durch

$$h(t) = 3 + \sqrt{2}\big(\cos(\frac{\pi}{4}t) + \sin(\frac{\pi}{4}t)\big)$$

gegeben ist. Der (vermeintliche) Unterschied zu oben lässt sich dadurch aufklären, dass die zweite Messreihe bei einem unterschiedlichen Anfangszeitpunkt beginnt (gegenüber der ersten um $\pi/4$ versetzt); mit

$$2\sin(\frac{\pi}{4}t + \frac{\pi}{4}) = \sqrt{2}\big(\cos(\frac{\pi}{4}t) + \sin(\frac{\pi}{4}t)\big)$$

sind damit beide Darstellungen, bis auf die Anfangsphase, identisch. Wir sehen also, dass sich das Signal auch durch das Abtasten mit einer geringer Rate rekonstruieren lässt, wobei die durchzuführende Analyse natürlich auch mit geringerem Aufwand verküpft ist.

6.3 Zusammenhang FOURIER-Transformation und FOURIER-Polynom; FFT zur Berechnung von FOURIER-Koeffizienten

In Kap. 6.1 hatten wir das FOURIER-Polynom

$$g_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n} \left(a_k \cos(kx) + b_k \sin(kx)\right)$$

kennengelernt, wobei mit den Koeffizienten

$$a_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(kx) \mathrm{d}x$$

$$b_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(kx) \mathrm{d}x$$

die quadratische Funktion

$$\|g_n(x) - f(x)\|^2$$

minimiert wird, falls $f(x)$ (reell vorausgesetzt) 2π-periodisch ist (vergleiche Gln. 6.2 bis 6.4).

Andererseits hatten wir die FOURIER-Transformation über

$$H(f) = \int_{-\infty}^{\infty} f(x) \mathrm{e}^{-2\pi \mathrm{i} f x} \mathrm{d}x$$

definiert. Sei nun $f(x)$ zwischen $0 \ldots 2\pi$ gegeben (und periodisch); die entsprechende FT lautet in diesem Fall

$$H(f) = \int_0^{2\pi} f(x) \mathrm{e}^{-2\pi \mathrm{i} f x} \mathrm{d}x.$$

Betrachtet man nun folgende Kombination der FOURIER-Koeffizienten,

$$a_k - \mathrm{i} b_k = \frac{1}{\pi} \int_0^{2\pi} f(x) \left(\cos(kx) - \mathrm{i} \sin(kx)\right) \mathrm{d}x = \frac{1}{\pi} \int_0^{2\pi} f(x) \mathrm{e}^{-\mathrm{i}kx} \mathrm{d}x$$

so lässt sich diese aus der Vorwärts-FT berechnen,

$$\pi(a_k - \mathrm{i} b_k) = \underbrace{\int_0^{2\pi} f(x) \mathrm{e}^{-2\pi \mathrm{i} f_k x} \mathrm{d}x}_{\text{FT (vorwärts) bzgl. } f_k} \qquad (k = 2\pi f_k)$$

Die **Numerische Berechnung** erfolgt analog über die FFT. In Satz 6.5 hatten wir gesehen, dass bei äquidistanten Stützstellen

$$x_j = \frac{2\pi}{N} \cdot j, \qquad (j = 0, \ldots, N-1)$$

das eindeutige, interpolierende FOURIER-polynom durch

$$g_n^*(x) = \frac{1}{2} a_0^* + \sum_{k=1}^{n-1} \left(a_k^* \cos(kx) + b_k \sin(kx)\right) + \frac{1}{2} a_n^* \cos(nx)$$

mit $n = N/2$ und den numerischen Näherungswerten

$$a_k^* = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\cos(kx_j) \qquad (k = 0,\ldots,N/2)$$
$$b_k^* = \frac{2}{N}\sum_{j=0}^{N-1} f(x_j)\sin(kx_j) \qquad (k = 1,\ldots,N/2-1)$$

gegeben ist; falls f(x) eine Sprungstelle bei x_0 aufweist, ist dabei $f(x_0) = \frac{1}{2}(f^+ + f^-)$ zu setzen.

Bezüglich dieser Stützstellen lautet die numerische Näherung der FOURIER-Komponenten

$$H_k = \sum_{j=0}^{N-1} f(x_j)\mathrm{e}^{-2\pi \mathrm{i}kj/N} = \qquad (x_j = \frac{2\pi}{N}j)$$
$$= \sum_{j=0}^{N-1} f(x_j)\left(\cos(kx_j) - \mathrm{i}\sin(kx_j)\right)$$

(vgl. Gl. 6.17). Damit ergeben sich zwei Möglichkeiten zur Berechnung der FOURIER-Koeffizienten, wenn wir im Weiteren N als Potenz von 2 voraussetzen.

(A) Einfache, aber langsamere Methode

Man bilde die FFT bezüglich $f(x_j)$, d.h. für N Stützstellen ergeben sich N komplexe Komponenten, wobei die übliche Indexkonvention

$$0 \le k^+ \le \frac{N}{2} < k^- \le N-1$$

eingehalten wird. Dann ergeben sich die gesuchten Koeffizienten zu

$$a_k^* = \frac{2}{N}\Re(H_k), \qquad (k = 0,\ldots,N/2)$$
$$b_k^* = -\frac{2}{N}\Im(H_k), \qquad (k = 1,\ldots,N/2-1) \qquad (6.21)$$

Man beachte, dass bei dieser Methode die Komponenten bei den negative Frequenzen zwar berechnet, aber nicht verwendet werden.

(B) Schnellere Methode

Um diese Redundanz zu vermeiden, lässt sich eine alternative Methode formulieren. Man definiere

$$y_j = f(x_{2j}) + \mathrm{i}f(x_{2j+1}), \qquad (j = 0,\ldots n-1,\ n = N/2) \qquad (6.22)$$

d.h. erzeuge aus der reellen Funktion f, definiert an N Stützstellen, die komplexe Funktion y, definiert an $N/2$ Stützstellen.

$$y_0 = f(x_0) + \mathrm{i}f(x_1) \qquad \text{bzw. } f(x) = \tfrac{1}{2}(f^+ + f^-) \text{ bei Sprungstellen}$$
$$y_1 = f(x_2) + \mathrm{i}f(x_3)$$
$$\vdots$$
$$y_{n-1} = f(x_{N-2}) + \mathrm{i}f(x_{N-1})$$

Danach transformiere man y,

$$H_k = \sum_{j=0}^{n-1} y_j \mathrm{e}^{-2\pi \mathrm{i} jk/n}, \qquad (k = 0, \ldots, n-1),$$

mittels FFT. Aus diesen Komponenten lassen sich die FOURIER-Koeffizienten a_k^*, b_k^* $(k = 0, \ldots, n)$ wie folgt berechnen[8].

$$n(a_k^* - \mathrm{i}b_k^*) = \tfrac{1}{2}(H_k + \bar{H}_{n-k}) + \tfrac{1}{2\mathrm{i}}(H_k - \bar{H}_{n-k})\mathrm{e}^{-\mathrm{i}\pi k/n}, \tag{6.23}$$

wenn $b_0^* = b_n^* = 0$ und $H_n = H_0$ gesetzt werden. ($\bar{H}$ ist hier der komplex konjugierte Wert von H). Das Vorgehen zur Bestimmung der Koeffizienten lautet damit folgendermaßen:

Man berechne

$$C_k = \tfrac{1}{2}(H_k + \bar{H}_{n-k}) - \tfrac{\mathrm{i}}{2}(H_k - \bar{H}_{n-k})\mathrm{e}^{-\mathrm{i}\pi k/n}$$

für $k = 0, \ldots, n$ und $H_n = H_0$. Daraus resultieren die gesuchten Größen

$$\begin{aligned} a_k^* &= \frac{1}{n}\Re(C_k), && (k = 0, \ldots, n) \\ b_k^* &= -\frac{1}{n}\Im(C_k), && (k = 1, \ldots, n-1) \end{aligned} \tag{6.24}$$

Auf diese Weise wird keine Information "verschenkt", und die Methode ist um einen Faktor von etwa zwei schneller als das Vorgehen (A). Man beachte, dass die trigonometrischen Größen $\mathrm{e}^{-\mathrm{i}\pi k/n}$ wiederum numerisch stabil berechnet werden müssen.

6.24 Beispiel (FOURIER-Koeffizienten der 2π-periodische x^2-Funktion). In Beispiel 6.4 und Abbildung 6.2 hatten wir die FOURIER-Polynome g_4 und g_{16} für die 2π-periodischen Funktion

$$f(x) = x^2 \qquad x = 0 \ldots 2\pi, \quad \text{periodisch},$$

betrachtet. Man erinnere sich, dass diese Funktion eine Sprungstelle bei $0, 2\pi, \ldots$ aufweist, so dass hier $f(x) = 2\pi^2$ gesetzt werden muss. Im Folgenden wollen wir nun die Koeffizienten des entsprechenden eindeutigen interpolierenden Polynomes über FFT berechnen, u.zw. für $N = 16$ Stützstellen und mit den beiden Methoden (A) und (B). Alle angegebenen Resultate wurden mit `single precision` ermittelt.

Methode A. Die Funktionswerte $y_i, i = 0, 15$ und die daraus mittels FFT berechneten FOURIER-Komponenten lauten

i	$\Re(y_i)$	$\Im(y_i)$	$\Re(H_k)$	$\Im(H_k)$
0	19.7392101	0.0000000	210.96280	0.0000000
1	0.1542126	0.0000000	32.4144211	99.2357101
2	0.6168503	0.0000000	8.4242344	47.6546631
3	1.3879131	0.0000000	3.9969778	29.5418015
4	2.4674013	0.0000000	2.4673982	19.7392082
5	3.8553145	0.0000000	1.7845042	13.1893101
6	5.5516524	0.0000000	1.4453807	8.1762466
7	7.5564170	0.0000000	1.2825289	3.9263840
8	9.8696051	0.0000000	1.2337036	0.0000000
9	12.4912186	0.0000000	1.2825174	-3.9263687
10	15.4212580	0.0000000	1.4453731	-8.1762409
11	18.6597233	0.0000000	1.7845067	-13.1893082
12	22.2066097	0.0000000	2.4673934	-19.7392082
13	26.0619259	0.0000000	3.9969728	-29.5418148
14	30.2256680	0.0000000	8.4242229	-47.6546669
15	34.6978264	0.0000000	32.4144096	-99.2357101

Die positiven Frequenzen (incl. der Null) liegen bei den Indizes 0...7, die negativen bei 9...15, und die (bei reellen Signalen) reelle Komponente der kritischen Frequenz $\pm f_c$ beim Index 8. Man beachte die "Symmetrie" der Komponenten, $H(-k) = [H(k)]^*$.

[8] siehe z.B. Schwarz, "Numerische Mathematik, Kap. 4.2.2

Die mittels Gl. 6.21 abgeleiteten FOURIER-Koeffizienten des interpolierenden Polynoms lauten dann

k	a_k^*	b_k^*
0	26.3703499	
1	4.0518026	-12.4044638
2	1.0530293	-5.9568329
3	0.4996222	-3.6927252
4	0.3084248	-2.4674010
5	0.2230630	-1.6486638
6	0.1806726	-1.0220308
7	0.1603161	-0.4907980
8	0.1542130	

Methode B. Die Funktionswerte $y_i, i = 0, 7$ (entsprechend Gl. 6.22, reelles Signal → komplexes Signal) und die daraus mittels FFT berechneten FOURIER-Komponenten lauten in diesem Fall ($n = 8$)

i	$\Re(y_i)$	$\Im(y_i)$	$\Re(H_k)$	$\Im(H_k)$
0	19.7392101	0.1542126)	106.09825	104.86456
1	0.6168503	1.3879131	-36.7630386	42.2965240
2	2.4674013	3.8553145	-17.2718067	2.4674015
3	5.5516524	7.5564156	-6.3075333	-11.1396198
4	9.8696051	12.4912186	2.4673958	-19.7392159
5	15.4212580	18.6597233	12.0890207	-27.4921246
6	22.2066097	26.0619259	27.1414127	-37.0110168
7	30.2256680	34.6978340	70.4599686	-53.0128021

Die Komponente bei der kritischen Frequenz $\pm f_c$ (Index 4) ist nun komplex! Aus den Komponenten lassen sich die Koeffizienten über (6.23, $H_8 = H_0$!) und (6.24) ermitteln. Die Übereinstimmung mit Koeffizienten aus Methode (A) liegt im Rahmen der vorgegebenen Genauigkeit.

k	a_k^*	b_k^*
0	26.3703518	
1	4.0518031	-12.4044628
2	1.0530295	-5.9568329
3	0.4996226	-3.6927273
4	0.3084248	-2.4674020
5	0.2230639	-1.6486645
6	0.1806720	-1.0220315
7	0.1603143	-0.4907985
8	0.1542091	

6.4 TSCHEBYSCHEFF-Interpolation

Motivation. Es sei daran erinnert (vgl. Kap. 2), dass sich der Interpolationsfehler der Polynominterpolation bei Verwendung von Stützstellen $x_0, \ldots, x_n$ über

$$|f(x) - P_n(x)| \leq \frac{\max|f^{(n+1)}(\xi)|}{(n+1)!} \left| \prod_{i=1}^{n} (x - x_i) \right|, \qquad (\xi \in [a, b])$$

abschätzen lässt. Der Fehler hängt also weitgehend von der Funktion

$$\phi(x) = \prod_{i=0}^{n} (x - x_i)$$

ab. Für *äquidistante* Stützstellen und größere n oszilliert diese Funktion sehr stark an den Rändern der Intervalls $[x_0, x_n]$, und ihr Wert ist viel größer als in mittleren Bereichen (Abb. 6.13).

Damit ist der Interpolationsfehler am größten an den Rändern; durch die Verwendung spezieller, zu den Rändern hin konzentrierten Stützstellen lässt sich dieses Problem jedoch beheben. Als besonders geeignet erweisen sich hierbei die Extremal- oder Nullstellen der sog. TSCHEBYSCHEFF- oder T-Polynome (vgl. Abb. 6.14). Im Folgenden wollen wir die Eigenschaften dieser Polynome vorstellen und die darauf basierenden Funktionenapproximationen und Interpolationsmethoden diskutieren.

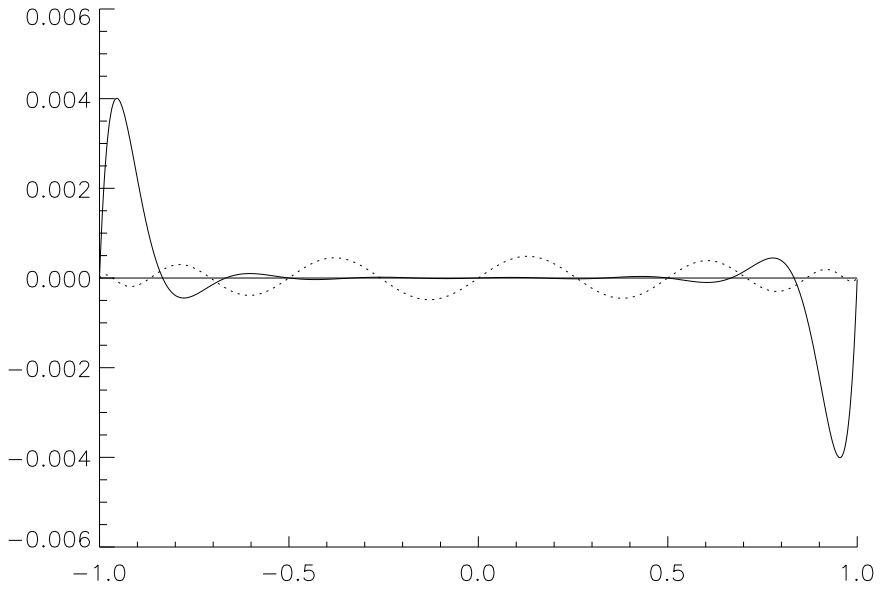


Abbildung 6.13: $\phi(x)$ (Seite 6-30) als Funktion von x für 13 äquidistante Stützstellen (durchgezogen) und für 13 nicht äquidistante Stützstellen, basierend auf den 13 Extremalstellen des T-Polynoms $T_{12}(x)$ (Gl. 6.28, gepunktet). Während bei äquidistanten Stützstellen der Interpolationsfehler an den Enden des Intervalles stark und oszillierend anwächst, lässt sich durch Wahl geeigneter Stützstellen, insbesondere bei Verwendung der Extremal- oder Nullstellen von T-Polynomen, ein besser nivellierter Verlauf erzielen, und auch der maximale Fehler wird kleiner.

6.4.1 Eigenschaften der TSCHEBYSCHEFF-Polynome

Die T-Polynome basieren auf der trigonometrische Identität

$$\cos[(n+1)\varphi] + \cos[(n-1)\varphi] = 2\cos(\varphi)\cos(n\varphi), \qquad (n \in \mathbb{N}) \tag{6.25}$$

Startend mit $n = 1$, ergeben sich somit sukzessive folgende Relationen

$$\begin{aligned}
\cos(2\varphi) &= 2\cos^2(\varphi) - 1 \\
\cos(3\varphi) &= 2\cos(\varphi)\cos(2\varphi) - \cos(\varphi) \\
&= 4\cos^3(\varphi) - 3\cos(\varphi) \\
\cos(4\varphi) &= 8\cos^4(\varphi) - 8\cos^2(\varphi) + 1.
\end{aligned}$$

Für $n \in \mathbb{N}_0$ ist $\cos(n\varphi)$ also als Polynom n-ten Grades in $\cos(\varphi)$ darstellbar, und das sog. n-te TSCHEBYSCHEFF- oder T-Polynom $T_n(x)$ ist definiert durch

$$\cos(n\varphi) =: T_n(\cos(\varphi)) = T_n(x), \qquad x = \cos(\varphi), \qquad (n \in \mathbb{N}_0) \tag{6.26}$$

Aufgrund $x = \cos(\varphi)$ ist x auf den Definitionsbereich $[-1, 1]$ beschränkt[9]. Die ersten vier T-Polynome lauten

$$\begin{aligned}
T_0(x) &= 1, & T_1(x) &= x, \\
T_2(x) &= 2x^2 - 1, & T_3(x) &= 4x^3 - 3x, \\
T_4(x) &= 8x^4 - 8x^2 + 1
\end{aligned}$$

[9] kann ggf. aber auf den gesamten $\mathbb{R}$ erweitert werden.

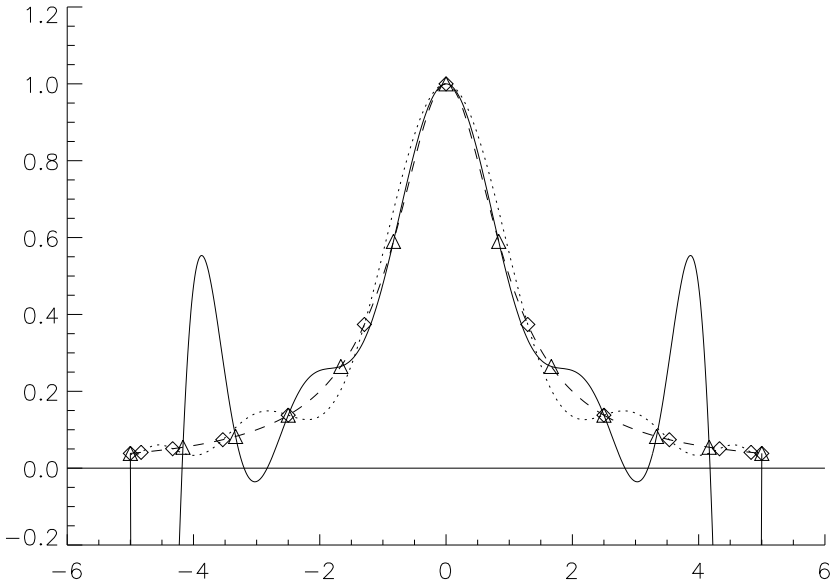


Abbildung 6.14: RUNGE-Funktion $(1+x^2)^{-1}$ (gestrichelt) und dazugehörige Polynominterpolation bei 13 Stützstellen (vgl. Kapitel 2). Die durchgezogene Kurve zeigt das interpolierende Polynom (zwölften Grades) bei äquidistanten Stützstellen (Dreiecke), und die gepunktete Kurve das entsprechende Polynom mit den Stützstellen (Diamanten) bzgl. der Extremalstellen von $T_{12}(x)$, skaliert auf den Bereich $[-5,5]$ (vgl. Abb. 6.13). Aufgrund der stärkeren Konzentration zu den Enden hin und des damit verbundenen verbesserten Verhaltens von $\phi(x)$ wird der (bzgl. äquidistanten Stützstellen völlig inakzeptable) Interpolationsfehler an den Intervallenden erheblich vermindert.

Insbesondere lassen sich die T-Polynome durch folgende *Rekursion* erzeugen,

$$\begin{aligned} T_{n+1}(x) =& 2xT_n(x) - T_{n-1}(x), \qquad (n \geq 1) \\ \text{wenn} \qquad T_0(x) =& 1, \quad T_1(x) = x \end{aligned} \tag{6.27}$$

explizit gesetzt werden. Aus ihrer Definition, $T_n(x) = \cos(n\varphi)$ folgt sofort, dass

$$|T_n(x)| \leq 1, \qquad (n \in \mathbb{N}_0,\ x \in [-1,1]),$$

und die Extremalstellen der T-Polynome lassen sich aus $n\varphi = k\pi$, $k = 0,\ldots,n$ (für $x \in [-1,1]$) berechnen, d.h.

$$x_k^{(e)} = \cos\left(\frac{k\pi}{n}\right), \qquad (k = 0,\ldots,n,\ n \geq 1). \tag{6.28}$$

Insgesamt sind dies $n+1$ Extrema, die zu den Enden des Intervalls $[-1,1]$ konzentriert sind (vgl. Abb. 6.13). Die n Nullstellen folgen analog aus $n\varphi = (2k-1)\frac{\pi}{2}$, $k = 1,\ldots,n$, d.h.

$$x_k^{(o)} = \cos\left(\frac{2k-1}{n}\frac{\pi}{2}\right) \qquad (k = 1,\ldots,n,\ n \geq 1). \tag{6.29}$$

Letztlich lautet der führende Koeffizient des n-ten T-Polynomes 2^{n-1} $(n \geq 1)$, und die Symmetrierelation ist durch

$$T_n(-x) = (-1)^n T_n(x), \qquad n \geq 0$$

gegeben. Die folgende Abbildung zeigt den Verlauf der ersten elf T-Polynome (bis auf $T_0(x) = 1$) im Intevall $[0,1]$.

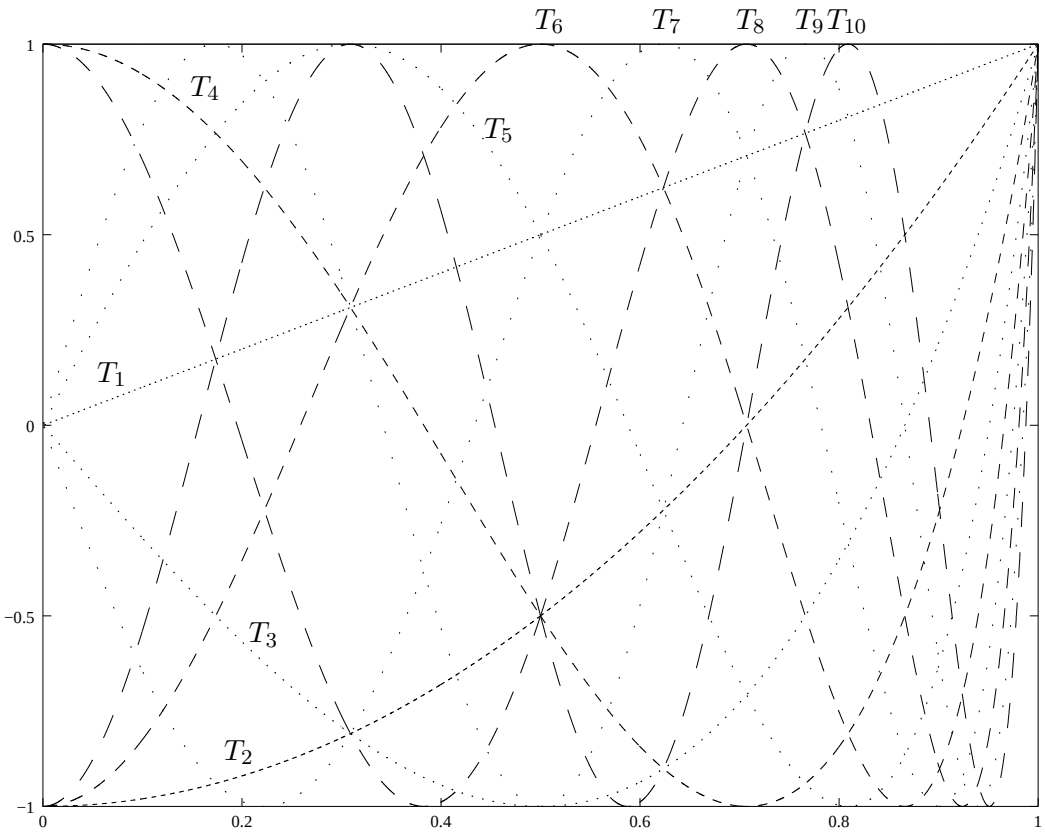


Abbildung 6.15: TSCHEBYSCHEFF-Polynome $T_1(x)$ bis $T_{10}(x)$ in Intervall $[0, 1]$. Der Verlauf im negativen Bereich $[-1, 0]$ ergibt sich aus der Symmetrierelation $T_n(-x) = (-1)^n T_n(x)$.

Genau wie die Basisfunktionen der FOURIER-Polynome ($\cos(kx), \sin(kx)$, vgl. (6.3)) erfüllen auch die TSCHEBYSCHEFF-Polynome bestimmte Orthogonalitätsrelationen, die sie zur Funktionenapproximation tauglich machen.

6.25 Satz. *Die T_n-Polynome ($n \in \mathbb{N}_0$) bilden ein System orthogonaler Polynome bezüglich des Intervalles $[-1, 1]$ und der Gewichtsfunktion* $\dfrac{1}{\sqrt{1-x^2}}$,

$$\int_{-1}^{1} T_k(x) T_j(x) \frac{\mathrm{d}x}{(1-x^2)^{1/2}} = \begin{cases} 0 & \text{für } k \neq j \\ \frac{\pi}{2} & \text{für } k = j > 0 \\ \pi & \text{für } k = j = 0 \end{cases} \qquad (k, j \in \mathbb{N}_0.) \tag{6.30}$$

(Beweis durch Nachrechnen.)

6.4.2 TSCHEBYSCHEFF-Entwicklung

Aufgrund dieser Orthogonalität lässt sich nun (in Analogie zur FOURIER-Entwicklung) die Aufgabe lösen, eine im Intervall $[-1, 1]$ gegebene stetige Funktion durch TSCHEBYSCHEFF-Polynome zu approximieren,

$$g_n(x) = \frac{1}{2} c_0 T_0(x) + \sum_{k=1}^{n} c_k T_k(x). \tag{6.31}$$

Hier muss man allerdings fordern, dass $\|f - g\|$ minimal *bezüglich der Gewichtsfunktion* $\dfrac{1}{\sqrt{1-x^2}}$ wird.

Führt man diese Minimierung (auf die übliche Weise) durch, so ergeben sich die Koeffizienten der T-Entwicklung als

$$c_k = \frac{2}{\pi} \int_{-1}^{1} f(x) T_k(x) \frac{\mathrm{d}x}{\sqrt{1-x^2}}, \qquad k = 0, \ldots, n. \tag{6.32}$$

Führen wir nun die Substitution $x = \cos\varphi$, $\mathrm{d}x = -\sin\varphi \mathrm{d}\varphi$, $\mathrm{d}\varphi = \dfrac{-\mathrm{d}x}{\sqrt{1-x^2}}$ durch, so ergibt sich folgende vereinfachte Darstellung

$$\begin{aligned} c_k &= \frac{2}{\pi} \int_{0}^{\pi} f(\cos\varphi) \underbrace{\cos(k\varphi)}_{T_k(\cos\varphi)} \mathrm{d}\varphi \\ &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(\cos\varphi) \cos(k\varphi) \mathrm{d}\varphi && \text{(auf Grund der Symmetrie)} \\ &= \frac{1}{\pi} \int_{0}^{2\pi} f(\cos\varphi) \cos(k\varphi) \mathrm{d}\varphi && k = 0, \ldots, n \end{aligned} \tag{6.33}$$

Die Koeffizienten der TSCHEBYSCHEFF-Entwicklung sind somit nichts anderes als die FOURIER-Koeffizienten bzgl. der geraden, 2π-periodischen Funktion $F(\varphi) = f(\cos\varphi)$ (vgl. Gl. 6.4).

Der große Vorteil der T-Entwicklung liegt nun darin, dass die Entwicklung fast immer schnell konvergiert, wie folgender Satz zeigt.

6.26 Satz. *Falls $f(x), x \in [-1, 1]$ stetig und einmal stückweise differenzierbar ist und die Reihe der T-Koeffizienten $\sum_{k=1}^{\infty} |c_k|$ konvergiert, dann konvergiert auch die* TSCHEBYSCHEFF*-Entwicklung*

$$g(x) = \frac{1}{2} c_0 T_0(x) + \sum_{k=1}^{\infty} c_k T_k(x)$$

gleichmäßig *gegen $f(x)$, und es gilt*

$$|f(x) - g_n(x)| \leq \sum_{k=n+1}^{\infty} |c_k| \qquad \forall x \in [-1, 1] \tag{6.34}$$

Der Beweis diese Satzes benutzt hauptsächlich die Eigenschaft der T-Polynome, dass $|T_k(x)| \leq 1$ ist. Normalerweise bilden nun die Koeffizienten c_k tatsächlich eine rasch konvergente Nullfolge, so dass de facto nur wenige Koeffizienten benötigt werden, um $f(x)$ sehr gut zu approximieren! Dies macht die T-Entwicklung zum Mittel der Wahl, komplizierte Funktionen auf einfache Weise zu nähern, vgl. Beispiel 6.1. Die Berechnung der numerischen Näherungswerte für die Koeffizienten erfolgt wiederum über Trapezintegration,

$$\begin{aligned} c_k^* &= \frac{2}{N} \sum_{j=0}^{N-1} f(\cos\varphi_j) \cos(k\varphi_j) \\ \varphi_j &= \frac{2\pi}{N} \cdot j && N \in \mathbb{N},\ k = 0, \ldots, N/2 \end{aligned}$$

und ist identisch zur Berechnung von FOURIER-Koeffizienten, *wenn wir äquidistante φ auf dem Intervall $[0, 2\pi]$ benützen und die zu approximierende Funktion f bzgl. des Argumentes $\cos(\varphi)$ auswerten.*

Meist wird es natürlich der Fall sein, dass $f(x)$ nicht auf dem Intervall $[-1,1]$ zu approximieren ist, sondern auf einem Intervall $[a,b]$. Eine einfache Variablentransformation erlaubt diese Verallgemeinerung. Wenn

$$f(x),\ x \in [a,b]$$

gegeben ist, dann lässt sich über

$$y = \frac{x-(a+b)/2}{(b-a)/2}$$

die Funktion $f(y)$ definieren, so dass

$$f(y),\ y \in [-1,1]$$

im für die T-Entwicklung "erlaubten" Bereich liegt. Inversion dieser Substitution resultiert in

$$x = y\frac{b-a}{2} + \frac{a+b}{2},$$

und unter Verwendung von $y := \cos\varphi$ ist die Funktion $f(\cos\varphi)$ nun über

$$f(\cos\varphi) := f\left(\cos\varphi \cdot 0.5(b-a) + 0.5(a+b)\right) \tag{6.35}$$

definiert. Aufgrund der Substitution ist jetzt $f(y)$ tatsächlich auf dem "erlaubten" Intervall $y \in [-1,1]$ definiert, und die Voraussetzung für die T-Entwicklung und ihre Konvergenz (Satz 6.26) ist erfüllt. Bis auf diese Substitution ändert sich an der Vorgehensweise nichts, insbesondere muss weiterhin über $\cos(k\varphi)\mathrm{d}\varphi$ mit $\varphi_j = (2\pi j/N)$ "integriert" werden.

6.27 Beispiel (T-Entwicklung der Exponentialfunktion). Als Beispiel wollen wir die Funktion $f(x) = \mathrm{e}^x$ im Intervall $[-5, 10]$ mittels T-Entwicklung approximieren. Man beachte, dass der Wertebereich dieser Funktion über ca. sechseinhalb Dekaden varriert, laut Satz 6.26 jedoch eine *gleichmäßige* Konvergenz zu erwarten ist, *wenn* die Reihe, gebildet aus den (absoluten) Koeffizienten, konvergiert. Laut (6.35) berechnen sich diese über

$$c_k^* = \frac{2}{N} \sum_{j=0}^{N-1} \exp\left(\cos\varphi_j \cdot 7.5 + 2.5\right) \cdot \cos(k\varphi_j) \qquad k = 0, \ldots, N/2,$$

wenn

$$\varphi_j = \frac{2\pi}{N} \cdot j, \qquad j = 0, \ldots, N-1$$

gesetzt wird. Wir lösen dieses Problem über FFT entsprechend Kap. 6.3 (man vergleiche mit Beispiel 6.24), wobei wir die Entwicklung bis T_{16} durchführen wollen. Demzufolge sind (unter Verwendung der "langsamen" Methode (A) zur Bestimmung der FOURIER-Koeffizienten) $N = 32$ Stützstellen im Intervall $[0, 2\pi]$ zu verwenden (der letzte Wert bei 2π entfällt auf Grund der Periodizität). Folgender `output` protokolliert die Ergebnisse einer Rechnung in `double precision`.

j	φ_j	$\Re[f(cos\varphi_j)]$	$\Im[f(cos\varphi_j)]$
0	0.0000000000000000	2.2026464843750000E+04	0.
1	0.1963495408493621	1.9070341796875000E+04	0.
2	0.3926990816987241	1.2445277343750000E+04	0.
3	0.5890486225480862	6.2230913085937500E+03	0.
4	0.7853981633974483	2.4486713867187500E+03	0.
5	0.9817477042468103	7.8585852050781250E+02	0.
6	1.1780972450961724	2.1488989257812500E+02	0.
7	1.3744467859455345	52.6242713928222656	0.
8	1.5707963267948966	12.1824941635131836	0.
9	1.7671458676442586	2.8202416896820068	0.
10	1.9634954084936207	0.6906474828720093	0.
11	2.1598449493429825	0.1888548135757446	0.
12	2.3561944901923448	6.0609668493270874E-02	0.
13	2.5525440310417071	2.3848783224821091E-02	0.
14	2.7488935718910690	1.1925259605050087E-02	0.
15	2.9452431127404308	7.7824071049690247E-03	0.
16	3.1415926535897931	6.7379469983279705E-03	0.
17	3.3379421944391554	7.7824071049690247E-03	0.
18	3.5342917352885173	1.1925259605050087E-02	0.
19	3.7306412761378791	2.3848783224821091E-02	0.
20	3.9269908169872414	6.0609668493270874E-02	0.
21	4.1233403578366037	0.1888548135757446	0.
22	4.3196898986859651	0.6906474828720093	0.
23	4.5160394395353274	2.8202416896820068	0.
24	4.7123889803846897	12.1824941635131836	0.
25	4.9087385212340520	52.6242713928222656	0.
26	5.1050880620834143	2.1488989257812500E+02	0.
27	5.3014376029327757	7.8585852050781250E+02	0.
28	5.4977871437821380	2.4486713867187500E+03	0.
29	5.6941366846315002	6.2230913085937500E+03	0.
30	5.8904862254808616	1.2445277343750000E+04	0.
31	6.0868357663302239	1.9070341796875000E+04	0.

Man beachte, dass die Funktionswerte symmetrisch sind, aufgrund der Symmetrie des `Cosinus`. Die (Fast) FOURIER-Transformierte dieses "Signales" lautet dann

k	$\Re[H_k]$	$\Im[H_k]$
0	1.0453995343106566E+05	0.0000000000000000E+000
1	9.7297919656759739E+04	3.1255098011051530E-05
2	7.8593840229526220E+04	-3.8222496027540132E-04
3	5.5381204050294204E+04	-1.1595890174533885E-03
4	3.4288877034830533E+04	7.0903985780468304E-05
5	1.8806402086648246E+04	-7.8894820054919990E-04
6	9.2136732116302483E+03	-1.5585546678916762E-03
7	4.0645230504350025E+03	-2.0836831679570400E-03
8	1.6265609446573071E+03	0.0000000000000000E+000
9	5.9452377901002842E+02	-2.9696210156512914E-04
10	1.9969979916058583E+02	-4.4728438800134551E-04
11	61.9866824278142303	-4.5596667075642472E-04
12	17.8676332434624783	8.3431840018022285E-05
13	4.8044370784846251	3.2716196262372321E-04
14	1.2119054069626145	6.6682705996345959E-04
15	0.2991467858519172	1.0308997798591513E-03
16	0.1269308137707412	0.0000000000000000E+000
17	0.2991299128407263	1.0531910683386769E-04
18	1.2121927019761642	2.2757197305889321E-04
19	4.8052080864508753	3.1677884462721906E-04
20	17.8681196685429313	-7.0903985780468304E-05
21	61.9869968386574328	-1.8454599365735902E-04
22	1.9970087805798266E+02	-3.4547437152421256E-04
23	5.9452546229002610E+02	-4.3476733827729763E-04
24	1.6265609446573071E+03	0.0000000000000000E+000
25	4.0645238999817966E+03	1.6038789672020992E-04
26	9.2136741520911055E+03	6.0193737521785362E-04
27	1.8806402914065544E+04	1.2987768435825942E-03
28	3.4288877521255614E+04	-8.3431840018022285E-05
29	5.5381204860194237E+04	6.4633223158283570E-04
30	7.8593840378384688E+04	1.2372019794524292E-03
31	9.7297918332039117E+04	1.4875507263751864E-03

Der Imaginärteil der Transformierten entspricht einer "numerischen Null", da das Signal eine reelle, *symmetrische* Funktion ist (vgl. Kap. 6.5.1). Die Koeffizienten der T-Entwicklung c_k^* (mit $a_k = \Re(c_k^*)$ und $b_k = \Im(c_k^*)$) berechen sich dann über (6.21, Methode "A") zu

k	a_k	b_k
0	6.5337470894416037E+03	
1	6.0811199785474837E+03	-1.9534436256907206E-06
2	4.9121150143453888E+03	2.3889060017212582E-05
3	3.4613252531433877E+03	7.2474313590836781E-05
4	2.1430548146769083E+03	-4.4314991112792690E-06
5	1.1754001304155154E+03	4.9309262534324994E-05
6	5.7585457572689052E+02	9.7409666743229761E-05
7	2.5403269065218765E+02	1.3023019799731500E-04
8	1.0166005904108169E+02	-0.0000000000000000E+000
9	37.1577361881267763	1.8560131347820572E-05
10	12.4812374475366141	2.7955274250084095E-05
11	3.8741676517383894	2.8497916922276545E-05
12	1.1167270777164049	-5.2144900011263928E-06
13	0.3002773174052891	-2.0447622663982701E-05
14	7.5744087935163407E-02	-4.1676691247716224E-05
15	1.8696674115744827E-02	-6.4431236241196954E-05
16	7.9331758606713265E-03	

Wie ersichtlich, bilden die (reellen) Koeffizienten tatsächlich eine rasch konvergente Nullfolge (b_k "numerisch Null"). Die letzte Tabelle dieses Beispieles gibt nun den Funktionswert $f(x) = \exp(x)$ und die Differenz $g_n(x) - f(x)$ im vorgegebenen Intervall [-5,10] an. Der Fehler der Entwicklung liegt *überall* in der Größenordung von einigen 10^{-3} und kleiner, d.h. die Funktion wird, wie vorhergesagt, *gleichmäßig* approximiert (6.34).

x	$f(x) = \exp(x)$	$g_n(x) - f(x)$
-5.0000000000000000	6.7379469990854670E-03	3.9817631604065571E-03
-4.2500000000000000	1.4264233908999256E-02	1.9190504296953924E-03
-3.5000000000000000	3.0197383422318501E-02	-2.0005059839144565E-03
-2.7500000000000000	6.3927861206707570E-02	3.8127874240669879E-03
-2.0000000000000000	0.1353352832366127	-3.3200215379011189E-03
-1.2500000000000002	0.2865047968601900	-9.9494534186284067E-04
-0.5000000000000002	0.6065306597126333	4.1335749838812275E-03
0.2499999999999998	1.2840254166877412	-6.9449068300397876E-04
0.9999999999999998	2.7182818284590446	-4.0062746615383560E-03
1.7499999999999998	5.7546026760057289	1.4740664342438592E-03
2.4999999999999996	12.1824939607034679	4.0131239146745656E-03
3.2499999999999996	25.7903399171930516	-1.7244667276656855E-03
3.9999999999999996	54.5981500331442149	-3.7183523583408373E-03
4.7500000000000000	1.1558428452718766E+02	2.4619129276857166E-03
5.5000000000000000	2.4469193226422038E+02	3.3578546015746724E-03
6.2500000000000000	5.1801282466834198E+02	-3.4422437537386941E-03
7.0000000000000000	1.0966331584284585E+03	-1.2914054382235918E-03
7.7499999999999991	2.3215724146110547E+03	4.1993012659986562E-03
8.5000000000000000	4.9147688402991344E+03	-3.6817486061408999E-03
9.2500000000000000	1.0404565716560723E+04	3.0161404392856639E-03
10.0000000000000000	2.2026465794806718E+04	2.7860833615704905E-03

6.4.3 Die CLENSHAW-Rekursion

Eine effiziente und numerisch stabile Berechnung der T-Entwicklung $g_n(x)$ (bei gegebenen Koeffizienten c_k^*) erlaubt die sog. CLENSHAW-Rekursion. Da dieses Verfahren nicht nur im hier diskutierten Rahmen, sondern von generellem Interesse ist, soll es kurz vorgestellt werden. (Die Berechnung von $g_n(x)$ im vorangegangenen Beispiel erfolgte mittels CLENSHAW-Rekursion.)

Die Methode lässt sich immer dann anwenden, falls die Summe

$$f(x) = \sum_{k=0}^{N} c_k F_k(x) \tag{6.36}$$

bei gegegeben Koeffizienten c_k ausgewertet werden soll, *und* die Basisfunktionen F_k über eine Rekursionsformel

$$F_{k+1}(x) = \alpha(k,x)F_k(x) + \beta(k,x)F_{k-1}(x) \tag{6.37}$$

definiert werden können. Unter Verwendung der Hilfsgrößen

$$\begin{aligned} y_{N+2} &= 0 \\ y_{N+1} &= 0 \\ y_k &= \alpha(k,x)y_{k+1} + \beta(k+1,x)y_{k+2} + c_k \qquad k = N, N-1, \ldots, 1 \\ \Rightarrow c_k &= y_k - \alpha(k,x)y_{k+1} - \beta(k+1,x)y_{k+2} \end{aligned} \tag{6.38}$$

lässt sich die Summe nämlich wie folgt darstellen:

$$
\begin{aligned}
f(x) = \sum_{k=0}^{N} c_k F_k(x) = & \qquad \text{jetzt Einsetzen der } c_k \\
= [\ & y_N &] \cdot F_N(x) \\
+ [\ & y_{N-1} - \alpha(N-1,x) y_N &] \cdot F_{N-1}(x) \\
+ [\ & y_{N-2} - \alpha(N-2,x) y_{N-1} - \beta(N-1,x) y_N &] \cdot F_{N-2}(x) \\
+ & \quad \vdots \\
+ [\ & y_8 - \alpha(8,x) y_9 - \beta(9,x) y_{10} &] \cdot F_8(x) \\
+ [\ & y_7 - \alpha(7,x) y_8 - \beta(8,x) y_9 &] \cdot F_7(x) \\
+ [\ & y_6 - \alpha(6,x) y_7 - \beta(7,x) y_8 &] \cdot F_6(x) \\
+ & \quad \vdots \\
+ [\ & y_2 - \alpha(2,x) y_3 - \beta(3,x) y_4 &] \cdot F_2(x) \\
+ [\ & y_1 - \alpha(1,x) y_2 - \beta(2,x) y_3 &] \cdot F_1(x) \\
+ [\ & c_0 + \underbrace{\beta(1,x) y_2 - \beta(1,x) y_2}_{\text{letzte Gl. ergänzt}} &] \cdot F_0(x)
\end{aligned}
$$

Auf Grund der Rekursion (6.37) fallen in dieser Summe fast alle Terme heraus, z.B. die zu y_8 gehörigen in den "mittleren" drei Zeilen,

$$y_8[F_8 - \alpha(7,x)F_7 - \beta(7,x)F_6] = 0.$$

Alles, was übrig bleibt, stammt aus der letzten und vorletzten Zeile,

$$f(x) = c_0 F_0(x) + \beta(1,x) y_2 F_0(x) + y_1 F_1(x). \tag{6.39}$$

Damit ergibt sich folgende Vorgehensweise zur Berechnung von $f(x)$

- Man berechne y_k, $k = N, \ldots, 1$ aus den Koeffizienten c_k entsprechend (6.38).
- Daraus ergibt sich $f(x)$ mittels y_1, y_2, c_0 und der Basisfunktionen $F_0(x)$ und $F_1(x)$ (Gl. 6.39). Man beachte, dass nur diese beiden explizit benötigt werden!

Anwendung auf die Entwicklung durch TSCHEBYSCHEFF-Polynome. Im Falle der T-Polynome lautete die Rekursion (6.27)

$$T_{k+1} = 2x \cdot T_k(x) - T_{k-1}(x)$$

Übertragen auf die Verallgemeinerung (6.37) bedeutet dies,

$$\begin{aligned} \alpha(k,x) &= 2x & \qquad T_0(x) &= 1 \\ \beta(k,x) &= -1 & \qquad T_1(x) &= x, \end{aligned}$$

dass die Koeffizienten vom Index k unabhängig sind. Des Weiteren gilt es zu berücksichtigen, dass der CLENSHAW- Koeffizient c_0 dem T-Koeffiziententen $\frac{c_0}{2}$ entspricht (vgl. Gl. 6.36 mit 6.31). Damit ergibt sich aus (6.39) die für die T-Entwicklung mit T-Koeffizienten c_k gültige Darstellung

$$
\begin{aligned}
g_n(x) &= -y_2 + x y_1 + \frac{c_0}{2} \qquad\qquad (6.40) \\
&\text{mit} \\
y_k &= 2x y_{k+1} - y_{k+2} + c_k \qquad (k = N, \ldots, -1 \text{ und } y_{N+1} = y_{N+2} = 0)
\end{aligned}
$$

Die CLENSHAW-Rekursion ist effizient und (fast immer) stabil!

Algorithmus. CLENSHAW-Rekursion für die T-Entwicklung $g_n(x)$ mit Koeffizienten c_k:

```
x2 = 2 * x
y(n) = c(n)
y(n-1) = c(n-1) + x2 * c(n)
DO k = n-2, 0, -1
  y(k) = c(k) + x2 * y(k+1) - y(k+2)
END DO
g_n = 0.5 * (y(0) - y(2))
```

Die letzte Zeile entspricht in diesem Fall (und kann durch diese Zeile ersetzt werden)

```
g_n = 0.5 * c(0) + x * y(1) - y(2)
```

6.4.4 TSCHEBYSCHEFF-Interpolation

Neben der (endlichen) TSCHEBYSCHEFF-Entwicklung (6.31) ist oftmals auch ein bestimmtes *Interpolations*polynom zweckmäßig, z.B. im Falle von Funktionenapproximation, Differentation und Integration. Die Stützstellen dieses Interpolationspolynom sind die Nullstellen von T-Polynomen, die sog. "TSCHEBYSCHEFF-Abszissen". Insbesondere erhält man mit dieser Interpolationsmethode die *kleinstmöglichen* Schranke für den Interpolationsfehlers. Um dieses zu begründen, benötigen wir zunächst folgenden Satz-

6.28 Satz. *Unter allen Polynomen $P_n(x)$ vom Grad $n \geq 1$, deren Koeffizienten von x^n gleich 1 ist, hat $\frac{T_n(x)}{2^{n-1}}$ die kleinste Maximumsnorm im Intervall $[-1,1]$, d.h.*

$$\min_{P_n(x)} \left(\max_{x\in[-1,1]} |P_n(x)| \right) = \max_{x\in[-1,1]} \left| \frac{T_n(x)}{2^{n-1}} \right| = \frac{1}{2^{n-1}} \tag{6.41}$$

Zur Erinnerung sei noch einmal darauf hingewiesen, dass der führende Koeffizient von $T_n(x) = 2^{n-1}$ ist und dass $\max|T_n(x)| = 1$ (bzgl. des Intervalles [-1,1]) ist.

