Numerische Mathematik I

Wintersemester 2007/08

Dr. Serge Kräutle

geTEXt von A. Schleich und Ch. Basting

Vorlesungsskript

Department für Mathematik

AM1

Friedrich–Alexander–Universität Erlangen–Nürnberg

Inhaltsverzeichnis

0 Einleitung

Die Numerik beschäftigt sich mit der konkreten, d.h. zahlenmäßigen Lösung von mathematischen Problemen. Die Rechenvorschriften/Algorithmen sollen in der Regel auf Computern implementiert werden. Das gegebene mathematische Problem wird dazu meist durch ein *angenähertes* Problem, dessen Lösung praktikabel ist (insbes. *endlich* viele Rechenoperationen/Speicher erfordert), approximiert.
Die Bedeutung der Numerik in den Ingenieur-, Natur-, Gesellschaftswissenschaften ist enorm und wächst ständig.

Anwendungen (winzige Auswahl):

Wettervorhersage, Klimaprognosen, Konstruktion von Fahrzeugen mit niedrigem Luftwiderstand, Crashtest, Simulation von Erdöllagerstätten, von Wirtschaftsabläufen, von Ausbreitung von Seuchen, Galaxienbildung, Optimierung von Prozessabläufen,... (keine Mathematik "um ihrer selbst willen"!)

Aspekte der Numerik:

1. Konstruktion durchführbarer Verfahren (Algorithmen) zur (approximativen) Lösung des Problems
2. Genauigkeit der Berechnung ("Fehler")
3. Effizienz?
4. Stabilität (Verstärkung von Fehlern im Verlauf der Rechnung)

Fehlerarten:

Eingabefehler: gegebene Daten sind Messwerte, behaftet mit Messfehlern, oder beruhen auf vorangegangenen Berechnungen, die fehlerbehaftet sind.

Rundungsfehler: Computer rechnet nur mit endlich vielen (Dezimal/Binär-) Stellen.

Verfahrensfehler: das "exakte" Problem wird durch ein angenähertes Problem (dessen Lösung nur endlich viele Operationen/Speicher benötigt) ersetzt, wie groß ist der Abstand der beiden Probleme?

Beispiel 0.1 *(Verfahrensfehler, Effizienz)*
Berechne $I = \int_0^1 e^{x^2} dx$*; Verfahren: Approximiere* $\int_a^b f(x)dx$ *durch:*

$$I_{\text{Riemann}} := h \cdot \sum_{i=0}^{n-1} f(x_i),\ x_i = a + i \cdot h,\ h = \frac{b-a}{n}$$

oder

$$I_{\text{Trapez}} := \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} \cdot h = h \cdot \left[\frac{f(x_0)}{2} + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2} \right]$$

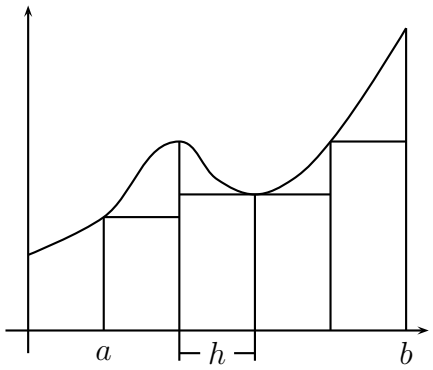


Abbildung 1: RIEMANN-Verfahren

oder

$$\begin{aligned} I_{\text{Simpson}} \underset{n \text{ gerade}}{:=} & \sum_{i=0, igerade}^{n-2} h \cdot \frac{f(x_i) + 4f(x_{i+1}) + f(x_{i+2})}{6} \\ = & \frac{h}{6} \cdot [f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) \\ & + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)] \end{aligned}$$

Es sind jeweils endlich viele Rechenschritte nötig, d. h. die Verfahren verwenden endlich viele Werte (Speicherplatz).
Numerisches Experiment: für z. B. $n = 10$ *ist der*

$$\textit{Verfahrensfehler} \begin{cases} |I - I_{Riemann}| & \approx 0,031 & (\approx n \ \textit{Add.}) \\ |I - I_{Trapez}| & \approx 0,00061 & (\approx n \ \textit{Add.}) \\ |I - I_{Simpson}| & \approx 8,2 \cdot 10^{-7} & (\approx n \ \textit{Add.}) \end{cases}$$

Theorie:

- *Für* $n \to \infty$ *konvergiert Verfahrensfehler gegen* 0 *und*
- *ist für das* RIEMANN*-Verfahren größer als für das Trapez als für* SIMPSON*.*
- *Voraussetzungen an* f *für Fehlerabschätzung?*

Fazit: SIMPSON*-Regel ist die effizienteste der drei!*

Beispiel 0.2 *(Fehlerverstärkung)*
Betrachte LGS

$$\begin{pmatrix} 10 & 7,1 \\ 2 & \sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 10 \\ 4 \end{pmatrix}$$

Mit Gauß

$$\Rightarrow \begin{pmatrix} 10 & 7,1 \\ 0 & \sqrt{2} - 1,42 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 10 \\ 2 \end{pmatrix} \tag{0.1}$$

$\Rightarrow$ *exakte Lösung:*

$$\begin{aligned} y &= \frac{2}{\sqrt{2}-1,42} = \frac{100}{50\sqrt{2}-71} \approx -345,635\ldots \\ x &= 1-0,71y = \frac{50\sqrt{2}-142}{50\sqrt{2}-71} \approx 246,401\ldots \end{aligned} \tag{0.2}$$

Der Computer kann nicht exakt mit $\sqrt{2}$*, sondern nur mit endlich vielen Dezimal (Binär-)stellen rechnen.*
Auswerung der Formel (0.2):

bei $rd(\sqrt{2}) = 1.41$	$y_{num} = -200$	$x_{num} = 143$
bei $rd(\sqrt{2}) = 1.414$	$y_{num} = -333,333\ldots$	$x_{num} = 237,666\ldots$
bei $rd(\sqrt{2}) = 1.4142$	$y_{num} = -344,827\ldots$	$x_{num} = 245,827\ldots$

Der Rundungsfehler in $\sqrt{2}$ *wird um Faktor* ~ 42000 *in* x*, um Faktor* ~ 60000 *in* y *verstärkt! (Hintergrund: Differenz von nahezu gleich großen Zahlen "* $50\sqrt{2}-71$*".)*
Es ist besser, statt (0.2) die folgende Formel auszuwerten:

$$y = \frac{100(50\sqrt{2}+71)}{(50\sqrt{2}-71)(50\sqrt{2}+71)} = \frac{-(5000\sqrt{2}+7100)}{41} \tag{0.3}$$

ergibt:

bei $rd(\sqrt{2}) = 1.41$	$y_{num} = -345,1219$
bei $rd(\sqrt{2}) = 1.414$	$y_{num} = -345,6097\ldots$
bei $rd(\sqrt{2}) = 1.4142$	$y_{num} = -345,6341\ldots$

Der Rundungsfehler wird also etwa um den Faktor ~ 122 *verstärkt.*

Inhaltsübersicht Numerik I:

- Fehleranalyse und Störungsrechnung
- QR-Zerlegung und lineare Ausgleichprobleme
- Fixpunktiteration und Newton-Verfahren
- Iterative Verfahren für LGS
- Eigenwertprobleme
- Interpolation
- Quadratur (Integration)

Weiterführende Vorlesungen:

- Numerik II : Numerische Methoden für gewöhnliche Differentialgleichungen
- Numerik partieller Differentialgleichungen

Literatur:

- W. Oevel, Einführung in die Numerische Mathematik, '95
- Schwarz & Köckler, Numerische Mathematik, 6.Auflage: 2006
- R. Schaback & H.Werner, Numerische Mathematik, '93
- J. Stoer, Einführung in die Numerische Mathematik, Band 1& 2, '72/'73
- M. Hanke-Bourgeois, Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens, 2. Auflage: 2006
- Skript von P. Knabner

Definition 0.3 *(O-Kalkül, Landau-Symbole)*
Für Funktionen $f, g : \mathbb{R} \to \mathbb{R}, x_0 \in \mathbb{R} \cup \{\infty\}$ *(und analog für* $f, g : \mathbb{N} \to \mathbb{R}, x_0 = \infty$*) definieren wir:*

(1) $f = O(g)$ *für* $x \to x_0 \;\Leftrightarrow\; \exists c > 0 : |f(x)| \leq c \cdot |g(x)|$ *in einer Umgebung von* x_0
d.h.: "f wächst höchstens so stark wie g".

(2) $f = o(g)$ *für* $x \to x_0 \;\Leftrightarrow\; \frac{f(x)}{g(x)} \to 0$ *für* $x \to x_0$
d.h.: "f wächst echt schwächer als g".

Beispiele:

- $5x^3 + \frac{1}{2}x^2 = O(x^3)$ für $x \to \infty$
- $5x^3 + \frac{1}{2}x^2 = O(x^2)$ für $x \to 0$
- $x \ln x = o(x^{1+\epsilon})$ für $x \to \infty \; \forall \epsilon > 0$

Ferner schreiben wir gelegentlich $f(x) \doteq g(x)$ für $x \to x_0$, falls

$$f(x) = g(x) + o(g(x)) \text{ für } x \to x_0$$

und analog auch $f(x) \dot{\leq} g(x), \text{falls} f(x) \leq g(x) + o(g(x))$.

1 Fehleranalyse und Störungsrechnung

1.1 Maschinenzahlen

Bisher sind wir gewohnt im Körper $\mathbb{R}$ zu rechnen. Man kann zeigen, dass jede Zahl $x \in \mathbb{R}, x \neq 0$, bei vorgegebener Basis $b \in \mathbb{N}, b \geq 2$, genau eine Darstellung der Form

$$x = \sigma \cdot \left(\sum_{\nu=1}^{\infty} x_\nu b^{-\nu} \right) \cdot b^n$$

besitzt mit $n \in \mathbb{Z},\ \sigma \in \{-1, 1\},\ x_\nu \in \{0, \ldots, b-1\},\ x_1 \neq 0$, und dass zu jedem $\nu_0 \in \mathbb{N}$ existiert ein $\nu \geq \nu_0$ mit $x_\nu \neq b-1$. Man schreibt auch

$$x = 0, x_1 x_2 x_3 \ldots \cdot b^n$$

und nennt die Folge (x_ν) die b-adische Darstellung von x.

Beispiel: $x = 205,43$ und $b = 10 \quad \Rightarrow \quad n = 3,\ x_1 = 2,\ x_2 = 0,\ x_3 = 5,\ x_4 = 4,\ x_5 = 3$ und $\sigma = +1 \quad \Rightarrow \quad x = 0,20543 \cdot 10^3$.

Computer können jedoch nur *endlich* viele Informationen (Ziffern) speichern und verarbeiten. Im Computer darstellbare Zahlen (= Maschinenzahlen) haben die Form

$$x = \sigma \cdot \left(\sum_{\nu=1}^{l} x_\nu b^{-\nu} \right) \cdot b^n, \tag{1.1}$$

mit b = Basis (fest), l = Mantissenlänge (fest), σ = Vorzeichen, n = Exponent und $m := (x_1, \ldots, x_l) \in \{0, \ldots, b-1\}^l$ = Mantisse.

1. **Möglichkeit: Festkommazahlen**

   Dabei wird $n \in \mathbb{N}$ fest gewählt und $x_1 = 0$ wird zugelassen in (1.1).

   Beispiel: für $b = 10$, $l = 7$ und $n = 3$ hat $x = 25.73$ die Darstellung $025,7300$ d.h. $x_1 = 0$, $x_2 = 2$, $x_3 = 5$, $x_4 = 7$, $x_5 = 3$, $x_6 = 0$, $x_7 = 0$ und $\sigma = +1$.

   Die Menge der Festkommazahlen ist endlich (Anzahl $= 2 \cdot b^l - 1$), die kleinste darstellbare positive Zahl ist $x_{\min} = b^{n-l}$ und die größte ist $x_{\max} = (1 - b^{-l}) \cdot b^n \approx b^n$.
   Sie sind *gleichabständig* mit Abstand b^{n-l} im Intervall $[-x_{\max}, x_{\max}]$. Die Speicherung einer Zahl erfordert, bei $b = 2$, $l+1$ Bits. l kann also nicht beliebig groß gewählt werden. Daher wird, wenn wir n "klein" wählen, $x_{\max}$ "klein", wenn wir n groß wählen, wird der Abstand zwischen den Zahlen relativ groß. $\Rightarrow$ wenig geeignet, um $\mathbb{R}$ zu repräsentieren.

2. **Möglichkeit: Gleitkommazahlen**

   Es werden $n_-, n_+ \in \mathbb{Z}, n_- < n_+, l \in \mathbb{N}, b \in N, b \geq 2$ fest gewählt. Die darstellbaren Zahlen sind von der Form (1.1) mit

   $$\sigma \in \{-1, 1\}, n_- \leq n \leq n_+,$$

   d.h. n ist *nicht* fest, sondern ergibt sich aus der Forderung $|x| \in [b^{n-1}, b^n)$. Außerdem ist

   $$x_0 \neq 0 \quad \text{und} \quad x_\nu \in \{0, \ldots, b-1\} \ \forall\ \nu = 1, \ldots, l.$$

Beispiel: Für $b = 10$ und $l = 5$ hat $x = 23,45$ die Darstellung $0,2345 \cdot 10^2$, d.h. $n = 2$, $x_1 = 2$, $x_2 = 3$, ...

Die Werte für n_-, n_+, l hängen von Hardware und Software (Compiler) ab. Typische Werte sind:

					Speicherbedarf (bits)			
	b	l	n_-	n_+	Mantisse	VZ	Exp	$\sum$
single prec.	2	23	-125	128	23	1	8	32
double prec.	2	52	-1021	1024	52	1	11	64

Damit ergibt sich:

$x_{\min}$	$x_{\max}$	τ
$2^{-126} \approx 1,2 \cdot 10^{-38}$	$\approx 2^{128} \approx 3,4 \cdot 10^{38}$	$2^{-23} \approx 1,2 \cdot 10^{-7}$
$2^{-1022} \approx 2,2 \cdot 10^{-308}$	$\approx 2^{1024} \approx 1,8 \cdot 10^{308}$	$2^{-52} \approx 2,2 \cdot 10^{-16}$

Es ist $x_{\max} = (1 - b^{-l}) \cdot b^{n_+} \approx b^{n_+}, x_{\min} = b^{n_- - 1}$
Die "freien Exponenten" werden zur Speicherung der 0 und des *Über-/Unterlaufs* (siehe unten) ("NaN" = "not a number") verwendet, der bei Rechenoperationen auftreten kann. Die Gleitkommazahlen sind *nicht* äquidistant, für $x \in [b^{n-1}, b^n)$ ist der Abstand benachbarter Maschinenzahlen $\Delta x = b^{n-l}$. Der relative Abstand dagegen ist über den gesamten Bereich $[-x_{max}, -x_{\min}] \cup [x_{\min}, x_{\max}]$ nahezu konstant:

$$b^{-l} = \frac{b^{n-l}}{b^n} \leq \frac{\Delta x}{|x|} \leq \frac{b^{n-l}}{b^{n-1}} = b^{1-l}$$

Zahlen $x \in \mathbb{R}$ müssen durch *Runden* oder durch Abschneiden (der b-adischen Entwicklung) in eine Maschinenzahl umgewandelt werden. Der dabei begangene relative Fehler ist also

$$\left|\frac{\mathrm{rd}(x) - x}{x}\right| \leq \frac{1}{2} \cdot b^{1-l} \quad \text{bzw.} \quad \left|\frac{\mathrm{chop}(x) - x}{x}\right| \leq b^{1-l};$$

die absoluten Fehler erfüllen

$$|\mathrm{rd} - x| \leq \frac{1}{2} \cdot b^{1-l} \cdot |x| \quad \text{bzw.} \quad |\mathrm{chop}(x) - x| \leq b^{1-l} \cdot |x|.$$

Definition 1.1 *Die Größe $\tau := 0,5 \cdot b^{1-l}$ heißt relative Maschinengenauigkeit der l-stelligen Gleitkommaarithmetik. Sie stellt eine Schranke (und eine gute Näherung) für den relativen Fehler, der bei der Darstellung von reellen Zahlen als Gleitkommazahlen auftritt ("relativer Darstellungsfehler"), dar: $\left|\frac{\Delta x}{x}\right| \leq \tau$.*

Tritt bei einer Berechnung ein Wert x mit $|x| > x_{\max}$ bzw. $|x| < x_{\min}$ auf, ("Overflow" bzw. "Underflow"), so liefern die meisten Compiler das Ergebnis "NaN". Die Rückgabe $x = 0$ im Falle eines Underflows ist oft nicht sinnvoll!