6.29 Beweis. *Der Beweis wird über einen Widerspruch geführt. Wir nehmen dazu an, dass ein Polynom $P_n(x) \neq \frac{T_n(x)}{2^{n-1}}$ mit führendem Koeffizienten 1 existiert, so dass*

$$\left|P_n(x)\right| < \frac{1}{2^{n-1}} \qquad \forall x \in [-1,1].$$

Unter dieser Annahme muss an den $n+1$ Extrema $x_k^{(e)}$ von $T_n(x)$ gelten,

$$\begin{aligned} P_n(x_0^{(e)}) &< \frac{T_n(x_0^{(e)})}{2^{n-1}} = \frac{1}{2^{n-1}} \\ P_n(x_1^{(e)}) &> \frac{T_n(x_1^{(e)})}{2^{n-1}} = -\frac{1}{2^{n-1}} \\ P_n(x_2^{(e)}) &< \frac{T_n(x_2^{(e)})}{2^{n-1}} = \frac{1}{2^{n-1}} \\ &\vdots \end{aligned}$$

Deshalb muss das Differenzpolynom $Q(x) = P_n(x) - \frac{T_n(x)}{2^{n-1}}$ an den $n+1$ Extremalstellen von T_n alternierende *Vorzeichen haben, d.h. es müssen mindestens n Nullstellen existieren!*

Der Grad des Differenzpolynoms $Q(x)$ kann aber maximal nur $n-1$ sein, da die führenden Koeffizienten von P_n und $T_n/2^{n-1}$ laut Voraussetzung gleich sind! Damit ergibt sich (Hautsatz der Algebra) ein Widerspruch zur Annahme, q.e.d.

Satz 6.28 wird sich nun bei der Betrachtung der kleinstmöglichen Schranke für den Interpolationsfehler als wesentlich herausstellen. In Kap. 2 hatten wir den bei einer Polynominterpolation auftretenden Fehler mittels

$$|f(x) - P_n(x)| \leq \frac{\max|f^{(n+1)}(\xi)|}{(n+1)!}\Big|(x-x_0)(x-x_1)\dots(x-x_n)\Big|$$

abgeschätzt, wobei das Maximum für alle ξ im Interpolationsbereich [a,b] zu nehmen war und dieser Fehler für alle $a \leq x \leq b$ galt. Im weiteren beschränken wir uns auf den Bereich $-1 \leq x \leq 1$.

Wir wollen jetzt die Stützstellen $x_0 \dots x_n$ so wählen, dass die Funktion $\phi(x)$ (vgl. Seite 6-30) in diesem Bereich minimiert wird

$$\max_{-1\leq x\leq 1} \Big| \underbrace{(x-x_0)\dots(x-x_n)}_{\phi(x)} \Big| = \text{min!}$$

Da $\phi(x)$ ein Polynom vom Grad $(n+1)$ mit führendem Koeffizienten 1 ist, folgt aus obigem Satz sofort, dass dieses Minimum dann angenommen wird, wenn die $(n+1)$ Stützstellen die Nullstellen von T_{n+1} sind, und der Wert dieses Minimums ist durch $\max|\phi(x)| = \frac{1}{2^n}$ gegeben!

Falls also die Stützstellen des Interpolationspolynom $P_n^*(x)$ (mit $n+1$ Stützstellen) die TSCHEBYSCHEFF-Abszissen zu T_{n+1} sind, ergibt sich die kleinstmögliche *Schranke* für den Interpolationsfehler

$$|f(x) - P_n^*(x)| \leq \frac{\max|f^{(n+1)}(\xi)|}{2^n(n+1)!} \qquad x \in [-1,1] \tag{6.42}$$

Beachte: Das Polynom $P_n^*(x)$ ist zwar nicht das "beste" im Sinne des sog. "Minmax-Polynoms", das unter allen Polynomen gleicher Ordnung die kleinste maximale Abweichung von der zu interpolierenden Funktion f(x) hat, kommt aber diesem sehr nahe (Faktor 3 und besser bzgl. des Fehlers)!

Es gilt also jetzt, das (eindeutige) interpolierende Polynom P_n^* zu den entsprechenden T-Abszissen (von T_{n+1}) zu konstruieren. Es stellt sich hier als besonders vorteilhaft heraus, dieses Polynom nach den T-Polynomen selbst zu entwickeln,

$$P_n^*(x) = \frac{1}{2}\gamma_0 T_0(x) + \sum_{k=1}^{n} \gamma_k T_k(x). \tag{6.43}$$

Die Koeffizienten ergeben sich dann aus den $n+1$ Interpolationsbedingungen

$$P_n^*(x_l) = f(x_l) \quad , \quad l = 1, \dots, n+1$$

$$x_l = \cos\left(\frac{2l-1}{n+1}\frac{\pi}{2}\right) = x_l^{(0)}.$$

Zur Berechnung der Koeffizienten γ_k benötigen wir die *diskreten* Orthogonalitätsbedingungen der TSCHEBYSCHEFF-Polynome

6.30 Satz. *Wenn $x_l, l = 1, \dots, n+1$ die n+ Nullstellen von $T_{n+1}(x)$ sind, gilt*

$$\sum_{l=1}^{n+1} T_k(x_l)T_j(x_l) = \begin{cases} 0 & \text{für } k \neq j \\ \frac{1}{2}(n+1) & \text{für } k = j > 0 \\ n+1 & \text{für } k = j = 0 \end{cases}, \qquad (0 \leq k,\ j \leq n) \tag{6.44}$$

Der Beweis erfolgt analog zu den diskreten Orthogonalitätsbedingungen der Basisfunktionen von FOURIER-Polynomen. Mit diesen Bedingungen ergeben sich die gesuchten Koeffizienten zu

$$\gamma_k = \frac{2}{n+1}\sum_{l=1}^{n+1} f(x_l)T_k(x_l) \qquad (k=0,\ldots n)$$
$$= \frac{2}{n+1}\sum_{l=1}^{n+1} f\left(\cos\left(\frac{2l-1}{n+1}\frac{\pi}{2}\right)\right)\cdot\cos\left(k\cdot\frac{2l-1}{n+1}\frac{\pi}{2}\right) \tag{6.45}$$

und lassen sich somit sehr einfach berechnen. Die Auswertung des interpolierenden Polynomes kann natürlich wiederum mittels CLENSHAW-Rekursion durchgeführt werden. Wir weisen ausdrücklich darauf hin, dass der Grad des Polynomes, n, (bei $n+1$ Stützstellen) beliebig sein kann und insbesondere keine Potenz von 2 sein muss, wie es der Fall ist, wenn man die Koeffizienten der T-*Entwicklung* über FFT berechnet (zumindest, wenn der FFT-Algorithmus mit Basis 2 arbeitet).

Zum Vergleich rekapitulieren wir hier nochmals diese Koeffizienten der TSCHEBYSCHEFF-*Entwicklung*:

$$c_k^* = \frac{2}{N}\sum_{j=0}^{N-1} f\left(\cos(\varphi_j)\right)\cos(k\varphi_j), \qquad \varphi_j = \frac{2\pi}{N}j \qquad k=0,\ldots,\frac{N}{2}$$

Wie ersichtlich, sind ca. doppelt so viele Stützstellen wie im Falle der T-Interpolation bis zur gleichen Ordnung, $n=\frac{N}{2}$, erforderlich. (Falls mit FFT gerechnet wird, muss N ausserdem eine Potenz von 2 sein, s.o.)

Abschließend weisen wir darauf hin, dass sich die TSCHEBYSCHEFF-Interpolation sehr gut zu Differentation und Integration von Funktionen verwenden lässt.

6.31 Beispiel (T-Entwicklung der Exponentialfunktion). Folgende Tabelle vergleicht die Koeffizienten der T-Entwicklung aus Beispiel 6.27 mit den entsprechenden Koeffizienten der T-Interpolation entsprechend (6.45).

k	c_k^*(T-Entwicklung)	γ_k(T-Interpolation)
0	6533.74708944160	6533.74714039634
1	6081.11991524672	6081.12007154975
2	4912.11500918558	4912.11512131640
3	3461.32528531150	3461.32534018100
4	2143.05482821754	2143.05484917160
5	1175.40016117037	1175.40016773129
6	575.854620965717	575.854625529881
7	254.032738419258	254.032766883469
8	101.660059041082	101.660127347304
9	37.1578517494493	37.1578285417872
10	12.4813119298780	12.4813388415623
11	3.87417131435393	3.87425826047734
12	1.11674061835083	1.11684770094192
13	0.300304199298580	0.300344088522110
14	7.569440316046894E-002	7.564564769821075E-002
15	1.860084169311449E-002	1.788054849738369E-002
16	7.912919469163171E-003	3.830923615567110E-003
	($N=32$ Stützstellen)	($n+1$=17 Stützstellen)

Offensichtlich ergben sich nur für große k Unterschiede in den Koeffizienten. Die Koeffizienten der T-Interpolation sind jedoch erheblich einfacher zu ermitteln!

Fehlerabschätzung mittels (6.42)

$$f(y) = \mathrm{e}^{y\cdot d+x_m}, \qquad y\in[-1,1], \quad d=\frac{b-a}{2}, \qquad x_m=\frac{a+b}{2}$$

$$|f(y)-P_n^*(y)| \le \frac{|\max\left(f^{(n+1)}(\xi)\right)|}{2^n(n+1)!}$$

$$\begin{aligned}\max\bigl(f^{(n+1)}(\xi)\bigr) &= \max_{y\in[-1,1]}\left(d^{n+1}\mathrm{e}^{yd+x_m}\right)\\ &= \left(\frac{1}{2}\right)^{n+1}(b-a)^{n+1}\mathrm{e}^{b}\end{aligned}$$

Damit ergibt sich

$$|f(y)-P_n^*(y)| \le \frac{(b-a)^{n+1}\mathrm{e}^{b}}{2^{2n+1}(n+1)!},$$

und für $n=16$ wird die Schranke $|f(y)-P_n^*(y)| \le 0.71$ für das gesamte Intervall vorhergesagt. Die tatsächlich maximale Abweichung von $1.3\cdot 10^{-3}$ ist noch erheblich kleiner.

6.5 Ergänzungen

6.5.1 Einige Eigenschaften der FOURIER-Transformation

In dieser Ergänzung tabellieren wir einige wichtige Eigenschaften der FOURIER-Transformation (vgl. Seite 6-11)

Sei ...		dann ist...
f(x)	reell	$F(-k) = (F(k))^*$
	imaginär	$F(-k) = -(F(k))^*$
f(x)	gerade	$F(-k) = F(k)$ gerade
	ungerade	$F(-k) = -F(k)$ ungerade
f(x)	reell und gerade	$F(k)$ reell und gerade
	reell und ungerade	$F(k)$ imaginär und ungerade
f(x)	imaginär und gerade	$F(k)$ imaginär und gerade
	imaginär und ungerade	$F(k)$ reell und ungerade

Falls $F(k)$ die FOURIER-Transformierte von $f(x)$ ist, dann korrespondieren folgende Paare

$$f(a \cdot x) \quad \Leftrightarrow \quad \frac{1}{|a|} F\left(\frac{k}{a}\right) \qquad \text{“Zeitskalierung”}$$

längere Zeiten $\hat{=}$ niedrigeren Frequenzen

$$F(a \cdot k) \quad \Leftrightarrow \quad \frac{1}{|a|} f\left(\frac{x}{a}\right) \qquad \text{“Frequenzskalierung“}$$

$$f(x - x_0) \quad \Leftrightarrow \quad F(k)\mathrm{e}^{-2\pi i k x_0} \qquad \text{Zeitverschiebung}$$

$$F(k - k_0) \quad \Leftrightarrow \quad f(x)\mathrm{e}^{2\pi i k_0 x} \qquad \text{Frequenzverschiebung}$$

Eine Änderung des Nullpunktes bewirkt also eine Phasenänderung der Transformierten.

6.5.2 Faltung

In vielen Situationen stellt sich das Problem, dass ein Signal $S(x')$ und eine sog. “Antwortfunktion“ $R(x, x')$ gegeben sind. R gibt dabei die Wahrscheinlichkeit an, dass ein bestimmter Anteil des Signals von “Kanal” x' im Kanal x auftaucht. Die Antwortfunktion “schmiert” also das Signal aus.

6.32 Beispiel (Auflösung eines Detektors). Ein stellares Spektrum $S(\nu')$ (ν Frequenz) werde mit einem Detektor (Spektograph) der endlichen “Auflösung” $\Delta\nu$ “beobachtet” (aufgenommen).

Die Antwortfunktion kann in diesem Fall durch

$$R(\nu, \nu') = R(\nu - \nu') = \frac{1}{\sqrt{\pi}\Delta\nu} \mathrm{e}^{-(\frac{\nu-\nu'}{\Delta\nu})^2}$$

genähert werden, wenn $\Delta\nu$ die Auflösung des Spektrographen ist (vgl. Skizze 6.17). Wenn $\Delta\nu$ größer als die Breite der einzelnen Spektrallinien ist, beobachtet man ein ausgeschmiertes Spektrum mit

$$S'(\nu) = \int_{-\infty}^{\infty} S(\nu') R(\nu - \nu')\, \mathrm{d}\nu' \qquad \text{“Faltung”}$$

Das Signal wird mit der Antwortfunktion *gefaltet*! Im Wesentlichen werden dabei, bedingt durch die endliche Auflösung, Beiträge von verschiedenen Signalen $S(\nu')$ bei einer Frequenz ν aufaddiert, hauptsächlich aus dem Bereich $\nu' = \nu \pm \Delta\nu$ (Abb. 6.16 unten).

Falls der Detektor eine sehr hohe Auflösung hat, kann die Antwortfunktion durch eine δ-Funktion approxmiert werden,

$$R(\nu - \nu') \approx \delta(\nu - \nu') \qquad \Rightarrow \qquad S'(\nu) \simeq S(\nu),$$

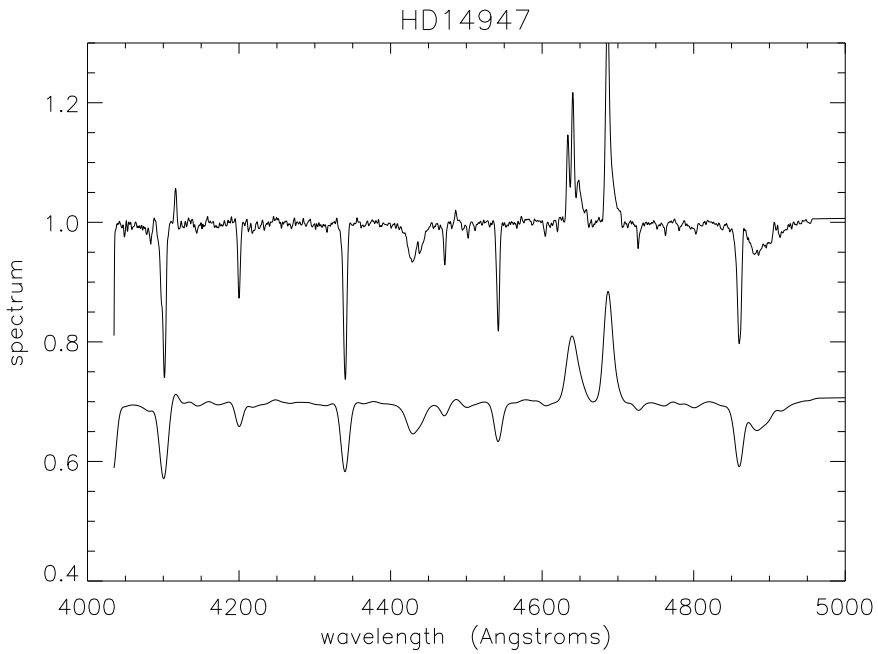


Abbildung 6.16: Oben: optisches Spektrum des heißen Überriesen HD 14947, beobachtet mit hoher Auflösung; Unten: das gleiche Spektrum, beobachtet mit 8-fach geringerer Auflösung (um -0.3 nach unten verschoben): das Signal wird "ausgeschmiert".

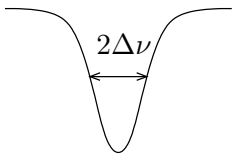


Abbildung 6.17: Auflösung eines Detektors: die "Antwortfunktion"

und das ursprüngliche Signal wird kaum verfälscht (Abb. 6.16 oben).

Will man nun die Beobachtung $S'(\nu)$ mit theoretischen Simulationen des Spektrums vergleichen, muss man diesen Sachverhalt berücksichtigen. Normalerweise muss dann das theoretische Signal mit der Antwortfunktion der Detektors gefaltet werden. Falls das Rauschen des beoachteten Signales gering ist, kann u. U. auch das beobachtete Signal "entfaltet" werden ($S' \to S$).

Falls $S(\nu')$ insgesamt N Frequenzpunkte und $R(\nu - \nu')$ M Frequenzpunkte im Bereich $R(\nu - \nu') \neq 0$ hat, erfordert diese Faltung entsprechend obiger Definition $N \cdot M$ wesentliche Operationen.

Unter Verwendung der FFT lässt sich der zur Faltung notwendige Rechenaufwand meistens erheblich reduzieren. Dazu benötigen wir folgenden

6.33 Satz (Faltungssatz). *Die* FOURIER-*Transformierte einer Faltung zweier Funktionen* $(g \circ h)$ *ist das Produkt der* FOURIER-*Transformierten der einzelnen Funktionen,* $G \cdot H$.

6.34 Beweis.

$$(g \circ h)(x) = \int_{-\infty}^{\infty} g(\xi) h(x - \xi) \mathrm{d}\xi$$

$$
\begin{aligned}
F_k(g \circ h) &= \int_{-\infty}^{\infty} \mathrm{d}x \mathrm{e}^{-2\pi i k x} \int_{-\infty}^{\infty} g(\xi) h(x-\xi) \mathrm{d}\xi \\
(y := x - \xi) \quad &= \int_{-\infty}^{\infty} \mathrm{d}y \int_{-\infty}^{\infty} \mathrm{e}^{-2\pi i k (y+\xi)} \cdot g(\xi) h(y) \mathrm{d}\xi \\
&= \Big(\int_{-\infty}^{\infty} \mathrm{e}^{-2\pi i k y} h(y) \mathrm{d}y \Big) \cdot \Big(\int_{-\infty}^{\infty} \mathrm{e}^{-2\pi i k \xi} g(\xi) \mathrm{d}\xi \Big) \\
&= H(k) \cdot G(k) \qquad \square.
\end{aligned}
$$

Aus diesem Satz ergibt sich folgendes Vorgehen zur schnellen Faltung zweier Funktionen mit Hilfe der (Fast) FOURIER-Transformation:

i) Man bilde G(k) und H(k) aus dem Signal g(x) und der Antwortfunktion h(x) mittels FFT. Die Antwortfunktion muss dabei zuvor (durch Zufügen von Nullen an den Rändern) auf die gleiche Zahl von Stützstellen wie das Signal ergänzt werden[10].

ii) Die resultierenden FOURIER-Komponenten werden dann multipliziert, und zwar $\forall k$, $k = 1 \ldots N$. Nach obigem Satz ist damit die Transformierte der Faltung, $(G \cdot H)(k)$, erzeugt. Diese (d.h. das Produkt) muss nur noch mittels

iii) inverser (Rückwärts-) Transformation auf die gesuchte Faltung rücktransformiert werden,

$$
(g \circ h)(x) = \int_{-\infty}^{\infty} \mathrm{d}k e^{2\pi i k x} (G \cdot H)(k)
$$

Da die FOURIER-Transformation über FFT $N \log_2 N$ Operationen kostet, ergibt sich somit eine Gesamtzahl von $N \cdot (1 + 3 \log_2(N))$ wesentlichen Operationen. Der erste Teil der Klammer entspricht dabei der Multiplikation $G_k \cdot H_k$, und der zweite Teil zwei Vorwärts- und einer Rückwärtstransformation.

Falls also $(1 + 3 \log_2(N)) < M$ (wenn M die ursprüngliche Zahl der Stützstellen der Anwortfunktion ist, s.o.), ist die hier vorgestellte Faltungsmethode schneller. Hätte man in obigem Beispiel die Faltung für 1000 Frequenzpunkte durchzuführen, wäre für $M > (1+3 \log_2(N)) \approx 30$ fast immer ein Zeitgewinn zu erzielen.

[10] Einzelheiten dazu findet man z.B. in "Numerical Recipes", Kap. 13.1

6.5.3 Spektrale Leistungsdichte bei "falscher" Abtastrate

In dieser Ergänzung wollen wir der Frage nachgehen, wie sich eine "falsche" (d.h. zu geringe) Abtastrate auf die "gemessene" spektrale Leistungsdichte auswirkt. Diese Untersuchung soll die entsprechenden Überlegungen in Kap. 6.2.3 komplementieren.

Dazu nehmen wir an, dass die *tatsächliche* Verteilung der FOURIER-Komponenten $H(f)$ eines beliebigen Signals $h(t)$ auf Grund einer Messreihe (1) bekannt sei. Diese Annahme setzt voraus, dass die maximale Frequenz des Signales $f_{\max}$ kleiner als die entsprechende *Nyquist*-Frequenz $f_c^{(1)} = \frac{1}{2\Delta}$ der Messung ist (vgl. Kap 6.2.3).

Dieses Signal soll nun in einer zweiten Messreihe (2) mit der halben Geschwindigkeit von (1) abgetastet werden, so dass jetzt $f_{\max} > f_c^{(2)} = \frac{1}{2} f_c^{(1)}$ ist.

Aufgrund des *Aliasing* werden nun die Komponenten mit positiven Frequenzen $f^+ = f_c^{(2)} + \epsilon$ in den Kanal $-f_c^{(2)} + \epsilon$ transformiert, beziehungsweise die Komponenten mit negativen Frequenzen $f^- = -f_c^{(2)} - \epsilon$ in den Kanal $f_c^{(2)} - \epsilon$ (obwohl i. A. $0 < \epsilon < 2f_c^{(2)}$ gilt, soll im Weiteren ϵ eine kleine positive Größe sein).

Die tatsächlichen Komponenten aus Messung (1) bezeichnen wir wie folgt (der Index "1" bezieht sich auf Komponenten innerhalb des Bereiches $[-f_c^{(2)}, f_c^{(2)}]$, der Index "2" auf die Komponenten außerhalb dieses Bereiches, vgl. Skizze 6.18)

$$
\begin{aligned}
H(f_c^{(2)} - \epsilon) &= H_1^+ \\
H(f_c^{(2)} + \epsilon) &= H_2^+ \\
H(-f_c^{(2)} + \epsilon) &= H_1^- \\
H(-f_c^{(2)} - \epsilon) &= H_2^-
\end{aligned}
$$

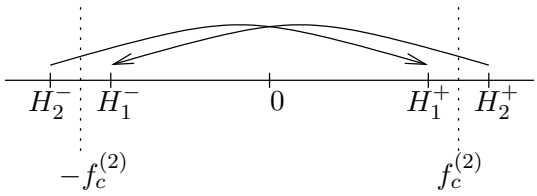


Abbildung 6.18: *Aliasing* der FOURIER-Kompnenten bei zu geringer Abtastrate, siehe Text.

Wie wirkt sich nun das *Aliasing* auf die mit Messung (2) erhaltenen FOURIER-Komponenten und die spektrale Leistungsdichte aus? Zunächst ergibt sich für die aus Messung (2) gewonnene Verteilung $H'(f)$

$$
\begin{aligned}
H'(f_c^{(2)} - \epsilon) &= H_1^+ + H_2^- \\
H'(-f_c^{(2)} + \epsilon) &= H_1^- + H_2^+
\end{aligned}
$$

(Addition der originalen und falsch übersetzten Komponenten), und bei $(f_c^{(2)} - \epsilon)$ resultiert eine spektrale Leistungsdichte

$$
\begin{aligned}
|H'(f_c^{(2)}-\epsilon)|^2 &= \left[\Re\left(H_1^+ + H_2^-\right]\right)^2 + \left[\Im\left(H_1^+ + H_2^-\right]\right)^2 \\
&= [\Re H_1^+]^2 + [\Im H_1^+]^2 && \leftarrow \quad P(H_1^+) \\
&+ [\Re H_2^-]^2 + [\Im H_2^-]^2 && \leftarrow \quad P(H_2^-) \\
&+ 2\left(\Re H_1^+ \cdot \Re H_2^- + \Im H_1^+ \cdot \Im H_2^-\right) \\
&\neq |H_1^+|^2 + |H_2^-|^2 && \leftarrow \quad P(H_1^+) + P(H_2^-)
\end{aligned}
$$

Demzufolge wird die Verteilung der spektralen Leistungsdichte durch *Aliasing* verfälscht. Insbesondere ist die gemessene Leistungsdichte *nicht* die Summe der Leistungsdichten von originaler und falsch übersetzter Komponente, sondern kann, gegenüber der Summe dieser Ausgangsgrößen, sowohl größer als auch kleiner werden! (Man vergleiche mit Beispiel 6.36)

Aliasing bei der NYQUIST-Frequenz.

Das Signal $h(t)$ soll nun reell sein, dann gilt $H(f) = (H(-f))^*$ (vgl. Kap. 6.5.1).

Damit ergibt sich $H_2^+ = (H_2^-)^*$, d.h. $\Re H_2^- = \Re H_2^+$, und $\Im H_2^- = -\Im H_2^+$. Der gemischte Term in obiger Gleichung,

$$2\left(\Re H_1^+ \cdot \Re H_2^- + \Im H_1^+ \cdot \Im H_2^-\right)$$

lautet dann

$$2\left(\Re H_1^+ \cdot \Re H_2^+ - \Im H_1^+ \cdot \Im H_2^+\right).$$

Wir lassen nun $\epsilon \to 0$, gehen, d.h. betrachten Frequenzen $f \to f_c^{(2)}$ sehr nahe der NYQUIST-Frequenz. In diesem Fall sind die Originalkomponenten $H_1^+ \approx H_2^+$ praktisch gleich, zumindest wenn ihre (originale) frequentielle Verteilung bei $f_c^{(2)}$ keinen Sprung aufweist. Die aus Messung (2) resultierende Leistungsdichte bei der NYQUIST-Frequenz lautet dann

$$
\begin{aligned}
P'(f_c^{(2)}) = \quad & |H'(f_c^{(2)})|^2 && \to \quad \text{(für } \epsilon \to 0\text{)} \\
= \Big\{ & [\Re H_1^+]^2 + [\Im H_1^+]^2 && \to \quad [\Re H_1^+]^2 + [\Im H_1^+]^2 \\
+ \quad & [\Re H_2^-]^2 + [\Im H_2^-]^2 && \to \quad [\Re H_1^+]^2 + [\Im H_1^+]^2 \\
+ 2 & \left(\Re H_1^+ \Re H_2^+ - \Im H_1^+ \Im H_2^+\right)\Big\} && \to \quad 2\left([\Re H_1^+]^2 - [\Im H_1^+]^2\right) \\
& && \quad = 4[\Re H_1^+]^2
\end{aligned}
$$

In Worten: Die spektrale Leistungsdichte bei der falsch gewählten NYQUIST-Frequenz entspricht dem vierfachen Quadrat des Realteils der entsprechenden originalen FOURIER-Komponente (vgl. Beispiel 6.35). Man beachte, dass zwar die verfälschten FOURIER-Komponenten bei der NYQUIST-Frequenz, $H'(f_c^{(2)})$, im Falle von reellen Signalen reell sein müssen (Seite 6-16), dass dies aber für die originalen Komponenten des Signales, $H_1^+(f_c^{(2)})$, nicht gilt, weil bzgl. Messung (1) eine andere (größere) NYQUIST-Frequenz vorliegt.

Falls allerdings das (reelle) Signal *gerade* ist, sind tatsächlich *alle* origininale FOURIER-Komponenten des Signales rein reell (→ Kap. 6.5.1) und wir finden,

$$P'(f_c^{(2)}) \quad = \quad 4[\Re H(f_c^{(2)})]^2 \quad \to \quad 4|H(f_c^{(2)})|^2 = 4P(f_c^{(2)}).$$

dass sich die verfälschte spektrale Leistungsdichte gegenüber dem Originalwert vervierfacht hat!

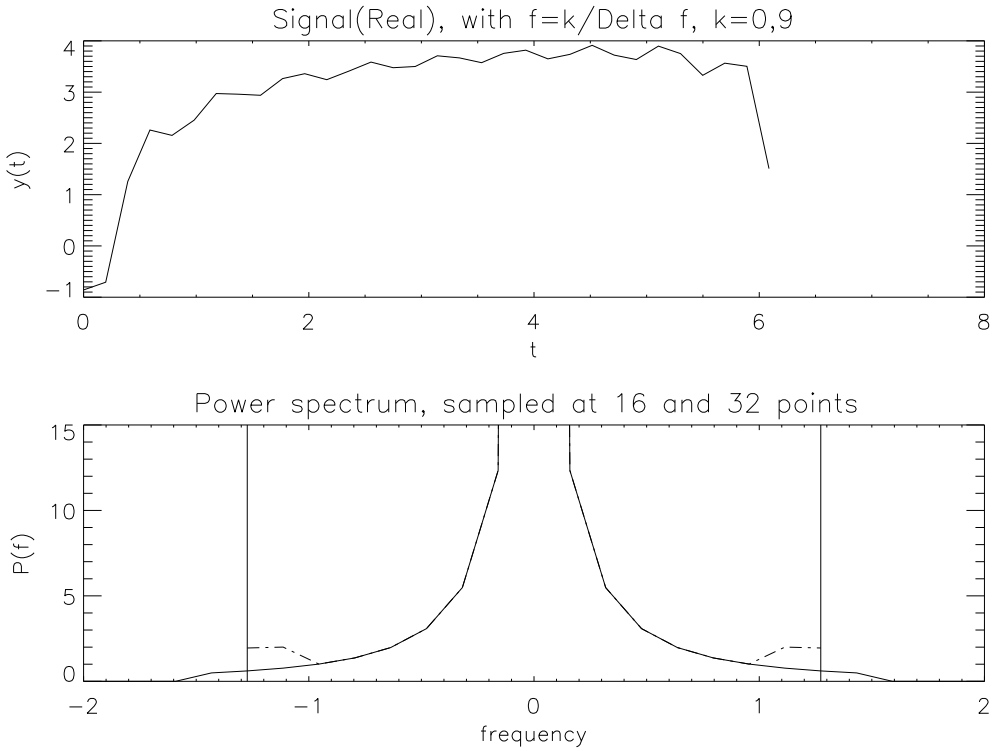


Abbildung 6.19: Signal (oben) und spektrale Leistungsdichte (unten) zu Beispiel 6.35. Durchgezogen ist die "tatsächliche" Leistungsdichte, wie sie mit $N = 32$ Messzeitpunkten resultiert, während das Resultat bzgl. $N = 16$ durch Strichpunkte dargestellt ist. Die "falsche" NYQUIST-Frequenz bzgl $N = 16, f_c^{(2)}$, ist mit einer Senkrechten markiert. Aus der Analyse der FOURIER-Komponenten lässt sich feststellen, dass $P'(f_c^{(2)})$ dem vierfachen Quadrat des Realteil des Originalwertes entspricht.

6.35 Beispiel (Reelles Signal, symmetrische Leistungsdichte). Als Beispiel für die soeben geschilderten Zusammenhänge betrachten wir zunächst ein reelles, asymmetrisches Signal, wobei nach Abtasten mit zu geringer Rate gelten sollte (und auch tatsächlich gilt)

$$P'(f_c^{(2)}) = 4[\Re H_{\text{act}}(f_c^{(2)})]^2,$$

wenn H_{act} die tatsächliche *Fourier*-Komponente bzgl. einer richtig gewählten Abtastrate mit $f_c^{(1)} = 2f_c^{(2)}$ ist.

Das zeitliche Signal laute dabei

$$S(t) = 5 - \sum_{k=0}^{9} \left[(\sin(2\pi k\Delta f \cdot t)) + (2\cos(2\pi k\Delta f \cdot t))/(k+1)\right],$$

d.h. bestehe aus 10 Frequenzen (inclusive der "0") mit geraden und ungeraden Komponenten, die mit k abklingen (vgl. Abb. 6.19 oben). Das Messintervall sei dabei durch $\Delta t = \frac{2\pi}{N}$ mit $N = 32$ bzw. $N = 16$ gegeben, und das Signal werde bei $t_i = i \cdot \Delta t$, $i = 0, \ldots N-1$ abgetastet.

Die kritische Frequenz liegt dann bei

$$f_c = \frac{1}{2\Delta t} = \frac{N}{4\pi},$$

und die Breite Δf eines Frequenzkanales ist (unabhängig von der Abtastrate) durch $\Delta f = \frac{2f_c}{N} = \frac{1}{N\Delta t} = \frac{1}{2\pi}$ gegeben. Dieser Wert soll ebenso die Frequenzen in obigem Signal ($\propto k\Delta f$) charakterisieren. Damit ergibt sich bei $N = 16$ Messpunkten ein *Aliasing* der Frequenzen $k = 8, \quad [8 \cdot \Delta f = 8/(2\pi) = 16/(4\pi) = f_c^{(2)}]$ und $k = 9$ (vgl. Abb. 6.19).

6.36 Beispiel (Komplexes Signal, asymmetrische Leistungsdichte). Als Beispiel für die Konsequenzen des "falschen" Abtastens eines beliebigen komplexen Signales betrachten wir einen zu (6.35) analogen Fall (d.h. gleiche Werte für $N, \Delta t, \Delta f$ etc.); "nur" das Signal sei jetzt durch

$$S(t) = 5 - \sum_{k=0}^{9} \left[(\sin(2\pi k\Delta f \cdot t)) + (2\cos(2\pi k\Delta f \cdot t)) + \mathrm{i}\,(3\sin(2\pi k\Delta f \cdot t)) + \mathrm{i}\left(\frac{1}{2}\cos(2\pi k\Delta f \cdot t)/(k+1)\right)\right]$$

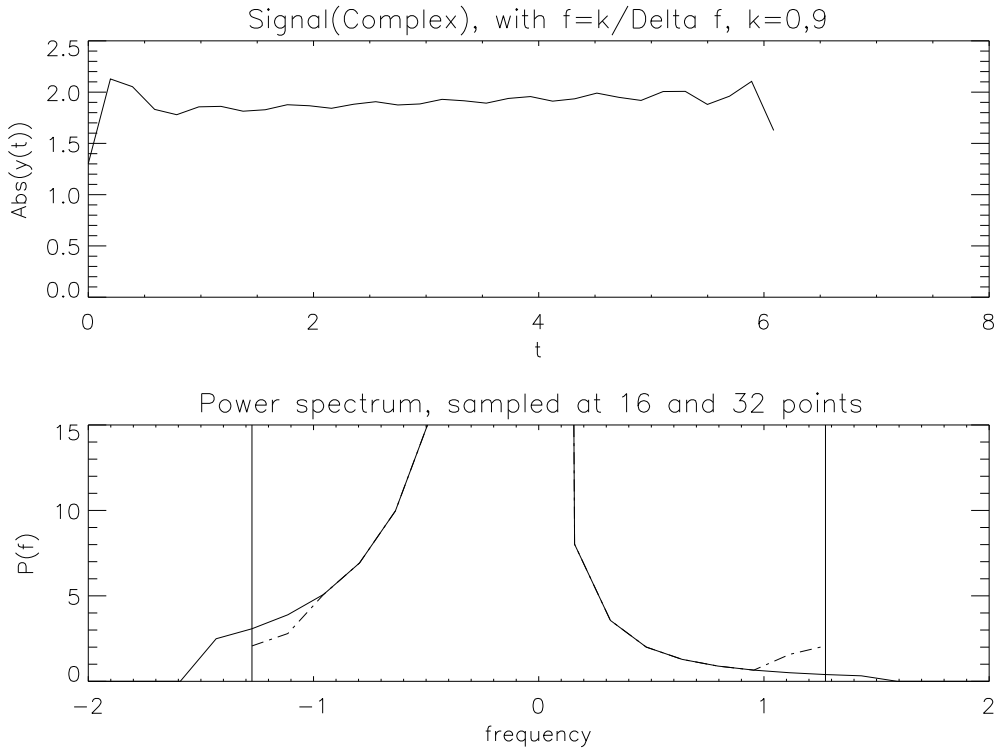


Abbildung 6.20: Wie Abb. 6.19, jedoch für das komplexes Signal aus Beispiel 6.36 (siehe Text).

gegeben. Wie aus Abb 6.20 ersichtlich, kann nun aufgrund des *Aliasing* die "tatsächliche" Leistungsdichte sowohl vergrößert als auch verkleinert werden. *Bei* der NYQUIST-Frequenz gilt allerdings weiterhin (wie immer), dass $P'(f_c^{(2)}) = P'(-f_c^{(2)})$ ist!

6.5.4 Fast-FOURIER-Transformation am Beispiel N=8

In einem der zentralen Abschnitte dieses Kapitels hatten wir gezeigt, dass die numerische FOURIER-Transformation für die einzelnen Komponenten zunächst folgendermaßen definiert ist (vgl. Seite 6-21):

$$H_n = \sum_{k=0}^{N-1} W_N^{n\cdot k} h_k, \qquad W_N = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}}.$$

Ausgehend von dieser Darstellung hatten wir in Kap. 6.2.5 mit Hilfe des DANIELSON-LANCZOS-Lemmas die FFT abgeleitet. In dieser Ergänzung wollen wir nun eine dazu alternative Ableitung vorstellen, die auf Grund eines etwas anderen Blickwinkels das Verständnis erleichtern sollte. Wir werden uns wiederum am Falle $N = 8$ ("8-Punkt-Transformation") orientieren.

In Matrixschreibweise lautet obige Gleichung

$$\begin{pmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ H_{N-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{1\cdot 1} & W_N^{1\cdot 2} & & W_N^{1\cdot (N-1)} \\ 1 & W_N^{2\cdot 1} & W_N^{2\cdot 2} & & W_N^{2\cdot (N-1)} \\ \vdots & & & & \vdots \\ 1 & W_N^{(N-1)\cdot 1} & W_N^{(N-1)\cdot 2} & \dots & W_N^{(N-1)\cdot (N-1)} \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N-1} \end{pmatrix}$$

Wie schon öfters angemerkt, wird die schnelle Abarbeitung obiger Transformation insbesondere dadurch ermöglicht, dass $W_N^{n\cdot k}$ periodisch mit Periode N ist. Ausnutzung dieser Periodizität bedeutet im Falle $N = 8$ für die 8-Punkt-Transformationmatrix Λ^8

$$\Lambda^8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ 1 & W_8^2 & W_8^4 & W_8^6 & 1 & W_8^2 & W_8^4 & W_8^6 \\ 1 & W_8^3 & W_8^6 & W_8^1 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\ 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 \\ 1 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^1 & W_8^6 & W_8^3 \\ 1 & W_8^6 & W_8^4 & W_8^2 & 1 & W_8^6 & W_8^4 & W_8^2 \\ 1 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1 \end{pmatrix}$$

Nun werden die Spalten der Matrix (und die dazugehörigen Elemente von h_k) derart vertauscht, dass zunächst alle geraden, dann alle ungeraden Indizes hintereinander angeordnet sind, d.h. die ursprüngliche Reihenfolge der Spalten Spalten $[0, 1, \ldots, 7]$ wird in die Reihenfolge $[0, 2, 4, 6, 1, 3, 5, 7]$ gebracht.

$$\begin{pmatrix} {}^8H_0 \\ {}^8H_1 \\ {}^8H_2 \\ {}^8H_3 \\ {}^8H_4 \\ {}^8H_5 \\ {}^8H_6 \\ {}^8H_7 \end{pmatrix} = \left(\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_8^2 & W_8^4 & W_8^6 & W_8^1 & W_8^3 & W_8^5 & W_8^7 \\ 1 & W_8^4 & 1 & W_8^4 & W_8^2 & W_8^6 & W_8^2 & W_8^6 \\ 1 & W_8^6 & W_8^4 & W_8^2 & W_8^3 & W_8^1 & W_8^7 & W_8^5 \\ \hline 1 & 1 & 1 & 1 & W_8^4 & W_8^4 & W_8^4 & W_8^4 \\ 1 & W_8^2 & W_8^4 & W_8^6 & W_8^5 & W_8^7 & W_8^1 & W_8^3 \\ 1 & W_8^4 & 1 & W_8^4 & W_8^6 & W_8^2 & W_8^6 & W_8^2 \\ 1 & W_8^6 & W_8^4 & W_8^2 & W_8^7 & W_8^5 & W_8^3 & W_8^1 \end{array}\right) \begin{pmatrix} h_0 \\ h_2 \\ h_4 \\ h_6 \\ h_1 \\ h_3 \\ h_5 \\ h_7 \end{pmatrix}$$

Die Matrix Λ_8 lässt sich nun durch 4 quadratische Untermatrizen der Ordnung 4×4 beschreiben, wobei die Untermatrizen "links oben" und "links unten" identisch sind. Zunächst bemerken wir, dass die Zeilen der der Untermatrix "rechts oben" alternativ wie folgt dargestellt werden können

$$\begin{array}{llll} (1 & 1 & 1 & 1) & = W_8^0 \cdot (1 & 1 & 1 & 1) \\ (W_8^1 & W_8^3 & W_8^5 & W_8^7) & = W_8^1 \cdot (1 & W_8^2 & W_8^4 & W_8^6) \\ (W_8^2 & W_8^6 & W_8^2 & W_8^6) & = W_8^2 \cdot (1 & W_8^4 & 1 & W_8^4) \\ (W_8^3 & W_8^1 & W_8^7 & W_8^5) & = W_8^3 \cdot (1 & W_8^6 & W_8^4 & W_8^2) \end{array} \qquad (A)$$

Die Untermatrix "links oben/unten" ist nichts anderes als die 4-Punkt-Transformationsmatrix Λ^4,

$$\Lambda^4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & W_8^2 & W_8^4 & W_8^6 \\ 1 & W_8^4 & 1 & W_8^4 \\ 1 & W_8^6 & W_8^4 & W_8^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & 1 & W_4^2 \\ 1 & W_4^3 & W_4^2 & W_4^1 \end{pmatrix}. \qquad (B)$$

Damit besteht die Matrix Λ^8 dann aus

i) "links oben" aus Λ^4

ii) "links unten" aus Λ^4

iii) "rechts oben" und Verwendung von (A)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & W_8^1 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{pmatrix} \cdot \Lambda^4 \quad =: \quad {}^8\mathrm{Diag}_4 \cdot \Lambda^4,$$

wenn wir mit ${}^8\mathrm{Diag}_4$ die 4×4 Diagonalmatrix bzgl. einer 8-Punkt-Transformation wie definiert bezeichnen.

iv) Geeignete Umformung die Untermatrix "rechts unten" ergibt schließlich

$$\begin{pmatrix} W_8^4 & W_8^4 & W_8^4 & W_8^4 \\ W_8^5 & W_8^7 & W_8^1 & W_8^3 \\ W_8^6 & W_8^2 & W_8^6 & W_8^2 \\ W_8^7 & W_8^5 & W_8^3 & W_8^1 \end{pmatrix} = W_8^4 \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ W_8^1 & W_8^3 & W_8^5 & W_8^7 \\ W_8^2 & W_8^6 & W_8^2 & W_8^6 \\ W_8^3 & W_8^1 & W_8^7 & W_8^5 \end{pmatrix}}_{= \text{ "rechts oben" } = {}^8\mathrm{Diag}_4 \cdot \Lambda^4}$$

Berücksichtigen wir nun noch, dass

$$W_8^4 = \mathrm{e}^{-2\pi \mathrm{i} \frac{4}{8}} = \mathrm{e}^{-\mathrm{i}\pi} = -1,$$

so ist die Untermatrix "rechts unten" durch $-{}^8\mathrm{Diag}_4 \cdot \Lambda^4$ gegeben.

Demzufolge lässt sich die 8-Punkt-Transformation ${}^8H(h)$ in folgender kompakter Schreibweise darstellen (wobei die Indizes von h die Ordnung der Elemente angeben)

$$\begin{aligned} {}^8H(h_{[0:7]}) &= \begin{pmatrix} \Lambda^4 & {}^8\mathrm{Diag}_4 \cdot \Lambda^4 \\ \Lambda^4 & -{}^8\mathrm{Diag}_4 \cdot \Lambda^4 \end{pmatrix} \begin{pmatrix} h_{[0:7:2]} \\ h_{[1:7:2]} \end{pmatrix} \\ &= \begin{pmatrix} 1 & {}^8\mathrm{Diag}_4 \\ 1 & -{}^8\mathrm{Diag}_4 \end{pmatrix} \begin{pmatrix} \Lambda^4 \cdot h_{[0:7:2]} \\ \Lambda^4 \cdot h_{[1:7:2]} \end{pmatrix} \end{aligned}$$

Der letzte Vektor,

$$\begin{pmatrix} \Lambda^4 \cdot h_{[0:7:2]} \\ \Lambda^4 \cdot h_{[1:7:2]} \end{pmatrix} = \begin{pmatrix} {}^4H(h_{[0:7:2]}) \\ {}^4H(h_{[1:7:2]}) \end{pmatrix}$$

entspricht aber gerade einer FOURIER-Transformion für 4 Punkte (einmal für die "geraden" und einmal für die "ungeraden" des Ausgangsproblemes). Wir haben also eine 8-Punkt-Transformation auf zwei 4-Punkt-Transformation zurückgeführt. Dies ist aber nichts anderes als das DANIELSON-LANCZOS-Lemma!

Analog führen wir nun diese zwei 4-Punkt-Transformationen

$$\begin{aligned} {}^4H(h_{[0:7:2]}) &= \begin{pmatrix} 1 & {}^4\mathrm{Diag}_2 \\ 1 & -{}^4\mathrm{Diag}_2 \end{pmatrix} \begin{pmatrix} {}^2H(h_{[0:7:4]}) \\ {}^2H(h_{[2:7:4]}) \end{pmatrix} \\ {}^4H(h_{[1:7:2]}) &= \begin{pmatrix} 1 & {}^4\mathrm{Diag}_2 \\ 1 & -{}^4\mathrm{Diag}_2 \end{pmatrix} \begin{pmatrix} {}^2H(h_{[1:7:4]}) \\ {}^2H(h_{[3:7:4]}) \end{pmatrix} \end{aligned}$$

auf vier 2-Punkt-Transformationen (mit wiederum geänderter Anordnung der h_k) zurück. Schließlich lautet im Falle $N = 8$ der letzte Schritt

$$\begin{aligned} {}^2H(h_{[0:7:4]}) &= \begin{pmatrix} 1 & {}^2\mathrm{Diag}_1 \\ 1 & -{}^2\mathrm{Diag}_1 \end{pmatrix} \begin{pmatrix} {}^1H(h_0) \\ {}^1H(h_4) \end{pmatrix} &&= \begin{pmatrix} 1 & {}^2\mathrm{Diag}_1 \\ 1 & -{}^2\mathrm{Diag}_1 \end{pmatrix} \begin{pmatrix} h(x_0) \\ h(x_4) \end{pmatrix} \\ {}^2H(h_{[2:7:4]}) &= \quad \ldots &&= \begin{pmatrix} 1 & {}^2\mathrm{Diag}_1 \\ 1 & -{}^2\mathrm{Diag}_1 \end{pmatrix} \begin{pmatrix} h(x_2) \\ h(x_6) \end{pmatrix} \\ {}^2H(h_{[1:7:4]}) &= \quad \ldots &&= \begin{pmatrix} 1 & {}^2\mathrm{Diag}_1 \\ 1 & -{}^2\mathrm{Diag}_1 \end{pmatrix} \begin{pmatrix} h(x_1) \\ h(x_5) \end{pmatrix} \\ {}^2H(h_{[3:7:4]}) &= \quad \ldots &&= \begin{pmatrix} 1 & {}^2\mathrm{Diag}_1 \\ 1 & -{}^2\mathrm{Diag}_1 \end{pmatrix} \begin{pmatrix} h(x_3) \\ h(x_7) \end{pmatrix} \end{aligned}$$

wobei ${}^1H(h_k)$ eine 1-Punkt-Transformation, d.h. die Identität ist:

$${}^1H(h_k) = W_1^{0\cdot 0} h_k = h_k = h(x_k).$$

Insgesamt haben wir also *eine* 8-Punkt-Transformation in 3 Schritten auf *acht* 1-Punkt-Transformationen zurückgeführt, wobei das "Ausgangsmuster" $[0, 1, 2, 3, 4, 5, 6, 7]$ in das Endmuster $[0, 4, 2, 6, 1, 5, 2, 7]$ übergeführt wurde. Damit haben wir das Prinzip der FFT (nochmals) aufgezeigt! Klar ersichtlich ist insbesondere die jeweilige Zusammenfassung der Terme mit den Gewichte $(1, +W)$ bzw. $(1, -W)$ (vgl. Seite 6-23).