Beispiel: (für einen Underflow)

Umwandlung von kartesischen Koordinaten (x, y) in Polarkoordinaten (r, ϕ);

$$r := \sqrt{x^2 + y^2}, \quad \text{ferner seien} \quad x_{\min} = 10^{-7},\ x = 10^{-5} \text{ und } y = 0,5 \cdot 10^{-5}.$$

Es ist $x \cdot x = 10^{-10} < x_{\min} \Rightarrow$ die nächste Maschinenzahl wäre $\mathrm{rd}(x \cdot x) = 0$, analog $\mathrm{rd}(y \cdot y) = 0 \Rightarrow r := \sqrt{0 + 0} = 0$ statt korrekt $r = 1,118... \cdot 10^{-5}$

Obwohl *Ein- und Ausgabe* im darstellbaren Bereich liegen, liefert die Berechnung wegen Darstellungsfehlern von Zwischenergebnissen falsches Ergebnis; die Rückgabe "NaN" wäre hier sinnvolles Compilerverhalten.

Übrigens: Man kann Underflow in diesem Fall durch Skalierung umgehen: berechne

$$z := \max\{|x|, |y|\}, \quad \tilde{x} = \frac{x}{z}, \quad \tilde{y} = \frac{y}{z} \quad \text{und} \quad r := z \cdot \sqrt{\tilde{x}^2 + \tilde{y}^2}.$$

Man erhält also

$$z = 10^{-5}, \quad \tilde{x} = 1, \quad \tilde{y} = 0,5 \quad \Rightarrow \quad \tilde{r} = \sqrt{\tilde{x}^2 + \tilde{y}^2} = 1,118$$

und damit $r = z\,\tilde{r} = 1,118 \cdot 10^{-5}$, d. h. kein Zwischenergebnis verlässt den darstellbaren Bereich $[x_{\min}, x_{\max}]$.

Bemerkung:

- Die bessere Repräsentation von $\mathbb{R}$ durch Gleitkommazahlen, im Vergleich zu Festkommazahlen (größerer darstellbarer Bereich, nahezu konstante relative Darstellungsfehler), erkauft man sich durch etwas aufwändigere Implementierung von Rechenoperationen.
- Sowohl bei Fest- als auch Gleitkommazahlen sind abweichende Speicherungen möglich("Speicherung im Zweierkomplement", siehe Informatik-Vorlesungen) zur Vereinfachung der Implementierung der Rechenoperationen.
- Von nun an betrachten wir nur noch Gleitkommadarstellung.

1.2 Grundarithmetik

Als *Grundarithmetik* werden die vier Grundrechenarten $+, -, \cdot, :$ sowie die Potenzbildung x^y und einfache Funktionsauswertungen wie $\exp, \log, \sin, \cos, \arctan, \sqrt{}$ bezeichnet.

Schon in 1.1 wurde angedeutet: Die Grundarithmetik kann den Bereich

$$[-x_{\max}, x_{\min}] \cup [x_{\min}, x_{\max}]$$

der darstellbaren Zahlen verlassen (z.B.: $x, y \leq x_{\max}, x \cdot y > x_{\max}$) $\Rightarrow$ Overflow/Underflow

Selbst wenn dies *nicht* passiert, so ist doch das Ergebnis im Allgemeinen keine darstellbare Zahl, sondern muss gerundet werden. Dabei entsteht, selbst wenn die Implementierung der Arithmetik "optimal" ist, ein relativer Darstellungsfehler von $\approx \tau$. In der Tat liefern die meisten Sprachen/Compiler eine Grundarithmetik mit relativem Fehler $\leq \tau$, d.h. statt der

Programmzeile	wird die *Ersatzarithmetik* ausgeführt (mit $\lvert\tau_z\rvert \leq \tau$).
$z := x + y$	$z := x \tilde{+} y := \mathrm{rd}(x+y) = x + y + (x+y) \cdot \tau_z$
$z := \exp(x)$	$z := \tilde{\exp}(x) := \mathrm{rd}(\exp(x)) = \exp(x) + \exp(x) \cdot \tau_z$
mit relativen Fehlern	$\frac{x \tilde{+} y - (x+y)}{x+y} = \tau_z$ bzw. $\frac{\tilde{\exp}(x) - \exp(x)}{\exp(x)} = \tau_z$

Bemerkung: τ ist die kleinste positive Zahl, so dass $\tau + 1$ nicht durch Rundung zu 1 wird. Damit lässt sich τ größenordnungsmäßig zum Beispiel bestimmen durch:

```
τ := 1.0;
while 1+τ > 1 do tau := τ/2
```

Vorsicht: Die Ersatzarithmetik ist nicht immer assoziativ:

$$\begin{aligned} (10^{-20} \tilde{+} 1) \tilde{+} (-1) &= 1 \tilde{+} (-1) = 0, \\ 10^{-20} \tilde{+} (1 \tilde{+} (-1)) &= 10^{-20} \tilde{+} 0 = 10^{-20}. \end{aligned}$$

1.3 Fehlerfortpflanzung in der Grundarithmetik

Modell

$$\text{Eingabeparameter } x \in \mathbb{R}^n \rightarrow \boxed{\text{numerische Berechnung}} \rightarrow \text{Ausgabe } y = f(x) \in \mathbb{R}^m$$

Die Eingabe x ist im Allgemeinen mit Fehlern behaftet (Darstellungsfehler, Messfehler bei Messwerten, Rundungsfehler falls x zuvor berechnet wurde). Also wird tatsächlich

$$x + \Delta x \rightarrow \boxed{\text{numerische Berechnung } y = f(x)} \rightarrow y + \Delta y$$

berechnet. Wie hängt der Ausgabefehler Δy von Δx und (x, f) ab? Unter der Annahme, dass f glatt und Δx "klein" ist, folgt mit der TAYLOR-Entwicklung:

$$f_i(x + \Delta x) = f_i(x) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x) \;\; \Delta x_j + O(||\Delta x||^2).$$

Wenn Δx "klein" ist, können wir als absoluten Fehler des Ergebnisses $y_i,\ i = 1, \dots n$,

$$\Delta y_i := \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x)\Delta x_j = \left(\frac{\partial f_i}{\partial x}, \Delta x\right),$$

also $\Delta y := \mathrm{Jf}(x)\ \Delta x$, $\mathrm{Jf}(x)$ = Jacobi-Matrix, erwarten. Außerdem tritt noch der absolute Darstellungsfehler $f_i(x) \cdot \tau_f,\ |\tau_f| \leq \tau$ auf. Wir definieren daher:

Definition 1.2

$$\Delta f := Jf(x)\Delta x + \overbrace{diag(\tau_1^f, \dots, \tau_m^f)}^{Diagonalmatrix} f(x)$$

heißt differentieller Fehler *der Abbildung $f : \mathbb{R}^n \to \mathbb{R}^m$. Er besteht aus der Fortpflanzung des Eingabefehlers Δx, der über die Matrix $Jf(x)$[bzw. $\nabla f(x)$ im Fall $m = 1$, bzw. f' im Fall $m = n = 1$] verstärkt wird, und dem absoluten Darstellungsfehler $diag(\tau^f) \cdot f(x),\ |\tau_j^f| \leq \tau,\ j = 1, \dots, m$.*
Der differentielle Fehler ist eine (i.A. sehr gute) Näherung *für den wirklich auftretenden Fehler. Man nennt die Verstärkungsfaktoren $\frac{\partial f_i}{\partial x_j}(x)$ des absoluten Fehlers auch* absolute Konditionszahlen.

Beispiel 1.3 $z := \exp(x) =: f(x)$

$$\Delta z = f'(x) \cdot \Delta x + \tau_z \cdot z = e^x \Delta x + e^x \tau_z$$

ist der differentielle Fehler, e^x die absolute Konditionszahl und

$$\frac{\Delta z}{z} = \Delta x + \tau_z = x \cdot \frac{\Delta x}{x} + \tau_z$$

der relativer Fehler. Also: Verstärkung des relativen Fehlers um den Faktor x, ein bisschen kritisch für $x \gg 1$ (allerdings tritt bei sehr großem x für $f = \exp$ ohnehin Overflow auf).

Allgemein gilt für $f : \mathbb{R} \to \mathbb{R}$ $\quad \Delta f = f'(x) \cdot \Delta x + \tau_z \cdot f(x)$:

$$\frac{\Delta f}{f} = \frac{f'(x) \cdot x}{f(x)} \cdot \frac{\Delta x}{x} + \tau_z,$$

d.h. der Verstärkungsfaktor des relativen Fehlers ist: $\frac{f'(x) \cdot x}{f(x)}$.

Beispiel 1.4 $z := \ln(x)$:

$$\frac{\Delta f}{f} = \frac{\frac{1}{x} \cdot x}{\ln(x)} \cdot \frac{\Delta x}{x} + \tau_z$$

d.h. Verstärkungsfaktor des relativen Fehlers: $\frac{1}{\ln x}$, kritisch für $x \approx 1$.

Beispiel 1.5 $z := x \cdot y = f(x, y)$:

$$\Delta z = \underbrace{\frac{\partial f}{\partial x}}_{=y} \cdot \Delta x + \underbrace{\frac{\partial f}{\partial y}}_{=x} \cdot \Delta y + \tau_z \cdot xy = y \cdot \Delta x + x \cdot \Delta y + \tau_z \cdot xy$$

$$\Rightarrow \frac{\Delta z}{z} = \frac{\Delta x}{x} + \frac{\Delta y}{y} + \tau_z,$$

d. h. relative Fehler addieren sich höchstens $\rightsquigarrow$ harmlos.

Beispiel 1.6 $z := x + y = f(x, y)$:

$$\begin{aligned} \Delta z &= 1 \cdot \Delta x + 1 \cdot \Delta y + \tau_z \cdot (x + y) \qquad (\rightsquigarrow \textit{absoluter Fehler harmlos}) \\ \frac{\Delta z}{z} &= \frac{x}{x+y} \cdot \frac{\Delta x}{x} + \frac{y}{x+y} \cdot \frac{\Delta y}{y} + \tau_z \end{aligned}$$

Haben x und y das gleiche Vorzeichen, so folgt

$$\left|\frac{x}{x+y}\right| \leq 1, \quad \left|\frac{y}{x+y}\right| \leq 1 \qquad (\rightsquigarrow \textit{ harmlos})$$

Doch Vorsicht: *Für $x \approx -y$ ist der Verstärkungsfaktor des relativen Fehlers unbeschränkt ("Auslöschung")!*

Beispiel 1.7 $z := \frac{x}{y}$:

$$\begin{aligned} \Delta z &= \frac{1}{y} \cdot \Delta x - \frac{x}{y^2} \Delta y + \tau_z \cdot z \\ &\Rightarrow \quad \frac{\Delta z}{z} = \frac{\Delta x}{x} - \frac{\Delta y}{y} + \tau_z \quad (\rightarrow \textit{ harmlos}) \end{aligned}$$

Beispiel 1.8 *(s. Übung):* $z := \sin x$, $z := \cos x$; *kritisch für $z \approx 0$ sowie für $|x| \gg 1$.*

Man sagt, die Berechnung der Differenz von in etwa gleich großen Zahlen ist "*schlecht konditioniert*" (in dem Sinne, dass die Verstärkungsfaktoren des relativen Fehlers sehr groß sind). Man spricht hier auch von "Auslöschung (von signifikanten Ziffern)", was anschaulich folgenden Vorgang beschreibt:

```
  0.1234567...   (1)
 -0.1234556...   (2)
 ================
  0.0000011...   (3)
```

In (1) und (2) sind es noch je 7 signifikante Ziffern, in (3) nur noch 2 signifikante Ziffern! Eine solche Differenzbildung innerhalb von Algorithmen sollte, wenn möglich, vermieden werden (siehe Kapitel 1.4, sowie Beispiel 0.2).

1.4 Differentielle Fehleranalyse

Wir betrachten nun einen Algorithmus, der aus *mehreren* Schritten der Grundarithmetik aufgebaut ist. In jedem Rechenschritt entstehen Fehler und werden Fehler weitergegeben. Wie analysiert man den Fehler des Gesamtalgorithmus?

Definition 1.9 *Ein Algorithmus, der Eingabedaten $x \in \mathbb{R}^n$ Ausgabedaten $y = f(x) \in \mathbb{R}^m$ zuordnet, ist eine Komposition $f = f^{(k)} \circ \ldots \circ f^{(1)}$ von elementaren Rechenschritten der Grundarithmetik:*

$$\texttt{input } x =: x^{(0)} \in \mathbb{R}^n \overset{f^{(1)}}{\rightarrow} x^{(1)} \in \mathbb{R}^{n_1} \overset{f^{(2)}}{\rightarrow} x^{(2)} \in \mathbb{R}^{n_2} \rightarrow \ldots \rightarrow x^{(k)} = y \in \mathbb{R}^m \texttt{ output}$$

Die differentielle Fehleranalyse besteht in der Analyse jedes einzelnen Teilschritts gemäß Kapitel 1.3.

Beispiel 1.10 $y = f(x) = \sqrt{x+1} - \sqrt{x-1}$ *(zu erwarten: "Auslöschung" für $x \gg 1$!)*

$$\begin{aligned}
x^{(0)} &:= x \\
f^{(1)}(x^{(0)}) &:= \begin{pmatrix} x^{(0)}+1 \\ x^{(0)}-1 \end{pmatrix} =: \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \end{pmatrix}, \quad f^{(1)} : \mathbb{R} \to \mathbb{R}^2 \\
f^{(2)}(x_1^{(1)}, x_2^{(1)}) &:= \begin{pmatrix} \sqrt{x_1^{(1)}} \\ \sqrt{x_2^{(1)}} \end{pmatrix} =: \begin{pmatrix} x_1^{(2)} \\ x_2^{(2)} \end{pmatrix}, \quad f^{(2)} : \mathbb{R}_+^2 \to \mathbb{R}_+^2 \\
f^{(3)}(x_1^{(2)}, x_2^{(2)}) &:= x_1^{(2)} - x_2^{(2)} =: x^{(3)}, \quad f^{(3)} : \mathbb{R}^2 \to \mathbb{R}
\end{aligned}$$

$$\begin{aligned}
\Delta x^{(0)} &= \Delta x \quad (Eingabefehler) \\
\Delta x_1^{(1)} &= 1 \cdot \Delta x^{(0)} + \tau_1^{(1)} x_1^{(1)} \\
\Delta x_2^{(1)} &= 1 \cdot \Delta x^{(0)} + \tau_2^{(1)} x_2^{(1)} \\
\Delta x_1^{(2)} &= \frac{1}{2\sqrt{x_1^{(1)}}} \cdot \Delta x_1^{(1)} + \tau_1^{(2)} x_1^{(2)} = \frac{\Delta x + \tau_1^{(2)}(x+1)}{2\sqrt{x+1}} + \tau_1^{(2)}\sqrt{x+1} \\
\Delta x_2^{(2)} &= \frac{1}{2\sqrt{x_1^{(2)}}} \cdot \Delta x_2^{(1)} + \tau_2^{(2)} x_2^{(2)} = \frac{\Delta x + \tau_2^{(1)}(x-1)}{2\sqrt{x-1}} + \tau_2^{(2)}\sqrt{x-1}
\end{aligned}$$

$$\begin{aligned}
\Delta y &= \Delta x^{(3)} = 1 \cdot \Delta x_1^{(2)} - 1 \cdot \Delta x_2^{(2)} + \tau^{(3)} x^{(3)} \\
&= \frac{\Delta x}{2}\left(\frac{1}{\sqrt{x+1}} - \frac{1}{\sqrt{x-1}}\right) + \left(\frac{\tau_1^{(1)}}{2} + \tau_1^{(2)}\right)\sqrt{x+1} - \left(\frac{\tau_2^{(1)}}{2} + \tau_2^{(2)}\right)\sqrt{x-1} + \tau^{(3)} y
\end{aligned}$$

Mit

$$\begin{aligned}
\frac{1}{\sqrt{x+1}} - \frac{1}{\sqrt{x-1}} &= \frac{\sqrt{x-1} - \sqrt{x+1}}{\sqrt{x+1}\sqrt{x-1}} \qquad und \\
\frac{\sqrt{x \pm 1}}{\sqrt{x+1} - \sqrt{x-1}} &= \frac{\sqrt{x \pm 1}(\sqrt{x+1} + \sqrt{x-1})}{(x+1) - (x-1)} = \frac{1}{2}(x \pm 1 + \sqrt{x^2-1})
\end{aligned}$$

folgt

$$\begin{aligned}
\frac{\Delta y}{y} &= -\frac{\Delta x}{2}\frac{1}{\sqrt{x+1}\sqrt{x-1}} + \left(\frac{\tau_1^{(1)}}{2} + \tau_1^{(2)}\right)\frac{1}{2}(x+1+\sqrt{x^2-1}) \\
&\quad + \left(\frac{\tau_2^{(1)}}{2} + \tau_2^{(2)}\right)\frac{1}{2}(x-1+\sqrt{x^2-1}) + \tau^{(3)} \qquad (1.2)
\end{aligned}$$