Zur algorithmischen Umsetzung der FFT ergeben sich somit zwei Möglichkeiten

- Zum einen läßt sich ein Algorithmus konstruieren, der "von oben nach unten" arbeitet, d.h. startend mit ${}^8H(h)$. Solch ein Algorithmus ist dann *rekursiver* Natur und nur relativ schwer zu durchschauen, setzt allerdings keine a priori Umordnung des Signalvektors h voraus.
- Zum anderen kann man "von unten nach oben" arbeiten, beginnend mit der 2-Punkt-Transformation. Dieser Algorithmus (manchmal auch COOLEY-TUKEY FFT oder "decimation in time" genannt) ist der üblicherweise verwendete und wurde in Kap. 6.2.5 vorgestellt. Hier muss natürlich, im Gegensatz zum rekursiven Algorithmus, als erstes der Ausgangsvektors durch Bitumkehr (siehe Seite 6-23) umgeordnet werden.

Kapitel 7

Integrale

Viele Problemstellungen in der Physik (z.B. die Berechnung von Trägheitsmomenten, elektrischen Feldern von Ladungsverteilungen, Zustandssummen) führen auf Integrale, die nicht mehr analytisch berechnet werden können, weil sie nicht durch bekannte Funktionen ausgedrückt werden können oder die Integranden nur an diskreten Punkten (z.B. Meßdaten) bekannt sind. Man ist dann auf numerische Verfahren angewiesen.

Wir betrachten hier nur bestimmte Integrale. Unbestimmte Integrale (Stammfunktionen) können als Anfangswertprobleme von Differentialgleichungen (siehe Kapitel 9) behandelt werden.

Zur numerischen Berechnung von bestimmten Integralen stehen mehrere Verfahren zur Verfügung. Die Auswahl eines geeigneten Verfahrens richtet sich insbesondere nach den Eigenschaften des Integranden und der gewünschten Genauigkeit.

Im folgenden betrachten wir das Problem, für das bestimmte Integral

$$I(f) := \int_a^b f(x)\mathrm{d}x \tag{7.1}$$

einer integrierbaren Funktion $f : [a, b] \to \mathbb{R}$ ($\mathbb{C}$) einen numerischen Näherungswert

$$\tilde{I}(f) \approx I(f) \tag{7.2}$$

zu berechnen. Die Formel zur Berechnung von $\tilde{I}(f)$, die sogenannte *Quadraturformel*, hat üblicherweise die Form

$$\tilde{I}(f) := \sum_{i=0(1)}^{n} w_i f(x_i) \tag{7.3}$$

mit *Knoten* $a \le x_i \le b$ und von f unabhängigen *Gewichten* w_i.

Eine Quadraturformel hat den *Genauigkeitsgrad* p ($p \in \{0, 1, \dots\}$), wenn sie für alle Polynome bis einschließlich zum Grad p, aber nicht $p + 1$, das exakte Ergebnis liefert.

7.1 Quadraturformeln von Newton und Cotes

Die Idee der Newton-Cotes-Formeln ist, den Integranden durch ein Integrationspolynom zu ersetzen. Seien die $n + 1$ Knoten gegeben durch

$$x_i = a + i\,h \qquad (i = 0, \dots, n) \tag{7.4}$$

mit der Schrittweite

$$h := \frac{b - a}{n} \tag{7.5}$$

und $p_n(x)$ das eindeutig bestimmte Interpolationspolynom höchstens n-ter Ordnung zu den Stützstellen x_i und Stützwerten $f(x_i)$. Dann approximiert man $I(f)$ durch

$$\tilde{I}(f) := \int_a^b p_n(x)\mathrm{d}x. \tag{7.6}$$

Mit der LAGRANGE'schen Interpolationsformel

$$p_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{7.7}$$

(siehe Unterabschnitt 2.1.B) erhält man

$$\begin{aligned} \tilde{I}(f) &= \sum_{i=0}^{n} f(x_i) \int_a^b \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}\,\mathrm{d}x \\ &\overset{x=:a+ht}{=} h \cdot \sum_{i=0}^{n} f(x_i) \int_0^n \prod_{j=0, j\neq i}^{n} \frac{t - j}{i - j}\,\mathrm{d}t \\ &= (b-a) \sum_{i=0}^{n} \sigma_i f(x_i) \end{aligned} \tag{7.8}$$

mit

$$\sigma_i := \frac{1}{n} \int_0^n \prod_{j=0, j\neq i}^{n} \frac{t - j}{i - j}\,\mathrm{d}t \qquad (i = 0, \ldots, n). \tag{7.9}$$

Es gilt

$$\begin{aligned} \sigma_{n-i} &= \frac{1}{n} \int_0^n \prod_{j=0, j\neq n-i}^{n} \frac{t - j}{n - i - j}\,\mathrm{d}t \\ &\overset{k:=n-j}{=} \frac{1}{n} \int_0^n \prod_{k=0, k\neq i}^{n} \frac{t - n + k}{k - i}\,\mathrm{d}t \\ &\overset{s:=n-t}{=} \frac{1}{n} \int_0^n \prod_{k=0, k\neq i}^{n} \frac{s - k}{i - k}\,\mathrm{d}s \\ &= \sigma_i \end{aligned} \tag{7.10}$$

Offensichtlich ist der Genauigkeitsgrad p der Quadraturformel mindestens gleich n. Für gerades n ist der Genauigkeitsgrad p sogar mindestens $n+1$. Denn einerseits gilt $I\left(\left(x - \frac{a+b}{2}\right)^{n+1}\right) = 0$, da

$\left(x - \frac{a+b}{2}\right)^{n+1}$ ungerade bezüglich $\frac{a+b}{2}$ ist; andererseits gilt

$$\begin{aligned}
\tilde{I}\left(\left(x - \frac{a+b}{2}\right)^{n+1}\right) &= (b-a)\sum_{i=0}^{n} \sigma_i \left(x_i - \frac{a+b}{2}\right)^{n+1} \\
&= (b-a)\sum_{i=0}^{n/2} \left[\sigma_i \left(x_i - \frac{a+b}{2}\right)^{n+1} + \underbrace{\sigma_{n-i}}_{=\sigma_i} \underbrace{\left(x_{n-i} - \frac{a+b}{2}\right)^{n+1}}_{=-\left(x_i - \frac{a+b}{2}\right)}\right] \\
&\quad - (b-a)\sigma_{n/2} \underbrace{\left(x_{n/2} - \frac{a+b}{2}\right)^{n+1}}_{=0} \\
&= 0.
\end{aligned}$$

Genauer ('mindestens' fällt weg):

$$p = \begin{cases} n\,, & \text{falls } n \text{ ungerade} \\ n+1\,, & \text{falls } n \text{ gerade} \end{cases}. \tag{7.11}$$

7.1.1 Wichtige Spezialfälle

i) $n = 1$: *Trapezregel*

Mit $\sigma_0 = \int_0^1 \frac{t-1}{0-1}\,\mathrm{d}t = \left[t - \frac{t^2}{2}\right]_{t=0}^{1} = \frac{1}{2}$ und $\sigma_1 = \sigma_0$ erhält man

$$\tilde{I}(f) = \frac{b-a}{2}\left(f(a) + f(b)\right) \quad \text{(siehe Abbildung 7.1)}. \tag{7.12}$$

ii) $n = 2$: SIMPSON-*Regel*

$$\begin{aligned}
\sigma_0 &= \frac{1}{2}\int_0^2 \frac{t-1}{0-1}\frac{t-2}{0-2}\,\mathrm{d}t = \frac{1}{6}, \\
\sigma_1 &= \frac{1}{2}\int_0^2 \frac{t-0}{1-0}\frac{t-2}{1-2}\,\mathrm{d}t = \frac{2}{3}, \\
\sigma_2 &= \sigma_0 = \frac{1}{6}
\end{aligned}$$

Man erhält also

$$\tilde{I}(f) = \frac{b-a}{6}\left(f(a) + 4\,f\left(\frac{a+b}{2}\right) + f(b)\right). \tag{7.13}$$

7.1 Beispiel. $f(x) = \sin(x)$, $x \in [0, \pi/2]$;
I(f)=1.

- Trapezregel: $\tilde{I}(f) = \frac{\pi}{4} = 0.7854\ldots$,
- SIMPSON-Regel: $\tilde{I}(f) = \frac{\pi}{12}\left(\sqrt{8}+1\right) = 1.002\ldots$

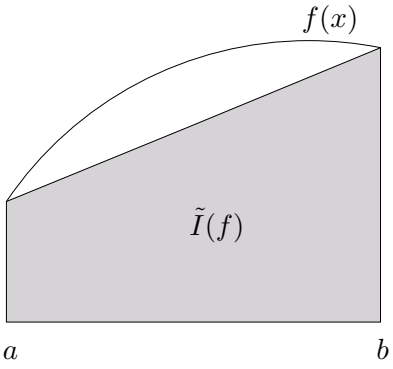


Abbildung 7.1: Trapezregel

7.1.2 Fehlerabschätzung

7.2 Satz. *Für ungerades n und $f \in \mathcal{C}^{(n+1)}([a,b])$ gilt*

$$\left|I(f) - \tilde{I}(f)\right| \leq c_n \max_{x\in[a,b]} \left|f^{(n+1)}(x)\right| h^{n+2} \tag{7.14}$$

mit

$$c_n := \frac{1}{(n+1)!} \left| \int_0^n \prod_{i=0}^n (t-i)\,\mathrm{d}t \right| \qquad (n \text{ ungerade}). \tag{7.15}$$

Ist n gerade und $f \in \mathcal{C}^{(n+2)}([a,b])$, so gilt

$$\left|I(f) - \tilde{I}(f)\right| \leq c_n \max_{x\in[a,b]} \left|f^{(n+2)}(x)\right| h^{n+3} \tag{7.16}$$

mit

$$c_n := \frac{1}{(n+2)!} \left| \int_0^n t \prod_{i=0}^n (t-i)\,\mathrm{d}t \right| \qquad (n \text{ gerade}). \tag{7.17}$$

7.3 Beweis. *(i) n ungerade.*

Sei p_n das eindeutig bestimmte Interpolationspolynom höchstens n-ter Ordnung mit $p_n(x_i) = f(x_i)$ $(i = 0,\ldots,n)$. Zu jedem $x \in [a;b]$ existiert ein $\xi(x) \in [a;b]$ mit

$$f(x) - p_n(x) = \left(\prod_{i=0}^n (x - x_i)\right) \frac{f^{(n+1)}(\xi(x))}{(n+1)!}$$

(siehe Beweis von Satz 2.8). Damit ist

$$I(f) - \tilde{I}(f) = \int_a^b (f(x) - p_n(x))\,\mathrm{d}x = \frac{1}{(n+1)!} \int_a^b \left(\prod_{i=0}^n (x - x_i)\right) f^{(n+1)}(\xi(x))\,\mathrm{d}x\,.$$

Nach dem Mittelwertsatz der Integralrechnung existiert ein $\eta \in [a;b]$ mit

$$\int_a^b \left(\prod_{i=0}^n (x - x_i)\right) f^{(n+1)}(\xi(x))\mathrm{d}x = f^{(n+1)}(\eta) \int_a^b \left(\prod_{i=0}^n (x - x_i)\right) \mathrm{d}x.$$

Mit

$$\int_a^b \left(\prod_{i=0}^n (x - x_i)\right) \mathrm{d}t \overset{x=:a+ht}{=} h^{n+2} \int_0^n \prod_{i=0}^n (t-i)\,\mathrm{d}t$$

ergibt sich damit

$$\left|I(f)-\tilde{I}(f)\right| = c_n \left|f^{(n+1)}(\eta)\right| h^{n+2} \le c_n \max_{x\in[a;b]} \left|f^{(n+1)}(x)\right| h^{n+2}.$$

(ii) n gerade.

Sei $\tilde{x} \in [a;b]$ mit $\tilde{x} \neq x_i \ \forall\ i \in \{0,1,\ldots,n\}$ ein weiterer Knoten und p_{n+1} das eindeutig bestimmte Interpolationspolynom höchstens $(n+1)$-ter Ordnung mit $p_{n+1}(x_i) = f(x_i)$ $(i = 0,\ldots,n)$ und $p_{n+1}(\tilde{x}) = f(\tilde{x})$. Zu jedem $x \in [a;b]$ existiert ein $\xi(x) \in [a;b]$ mit

$$f(x) - p_{n+1}(x) = \left(\prod_{i=0}^{n}(x-x_i)\right)(x-\tilde{x})\frac{f^{(n+2)}(\xi(x))}{(n+2)!}.$$

Wegen $\tilde{I}(f) = \tilde{I}(p_{n+1}) \stackrel{!}{=} I(p_{n+1})$ ist dann

$$\begin{aligned} I(f)-\tilde{I}(f) &= \int_a^b (f(x)-p_{n+1}(x))\,\mathrm{d}x \\ &= \frac{1}{(n+2)!}\int_a^b \left(\prod_{i=0}^{n}(x-x_i)\right)(x-\tilde{x})f^{(n+2)}(\xi(x))\,\mathrm{d}x \\ &= \frac{1}{(n+2)!}f^{(n+2)}(\eta)\int_a^b \left(\prod_{i=0}^{n}(x-x_i)\right)(x-\tilde{x})\,\mathrm{d}x \end{aligned}$$

mit geeignetem $\eta \in [a;b]$. Mit

$$\begin{aligned} \int_a^b \left(\prod_{i=0}^{n}(x-x_i)\right)(x-\tilde{x})\,\mathrm{d}x &= \int_a^b x\left(\prod_{i=0}^{n}(x-x_i)\right)\mathrm{d}x \\ &\quad -\tilde{x}\int_a^b \underbrace{\left(\prod_{i=0}^{n}(x-x_i)\right)}_{\text{ungerade bzgl. } \frac{a+b}{2}}\mathrm{d}x \\ &= h^{n+3}\int_0^n t\left(\prod_{i=0}^{n}(t-i)\right)\mathrm{d}t - \tilde{x}\cdot 0 \end{aligned}$$

ergibt sich damit

$$\left|I(f)-\tilde{I}(f)\right| = c_n \left|f^{(n+2)}(\eta)\right| h^{n+3} \le c_n \max_{x\in[a;b]} \left|f^{(n+2)}(x)\right| h^{n+3}.$$

Die Koeffizienten der Fehlerabschätzung lassen sich analytisch berechnen:

$$c_1 = \frac{1}{12}, \qquad c_2 = \frac{1}{90}, \qquad \ldots \tag{7.18}$$

Speziell gilt für

$$n=1: \qquad \left|I(f)-\tilde{I}(f)\right| \le \frac{1}{12}\max_{x\in[a;b]}\left|f^{(2)}(x)\right| h^3\,, \tag{7.19}$$

$$n=2: \qquad \left|I(f)-\tilde{I}(f)\right| \le \frac{1}{90}\max_{x\in[a;b]}\left|f^{(4)}(x)\right| h^5\,. \tag{7.20}$$

Das Maximum der $(n+1)$-ten bzw. $(n+2)$-ten Ableitung ist oftmals schwer zu bestimmen. Dann ist die Formel (7.14) bzw. (7.16) zur Fehlerabschätzung von geringem praktischen Nutzen. Sie liefert aber eine Information, von welcher Potenz der Schrittweite der Fehler abhängt.

Wie bei der Polynominterpolation hat man auch bei den NEWTON-COTES-Formeln nicht immer eine Konvergenz für $n \to \infty$, z.B. $\tilde{I}(f) \xrightarrow{n\to\infty} \infty$ für $f(x) = \dfrac{1}{1+(5x)^2}$ $(x \in [-1;1])$. Zudem treten bei $n = 8$ und $n \geq 10$ negative Gewichte auf. Dies kann zu einer numerischen Instabilität führen. Bei der Beschränkung auf kleine n sind aber im Allgemeinen keine genauen Ergebnisse zu erwarten. Daher verwendet man sogenannte *zusammengesetzte Quadraturformeln*. Hierbei wird eine Quadraturformel nicht für das gesamte Intervall, sondern nur für kleinere Teilintervalle benützt.

7.2 Zusammengesetzte Quadraturformeln von NEWTON und COTES

Das Interval $[a,b]$ wird in N Teilintervalle unterteilt. In jedem dieser Teilintervalle wird eine Quadraturformel (meist $n=1$ oder $n=2$) angewendet. Eine höhere Genauigkeit kann durch eine feinere Intervallzerlegung erreicht werden.

Sei N vorgegeben und seien

$$x_i := a + i\,h \qquad (i = 0, 1, \ldots, N) \tag{7.21}$$

mit

$$h := \frac{b-a}{N}. \tag{7.22}$$

Zusammengesetzte Trapezregel (n=1).

$$[a,b] = \bigcup_{i=1}^{N} [x_{i-1}, x_i] \tag{7.23}$$

$$I(f) = \int_a^b f(x)\mathrm{d}x = \sum_{i=1}^{N} \underbrace{\int_{x_{i-1}}^{x_i} f(x)\mathrm{d}x}_{=:I_i(f)} = \sum_{i=1}^{N} I_i(f) \tag{7.24}$$

$$I_i(f) \approx \tilde{I}_i(f) := \frac{h}{2}\left(f(x_{i-1}) + f(x_i)\right) \tag{7.25}$$

$$I(f) \approx \tilde{I}_{\mathrm{T}}(f) := \sum_{i=1}^{N} \tilde{I}_i(f) = \frac{h}{2}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{N-1}) + f(x_N)\right) \tag{7.26}$$

(siehe Abbildung 7.2)

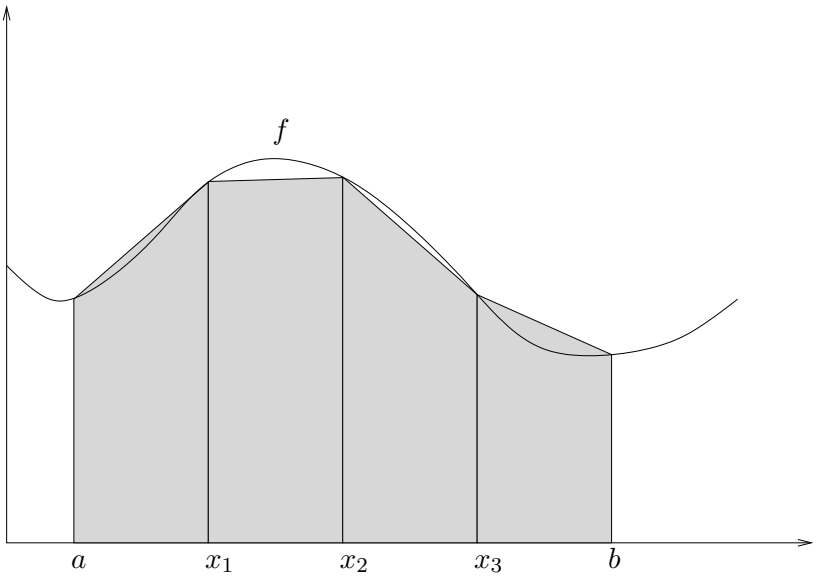


Abbildung 7.2: Zusammengesetzte Trapezregel

Zusammengesetzte SIMPSON-Regel ($n = 2$; wird häufig verwendet).

$$[a,b] = \bigcup_{i=1}^{N/2} [x_{2i-2}, x_{2i}] \qquad \text{für gerades } N \tag{7.27}$$

$$I(f) = \int_a^b f(x)\mathrm{d}x = \sum_{i=1}^{N/2} \underbrace{\int_{x_{2i-2}}^{x_{2i}} f(x)\mathrm{d}x}_{=:I_i(f)} = \sum_{i=1}^{N/2} I_i(f) \tag{7.28}$$

$$I_i(f) \approx \tilde{I}_i(f) := \frac{2h}{6}\left(f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})\right) \tag{7.29}$$

$$I(f) \approx \tilde{I}_{\mathrm{S}}(f) := \sum_{i=1}^{N/2} \tilde{I}_i(f) = \frac{h}{3}\big(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + 4f(x_5) + \cdots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N)\big) \tag{7.30}$$

Auch für diese zusammengesetzten Regeln kann man eine Fehlerabschätzung herleiten, indem man die Fehlerabschätzung (7.19) bzw. (7.20) auf jedes Teilintervall anwendet und anschließend aufsummiert.

Fehlerabschätzungen.

a) Zusammengesetzte Trapezregel:
Für $f \in \mathcal{C}^{(2)}([a;b])$ gilt

$$\begin{aligned} \left|I(f) - \tilde{I}_{\mathrm{T}}(f)\right| &\leq \sum_{i=1}^{N} \left|I_i(f) - \tilde{I}_i(f)\right| \\ &\overset{(7.19)}{\leq} \frac{1}{12}\left(\sum_{i=1}^{N} \max_{x\in[x_{i-1},x_i]} \left|f^{(2)}(x)\right|\right) h^3 \\ &\leq \frac{b-a}{12} \max_{x\in[a,b]} \left|f^{(2)}(x)\right| h^2 \end{aligned} \tag{7.31}$$

b) Zusammengesetzte SIMPSON-Regel:
Für $f \in \mathcal{C}^{(4)}([a;b])$ gilt

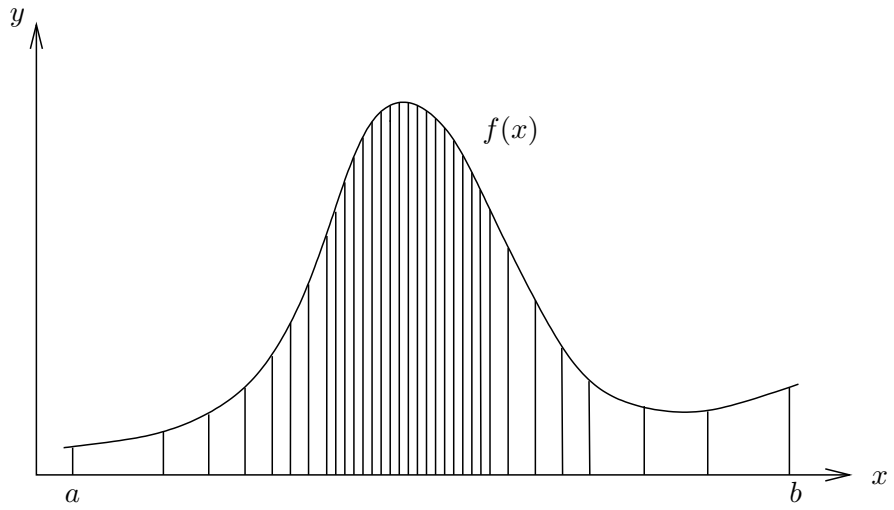
$$\begin{aligned}\left|I(f) - \tilde{I}_S(f)\right| &\leq \sum_{i=1}^{N/2} \left|I_i(f) - \tilde{I}_i(f)\right| \\ &\overset{(7.20)}{\leq} \frac{1}{90}\left(\sum_{i=1}^{N/2} \max_{x\in[x_{2i-2},x_{2i}]} \left|f^{(4)}(x)\right|\right) h^5 \\ &\leq \frac{b-a}{180} \max_{x\in[a,b]} \left|f^{(4)}(x)\right| h^4 \end{aligned} \tag{7.32}$$

7.4 Beispiel. $f(x) = \sin(2x)$, $x \in [0; \frac{\pi}{4}]$.
Das exakte Resultat ist $I(f) = \frac{1}{2}$. Die Näherungswerte für verschiedene N zeigt die folgende Tabelle:

N	Trapezregel	SIMPSONregel
2	0.47402972	0.50006729
3	0.48852431	0.50001316
4	0.49355790	0.50000415
32	0.49989960	
33	0.49990559	
41	0.49993384	

7.3 Schrittweitensteuerung

Wenn der Integrand in verschiedenen Bereichen des Integrationsintervalls stark unterschiedlich variiert, ist es zweckmäßig, das Intervall nicht mehr in gleichlange Teilintervalle zu zerlegen. Denn in Bereichen starker Variation des Integranden sind wesentlich mehr Knoten notwendig, um eine bestimmte Genauigkeit zu erreichen, als in Bereichen schwacher Variation des Integranden. Es wäre nicht effizient, die notwendigerweise besonders kleine Schrittweite in Bereichen starker Variation auch in Bereichen schwacher Variation des Integranden zu verwenden. Die Unterteilung sollte dort feiner sein, wo der Integrand stark variiert und dort gröber sein, wo er schwach variiert.



7.5 Beispiel. Für $f(x) = \dfrac{x}{x^2-1}$ $(x \in [1.001; 10])$ bewegt sich die vierte Ableitung, die den Fehler bei der zusammengesetzten SIMPSON-Regel kontrolliert, zwischen $1.2\cdot10^8$ am linken Rand und $2.7\cdot10^{-4}$ am rechten Rand. Erwartungsgemäß kann man am rechten Ende des Intervalles mit wesentlich weniger Stützstellen, d.h. wesentlich geringerem Rechenaufwand, eine akzeptable Näherung des Integrals $I(f)$ bestimmen als in der Nähe des linken Randes.

Seien

$$a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b \tag{7.33}$$

und

$$h_i := x_i - x_{i-1} \qquad (i = 1, \ldots, N) \tag{7.34}$$

(Zerlegung i.A. nicht äquidistant). Das Integral kann dann als Summe

$$I(f) = \sum_{i=1}^{N} I_i(f) \tag{7.35}$$

der Teilintegrale

$$I_i(f) = \int_{x_{i-1}}^{x_i} f(x)\mathrm{d}x \qquad (i = 1, \ldots, N) \tag{7.36}$$

geschrieben werden. Diese Teilintegrale können mit der SIMPSON-Regel näherungsweise berechnet werden:

$$I_i(f) \approx \tilde{I}_i(f) := \frac{h_i}{6}\left(f(x_{i-1}) + 4f\left(\frac{x_{i-1}+x_i}{2}\right) + f(x_i)\right). \tag{7.37}$$

Durch Aufsummieren ergibt sich dann

$$I(f) \approx \tilde{I}_{\mathrm{S}}(f) := \sum_{i=1}^{N} \tilde{I}_i(f). \tag{7.38}$$

Der Fehler soll innerhalb einer vorgegebenen Toleranz ε liegen:

$$\left|I(f) - \tilde{I}(f)\right| \overset{!}{\leq} \varepsilon. \tag{7.39}$$

Er soll auch gleichmäßig auf die Teilintervalle verteilt werden, d.h. h_i soll so gewählt werden, daß

$$\left|I_i(f) - \tilde{I}_i(f)\right| \leq \frac{h_i}{b-a}\varepsilon \tag{7.40}$$

erfüllt ist. Aber diese Bedingung können wir in der Praxis nicht überprüfen, weil $I_i(f)$ unbekannt ist. Hier hilft uns folgende Beobachtung weiter: Für ein vorgegebenes i gilt

$$I_i(f) - \tilde{I}_i(f) \approx c_i h_i^5 \tag{7.41}$$

bei der Schrittweite h_i und

$$I_i(f) - \tilde{I}'_i(f) \approx 2c_i {h'_i}^5 \tag{7.42}$$

bei der halben Schrittweite

$$h'_i := \frac{h_i}{2}, \tag{7.43}$$

also

$$\tilde{I}'_i(f) - \tilde{I}_i(f) \approx c_i\left(1 - 2^{-4}\right) h_i^5. \tag{7.44}$$

Damit ist

$$I_i(f) - \tilde{I}_i(f) \approx \frac{\tilde{I}'_i(f) - \tilde{I}_i(f)}{1 - 2^{-4}}. \tag{7.45}$$

Man erhält die überprüfbare Bedingung

$$\left|\tilde{I}'_i(f) - \tilde{I}_i(f)\right| \leq \left(1 - 2^{-4}\right)\frac{h_i}{b-a}\varepsilon. \tag{7.46}$$

Weder die Anzahl der Teilintervalle noch die Unterteilungspunkte sind am Anfang des Quadraturverfahrens bekannt. Die Schrittweiten werden sukzessive im Verlauf des Verfahrens bestimmt.

Schrittweitenwahl. Angenommen $h_1, h_2, \ldots, h_{i-1}$ (und damit $x_1, \ldots, x_{i-1}$) seien bereits bestimmt. Außerdem sei eine Vorschlagsschrittweite h_i gegeben.

i) Berechne $\tilde{I}_i(f)$.

ii) Berechne $\tilde{I}'_i(f)$.

iii) Überprüfe, ob

$$\left|\tilde{I}'_i(f) - \tilde{I}_i(f)\right| \leq (1 - 2^{-4})\frac{h_i}{b-a}\varepsilon$$

erfüllt ist.

Falls ja: Akzeptiere $\tilde{I}_i(f)$ als Näherung für $I_i(f)$.

Falls nein: Halbiere h_i, setze $\tilde{I}_i(f)$ gleich $\tilde{I}'_i(f)$ und gehe nach ii.

iv) Überprüfe, ob

$$\left|\tilde{I}'_i(f) - \tilde{I}_i(f)\right| \leq \frac{1 - 2^{-4}}{(2.5)^4}\frac{h_i}{b-a}\varepsilon$$

mit dem Sicherheitsfaktor von z.B. 2.5 gilt.

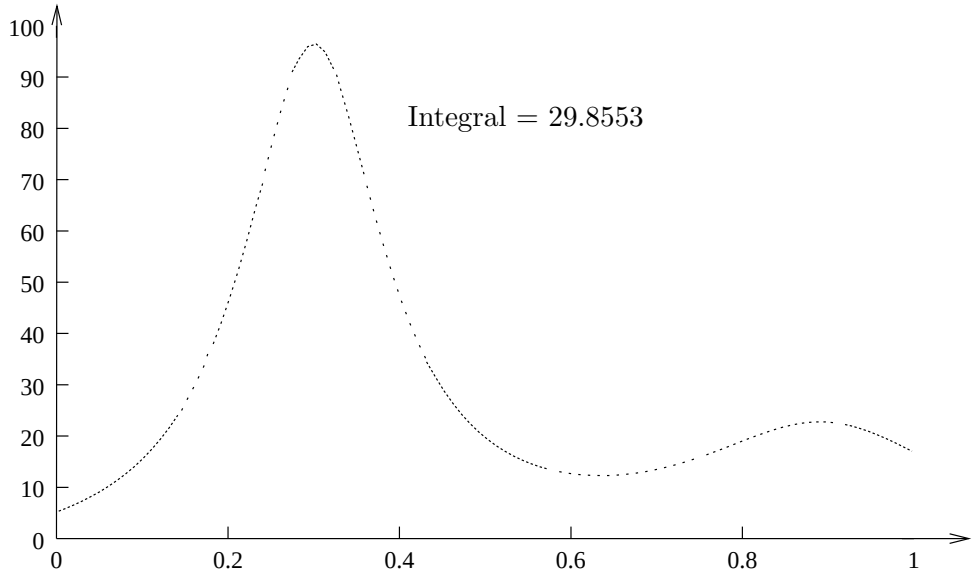
Falls ja: neue Vorschlagsschrittweite $h_{i+1} = 2h_i$.

Falls nein: neue Vorschlagsschrittweite $h_{i+1} = h_i$.

Für den praktischen Einsatz des Verfahrens sollte eine Unter- und Oberschranke für h_i eingebaut werden. Denn zu kleine Schrittweiten führen zu starken Rundungsfehlern, bei zu großen Schrittweiten werden Bereiche, in denen der Integrand stark variiert, übersprungen.

7.6 Beispiel.

$$f(x) := \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6, \qquad a = 0, \quad b = 1$$



7.4 Quadraturverfahren von ROMBERG

(hohe Genauigkeit erreichbar)

Bei einer Verkleinerung der Schrittweite nimmt der Fehler bei der Berechnung eines Integrals nach der zusammengesetzten Trapezregel ab. Ist der Integrand $f \in \mathcal{C}^{(2)}([a;b])$, so konvergiert die Trapezsumme $\tilde{I}_{T,h}(f)$ gegen das Integral $I(f)$ im Grenzfall Schrittweite $h \to 0$ (siehe Fehlerabschätzung (7.31)). Die Trapezsumme kann nicht direkt bei verschwindender Schrittweite ausgewertet werden.

Formel von Euler und MacLaurin.

$$\tilde{I}_{\mathrm{T},h}(f) - I(f) = \alpha_2 h^2 + \alpha_4 h^4 + \cdots + \alpha_{2m} h^{2m} + \underbrace{\mathcal{O}(h^{2m+2})}_{\text{i. A. kein Polynom}} \tag{7.47}$$

für $f \in \mathcal{C}^{(2m+2)}([a;b])$ mit

$$\alpha_{2\mu} := \frac{B_{2\mu}}{(2\mu)!}\left(f^{(2\mu-1)}(b) - f^{(2\mu-1)}(a)\right) \qquad (\mu = 1, \ldots, m) \tag{7.48}$$

liefert die Asymptotik des Fehlers für Schrittweite $h \to 0$. Hierbei sind

$$B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_6 = \frac{1}{42}, \quad B_8 = -\frac{1}{30}, \quad B_{10} = \frac{5}{66}, \quad B_{12} = -\frac{691}{2730}, \quad \ldots \tag{7.49}$$

die sogenannten Bernoulli-*Zahlen* (y-Achsenabschnitte der Bernoulli-Polynome).
Das heißt: Bei Vernachlässigung des $\mathcal{O}(h^{2m+2})$-Termes verhält sich der Fehler wie ein Polynom.

Bemerkung. Bei der Quadratur einer periodischen Funktion $f \in \mathcal{C}^{(2m+2)}(]-\infty;\infty[)$ mit $m > 1$ über eine volle Periodenlänge ist die Trapezregel besser als man erwarten würde. Denn nach der Euler-MacLaurin-Formel ist $\tilde{I}_{\mathrm{T},h}(f) - \tilde{I}(f) = \mathcal{O}(h^{2m})$ und nicht nur $\mathcal{O}(h^2)$.

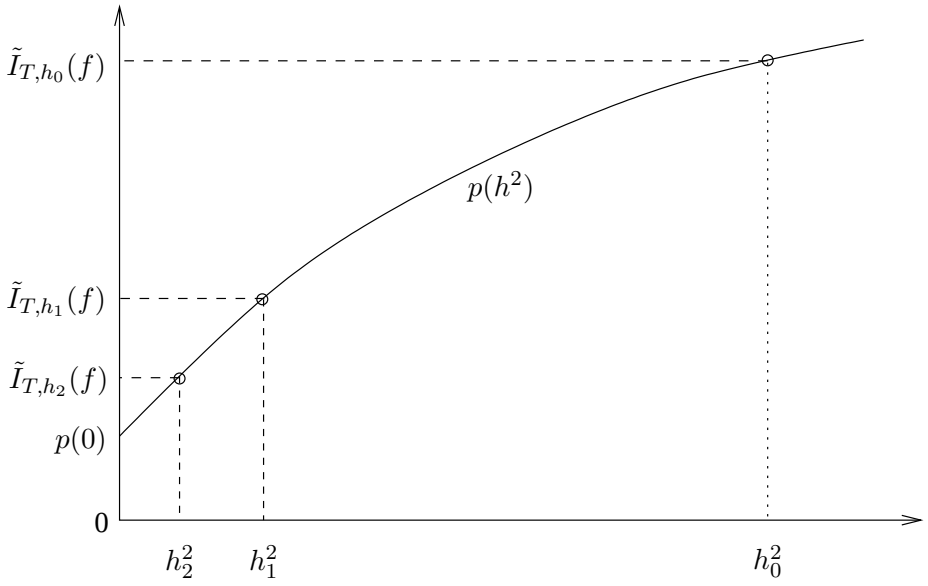
Romberg**-Verfahren.** Auf der Entwicklung (7.47) der Trapezsumme nach Potenzen der Schrittweite beruht das sogenannte Romberg-*Verfahren*: Zu den Schrittweiten

$$h_0, \quad h_1 := \frac{h_0}{2}, \quad h_2 := \frac{h_1}{2}, \quad \ldots, \quad h_r := \frac{h_{r-1}}{2} \tag{7.50}$$

wird jeweils die Trapezsumme $\tilde{I}_{\mathrm{T},h_j}(f)$ berechnet. Anschließend wird der Wert des Interpolationspolynoms $p(h^2)$ höchstens r-ter Ordnung in h^2 durch die Punkte

$$\left(h_0^2, \tilde{I}_{\mathrm{T},h_0}(f)\right), \quad \left(h_1^2, \tilde{I}_{\mathrm{T},h_1}(f)\right), \quad \ldots, \quad \left(h_r^2, \tilde{I}_{\mathrm{T},h_r}(f)\right)$$

an der Stelle 0 bestimmt.



Trapezsummenberechnung. Bei der Berechnung von $\tilde{I}_{\mathrm{T},h_{j+1}}(f)$ sollten die bereits bei $\tilde{I}_{\mathrm{T},h_j}(f)$ berechneten Funktionswerte wiederverwendet werden. Dies kann ohne aufwendige Verwaltung der Funktionswerte geschehen.

- Trapezregel bei Schrittweite $h := \dfrac{b-a}{N}$:

$$\tilde{I}_{\mathrm{T},h}(f) = \frac{h}{2}\left(f(x_0) + 2\sum_{i=1}^{N-1} f(x_i) + f(x_N)\right) \tag{7.51}$$

  mit Knoten

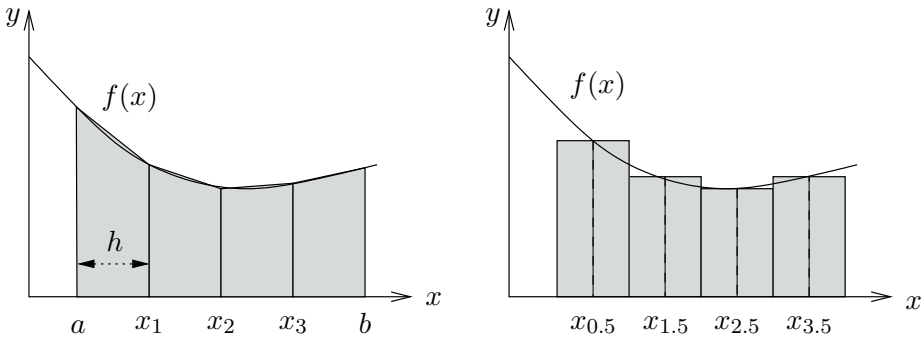
$$x_i := a + i\,h \qquad (i = 0, \ldots, N). \tag{7.52}$$

- Mittelpunktsregel bei Schrittweite h:

$$\tilde{I}_{\mathrm{M},h}(f) = h\sum_{i=0}^{N-1} f(x_{i+1/2}) \tag{7.53}$$

  mit Knoten

$$x_{i+1/2} := a + \left(i + \frac{1}{2}\right) h \qquad (i = 0, \ldots, N-1). \tag{7.54}$$



- Trapezregel bei halber Schrittweite $\frac{h}{2}$:

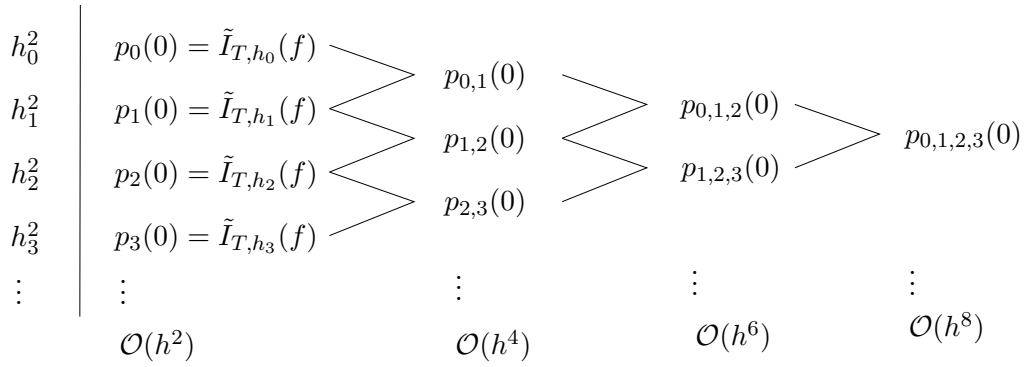
$$\begin{aligned}\tilde{I}_{\mathrm{T},h/2}(f) &= \frac{h}{4}\left(f(x_0) + 2\sum_{i=1}^{N-1} f(x_i) + 2\sum_{i=1}^{N-1} f(x_{i+1/2}) + f(x_N)\right)\\ &= \frac{1}{2}\left(\tilde{I}_{\mathrm{T},h} + \tilde{I}_{\mathrm{M},h}\right),\end{aligned} \tag{7.55}$$

  d.h. $\tilde{I}_{\mathrm{T},h/2}$ kann aus $\tilde{I}_{\mathrm{T},h}$ und $\tilde{I}_{\mathrm{M},h}$ berechnet werden.

Extrapolation. Zur Bestimmung von $p(0)$ eignet sich der NEVILLE-*Algorithmus* (siehe Unterabschnitt 2.1.3) mit

$$p_{j,\ldots,k}(0) = p_{j,\ldots,k-1}(0) + \frac{p_{j+1,\ldots,k}(0) - p_{j,\ldots,k-1}(0)}{1 - \left(\dfrac{h_k}{h_j}\right)^2} \qquad \text{für } k > j. \tag{7.56}$$

Durch diese Rekursion werden sukzessive Quadraturformeln höherer Ordnung erzeugt.



7.7 Beispiel. Für $h_0 := b - a$ ist

$$p_0(0) = \frac{b-a}{2}(f(a) + f(b)) \qquad \text{(Trapezregel)}$$

und

$$p_1(0) = \frac{b-a}{4}\left(f(a) + 2f\left(\frac{a+b}{2}\right) + f(b)\right),$$

also

$$p_{0,1}(0) = p_0(0) + \frac{p_1(0) - p_0(0)}{1 - (1/2)^2} = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right).$$

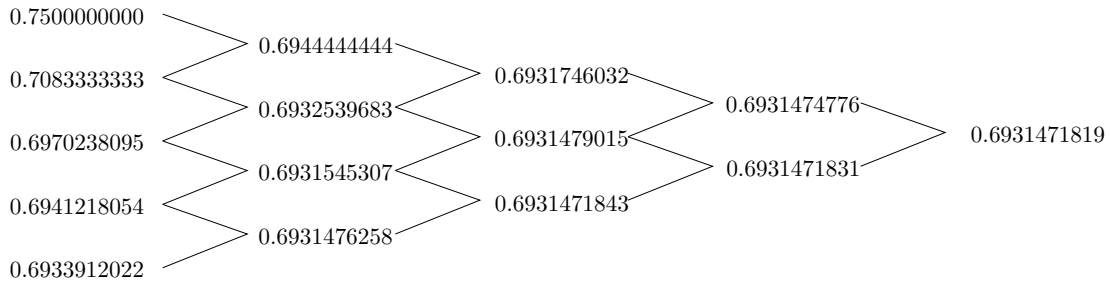
Letzteres ist nichts anderes als die SIMPSON-Regel.

Hinweis. Die Schrittweiten sollten nicht zu klein gewählt werden, denn sonst treten viele Rundungsfehler auf und zusätzlich wird der Rechenaufwand sehr hoch. Im Gegensatz zum Verfahrensfehler nehmen die Rundungsfehler mit abnehmender Schrittweite zu. Bei nicht zu kleinen Schrittweiten dominiert der Verfahrensfehler, bei sehr kleinen Schrittweiten überwiegt der Einfluß der Rundungsfehler. Daher nimmt der Gesamtfehler zunächst ab, dann wieder zu bei abnehmender Schrittweite.

7.8 Beispiel. Das Integral

$$\int_1^2 \frac{1}{x}\mathrm{d}x.$$

Der exakte Wert ist $\ln 2 \approx 0.6931471806\ldots$.
ROMBERG-Tafel ($h_0 = 1$):



7.5 Spezielle Quadraturen

Wir haben bisher nur Integrale der Form $\int_a^b f(x)\mathrm{d}x$ mit einem hinreichend glatten Integranden $f : [a; b] \to \mathbb{R}$ und endlichen Integralgrenzen $-\infty < a < b < \infty$ betrachtet. Jetzt befassen wir uns mit der numerischen Behandlung von Integralen mit einem nicht ausreichend glatten Integranden oder unendlichen Integrationsbereich.

7.5.1 Aufteilung des Integrationsbereiches

Sei $f : [a, b] \to \mathbb{R}$ integrierbar und $f\big|_{[t_{k-1},t_k]}$ hinreichend glatt für alle $k = 1, \ldots, K$ mit $a = t_0 < t_1 < \cdots < t_{K-1} < t_K = b$ (siehe Abbildung 7.3). Dann ist

$$\int_a^b f(x)\mathrm{d}x = \sum_{k=1}^{K} \int_{t_{k-1}}^{t_k} f(x)\mathrm{d}x. \tag{7.57}$$

Jedes Teilintervall kann mit einem der bisherigen Quadraturverfahren näherungsweise berechnet werden.

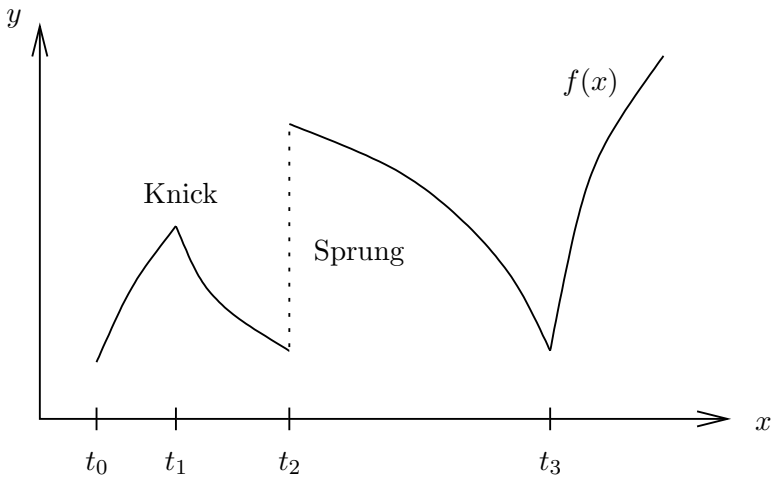


Abbildung 7.3: Aufteilung des Integrationsbereiches

7.5.2 Reihenentwicklung des Integranden

Es kann sein, daß z.B.

$$\lim_{x \to a^+(\text{ oder } b^-)} f'(x) \qquad \text{oder} \qquad \lim_{x \to a^+(\text{ oder } b^-)} f''(x)$$

nicht existiert. Für $0 < \varepsilon < b - a$ gilt

$$\int_a^b f(x)\mathrm{d}x = \int_a^{a+\varepsilon} f(x)\mathrm{d}x + \int_{a+\varepsilon}^b f(x)\mathrm{d}x\,. \tag{7.58}$$

Das zweite Integral kann mit einem der bisherigen Quadraturverfahren numerisch berechnet werden. Das erste Integral kann in Potenzen von ε entwickelt werden. Für kleine ε kann diese Entwicklung bereits nach wenigen Termen abgebrochen werden. ε darf aber nicht zu klein gewählt werden, da sonst der Fehler bei der Berechnung des zweiten Integrals zu groß wird.

7.9 Beispiel. $f(x) = \sqrt{x}\sin(x)$, $x \in [0; 1]$. Es gilt

$$f'(x) = \frac{\sin x + 2x\cos x}{2\sqrt{x}}$$

und

$$f''(x) = \frac{4x\cos x - (4x^2 + 1)\sin(x)}{4x^{3/2}},$$

also $\lim_{x \to 0^+} f''(x) = \infty$. Die Reihenentwicklung lautet

$$\int_0^\varepsilon f(x)\mathrm{d}x = \int_0^\varepsilon \sqrt{x}\left(x - \frac{x^3}{6} + \cdots\right) = \frac{2}{5}\varepsilon^{5/2} - \frac{1}{27}\varepsilon^{9/2} + \cdots$$

.

7.5.3 Subtraktion einer Funktion vom Integranden

In der Situation vom letzten Unterabschnitt kann man auch von $f(x)$ eine Funktion $g(x)$ subtrahieren, die das selbe Verhalten für $x \to a^+$ (oder b^-) hat und deren Stammfunktion bekannt ist. Es gilt dann

$$\int_a^b f(x)\mathrm{d}x = \int_a^b (f(x) - g(x))\mathrm{d}x + \int_a^b g(x)\mathrm{d}x. \tag{7.59}$$

Das erste Integral auf der rechten Seite kann dann mit einem der bisherigen Quadraturverfahren berechnet werden, das zweite Integral durch bekannte Funktionen ausgedrückt werden.

7.10 Beispiel (Fortsetzung von Beispiel 7.9). Mit $g(x) := x^{3/2}$ für $x \in [0;1]$ ergibt sich

$$\int_0^1 (f(x) - g(x))\mathrm{d}x = \int_0^1 \sqrt{x}(\sin x - x)\mathrm{d}x\,.$$

Der Integrand der rechten Seite ist nun drei mal stetig differenzierbar. Das Integral $\int_0^1 x^{3/2}\mathrm{d}x$ hat den Wert $\frac{2}{5}$.

7.5.4 Variablentransformation

Integrierbare Singularität. $\int_0^1 x^{p/q} f(x)\mathrm{d}x$ mit $q = 2, 3, \ldots$; $p > -q$ ganzzahlig und $f(x)$ analytisch in $[0;1]$.

Die Substitution $t := x^{1/q}$ liefert

$$\int_0^1 x^{p/q} f(x)\mathrm{d}x = q \int_0^1 t^{p+q-1} f(t^q)\mathrm{d}t,$$

wobei auf der rechten Seite keine Randsingularität auftritt, da $p + q - 1 > 0$ gilt.

Unendlicher Integrationsbereich.

(i) $\int_1^\infty f(x)\mathrm{d}x$ kann durch die Substitution $t := \dfrac{1}{x}$ umgeformt werden:

$$\int_1^\infty f(x)\mathrm{d}x = \int_0^1 \underbrace{t^{-2} f(1/t)}_{=:g(t)}\,\mathrm{d}t.$$

Hat $g(t)$ oder z.B. $g'(t)$ oder $g''(t)$ eine Singularität bei $t = 0$, so kann man wie oben vorgehen.

(ii) $\int_0^\infty f(x)\mathrm{d}x$ mit langsam abklingender Funktion $f(x)$ für $x \to \infty$: Man benutzt die Substitution $x := \sinh t$ und erhält

$$\int_0^\infty f(x)\mathrm{d}x = \int_0^\infty \underbrace{f(\sinh t)\cosh t}_{(*)}\,\mathrm{d}t.$$

Bei einem stark nach unendlich abfallenden Integranden $(*)$ kann das Integral bei endlicher Grenze abgeschnitten werden (Kriterium: $|f(x)| < \delta$ mit vorgegebener Toleranz δ).

7.11 Beispiel. $f(x) := \dfrac{1}{1+x^2}$ für $x \in [0;\infty[$.
(ii) liefert

$$\int_0^\infty \frac{1}{1+x^2}\,\mathrm{d}x \overset{x=:\sinh t}{=} \int_0^\infty \frac{1}{1+\sinh^2 t}\cosh t\,\mathrm{d}t = \int_0^\infty \frac{1}{\cosh t}\,\mathrm{d}t.$$

Dabei fällt der Integrand des ganz rechts stehenden Integrals exponentiell ab.

7.6 Quadraturformeln von Gauss
(geeignet bei unendlichem Integrationsbereich)

Bei der Newton-Cotes-Formel sind die $n+1$ Knoten äquidistant vorgegeben. Diese Formel integriert alle Polynome höchstens n-ten bzw. $(n+1)$-ten Grades exakt. Man kann sich nun die Frage stellen, ob es nicht möglich ist, die Knoten so anzupassen, daß man einen höheren Genauigkeitsgrad erreicht. Dieser Ansatz führt zu einer sogenannten Gauss-*Formel*.

Wir betrachten das allgemeinere Integral

$$I(f) := \int_a^b w(x)f(x)\mathrm{d}x, \tag{7.60}$$

wobei $w(x)$ eine positive Gewichtsfunktion mit

$$\left|\int_a^b w(x)p(x)\mathrm{d}x\right| < \infty$$

für alle Polynome $p(x)$ ist. $a = -\infty$ und $b = \infty$ sind hier zugelassen.

Für eine Näherung $\tilde{I}(f)$ von $I(f)$ machen wir den Ansatz

$$\tilde{I}(f) := \sum_{i=1}^{n} w_i f(x_i) \qquad \text{(Formel von Gauss)} \tag{7.61}$$

mit den noch zu bestimmenden n Knoten x_i und Gewichten w_i.

7.6.1 Konstruktion der Knoten und Gewichte

Im folgenden benötigen wir den Begriff der orthogonalen Funktionen:

7.12 Definition. *(i) Durch* $\langle g,h\rangle := \int_a^b w(x)g(x)h(x)\mathrm{d}x$ *für reellwertige Funktionen* g *und* h *ist ein Skalarprodukt gegeben. Wir sagen,* g *und* h *sind orthogonal (in Zeichen:* $g \perp h$*), wenn* $\langle g,h\rangle = 0$*. Natürlich gilt* $g \perp h \;\Leftrightarrow\; h \perp g$*.*

(ii) Im folgenden bezeichnen wir mit P_j *die Menge aller Polynome höchstens* j*-ten Grades.*

Nehmen wir nun an, wir hätten ein Polynom $p_n \in P_n$ mit $p_n \perp P_{n-1}$, d.h. mit

$$p_n \perp q \qquad \forall q \in P_{n-1}$$

und n paarweise verschiedene reelle Nullstellen $a < x_1 < x_2 < \cdots < x_{n-1} < x_n < b$. Diese Nullstellen nehmen wir als Knoten in der Quadraturformel (7.61). Damit kann ein Genauigkeitsgrad von $2n-1$ erreicht werden.

7.13 Beweis. *Für das Polynom*

$$q(x) := \prod_{i=1}^{n}(x - x_i)^2$$

gilt offensichtlich: $q \in P_{2n}$ *mit* $I(q) > 0$*, aber* $\tilde{I}(q) = 0$*. Die Quadraturformel kann daher höchstens den Genauigkeitsgrad* $2n - 1$ *haben.*

Für alle $j = 0, \ldots, n-1$ *gilt* $I(p_n x^j) = \langle p_n, x^j \rangle = 0$*, da* $x^j \in P_{n-1}$ *und*

$$\tilde{I}(p_n x^j) = \sum_{i=1}^{n} w_i p_n(x_i) x_i^j = 0,$$

da $p_n(x_i) = 0$*. Damit ist* $\tilde{I}(f) = I(f)$ *für alle Funktionen aus*

$$\text{Span}\left(p_n, xp_n, x^2 p_n, \ldots, x^{n-1} p_n\right).$$

Wenn wir nun die Gewichte $w_1, w_2, \ldots, w_n$ *so wählen können, daß außerdem* $I(x^j) = \tilde{I}(x^j)$ *für alle* $j = 1, \ldots, n-1$ *gilt, so ist die Quadraturformel exakt in*

$$\text{Span}\left(1, x, x^2, \ldots, x^{n-1}, p_n, xp_n, x^2 p_n, \ldots, x^{n-1} p_n\right) = P_{2n-1}.$$

Dazu müssen die Gewichte das lineare Gleichungssystem

$$\sum_{i=1}^{n} w_i x_i^j = I(x^j), \qquad j = 0, \ldots, n-1$$

erfüllen. Die Koeffizientenmatrix M *dieses Gleichungssystems ist regulär.*
Denn: Es ist

$$M^\intercal = \begin{pmatrix} x_1^0 & x_1^1 & \ldots & x_1^{n-1} \\ x_2^0 & x_2^1 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^0 & x_n^1 & \ldots & x_n^{n-1} \end{pmatrix}.$$

Sei $a = (a_1, \ldots, a_n)^\intercal$ *mit* $M^\intercal a = 0$*. Dann hat das Polynom*

$$\sum_{i=1}^{n} a_i x^{i-1} \in P_{n-1}$$

n *paarweise verschiedene Nullstellen. Nach dem Fundamentalsatz der Algebra muß daher* $a = 0$ *sein, d.h.* $M^\intercal$ *(und damit* M*) ist regulär.*
Damit besitzt das obige Gleichungssystem genau eine nichttriviale Lösung $(w_1, \ldots, w_n)^\intercal$*.*

Bemerkung. Die LAGRANGE-Polynome $L_j(x) := \prod_{k=1, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}$ $(j = 1, 2, \ldots, n)$ mit $n \geq 2$ (siehe 2.1.2) bilden wie die Polynome $1, x, x^2, \ldots, x^{n-1}$ eine Basis in P_{n-1}. Wir können daher auch $I(L_j) = \tilde{I}(L_j)$ $(j = 1, 2, \ldots, n)$ fordern und erhalten dann die Gewichte direkt nach

$$w_j = \sum_{i=1}^{n} w_i L_j(x_i) = \int_a^b w(x) L_j(x) \mathrm{d}x \qquad (j = 1, \ldots, n \text{ mit } n \geq 2). \tag{7.62}$$

Wir sind von einem Polynom $p_n \in P_n$ mit $p_n \perp P_{n-1}$ und n paarweise verschiedenen reellen Nullstellen ausgegangen, Ein solches Polynom kann mit dem SCHMIDT*'schen Orthogonalisierungsverfahren* konstruiert werden: Für eine gegebene Gewichtsfunktion werden rekursiv die Polynome

$$p_0 := 1, \tag{7.63a}$$

$$p_j := x^j - \sum_{k=0}^{j-1} \frac{\langle x^j, p_k\rangle}{\langle p_k, p_k\rangle} p_k, \qquad j = 1, \ldots, n \tag{7.63b}$$

definiert. Dann gilt für $j = 1, \ldots, n$:

(i) $p_j \in P_j$,

(ii) $p_j \perp q$ für alle $q \in P_{j-1}$,

(iii) p_j hat j paarweise verschiedene reelle Nullstellen im Inneren des Definitionsbereiches der Gewichtsfunktion.

Diese Polynome werden als die *zur Gewichtsfunktion zugehörigen orthogonalen Polynome* bezeichnet.

Bekannte Orthogonalpolynome.

(a) Für $w(x) = 1$ $(x \in [-1;1])$ erhält man die LEGENDRE-Polynome

$$p_j(x) := \frac{1}{2^j j!}\sqrt{\frac{2j+1}{2}}\frac{\mathrm{d}^j}{\mathrm{d}x^j}(x^2-1)^j \qquad (j = 0, 1, 2, \ldots).$$

(b) Für $w(x) = (1-x^2)^{-1/2}$ $(x \in [-1;1])$ erhält man die TSCHEBYSCHEFF-Polynome

$$p_0(x) := \pi^{-1/2}, \qquad p_j(x) = \sqrt{\frac{2}{\pi}}\cos(j \cdot \arccos x) \qquad (j = 1, 2, 3, \ldots).$$

(c) Für $w(x) = \mathrm{e}^{-x}$ $(x \in [0;\infty[)$ erhält man die LAGUERRE-Polynome

$$p_j(x) := \frac{(-1)^j}{j!}\mathrm{e}^{-x}\frac{\mathrm{d}^j}{\mathrm{d}x^j}(x^j\mathrm{e}^{-x}) \qquad (j = 0, 1, 2, \ldots).$$

(d) Für $w(x) = \mathrm{e}^{-x^2}$ $(x \in]-\infty;\infty[)$ erhält man die HERMITE-Polynome

$$p_j(x) := \frac{(-1)^j}{\sqrt{\pi^{1/2}2^j j!}}\mathrm{e}^{x^2}\frac{\mathrm{d}^j}{\mathrm{d}x^j}\mathrm{e}^{-x^2} \qquad (j = 0, 1, 2, \ldots).$$

Die Forderung nach einem Genauigkeitsgrad von $2n-1$ legt die n Knoten und n Gewichte in der Quadraturformel (7.61) eindeutig fest. Dabei sind die Knoten die Nullstellen des Polynoms $p_n(x)$. Wie auch die Gewichte sind diese Nullstellen für mehrere Gewichtsfunktionen bekannt (tabelliert); beide müssen also nicht nochmals berechnet werden (siehe z.B. Tabelle 7.1).

Die Quadraturformel (7.61) wird als *Formel von* GAUSS oder genauer z.B. *Formel von* GAUSS *und* LEGENDRE $(w(x) = 1)$ bezeichnet.

7.14 Beispiel. $f(x) := x\sin(3x)$ $(x \in [-1, 1])$. $I(f) = 0.691355$

Zusammengesetze Trapezregel:

N	$\tilde{I}(f)$
10	0.564319
15	0.623384
20	0.648170

GAUSS-LEGENDRE-Regel:

n	$\tilde{I}(f)$
3	0.627979

Beim Trapezverfahren sind hier etwa fünf mal so viele Funktionsauswertungen erforderlich wie beim GAUSS-LEGENDRE-Verfahren, um eine vergleichbare Genauigkeit zu erzielen.

$n=1$:	$x_1 = 0$	$w_1 = 2$	$(\sqrt{3/2}x)$
$n=2$:	$x_1 = -0.5773502692$	$w_1 = 1$	$(\sqrt{5/8}(3x^2-1))$
	$x_2 = 0.5773502692$	$w_2 = 1$	
$n=3$:	$x_1 = -0.7745966692$	$w_1 = 0.5555555556$	$(\sqrt{7/8}(5x^2-3x))$
	$x_2 = 0$	$w_2 = 0.8888888889$	
	$x_3 = 0.7745966692$	$w_3 = 0.5555555556$	
$n=4$:	$x_1 = -0.8611363116$	$w_1 = 0.3478548451$	$(\sqrt{9/128}(35x^4-30x^2+3))$
	$x_2 = -0.3399810436$	$w_2 = 0.6521451549$	
	$x_3 =$	$w_3 = 0.6521451549$	
	$x_4 = -0.8611363116$	$w_4 = 0.3478548451$	

Tabelle 7.1: Koeffizienten für die Formel von GAUSS und LEGENDRE

7.7 Mehrdimensionale Integrale

Wir betrachten hier nur zweidimensionale Integrale. Bei solchen Integralen sind ähnliche Quadraturverfahren wie in einer Dimension sinnvoll einsetzbar.

Sei $K \subset \mathbb{R}^2$ ein kompakter Bereich und $f : K \to \mathbb{R}$ integrierbar. Gesucht ist ein Näherungswert für das Integral

$$\int_K f(x,y)\mathrm{d}x\mathrm{d}y. \tag{7.64}$$

Hat der Integrationsbereich die Rechteckform

$$K = \{(x,y) | a \le x \le b,\ c \le y \le d\} \tag{7.65}$$

mit $a < b$ und $c < d$, so wird

$$\int_K f(x,y)\mathrm{d}x\mathrm{d}y = \int_a^b \left(\int_c^d f(x,y)\mathrm{d}y \right) \mathrm{d}x. \tag{7.66}$$

Für festes x kann man nun eine (summierte) Quadraturformel bezüglich y zwischen c und d benutzen. Für das verbleibende Integral wird wieder eine eindimensionale (summierte) Quadraturformel verwendet.

Ist beispielsweise die zugrunde liegende Formel in beiden Fällen die SIMPSON-Formel, so gilt die *Kubaturformel*

$$\begin{aligned} \int_K f(x,y)\mathrm{d}x\mathrm{d}y \quad \approx \quad & \frac{b-a}{6}\Bigg[\frac{d-c}{6}\left(f(a,c) + 4f(a,\tfrac{c+d}{2}) + f(a,d)\right) \\ & + 4\frac{d-c}{6}\left(f(\tfrac{a+b}{2},c) + 4f(\tfrac{a+b}{2},\tfrac{c+d}{2}) + f(\tfrac{a+b}{2},d)\right) \\ & + \frac{d-c}{6}\left(f(b,c) + 4f(b,\tfrac{c+d}{2}) + f(b,d)\right)\Bigg] \end{aligned} \tag{7.67}$$

Ist K polygonal berandet, so ist eine Zerlegung in Dreiecke $T_1, \ldots, T_N$ möglich (sonst nur näherungsweise). Man spricht dann von einer *Triangulierung* (siehe Abbildung 7.4). Dann gilt

$$\int_K f(x,y)\mathrm{d}x\mathrm{d}y = \sum_{j=1}^{N} \int_{T_j} f(x,y)\mathrm{d}x\mathrm{d}y. \tag{7.68}$$

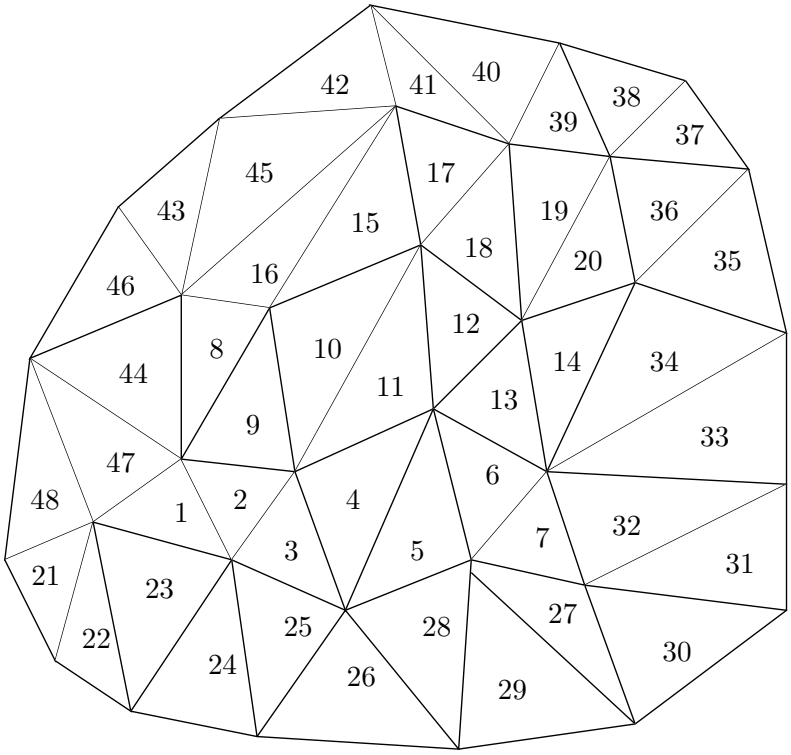


Abbildung 7.4: Triangulierung

Alle Einzelintegrale $\int\limits_{T_j} f(x,y)\mathrm{d}x\mathrm{d}y$ lassen sich auf ein Integral über das sogenannte *Standarddreieck*

$$T_S := \{(x,y) : 0 \le x \le 1,\ 0 \le y \le 1-x\} \tag{7.69}$$

zurückführen: Hat T_j die Eckpunkte $(x_{j,1}, y_{j,1})$, $(x_{j,2}, y_{j,2})$, $(x_{j,3}, y_{j,3})$, so bildet die lineare Transformation

$$\varphi_j : \begin{pmatrix}\xi\\ \eta\end{pmatrix} \mapsto \begin{pmatrix}\varphi_{j,1}(\xi,\eta)\\ \varphi_{j,2}(\xi,\eta)\end{pmatrix} := \begin{pmatrix}x_{j,1}\\ y_{j,1}\end{pmatrix} + \begin{pmatrix}x_{j,2}-x_{j,1} & x_{j,3}-x_{j,1}\\ y_{j,2}-y_{j,1} & y_{j,3}-y_{j,1}\end{pmatrix}\begin{pmatrix}\xi\\ \eta\end{pmatrix} \tag{7.70}$$

T_S auf T_j bijektiv ab. Mit der Substitution $(x,y) =: (\phi_{j,1}(\xi,\eta), \phi_{j,2}(\xi,\eta))$ ergibt sich

$$\int\limits_{T_j} f(x,y)\mathrm{d}x\mathrm{d}y = \underbrace{\begin{vmatrix}x_{j,2}-x_{j,1} & x_{j,3}-x_{j,1}\\ y_{j,2}-y_{j,1} & y_{j,3}-y_{j,1}\end{vmatrix}}_{\text{Funktionaldeterminante von } \varphi} \int\limits_{T_S} f(\varphi_{j,1}(\xi,\eta), \varphi_{j,2}(\xi,\eta))\,\mathrm{d}\xi\mathrm{d}\eta\,. \tag{7.71}$$

Somit ergibt sich

$$\int\limits_K f(x,y)\mathrm{d}x\mathrm{d}y = \sum_{j=1}^{N} \left[(x_{j,2}-x_{j,1})(y_{j,3}-y_{j,1}) - (x_{j,3}-x_{j,1})(y_{j,2}-y_{j,1})\right] \int\limits_{T_S} f(\varphi_{j,1}(\xi,\eta), \varphi_{j,2}(\xi,\eta))\mathrm{d}\xi\mathrm{d}\eta\,. \tag{7.72}$$

Es genügt daher, Integrale der Bauart $\int\limits_{T_S} g(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta$ zu approximieren.

Einige Kubaturformeln.

(a) $\int\limits_{T_S} g(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta \approx \frac{1}{2} g(\frac{1}{3},\frac{1}{3})$ (*Schwerpunkstregel*); Exaktheitsgrad 1.

(b) $\int\limits_{T_S} g(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta \approx \frac{1}{6}\left(g(\frac{1}{2},0)+g(\frac{1}{2},\frac{1}{2})+g(0,\frac{1}{2})\right)$; Exaktheitsgrad 2.

(c) $\int\limits_{T_S} g(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta \approx \frac{1}{60}\left(g(\frac{1}{2},0)+g(\frac{1}{2},\frac{1}{2})+g(0,\frac{1}{2})\right)+\frac{9}{60}\left(g(\frac{1}{6},\frac{4}{6})+g(\frac{1}{6},\frac{1}{6})+g(\frac{4}{6},\frac{1}{6})\right)$ (*Regel von Collatz und Albrecht*); Exaktheitsgrad 3.

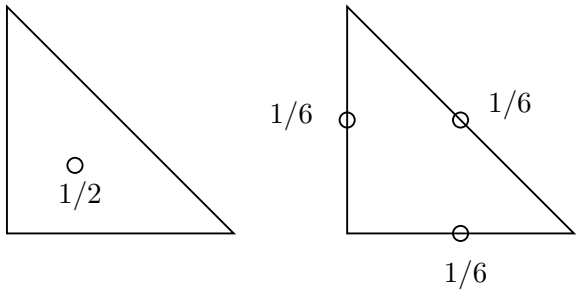


Abbildung 7.5: Kubaturformal (a) (links) und Kubaturformel (b) (rechts)

Die oben beschriebenen Verfahren können auf dreidimensionale Integrale übertragen werden und sind dann noch sinnvoll einsetzbar. Aber in höheren Dimensionen wird der Aufwand untragbar. Hier hilft nur das sogenannte Monte-Carlo-Verfahren (siehe Kapitel 8).

Kapitel 8

Zufallszahlen und Monte-Carlo-Methoden

In diesem Kapitel werden wir uns mit Monte-Carlo-Methoden beschäftigen. Mit diesem Verfahren können Näherungslösungen für mathematisch-naturwissenschaftliche (und andere, z.B. ökonomische) Probleme *unter Verwendung von Zufallszahlen* ermittelt werden. Der wesentliche Ansatz besteht hierbei darin, theoretische *Erwartungswerte* durch *Mittelwerte* über geeignete Stichproben zu nähern. Die dabei auftretenden Fluktuationen dieser Mittelwerte führen zu statistischen Varianzen, die bei der Interpretation der Ergebnisse zu berücksichtigen sind. Interessanterweise lassen sich mit Monte-Carlo-Methoden Problemstellungen bearbeiten, die sowohl stochastischer als auch *deterministischer* Natur sein können.

Im Folgenden wollen wir einige Anwendungsgebiete aufzählen.

- Statistik (Hypothesen-Tests, Parameternäherungen)
- Optimierung
- Hochdimensionale Integration
- Differentialgleichungen
- Physikalische Prozesse

Voraussetzung für das Verfahren sind ein Zufallszahlengenerator und die statistische Auswertung der Stichproben.

8.1 Einige zufällige Bemerkungen über Zufallszahlen

Man unterscheidet:

- *Echte Zufallszahlen*, z.B. aus radioaktiven Zerfällen (umständlich, wenige)
- *Pseudozufallszahlen*, deterministisch über einen Algorithmus erzeugt.
  Forderung: Statistsch unabhängig, gleichverteilt!
- *Quasizufallszahlen* ("subrandom")
  optimale Gleichverteilung, aber nicht unabhängig (z.B. für Monte-Carlo-Integration). Vgl. Ergänzung 8D.

Anforderungen an Pseudozufallszahlengenerator:

- "gute" Statistik → muss durch Tests ermittelts werden!

- lange Periode (aufgrund des erzeugenden Algorithmus wiederholen sich Zufallszahlen)
- Reproduzierbarkeit (insbesondere zum Testen von Programmen)
- Portabilität: Es sollte möglich sein, auf verschiedenen Rechnern (und mit verschiedenen Programmiersprachen) die gleichen Ergebnisse zu erzielen.
- schnell

8.1.1 Der (multiplikative) lineare Kongruenz-Zufallszahlengenerator

- (fast) ausschließlich für "normale" Zwecke implementiert
- es existieren aber auch andere Methoden, z.B. auf Verschlüsselungsverfahren basierend.

Prinzipieller Algorithmus.

$$\begin{aligned} \texttt{SEED} &= (A \cdot \texttt{SEED} + \texttt{C}) \bmod \texttt{M} \qquad (\texttt{SEED},\ A,\ C,\ M \text{ ganze Zahlen } (\texttt{Integer})) \\ x &= \texttt{SEED}/\texttt{M} \qquad \text{Normalisierung, Umwandlung in } \texttt{Real}\text{-Zahl} \end{aligned} \tag{8.1}$$

- Bei gegebenem Anfangswert `SEED` wird ein neuer `SEED`-Wert ("1. Zufallszahl") berechnet, darauf basierend die 2. Zufallszahl, usw. Da `SEED` immer `Integer` mod M ist, gilt

$$0 \leq \texttt{SEED} < \texttt{M}, \qquad \text{d.h. } x \in [0, 1).$$

- M ist die maximale Zahl der verschiedenen Zufallszahlen.
- zyklisches Verhalten: ein neuer Zyklus beginnt, wenn

$$\texttt{SEED}(\text{aktuell}) = \texttt{SEED}(\text{Start})$$

- wollen einen möglichst langen Zyklus (Periode)
  $\Rightarrow M$ sollte (in etwa) der größte darstellbare `Integer` auf dem System sein.
- des Weiteren wollen wir eine *gleichförmige* Verteilung der Zufallszahlen
  $\Rightarrow$ Abstand der Zufallszahlen, die in einem Zyklus erzeugt werden, sollte gleich sein
  $\Rightarrow$ Optimum: *alle* Zahlen $0, \ldots, M-1$ werden erzeugt!

8.1 Satz (Linearer Kongruenz-Zufallszahlengenerator mit $C > 0$). *Die maximal mögliche Periode M (d.h.,* alle *Zahlen $0, \ldots, M-1$ werden erzeugt) wird unter folgenden Bedingungen erreicht:*

(i) C muss teilerfremd gegenüber M sein.

(ii) Für jede Primzahl, die M dividiert, muss $(A-1)$ ein Vielfaches von dieser Primzahl sein.

(iii) Wenn M ein Vielfaches von 4 ist, muss auch $(A-1)$ eim Vielfaches von 4 sein.

8.2 Beispiel (Der Fall $M = 2^n$). ($n \leq 31$ für 4-Byte `Integer`)
Aufgrund Bedingung (i) muss C ungerade sein (z.B. $C = 1$). Aufgrund (iii) muss $(A-1)$ ein Vielfaches von 4 sein. Hierbei ist dann Bedingung (ii) enthalten, da 2 die einzige Primzahl ist, die M dividiert; wenn $(A-1)$ ein Vielfaches von 4 ist, ist es natürlich auch ein Vielfaches von 2. Ein "legitimer" Wert ist z.B. $A = 5$.

- Im Folgenden betrachten wir den Zyklus für $n = 5$, d.h. $M = 32$ und den Werten $A = 5$ und $C = 1$.
- Mit diesen Zahlen sollte dann laut obiger Aussage der *maximale* Zyklus "durchfahren", d.h. 32 verschiedene Zahlen erzeugt werden
- und demzufolge auch unabhängig vom Startwert sein.

↓	9	1	25	17	9 (← neuer Zyklus)
	14	6	30	22	14
	7	31	23	15	7
	4	28	20	12	·
	21	13	5	29	·
	10	2	26	18	·
	19	11	3	27	
	0	24	16	8	

Tabelle 8.1: Sequenz der Zufallszahlen $\texttt{SEED} = (5 \cdot \texttt{SEED} + 1) \bmod 32$ mit Startwert 9.

Die betrachtete Sequenz lautet z.B.

$$\texttt{SEED} \quad (\text{Start}) = 9$$
$$\texttt{SEED} \quad (1) = 5 \cdot 9 + 1 \bmod 32 = 46 \bmod 32 = 14, \text{ etc.},$$

vgl. obige Tabelle 8.1

Problem: die Portabilität.

Da wir einen möglichst langen Zyklus erzielen wollen, sollte $(M-1)$ in etwa dem größten darstellbaren `Integer` entsprechen ($2^{31}-1$ bei 4-Byte). Aufgrund des erzeugenden Algorithmus (8.1) ergeben sich bei der Berechnung von `SEED` allerdings Zwischenwerte, die *größer* als M sind und damit nicht mehr dargestellt werden können (z.B. dann, wenn `SEED` irgendwann im Zyklus in der Größenordnung von M selbst liegt. Man vergleiche mit obigem Beispiel, wo dieses Problem schon bei der Erzeugung der ersten Zufallszahl, 46 mod 32, auftritt).

Dieses Problem läßt sich zwar in Maschinensprache unter Verwendung von 8-Byte-Produktregistern umgehen, dann ist das Verfahren aber nicht mehr portabel!

Einen Ausweg aus dem Dilemma wies Schrage (1979). Er zeigte nämlich, dass sich Multiplikation des Typs

$$(A \cdot \texttt{SEED}) \bmod \texttt{M}, \tag{8.2}$$

wobei A, `SEED` und M eine bestimmte Länge (z.B. 4 Byte) haben, durchführen lassen, *ohne Zwischenwerte zu erzeugen, die größer als diese Länge sind* (incl. Vorzeichenbit). (Das eigentliche Verfahren wird weiter unten diskutiert.)

Falls dieser "Trick" verwendet werden soll, heißt das natürlich, dass der Zufallszahlenalgorithmus mit $C \equiv 0$ auskommen muß! Insbesondere ist damit der Startwert $\texttt{SEED} = \texttt{0}$ verboten, und die "0" kommt im gesamten Zyklus nicht vor. Es gilt dann, dass $x \in (0, 1)$, und es lassen sich folgende Aussagen in Abhängigkeit der Werte von M und A treffen.

Wesentliche Aussagen für einen linearen Kongruenz-Zufallszahlengenerator mit $C \equiv 0$.

8.3 Satz (Fall A: Linearer Kongruenz-Zufallszahlengenerator mit $C \equiv 0$ und $M = 2^n$). *Falls $M = 2^n$, läßt sich eine Periode $P = 2^{n-2}$ erzielen, falls der Startwert ungerade und $(A-3)$ oder $(A-5)$ ein Vielfaches von 8 sind. Dann gilt*

- *Alle Werte im Zyklus sind ungerade.*
- *Es existieren zwei separate und gleich lange Zyklen mit den Minima 1 bzw. 5.*
  $\Rightarrow$ (ungerade und 2 Zyklen) $P = 2^{n-2}$

Falls $M = 2^n$ und der Startwert *gerade* ist, ergeben sich viele verschiedene Unterzyklen.

8.4 Beispiel (Linearer Kongruenz-Zufallszahlengenerator mit $C \equiv 0$). In Tabelle 8.2 geben wir die zwei Sequenzen an, die sich im Falle $n = 7$ mit $A = 11$ und ungeraden Startwerten (1 und 5) ergeben. Aufgrund obiger Aussage haben beide Zyklen eine Länge von $P = 32$.

1	5
11	55
121	93
51	127
49	117
27	7
41	77
67	79
97	101
43	87
89	61
83	31
17	85
59	39
9	45
99	111
65	69
75	119
57	29
115	63
113	53
91	71
105	13
3	15
33	37
107	23
25	125
19	95
81	21
123	103
73	109
35	47
–	–
1	5
11	55

Tabelle 8.2: Sequenzen der Zufallszahlen $\texttt{SEED} = (\texttt{11} \cdot \texttt{SEED}) \bmod \texttt{128}$ mit Startwerten 1 und 5. Die Länge der Periode ist $128/4 = 32$.

Die Verwendung eines solchen Zufallszahlengenerators nach Fall (A) birgt zumindest zwei Probleme.

- Die erzeugte Sequenz ist nicht besonders gleichmäßig: Wenn die Zahlen entsprechend ihrem numerischen Wert geordnet sind, lauten die Differenzen zweier benachbarter Zahlen nicht überall 4 (wie es sein sollte), sondern entweder 2 oder 6.
- Erheblich gravierender ist es allerdings, dass derart erzeugte Zufallszahlen stark miteinander korreliert, d.h. die Zahlen *nicht* statistisch unabhängig sind. Mehr dazu in Kap. 8.1.3 und Beispiel 8.14.

8.5 Satz (Fall B: Linearer Kongruenz-Zufallszahlengenerator mit $C \equiv 0$ und M Primzahl). *Falls M eine Primzahl ist, dann wird die Periode $M-1$ (keine Null!) erzielt, falls der Startwert > 0 und "A primitiv modulo M" (komplexe Definition) ist.*

Mit einem Zufallszahlengenerator entsprechend Fall (B) läßt sich also die maximal mögliche Periode und damit eine gleichmäßige Verteilung erreichen. Ebenso tritt hier das Problem der Korrelation der erzeugten Zahlen praktisch nicht auf (vgl. Bsp. 8.14).

8.1.2 Der Minimum-Standard Zufallszahlengenerator

Ausgehend von diesen Überlegungen und unter Verwendung des im Weiteren geschilderten "Tricks" von SCHRAGE, schlugen PARK und MILLER (1988) als einfachst möglichen Zufallszahlengenerator für den Fall $C \equiv 0$ den sog. "Minimum-Standard" Generator vor. Hierbei wird

$$M = 2^{31} - 1 = 2147483647$$

verwendet, wobei dieser größte darstellbare 4-Byte-`Integer` eine Primzahl ist. (Tatsächlich ist diese Zahl eine sog. MERSENNEsche Primzahl : Wenn p eine Primzahl ist, ist $2^p - 1$ oftmals - aber nicht immer - auch eine Primzahl, die dann als MERSENNEsch bezeichnet wird.) Mit diesem Wert von M erhalten wir dann entsprechend Fall (B) ca. $2.15 \cdot 10^9$ verschiedene Zufallszahlen $\in (0, 1)$, falls wir für A die Zahlen

$$A = \begin{cases} 16807 = 7^5 \\ 48271 \\ 69621 \end{cases}$$

verwenden, die sowohl die Bedingung "A primitiv modulo M" erfüllen als auch die im Folgenden geschilderte Zusatzbedingung, die zur Durchführung von SCHRAGEs Multiplikation ohne Zwischenwerte $> M$ erforderlich ist.

SCHRAGES **Multiplikation** $(A \cdot$ `SEED`$)$ mod `M`**:**

Man zerlege zunächst M so, dass

$$M = A \cdot q + r \qquad \text{mit } q = \left[\frac{M}{A}\right] \tag{8.3}$$

wobei die eckige Klammer hier eine `Integer`-Operation bedeuten soll. Damit gilt, dass

$$r = M \bmod A$$

.

8.6 Beispiel. Sei $M = 2^{31} - 1$ und $A = 16807$. Dann ist

$$\begin{aligned} q &= \left[\frac{M}{A}\right] = 127773 \\ r &= M \bmod A = 2836 \end{aligned}$$

und damit

$$M = 16807 \cdot 127773 + 2836 = 2^{31} - 1.$$

Mit diesen Definitionen von q und r läßt sich Folgendes beweisen:

8.7 Satz. *Falls in obiger Zerlegung (8.3)* $r < q$ *und* $0 < \texttt{SEED} < \texttt{M} - 1$*, dann gilt*

(i)

$$\left.\begin{array}{r} A \cdot (\texttt{SEED} \bmod \texttt{q}) \\ r \cdot \left[\frac{\texttt{SEED}}{q}\right] \end{array}\right\} < M - 1, \tag{8.4}$$

d.h. darstellbar*! Gesucht ist*

$\texttt{SEED}^{\text{neu}} = (\texttt{A} \cdot \texttt{SEED}) \bmod \texttt{M}$*. Sei nun*

$$\texttt{DIFF} := \texttt{A} \cdot (\texttt{SEED} \bmod \texttt{q}) - \texttt{r} \cdot \left[\frac{\texttt{SEED}}{\texttt{q}}\right]. \tag{8.5}$$

Aufgrund von (8.4) gilt damit auch, dass

$$|\texttt{DIFF}| < \texttt{M} - 1$$

ebenfalls darstellbar ist.

(ii) Der gesuchte Wert $\texttt{SEED}^{\text{neu}}$ *lässt sich dann ohne Zwischenwerte* $> M - 1$ *über*

$$\texttt{SEED}^{\text{neu}} = \begin{cases} \texttt{DIFF}, & \textit{falls } \texttt{DIFF} > 0 \\ \texttt{DIFF} + \texttt{M}, & \textit{falls } \texttt{DIFF} < 0 \end{cases} \tag{8.6}$$

berechnen!

8.8 Beispiel. Berechnung von $(3 \cdot 5) \bmod 10$ ohne Zwischenwerte $> (M - 1) = 9$ ($A = 3$ und $\texttt{SEED} = 5$):

$$\left.\begin{array}{r} q = \left[\frac{10}{3}\right] = 3 \\ r = 10 \bmod 3 = 1 \end{array}\right\} \qquad \text{also } r < q \text{ und } \texttt{SEED} < \texttt{M} - 1 = 9.$$

Die Voraussetzungen sind also erfüllt, und wir haben entsprechend (8.5, 8.6)

$$(A \cdot \texttt{SEED}) \bmod \texttt{M} = \texttt{A} \cdot (\texttt{SEED} \bmod \texttt{q}) - \texttt{r}\left[\frac{\texttt{SEED}}{\texttt{q}}\right] = \underbrace{3 \cdot 2}_{<\texttt{M}-1} - \underbrace{1 \cdot 1}_{<\texttt{M}-1} = 5.$$

Letztlich bleibt es noch zu zeigen, dass die Voraussetzung $r < q$ für den Minimum-Standard Generator ($M = 2^{31} - 1$) und den angegebenen Werten von A erfüllt ist:

A	$q = \left[\frac{M}{A}\right]$	$r = M \bmod A$		
16807	127773	2836	$< q$	ok
48271	44448	3399	$< q$	ok
69621	30845	23902	$< q$	ok

8.9 Beispiel. $M = 2^{31} - 1 = 2147483647$, $\texttt{SEED} = \texttt{1147483647}$, $A = 69621$. Direkte Berechnung führt zu

$$\texttt{SEED}^{\text{neu}} = \underbrace{79888958987787}_{>\texttt{M}(*)} \bmod \texttt{M} = \texttt{419835740},$$

wobei $(*)$ nicht mit 4 Bytes darstellbar ist.

Mit dem Verfahren nach SCHRAGE erhält man andererseits ohne "Überlauf"

$$\begin{aligned} \texttt{DIFF} &= A(\texttt{SEED} \bmod \texttt{M}) - \texttt{r}\left[\frac{\texttt{SEED}}{\texttt{q}}\right] \\ &= \underbrace{1309014042}_{<M-1} - 889178302 \\ &= 419835740 \overset{!}{=} \texttt{SEED}^{\text{neu}}. \end{aligned}$$

8.10 Beispiel. `SEED` = `308450001` (`SEED` mod `q` = `1`). Direkte Berechnung führt zu

$$\texttt{SEED}^{\text{neu}} = \texttt{21474597519621} \bmod \texttt{M} = \texttt{1908533268}.$$

Mit SCHRAGE erhält man

$$\texttt{DIFF} = \texttt{69621} - \texttt{23902000} = -\texttt{238950379},$$

und wegen `DIFF` < 0 – vgl. (8.6) – ist also

$$\texttt{SEED}(\text{neu}) = \texttt{DIFF} + \texttt{M} = \texttt{1908533268}.$$

Wiederum ergeben sich keine Zwischenwerte $> M - 1$.

Algorithmus für PARK-MILLER Minimum-Standard Generator mit SCHRAGEs Multiplikation:

```
RAN(ISEED)
Real RAN
Integer (4 Byte), Parameter[1]::
M = 2147483647, A = 69621,
q = 30845, r = 23902
Real, Parameter:: M1 = 1./Float(M)
Integer (4 Byte):: ISEED, K


  K = ISEED / q
  ISEED = A * (ISEED - K * q) - r * K

  IF (ISEED < 0)  ISEED = ISEED + M
  RAN = ISEED * M1
  END
```

(Man beachte, dass `(ISEED - K * q)` der Operation `SEED` mod `q` entspricht.) Zur Berechnung einer Zufallszahl mittels Standard-Minimum Generator werden also nur vier Anweisungen benötigt, der Rest sind Deklarationen! Vor dem ersten Aufruf von `RAN` ist die Variable `ISEED` (der Startwert) zu initialisieren, mit $0 <$ `ISEED` $< M - 1$. Danach darf `ISEED` nur noch durch `RAN` selbst verändert werden!

8.1.3 Tests von Zufallszahlengeneratoren

Zunächst gilt es zu testen, ob die von dem jeweiligen Generator erzeugten Zufallszahlen gleichmäßig in $(0, 1)$ verteilt sind[2]. Dazu betrachten wir zuerst folgendes generelles Problem:

Es sei $N_i, i = 1, \ldots, k$ mit

$$\sum_{i=1}^{k} N_i =: N$$

die jeweilige Anzahl eines bestimmten Ereignisses i und wir wollen testen, wie groß die Warscheinlichkeit ist, dass die jeweiligen N_i's einer *vorgegebenen* Zahl n_i entsprechen.

8.11 Beispiel (Test von Würfeln). Wir würfeln; dabei sei

N_1 die Anzahl der Würfe mit Zahlen $1, \ldots, 4$,
N_2 die Anzahl der Würfe mit Zahlen 5 und 6,
$N = N_1 + N_2$.

Wir wollen testen, ob N_1 und N_2 entsprechend der Erwartung $n_1 = \frac{2}{3}N$, $n_2 = \frac{1}{3}N$ verteilt sind, d.h. ob der Würfel nicht manipuliert ist.

[1] entspricht einer *globalen Konstanten* in `C++`

[2] "Gleichmäßig" bedeutet, dass die Wahrscheinlichkeitsdichte $p(x)\mathrm{d}x = \mathrm{d}x$, d.h. konstant ist (vgl. Kap. 8.2.3)

Methode. Um solche Tests durchführen zu können, berechnet man PEARSONs χ^2_p,

$$\chi^2_p = \sum_{i=1}^{k} \frac{(N_i - n_i)^2}{n_i}. \tag{8.7}$$

Dann gilt folgender

8.12 Satz. PEARSONs χ^2_p *(8.7) ist eine Stichprobenfunktion, die* asymptotisch *(d.h. für $N_i \gg 1 \; \forall \; i$) der Quadratsumme von f unabhängigen Zufallsvariablen entspricht, die normalverteilt mit Mittelwert 0 und Varianz (=σ^2) 1 sind. Die Zahl der Freiheitsgrade ist dabei wie folgt definiert:*

- *Falls n_i fixiert ist, gilt $f = k$.*
- *Falls n_i so normalisiert ist, dass $\sum n_i = \sum N_i = N$, gilt $f = k - 1$ (*eine *Zwangsbedingung).*
- *Falls m zusätzliche Zwangsbedingungen vorhanden sind, gilt $f = k - m - 1$.*

Aufgrund dieses Satzes[3] hat also PEARSONs χ^2_p asymptotisch die "normale" χ^2-Verteilung (bzw. $Q(\chi^2, f)$-Verteilung, vgl. Kapitel 5.3), wobei "asymptotisch" in Praxis heißt, das in jedem "Kanal" i mindestens $n_i = 5$ Ereignisse *zu erwarten* sind.

8.13 Beispiel (Fortsetzung: Test von Würfeln). Mit den Werten für $N_i, n_i, i = 1, 2$ wie zuvor angegeben berechnet sich PEARSONs χ^2_p zu

$$\chi^2_p = \sum \frac{(N_i - n_i)^2}{n_i} = \frac{(N_1 - \frac{2}{3}N)^2}{\frac{2}{3}N} + \frac{(N_2 - \frac{1}{3}N)^2}{\frac{1}{3}N}.$$

Aufgrund der Forderung, dass in jedem Kanal mindestens 5 Ereignisse zu erwarten sein sollen, müssen mindestens $N = 15$ Würfe durchgeführt werden. Mit dieser Zahl ergibt sich

$$\chi^2_p = \frac{(N_1 - 10)^2}{10} + \frac{(N_2 - 5)^2}{5},$$

und die Zahl der Freiheitsgrade ist $f = 1$.

In Praxis könnte dies dann wie folgt aussehen: Wir testen vier verschiedene Würfel, und erhalten noch jeweils 15-maligem Würfeln die in Tabelle 8.3 angegebenen Resultate. Die Größe Q berechnet sich hierbei entsprechend Gl. 5.16.

Würfel	N_1	N_2	χ^2_p	$Q(\chi^2_p, f = 1)$
1	10	5	0	1
2	1	14	24.3	$8.24 \cdot 10^{-7}$
3	4	11	10.8	10^{-3}
4	8	7	1.2	0.27

Tabelle 8.3: Vier Würfel im Test

Entsprechend unserer Diskussion der Q-Werte in Kap. 5.3.2 lassen sich die Ergebnisse wie folgt interpretieren:

Würfel 1: Das Ergebnis ist "zu gut", nochmals würfeln.

Würfel 2: Das Ergebnis ist äußerst unwahrscheinlich, und damit der Würfel mit ziemlicher Sicherheit gefälscht. (Der wirtschafliche Erfolg von Spielbanken zeigt aber, dass derartige Ergebnisse durchaus auftreten können!)

Würfel 3: Das Ergebnis ist unwahrscheinlich, aber durchaus möglich; nochmals würfeln.

Würfel 4: Das Ergebnis ist OK, der Würfel scheint nicht gefälscht zu sein.

Bemerkung. "Sicherheit" gibt nur das mehrmalige Durchführen des Experiments, danach ist zu testen, ob die individuellen χ^2_p der χ^2-Verteilung folgen!

[3] Der Beweis ist ziemlich komplex, man findet ihn z.B. in FISZ, M., *Warscheinlichkeitsrechnung und mathematische Statistik*. VEB Deutscher Verlag der Wissenschaften, Berlin 1966; Seite 365.

Eine alternative Formulierung von PEARSONS χ^2_p lautet wie folgt:

$$\chi^2_p = \sum_{i=1}^{k} \frac{(N_i - n_i)^2}{n_i} = \sum_{i=1}^{k} \frac{(N_i - p_i N)^2}{p_i N}, \tag{8.8}$$

wobei

- k die Anzahl der Kanäle,
- N_i die Zahl der Ereignisse im Kanal i mit $\sum_i N_i = N$,
- p_i die Wahrscheinlichkeit der Ereignisse im Kanal i.

sind. Die Kanäle sind dabei so gewählt, dass *alle* möglichen Ereignisse aufgefangen werden, d.h. $\sum_i p_i = 1$.

Nach diesen generellen Vorbemerkungen kommen wir nun zurück zum

Test von Zufallszahlen. Die Frage war: Sind die erzeugten Zufallszahlen gleichmäßig in $(0, 1)$ verteilt?

Zur Beantwortung der Frage teilen ("binnen") wir das Intervall $(0, 1)$ in k gleich große Bereiche auf, erzeugen N Zufallszahlen und verteilen diese ihrem Wert gemäß auf die entsprechenden Kanäle. Dann zählen wir die Häufigkeiten N_i in jedem Kanal. Bei einer Gleichverteilung muss $p_i = \frac{1}{k}$ für *alle* i gelten!

Wir berechnen also

$$\chi^2_p = \sum_{i=1}^{k} \frac{(N_i - \frac{N}{k})^2}{N/k},$$

wobei in jedem Kanal der zu erwartende Wert N/k mindestens 5 sein sollte, also mindesten $N = 5k$ Zufallszahlen zu ziehen sind.

- Wir berechne nun die Wahrscheinlichkeit $Q(\chi^2_p, k-1)$ und
- wiederholen den Test l-mal, wobei typscherweise $l \approx 10$ Testreihen durchgeführt werden sollten.
- Dann testen wir, ob die individuellen $\chi^2_{p,j} (j = 1, \ldots, l)$ χ^2-verteilt sind (mit $k-1$ Freiheitsgraden). Dies lässt sich mit dem sog. KOLMOGOROFF-SMIRNOV-Test erzielen[4] .
- Falls die individuellen Q_j "vernünftig" sind und der KOLMOGORROFF-SMIRNOV-Test ebenfalls eine "vernünftige" Warscheinlichkeit ergibt, liefert der Zufallsgenerator tatsächlich *gleichförmig verteilte* Zahlen, zumindest bzgl. der definierten Kanäle.

Damit sind wir zwar einen Schritt vorangekommen, aber noch nicht fertig! Die Zufallszahlen sollen nämlich nicht nur gleichförmig verteilt, sondern auch statistisch unabhängig sein.