Für den Fall $x \gg 1$ können wir die Approximationen $\sqrt{x^2+1} \approx x$, $x \pm 1 \approx x$ verwenden und bekommen:

$$\frac{\Delta y}{y} \approx -\frac{\Delta x}{2x} + \left(\frac{\tau_1^{(1)}}{2} + \tau_1^{(2)}\right) x - \left(\frac{\tau_2^{(1)}}{2} + \tau_2^{(2)}\right) x + \tau^{(3)},$$

also mit $|\tau_j^{(i)}| \le \tau$ *:*

$$\left|\frac{\Delta y}{y}\right| \lesssim \underbrace{\frac{1}{2}\left|\frac{\Delta x}{x}\right|}_{harmlos} + \underbrace{(3x+1)\,\tau}_{kann\ groß\ sein}.$$

Beispiel: sei $x = 1.234 \cdot 10^{14}$ *und* $\tau = 10^{-16}$, $\frac{\Delta x}{x} \approx \tau$. *Aus* $y_{exakt} = 9.002 \cdot 10^{-8}$ *und* $y_{Rechner} = 8.941 \cdot 10^{-8}$ *folgt:*

$$\frac{y_{exakt} - y_{Rechner}}{y_{exakt}} \approx 7 \cdot 10^{-3}$$

berechnete Schranke $(3x + \frac{3}{2}) \cdot \tau \approx 3.7 \cdot 10^{-2}$ *(passt!)*

Anmerkung: Viel besser geeeignet ist die Implementierung

$$f(x) = \sqrt{x+1} - \sqrt{x-1} = \frac{2}{\sqrt{x+1} + \sqrt{x-1}}$$ *(keine Auslöschung zu erwarten, siehe Übung)*

Formalisierung der Differentiellen Fehleranalyse:

$$x = x^{(0)} \xrightarrow{f^{(1)}} x^{(1)} \xrightarrow{f^{(2)}} x^{(2)} \to \ldots \xrightarrow{f^{(k)}} x^{(k)} = y$$

$$\begin{aligned}
\Delta x^{(1)} &= Jf^{(1)}(x^{(0)})\Delta x^{(0)} + T^{(1)}x^{(1)} \qquad \text{wobei } T^{(i)} := \mathrm{diag}(\tau_j^{(i)}) \text{ Diagonalmatrix} \\
\Delta x^{(2)} &= Jf^{(2)}(x^{(1)})\Delta x^{(1)} + T^{(2)}x^{(2)} \\
&= Jf^{(2)}(x^{(1)})Jf^{(1)}(x^{(0)})\Delta x + Jf^{(2)}(x^{(1)})T^{(1)}x + T^{(2)}x^{(2)} \\
&\vdots \\
\Delta x^{(k)} &= \overbrace{Jf^{(k)}(x^{(k-1)}) \cdot \ldots \cdot Jf^{(1)}(x^{(0)})}^{=Jf(x)}\Delta x \qquad (1.3) \\
&\quad + Jf^{(k)}(x^{(k-1)}) \cdot \ldots \cdot Jf^{(2)}(x^{(1)})T^{(1)}x^{(1)} \qquad (1.4) \\
&\quad + \ldots \\
&\quad + Jf^{(k)}(x^{(k-1)}) \cdot T^{(k-1)}x^{(k-1)} \\
&\quad + T^{(k)}\underbrace{x^{(k)}}_{=y}
\end{aligned}$$

(1.3): fortgepflanzter Eingangsfehler (unvermeidbar, egal wie f in $f^{(i)}$ zerlegt wird)
(1.4): ab hier: fortgepflanzter Darstellungsfehler der Zwischenschritte

Falls man alle Rundungsfehler = Darstellungsfehler vernachlässigt, bekommt man

$$\begin{aligned}
\Delta y &= Jf^{(k)}(x^{(k-1)}) \cdot \ldots \cdot Jf^{(1)}(x^{(0)})\Delta x = Jf(x)\Delta x, \\
\text{also} \quad \Delta y_i &= \sum_j \frac{\partial f_i(x)}{\partial x_j}\Delta x_j, \\
\text{und} \quad \frac{\Delta y_i}{y_i} &= \sum_j \frac{\partial f_i(x)}{\partial x_j} \cdot \frac{x_j}{y_i} \cdot \frac{\Delta x_j}{x_j}.
\end{aligned}$$

Definition 1.11 *Die Faktoren* $\frac{\partial f_i(x)}{\partial x_j}$ *bzw.* $\frac{\partial f_i(x)}{\partial x_j} \cdot \frac{x_j}{y_i}$ *heißen* absolute bzw. relative Konditionszahlen *eines Algorithmus* $f = f^{(k)} \circ \ldots \circ f^{(1)}$.

Bemerkung 1.12 *Die Konditionszahlen sagen nur etwas über die Verstärkung von Eingangsfehlern aus, nicht aber über die entstehenden Rundungsfehler in den Teilalgorithmen; die Konditionszahlen sind sogar unabhängig von den Teilalgorithmen* $f^{(j)}$*; bei gegebenem* f *ist der fortgepflanzte Eingangsfehler unvermeidbar. Ein Algorithmus ist nur dann " gutmütig", wenn jeder Teilalgorithmus* $f^{(j)}$ *gut konditioniert ist, denn nur dann weiß man sicher, dass die fortgepflanzten Darstellungsfehler der Zwischenschritte klein gegenüber dem (unvermeidlichen) fortgepflanzten Eingangsfehler sind.*

Im Beispiel

$$f(x) = \underbrace{\sqrt{x+1} - \sqrt{x-1}}_{(1)} = \underbrace{\frac{2}{\sqrt{x+1} + \sqrt{x-1}}}_{(2)}$$

hat f (natürlich gleichermaßen in beiden Darstellungen (1) und (2)) eine kleine relative Konditionszahl

$$|\kappa_{\text{rel}}| \overset{(1.2)}{=} \frac{x}{2\sqrt{x+1}\sqrt{x-1}} \approx \frac{1}{2}.$$

Die Darstellung (1) führt zu starken Rundungsfehlern, die Darstellung (2) wegen Verwendung unterschiedlich gut konditionierter Teilalgorithmen hingegen nicht.

Bemerkung: Wie passt sich Beispiel 0.2 ein? Wieso hängt hier die Fehlerverstärkung offenbar doch von der Darstellung ab?

$$y = \underbrace{\frac{100}{50\sqrt{2} - 71}}_{(1)} = \underbrace{-\frac{5000\sqrt{2} + 7100}{41}}_{(2)}$$

Die *zwei verschiedenen* Funktionen

$$f(x) := \frac{100}{50x - 71} \text{ und } g(x) := -\frac{5000x + 7100}{41}$$

werden an der Stelle $\sqrt{2}$ ausgewertet, und die Verstärkung des Eingangsfehlers beträgt:

$$\begin{array}{lcccl} \text{bei (1)} & : & \left|\frac{df}{dx}(\sqrt{2})\right| & = & \frac{5000}{(50\sqrt{2}-71)^2} \approx 59732 \\ \text{bei (2)} & : & \left|\frac{dg}{dx}(\sqrt{2})\right| & = & \frac{5000}{41} \approx 122 \end{array}$$

Unterschiedliche Funktionen f und g können natürlich *unterschiedliche* Konditionszahlen haben.

Neben der bisherigen Betrachtungsweise, bei der die Kondition im allgemeinen Fall $f : \mathbb{R}^n \to \mathbb{R}^m$ durch eine *Matrix* gegeben ist, ist auch noch die normweise Konditionsanalyse möglich (vgl. auch 2. Semester für lineare Gleichungssysteme, Kapitel 6.1).