Mit anderen Worten: Die Wahrscheinlichkeit, dass nach der Ziehung der Zahl x_i die Zahl x_{i+1} gezogen wird, soll gleich der Wahrscheinlichkeit für x_{i+1} selbst sein, d.h. unabhängig von x_i.

$$P(x_{i+1}|x_i) \stackrel{!}{=} P(x_{i+1}) \quad \forall i,$$

(vgl. Ergänzung 8.A).

Zu diesem Zwecke benutzt man *mehrdimensionale* Tests, die die Korrelation mehrerer Zufallszahlen überprüfen.

[4] siehe z.B. *Numerical Recipes*, Kap. 14.3

a) Zweidimensionale Tests.
$P(x_{i+1}|x_i) \stackrel{!}{=} P(x_{i+1})$ überprüft, ob zwei aufeinanderfolgende Zahlen statistisch unabhängig sind.

Dazu *binnen* wir das Einheitsquadrat $(0,1)^2$ in zwei Dimensionen, erzeugen Paare von Zufallszahlen und sortieren diese in die entsprechenden "quadratischen" Kanäle ein (vgl. Abb. 8.1 für die Zufallszahlen $x_1 = 0.1, x_2 = 0.3, x_3 = 0.5, x_4 = 0.9$). Für k Kanäle in einer Dimension erhält man k^2 Kanäle in zwei Dimensionen, und wir erwarten $p = 1/k^2$ Zufallszahlen in jedem "quadratischen" Kanal. Damit ergibt sich PEARSONs χ^2_p zu

$$\chi^2_p = \sum_{i=1}^{k^2} \frac{(N_i - N/k^2)^2}{N/k^2},$$

wenn N_i die Zahl der gezogenen Paare im Kanal i ist. Als Minimum sind hier $N = 5k^2$ Zahlen zu ziehen. Wiederum sollten mehrere Testreihen j durchgeführt werden und sowohl die individuellen $\chi^2_{p,j}$ als auch ihre Verteilung mittels KOLMOGOROFF-SMIRNOV überprüft werden.

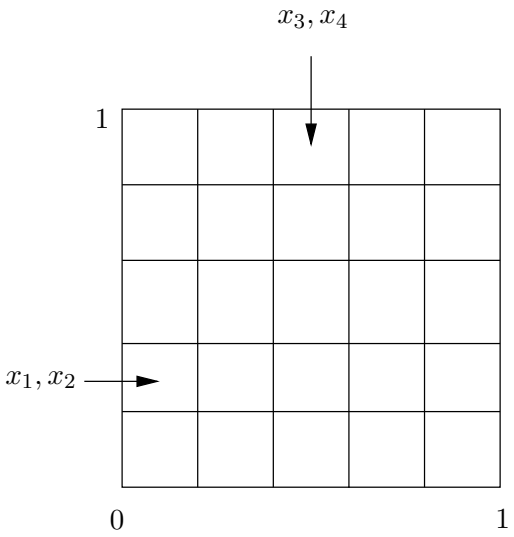


Abbildung 8.1: Zum 2-D Test (vgl. Text)

b) Dreidimensionale Tests.
$P(x_{i+2}|x_{i+1}, x_i) \stackrel{!}{=} P(x_{i+2})$ überprüft die Unabhängigkeit der dritten gezogenen Variablen von den beiden zuvor gezogenen. Dazu betrachtet man analog a) Kuben in drei Dimensionen im Quader $(0,1)^3$. Man bildet Tripel von drei aufeinander folgenden Zufallszahlen und sortiert diese in die Kuben ein. Die Wahrscheinlichkeit p_i ist hier durch $p_i = 1/k^3$ gegeben. Die Überprüfung erfolgt wie oben.

c) Man überprüft typischerweise bis zu 8 Dimensionen.

8.14 Beispiel (`RANDU`: Eine berühmte Panne). Einer der ersten auf Rechenanlagen implementieren Zufallszahlengeneratoren, `RANDU` (von IBM!), verwendete $M = 2^{31}$, $A = 65539$ und $C = 0$ entsprechend Fall (A) auf Seite 8-3 (mit Periode $P = 2^{29}$ für ungerade Startwerte). Führt man die oben angeführten Tests durch, so stellt sich heraus, dass der 1-D und 2-D Test anstandslos passiert wird. Führt man allerdings einen 3-D Test durch (was seinerzeit anscheinend für überflüssig gehalten wurde), so ergibt sich ein anderes Bild.

Wir betrachten hier einen Test in drei Dimentsionen mit $30^3 = 27\,000$ Kanälen und 270 000 erzeugten Zufallszahlen, d.h. es sind $N/(30^3) = 10$ Ereignisse pro Kanal zu erwarten. Da $f \gg 1$, sollte χ^2 in der Größenordnung von $f = (27\,000 - 1)$ liegen (vgl. Gl. 5.11). Im Folgenden vergleichen wir die Ergebnisse von 10 Testreihen für den PARK/MILLER Minimum-Standard Generator und für `RANDU`.

```
Park/Miller Minimum-Standard
----------------------------

chi2 =  26851.98     Q =  0.7373354
chi2 =  27006.20     Q =  0.4844202
chi2 =  26865.37     Q =  0.7192582
chi2 =  27167.66     Q =  0.2369688
chi2 =  26666.29     Q =  0.9249226
chi2 =  27187.38     Q =  0.2070863
chi2 =  27079.77     Q =  0.3661267
chi2 =  26667.37     Q =  0.9247663
chi2 =  27142.82     Q =  0.2668213
chi2 =  27135.24     Q =  0.2786968
```

Die Wahrscheinlichkeit, dass diese χ_p^2 der χ^2-Verteilung folgen, ist (mittels KOLMOGOROFF-SMIRNOV-Test) durch

```
KS-TEST: prob = 0.7328884
```

gegeben. Eine Betrachtung sowohl der individuellen χ_p^2 als auch letzterer Wahrscheinlichkeit zeigt, dass der Minimum-Standard Generator den 3-D Test ohne Probleme absolviert. Anders allerdings `RANDU`. Die analogen Resultate lauten hier

```
RANDU
-----

chi2 =  452505.8     Q =  0.0000000E+00  (in single precision)
chi2 =  454412.7     Q =  0.0000000E+00
chi2 =  456254.8     Q =  0.0000000E+00
chi2 =  454074.1     Q =  0.0000000E+00
chi2 =  452412.2     Q =  0.0000000E+00
chi2 =  452882.0     Q =  0.0000000E+00
chi2 =  453038.7     Q =  0.0000000E+00
chi2 =  455033.4     Q =  0.0000000E+00
chi2 =  453992.2     Q =  0.0000000E+00
chi2 =  453478.2     Q =  0.0000000E+00
```

(man betrachte die Gößenordnung von $\chi_p^2 \approx 17 \cdot f$) und der KS-Test liefert

```
KS-TEST: prob = 5.546628E-10 (!!!)
```

Mit anderen Worten: Obwohl `RANDU` gleichförmig verteilte Zahlen liefert, ist die jeweilig dritte Zahl sehr stark mit den beiden vorhergehenden korreliert! Demzufolge sind die Zahlen **nicht** statistisch unabhängig und damit keine (Pseudo)Zufallszahlen!

Eine zusätzliche Testmöglichkeit läßt sich durch eine graphische Darstellung der gezogenen Zufalls?-zahlen erzielen, die wir im Folgenden kurz behandeln.

Graphische Darstellung

Dabei zeigt sich sogleich ein generelles Problem, mit dem alle linear kongruenten Zufallszahlengeneratoren behaftet sind. *Plottet* man nämlich k hintereinander liegende Zufallszahlen in einem k-dimensionalen Raum, füllen sie den Raum nicht gleichmäßig aus, sondern liegen auf $(k-1)$-dimensionalen Hyperflächen. Die maximale Zahl dieser Hyperflächen ist dabei durch $M^{1/k}$ gegeben, wobei eine sorgfältige Wahl von A und C erforderlich ist, um dieses Maximum überhaupt zu erreichen!

In folgender Abbildung 8.2 veranschaulichen wir dieses Phänomen am Beispiel eines (kurzperiodischen) Generators mit dem Algorithmus

$$\texttt{SEED} = (65 * \texttt{SEED} + 1) \bmod 2048.$$

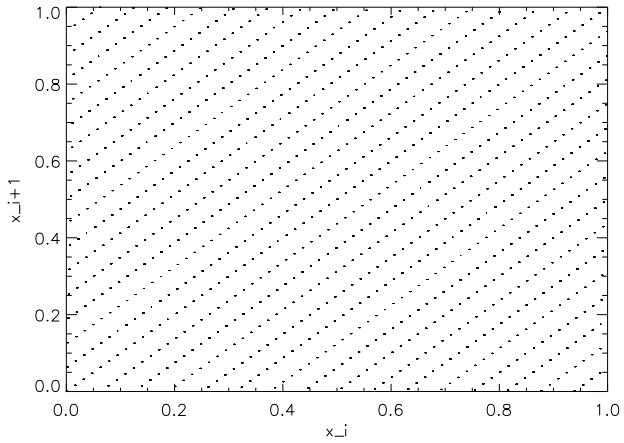


Abbildung 8.2: 2-D Darstellung der Zufallszahlen, die mittels $\texttt{SEED} = (65 * \texttt{SEED} + 1) \bmod 2048$ erzeugt wurden. Die Zahlen liegen auf 31 Hyperflächen.

Bezüglich einer 2-D Darstellung ist dann die maximale Zahl der Hyperflächen (hier: Geraden) durch $2048^{1/2} \approx 45$ gegeben. Wie ersichtlich, werden bei Verwendung von $A = 65$ und $C = 1$ zumindest 31 Hyperflächen erreicht.

Bei Verwendung von $M = 2^{31}$ sollte eine 3-D Darstellung dann maximal $(2^{31})^{1/3} \approx 1290$ (Hyper)Flächen ergeben. Führt man diese Auftragung für die von `RANDU` gelieferten Zahlen durch, liegen alle Zahlen jedoch auf nur 15 Flächen (vgl. Abb. 8.3), d.h. der zur Verfügung stehende Raum wird keinesfalls auch nur annähernd gleichmäßig ausgefüllt!

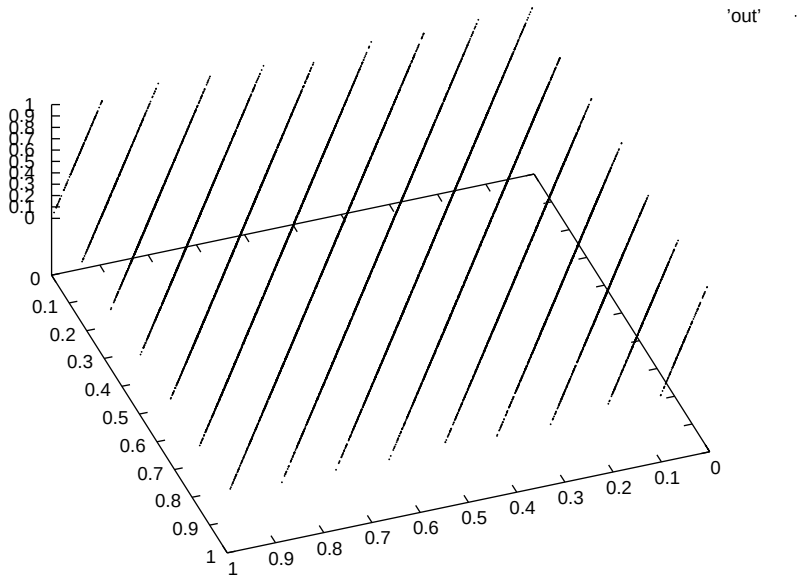


Abbildung 8.3: 3-D Darstellung der mittels `RANDU` erzeugten unzufälligen "Zufalls"zahlen. Obwohl bis zu 1290 Hyperflächen möglich sind, werden nur 15 realisiert!

Reshuffling

Um die soeben beschriebene und immer vorhandende Korrelation zu beseitigen, lässt sich als billigste Abhilfe das sogenannte *Reshuffling* verwenden. Dazu benutzt man die gezogene Zufallszahl selbst, um die Ordnung hintereinander folgender Zufallszahlen zu zerstören. Abbildung 8.4 zeigt den Algorithmus und Abbildung 8.5 das Resultat. Der Algorithmus läuft dabei folgendermaßen ab: Sei y die Zufallszahl aus der vorhergehenden Ziehung. Die neue Zufallszahl ergibt sich dann aus

(1) I = 1 + INTEGER(y · N) (Registerindex)

(2) OUTPUT = V(I) (Auslesen der Zufallszahl)
Y = OUTPUT (für die nächste Ziehung bereitstellen)

(3) V(I) = RAN (Auffüllen des Registers mit neuer Zufallszahl)

Oftmals benutzt man $N = 32$ Register V(1...N), die zunächst in einer "Warmlaufphase" aufgefüllt werden müssen (die ersten 8 erzeugten Zufallszahlen werden sogar verworfen), bevor die erste Zufallszahl gezogen werden kann.

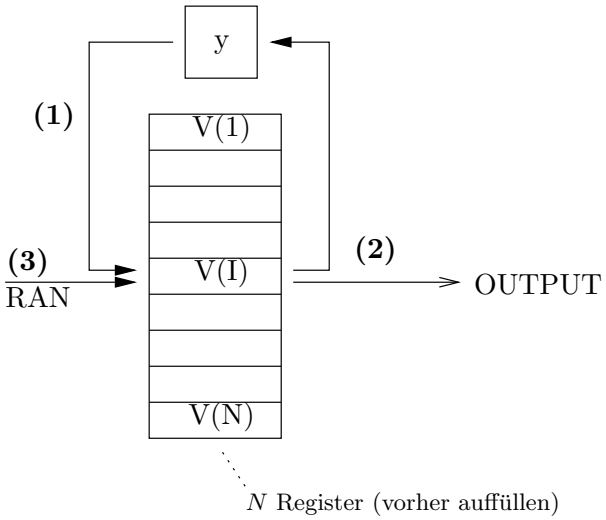


Abbildung 8.4: Der Algorithmus zum *Reshuffling*

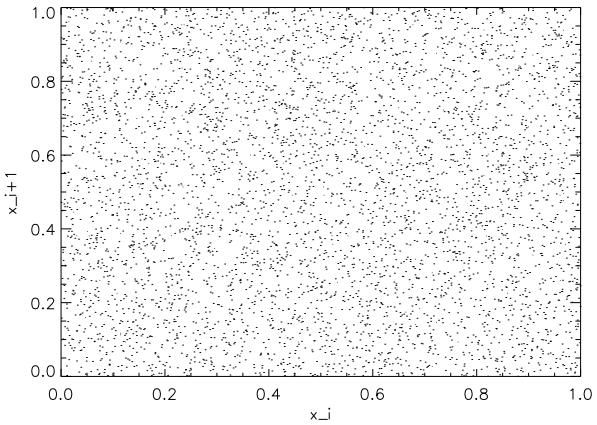


Abbildung 8.5: Wie Abb. 8.2, aber mit *Reshuffling* Augenscheinlich ist die Korrelation vollkommen zerstört.

8.2 Monte-Carlo-Methoden

8.2.1 Kurze Einführung des Wahrscheinlichkeitsbegriffes

- Ein physikalischer oder mathematischer Prozess werde durch ein Experiment simuliert.
- Dieses Experiment habe eine bestimmte Zahl (möglicherweise ∞) von möglichen Ergebnissen.
- Diesen Ergebnissen wird eine Wahrscheinlichkeit zugeordnet.
- Der sog. *Stichprobenraum* S umfasst dann den Raum *aller* möglichen Ergebnisse s, d.h. falls man das Experiment durchführt, liegt das Ergebnis dieses Experiments innerhalb S.
- Nach Durchführung des Experiments (mit möglichen Ergebnissen) hat also ein bestimmtes *Ereignis* E_k stattgefunden.
- Die Wahrscheinlichkeit, dass dieses Ereignis auftritt,

  $$P(E_k) = p_k,$$

  muss folgenden Bedingungen genügen:

  (a) $0 \leq p_k \leq 1$

  (b) Falls E_k nicht auftreten kann, ist $p_k = 0$.
  Tritt E_k mit Sicherheit auf, so ist $p_k = 1$.

  (c) Falls sich zwei Ereignisse E_i und E_j gegenseitig ausschließen, so gilt

  $$P(E_i \text{ und } E_j) = 0, \qquad P(E_i \text{ oder } E_j) = p_i + p_j$$

  (d) Falls sich N Ereignisse E_i, $i = 1, \ldots, N$, gegenseitig ausschließen und *vollständig* bzgl. S sind, gilt

  $$\sum_{i=1}^{N} p_i = 1. \tag{8.9}$$

- Führt man ein zweistufiges Experiment mit Ereignissen F_i und G_j durch, so bezeichnet man mit E_{ij} das zusammengesetzte Ereignis (G_j hat nach Eintreten von F_i stattgefunden).
  Falls F_i und G_j von einander *unabhängig* sind, gilt für die Wahrscheinlichkeit des zusammengesetzten Ereignisses

  $$p(E_{ij}) = p_{ij} = p(G_j) \cdot p(F_i), \tag{8.10}$$

  vgl. Anhang 8A.

8.2.2 Zufallsvariablen und darauf basierende Funktionen

Jedem Ereignis E_i wird nun eine reelle *Zufallsvariable* (Z.V.) x_i zugeordnet (z.B. dem Ereignis E_6, dass der Ausgang eines "Würfelexperimentes" der Wurf einer Sechs ist, wird die Z.V. $x_6 = 6$ zugeordnet).

Der *Erwartungswert* einer Zufallsvariablen (manchmal auch etwas salopp mit "*Mittelwert*" bezeichnet) bzw. der Erwartungswert einer beliebigen Funktion $g(x)$ dieser Z.V. – die dann selbst eine Z.V. ist – ist durch

$$E(x) = \overline{x} = \sum_{i=1}^{N} x_i p_i \tag{8.11}$$

$$E(g(x)) = \overline{g} = \sum_{i=1}^{N} g(x_i) p_i, \tag{8.12}$$

definiert, falls die Ereignisse E_i, $i = 1, \ldots, N$ vollständig sind und sich gegenseitig ausschließen. Man beachte, dass E ein linearer Operator ist, da

$$E(ag(x) + bh(x)) = aE(g) + bE(h). \tag{8.13}$$

Als *n-tes zentrales Moment* einer Zufallsvariablen bezeichnet man den Ausdruck

$$\overline{(x - \overline{x})^n},$$

das die "mittlere" n-te Potenz der Abweichung einer Z.V. von ihrem Erwartungswert angibt. Das wichtigste Beispiel dafür ist der Fall $n = 2$, die sog. *Varianz*,

$$\mathrm{Var}\,(x) = \overline{(x - \overline{x})^2} = \overline{x^2 - 2x \cdot \overline{x} + \overline{x}^2} = \overline{x^2} - \overline{x}^2, \tag{8.14}$$

die die "mittlere quadratische Abweichung vom Mittel" angibt. Die *Standardabweichung* σ ergibt sich daraus über

$$\sigma(x) = \left(\mathrm{Var}\,(x)\right)^{1/2}.$$

Im Gegensatz zum Erwartungswert ist die Varianz *kein* linearer Operator ist, da

$$\begin{aligned}
\mathrm{Var}\,(ag(x) + bh(x)) &= \overline{(ag + bh)^2} - \overline{ag + bh}^2 \\
&= \overline{a^2g^2 + 2abgh + b^2h^2} - \left(a^2\overline{g}^2 + 2ab\overline{g}\,\overline{h} + b^2\overline{h}^2\right) \\
&= a^2\left(\overline{g^2} - \overline{g}^2\right) + b^2\left(\overline{h^2} - \overline{h}^2\right) + 2ab\left(\overline{gh} - \overline{g}\,\overline{h}\right) \\
&= a^2\mathrm{Var}\,(g) + b^2\mathrm{Var}\,(h) + 2ab\underbrace{\left(\overline{gh} - \overline{g}\,\overline{h}\right)}_{=:\mathrm{cov}(g,h)},
\end{aligned} \tag{8.15}$$

wobei $\mathrm{cov}\,(g, h)$ die sog. *Kovarianz* ist.

Falls g und h statistisch *unabhängig* sind,

$$\overline{gh} = E(gh) = E(g) \cdot E(h) = \overline{g} \cdot \overline{h},$$

ist $\mathrm{cov}\,(g, h) = 0$ und es gilt

$$\mathrm{Var}\,(ag + bh) = a^2\mathrm{Var}\,(g) + b^2\mathrm{Var}\,(h)\,. \tag{8.16}$$

Schließlich ist der sog. *Korrelationskoeffizient* ϱ $(-1 \leq \varrho \leq 1)$ durch

$$\varrho(g, h) := \frac{\mathrm{cov}\,(g, h)}{\left(\mathrm{Var}\,(g)\,\mathrm{Var}\,(h)\right)^{\frac{1}{2}}} \tag{8.17}$$

definiert. Man beachte, dass $\mathrm{cov}\,(g, h) = 0$ auch dann möglich ist, wenn g, h statistisch *abhängig* sind!

8.2.3 Kontinuierliche Zufallsvariablen

Wir hatten bisher nur diskrete Ereignisse E_i betrachtet, denen diskrete Z.V. x_i zugeordnet wurden. Was passiert aber, wenn verschiedene Ereignisse nicht mehr aufgezählt werden können, sondern kontinuierlich sind (Beispiel: die Streuwinkel in einem Streuexperiment)?

Die Wahrscheinlichkeit, dass ein *bestimmtes* Ereignis auftritt (z.B. dass exakt ein bestimmter Streuwinkel auftritt) ist gleich Null; wohldefiniert ist aber die Wahrscheinlichkeit, dass der auftretende Wert innerhalb eines bestimmten (infinitesimalen) Intervalles liegt,

$$P(x \leq x' \leq x + \mathrm{d}x) = f(x)\mathrm{d}x, \tag{8.18}$$

wobei $f(x)$ als *Wahrscheinlichkeitsdichtefunktion* (pdf, "probability density function") bezeichnet wird. (Wir erinnern hier an die pdf der χ^2-Verteilung, $p(\chi^2, f)$, Kap. 5.3.1)

Die Wahrscheinlichkeit, dass das Ergebnis innerhalb eines Bereiches $[a, b]$ liegt, berechnet sich dann mittels pdf zu

$$P(a \leq x \leq b) = \int_a^b f(x')\mathrm{d}x'. \tag{8.19}$$

In Analogie zur diskreten Wahrscheinlichkeit p_i muss auch die pdf gewissen Anforderungen genügen:

$$\text{(a)} \qquad f(x) \geq 0, \qquad -\infty < x < \infty \tag{8.20a}$$

$$\text{(b)} \qquad \int_{-\infty}^{\infty} f(x')\mathrm{d}x' = 1 : \tag{8.20b}$$

die Wahrscheinlichkeit dafür, dass alle möglichen x auftreten können, ist gleich 1.

Die sog. *kumulative Wahrscheinlichkeitsverteilung* $F(x)$ gibt die Wahrscheinlichkeit an, dass alle Ereignisse bis zu einem bestimmten Maximalwert x auftreten können,

$$P(x' \leq x) \equiv F(x) = \int_{-\infty}^{x} f(x')\mathrm{d}x'. \tag{8.21}$$

Daraus folgt sofort, dass

- $F(x)$ eine monoton wachsende Funktion ist (aufgrund $f(x) \geq 0$),
- $F(-\infty) = 0$ und
- $F(\infty) = 1$.

8.15 Beispiel (1). Wir erinnern zunächst an den Zusammenhang zwischen χ^2-Verteilung und Fitgüte Q,

$$W(\chi^2 \leq \chi_0^2, f) = P(f/2, \chi_0^2/2) = 1 - Q(\chi_0^2, f),$$

vgl. Kap. 5.3

8.16 Beispiel (2). Eine *gleichförmige Verteilung* zwischen $[a, b]$ ist durch eine *konstante* pdf

$$f(x) = C \qquad \forall\, x \in [a, b]$$

definiert: Die Wahrscheinlichkeit, dass ein Wert $x \leq x' \leq x + \mathrm{d}x$ angenommen wird, ist konstant für alle $x \in [a, b]$. Für den restlichen Bereich gilt andererseits

$$f(x) = 0, \qquad (-\infty, a) \text{ und } (b, \infty).$$

Daraus folgt sofort, dass

$$F(\infty) = \int_{-\infty}^{\infty} f(x)\mathrm{d}x = \underbrace{\int_{-\infty}^{a} f(x)\mathrm{d}x}_{=0} + \int_a^b f(x)\mathrm{d}x + \underbrace{\int_b^{\infty} f(x)\mathrm{d}x}_{=0}$$

$$= \int_a^b f(x)\mathrm{d}x = C \cdot (b - a) \stackrel{!}{=} 1,$$

also

$$f(x) = \frac{1}{b - a} \qquad \text{für alle } x \in [a, b]. \tag{8.22}$$

Für gleichförmig verteilte Zufallszahlen, gezogen von einem Zufallszahlengenerator, bedeutet dies, dass die zugehörige pdf durch $f(x) = 1$ definiert ist. Die zugehörige kumulative Wahrscheinlichkeitsverteilung ergibt sich dann zu

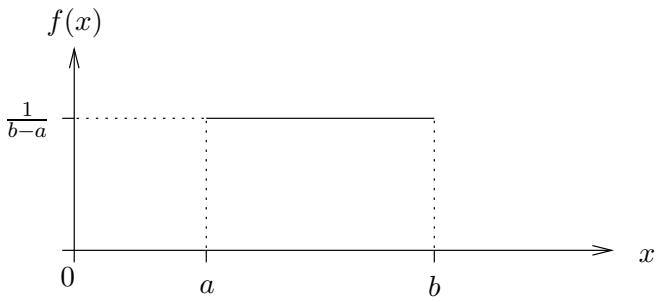


Abbildung 8.6: Wahrscheinlichkeitsdichtefunktion (pdf) $f(x)$ einer *gleichförmigen* Verteilung im Intervall [a,b].

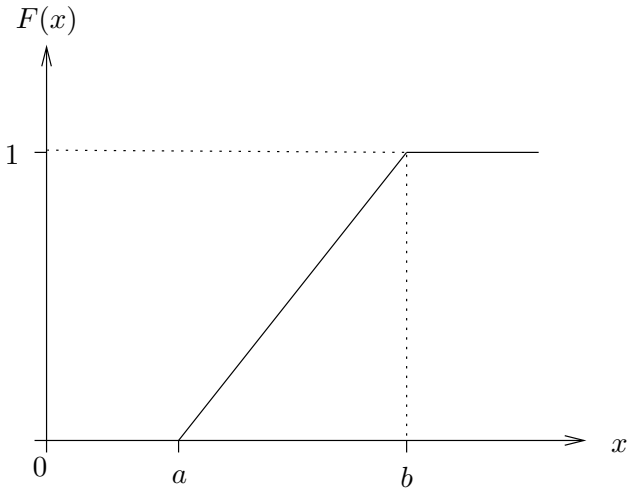


Abbildung 8.7: Zugehörige kumulative Wahrscheinlichkeitsverteilung $F(x)$

$$F(x) = \int_a^x f(x)\mathrm{d}x = \int_a^x \frac{\mathrm{d}x}{b-a} = \frac{x-a}{b-a} \qquad \forall\ x \in [a,b], \tag{8.23}$$

$$F(x) = 0 \qquad \text{für } x < a$$

$$F(x) = 1 \qquad \text{für } x > b$$

Für die Wahrscheinlichkeitsverteilung eines Zufallszahlengenerators bedeutet dies, dass $F(x) = x$ für $x \in [0,1] \approx (0,1)$. Abb. 8.6 und 8.7 veranschaulichen den geschilderten Sachverhalt.

Wiederum in Analogie zum diskreten Fall, sind Erwartungswert und Varianz von kontinuierlichen Z.V. folgendermaßen definiert:

$$E(x) \equiv \overline{x} \equiv \mu = \int_{-\infty}^{\infty} x' f(x')\mathrm{d}x' \tag{8.24a}$$

$$E(g) \equiv \overline{g} = \int_{-\infty}^{\infty} g(x') f(x')\mathrm{d}x' \tag{8.24b}$$

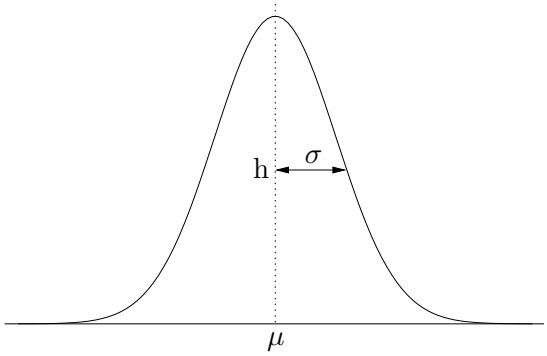


Abbildung 8.8: Die GAUSS-Verteilung, siehe Text ($h = f(\mu)\mathrm{e}^{-1/2} \approx 0.61 \cdot f(\mu)$).

$$\mathrm{Var}\,(x) \equiv \sigma^2 = \int_{-\infty}^{\infty} (x' - \mu)^2 f(x')\mathrm{d}x' \tag{8.25a}$$

$$\mathrm{Var}\,(g) \equiv \sigma^2(g) = \int_{-\infty}^{\infty} \left(g(x') - \overline{g}\right)^2 f(x')\mathrm{d}x' \tag{8.25b}$$

Abschließend wollen wir bemerken, dass sich diskrete Wahrscheinlichkeiten auch durch kontinuierliche Verteilungsfunktionen beschreiben lassen, wobei folgende Zusammenhang gilt:

$$f(x) = \sum_{i=1}^{N} p_i \delta(x - x_i). \tag{8.26}$$

$\delta(x)$ ist hier die DIRACsche δ-Funktion. Aus (8.24a) folgt dann für den Erwartungswert

$$\begin{aligned}\overline{x} = \int_{-\infty}^{\infty} x f(x)\mathrm{d}x &= \int_{-\infty}^{\infty} x \left[\sum_{i=1}^{N} p_i \delta(x - x_i)\right] \mathrm{d}x \\ &= \sum_{i=1}^{N} p_i \int_{-\infty}^{\infty} x\delta(x - x_i)\mathrm{d}x \stackrel{!}{=} \sum_{i=1}^{N} p_i x_i\end{aligned}$$

das schon in (8.11) angegebene Resultat.

8.17 Beispiel (Die GAUSS- oder Normalverteilung). Die pdf der GAUSS-Verteilung ist folgendermaßen definiert, man vergleiche mit Abb. 8.8:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Damit ergeben sich Erwartungswert und Varianz zu

$$E(x) = \mu \qquad \mathrm{Var}\,(x) = \sigma^2$$

und man findet als "1σ-Abweichung"

$$P(\mu - \sigma \leq x \leq \mu + \sigma) \approx 0.6826$$

8.2.4 Summen von Zufallsvariablen

Eine der wichtigsten Anwendungsgebiete von Monte-Carlo-Methoden ist die Monte-Carlo-*Integration*. Zur Einführung in dieses äußerst einfache Integrationsverfahren benötigen wir einige vorbereitende Betrachtungen, die insbesondere bestimmte Summen von Zufallsvariablen betreffen. Wir gehen dabei folgendermaßen vor:

- Zunächst werden N "Stichproben" $x_1, \ldots, x_N$ bzgl. einer vorgegebenen pdf $f(x)$ gezogen. Ein Beispiel hierfür wäre die Ziehung von gleichförmig verteilten Zufallszahlen, vgl. Beispiel 8.16. (Für beliebige pdf's wird das entsprechende Verfahren in Kap. 8.2.8 vorgestellt.)
- Danach berechnen wir eine Funktion $g_i(x_i) \in \mathbb{R}$, die dann ebenfalls eine Z.V. ist.
- Mit der Definition
$$G := \sum_{n=1}^{N} \lambda_n g_n(x_n) \qquad \lambda_n \in \mathbb{R}$$
ist ebenfalls G eine Z.V. Als Erwartungswert und Varianz ergibt sich
$$\text{(A)} \qquad E(G) = \overline{G} = E\left(\sum_{n=1}^{N} \lambda_n g_n(x_n)\right) \overset{(1)}{=} \sum \lambda_n E(g_n) = \sum \lambda_n \overline{g}_n$$
$$\text{(B)} \qquad \mathrm{Var}\,(G) = \mathrm{Var}\left(\sum_{n=1}^{N} \lambda_n g_n(x_n)\right) \overset{(2)}{=} \sum \lambda_n^2 \mathrm{Var}\,(g_n)\,,$$
wobei in (1) die Linearität von E verwendet wurde und (2) berücksichtigt, dass die jeweiligen Zufallsvariablen x_i (und damit g_i) statistisch unabhängig sind.
- Wir betrachten nun den Spezialfall
$$\lambda_n =: 1/N, \qquad g_n(x_n) =: g(x_n).$$
Daraus folgt, dass die damit konstruierte Zufallsvariable G,
$$G = \frac{1}{N} \sum_{n=1}^{N} g(x_n) \tag{8.27}$$
der *arithmetischer Mittelwert* der Z.V. $g(x)$ ist! Andererseits folgt aus obiger Relation (A)
$$E(G) = \overline{G} = \sum_{n=1}^{N} \frac{1}{N} \overline{g} = \overline{g}. \tag{8.28}$$
In Worten: Der Erwartungswert des arithmetischen Mittels von N gezogenene Zufallsvariablen $g(x)$ ist der Erwartungswert von g selbst (unabhängig von N)!
- Dies bildet die Basis der Monte-Carlo-Integration.

  Nähere den Erwartungswert einer Funktion durch das arithmetische Mittel über eine entsprechende Stichprobe!!!
$$\overline{g} = \overline{G} \approx G$$
- Aufgrund der Relation (B) finden wir für die Varianz des arithmetischen Mittels
$$\mathrm{Var}\,(G) = \sum_{n=1}^{N} \frac{1}{N^2} \mathrm{Var}\,(g(x_n)) = \frac{1}{N} \mathrm{Var}\,(g) \tag{8.29}$$
In Worten: Die Varianz des arithmetischen Mittels von N gezogenen Zufallszahlen ist um den Faktor $1/N$ kleiner als diejenige der originalen Zufallsvariablen $g(x)$.

  Implikation: Je größer die Stichprobe, um so kleiner wird die Varianz des arithmetischen Mittels, und damit die Näherung
$$\overline{g} \approx G$$
immer besser; insbesondere gilt, dass
$$G \approx \overline{G} \pm \sqrt{\mathrm{Var}\,(G)}, \qquad \text{d.h. } G \to \overline{G} = \overline{g} \qquad \text{für } N \to \infty$$
(Man erinnere sich, dass $\mathrm{Var}\,(G)$ die mittlere quadratische Abweichung von G bzgl. $\overline{G}$ ist.)

Zusammenfassung. Bevor wir diese Erkenntnisse nun auf die Monte-Carlo-Integration anwenden, wollen wir die abgeleiteten Zusammenhänge nochmals kurz zusammenfassen.

$$\left.\begin{array}{rl} \text{pdf} & f(x) \\ \text{Zufallsvariable} & g(x) \end{array}\right\} \quad \left.\begin{array}{l} E(g) \\ \text{Var}\,(g) \end{array}\right\} \quad \left\{\begin{array}{l} \text{wahrer Erwartungswert} \\ \text{wahre Varianz} \end{array}\right.$$

$$E(g) = \overline{g} = \int g(x) f(x) \mathrm{d}x$$
$$\text{Var}\,(g) = \int \left(g(x) - \overline{g}\right)^2 f(x) \mathrm{d}x$$

Ziehe x_n, $n = 1, \ldots, N$ entsprechend $f(x)$ und bilde das arithmetische Mittel

$$G(x) = \frac{1}{N} \sum_{n=1}^{N} g(x_n)$$

Dann gilt

$$\overline{G} = E(G) = E(g) = \overline{g}$$
$$\text{Var}\,(G) = \frac{1}{N} \text{Var}\,(g)$$
$$\sigma(G) = \left(\text{Var}\,(G)\right)^{1/2} = \frac{1}{\sqrt{N}} \sigma(g)$$

Die letzte Identität ist als "$1/\sqrt{N}$-Gesetz" der Monte-Carlo-Simulation bekannt. Der eigentliche "Trick" ist nun die

> Monte-Carlo-Näherung: Man nähere $\overline{g} = \overline{G}$ durch G

8.2.5 Monte-Carlo-Integration

Nach diesen vorbereitenden Betrachtungen kommen wir nun zum eigentlichen Verfahren. Zunächst formulieren wir das Integral I, das wir berechnen wollen, in geeigneter Weise um:

$$I = \int_a^b g(x) \mathrm{d}x = \int_a^b \frac{g(x)}{f(x)} f(x) \mathrm{d}x = \int_a^b h(x) f(x) \mathrm{d}x \tag{8.30}$$

wobei $f(x)$ nun eine Wahrscheinlichkeitsdichte sein soll. Mit dieser Schreibweise wird das Integral als ein bestimmter Erwartungswert von $h(x)$ interpretiert.

Bis auf Weiteres verwenden wir dabei eine konstante pdf, d.h. betrachen eine gleichförmige Verteilung im Intervall $[a, b]$,

$$f(x) = \begin{cases} \dfrac{1}{b-a} & a \le x \le b \\ 0 & \text{sonst,} \end{cases}$$

vgl. (8.22). Die "neue" Funktion $h(x)$ lautet also in diesem Fall

$$h(x) = \frac{g(x)}{f(x)} = (b-a) g(x),$$

und das gesuchte Integral ist proportional dem Erwartungswert von g:

$$I = (b-a)\int_a^b g(x)f(x)\mathrm{d}x \overset{\text{exakt!}}{=} (b-a)\overline{g}. \tag{8.31}$$

Es sei ausdrücklich betont, dass dieser Sachverhalt *exakt* gilt. Andererseits ziehen wir nun Stichproben x_n, $n = 1, \ldots, N$ aus der pdf, berechnen $g(x_n)$ und bilden das arithmetische Mittel

$$G = \frac{1}{N}\sum_{n=1}^{N} g(x_n).$$

Mit Hilfe der Monte-Carlo-Näherung ergibt sich damit eine äußerst einfache Abschätzung des Integrals I,

$$I = (b-a)\overline{g} \overset{\text{exakt}}{=} (b-a)\overline{G} \overset{\text{Monte-Carlo-Näherung}}{\approx} (b-a)G \tag{8.32}$$

Die auf diese Weise durchgeführte *Monte-Carlo-Integration* ist somit der (arithmetische) Mittelwert aus N "Beobachtungen" des Integranden, wobei die Zufallsvariable x gleichförmig aus dem Intervall $[a, b]$ gezogen wird.

Der Fehler, der aufgrund der Monte-Carlo-Näherung gemacht wird, läßt sich ebenfalls auf einfache Weise abschätzen.

$$\mathrm{Var}\,(G) = \frac{1}{N}\mathrm{Var}\,(g) = \frac{1}{N}\Big(\underbrace{\overline{g^2}}_{\overline{J}} - \underbrace{\overline{g}^2}_{\overline{G}^2}\Big) \approx \frac{1}{N}(J - G^2) \tag{8.33}$$

wenn

$$J = \frac{1}{N}\sum_{n=1}^{N} g^2(x_n)$$

der Monte-Carlo-Schätzwert für $E(J) = \overline{J} = \overline{g^2}$ ist und des Weiteren $\overline{g}^2 = \overline{G}^2$ durch G^2 genähert wird.

Identifizieren wir nun den Fehler des Integrals, δI, mit der Standardabweichung, die sich aus $\mathrm{Var}\,(G)$ ergibt[5] (dies läßt sich durch den "zentralen Grenzwertsatz" der Statistik beweisen), finden wir, dass dieser Fehler dem "$1/\sqrt{N}$-Gesetz" folgt,

$$\delta I \propto 1/\sqrt{N}.$$

Verallgemeinern wir nun die Integrationsvariable $\mathrm{d}x$ auf ein beliebig- dimensionales Volumenelement $\mathrm{d}V$, so ergibt sich folgendes *Rezept für die Monte-Carlo-Integration*:

$$\int_V g\mathrm{d}V \approx V\left(\langle g\rangle \pm \sqrt{\frac{1}{N}\left(\langle g^2\rangle - \langle g\rangle^2\right)}\right) \tag{8.34}$$

wenn

$$\langle g\rangle = G = \frac{1}{N}\sum_n g(x_n)$$
$$\langle g^2\rangle = J = \frac{1}{N}\sum_n g^2(x_n)$$

und V das Volumen des betrachteten Integrationsbereiches ist.

[5] multipliziert mit $(b-a)$

8.18 Beispiel. [Monte-Carlo-Integration der Exponentialfunktion] Als einfaches Bespiel wollen wir zunächst das Integral

$$\int_0^2 \mathrm{e}^{-x}\mathrm{d}x = 1 - \mathrm{e}^{-2} = 0.86467$$

über Monte-Carlo bestimmen.

(1) Wir ziehen dazu gleichförmig verteilte Zufallszahlen $\in [0, 2]$, und führen die Integration zunächst für die beiden Fälle a) $N = 5$ und b) $N = 25$ durch. (Da der Zufallszahlengenerator Zahlen liefert, die gleichförmig auf dem Intervall (0,1) verteilt sind, müssen sie zunächst mit einem Faktor zwei multipliziert werden, vgl. Kap 8.2.8, Gl. 8.39). Es werden z.B. folgende Zahlen gezogen:

	Fall a)		Fall b)
1	7.8263693E-05	1	0.1895919
2	1.3153778	2	0.4704462
3	1.5560532	3	0.7886472
4	0.5865013	4	0.7929640
5	1.3276724	5	1.3469290
		6	1.8350208
		7	1.1941637
		8	0.3096535
		9	0.3457211
		10	0.5346164
		11	1.2970020
		12	0.7114938
		13	7.6981865E-02
		14	1.8341565
		15	0.6684224
		16	0.1748597
		17	0.8677271
		18	1.8897665
		19	1.3043649
		20	0.4616689
		21	1.2692878
		22	0.9196489
		23	0.5391896
		24	0.1599936
		25	1.0119059

(2) Danach berechnen wir

$$G = \langle g \rangle = \frac{1}{N} \sum \mathrm{e}^{-X_n}$$
$$J = \langle g^2 \rangle = \frac{1}{N} \sum \left(\mathrm{e}^{-X_n}\right)^2$$

Die Resultate lauten
a) $G = 0.4601251$, $J = 0.2992171$
b) $G = 0.4904828$, $J = 0.2930297$

(3) Mit

$$I \approx V \left(\langle g \rangle \pm \sqrt{\frac{1}{N} \left(\langle g^2 \rangle - \langle g \rangle^2 \right)} \right)$$

und einem "Volumen" $V = 2$ resultiert

Fall a) $I \approx 0.9202501 \pm 0.2645783$
Fall b) $I \approx 0.9809657 \pm 0.091613322$.

Man beachte, dass das "$1/\sqrt{N}$-Gesetz" eine Reduktion des Fehlers von $1/\sqrt{5} = 0.447$ vorhersagt, was in unserem Fall (die erste Simulation wurde mit einer sehr kleinen Stichprobe durchgeführt) noch nicht ganz zutrifft, und dass das Resultat im Falle b) (bei dem angegeben Fehler) noch nicht den Realitäten entspricht.

Der Fehler bei $N = 25$ liegt in der Größenordnung ± 0.1. Um den Fehler auf ± 0.001 (Faktor 100) zu reduzieren, muß die Stichprobe demzufolge um einen Faktor 100^2 vergrößert werden, d.h. 250 000 Zufallszahlen müssen gezogen werden.

Das Ergebnis einer solchen Simulation mit $N = 250\,000$ lautet wie folgt:
$G = \langle g \rangle = 0.4322112$, $J = \langle g^2 \rangle = 0.2453225$ und damit

$$I = 0.8644224 \pm 0.9676 \cdot 10^{-3}$$

Wir stellen also fest, dass der (abgeschätze) Fehler tatsächlich auf die vorhergesagte Größenordnung abgefallen ist, und ein Vergleich des MC-Resultates mit dem exakten Wert des Integrals, $I(\text{exakt}) = 0.86476$, zeigt, dass er nun (bei dem angegebenen Fehler) in der richtigen Größenordung liegt.

Als vorläufiges Fazit läßt sich schon jetzt festhalten, dass die MC-Integration schnell zu programmieren ist, aber viele Rechnungen (Auswertungen des Integranden!) notwendig sind, um eine vernünftige Genauigkeit zu erzielen.

8.2.6 Wann ist die Monte-Carlo-Integration vorteilhaft?

Aufgrund der (sehr) großen Zahl an Auswertungen des Integranden, die auszuführen sind, um den Fehler deutlich einschränken (und der damit verbundenen Rechenzeit) stellt sich natürlich die Frage, ob überhaupt und wann eine Monte-Carlo-Integration vorteilhaft gegenüber Standardintegrationsverfahren (vgl. Kap. 7) ist. Zunächst ist festzuhalten, dass das "$1/\sqrt{N}$-Gesetz" des Fehlers unabhängig von der Dimensionalität des Problemes ist. Anders schaut es dagegen bei der Standardquadratur aus:

- Im Allgemeinen ist die Fehleranalsye bei mehr-D Integrationen äußerst komplex[6], aber für einfache Grenzen und NEWTON-COTES-Formeln in Analogie zum 1-D Fall durchführbar: Man erzeugt das mehr-D LAGRANGE-Polynom, integriert über das Restglied und erhält als Fehler

  $$\delta I = \left| I(f) - \tilde{I}(\tilde{f}) \right| \propto h^{2+d} \qquad \text{für Trapez- bzw. } \propto h^{4+d} \qquad \text{für Simpsonintegration}$$

  und *einfache* Formeln. d ist dabei die Dimension, vgl. Gln (7.15, 7.17) für 1-D Fall.

- Für zusammengesetzte Formeln und N äquidistante Stützstellen ergibt sich dann

  $$\delta I \propto N h^{2(4)+d}.$$

  Zum Vergleich der 1-D Fall explizit:

  $$N \propto 1/h \;\rightarrow\; \delta I \propto \frac{1}{h} h^{2(4)+1} = h^{2(4)},$$

  vgl. Gln (7.31, 7.32).

- Allgemein gilt $N \propto 1/h^d$, d.h. $h \propto (1/N)^{1/d}$, so dass der Integrationsfehler wie

  $$\delta I \propto N \cdot \left(N^{-1/d} \right)^{2(4)+d} = N^{-2(4)/d} \tag{8.35}$$

  skaliert.

Damit ergibt sich folgende Abhängigkeit des Fehlers von der *insgesamt* verwendeten Zahl an Stützstellen (bzw. gezogenen Zufallszahlen):

Bei der Monte-Carlo-Integration skaliert der Fehler mit $\sim N^{-1/2}$, während er bei der Standardquadratur (mit äquidistanten Stützstellen) von $\sim N^{-2(4)}$ (1-D), $\sim N^{-1(2)}$ (2-D) und $\sim N^{-\frac{2}{3}(\frac{4}{3})}$ (3-D) abhängt.

Erst ab $d = 5$ (Trapez) bzw. $d = 9$ (Simpson) verringert sich der Fehler der Monte-Carlo-Integration schneller als bei der Standardquadratur!

Allerdings gilt es zusätzlich folgendes zu berücksichtigen:

- Gerade bei mehrdimensionaler Integration sind *komplizierte Grenzen* der "Normalfall"

[6] siehe z.B. Stroud, A.H., 1971, *Approximate Calculation of Multiple Integrals*, Prentice-Hall

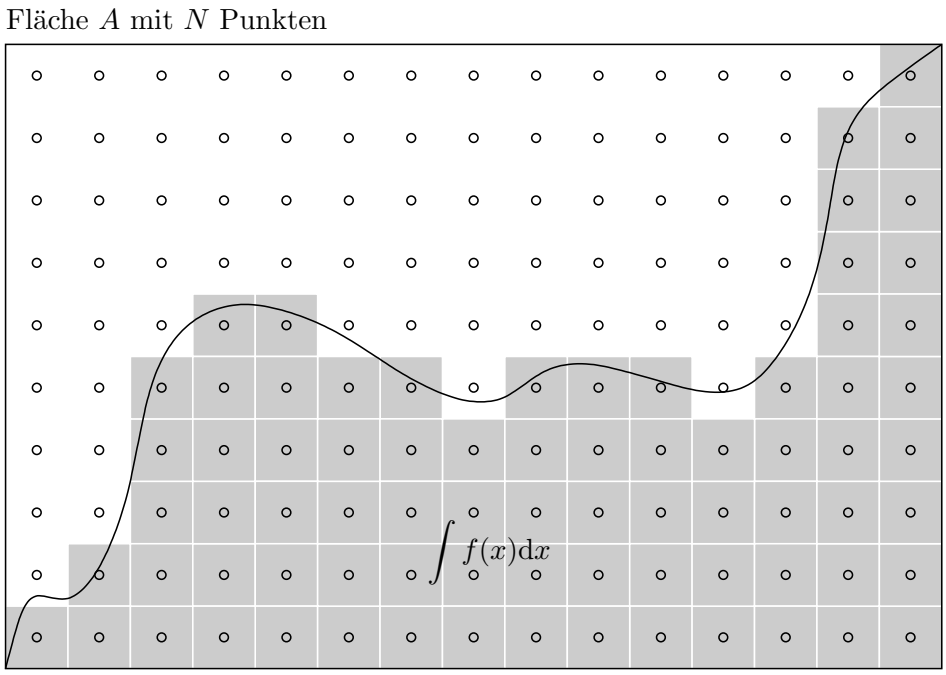


Abbildung 8.9: "Hit or miss"-Verfahren zur Berechnung von 1-D Integralen

- und die MC-Integration sollte hier bevorzugt werden, solange der Integrand nicht lokal konzentriert ist.
- Falls die Genauigkeitsanforderungen gering sind und insbesondere bei *Abschätzungen* ist die MC-Integration aufgrund ihrer Einfachheit ebenso zu bevorzugen.