Betrachte wieder die Linearisierung:

$$\Delta f = \frac{df}{dx} \Delta x$$

sowie eine Norm $\|\cdot\|_1$ im $\mathbb{R}^n$ und eine Norm $\|\cdot\|_2$ im $\mathbb{R}^m$ und eine Matrix-Norm $\|\cdot\|$, die zu diesen Normen verträglich ist (d.h. $||Ax||_2 \leq ||A|| \cdot ||x||_1$), z.B. die Operator-Norm

$$||A|| := \sup_{x \neq 0} \frac{||Ax||_2}{||x||_1}$$

(dh. die $\|\cdot\|_1$ und $\|\cdot\|_2$ zugeordnete Norm, vgl. 2.Sem., Satz 5.19)

$$\begin{aligned} \Rightarrow \quad ||\Delta f||_2 &\leq ||\frac{df}{dx}||\ ||\Delta x||_1 \\ \frac{||\Delta f||_2}{||f||_2} &\leq ||\frac{df}{dx}||\frac{||x||_1}{||f||_2}\ \frac{||\Delta x||_1}{||x||_1} \end{aligned}$$

$\kappa_{\text{abs}} := ||\frac{df}{dx}||$, $\kappa_{\text{rel}} := \kappa_{\text{abs}} \frac{||x||_1}{||f||_2}$ heißen *absolute/ relative Konditionszahlen* bzgl. der Normen $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|$. Im Allgemeinen liefert die normweise Konditionsanalyse jedoch ungenauere Ergebnisse als die komponentenweise. Für die Multiplikation ergibt sich zum Beispiel:

$$f : \mathbb{R}^2 \to \mathbb{R},\ \begin{pmatrix} a \\ b \end{pmatrix} \to a \cdot b : \qquad Jf = (b, a).$$

Komponentenweise Konditionsanalyse (s. Kapitel 1.3) :

$$\Delta f = b \cdot \Delta a + a \cdot \Delta b \quad \rightsquigarrow \quad \frac{\Delta f}{f} = 1 \cdot \frac{\Delta a}{a} + 1 \cdot \frac{\Delta b}{b},$$

d. h. beide relative Konditionszahlen sind $= 1$, also harmlos. Normweise Konditionsanalyse, beispielsweise mit der 1-Norm (die zugehörige Matrix-Norm ist die Spaltensummennorm (s. 2. Sem., Satz 5.25)) liefert:

$$||Jf|| = \max\{|b|, |a|\} = \kappa_{\text{abs}}$$

$$\kappa_{\text{rel}} = \kappa_{\text{abs}} \cdot \frac{||\binom{a}{b}||_1}{|ab|} = \max\{|a|, |b|\} \cdot \frac{|a| + |b|}{|ab|} \underset{\text{oBdA}|a| \geq |b|}{=} \frac{|a| + |b|}{|b|} = 1 + \frac{|a|}{|b|}$$

ist groß für $|a| \gg |b|$. Jedoch zeigte die komponentenweise Konditionsanalyse, dass auch für $|a| \gg |b|$ keine Probleme zu erwarten sind.

Zusammenhang zwischen dem Begriff "Kondition einer Matrix" aus L.A. II und der oben definierten relativen Kondition eines Algorithmus:

In beiden Fällen geht es um die Verstärkung von relativen Eingangsfehlern: In Satz 6.2, LA II, wurde für die Fehlerverstärkung beim Lösen von linearen Gleichungssystemen, also für die Abbildung (= den Algorithmus) $(A, b) \mapsto x = A^{-1}b$ die Abschätzung

$$\frac{||\Delta x||}{||x||} \leq ||A||\ ||A^{-1}|| \underbrace{\frac{1}{1 - ||A^{-1}||\ ||\Delta A||}}_{\approx 1} \cdot \left(\frac{||\Delta b||}{||b||} + \frac{||\Delta A||}{||A||} \right)$$

gezeigt. Für $||\Delta A|| \ll ||A^{-1}||$ ist also der Verstärkungsfaktor für relative Fehler in etwa beschränkt durch $||A||\ ||A^{-1}||$, daher wurde dort die Kondition einer Matrix

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

definiert. Die Kondition $\kappa(A)$ der Matrix A stimmt also für $||\Delta A||$ klein und unter der Annahme, dass obige Abschätzung "scharf" ist, mit der Kondition κ_{rel} des Algorithmus $(A, b) \mapsto A^{-1}b$ überein.

1.5 Anwendung der Differentiellen Fehleranalyse: Numerische Differentiation

Gegeben: $f \in \mathcal{C}^1(\mathbb{R}),\ x \in \mathbb{R}$; es soll $f'(x)$ näherungsweise berechnet werden durch

$$\frac{f(x+h)-f(x)}{h} =: g_h(x)$$

Numerisches Experiment: $f := \exp,\ x := 0 \quad \Rightarrow \quad f'(x) = 1$

h	$g_h(1)$
1	1.175201
10^{-1}	1.001667
10^{-2}	1.000017
10^{-3}	1.000017
10^{-4}	1.000166
10^{-5}	1.001358
10^{-6}	0.983477
10^{-7}	0.192093
10^{-8}	0.000000

Was ist passiert?

Für großes h ist der Verfahrensfehler, d.h. der Abstand von $f'(x)$ zu unserer Näherung $g_h(x)$, groß. Für kleines $h > 0$ tritt Auslöschung ein, da eine Differenz von ähnlich großen Zahlen gebildet wird!

Frage: Wie groß sollte man h im Allgemeinen wählen?

Zur Vereinfachung sei $f \in \mathcal{C}^3(\mathbb{R})$. Der Gesamtfehler setzt sich zusammen aus dem Verfahrensfehler

$$g_h - f'(x) = \frac{f(x+h)-f(x)}{h} - f'(x) = \frac{f'(x)h + f''(x)\frac{h^2}{2} + O(h^3)}{h} - f'(x) = \frac{h}{2}f''(x) + O(h^2)$$

und den Rundungsfehlern. Zu deren Bestimmung: Differentielle Fehleranalyse:

$$x \overset{g^{(1)}}{\to} \begin{pmatrix} f(x+h) \\ f(x) \end{pmatrix} \overset{g^{(2)}}{\to} f(x+h) - f(x) \overset{g^{(3)}}{\to} \frac{f(x+h)-f(x)}{h}$$

$$\begin{aligned}
\Delta x_1^{(1)} &= f'(x+h)\cdot\Delta x + \tau_1^{(1)} x_1^{(1)} = f'(x+h)\Delta x + \tau_1^{(1)} f(x+h) \\
\Delta x_2^{(1)} &= f'(x)\cdot\Delta x + \tau_2^{(1)} x_2^{(1)} = f'(x)\Delta x + \tau_2^{(1)} f(x) \\
\Delta x^{(2)} &= 1\cdot\Delta x_1^{(1)} - 1\cdot x_1^{(1)} + \tau^{(2)} x^{(2)} \\
&= [f'(x+h) - f'(x)]\Delta x + \tau_1^{(1)} f(x+h) - \tau_2^{(1)} f(x) + \tau^{(2)}\cdot[f(x+h)-f(x)] \\
\Delta x^{(3)} &= \frac{1}{h}\cdot\Delta x^{(2)} + \tau^{(3)} x^{(3)} \\
&= \underbrace{\frac{f'(x+h)-f'(x)}{h}}_{\approx f''(x)}\cdot\Delta x + \frac{\tau_1^{(1)}}{h} f(x+h) - \frac{\tau_2^{(1)}}{h} f(x) + [\tau^{(2)} + \tau^{(3)}] \underbrace{\frac{f(x+h)-f(x)}{h}}_{\approx f'(x)}
\end{aligned}$$

Annahme: Eingangsfehler $|\Delta x| \le \tau$ sowie relative Darstellungsfehler $|\tau_j^{(i)}| \le \tau$

$$\Rightarrow \quad |\text{Gesamtfehler}| \le \underbrace{\frac{h}{2}|f''(x)|}_{\text{Verfahrensfehler}} + \tau|f''(x)| + \frac{2\tau}{h}||f||_{L^\infty(U)} + 2\tau|f'(x)| =: \varphi(h)$$

wobei U eine Umgebung von x ist, so dass $x + h \in U$.

Minimieren des Fehlers (genauer, der Fehlerschranke $\varphi(h)$):

$$\varphi'(h) = \frac{1}{2}|f''(x)| - \frac{2\tau}{h^2}||f||_{L^\infty(U)} \stackrel{!}{=} 0$$

$$\Leftrightarrow \quad \boxed{h_{\text{opt}} = \sqrt{\tau} \cdot 2\sqrt{\frac{||f||_{L^\infty(U)}}{|f''(x)|}}}$$

Einsetzen von h_{opt} in $\varphi(h)$ liefert: Der absolute Gesamtfehler ist dann $\approx \sqrt{\tau}\cdot$const, d.h., bei $\tau \approx 10^{-16}$ wähle $h \approx 10^{-8}$ (sofern f und f'' moderate Werte annehmen). Der zu erwartende Fehler liegt dann auch etwa bei 10^{-8}. Besser geht's nicht! Das Problem ist schlecht konditioniert!

2 Direkte Verfahren, mit Schwerpunkt Orthogonalisierungsverfahren

2.1 Motivation/Wiederholung LR-Zerlegung

Ein Standard-Verfahren zum Lösen von LGS $Ax = b$ ist die LR-Zerlegung, siehe L.A.I, Kap. 2.4. Hier eine kurze Wiederholung. Ausgangspunkt ist das Gauß-Verfahren. Dabei wird durch sukzessive elementare Umformungen A auf obere Dreiecksgestalt $Rx = \tilde{b}$ gebracht. Die elementaren Umformungen können als Multiplikation des LGS mit gewissen Matrizen M_k beschrieben werden. Falls keine Pivotisierung (= keine Vertauschung von Spalten) nötig ist, haben alle M_k die Form

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \alpha_k & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix} \tag{2.1}$$

(α_k ist der Eintrag der j-ten Spalte und der i-ten Zeile, mit $j < i$, d.h. Multiplikation mit M_k von links entspricht Addition des α_k-fachen der j-ten Zeile zur i-ten Zeile)
Zur Elimination eines Eintrags von A ist jeweils *eine* solche Multiplikation nötig, insgesamt also $\frac{n}{2}(n-1) =: l$ viele. Der erforderliche Wert α_k zur Elimination eines Eintrags kann an der gerade aktuellen Matrix $M_{k-1} \cdot \ldots \cdot M_1 A$ abgelesen werden, wie aus der L.A.-Vorlesung bekannt ist:

$$\alpha_k = -\frac{a_{ij}^{(k)}}{a_{jj}^{(k)}},$$

wobei $a_{ij}^{(k)}$ der Eintrag an der Stelle (i,j) der Matrix $M_{k-1} \cdot \ldots \cdot M_1 A$ ist.

$$\begin{aligned} Ax &= b && \mid \cdot M_1 \\ M_1 Ax &= M_1 b && \mid \cdot M_2 \\ &\vdots \\ \overbrace{\underbrace{M_l \cdot \ldots \cdot M_1}_{=:L^{-1}} A}^{=:R} x &= \overbrace{\underbrace{M_l \cdot \ldots \cdot M_1}_{=:L^{-1}} b}^{=:\tilde{b}} \end{aligned}$$

Die M_n sind offensichtlich alle invertierbar mit

$$M_n^{-1} = \begin{pmatrix} 1 & & \\ & \ddots & \\ -\alpha_k & & 1 \end{pmatrix}$$

(wieder ist $-\alpha_k$ der Eintrag in der j-ten Spalte und i-ten Zeile) und sind wieder untere Dreiecksmatrizen, ebenso deren Produkt.

$$\Rightarrow \quad L := (M_l \cdot \ldots \cdot M_1)^{-1} = M_1^{-1} \cdot \ldots \cdot M_l^{-1}$$

ist untere Dreiecksmatrix von der Gestalt

$$L = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ * & & 1 \end{pmatrix},$$

wobei die $\frac{n}{2}(n-1)$ vielen Einträge unterhalb der Diagonalen die Werte $-\alpha_k$ sind, mit α_k aus den l Matrizen (2.1). Das heißt L kann leicht während des Gauß-Algorithmus explizit berechnet werden und zwar ohne Zusatzaufwand, da die α_k während des Gauß-Algorithmus ohnehin berechnet werden. Man erhält $L^{-1}A = R$, also $A = LR$. Der Aufwand beträgt $\doteq \frac{1}{3}n^3$ Additionen und Multiplikationen, falls A voll besetzt ist.
Bei der LR-Zerlegung kann sich die Konditionszahl der Matrix vergrößern:

Beispiel 2.1 *Die Matrix*

$$A = \begin{pmatrix} 1 & & & & 1 \\ & \ddots & & 0 & \vdots \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & -1 & & & \vdots \\ & & & & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}$$

hat (siehe Übung)

$$||A||_1 = ||A||_\infty = n, \quad ||A^{-1}||_1 = ||A^{-1}||_\infty = 1, \quad \kappa_1(A) = \kappa_\infty(A) = n$$

$Ax = b$ wird mittels LR-Zerlegung auf $Rx = \tilde{b}, \tilde{b} = L^{-1}b$, transformiert, wobei

$$R = \begin{pmatrix} 1 & & & & & 1 \\ & \ddots & & & & 2 \\ & & \ddots & 0 & & 4 \\ & 0 & & \ddots & & \vdots \\ & & & & 1 & 2^{n-2} \\ 0 & \dots & \dots & \dots & 0 & 2^{n-1} \end{pmatrix},$$

und es sind $||R||_1 = 2^n - 1$, $||R||_\infty = 2^{n-1}$; $||R^{-1}||_1 = 1$; $||R^{-1}||_\infty = \frac{3}{2}$, also

$$\kappa_1(R) = 2^n - 1; \quad \kappa_\infty(R) = 3 \cdot 2^{n-2}.$$

Eine große Konditionszahl von Matrizen sorgt für Fehlerverstärkung beim Lösen der zugehörigen linearen Gleichungssysteme (siehe Satz 6.2, 2.Semester).

Daher verfolgen wir eine neue Strategie: Bringe $Ax = b$ auf rechte obere Dreiecksform durch sukzessive Multiplikation mit gewissen *Orthogonalmatrizen*! Zwei Eigenschaften von Orthogonalmatrizen sind dabei wesentlich:

(1) Bei Multiplikation mit ihnen ändert sich die κ_2-Kondition einer Matrix nicht (siehe Lemma 2.2).

(2) Sie sind, wie schon die Matrizen M_k, die bei der LR-Zerlegung zum Einsatz kommen, ebenfalls leicht zu invertieren.

Wir wollen mit $\mathcal{O}^n \subset \mathbb{R}^{n\times n}$ die Menge der $n \times n$- Orthogonalmatrizen bezeichnen.

Lemma 2.2 *Sei $Q \in \mathcal{O}^n \subset \mathbb{R}^{n\times n}$ eine Orthogonalmatrix. Dann gilt:*

(a) $||Qx||_2 = ||x||_2 \ \forall x \in \mathbb{R}^n \quad (||\cdot||_2 = l_2\text{-Norm})$.

(b) $||Q||_2 = 1 \quad (||\cdot||_2 = \text{von } l_2 \text{ erzeugte Matrix-Norm})$.

(c) $||QA||_2 = ||A||_2 = ||AQ||_2 \ \forall A \in \mathbb{R}^{n\times n}$ *und*
$\kappa_2(QA) = \kappa_2(A) \ \forall$ *invertierbare Matrizen* $A \in \mathbb{R}^{n\times n}$.

Beweis:

(a) $||Qx||_2^2 = (Qx)^t Qx = x^t Q^t Qx = x^t Q^{-1} Qx = x^t x = ||x||_2^2$.

(b) $||Q||_2 \underset{\text{Def.}}{=} \sup\limits_{0\neq x\in\mathbb{R}^n} \frac{||Qx||_2}{||x||_2} \underset{\text{(a)}}{=} 1$.

(c)
$$||A||_2 \underset{\text{Def.}}{=} \sup_{0\neq x\in\mathbb{R}^n} \frac{||Ax||_2}{||x||_2} \underset{\text{(a)}}{=} \sup_x \frac{||QAx||_2}{||x||_2} \underset{\text{Def.}}{=} ||QA||_2 \tag{2.2}$$

$$\text{und } ||A||_2 = \sup_x \frac{||Ax||_2}{||x||_2} = \sup_x \frac{||AQQ^t x||_2}{||x||_2} = \sup_{\substack{0\neq y\in\mathbb{R}^n,\\ y:=Q^t x}} \frac{||AQy||_2}{||Qy||_2}$$

$$\underset{\text{(a)}}{=} \sup_y \frac{||AQy||_2}{||y||_2} = ||AQ||_2 \tag{2.3}$$

$$\Rightarrow \kappa_2(QA) = ||QA||_2||(QA)^{-1}||_2 = ||QA||_2||A^{-1}Q^t||_2 \underset{\substack{(2.2),\ (2.3),\\ Q,Q^t\in\mathcal{O}^n}}{=} ||A||_2||A^{-1}||_2 = \kappa_2(A)$$

□

Es gibt zwei verschiedene 'klassische' Verfahren, um ein lineares Gleichungssystem $Ax = b$ mittels Multiplikation von Orthogonalmatrizen auf Dreiecksgestalt zu bringen:

(1) Householder-Transformationen: Kap. 2.2

(2) Givens-Rotationen: Kap. 2.4

2.2 QR-Zerlegung durch HOUSEHOLDER-Transformation

Definition 2.3 *Für einen gegebenen Vektor $v \in \mathbb{R}^n,\ v \neq 0$, heißt die Matrix*

$$H = H(v) := Id - 2\frac{vv^t}{v^tv} = Id - 2\frac{vv^t}{||v||_2^2}$$

HOUSEHOLDER*-Matrix. Für späteren Gebrauch wollen wir außerdem $H(0) := Id$ setzen.*