8.2.7 Komplizierte Grenzen: "Hit or miss"

Zur Berechnung von mehr-D Integralen mit komplizierten Grenzen wird oftmals eine Variante der MC-Integration verwendet, die unter dem Namen "Hit or miss"-Verfahren bekannt ist. Bevor wir den allgemeinen Fall behandeln, sei das prinzipielle Verfahren für den 1-D Fall mit Hilfe von Abb. 8.9 kurz erläutert. Zur Berechnung des Integrales $I = \int f(x)\mathrm{d}x$

- verteilt man N Punkte (x_i, y_i) gleichförmig im dargestellten Rechteck mit der Fläche A.
- Man zählt all diejenigen Punkte als "Treffer", deren y-Koordinaten der Bedingung

$$y_i \leq f(x_i) \tag{8.36a}$$

  genügen. Bei n Treffern ergibt sich das Integral dann zu

$$I = \int f(x)\mathrm{d}x \approx A \cdot \frac{n}{N}. \tag{8.36b}$$

- Ein Nachteil des Verfahrens ist es natürlich, dass diejenigen $N - n$ Punkte, die keine Treffer sind (d.h. oberhalb von $f(x)$ liegen), "verschenkt" sind und keine Information beitragen: Aus diesem Grund sollte die umschreibende Fläche A so gering wie möglich gehalten werden.

Verallgemeinerung auf mehrere Dimensionen. Die Vorgehensweise soll anhand des (einfaches) Beispiel der Berechnung des Integrales von x über den halben Einheitskreis erläutert werden

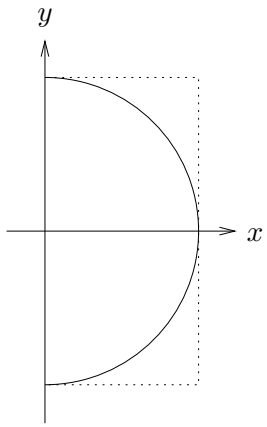


Abbildung 8.10: Halber Einheitskreis

(vgl. Abb. 8.10). Die analytische Lösung des Problemes lautet

$$\underset{\substack{\text{halber}\\ \text{Einheitskreis}}}{\iint} x\mathrm{d}x\mathrm{d}y = \int_{-\pi/2}^{\pi/2} \int_0^1 r\mathrm{d}r\ r\cos\varphi \mathrm{d}\varphi = \left.\frac{r^3}{3}\right|_0^1 \cdot \left[-\sin\varphi\right]_{-\pi/2}^{\pi/2} = \frac{2}{3}$$

Bei der Monte-Carlo-Integration mittels "hit or miss" geht man folgendermaßen vor:

(i) Zunächst erzeugt man im umschreibendem "Volumen" (hier das "obere" und "untere" Einheitsquadrat) gleichförmig verteilte "Zufallskoordinaten". In unserem Beispiel bedeutet dies

$$\begin{aligned} 0 \le x \le 1, \quad & \text{d.h. } \texttt{x = RANDOM(ISEED)}, \\ -1 \le y \le 1, \quad & \text{d.h. } \texttt{y = -1 + 2*RANDOM(ISEED)}, \end{aligned}$$

wenn `RANDOM(ISEED)` der entsprechende Zufallszahlengenerator ist und man berücksichtigt, dass nur Zahlen im Bereich (0,1) erzeugt werden (vgl. Kap. 8.2.8, Gl. 8.39).

(ii) Falls die Koordinaten innerhalb des Integrationsgebietes liegen, summiert man den Integranden als Funktion der Koordinaten auf (und sein Quadrat für die Varianz). In unserem Beispiel bedeutet dies:

`IF` $(x_i^2 + y_i^2) \le 1$ `THEN`
$\Sigma = \Sigma + x_i$
`Vari` $=$ `Vari` $+ x_i^2$
`ENDIF`

Man beachte, dass die Punkte außerhalb des Integrationsgebietes nichts zu den Summen beitragen.

(iii) Schließlich wird das Monte-Carlo-Integral und die Näherung für die Varianz wie üblich gebildet, d.h.

$$\begin{aligned} \text{Integral} &= \text{Vol} \cdot \Sigma/N \\ \sigma = (\text{Var})^{1/2} &= \text{Vol} \cdot \sqrt{\frac{\texttt{Vari}/N - (\Sigma/N)^2}{N}}, \end{aligned}$$

wobei Vol das umschreibende Volumen und N die Gesamtpunktzahl (incl. der nicht beitragenden Punkte) ist.

Wiederum ist das umschreibende Gebiet so klein wie möglich zu wählen. Falls das umschreibende Gebiet *bei gleicher Punktzahl* vergrößert wird, vergrößert sich insbesondere die Varianz, mit einem Faktor von ca. $\sqrt{\mathrm{Vol}_2/\mathrm{Vol}_1}$.

Dies läßt sich folgendermaßen begründen: Einerseits bleiben die MC-Schätzwerte für die Integrale, $\mathrm{Vol}\cdot\langle f\rangle$ bzw. $\mathrm{Vol}\cdot\langle f^2\rangle$ ungefähr gleich. Bei größerem umschreibenden Gebiet werden zwar die Mittelwerte $\langle f\rangle$, $\langle f^2\rangle$ kleiner, aber das Volumen wird größer. Damit gilt andererseits, dass

$$\mathrm{var}^2 = \frac{1}{N}\Big(\underbrace{\mathrm{Vol}^2\langle f^2\rangle}_{\substack{\text{wird ca. Faktor}\\ \text{Vol größer}}} - \underbrace{\mathrm{Vol}^2\langle f\rangle^2}_{\substack{\text{bleibt unge-}\\ \text{fähr konstant}}}\Big)$$

$$\rightarrow \qquad \frac{\mathrm{var}_2}{\mathrm{var}_1} = \frac{\mathrm{Vol}_2^2\langle f_2^2\rangle - \mathrm{Vol}_2^2\langle f_2\rangle^2}{\mathrm{Vol}_1^2\langle f_1^2\rangle - \mathrm{Vol}_1^2\langle f_1\rangle^2} \lesssim \frac{\mathrm{Vol}_2^2\langle f_2^2\rangle}{\mathrm{Vol}_1^2\langle f_1^2\rangle} \approx \frac{\mathrm{Vol}_2}{\mathrm{Vol}_1}$$

8.19 Beispiel (Berechnung des Integrals für zwei verschiedene Volumina). Im Folgenden ist der wesentliche Teil des Programmcodes (in `FORTRAN 90`) zur Berechnung des gesuchten Integrals angegeben, und zwar für zwei umschreibende Volumina, einmal mit den "vernünftigen" Grenzen $0 \le x \le 1$ und $-1 \le y \le 1$ und einmal für das 16-fach größere Gebiet $0 \le x \le 4$ und $-4 \le y \le 4$, um den Einfluss des Volumens auf die Varianz zu untersuchen.

```
call random_seed(put=iseed)              !Initialisierung

vol=2                                    !Vol_1
summ=0._sp

do i=1,n                                 !Schleife über n Zufallszahlen
call random_number(x1)                   !0 < x < 1
call random_number(y1)
y1=-1.+2*y1                              !-1< y < 1

  if(x1**2+y1**2 .le. 1.) then           !im Gebiet?
    summ=summ+x1                         !dann Aufsummation
    var=var+x1*x1
  endif
enddo

summ=summ*vol/n                          !MC Schätzwert für das Integral
var=vol*sqrt(((var/n)-(summ/n)**2)/n)    !MC Schätzwert für die Varianz
print*,summ,' +/- ',var                  !FERTIG mit Vol_1

call random_seed(put=iseed)               !Neue Initialisierung, um gleiche Sequenz zu erhalten
vol=32                                   !Grösseres Volumen Vol_2
summ=0._sp

do i=1,n
call random_number(x1)
call random_number(y1)
x1=4*x1                                  !0 < x < 4
y1=-4.+8*y1                              !-4< x < 4
  if(x1**2+y1**2 .le. 1.) then
    summ=summ+x1
    var=var+x1*x1
  endif
enddo

summ=summ*vol/n
```

```
var=vol*sqrt(((var/n)-(summ/n)**2)/n)
print*,summ,' +/- ',var                    !Fertig mit Vol_2
```

Bei $N = 1000$ Zufallszahlen erhalten wir beispielsweise folgende Ergebnisse (zur Erinnerung: das analytische Ergebnis lautet 2/3)

$$\begin{aligned} \mathrm{Vol}_1 = 2 \qquad & I = 0.6560 \pm 2.79 \cdot 10^{-2} \\ \mathrm{Vol}_2 = 32 \qquad & I = 0.6883 \pm 0.114, \end{aligned}$$

während sich bei $N = 10000$ Zahlen

$$\begin{aligned} \mathrm{Vol}_1 = 2 \qquad & I = 0.6607 \pm 8.80 \cdot 10^{-3} \\ \mathrm{Vol}_2 = 32 \qquad & I = 0.6769 \pm 3.35 \cdot 10^{-2} \end{aligned}$$

ergibt. Man beachte, dass sowohl das "$1/\sqrt{N}$-Gesetz" (Faktor $\sqrt{10}$) als auch die Abhängigkeit des Fehlers vom umschreibenden Volumen (ca. Faktor $\sqrt{(32/2)} = 4$) richtig wiedergegeben werden.

8.2.8 Erzeugung von Stichproben und Monte-Carlo-Simulation

Bislang haben wir uns im Rahmen der MC-Integration auf den einfachst möglichen Fall von pdf's beschränkt, nämlich denjenigen einer *gleichförmigen* (= konstanten) Verteilung, wie sie von Zufallszahlengeneratoren geliefert wird. Für viele Zwecke sind aber Stichproben bzgl. andersartiger Verteilungen zu ziehen, und in diesem Kapitel wollen wir diskutieren, wie dann vorzugehen ist. Stichproben bzgl. nicht-gleichförmiger Verteilungen werden dabei insbesondere in zwei Fällen benötigt:

(i) im Rahmen von sog. Varianzreduktionsverfahren; hierbei wird von der Möglichkeit Gebrauch gemacht, die bei MC-Simulationen (insbesondere bei Integrationen) auftretenden Varianzen durch "geschickt" gewählte pdf's (nicht-gleichförmig) teilweise erheblich zu reduzieren (siehe Ergänzung 8C).

(ii) im Rahen von Monte-Carlo-*Simulationen* im eigentlichen Sinne des Wortes, wobei die unterliegende "Philosophie" solcher Simulationen zunächst kurz skizziert werden soll.

Monte-Carlo-Simulationen.

- Wir haben eine komplexes physikalisches (oder wirtschaftliches etc.) Problem, das nur schwer oder gar nicht geschlossen lösbar ist.
- Wir kennen aber die "Physik" der einzelnen Subprozesse.
- Diese einzelnen Subprozesse werden dann im Rahmen von pdf's beschrieben, und in geeigneter Weise aneinander gehängt, um so das gesamte Problem zu beschreiben. Das finale Resultat ergibt sich dann durch einen vielfachen Durchlauf der Prozesskette, auf der Basis von mittels Zufallszahlen gezogenen Stichproben.

Vorteile: schnell, einfach zu programmieren, schnelle Resultate

Nachteile: tieferes Verständnis der Resultate schwierig, keine Möglichkeit, analytische Näherungen zu entwickeln.

8.20 Beispiel (Strahlungstransport in Sternatmosphären). Hier muss die sog. Strahlungstransportgleichung gelöst werden. Die Berechnung einer geschlossenen Lösung ist schon in einfacher Geometrie ein nicht-triviales Unterfangen, und in komplizierten Situationen (mehr-D, keine Symmetrie) praktisch unmöglich. Aus diesem Grund verwendet man oftmals Monte-Carlo- Simulationen, um die entsprechenden Ergebnisse zu erzielen.

Als Bespiel betrachten wir dazu folgendes Problem: Man berechne die räumliche Strahlungsenergieverteilung $E(\tau)$ in einer Atmosphäre, in der die Photonen nur an freien Elektronen gestreut werden (diese Problematik stellt sich oftmals in den äußeren Bereichen von sehr heißen Sternen).

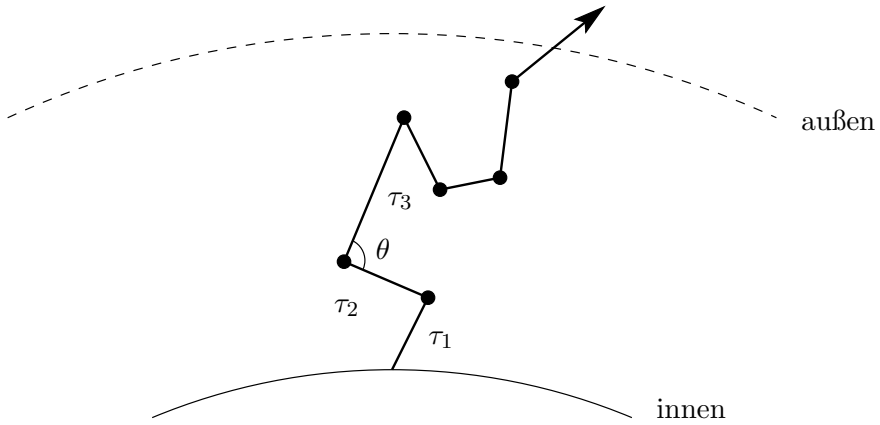


Abbildung 8.11: Monte-Carlo-Simulation: einfacher Strahlungstransport in Sternatmosphären.

Es läßt sich zeigen, dass dieses Problem durch die sog. MILNE-*Integralgleichung* beschrieben wird,

$$E(\tau) = \frac{1}{2}\int_0^\infty E(t)E_1\left|t-\tau\right|\mathrm{d}t,$$

wobei τ die sog. "optische Tiefe" beschreibt. Diese für den Photoentransport relevante Größe (die die typische Tiefenskala in Sternatmosphären darstellt) hängt hauptsächlich von den Absorberdichten ab.

$E_1(x)$ ist das sog. erste Exponentialintegral,

$$E_1(x) = \int_1^\infty \frac{\mathrm{e}^{-x\cdot t}}{t}\mathrm{d}t.$$

Eine (äußerst aufwendige) analytische Lösung ergibt schließlich, dass der räumliche Verlauf der Strahlungsenergie (normiert auf die außen vorhandene Energie) sich folgendermaßen beschreiben lässt:

$$\frac{E(\tau)}{E(0)} = \sqrt{3}\left(\tau + q(\tau)\right).$$

$q(\tau)$ ist dabei die sog. "HOPF-Funktion", mit $\frac{1}{\sqrt{3}} \le q(\tau) < 0.710446$.

Eine adäquate Löung durch eine Monte-Carlo-Simulation sieht dann folgendermaßen aus (vgl. Skizze 8.11)

- Man "erzeuge" Photonen, die die tiefsten Schichten verlassen.
- Die Wahrscheinlichkeit, dass ein bestimmter Austrittswinkel auftritt, ist dabei durch
$$p(\mu)\mathrm{d}\mu \sim \mu\mathrm{d}\mu, \qquad \mu = \cos\theta$$
gegeben
- Die Fluglänge bis zur nächsten Streuung lässt sich durch die Wahrscheinlichkeit
$$p(\tau)\mathrm{d}\tau \sim \mathrm{e}^{-\tau}\mathrm{d}\tau$$
beschreiben, und der
- Streuwinkel (bei niedrigen Energien) ist in guter Näherung (die auch in die analytische Lösung eingeht) isotrop:
$$p(\mu)\mathrm{d}\mu \sim \mathrm{d}\mu$$
- Man verfolgt das Photon solange, bis es die Atmosphäre verlassen hat, und summiert die einzelnen Energien jeweils als Funktion der Tiefe τ auf.
- Führt man diese Simulation für eine sehr große Anzahl von Photonen durch, erzielt man schließlich das gewünschte Ergebnis, vgl. Seite 8-31.

Aus diesem Beispiel wird die schon oben angesprochene Problematik klar: Einerseits können wir bislang zwar gleichförmige Verteilungen auf dem Intervall $[0,1)$ (bzw, $(0,1)$) durch einen Zufallszahlengenerator erzeugen,

$$p(x) = \begin{cases} 1 & \text{für } x \in [0,1) \\ 0 & \text{sonst.} \end{cases}$$

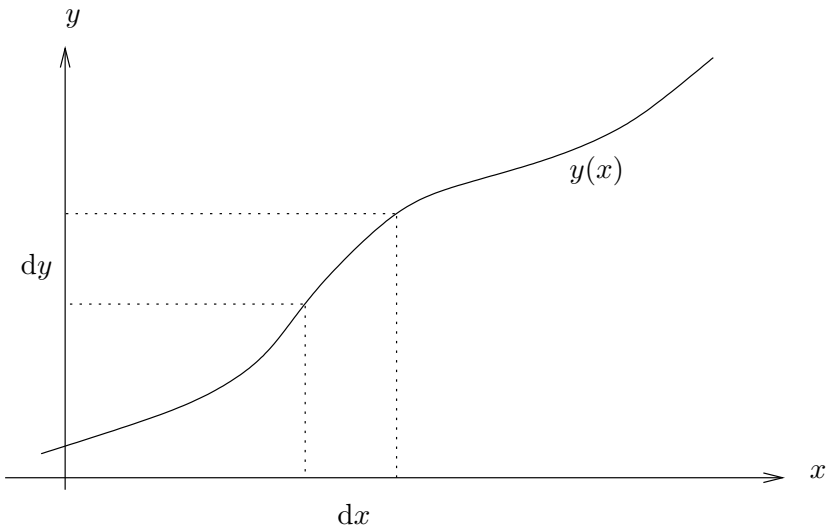


Abbildung 8.12: Zum Transformationsverfahren, siehe Text. $f(y)dy$ ist die Wahrscheinlichkeit, dass y in dy liegt, und $p(x)dx$ die Wahrscheinlichkeit, dass x in dx liegt.

Es stellt sich dann allerdings die Frage, wie wir daraus Zufallsvariablen y erzeugen, die mit einer pdf $f(y)$ verteilt sind. Wie erzeugen wir z.B. eine optische Weglänge τ in Beispiel 8.20, die entsprechend einer pdf $\mathrm{e}^{-\tau}\mathrm{d}\tau$ verteilt ist, d.h. exponentiell?

Das Inversionsverfahren (auch "Transformationsverfahren genannt").
Seien $p(x)$ und $f(y)$ verschiedene Wahrscheinlichkeitsdichteverteilungen (pdf's),

$$p(x)\mathrm{d}x = P(x \le x' \le x + \mathrm{d}x)$$
$$f(y)\mathrm{d}y = P(y \le y' \le y + \mathrm{d}y)$$

mit $y = y(x)$. Die Zufallsvariable y soll also eine Funktion der Zufallsvariablen x sein; eine physikalische Transformation bedeutet nun, dass die Wahrscheinlichkeiten *gleich sein müssen*: Wenn $p(x)\mathrm{d}x$ die Wahrscheinlichkeit ist, dass x im Bereich $x, x + \mathrm{d}x$ liegt und y eine Funktion von x ist, dann muß die Wahrscheinlichkeit dafür, dass y im Bereich $y, y + \mathrm{d}y$ liegt, die gleiche sein!

8.21 Beispiel. Sei x in $[0, 1]$ gleichförmig verteilt und $y = 2x$. Die Wahrscheinlichkeit, dass x in $0.1 \ldots 0.2$ liegt, muß gleich der Wahrscheinlichkeit sein, dass y in $0.2 \ldots 0.4$ liegt!

Ebenso muss die Gesamtwahrscheinlichkeit, dass x in $[0, 1]$ liegt ($= 1$), der Gesamtwahrscheinlichkeit entsprechen, dass $y \in [0, 2]$ (ebenfalls $= 1$).

Demzufolge gilt also

$$|p(x)\mathrm{d}x| = |f(y)\mathrm{d}y| \tag{8.37a}$$

bzw.

$$f(y) = p(x)\left|\frac{\mathrm{d}x}{\mathrm{d}y}\right|$$

Die Betragsfunktion muss deshalb verwendet werden, da $y(x)$ monoton wachsend oder fallend sein kann, die Wahrscheinlichkeiten aber immer positiv sein müssen. Die mehrdimensionale Verallgemeinerung dieses Sachverhalts erfolgt über die JACOBI-Determinante

$$|p(x_1, x_2)\mathrm{d}x_1\mathrm{d}x_2| = |f(y_1, y_2)\mathrm{d}y_1\mathrm{d}y_2|\,,$$

d.h.

$$f(y_1, y_2) = p(x_1, x_2)\left|\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)}\right| \tag{8.37b}$$

Sei nun $p(x)$ gleichförmig auf $[0,1]$ verteilt, z.B. werde x von einem Zufallszahlengenerator erzeugt $\Rightarrow p(x) = 1$, d.h.

$$\mathrm{d}x = |f(y)\mathrm{d}y| \quad \rightarrow \quad \int_0^x \mathrm{d}x' = \left| \int_{y_{\min}}^{y(x)} f(y)\mathrm{d}y \right|$$

$$\Rightarrow \qquad x = F(y) = \Bigg| \underbrace{\int_{y_{\min}}^{y} f(y')\mathrm{d}y'}_{\substack{\text{kumulative Wahr-}\\ \text{scheinlichkeitsver-}\\ \text{teilung}}} \Bigg| \qquad \text{mit} \qquad F(y_{\min}) = 0,\ F(y_{\max} = y(1)) = 1.$$

Insgesamt gilt also, dass

$$y = F^{-1}(x), \qquad \text{wenn } x \text{ gleichförmig verteilt ist.} \tag{8.38}$$

Das Inversionsverfahren erfordert demzufolge, dass

- $F(y)$ analytisch berechenbar und
- invertierbar ist.

Zur Erzeugung einer Stichprobe, die entsprechend einer pdf $f(y)$ verteilt sein soll, geht man folgendermaßen vor:

Schritt 1: Man erzeugt zunächst eine Stichprobe mit Hilfe eines Zufallszahlengenerators, $x \in [0,1)$,

Schritt 2: setzt dann $F(y) =: x$ und

Schritt 3: löst für $y = F^{-1}(x)$

8.22 Beispiel (1). Man ziehe y aus einer Verteilung, die gleichförmig auf $[a,b]$ verteilt ist (vgl. Beispiel 8.18 und Seite 8-25)

$$f(y) = \frac{1}{b-a} \quad \Rightarrow \quad F(y) = \int_a^y \frac{1}{b-a}\mathrm{d}y = \frac{y-a}{b-a}$$

(Test: $F(a) = 0$, $F(b) = 1$, ok). Damit gilt $\dfrac{y-a}{b-a} = x$, und y wird entsprechend der Vorschrift

$$y = a + (b-a)\cdot x, \qquad x \in [0,1) \tag{8.39}$$

gezogen.

8.23 Beispiel (2). Man ziehe y aus der exponentiellen Verteilung, $f(y) = \mathrm{e}^{-y}$, $y \geq 0$.

$$F(y) = \int_0^y \mathrm{e}^{-y'}\mathrm{d}y' = 1 - \mathrm{e}^{-y} =: x$$

$$\Rightarrow \qquad y = -\ln(1-x) \quad \rightarrow \quad y = -\ln x, \qquad \text{da } (1-x) \text{ wie } x \text{ verteilt ist!}$$

Berechnet man also

$$y = -\ln x, \qquad x \in [0,1) \tag{8.40}$$

so ist y exponentiell verteilt!
Test: $f(y)\mathrm{d}y = \mathrm{e}^{-(-\ln(1-x))}\mathrm{d}y(x)$ und $\mathrm{d}y = \dfrac{1}{1-x}\mathrm{d}x \Rightarrow f(y)\mathrm{d}y = \dfrac{1-x}{1-x}\mathrm{d}x \stackrel{!}{=} p(x)\mathrm{d}x$.

8.24 Beispiel (3). Man ziehe y aus einer *normalverteilten* Stichprobe.

$$f(y)\mathrm{d}y = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-y^2/2}$$

(hier: Mittelwert 0, Standardabweichung = 1)

Sei nun x_1, x_2 gleichförmig auf $(0,1)$ verteilt; betrachte

$$y_1 = \sqrt{-2\ln x_1}\cos(2\pi x_2) \tag{8.41a}$$
$$y_2 = \sqrt{-2\ln x_1}\sin(2\pi x_2) \tag{8.41b}$$

d.h.

$$y_1^2 + y_2^2 = -2\ln x_1 \qquad x_1 = \exp\left(-\frac{1}{2}(y_1^2+y_2^2)\right)$$
$$\frac{y_2}{y_1} = \tan(2\pi x_2) \qquad 2\pi x_2 = \arctan\left(\frac{y_2}{y_1}\right)$$

Mit der Transformation (8.37b) und $p(x_1,x_2)=1$ folgt daraus

$$f(y_1,y_2) = \left|\frac{\partial(x_1,x_2)}{\partial(y_1,y_2)}\right| = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = \left| -\frac{1}{\sqrt{2}\pi}\mathrm{e}^{-y_1^2/2}\cdot\frac{1}{\sqrt{2}\pi}\mathrm{e}^{-y_2^2/2}\right|.$$

$f(y_1,y_2)$ ist also das Produkt von Funktionen, die nur von y_1 bzw. y_2 abhängen und die jeweils normalverteilt sind,

$$f(y_1,y_2) = f(y_1)\cdot f(y_2).$$

Also: Ziehe zwei Stichproben x_1, x_2 aus Zufallszahlengenerator, dann sind y_1 und y_2 entsprechend (8.41) normalverteilt!

Wenn das Inversionsverfahren nicht angewendet werden kann (d.h. wenn $\int f(y)\mathrm{d}y$ nicht analytisch berechenbar oder $F(y)$ nicht invertierbar ist), verwendet man zur Erzeugung von Stichproben die VON NEUMANNsche Verwerfungsmethode (s. Anhang 8B).

8.25 Beispiel (Fortsetzung von Bsp. 8.20). Nachdem wir nun die entsprechenden Methoden zur Erzeugung von Stichproben, die gemäß einer vorgegebenen pdf verteilt sind, kennengelernt haben, kommen wir auf unser Ausgangsbeispiel zum Strahlungstransport in Atmosphären (nur Streuung an freien Elektronen) zurück. Im Folgenden geben wir das entsprechende Programm (in FORTRAN90) und das dazugehörige Ergebnis an. Wie ersichtlich, ist bei Verwendung von 10^6 Photonen die MC-Lösung von der analytischen nicht zu unterscheiden.

```
module my_type
INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)       !integer*4
INTEGER, PARAMETER :: SP = SELECTED_REAL_KIND(6,37)    !real*4
end module my_type

program milne

!solves milne's equation by Monte Carlo

use my_type
implicit none

integer(i4b), dimension(2) :: seed_f90=(/5,2147483398/)
real(sp) :: taumax
real(sp) :: x, mu, tau, dtaur,e_in,e_out,u
integer(i4b) :: n,i

print*,' Give in n,tau'
read*,n,taumax

! limb darkening coefficient
u=3./(5.+3.*taumax)

e_in=0.
e_out=0.

call random_seed(put=seed_f90)              !Initialisierung Zufallszahlengenerator

do i=1,n                                    !Schleife über alle Photonen
```

```
! start angle
call start_angle(mu,u)                      !Austrittswinkel, mu approx sqrt(x)

tau=taumax

e_in=e_in+1./abs(mu)                        !update E(tau)

10 call random_number(x)
! next path length                          !nächste Weglänge (p(tau) = exp(-tau)
dtaur=-alog(x)

! radial projection
dtaur=dtaur*mu

!radial tau
tau=tau-dtaur                               !radiales tau

if(tau .lt. 0.) then
! photon has left atmosphere                !update E(0)
 e_out=e_out+1./abs(mu)

else if(tau .gt. taumax) then
! photon back-scattered into core           !update E(tau_max)
 e_in=e_in+1./abs(mu)

else
! new scattering angle, isotropic scattering
5 call random_number(x)
mu=-1.+2*x                                  !neuer Streuwinkel (p(mu) = 1)
if (mu.eq.0.) goto 5                        !vermeide mu=0

goto 10

endif

enddo                                       !FERTIG
print*,'n =',n,'e_in/e_out =',e_in/e_out

end

subroutine start_angle(mu,u)
! calculates start_angle mu from linear limb darkening coefficient u, &
! using von Neuman (siehe Ergänzung 8B)
! original pdf:   6/(3-u) (1-u + u*mu) mu, for boundaries 0...mu
! comparison pdf  6/(3-u) mu

use my_type
implicit none

real(sp) :: a,u,x,xp,hxp,fxp,yp,mu

a=6./(3.-u)
i=0
10 call random_number(x)
xp=sqrt(x) ! Normalization constant cancels out
hxp=a*xp
fxp=a*(1.-u+u*xp)*xp
call random_number(x)
yp=hxp*x
if(yp.gt.fxp) goto 10
mu=xp
return
end
```

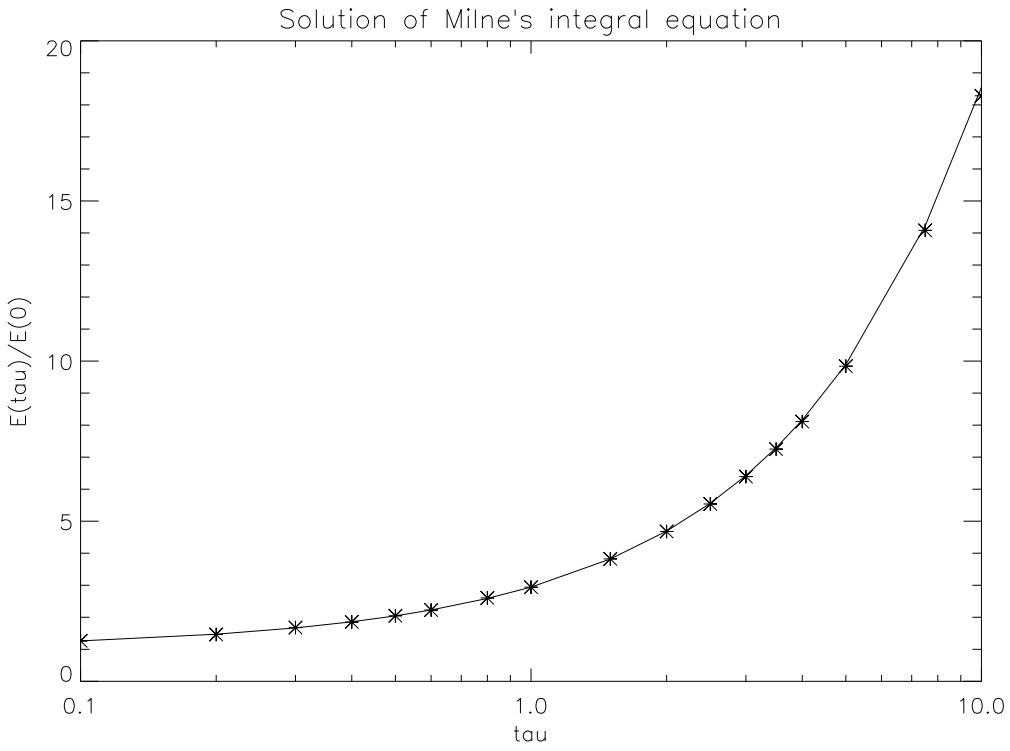


Abbildung 8.13: Lösung von MILNES Integralgleichung (Bsp. 8.20) mit Hilfe einer Monte-Carlo-Simulation (10^6 Photonen). Die MC-Lösung ist mit "Sternen" gekennzeichnet, die analytische Lösung ist die durchgezogene Line

.

8.3 Ergänzungen

sind bisher nur handschriftlich verfügbar.

8.3.1 8A: Zusammengesetzte Ereignisse

8.3.2 8B: VON NEUMANNsche Verwerfungsmethode

8.3.3 8C: Varianzreduktion mit *importance sampling*

8.3.4 8D: Quasi-Zufallszahlen

Kapitel 9

Gewöhnliche Differentialgleichungen

Zahlreiche Problemstellungen der Physik führen auf gewöhnliche Differentialgleichungen (d.h. Differentialgleichungen nur einer unabhängigen Variablen), welche oftmals die Änderung einer Größe mit der Zeit beschreiben (z.B. NEWTON'sche Bewegungsgleichung). Nur in wenigen Fällen kann die Lösung in geschlossenen Form mit bekannten Funktionen angegeben werden. Deshalb benötigt man numerische Verfahren, welche eine genaue Näherung der Lösungsfunktion liefern. Bei der Auswahl eines Verfahrens sind die Eigenschaften der Differentialgleichung (und deren Lösungsfunktionen) zu berücksichtigen. Sonst können Instabilitäten auftreten. Wir beschränken uns hier auf die sogenannten Anfangswertprobleme.

Wir betrachten die skalare gewöhnliche Differentialgleichung erster Ordnung

$$y' = f(x, y) \tag{9.1}$$

für die gesuchte Lösungsfunktion $y(x)$ mit der Anfangsbedingung

$$y(x_0) = y_0 \tag{9.2}$$

zu vorgegebenen x_0 und y_0 (*Anfangswertproblem*).

Skalare Differentialgleichungen höherer Ordnung können auf vektorielle Differentialgleichungen (Systeme von Differentialgleichungen) erster Ordnung zurückgeführt werden. Letztere können wie skalare Differentialgleichungen behandelt werden.

9.1 Beispiel. $y'' = f(x, y, y')$ mit $y(x_0) = y_0$, $y'(x_0) = y_0'$.
Sei

$$\boldsymbol{y} := \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad \text{und} \quad \boldsymbol{f}(x, \boldsymbol{y}) := \begin{pmatrix} y_2 \\ f(x, y_1, y_2) \end{pmatrix}.$$

Dann ist

$$\boldsymbol{y}' = f(x, \boldsymbol{y}) \qquad \text{mit} \qquad \boldsymbol{y}(x_0) = \begin{pmatrix} y_0 \\ y_0' \end{pmatrix}$$

zu lösen.

9.1 Existenz und Eindeutigkeit

9.2 Satz. *Sei $G \subset \mathbb{R} \times \mathbb{R}$ und $f : G \to \mathbb{R}$. Man sagt, daß f einer* LIPSCHITZ-Bedingung *mit* LIPSCHITZ-Konstante *$L \geq 0$ genügt, falls*

$$|f(x, y) - f(x, \tilde{y})| \leq L\,|y - \tilde{y}| \qquad \textit{für alle } (x, y), (x, \tilde{y}) \in G \textit{ gilt.} \tag{9.3}$$

Kriterium. Sei $f \in \mathcal{C}^1(G)$ mit $G \subset \mathbb{R} \times \mathbb{R}$ kompakt und konvex. Dann genügt f einer LIPSCHITZ-Bedingung mit LIPSCHITZ-Konstante

$$L = \max_{(x,y) \in G} |\partial_y f(x, y)|$$

(unmittelbare Folge des Mittelwertsatzes der Differentialrechnung).

9.3 Satz (PICARD-LINDELÖF). *Seien* $\alpha, \beta > 0$, $(x_0, y_0) \in \mathbb{R}^2$ *und*

$$R := \{(x,y) | \, |x - x_0| \le \alpha, |y - y_0| \le \beta\}.$$

Sei ferner $f \in \mathcal{C}^0(R)$ *eine Funktion mit* $0 < \gamma := \max|f| < \infty$, *die einer* LIPSCHITZ-*Bedingung genügt.*

Dann gibt es genau eine Funktion $y \in \mathcal{C}^1(I)$ *in* $I := [x_0 - \delta, x_0 + \delta]$, $\delta := \min\left\{\alpha, \frac{\beta}{\gamma}\right\}$ *mit* $y'(x) = f(x, y(x)) \; \forall \; x \in I$ *und* $y(x_0) = y_0$.

Zum Beweis siehe ein Lehrbuch zu gewöhnlichen Differentialgleichungen.

Das Anfangswertproblem $y'(x) = f(x,y)$ mit $y(x_0) = y_0$ hat auch dann noch eine Lösung, wenn nur die Stetigkeit von f vorausgesetzt wird. Die Eindeutigkeit ist dann aber nicht mehr gesichert.

9.4 Beispiel. $f(x,y) = y^{2/3}$, $(x,y) \in \mathbb{R} \times \mathbb{R}$, ist stetig, genügt aber keiner LIPSCHITZ-Bedingung. Denn $|y^{2/3} - 0| \le L|y - 0|$, also $1/|y|^{1/3} \le L$ ist für kein $L \ge 0$ erfüllbar für alle y um 0.

Offensichtlich sind $y_1(x) = 0$ $(x \in \mathbb{R})$ und $y_2(x) = \frac{1}{27}(x - x_0)^3$ $(x \in \mathbb{R})$ zwei verschiedene Lösungen des Anfangswertproblems $y' = f(x,y)$, $y(x_0) = 0$ mit $x_0 \in \mathbb{R}$.

Im Folgenden sei

$$f : [a,b] \times J \to \mathbb{R} \qquad \text{mit einem Intervall } J \tag{9.4}$$

und das Anfangswertproblem laute

$$y' = f(x,y), \qquad y(x_0) = y_0 \qquad \text{mit } x_0 = a,\ y_0 \in J. \tag{9.5}$$

Wir nehmen an: Es existiert genau eine Lösung des Anfangswertproblems mit Definitionsbereich $[a,b]$.

9.2 Konsistenz und Konvergenz

Bei der numerischen Lösung des Anfangswertproblems werden an gewissen Abszissenwerten

$$a =: x_0 < x_1 < \ldots < x_N := b \tag{9.6}$$

mit den Schrittweiten

$$h_n := x_{n+1} - x_n \qquad (n = 0, 1, \ldots, N-1) \tag{9.7}$$

Näherungswerte

$$y_n \approx y(x_n) \tag{9.8}$$

für die Werte der gesuchten Lösung konstruiert. Manchmal wird eine äquidistante Schrittweite

$$h := \frac{b-a}{N} \tag{9.9}$$

gewählt.

Einschrittverfahren. Bei einem *Einschrittverfahren* wird der Näherungswert y_{n+1} an der Stelle x_{n+1} allein aufgrund des Näherungspunktes (x_n, y_n) bestimmt (s. Abschnitt 5 zu Mehrschrittverfahren). Die zugehörige Verfahrensvorschrift hat die allgemeine Form

$$y_{n+1} = y_n + h_n \varphi(x_n, y_n, y_{n+1}, h_n) \tag{9.10}$$

mit einer Verfahrensfunktion φ. Hängt φ tatsächlich nicht von y_{n+1} ab, so spricht man von einem *expliziten Einschrittverfahren*, andernfalls von einem *impliziten Einschrittverfahren*. Im letzten Fall muß eine implizite Gleichung für y_{n+1} numerisch gelöst werden. Dies ist ein Nachteil der impliziten Verfahren; ihr Vorteil liegt in der Stabilität (s. Abschnitt 6).

In der Praxis ist der Fehler wichtig, den die Näherung nach einer bestimten Anzahl von Schritten gegenüber der exakten Lösung aufweist. Dabei lassen wir Rundungsfehler zunächst außer Betracht. Einfachheitshalber gehen wir von einer äquidistanten Schrittweite aus.

Globaler Diskretisierungsfehler. Als *globalen Diskretisierungsfehler* an der Stelle x_n bezeichnet man die Differenz

$$g_n := y(x_n) - y_n \qquad (n = 0, 1, \ldots, N). \tag{9.11}$$

Das Einschrittverfahren heißt *konvergent*, falls

$$\max_{n=0,1,\ldots,N} |g_n| \to 0 \qquad \text{für } h \to 0^+ \tag{9.12}$$

gilt. Es besitzt die *Konvergenzordnung* $p > 0$, falls sich der globale Diskretisierungsfehler in der Form

$$\max_{n=0,1,\ldots,N} |g_n| \le ch^p = \mathcal{O}(h^p) \tag{9.13}$$

mit von h unabhängigen Konstanten $c \ge 0$ und $p > 0$ schreiben läßt.

Für die Abschätzung des globalen Diskretisierungsfehlers spielt der folgende Begriff eine wichtige Rolle:

Lokaler Diskretisierungsfehler. Unter dem lokalen Diskretisierungsfehler an der Stelle x_{n+1} versteht man den Wert

$$l_{n+1} := y(x_{n+1}) - y(x_n) - h\varphi(x_n, y(x_n), y(x_{n+1}), h). \tag{9.14}$$

Er stellt die Abweichung dar, um welche die exakte Lösung die Verfahrensvorschrift nicht erfüllt. Im Falle eines expliziten Verfahrens ist l_{n+1} die Differenz zwischen dem exakten Wert $y(x_{n+1})$ und dem Näherungswert y_{n+1}, falls an der Stelle x_n vom exakten Wert $y(x_n)$ ausgegangen wird (Fehler in einem Schritt; s. Abbildung 9.1).

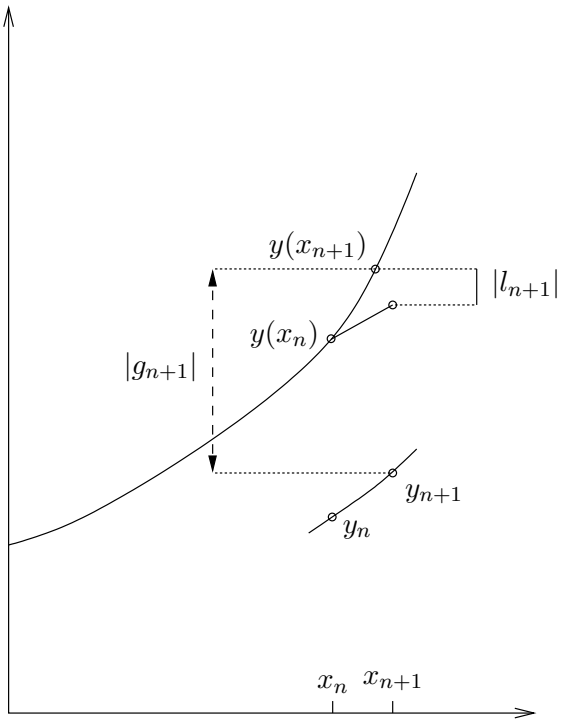


Abbildung 9.1: Lokaler und globaler Diskretisierungsfehler

Das Einschrittverfahren heißt *konsistent*, falls

$$\frac{1}{h} l_{n+1} \to 0 \quad \text{für} \quad h \to 0^+ \qquad (n = 0, 1, \ldots, N-1) \tag{9.15}$$

gilt. Wegen

$$\frac{l_{n+1}}{h} = \underbrace{\frac{y(x_{n+1}) - y(x_n)}{h}}_{\text{Sekantensteigung der exakten Lösung}} - \underbrace{\varphi(x_n, y(x_n), y(x_{n+1}), h)}_{\text{Näherung dieser Steigung}} \tag{9.16}$$

ist

$$\frac{1}{h}|l_{n+1}| \xrightarrow{h\to 0^+} 0 \qquad \Leftrightarrow \qquad \varphi(x_n, y(x_n), y(x_{n+1}), h) \xrightarrow{h\to 0^+} f(x_n, y(x_n)). \tag{9.17}$$

Das Verfahren besitzt die *Konsistenzordnung* (oder kurz: Ordnung) p, falls für den lokalen Diskretisierungsfehler die Ungleichung

$$|l_{n+1}| \le ch^{p+1} = \mathcal{O}(h^{p+1}) \tag{9.18}$$

mit von h unabhängigen Konstanten $c \ge 0$ und $p > 0$ erfüllt ist.

Um den globalen Diskretisierungsfehler mit dem lokalen Diskretisierungsfehler in Beziehung zu bringen, setzen wir die LIPSCHITZ-Bedingung

$$|\varphi(x, y_1, y_2, h) - \varphi(x, \tilde{y}_1, y_2, h)| \le L|y_1 - \tilde{y}_1|\,, \tag{9.19a}$$
$$|\varphi(x, y_1, y_2, h) - \varphi(x, y_1, \tilde{y}_2, h)| \le L|y_2 - \tilde{y}_2| \tag{9.19b}$$

an die Verfahrensfunktion $\varphi(x, y_1, y_2, h)$ mit einer LIPSCHITZ-Konstanten $L > 0$ voraus. Im Falle eines expliziten Verfahrens entfällt der zweite Teil.

Nach der Definition (9.14) des lokalen Diskretisierungsfehlers ist

$$y(x_{n+1}) = y_n(x_n) + h\varphi(x_n, y(x_n), y(x_{n+1}), h) + l_{n+1}. \tag{9.20}$$

Die Subtraktion der Verfahrensvorschrift (9.10) von dieser Gleichung ergibt für den globalen Diskretisierungsfehler

$$g_{n+1} = g_n + h\Big(\varphi\big(x_n, y(x_n), y(x_{n+1}), h\big) - \varphi\big(x_n, y_n, y(x_{n+1}), h\big) + \varphi\big(x_n, y_n, y(x_{n+1}), h\big) - \varphi\big(x_n, y_n, y_{n+1}, h\big)\Big) + l_{n+1}. \tag{9.21}$$

Wegen der LIPSCHITZ-Bedingung der Verfahrensfunktion gemäß (9.19a), (9.19b) gilt dann

$$|g_{n+1}| \le (1 + Lh)\,|g_n| + |l_{n+1}| \tag{9.22}$$

bei einem expliziten Verfahren bzw.

$$|g_{n+1}| \le \left(1 + \frac{2Lh}{1 - Lh}\right)|g_n| + \frac{1}{1 - Lh}|l_{n+1}| \quad \text{für } Lh < 1 \tag{9.23}$$

bei einem impliziten Verfahren.

Den lokalen Diskretisierungsfehler schätzen wir durch

$$|l_{n+1}| \le l \quad (n = 0, 1, \ldots, N-1) \qquad \text{mit } l > 0 \tag{9.24}$$

ab. Im expliziten und impliziten Fall erfüllt der globale Diskretisierungsfehler eine Ungleichung der Form

$$|g_{n+1}| \le (1 + a)|g_n| + b \qquad (n = 0, 1, \ldots, N-1) \tag{9.25}$$

mit jeweils entsprechender Festsetzung der Konstanten $a > 0$ und $b \geq 0$. Eine wiederholte Anwendung von (9.25) ergibt

$$\begin{aligned}
|g_{n+1}| &\leq (1+a)|g_n| + b \\
&\leq (1+a)^2|g_{n-1}| + ((1+a)+1)\,b \\
&\vdots \\
&\leq (1+a)^{n+1}|g_0| + \underbrace{\left((1+a)^n + (1+a)^{n-1} + \cdots + (1+a) + 1\right)}_{=((1+a)^{n+1}-1)/a} b \\
&\leq \frac{b}{a}\left(\mathrm{e}^{(n+1)a} - 1\right) + \mathrm{e}^{(n+1)a}|g_0| \qquad (9.26)
\end{aligned}$$

(Im letzten Schritt wurde $1 + a < e^a$ benutzt.). Wegen $g_0 = 0$ folgt daraus

$$|g_{n+1}| \leq \frac{l}{Lh}\left(\mathrm{e}^{(n+1)Lh} - 1\right) \qquad (9.27)$$

für ein explizites Einzelschrittverfahren bzw.

$$|g_{n+1}| \leq \frac{l}{2Lh}\left(\mathrm{e}^{2(n+1)Lh/(1-Lh)} - 1\right) \qquad (9.28)$$

für ein implizites Einzelschrittverfahren.

Für ein Verfahren mit der Konsistenzordnung p können wir

$$l \leq ch^{p+1} \qquad (9.29)$$

mit $c \geq 0$ ansetzen. Damit ist

$$|g_{n+1}| \leq \frac{c}{L}\left(\mathrm{e}^{(n+1)Lh} - 1\right) h^p, \qquad \text{also} \qquad (9.30)$$

$$\max_{n=0,1,\ldots,N-1} |g_{n+1}| \leq \frac{c}{L}\left(\mathrm{e}^{L(b-a)} - 1\right) h^p \qquad (9.31)$$

im expliziten Fall bzw.

$$|g_{n+1}| \leq \frac{c}{2L}\left(\mathrm{e}^{2(n+1)Lh/(1-Lh)} - 1\right) h^p, \qquad \text{also} \qquad (9.32)$$

$$\begin{aligned}
\max_{n=0,1,\ldots,N-1} |g_{n+1}| &\leq \frac{c}{2L}\left(\mathrm{e}^{2L(b-a)/(1-Lh)} - 1\right) h^p \\
&= \frac{c}{2L}\left(\mathrm{e}^{2L(b-a)-1}\right) h^p + \mathcal{O}(h^{p+1}) \qquad (9.33)
\end{aligned}$$

im impliziten Fall.

Erwartungsgemäß führen lokale Fehler von $\mathcal{O}(h^{p+1})$ nach $N = \frac{b-a}{h}$ Schritten zu einem globalen Fehler von $\mathcal{O}(Nh^{p+1}) = \mathcal{O}(h^p)$.

Rundungsfehler. Jetzt berücksichtigen wir auch die Auswirkungen von Rungsfehlern und einem fehlerbehafteten Anfangswert. Hierzu wird angenommen, daß eine fehlerbehaftete Verfahrensvorschrift von der Form

$$\tilde{y}_{n+1} = \tilde{y}_n + h\varphi(x_n, \tilde{y}_n, \tilde{y}_{n+1}, h) + \delta_{n+1} \qquad (n = 0, 1, \ldots, N-1) \qquad (9.34)$$

mit

$$\tilde{y}_0 = y_0 + \varepsilon_0 \qquad (9.35)$$

vorliegt. Sei

$$|\delta_{n+1}| \leq \delta \qquad (n = 0, 1, \ldots, N-1) \qquad (9.36)$$

und

$$\varepsilon := |\varepsilon_0|. \tag{9.37}$$

Bei der Abschätzung des (globalen) Gesamtfehlers

$$t_n := y(x_n) - \tilde{y}_n \qquad (n = 0, 1, \ldots, N-1) \tag{9.38}$$

können wir ähnlich wie beim globalen Diskretisierungsfehler vorgehen: Nach Subtraktion der fehlerbehafteten Verfahrensvorschrift (9.34) von (9.20) ergibt sich

$$t_{n+1} = t_n + h\Big(\varphi\big(x_n, y(x_n), y(x_{n+1}), h\big) - \varphi\big(x_n, \tilde{y}_n, \tilde{y}_{n+1}, h\big)\Big) + l_{n+1} - \delta_{n+1}. \tag{9.39}$$

Unter Beachtung der Lipschitz-Bedingung (9.19a), (9.19b) folgt daraus

$$|t_{n+1}| \le (1 + Lh)|t_n| + l + \delta \qquad \text{im expliziten Fall bzw.} \tag{9.40}$$

$$|t_{n+1}| \le \left(1 + \frac{2Lh}{1 - Lh}\right)|t_n| + \frac{1}{1 - Lh}(l + \delta) \qquad \text{im impliziten Fall.} \tag{9.41}$$

Die Hilfsformel (9.26) für t_n anstelle von g_n liefert dann

$$|t_{n+1}| \le \frac{l + \delta}{Lh}\left(\mathrm{e}^{(n+1)Lh} - 1\right) + \mathrm{e}^{(n+1)Lh}\varepsilon \tag{9.42}$$

bei einem expliziten Verfahren bzw.

$$|t_{n+1}| \le \frac{l + \delta}{Lh}\left(\mathrm{e}^{2(n+1)Lh/(1-Lh)} - 1\right) + \mathrm{e}^{2(n+1)Lh/(1-Lh)}\varepsilon \tag{9.43}$$

bei einem impliziten Verfahren. Mit (9.29) ergibt sich schließlich

$$\max_{n=0,1,\ldots,N-1} |t_{n+1}| \le \mathrm{e}^{L(b-a)}\varepsilon + \frac{1}{L}\left(\mathrm{e}^{L(b-a)} - 1\right)\left(ch^p + \frac{\delta}{h}\right) \tag{9.44}$$

im expliziten Fall bzw.

$$\max_{n=0,1,\ldots,N-1} |t_{n+1}| \le \mathrm{e}^{L(b-a)}\varepsilon + \frac{1}{2L}\left(\mathrm{e}^{2L(b-a)/(1-Lh)} - 1\right)\left(ch^p + \frac{\delta}{h}\right) \tag{9.45}$$

im impliziten Fall.

Speziell gilt nach (9.44) für ein explizite Einschrittverfahren ohne fehlerbehafteten Anfangswert

$$\max_{n=0,1,\ldots,N-1} |t_{n+1}| \le \frac{1}{L}\left(\mathrm{e}^{L(b-a)} - 1\right)\left(ch^p + \frac{\delta}{h}\right) \tag{9.46}$$

Die optimale Schrittweite $h_{\text{opt.}}$ ist durch das Minimum des Gesamtfehlers (vgl. Abbildung 9.2) gegeben. Das Minimum der rechten Seite von (9.46) liegt bei $\left(\frac{\delta}{cp}\right)^{1/(1+p)}$ und hat den Wert $c(1 + p)\left(\frac{\delta}{cp}\right)^{p/(1+p)}$.

9.3 Einschrittverfahren

Geht der Graph einer Lösung $y(x)$ der Differentialgleichung $y' = f(x, y)$ durch einen Punkt $(\tilde{x}, \tilde{y})$, so hat der Graph in diesem Punkt die Steigung $f(\tilde{x}, \tilde{y})$. Der Wert $f(\tilde{x}, \tilde{y})$ legt die Steigung der Tangente der Lösungskurve im Punkt $(\tilde{x}, \tilde{y})$ fest. Trägt man in der xy-Ebene für viele Punkte die durch die Differentialgleichung gegebenen Steigungen ein, so kann man den Graphen der Lösung des Anfangswertproblems $y' = f(x, y)$, $y(x_0) = y_0$ grob skizzieren, in dem man sich beim Zeichnen ausgehend vom Anfangspunkt (x_0, y_0) durch das Richtungsfeld leiten läßt.

9.5 Beispiel. $y' = -\sin(x)y^2$ mit dem Anfangswert $y(0) = 2$.
Die exakte Lösung ist $y(x) = \dfrac{-2}{2\cos(x) - 3}$. Das Richtungsfeld und die Lösung sind in Abbildung 9.3 zu sehen.

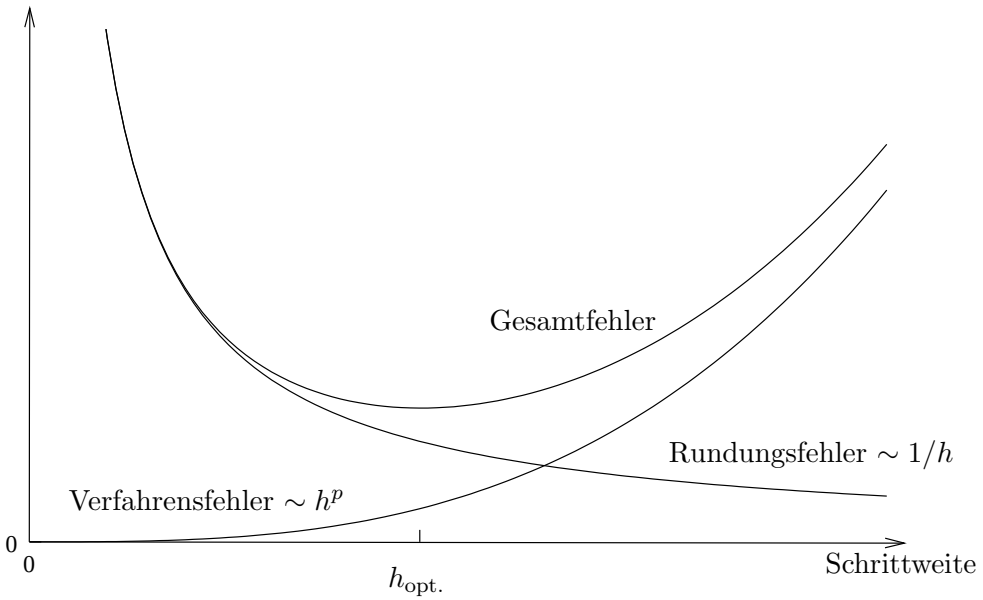


Abbildung 9.2: Gesamtfehler als Summe von Verfahrens- und Rundungsfehler

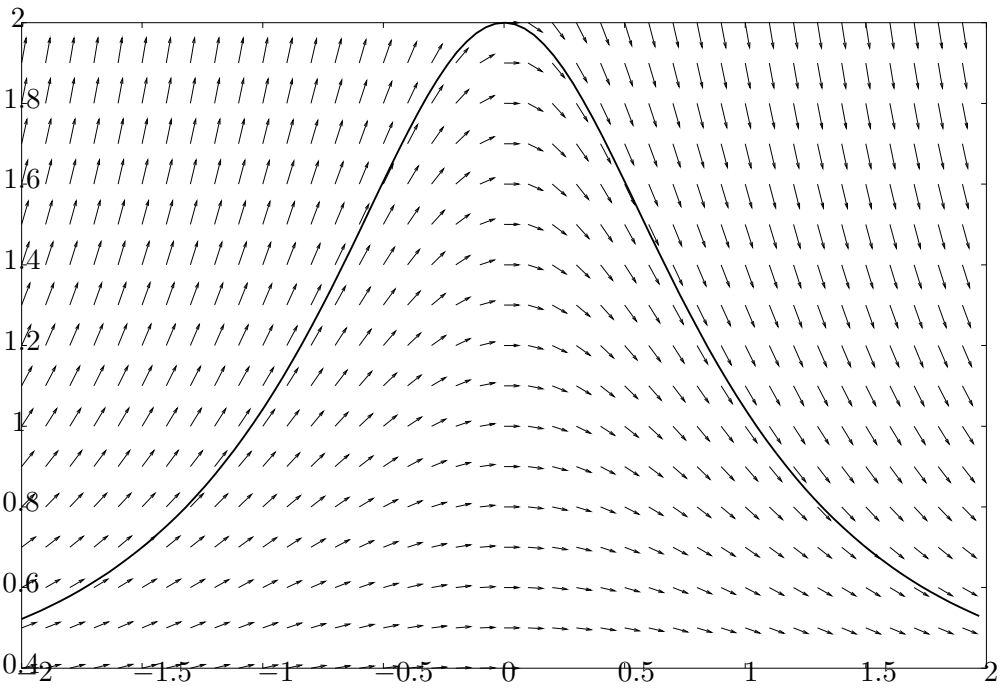


Abbildung 9.3: Das Richtungsfeld zu Beispiel 9.5

9.3.1 Verfahren von EULER

Die einfachste numerische Methode zur Lösung des Anfangswertproblemes (9.5) besteht darin, die Lösungskurve durch einen Polygonzug (= stückweise lineare Funktion) zu approximieren, der das Richtungsfeld respektiert. Für jedes n zeigt dabei das lineare Stück des Polygonzuges zwischen x_n und x_{n+1} in Richtung $f(x_n, y_n)$ (= Steigung der Tangente an eine i.A. andere Lösungskurve der Differentialgleichung durch den Punkt (x_n, y_n)).

Demnach werden bei diesem sogenannten Verfahren von EULER die Näherungswerte der Lösung nach der Vorschrift

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (n = 0, \ldots, N-1) \tag{9.47}$$

berechnet (s. Abbildung 9.4).

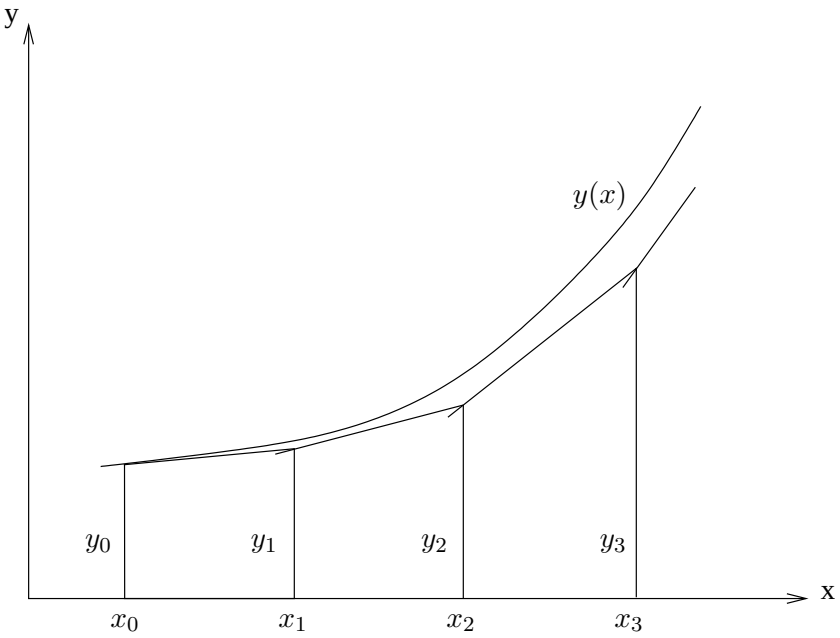


Abbildung 9.4: Das EULER-Verfahren

Für den lokalen Diskretisierungsfehler ergibt sich

$$\begin{aligned} l_{n+1} &= \underbrace{y(x_{n+1})}_{(*)} - y(x_n) - hf\big(x_n, y(x_n)\big) \\ &= \frac{1}{2}\Big(\partial_x f\big(x_n, y(x_n)\big) + f\big(x_n, y(x_n)\big)\partial_y f\big(x_n, y(x_n)\big)\Big)h^2 + \mathcal{O}(h^3), \end{aligned} \tag{9.48}$$

nach der TAYLOR-Entwicklung

$$\begin{aligned} (*) &= y(x_n) + y'(x_n)h + \frac{1}{2}y''(x_n)h^2 + \mathcal{O}(h^3) \\ &= y(x_n) + f\big(x_n, y(x_n)\big)h + \frac{1}{2}\Big(\partial_x f\big(x_n, y(x_n)\big) + f\big(x_n, y(x_n)\big)\partial_y f\big(x_n, y(x_n)\big)\Big)h^2 + \mathcal{O}(h^3), \end{aligned}$$

d.h. die Konsistenzordnung des EULER-Verfahrens ist 1.

9.6 Beispiel. $y' = -2xy^2$, $y(0) = 1$. Tabelle 9.1 enthält die Werte für verschiedene Parameterwerte des EULER-Verfahrens. In Abbildung 9.5 findet man das Ergebnis; der Gesamtfehler – er nimmt etwa proportional zur Schrittweite ab – ist in Abbildung 9.6 aufgetragen. Die exakte Lösung ist $y(x) = \dfrac{1}{1+x^2}$.

		$h = 0.1$		$h = 0.01$		h = 0.001	
x_n	$y(x_n)$	y_n	t_n	y_n	t_n	y_n	t_n
0.0	1.00000	1.00000		1.00000		1.00000	
0.1	0.99010	1.00000	-0.00990	0.99107	-0.00097	0.99020	-0.00010
0.2	0.96154	0.98000	-0.01846	0.96330	-0.00176	0.96171	-0.00018
0.3	0.91743	0.94158	-0.02415	0.91969	-0.00226	0.91766	-0.00022
0.4	0.86207	0.88839	-0.02632	0.86448	-0.00242	0.86231	-0.00024
0.5	0.80000	0.82525	-0.02525	0.80229	-0.00229	0.80023	-0.00023
0.6	0.73529	0.75715	-0.02185	0.73727	-0.00198	0.73549	-0.00020

Tabelle 9.1: Wertetabelle zum EULER-Verfahren (Beispiel 9.6)

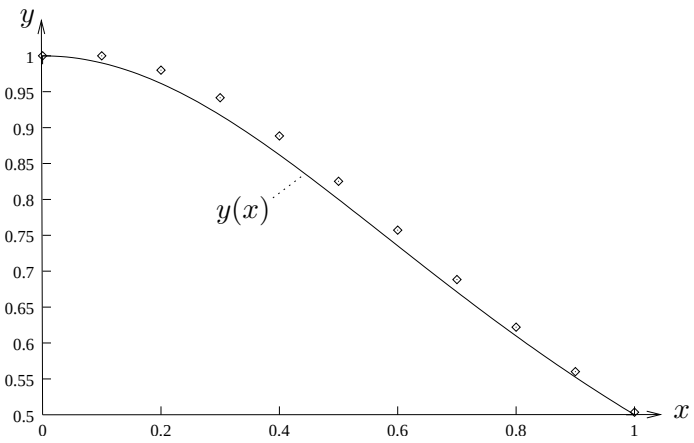


Abbildung 9.5: EULER-Verfahren (Beispiel 9.6)

9.7 Beispiel. $y' = y$, $y(0) = 1$. EULER-Verfahren mit $h = 0.1$; das Ergebnis zeigt Abbildung 9.7. Die exakte Lösung ist $y(x) = \mathrm{e}^x$.

Man sieht in Abbildung 9.7, daß der Gesamtfehler mit wachsendem n betragsmäßig immer größer wird. Die numerische Lösung entfernt sich in jedem Schritt ein Stückchen weiter von der exakten Lösung (typische Situation). Um diesen Effekt abzuschwächen, kann man eine kleinere Schrittweite wählen, also bei gleichbleibendem Lösungsintervall die Anzahl der Schritte erhöhen (globaler Disketrisierungsfehler von $\mathcal{O}(h)$). Man kann aber auch ein Verfahren höherer Ordnung einsetzen. Letzteres ist im Hinblick auf die Rundungsfehler und die Rechenzeit bei einer höheren Genauigkeitsanforderung zu tun.

9.3.2 Verfahren von HEUN

Im Allgemeinen ändert sich der Wert des Richtungsfeldes von einem zum nächsten Näherungspunkt. Im Gegensatz zum EULER-Verfahren wird dies beim Verfahren von HEUN bei der Durchführung eines einzelnen Schrittes berücktsichtigt:

$$\begin{aligned} k_1 &:= f(x_n, y_n)\,, \\ k_2 &:= f(x_n + h, y_n + hk_1)\,; \\ y_{n+1} &= y_n + \frac{h}{2}\,(k_1 + k_2) \end{aligned} \tag{9.49}$$

Es wird hier zunächst ein Näherungswert

$$\tilde{y}_{n+1} := y_n + hf(x_n, y_n) \tag{9.50}$$

nach dem EULER-Verfahren vorausgesagt, der dann nach

$$y_{n+1} = y_n + \frac{h}{2}\big(f(x_n, y_n) + f(x_{n+1}, \tilde{y}_{n+1})\big) \tag{9.51}$$

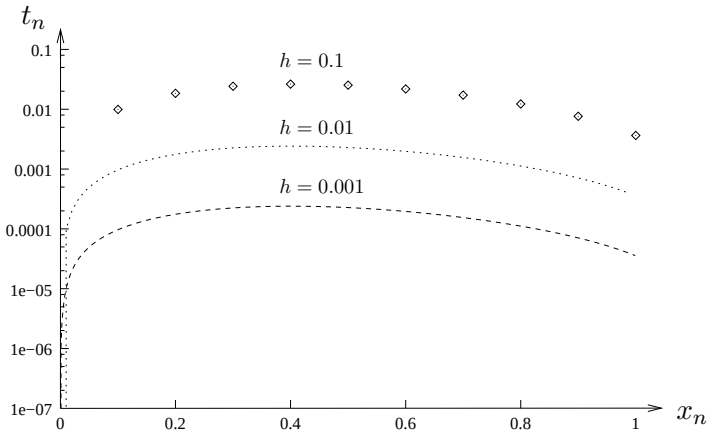


Abbildung 9.6: Fehler beim EULER-Verfahren (Beispiel 9.6)

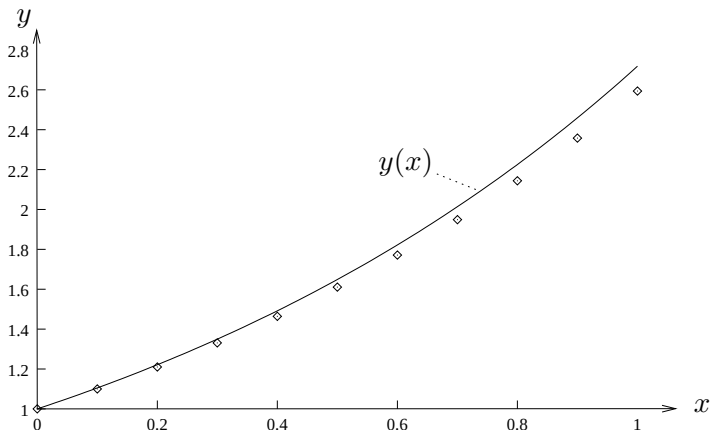


Abbildung 9.7: EULER-Verfahren (Beispiel 9.7)

nachgebessert wird. Man spricht daher von einem sogenannten *Prädikator-Korrektor-Verfahren.* Zur Bestimmung von y_{n+1} werden die beiden Steigungen an den Punkten (x_n, y_n) und $(x_{n+1}, \tilde{y}_{n+1})$ arithmetrisch gemittelt (s. Abbildung 9.8).

Dadurch wird eine Konsistenzordnung von 2 erreicht, denn der lokale Diskretisierungsfehler

$$l_{n+1} = y(x_{n+1}) - y(x_n) - \frac{h}{2}\Big(f\big(x_n, y(x_n)\big) + f\Big(x_{n+1}, y(x_n) + hf\big(x_n, y(x_n)\big)\Big)\Big) \tag{9.52}$$

ist nach einer TAYLOR-Entwicklung in der Schrittweite h gegeben durch

$$l_{n+1} = \frac{1}{6}\left(F(x_n, y(x_n))\,\partial_y f(x_n, y(x_n)) - \frac{1}{2}G(x_n, y(x_n))\right)h^3 + \mathcal{O}(h^4) \tag{9.53}$$

mit

$$F(x,y) := \partial_x f(x,y) + f(x,y)\partial_y f(x,y), \tag{9.54a}$$
$$G(x,y) := \partial_x^2 f(x,y) + 2f(x,y)\partial_x\partial_y f(x,y) + (f(x,y))^2\,\partial_y^2 f(x,y)\,. \tag{9.54b}$$

9.8 Beispiel (Fortsetzung von Beispiel 9.6, S. 9-8). Wertetabelle zum HEUN-Verfahren:

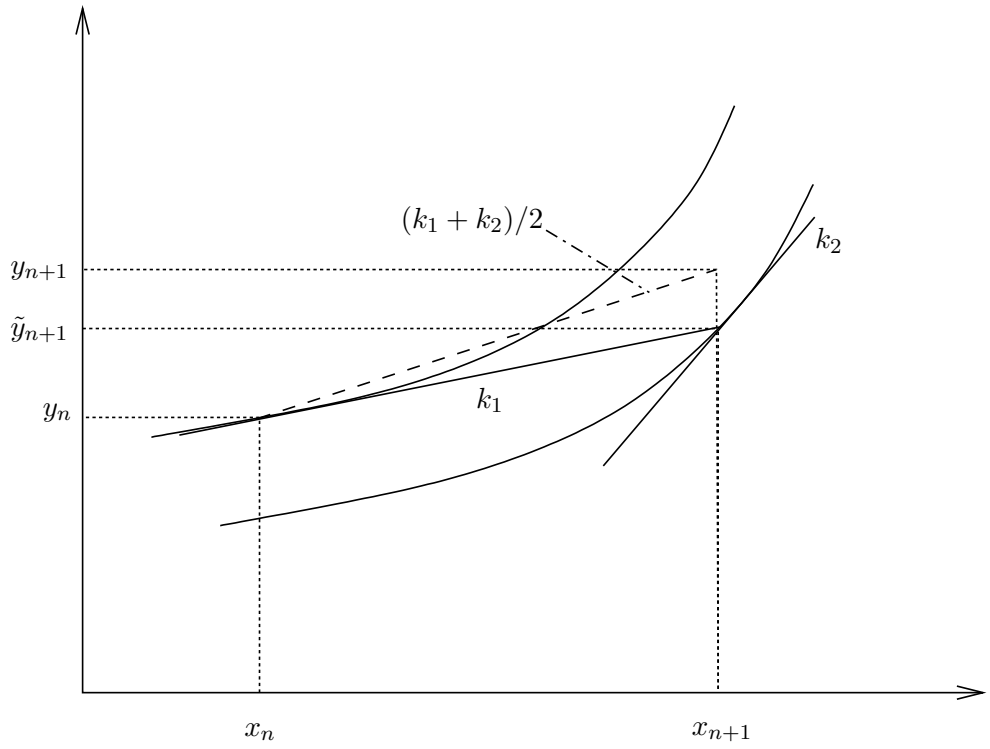


Abbildung 9.8: Verfahren von HEUN

$h = 0.1$		$h = 0.05$	
y_n	t_n	y_n	t_n
1.00000		1.00000	
0.99000	0.00010	0.99909	0.00001
0.96137	0.00017	0.96152	0.00002
0.91725	0.00019	0.91742	0.00001
0.86195	0.00011	0.86208	-0.00001
0.80003	-0.00003	0.80004	-0.00004
0.73553	-0.00023	0.73538	-0.00009
0.67159	-0.00045	0.67128	-0.00014
0.61040	-0.00064	0.60993	-0.00018
0.55329	-0.00080	0.55270	-0.00021
0.50092	-0.00092	0.50024	-0.00024

Die Ergebnisse zeigen den betragsmäßig kleineren Gesamtfehler des HEUN-Verfahrens im Vergleich zum EULER-Verfahren bei gleicher Schrittweite ($h = 0.1$).

9.3.3 Verfahren von COLLATZ

Dieses Verfahren wird auch *verbessertes* oder *modifiziertes* EULER-*Verfahren* genannt.

Aufgrund der Konsistenz des EULER-Verfahrens kann eine Extrapolation nach verschwindender Schrittweite bei jedem einzelnen Schritt durchgeführt werden. Dabei werden die Werte von einem Einzelschritt mit der Schrittweite h und einem Doppelschritt mit der halben Schrittweite verwendet.

EULER-Einzelschritt:

$$y_{n+1}^{(1)} = y_n + hf(x_n, y_n) \tag{9.55}$$

EULER-Doppelschritt:

$$y^{(2)}_{n+\frac{1}{2}} = y_n + \frac{h}{2} f(x_n, y_n) \tag{9.56a}$$

$$y^{(2)}_{n+1} = y_{n+\frac{1}{2}} + \frac{h}{2} f\left(x_{n+\frac{1}{2}}, y^{(2)}_{n+\frac{1}{2}}\right), \qquad x_{n+\frac{1}{2}} := x_n + \frac{h}{2} \tag{9.56b}$$

RICHARDSON-Extrapolation:

$$y_{n+1} = y^{(1)}_{n+1} - \frac{y^{(1)}_{n+1} - y^{(2)}_{n+1}}{h - \frac{h}{2}} h = 2y^{(2)}_{n+1} - y^{(1)}_{n+1} \tag{9.57}$$

(s. Abbildung 9.9).

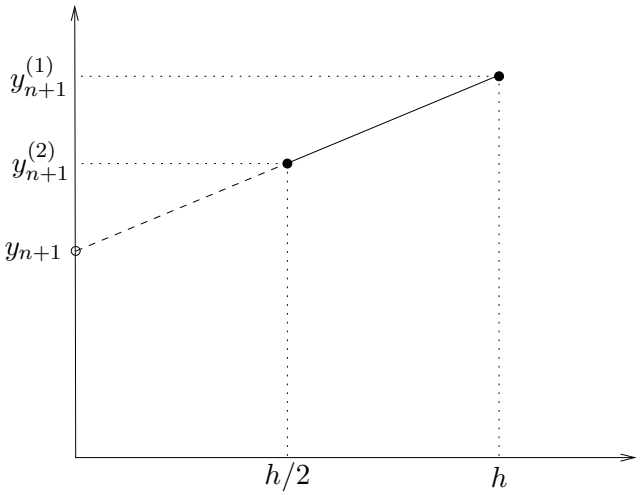


Abbildung 9.9: RICHARDSON-Extrapolation beim Verfahren von COLLATZ

Also:

$$y_{n+1} = y_n + h\, f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} f(x_n, y_n)\right) \tag{9.58}$$

COLLATZ-Verfahren (ein Beispiel für ein sogenanntes *Extrapolationsverfahren*, s. Abbildung 9.10):

$$\begin{aligned} k_1 &:= f(x_n, y_n)\,, \\ k_2 &:= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right)\,; \\ y_{n+1} &= y_n + h \cdot k_2 \end{aligned} \tag{9.59}$$

Für den lokalen Diskretisierungsfehler gilt

$$l_{n+1} = \frac{1}{6}\left(F(x_n, y(x_n))\, \partial_y f(x_n, y(x_n)) + \frac{1}{4} G(x_n, y(x_n))\right) h^3 + \mathcal{O}(h^4) \tag{9.60}$$

mit den in (9.54a), (9.54b) definierten Funktionen $F(x, y)$, $G(x, y)$, d.h. Konsistenzordnung des COLLATZ-Verfahrens ist 2 (wie beim HEUN-Verfahren).

9.9 Beispiel (Fortsetzung von Beispiel 9.6). Wertetabelle zum COLLATZ-Verfahren:

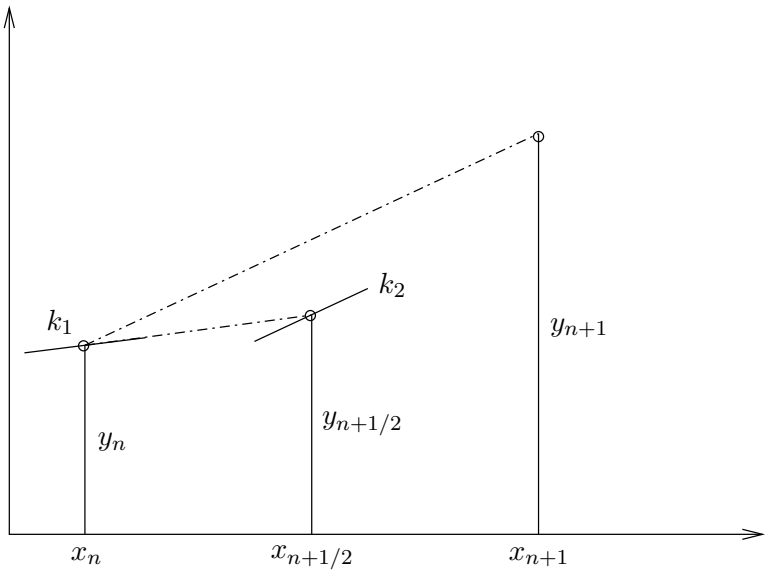


Abbildung 9.10: Verfahren von COLLATZ

	$h = 0.1$		$h = 0.05$	
x_n	y_n	t_n	y_n	t_n
0.0	1.00000		1.00000	
0.1	0.99000	0.00010	0.99007	0.00002
0.2	0.96118	0.00036	0.96145	0.00009
0.3	0.91674	0.00069	0.91727	0.00016
0.4	0.86110	0.00096	0.86184	0.00023
0.5	0.79889	0.00111	0.79974	0.00026
0.6	0.73418	0.00111	0.73503	0.00026
0.7	0.67014	0.00100	0.67091	0.00023
0.8	0.60895	0.00080	0.60957	0.00018
0.9	0.55191	0.00058	0.55236	0.00013
1.0	0.49964	0.00036	0.49992	0.00008

Die Resultate zeigen die Konsistenzordnung 2.

9.3.4 Explizite Verfahren von RUNGE und KUTTA

Jetzt: Systematischeres Vorgehen bei der Entwicklung von Verfahren höherer Ordnung. Für die Entwicklung von Einschrittverfahren höherer Ordnung ist es zweckmäßig, die Differentialgleichung $y'(x) = f(x, y(x))$ bezüglich der unabhängigen Variable x über das Intervall $[x_n, x_{n+1}]$ der Länge $h = x_{n+1} - x_n$ zu integrieren. Auf der linken Seite der Differentialgleichung kann die Integration unmittelbar ausgeführt werden. Man erhält

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x))\mathrm{d}x. \tag{9.61}$$

Hinweis: Das Anfangswertproblem (9.5) ist äquivalent zur Integralgleichung

$$y(x) = y_0 + \int_{x_0}^{x} f(x', y(x'))\mathrm{d}x' \tag{9.62}$$

Der Wert des Integrals (9.61) wird durch eine Quadraturformel approximiert:

$$\int_{x_n}^{x_{n+1}} f(x, y(x))\mathrm{d}x \approx h \sum_{r=1}^{m} \gamma_r \, f\left(x_n + \alpha_r h, y(x_n + \alpha_r h)\right) \tag{9.63}$$

mit Gewichten $\gamma_r \geq 0$ und Knoten $x_n + \alpha_r h$ mit $0 \leq \alpha_r \leq 1$ $(r = 1, 2, \ldots, m)$, $\alpha_1 := 0$.

Hier besteht noch das Problem, daß die $y(x_n + \alpha_r h)$ in (9.63) unbekannt sind. Sie müssen daher durch Approximationen ersetzt werden. Dazu macht man folgenden Ansatz:

$$\begin{aligned}
f(x_n + \alpha_1 h, y(x_n + \alpha_1 h)) &=: k_1(x_n, y(x_n)),\\
f(x_n + \alpha_2 h, y(x_n + \alpha_2 h)) &\approx f\left(x_n + \alpha_2 h, y(x_n) + h\beta_{21} k_1(x_n, y(x_n))\right)\\
&=: k_2(x_n, y(x_n)),\\
f(x_n + \alpha_3 h, y(x_n + \alpha_3 h)) &\approx f\left(x_n + \alpha_3 h, y(x_n) + h\left(\beta_{31} k_1(x_n, y(x_n)) + \beta_{32} k_2(x_n, y(x_n))\right)\right)\\
&=: k_3(x_n, y(x_n))\\
&\vdots\\
f(x_n + \alpha_m h, y(x_n + \alpha_m h)) &\approx f\left(x_n + \alpha_m h, y(x_n) + h \sum_{s=1}^{m-1} \beta_{ms} k_s(x_n, y(x_n))\right)\\
&=: k_m(x_n, y(x_n))
\end{aligned} \tag{9.64}$$

mit Konstanten β_{rs}. Dabei soll

$$\sum_{s=1}^{r-1} \beta_{rs} = \alpha_r, \qquad (r = 2, 3, \ldots, m) \tag{9.65}$$

gelten, so daß die Approximationen (9.64) mindestens in $\mathcal{O}(h)$ exakt sind. Denn eine TAYLOR-Entwicklung in der Schrittweite liefert

$$\begin{aligned}
f(x_n + \alpha_r h, y(x_n + \alpha_r h)) - f\left(x_n + \alpha_r h, y(x_n) + h \sum_{s=1}^{r-1} \beta_{rs} k_s(x_n, y(x_n))\right) =\\
= \left(\alpha_r - \sum_{s=1}^{r-1} \beta_{rs}\right) f(x_n, y(x_n))\, \partial_y f(x_n, y(x_n))\, h + \mathcal{O}(h^2).
\end{aligned} \tag{9.66}$$

Der Ansatz (9.64) führt mit (9.63) zu der Verfahrensvorschrift

$$y_{n+1} = y_n + h \sum_{r=1}^{m} \gamma_r k_r(x_n, y(x_n)). \tag{9.67}$$

Aus der Konsistenzforderung

$$\sum_{r=1}^{m} \gamma_r k_r(x_n, y_n) \xrightarrow{h \to 0^+} f(x_n, y_n)$$

folgt

$$\sum_{r=1}^{m} \gamma_r = 1 \tag{9.68}$$

$(k_1(x_n, y_n) = k_2(x_n, y_n) = \cdots = k_m(x_n, y_n)$ bei $h = 0)$.

Das hier beschriebene Verfahren wird als *explizites m-stufiges* RUNGE-KUTTA-*Verfahren* bezeichnet. Die Stufe m gibt die Anzahl der Auswertungen der Funktion f in einem Schritt an. Die Verfahrenskoeffizienten werden üblicherweise in dem in Tabelle 9.2 dargestellten Schema zusammengefasst.

Spezialfälle.

- $m = 1$:

0	
	1

α_1					
α_2	β_{21}				
α_3	β_{31}	β_{32}			
$\vdots$					
α_m	β_{m1}	β_{m2}	$\cdots$	$\beta_{m,m-1}$	
	γ_1	γ_2	$\cdots$	γ_{m-1}	γ_m

Tabelle 9.2: Schema für ein explizites RUNGE-KUTTA-Verfahren

$$\begin{aligned} k_1 &= f(x_n, y_n)\,; \\ y_{n+1} &= y_n + hk_1 \end{aligned} \tag{9.69}$$

EULER-Verfahren; (9.63): Rechtecksregel

- $m = 2$:

0		
1	1	
	$\frac{1}{2}$	$\frac{1}{2}$

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + h, y_n + hk_1)\,; \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2) \end{aligned} \tag{9.70}$$

HEUN-Verfahren; (9.63): Trapezregel

- $m = 2$:

0		
$\frac{1}{2}$	$\frac{1}{2}$	
	0	1

$$\begin{aligned} k_1 &= f(x_n, y_n)\,, \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)\,; \\ y_{n+1} &= y_n + hk_2 \end{aligned} \tag{9.71}$$

COLLATZ-Verfahren; (9.63): Mittelpunktsregel

- $m = 4$:

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

$$
\begin{aligned}
k_1 &= f(x_n, y_n)\,, \\
k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)\,, \\
k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)\,, \\
k_4 &= f\left(x_n + h, y_n + hk_3\right)\,; \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)
\end{aligned}
\tag{9.72}
$$

Klassisches RUNGE-KUTTA-*Verfahren* (wird häufig verwendet); (9.63): SIMPSON-Regel

Bei der sukzessiven Bestimmung der Steigungen wird hier immer nur der direkt vorangegangene Wert benötigt. Die Koeffizienten sind sehr einfach. Konsistenzordnung: 4.

System von Differentialgleichungen:

$$
\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(x_n, \boldsymbol{y}_n)\,, \\
\boldsymbol{k}_2 &= \boldsymbol{f}\left(x_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1\right)\,, \\
\boldsymbol{k}_3 &= \boldsymbol{f}\left(x_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_2\right)\,, \\
\boldsymbol{k}_4 &= \boldsymbol{f}\left(x_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3\right)\,; \\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right)
\end{aligned}
\tag{9.73}
$$

Die maximale erreichbare Konsistenzordnung $p_{\max}$ eines expliziten m-stufigen RUNGE-KUTTA-Verfahrens ist im allgemeinen aufwendig zu bestimmen: TAYLOR-Entwicklungen mit Koeffizientenvergleich führen auf gekoppelte nichtlineare Gleichungen für die Parameter des Schemas.

Die Zahl dieser Bedingungsgleichungen wächst mit wachsendem m rasch an (Tabelle 9.3). Die Parameter des Schemas sind dabei noch nicht vollständig festgelegt. Häufig wird die verbleibende Wahlfreiheit im Hinblick auf sehr einfache Parameterwerte genutzt. Die maximal erreichbare Konsistenzordnung $p_{\max}$ findet man in Tabelle 9.4.

m	1	2	3	4	5	6	7	8
Bedingungsgl.	1	2	4	8	17	37	85	200

Tabelle 9.3: Anzahl der Bedingungsgleichungen im RUNGE-KUTTA-Verfahren

m	1	2	3	4	5	6	7	8	9	10	> 10
$p_{\max}$	1	2	3	4	4	5	6	6	7	7	$\leq m-3$

Tabelle 9.4: Konsistenzordnung $p_{\max}(m)$ für RUNGE-KUTTA-Verfahren

9.10 Beispiel. $y' = -x^3/y^3$, $y(-1) = 1$.
Exakte Lösung $(x^4 + (y(x))^4 = 2)$ bei 0: $y(0) = 2^{1/4}$.
Numerische Lösung bei 0: $\tilde{y}(0)$ mit dem Fehler $|y(0) - \tilde{y}(0)|$, Rechenzeit bezüglich Pentium 133 MHz bei Linux.
Im Hinblick auf die Rechenzeit bei vorgegebener Fehlertoleranz ist das klassische RUNGE-KUTTA-Verfahren den anderen Verfahren vorzuziehen (s. Abbildung 9.11).

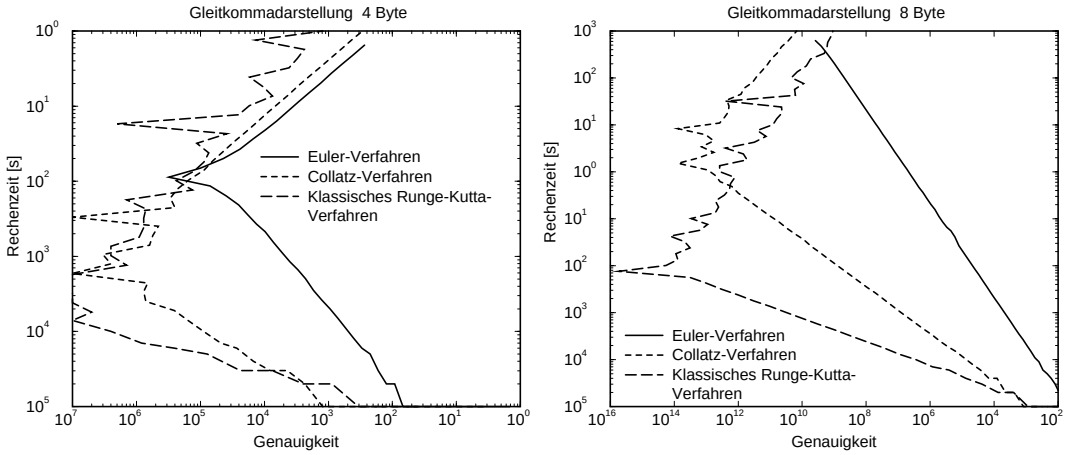


Abbildung 9.11: Vergleich verschiedener Verfahren (Beispiel 9.10)

9.3.5 Implizite Verfahren von RUNGE und KUTTA

Bei einigen Anfangswertproblemen ist es ratsam, implizite Verfahren zu verwenden (s. Abschnitt 9.6).

Implizites m-stufiges RUNGE-KUTTA-Verfahren:

$$k_r(x_n, y_n) = f\left(x_n + \alpha_r h, y_n + h\sum_{s=1}^{m} \beta_{rs} k_s(x_n, y_n)\right) \qquad (r = 1, 2, \ldots, m), \tag{9.74}$$

$$y_{n+1} = y_n + h\sum_{r=1}^{m} \gamma_r k_r(x_n, y_n) \tag{9.75}$$

mit $0 \le \alpha_r \le 1$, $\beta_{rs} \neq 0$ für ein $s \ge r$ und

$$\sum_{s=1}^{m} \beta_{rs} = \alpha_r, \qquad (r = 1, \ldots, m), \tag{9.76}$$

sowie

$$\sum_{r=1}^{m} \gamma_r = 1. \tag{9.77}$$

Das zugehörige Schema findet man in Tabelle 9.5.

α_1	β_{11}	β_{12}	$\cdots$	β_{1m}
α_2	β_{21}	β_{22}	$\cdots$	β_{2m}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
α_m	β_{m1}	β_{m2}	$\cdots$	β_{mm}
	γ_1	γ_2	$\cdots$	γ_m

Tabelle 9.5: Schema für das implizite m-stufige RUNGE-KUTTA-Verfahren

Die Steigungen sind durch ein *implizites*, nichtlineares System von Gleichungen definiert. Für $\beta_{rs} = 0$ für alle s, r mit $s \ge r$ wäre das Verfahren explizit. Ein implizites m-stufiges RUNGE-KUTTA-Verfahren kann maximal die Konsistenzordnung $2m$ haben.

Spezialfälle.

- $m = 1$:

$\frac{1}{2}$	$\frac{1}{2}$
	1

$$\begin{aligned} k_1 &= f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1\right); \\ y_{n+1} &= y_n + hk_1 \end{aligned} \tag{9.78}$$

Konsistenzordnung: 2

- $m = 2$:

0	0	
1	$\frac{1}{2}$	$\frac{1}{2}$
	$\frac{1}{2}$	$\frac{1}{2}$

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f\left(x_n + h, y_n + \frac{1}{2}(k_1 + k_2)\right); \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2) \end{aligned} \tag{9.79}$$

Trapezverfahren, Konsistenzordnung: 2

y_{n+1} kann auch direkt über die implizite Gleichung

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right) \tag{9.80}$$

iterativ berechnet werden. Beschränkt man sich auf nur einen Schritt bei der Iteration mit Startwert $y_n + hf(x_n, y_n)$, so erhält man das Verfahren von Heun (explizit).

- $m = 2$:

$(3-\sqrt{3})/6$	$\frac{1}{4}$	$(3-2\sqrt{3})/12$
$(3+\sqrt{3})/6$	$(3+2\sqrt{3})/12$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

$$\begin{aligned} k_1 &= f\left(x_n + \frac{3-\sqrt{3}}{6}h, y_n + h\left(\frac{1}{4}k_1 + \frac{3-2\sqrt{3}}{12}k_2\right)\right), \\ k_2 &= f\left(x_n + \frac{3+\sqrt{3}}{6}h, y_n + h\left(\frac{3+2\sqrt{3}}{12}k_1 + \frac{1}{4}k_2\right)\right); \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2) \end{aligned} \tag{9.81}$$

Die Konsistenzordnung ist hier 4.

9.4 Schrittweitensteuerung

Häufig arbeitet man nicht mit konstanter Schrittweite. Man versucht vielmehr, die Schrittweite an das Verhalten der Lösung anzupassen (*Schrittweitensteuerung*): Ändert sich die Lösung in einem Bereich schnell, so ist dort eine kleine Schrittweite angebracht. Dagegen ist in Bereichen, in denen die Lösung kaum variiert, eine größere Schrittweite ausreichend.

Die zugrundeliegende Idee ist nun, den Fehler in einem Schritt zu schätzen. Falls die Schätzung deutlich oberhalb einer Toleranzgrenze liegt, wird die Schrittweite verkleinert. Ergibt die Schätzung einen sehr kleinen Fehler, wird die Schrittweite im nächsten Schritt vergrößert. Hinsichtlich der Rechenzeit sollte die Schrittweite so groß wie möglich gewählt werden.

Wir betrachten im folgenden den $(n+1)$-ten Schritt

$$y_{n+1} = y_n + h\varphi(x_n, y_n, h) \tag{9.82}$$

eines expliziten Einschrittverfahrens mit Konsistenzordnung p zur numerischen Bestimmung der Lösung $y(x)$ des Anfangswertproblems

$$y' = f(x,y)\,, \quad y(x_0) = y_0. \tag{9.83}$$

Der Fehler in diesem einzelnen Schritt ist $\tilde{y}(x_{n+1}) - y(x_{n+1})$, wobei $\tilde{y}(x)$ die Lösung des Anfangswertproblems

$$\tilde{y}' = f(x,\tilde{y})\,, \quad \tilde{y}_n = y_n \tag{9.84}$$

sei. Die Schrittweite soll nun so gewählt werden, daß für diesen *lokalen* Fehler

$$|\tilde{y}(x_{n+1}) - y_{n+1}| \lessapprox \varepsilon, \tag{9.85a}$$

$$|\tilde{y}(x_{n+1}) - y_{n+1}| \not\lessapprox \varepsilon \tag{9.85b}$$

mit einer vorgegebenen Fehlertoleranz $\varepsilon > 0$ gilt. Hierbei ist $\tilde{y}(x_{n+1})$ unbekannt. Daher muß der Fehler geschätzt werden.

Der lokale Diskretisierungsfehler

$$\tilde{l}_{n+1} := \tilde{y}(x_{n+1}) - \underbrace{\tilde{y}(x_n)}_{=y_n} - h_n\varphi(x_n, \underbrace{\tilde{y}(x_n)}_{=y_n}, h_n) \tag{9.86}$$

mit $\tilde{y}(x)$ anstelle von $y(x)$ ist gleich dem globalen Diskretisierungsfehler

$$\tilde{g}_{n+1} := \tilde{y}(x_{n+1}) - y_{n+1}. \tag{9.87}$$

Eine asymptotische Entwicklung des globalen Diskretisierungsfehlers in der Schrittweite zeigt

$$\tilde{g}_{n+1} \approx c(x_{n+1})h_n^p \qquad (\ +\mathcal{O}(h_n^{p+1})\ ) \tag{9.88}$$

(Konvergenz des Verfahrens vorausgesetzt). Dabei ist hier

$$c(x_{n+1}) \approx c'(x_n)h_n \tag{9.89}$$

($c(x_n) = 0$ wegen $\tilde{g}_{n+1} = \tilde{l}_{n+1} = \mathcal{O}(h_n^{p+1})$).

Schätzung des Fehlers im $(n+1)$-ten Schritt. Einfachschritt mit Schrittweite h_n:

$$y_{n+1}^{(1)} = y_n + h_n\varphi(x_n, y_n, h_n), \tag{9.90}$$

$$\tilde{g}_{n+1}^{(1)} = \tilde{y}(x_{n+1}) - y_{n+1}^{(1)} \approx c(x_{n+1})h_n^p \approx c'(x_n)h_n^{p+1} \tag{9.91}$$

Doppelschritt mit Schrittweite $\frac{h_n}{2}$:

$$y^{(2)}_{n+\frac{1}{2}} = y_n + \frac{h_n}{2}\varphi\left(x_n, y_n, \frac{h_n}{2}\right), \tag{9.92a}$$

$$y^{(2)}_{n+1} = y^{(2)}_{n+\frac{1}{2}} + \frac{h_n}{2}\varphi\left(x_{n+\frac{1}{2}}, y^{(2)}_{n+\frac{1}{2}}, \frac{h_n}{2}\right) \qquad \left(x_{n+\frac{1}{2}} := x_n + \frac{h_n}{2}\right), \tag{9.92b}$$

$$\tilde{g}^{(2)}_{n+1} = \tilde{y}(x_{n+1}) - y^{(2)}_{n+1} \approx c(x_{n+1})\left(\frac{h_n}{2}\right)^{p+1} \approx c'(x_n)\frac{h_n^{p+1}}{2^p} \tag{9.93}$$

Eine Subtraktion von (9.93) von (9.91) ergibt

$$y^{(2)}_{n+1} - y^{(1)}_{n+1} = \left(1 - 2^{-p}\right) c'(x_n) h_n^{p+1} \tag{9.94}$$

und dann wiederum mit (9.91)

$$\tilde{y}(x_{n+1}) - y^{(1)}_{n+1} \approx c'(x_n) h_n^{p+1} \approx \frac{y^{(2)}_{n+1} - y^{(1)}_{n+1}}{1 - 2^{-p}}. \tag{9.95}$$

Optimale Schrittweite. Sei $\bar{h}_n$ die Schrittweite, für die

$$\left|\tilde{y}(x_n + \bar{h}_n) - y_{n+1}\right| = \varepsilon \tag{9.96}$$

mit

$$\bar{y}_{n+1} = y_n + \bar{h}_n \varphi(x_n, y_n, \bar{h}_n) \tag{9.97}$$

gelte. Nach Gleichung (9.88) mit (9.89) gilt dann

$$\varepsilon \approx \left|c'(x_n)\right| \bar{h}_n^{p+1}. \tag{9.98}$$

Schließlich erhält man aus (9.95) und (9.98)

$$\frac{h_n}{\bar{h}_n} \approx \left(\frac{\left|y^{(2)}_{n+1} - y^{(1)}_{n+1}\right|}{\left(1 - 2^{-p}\right)\varepsilon}\right)^{1/(p+1)}. \tag{9.99}$$

Steuerung der Schrittweite. Nach Berechnung von $y^{(1)}_{n+1}$ und $y^{(2)}_{n+1}$ wird $\frac{h_n}{\bar{h}_n}$ berechnet. Ist $\frac{h_n}{\bar{h}_n} \lessapprox 2$, so wird $y^{(2)}_{n+1}$ ($\leftarrow \frac{h_n}{2}$) als Näherungswert akzeptiert. Ist dagegen $\frac{h_n}{\bar{h}_n} \gg 2$, so ist der lokale Fehler sehr viel größer als die vorgegebene Toleranz ε. Daher muß die Schrittweite reduziert werden. Dazu wird h_n durch $2\bar{h}_n$ ersetzt und $y^{(1)}_{n+1}$ und $y^{(2)}_{n+1}$ sowie $\frac{h_n}{\bar{h}_n}$ erneut berechnet. Falls notwendig, wird die Schrittweite erneut verkleinert. Im nächsten Schritt wird dann $2\bar{h}_n$ als Startschrittweite verwendet. Die akzeptierte Schrittweite in diesem Schritt kann größer als im vorhergehenden Schritt sein. In jedem Schritt wird die Schrittweite nahe des optimalen Wertes eingestellt.

9.11 Beispiel. $y' = -200xy^2$, $y(-3) = 901^{-1}$. Die exakte Lösung ist

$$y(x) = \frac{1}{1 + 100x^2}.$$

Wir benutzen das klassische RUNGE-KUTTA-Verfahren: Näherungswert $y_{\approx}(0) =?$ für $y(0) = 1$

- Bei oben beschriebener Schrittweitensteuerung (Kriterium $h_n/\overline{h}_n \geq 3$ für Verkleinerung der Schrittweite, 12-stellige Rechnung):

$y(0) - y_{\approx}(0)$:	$0.13585 \cdot 10^{-6}$
Zahl der benötigten Schritte:	1476
auftretende kleinste Schrittweite:	$0.1227 \cdot 10^{-2}$

- Bei fester Schrittweite (12-stellige Rechnung):

h	$y(0) - y_{\approx}(0)$	Zahl der Schritte
$0.2032 \cdot 10^{-2}$	$0.5594 \cdot 10^{-6}$	1476
$0.1226 \cdot 10^{-2}$	$0.4052 \cdot 10^{-6}$	2446

Eine feste Schrittweitenwahl liefert in beiden Fällen ein schlechteres Ergebnis als die Schrittweitensteuerung. Im ersten Fall dürfte die feste Schrittweite $0.2032 \cdot 10^{-2}$ im "kritischen" Bereich nahe 0 zu groß sein (größerer lokaler Diskretisierungsfehler). Im zweiten Fall dürfte die feste Schrittweite $0.1226 \cdot 10^{-2}$ im "harmlosen" Bereich von -3 bis nicht zu Nahe 0 zu klein sein (größere Rundungsfehler).

Bemerkung. Die Steuerung der Schrittweite kann auch auf eine andere Weise geschehen. Anstatt zwei Näherungswerte mit demselben Verfahren bei unterschiedlicher Schrittweite zu vergleichen, können auch zwei Näherungen mit unterschiedlichen Verfahren bei gleicher Schrittweite, die aber verschiedene Konsistenzordnungen haben, zur Steuerung der Schrittweite herangezogen werden.

$(n+1)$-ter Schritt:

- Explizites Einschrittverfahren mit Konsistenzordnung p:

$$y_{n+1}^{(1)} = y_n + h_n \varphi_1(x_n, y_n, h_n)\,, \tag{9.100}$$

$$\tilde{g}_{n+1}^{(1)} = \tilde{y}(x_{n+1}) - y_{n+1}^{(1)} \approx c_1'(x_n) h_n^{p+1} \tag{9.101}$$

- Explizites Einschrittverfahren mit Konsistenzordnung $p+1$:

$$y_{n+1}^{(2)} = y_n + h_n \varphi_2(x_n, y_n, h_n)\,, \tag{9.102}$$

$$\tilde{g}_{n+1}^{(2)} = \tilde{y}(x_{n+1}) - y_{n+1}^{(2)} \approx c_2'(x_n) h_n^{p+2} \tag{9.103}$$

Schätzung des Fehlers $\tilde{y}(x_{n+1}) - y_{n+1}^{(1)}$:

$$\tilde{y}(x_{n+1}) - y_{n+1}^{(1)} \approx y_{n+1}^{(2)} - y_{n+1}^{(1)} + \mathcal{O}(h^{p+2}) \tag{9.104}$$

Optimale Schrittweite $\bar{h}_n$ zur Fehlertoleranz $\varepsilon > 0$:

$$|\tilde{y}(x_{n+1}) - \bar{y}_{n+1}| = \varepsilon \qquad \text{mit} \tag{9.105}$$

$$\bar{y}_{n+1} = y_n + \bar{h}_n \varphi_1(x_n, y_n, \bar{h}_n). \tag{9.106}$$

Dann gilt

$$\varepsilon \approx \left| c_1'(x_n) \right| \bar{h}_n^{p+1}\,. \tag{9.107}$$

Aus (9.101) und (9.104) folgt schließlich

$$\bar{h}_n \approx \left(\frac{\varepsilon}{\left| y_{n+1}^{(2)} - y_{n+1}^{(1)} \right|} \right)^{1/(p+1)} h_n. \tag{9.108}$$

Steuerung der Schrittweite:

- Berechnung von $y_{n+1}^{(1)}$ und $y_{n+1}^{(2)}$, sowie $\bar{h}_n$.
- Falls $h_n \leq \tau \bar{h}_n$ (τ ist ein Sicherheitsfaktor, z.B. $\tau = 0.9$.): $y_{n+1}^{(1)}$ wird akkzeptiert.

  Sonst: h_n wird durch $\tau \bar{h}_n$ ersetzt. Dann erfolgt eine neue Berechnung von $y_{n+1}^{(1)}$, $y_{n+1}^{(2)}$ und $\bar{h}_n$. Falls nötig, erneute Verkleinerung der Schrittweite.
- $\tau \bar{h}_n$ als Startschrittweite für nächsten Schritt.

$y_{n+1}^{(2)}$ sollte durch möglichst wenig zusätzlichen Aufwand gegenüber $y_{n+1}^{(1)}$ berechnet werden können. Daher werden zwei explizite RUNGE-KUTTA-Verfahren mit gleichen Parameterwerten α_r und β_{rs} gewählt. Man spricht dann von einer *Einbettung* des Verfahrens niedriger Ordnung in das Verfahren höherer Ordnung.

Siehe Tabelle 9.6 für ein bekanntes Beispiel.

0							
$\frac{1}{4}$	$\frac{1}{4}$						
$\frac{3}{8}$	$\frac{1}{32}$	$\frac{9}{32}$					
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$				
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$			
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{1404}$	$-\frac{11}{40}$		
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0	Ordnung 4
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$	Ordnung 5

Tabelle 9.6: Einbettung eines RUNGE-KUTTA-Verfahrens

9.5 Mehrschrittverfahren

Bei Einschrittverfahren geht nur der Näherungspunkt (x_n, y_n) zur Berechnung des Näherungswertes y_{n+1} an der Stelle x_{n+1} ein. Ein Nachteil dieser Verfahren besteht darin, daß viele Funktionsauswertungen pro Schritt notwendig sind, um eine hohe Konsistenzordnung zu erreichen. Ein Vorteil der Einschrittverfahren liegt in einer einfach durchzuführenden Schrittweitensteuerung.

Im Gegensatz zu Einschrittverfahren werden bei einem Mehrschrittverfahren die vorhergenenden $r \geq 2$ Näherungspunkte

$$(x_{n-r+1}, y_{n-r+1}), \quad (x_{n-r+2}, y_{n-r+2}), \quad \ldots, \quad (x_n, y_n)$$

zur Berechnung des Näherungswertes y_{n+1} an der Stelle x_{n+1} herangezogen. Üblicherweise werden dabei dabei die Stellen x_{n-r+1}, x_{n-r+2}, $\ldots$, x_n als äquidistant vorausgesetzt (Sonst wird es zu kompliziert.). Ein Vorteil von Mehrschrittverfahren ist eine erreichbare hohe Konsistenzordnung bei relativ geringem Rechenaufwand. Ein Nachteil dieser Verfahren ist eine komplizierte Schrittweitensteuerung, weil bei nichtäquidistanten Stellen zusätzliche Näherungswerte an äquidistanten Stellen für die Auswertung der Verfahrensfunktion durch eine Interpolation berechnet werden müssen.

Ausgehend von

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x))\,\mathrm{d}x \tag{9.109}$$

mit den äquidistanten Stellen

$$x_n := a + nh \qquad (n = 0, \ldots, N), \tag{9.110}$$

$$h := \frac{b-a}{N} \tag{9.111}$$

können Mehrschrittverfahren über numerische Integration hergeleitet werden.

Eine naheliegende Idee ist es, den Integranden auf der rechten Seite von Gleichung (9.109) durch das Interpolationspolynom $p_{r-1 \text{ bzw. } r}(x)$ höchstens $(r-1)$-ter bzw. r-ter Ordnung zu den Stützpunkten $(x_{n-r+1+j}, y_{n-r+1-j})$ mit $j = 0, \ldots, r-1$ bzw. r zu ersetzen. Nach der LAGRANGE'schen

Interpolationsformel

$$p_{r-1 \text{ bzw. } r}(x) = \sum_{j=0}^{r-1 \text{ bzw. } r} f(x_{n-r+1+j}, y(x_{n-r+1+j}))\, L_{n-r+1+j}(x) \tag{9.112}$$

mit den LAGRANGE-Polynomen

$$L_{n-r+1+j}(x) := \prod_{k=0, k\neq j}^{r-1 \text{ bzw. } r} \frac{x - x_{n-r+1+k}}{x_{n-r+1+j} - x_{n-r+1+k}} \qquad (j = 0, \ldots, r-1 \text{ bzw. } r) \tag{9.113}$$

(siehe Unterabschnitt 2.1.2) ist dann

$$y(x_{n+1}) \approx y(x_n) + \sum_{j=0}^{r-1 \text{ bzw. } r} f(x_{n-r+1+j}, y(x_{n-r+1+j})) \underbrace{\int_{x_n}^{x_{n+1}} L_{n-r+1+j}(x)\,\mathrm{d}x}_{=:hb_j}. \tag{9.114}$$

Damit ergibt sich die Verfahrensvorschrift

$$y_{n+1} \approx y_n + h \sum_{j=0}^{r-1 \text{ bzw. } r} b_j\, f(x_{n-r+1+j}, y_{n-r+1+j}) \tag{9.115}$$

für ein *explizites* bzw. *implizites r-Schritt-Verfahren*.

Im Folgenden werden einige bekannte Spezialfälle erörtert.

9.5.1 Verfahren von ADAMS und BASHFORTH

Das Integral in (9.109) wird durch das Integral von $p_{r-1}(x)$ über das überhängende Invervall $[x_n, x_{n+1}]$ angenähert (s. Abbildung 9.12).

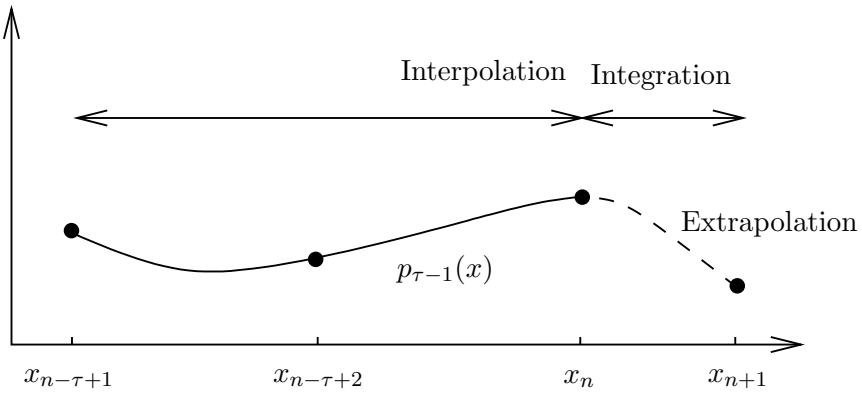


Abbildung 9.12: Verfahren von ADAMS und BASHFORTH

Wir wählen exemplarisch $r = 4$ und berechnen die b_j:

$$\begin{aligned} b_0 &= \frac{1}{h} \int_{x_n}^{x_{n+1}} L_{n-3}(x)\,\mathrm{d}x \\ &= \frac{1}{h} \int_{x_n}^{x_{n+1}} \frac{(x - x_{n-2})(x - x_{n-1})(x - x_n)}{(x_{n-3} - x_{n-2})(x_{n-3} - x_{n-1})(x_{n-3} - x_n)}\,\mathrm{d}x \\ &\overset{x=:x_n+ht}{=} \int_0^1 \frac{(t+2)(t+1)t}{(-1)(-2)(-3)}\,\mathrm{d}t = -\frac{1}{6}\int_0^1 (t^3 + 3t^2 + 2t)\,\mathrm{d}t = -\frac{9}{24}. \end{aligned}$$

Analog berechnet man

$$b_1 = \frac{37}{24}, \qquad b_2 = -\frac{59}{24}, \qquad b_3 = \frac{55}{24}.$$

Wir erhalten somit

$$y_{n+1} = y_n + \frac{h}{24}\left(55f(x_n, y_n) - 59f(x_{n-1}, y_{n-1}) + 37f(x_{n-2}, y_{n-2}) - 9f(x_{n-3}, y_{n-3})\right). \tag{9.116}$$

Pro Schritt ist jeweils nur eine neue Funktionsauswertung notwendig.

Eine TAYLOR-Entwicklung des lokalen Diskretisierungsfehlers

$$l_{n+1} := y(x_{n+1}) - y(x_n) - h\sum_{j=0}^{r-1} b_j f(x_{n-r+1+j}, y(x_{n-r+1+j})) \tag{9.117}$$

bezüglich h ergibt

$$l_{n+1} = cy^{(r+1)}(x_n)h^{r+1} + \mathcal{O}(h^{r+1}) \tag{9.118}$$

mit einer Konstante c, d.h. die Konsistenzordnung des expliziten r-Schritt-Verfahrens von ADAMS und BASHFORTH ist r.

$$c = \frac{251}{720} \qquad \text{für } r = 4. \tag{9.119}$$

Für den Start eines r-Schritt-Verfahrens sind neben y_0 auch noch $r-1$ weitere Startwerte $y_1, y_2, \ldots, y_{r-1}$ notwendig. Sie können über Einschrittverfahren bestimmt werden. Dabei darf das Anlaufverfahren keine kleinere Konsistenzordnung als das Verfahren selbst haben.

9.5.2 Verfahren von ADAMS und MOULTON

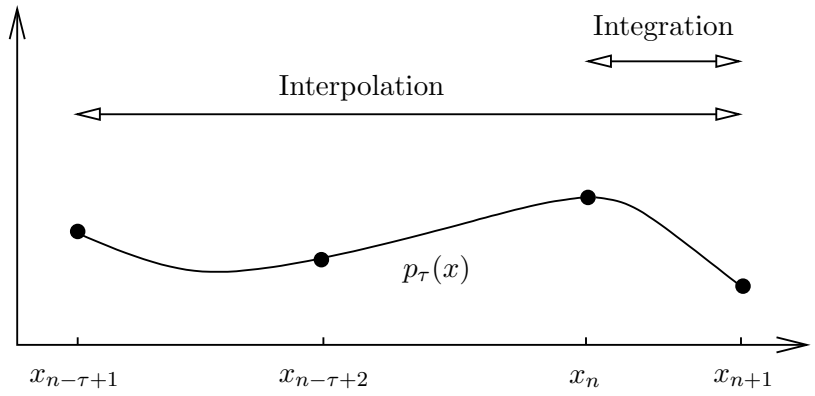


Abbildung 9.13: Verfahren von ADAMS und MOULTON

Wir wählen wieder exemplarisch $r = 4$ und berechnen die b_j:

$$\begin{aligned} b_0 &= \frac{1}{h}\int_{x_n}^{x_{n+1}} L_{n-3}(x)\mathrm{d}x \\ &= \frac{1}{h}\int_{x_n}^{x_{n+1}} \frac{(x-x_{n-2})(x-x_{n-1})(x-x_n)(x-x_{n+1})}{(x_{n-3}-x_{n-2})(x_{n-3}-x_{n-1})(x_{n-3}-x_n)(x_{n-3}-x_{n+1})}\mathrm{d}x \\ &\overset{x=:x_n+ht}{=} \int_0^1 \frac{(t+2)(t+1)t(t-1)}{(-1)(-2)(-3)(-4)}\mathrm{d}t = -\frac{1}{24}\int_0^1 (t^4+2t^3-t^2+2t)\mathrm{d}t = -\frac{19}{720}. \end{aligned}$$

Analog berechnet man

$$b_1 = \frac{106}{720}, \qquad b_2 = -\frac{264}{720}, \qquad b_3 = \frac{646}{720}, \qquad b_4 = \frac{251}{720}.$$

Die Verfahrensvorschrift lautet also

$$y_{n+1} = y_n + \frac{h}{720}\big(251f(x_{n+1}, y_{n+1}) + 646f(x_n, y_n) \\ - 264f(x_{n-1}, y_{n-1}) + 106f(x_{n-2}, y_{n-2}) - 19f(x_{n-3}, y_{n-3})\big). \quad (9.120)$$

Hier geht die bekannte Information an $r = 4$ Stellen mit ein.

Für den lokalen Diskretisierungsfehler

$$l_{n+1} := y(x_{n+1}) - y(x_n) - h\sum_{j=0}^{r} b_j f(x_{n-r+1+j}, y(x_{n-r+1+j})) \quad (9.121)$$

gilt

$$l_{n+1} = cy^{(r+2)}(x_n)h^{r+2} + \mathcal{O}(h^{r+3}) \quad (9.122)$$

mit einer Konstanten c, d.h. die Konsistenzordnung des impliziten r-Schritt-Verfahrens von Adams und Moulton ist $r + 1$. Es ist

$$c = -\frac{3}{160} \qquad \text{für } r = 4. \quad (9.123)$$

Bemerkung. Die iterative Berechnung von y_{n+1} beim impliziten r-Schritt-Verfahren von Adams und Moulton kann mit der *Prädikator-Korrektor-Technik* vermieden werden: r-Schritt-Verfahren von Adams und Bashforth als Prädikator und r-Schritt-Verfahren von Adams und Moulton als Korrektor, z.B. im Falle $r = 4$:

$$\tilde{y}_{n+1} = y_n + \frac{h}{24}\left(55f(x_n, y_n) - 59f(x_{n-1}, y_{n-1}) + 37f(x_{n-2}, y_{n-2}) - 9f(x_{n-3}, y_{n-3})\right),$$
$$y_{n+1} = y_n + \frac{h}{720}\big(251f(x_{n+1}, \tilde{y}_{n+1}) + 646f(x_n, y_n) \\ - 264f(x_{n-1}, y_{n-1}) + 106f(x_{n-2}, y_{n-2}) - 19f(x_{n-3}, y_{n-3})\big).$$

Die Konsistenzordnung des Prädikator-Korrektor-Verfahrens ist $r+1$, also gleich derjenigen des Korrektors. Der Fehlerkoeffizient c des Prädikator-Korrektor-Verfahrens ist aber von der Größenordnung des Fehlerkoeffizienten des Prädikators. Letzterer ist viel größer als derjenige des Korrektors. Eine Verbesserung kann durch die Verwendung des $(r + 1)$-Schritt-Verfahrens von Adams und Bashforth als Prädikator erreicht werden. Die Fehlerkonstante des Prädikator-Korrektor-Verfahrens ist dann gleich derjenigen des Korrektors.

9.5.3 Allgemeiner Ansatz eines Mehrschrittverfahrens

Bei einem allgemeinen Ansatz für eine Verfahrensfunktion eines linearen r-Schritt-Verfahrens gehen auch die Näherungswerte y_{n-r+1}, y_{n-r+2}, …, y_n selbst in Form einer Linearkombination zur Bestimmung von y_{n+1} ein:

$$\sum_{j=0}^{r} a_j y_{n-r+1+j} = h\sum_{j=0}^{r} b_j f(x_{n-r+1+j}, y_{n-r+1+j}) \quad (9.124)$$

mit $a_r \neq 0$ (o.B.d.A. sei $a_r = 1$) und $a_0 \neq 0$ oder $b_0 \neq 0$. Ist $b_r = 0$, so ist das Verfahren explizit ($2r$ Parameter), andernfalls implizit ($2r + 1$ Parameter).

Für den lokalen Diskretisierungsfehler

$$l_{n+1} := \sum_{j=0}^{r} \Big(a_j y(x_{n-r+1+j}) - h b_j \underbrace{f(x_{n-r+1+j}, y(x_{n-r+1+j}))}_{=y'(x_{n-r+1+j})} \Big) \tag{9.125}$$

gilt nach TAYLOR-Entwicklung von $y(x_{n-r+1+j})$ und $y'(x_{n-r+1+j})$ um x_{n-r+1} (hier nicht um x_n, Konsistenzordnung davon unabhängig)

$$l_{n+1} := c_0 y(x_{n-r+1}) + c_1 h y'(x_{n-r+1}) + \cdots + c_q h^q y^{(q)}(x_{n-r+1}) + \cdots \tag{9.126}$$

mit

$$\begin{aligned} c_0 &= a_0 + a_1 + \cdots a_r \\ c_1 &= a_1 + 2a_2 + \cdots + r a_r - (b_0 + b_1 + \cdots + b_r) \\ c_q &= \frac{1}{q!}\left(a_1 + 2^q a_2 + \cdots + r^q a_r\right) - \frac{1}{(q-1)!}\left(b_1 + 2^{q-1} b_2 + \cdots + r^{q-1} b_r\right) \quad (q = 2, 3, 4, \dots). \end{aligned} \tag{9.127}$$

Zu vorgegebenem r werden die Parameter a_j und b_j so bestimmt, daß unter gewissen Bedingungen an die Struktur des Verfahrens die Konsistenzordnung maximal wird.

9.12 Beispiel (Explizites lineares 3-Schritt-Verfahren maximaler Konsistenzordnung).

$$a_0 y_{n-2} + a_1 y_{n-1} + a_2 y_n + \underset{(=1)}{a_3}\, y_{n+1} = h\left(b_0 f(x_{n-2}, y_{n-2}) + b_1 f(x_{n-1}, y_{n-1}) + b_2 f(x_n, y_n)\right) \tag{9.128}$$

Sei $a_0 = 0 = a_2$. Nach (9.127) ist

$$\begin{aligned} c_0 &= a_1 + 1 \\ c_1 &= a_1 + 3 - b_0 - b_1 - b_2 \\ c_2 &= \frac{1}{2}a_1 + \frac{9}{2} - b_1 - 2b_2 \\ c_3 &= \frac{1}{6}a_1 + \frac{9}{2} - \frac{1}{2}b_1 - 2b_2 \\ c_4 &= \frac{1}{24}a_1 + \frac{27}{8} - \frac{1}{6}b_1 - \frac{4}{3}b_2 \ \dots . \end{aligned} \tag{9.129}$$

Verlangt man $c_0 = c_1 = c_2 = c_3 = 0$, so ergibt sich

$$a_1 = -1, \qquad b_0 = \frac{1}{3}, \qquad b_1 = -\frac{2}{3}, \qquad b_2 = \frac{7}{3}. \tag{9.130}$$

Dann ist

$$c_4 = \frac{1}{3} \neq 0. \tag{9.131}$$

Die maximal erreichbare Konsistenzordnung ist also gleich 3.

Im Gegensatz zu Einschrittverfahren ist bei linearen Mehrschrittverfahren mit einer die LIPSCHITZ-Bedingung erfüllenden Verfahrensfunktion die Konsistenz nicht hinreichend für die Konvergenz des Verfahrens. Dazu muß noch die sogenannte Nullstabilität erfüllt sein. Dann ist die Konvergenzordnung gleich der Konsistenzordnung.

LIPSCHITZ-Bedingung:

$$|\varphi(x, y_0, y_1, \dots, y_r, h) - \varphi(x, \tilde{y}_0, \tilde{y}_1, \dots, \tilde{y}_r, h)| \leq L \cdot \sum_{j=0}^{r} |y_j - \tilde{y}_j| \tag{9.132}$$

mit

$$\varphi(x, y_0, \dots, y_r) := \sum_{j=0}^{r} b_j f(x + jh, y_j). \tag{9.133}$$

9.5.4 Nullstabilität

Zu einem Verfahren gemäß (9.124) definiert man das *erste charakteristische Polynom*

$$\varrho(z) := \sum_{j=0}^{r} a_j z^j \tag{9.134}$$

und das *zweite charakteristische Polynom*

$$\sigma(z) := \sum_{j=0}^{r} b_j z^j. \tag{9.135}$$

Nach (9.127) ist das lineare r-Schritt-Verfahren genau dann konsistent, wenn

$$\varrho(1) = c_0 = 0 \qquad \text{und} \tag{9.136a}$$
$$\varrho'(1) - \sigma(1) = c_1 = 0 \tag{9.136b}$$

gilt.

Zur Illustrationszwecken betrachten wir im folgenden das Test-Anfangswertproblem

$$y' = \lambda y, \qquad y(0) = 1 \tag{9.137}$$

für $\lambda < 0$ mit der bekannten Lösung

$$y(x) = \mathrm{e}^{\lambda x}. \tag{9.138}$$

Die zugehörige Verfahrensvorschrift lautet dann

$$\sum_{j=0}^{r} (a_j - h\lambda b_j) y_{n-r+1+j} = 0. \tag{9.139}$$

Der Lösungsansatz

$$y_k = z^k \tag{9.140}$$

für diese Differentialgleichung führt zu der sogenannten *charakteristischen Gleichung*

$$\sum_{j=0}^{r} (a_j - b\lambda b_j) z^j = 0, \tag{9.141}$$

die mit

$$\chi(z) := \varrho(z) - h\lambda\sigma(z) \tag{9.142}$$

als

$$\chi(z) = 0 \tag{9.143}$$

geschrieben werden kann. Besitzt $\chi(z)$ r paarweise verschiedene Nullstellen $z^{(1)}$, $z^{(2)}$, …, $z^{(r)}$, so bilden

$$\left(z^{(1)}\right)^k, \quad \left(z^{(2)}\right)^k, \quad \ldots, \quad \left(z^{(r)}\right)^k$$

ein vollständiges System linear unabhängiger Lösungen der Differenzengleichung (9.139). Fallen zwei Nullstellen $z^{(j_1)}$, $z^{(j_2)}$ zusammen, so gehen $\left(z^{(j_1)}\right)^k$ und $k\left(z^{(j_1)}\right)^k$ (anstelle von $\left(z^{(j_2)}\right)^k$) in dieses System ein. Die allgemeine Lösung der Differentialgleichung (9.139) ist eine beliebige Linearkombination dieser fundamentalen Lösungen. Die zugehörigen Koeffizienten können durch die Startwerte $y_0, \ldots, y_{r-1}$ ausgedrückt werden.

Die Näherungswerte y_k sollen sich qualitativ so wie die exakte Lösung $y(x)$ verhalten, d.h. $y_k \to 0$ für $k \to \infty$. Deshalb müssen die Nullstellen der charakteristischen Gleichung betragsmäßig kleiner

als eins sein. Dies soll für beliebig kleine h gelten. Für $h \to 0$ gehen diese Nullstellen in Nullstellen des ersten charakteristischen Polynoms über. Diese sind dann betragsmäßig nicht größer als eins. Mehrfache Nullstellen des ersten charakteristischen Polynoms dürfen sogar betragsmäßig nicht gleich eins sein (wie der Grenzfall $\lambda \to 0$ zeigt). Ein Mehrschrittverfahren mit der Eigenschaft, daß alle Nullstellen des zugehörigen ersten charakteristischen Polynoms betragsmäßig höchstens gleich eins sind, mehrfache Nullstellen betragsmäßig sogar kleiner als eins sind, heißt *nullstabil*.

Zum Beispiel sind das r-Schritt-Verfahren von ADAMS und BASHFORTH und das r-Schritt-Verfahren von ADAMS und MOULTON nullstabil, da ihr erstes charakteristisches Polynom

$$\rho(z) = z^r - z^{r-1} = (z-1)z^{r-1}$$

eine einfache Nullstelle 1 und eine $(r-1)$-fache Nullstelle 0 hat.

9.6 Absolute Stabilität. Steife Differentialgleichungssysteme

Bei unsachgemäßer Anwendung von numerischen Verfahren zur Lösung von Anfangswertproblemen können Instabilitäten auftreten. Im Folgenden geht es darum, wie sie vermieden werden können.

Zur Illustration betrachten wir das Test-Anfangswertproblem

$$y' = \lambda y, \qquad y(0) = 1 \tag{9.144}$$

für $\Re(\lambda) < 0$ mit der bekannten Lösung

$$y(x) = \mathrm{e}^{\lambda x}. \tag{9.145}$$

Dieses Anfangswertproblem wollen wir mit dem klassischen RUNGE-KUTTA-Verfahren (siehe (9.72)) lösen:

$$\begin{aligned} k_1 &= \lambda y_n, \\ k_2 &= \lambda \left(y_n + \frac{h}{2} k_1 \right), \\ k_3 &= \lambda \left(y_n + \frac{h}{2} k_2 \right), \\ k_4 &= \lambda \left(y_n + h k_3 \right); \\ y_{n+1} &= y_n + \frac{h}{6} \left(k_1 + 2k_2 + 2k_3 + k_4 \right). \end{aligned} \tag{9.146}$$

Es ist also

$$y_{n+1} = F(\lambda h) y_n \tag{9.147}$$

mit

$$F(\lambda h) = 1 + \lambda h + \frac{\lambda^2 h^2}{2} + \frac{\lambda^3 h^3}{6} + \frac{\lambda^4 h^4}{24}. \tag{9.148}$$

Für die exakte Lösung gilt

$$y(x_{n+1}) = \mathrm{e}^{\lambda h} y(x_n). \tag{9.149}$$

Offensichtlich ergibt eine TAYLOR-Entwicklung von $\mathrm{e}^{\lambda h}$ bis zur 4. Ordnung in λh den Faktor $F(\lambda h)$. Dies ist im Einklang damit, daß der lokale Diskretisierungsfehler von $\mathcal{O}(h^5)$ ist.

Die exakte Lösung klingt ab ($|y(x)| \to 0$ für $x \to \infty$). Die Näherungslösung klingt nur dann ab ($y_n \to 0$ für $n \to \infty$), wenn

$$|F(\lambda h)| < 1 \tag{9.150}$$

gilt. Wegen $|F(\lambda h)| \to \infty$ für $\Re(\lambda) h \to -\infty$ ist dies nicht für alle λh erfüllt. Für hinreichend kleine h gilt aber (9.150).

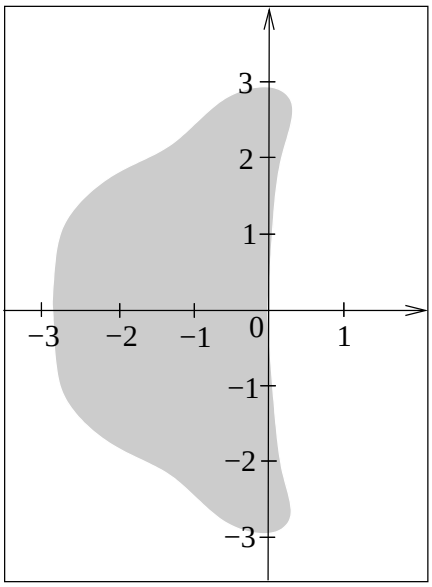


Abbildung 9.14: Stabilitätsbereich beim klassischen RUNGE-KUTTA-Verfahren

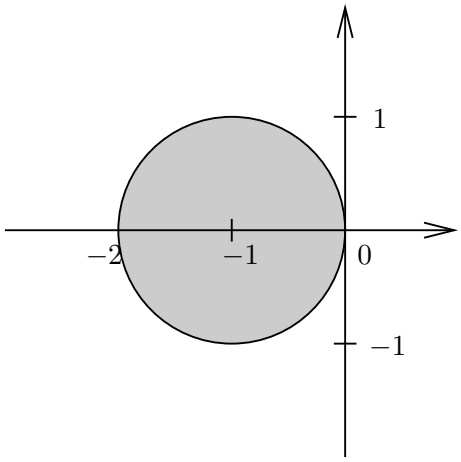


Abbildung 9.15: Stabilitätsbereich des EULER-Verfahrens

Die Menge

$$\{\mu \in \mathbb{C} |\ |F(\mu)| < 1\}$$

heißt *Bereich absoluter Stabilität* des Verfahrens. Ein Maß für dessen Größe ist das sogenannte *Stabilitätsintervall* := Stabilitätsbereich ∩ reelle Achse.

Der Stabilitätsbereich beim klassischen RUNGE-KUTTA-Verfahren liegt symetrisch zur reellen Achse (s. Abbildung 9.14). Das zugehörige Stabilitätsintervall ist $]-2.78, 0[$.

9.13 Beispiel (Stabilitätsbereich des EULER-Verfahrens). Beim EULER-Verfahren ist hier

$$F(\lambda h) = 1 + \lambda h, \tag{9.151}$$

der Stabilitätsbereich ist $\{\mu \in \mathbb{C} : |\mu - 1| < 1\}$. Das Stabilitätsintervall ist $]-2; 0[$ (s. Abbildung 9.15).

Die Schrittweite $h > 0$ ist so zu wählen, daß λh im Bereich absoluter Stabilität liegt. Sonst liefert das Verfahren ein unsinniges Ergebnis.

Umfasst der Bereich absoluter Stabilität die ganze Halbebene $\Re(\mu) < 0$, so heißt das Verfahren *absolut stabil*.

9.14 Beispiel (Implizites 2-stufiges RUNGE-KUTTA-Verfahren gemäß (9.81)).

$$\begin{aligned} k_1 &= \lambda\left(y_n + \frac{1}{4}hk_1 + \frac{3-2\sqrt{3}}{12}hk_2\right), \\ k_2 &= \lambda\left(y_n + \frac{3+2\sqrt{3}}{12}hk_1 + \frac{1}{4}hk_2\right); \\ y_{n+1} &= y_n + \frac{h}{2}(k_1+k_2) \end{aligned} \tag{9.152}$$

Eine einfache Rechnung ergibt

$$y_{n+1} = F(\lambda h)y_n \qquad \text{mit} \qquad F(\mu) = \frac{1+\mu/2+\mu^2/12}{1-\mu/2+\mu^2/12}. \tag{9.153}$$

Sei $|F(\mu)| = 1$. Dann gilt

$$1 + \frac{\mu}{2} + \frac{\mu^2}{12} = \mathrm{e}^{i\vartheta}\left(1 - \frac{\mu}{2} + \frac{\mu^2}{12}\right) \qquad \text{mit geeignetem } 0 \le \vartheta < 2\pi, \tag{9.154}$$

also

$$\mu^2 + 6\,\mathrm{i}\,\cot\left(\frac{\vartheta}{2}\right)\mu + 12 = 0. \tag{9.155}$$

Die Lösungen

$$\mu_\pm = -\mathrm{i}\left(3\cot\left(\frac{\vartheta}{2}\right) \pm \sqrt{12 + 9\cot^2\left(\frac{\vartheta}{2}\right)}\right) \tag{9.156}$$

dieser quadratischen Gleichung sind rein imaginär. Mithin bildet die imaginäre Achse den Rand des Stabilitätsbereiches. Für negative μ ist offenbar $|F(\mu)| < 1$. Demnach ist das Verfahren absolut stabil.

Auch die anderen impliziten RUNGE-KUTTA-Verfahren sind absolut stabil. Dagegen sind die expliziten RUNGE-KUTTA-Verfahren nicht absolut stabil.

Das analoge Stabilitätsproblem stellt sich bei den linearen Mehrschrittverfahren. Der Bereich der absoluten Stabilität ist hier durch die Menge aller $\mu \in \mathbb{C}$ gegeben, für welche die charakteristische Gleichung $\varrho(z) - \mu\sigma(z) = 0$ nur Lösungen im Inneren des Einheitskreises besitzt (s. Abschnitt 5). Die Stabilitätsintervalle sowohl der ADAMS-BASHFORTH-Verfahren (explizit) als auch des ADAMS-MOULTON-Verfahren (implizit) sind endlich.

Die Lösungen von Differentialgleichungssystemen, die physikalische (oder chemische oder biologische) Vorgänge beschreiben, haben oftmals die Eigenschaft, daß sie sich exponentiell u.U. mit Oszillationen einer stationären Lösung annähern (z.B. Einschwingvorgänge). Dabei können die einzelnen Komponenten der Lösung stark verschieden rasch auf einen konstanten Wert zulaufen. Um solche Systeme mit einer nicht allzu kleinen Schrittweite numerisch lösen zu können, sollten nur absolut stabile Verfahren oder zumindest Verfahren mit einem großen Bereich absoluter Stabilität gewählt werden.

Zur Illustration betrachten wir das Test-Anfangswertproblem

$$\boldsymbol{y}' = A\boldsymbol{y} + \boldsymbol{b}, \qquad \boldsymbol{y}(x_0) = \boldsymbol{y}_0 \tag{9.157}$$

mit einer $d \times d$-Matrix A, deren Eigenwerte λ_i $(i = 1, \ldots, d)$ alle negative Realteile haben sollen.

Sind diese Realteile sehr unterschiedlich, so wird das Anfangswertproblem als *steif* bezeichnet. Als Maß für die *Steifheit* wird der sogenannte *Steifheitsquotient*

$$S := \frac{\max_i |\Re(\lambda_i)|}{\min_i |\Re(\lambda_i)|} \tag{9.158}$$

definiert. In manchen Fällen erreicht S Werte von der $\mathcal{O}(10^6)$.

Bei der numerischen Lösung des Anfangswertproblems müssen alle Produkte $h\lambda_i$ im Bereich absoluter Stabilität liegen.

9.15 Beispiel.

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix} = \begin{pmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \qquad \begin{pmatrix} y_1(0) \\ y_2(0) \\ y_3(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \tag{9.159}$$

Die exakte Lösung ist

$$\begin{pmatrix} y_1(x) \\ y_2(x) \\ y_3(x) \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\mathrm{e}^{-2x} + \frac{1}{2}\left(\cos(40x) + \sin(40x)\right)\mathrm{e}^{-40x} \\ \frac{1}{2}\mathrm{e}^{-2x} - \frac{1}{2}\left(\cos(40x) + \sin(40x)\right)\mathrm{e}^{-40x} \\ -\left(\cos(40x) - \sin(40x)\right)\mathrm{e}^{-40x} \end{pmatrix}. \tag{9.160}$$

Numerische Lösung mit dem EULER-Verfahren: Startwert $(y_1(0.1), y_2(0.1), y_3(0.1))^\intercal$ bei Startstelle 0.1, Schrittweite 0.04. Das Ergebnis ist in der Abbildung 9.16 zu sehen. Die letzten Näherungswerte sind nicht mehr akzeptabel.

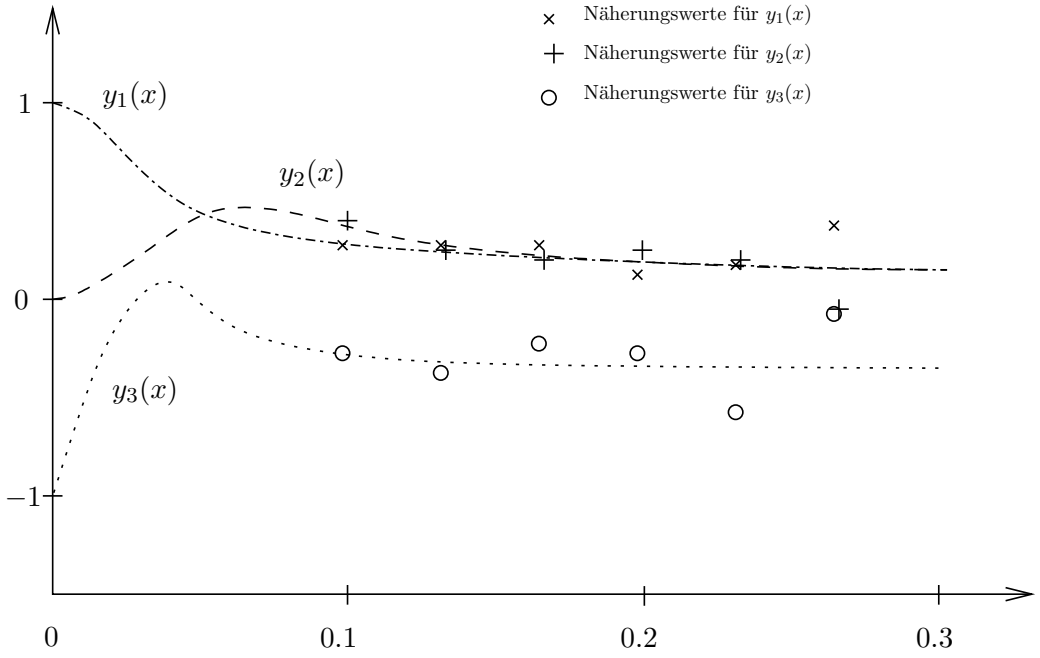


Abbildung 9.16: Analytische und numerische Lösung von Beispiel 9.15 bei kleinen Abszissenwerten

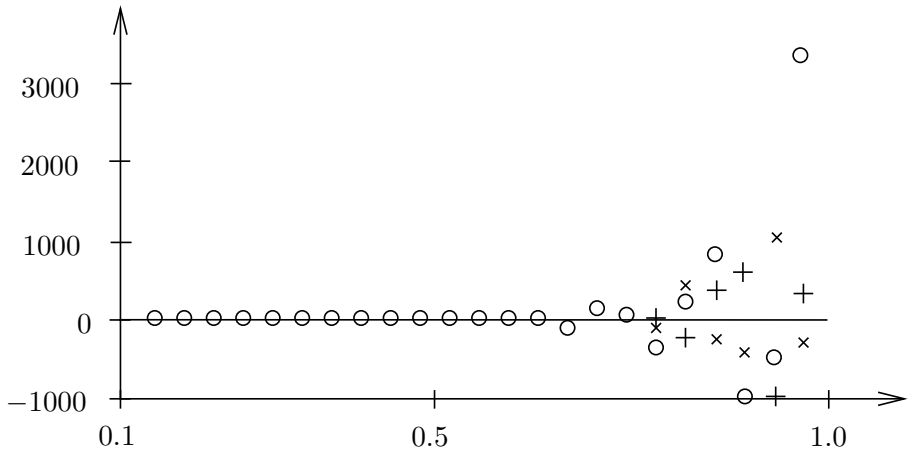


Abbildung 9.17: Analytische und numerische Lösung von Beispiel 9.15 bei größeren Abszissenwerten (Beschriftung wie in Abbildung 9.16)

Viel schlimmer sieht die Situation aus, wenn das größere Intervall von 0.1 bis 1.0 betrachtet wird (beachte geänderte Skalierung der Ordinate in Abbildung 9.17 im Vergleich zur Ordinate in Abbildung 9.16).

Die Matrix in (9.159) hat die Eigenwerte $\lambda_1 = -2$, $\lambda_2 = -40 + 40\mathrm{i}$ und $\lambda_3 = -40 - 40\mathrm{i}$. Somit ist der Steifheitsquotient $S = 20$ (nicht besonders steif). Bei der verwendeten Schrittweite $h = 0.04$ liegt $h\lambda_1 = 0.08$ im Stabilitätsbereich

$\{\mu \in \mathbb{C} : |\mu + 1| < 1\}$ des EULER-Verfahrens; $h\lambda_2 = -1.6 + 1.6\mathrm{i}$ und $h\lambda_3 = -1.6 - 1.6\mathrm{i}$ liegen außerhalb des Stabilitätsbereiches. Verantwortlich für die starke Abweichung der Näherungslösung von der exakten Lösung ist die Verletzung der Stabilitätsbedingung bei den Eigenwerten λ_2 und λ_3, *obwohl* deren Anteile an der Lösung ab 0.1 fast verschwinden.

Damit alle Produkte $h\lambda_1$, $h\lambda_2$, $h\lambda_3$ im Stabilitätsbereich liegen, muß die Schrittweite $h < 0.025$ sein. Dann erhält man eine befriedigende Näherungslösung.

Bei einem allgemeinen Anfangswertproblem

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \qquad \boldsymbol{y}(x_0) = \boldsymbol{y}_0 \tag{9.161}$$

kann die Steifheit über eine lokale Linearisierung erfaßt werden. Dazu wird das Verhalten der exakten Lösung $\boldsymbol{y}(x)$ der Differentialgleichung zu der Anfangsbedingung $\boldsymbol{y}(x_n) = y_n$ nahe x_n untersucht, wobei $\boldsymbol{y}_n$ der berechnete Näherungswert an der Stelle x_n ist. Der Ansatz

$$\boldsymbol{y}(x) = \boldsymbol{y}_n + \boldsymbol{\delta}(x) \tag{9.162}$$

führt zu

$$\boldsymbol{\delta}'(x) = \boldsymbol{f}(x_n + (x - x_n), \boldsymbol{y}_n + \boldsymbol{\delta}(x)) \approx \boldsymbol{f}(x_n, \boldsymbol{y}_n) + \partial_x \boldsymbol{f}(x_n, \boldsymbol{y}_n)(x - x_n) + \partial_{\boldsymbol{y}} \boldsymbol{f}(x_n, \boldsymbol{y}_n)\, \boldsymbol{\delta}(x) \tag{9.163}$$

nahe x_n mit der Funktionalmatrix

$$\partial_{\boldsymbol{y}} \boldsymbol{f}(x_n, \boldsymbol{y}_n) := \left(\frac{\partial f_i}{\partial y_j}(x_n, \boldsymbol{y}_n) \right)_{1 \le i,j \le d}. \tag{9.164}$$

Der mit ihren Eigenwerten (negative Realteile angenommen) definierte Steifheitsquotient ist jetzt sowohl von der Stelle x_n als auch vom Näherungswert $\boldsymbol{y}_n$ an dieser Stelle abhängig. Im Verlauf der numerischen Lösung des Anfangswertproblems kann sich daher der Steifheitsquotient ändern (auch stark).

Index