Eigenschaften (für $v \neq 0$):

$$H(v)v = v - 2\frac{vv^tv}{v^tv} = v - 2v = -v.$$

Für alle $w \perp v$ ist

$$H(v)w = w - 2\,\frac{v\overbrace{\langle v, w\rangle}^{=0}}{||v||_2^2} = w$$

Geometrische Interpretation : Die $n-1$-dimensionale "Ebene", die orthogonal zu v ist, wird durch H fix gelassen; der Normalenverktor v wird an der Ebene gespiegelt. Durch Darstellung eines beliebigen Vektors bezüglich einer Orthogonalbasis, der v angehört folgt: H **stellt eine Spiegelung an der zu v orthogonalen Ebene dar.**

Ferner:

$$\begin{array}{rrcll} & H^2 & = & Id & \text{(geometrisch klar)} \\ & H^t & = & H & \text{(trivial)} \quad \Rightarrow \quad H \text{ ist symmetrisch} \\ \Rightarrow & H^tH = H^2 & = & Id & \Rightarrow \quad H \text{ ist Orthogonalmatrix} \end{array}$$

Aus einem gegebenen Vektor $a \in \mathbb{R}^n$ und einem Index $j \in \{1, \ldots, n\}$ definieren wir einen Vektor $v \in \mathbb{R}^n$ wie folgt:

$$c = c(j,a) := \text{sign}(a_j)\sqrt{a_j^2 + \ldots + a_n^2}, \quad \text{dabei sei sign}(x) := \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

$$v = v(j,a) := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_j + c \\ a_{j+1} \\ \vdots \\ a_n \end{pmatrix} \in \mathbb{R}^n \quad \text{und} \quad H = H(v(j,a)) = \begin{cases} Id - 2\frac{v(j,a)v(j,a)^t}{||v(j,a)||_2^2}, & v \neq 0 \\ Id & , v = 0 \end{cases} \tag{2.4}$$

Lemma 2.4 *Die durch (2.4) definierte* HOUSEHOLDER*-Matrix $H = H(v(j,a))$ hat folgende Eigenschaften:*

(a) $Ha = \begin{pmatrix} a_1 \\ \vdots \\ a_{j-1} \\ -c \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

(b) Jeder Vektor der Form $w = \begin{pmatrix} w_1 \\ \vdots \\ w_{j-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ *wird von* H *invariant gelassen.*

(c) Für alle $b \in \mathbb{R}^n$ *ist* $(Hb)_k = b_k \ \forall k = 1, \ldots, j-1$.

Beweis:

(a) Sei $v \neq 0$. Es gilt

$$\begin{aligned} \frac{v^t a}{v^t v} &= \frac{(a_j + c)a_j + a_{j+1}^2 + \ldots + a_n^2}{(a_j + c)^2 + a_{j+1}^2 + \ldots + a_n^2} = \frac{ca_j + \sum_{i=j}^n a_i^2}{c^2 + 2ca_j + \sum_{i=j}^n a_i^2} = \frac{ca_j + \sum_{i=j}^n a_i^2}{2ca_j + 2\sum_{i=j}^n a_i^2} = \frac{1}{2} \\ &\Rightarrow \quad Ha = a - 2v \cdot \frac{1}{2} = a - v. \end{aligned}$$

Im Fall $v = 0$ ist $a_j + c = 0,\ a_{j+1} = \ldots = a_n = 0$. Da a_j und c das gleiche Vorzeichen haben, folgt $a_j = c = 0$;

$$\Rightarrow \begin{pmatrix} a_1 \\ \vdots \\ a_{j-1} \\ -c \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_{j-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{sowie} \quad H(v)a = \mathrm{Id}(a) = a = \begin{pmatrix} a_1 \\ \vdots \\ a_{j-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

beide Vektoren sind also gleich.

(b) Für $v = 0$ ist $H(v) = Id$; Eigenschaft (b) folgt trivialerweise. Für $v \neq 0$ ist $v^t w = 0 \Rightarrow Hw = w - \frac{2v}{||v||_2^2} \cdot v^t w = w$

(c) Für $v = 0$ ist $H(v) = Id$; Eigenschaft (c) folgt trivialerweise. Für $v \neq 0$ ist

$$Hb = b - \frac{2}{||v||_2^2} \underbrace{\begin{pmatrix} v_1 v_1 & \ldots & v_1 v_n \\ \vdots & & \vdots \\ v_n v_1 & \ldots & v_n v_n \end{pmatrix} \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}}_{(*)} \tag{2.5}$$

wobei $v_1 = \ldots = v_{j-1} = 0$ nach (2.4), d.h. die ersten $j-1$ Komponenten des Vektors $(*)$ sind null $\Rightarrow$ Behauptung. $\square$

Die Eigenschaften aus Lemma 2.4 werden ausgenutzt, um $Ax = b$ auf Dreiecksform zu bringen. Es wird eine Folge von HOUSEHOLDER-Transformationen durchgeführt:

$$\begin{array}{rcll} A^{(1)} & := & A,\ b^{(1)} := b & \\ A^{(1)}x & = & b^{(1)} & |H^{(1)}\cdot \\ & \downarrow & & \\ A^{(2)}x & = & b^{(2)} & |H^{(2)}\cdot \\ & \downarrow & & \\ & \vdots & & \\ & \downarrow & & \\ A^{(n)}x & = & b^{(n)} & \end{array}$$

wobei $A^{(n)} =: R$ rechte obere Dreicksform hat.

Um den ersten Transformationsschritt $A^{(1)}x = b^{(1)} \to A^{(2)}x = b^{(2)}$ durchzuführen, betrachten wir die erste Spalte $a^{(1,1)}$ der Matrix $A^{(1)}$, bilden den Vektor $v^{(1)} := v(1, a^{(1,1)})$ sowie die HOUSEHOLDER-Matrix $H^{(1)}$ gemäß (2.4), und wenden $H^{(1)}$ auf jeden Spaltenvektor von $A^{(1)}$ und auf $b^{(1)}$ an. Nach dem Lemma 2.4 Teil (a) bekommen wir ein System

$$A^{(2)}x = b^{(2)}, \text{ wobei } A^{(2)} \text{ die Form } \begin{pmatrix} * & \\ 0 & \\ \vdots & * \\ 0 & \end{pmatrix} \text{ hat.}$$

Dies wird iteriert:
Nun bilden wir $v^{(2)} := v(2, a^{(2,2)})$ aus der 2. Spalte $a^{(2,2)}$ der Matrix $A^{(2)}$, wenden das zugehörige $H^{(2)}$ auf die Spalten von $A^{(2)}$ und auf $b^{(2)}$ an und bekommen $A^{(3)}x = b^{(3)}$, wobei die zweite Spalte von $A^{(3)}$ nach Teil (a), Lemma 2.4, die Form

$$\begin{pmatrix} * \\ * \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

hat und wobei die erste Spalte von $A^{(2)}$ sich nach Teil (b), Lemma 2.4 nicht verändert hat. Nach $n-1$ solchen Schritten haben wir also

$$A^{(n)} = R = \begin{pmatrix} * & \ldots & * \\ & \ddots & \vdots \\ 0 & & * \end{pmatrix}.$$

Beachte : Wegen Lemma 2.4 Teil (c) bleiben im i-ten HOUSEHOLDER–Schritt die ersten $i-1$ Zeilen des Schemas invariant!

$$\begin{aligned} Ax &= b && \mid H^{(1)} \cdot \\ H^{(1)}Ax &= H^{(1)}b && \mid H^{(2)} \cdot \\ &\vdots \\ \overbrace{\underbrace{H^{(n-1)} \cdot \ldots \cdot H^{(1)}}_{=:R} A}^{Q^t} x &= \underbrace{H^{(n-1)} \cdot \ldots \cdot H^{(1)}}_{=:Q^t} b \end{aligned} \tag{2.6}$$

$H^{(n-1)} \cdot \ldots \cdot H^{(1)}$ ist als Produkt orthogonaler Matrizen orthogonal. Alle Transformationen waren Äquivalenzumformungen, also gilt gemäß (2.6):

$$Ax = b \Leftrightarrow Rx = Q^t b \Leftrightarrow QRx = b$$

und, wiederum mit (2.6), $Q^t A = R$, also $A = QR$.

Beachte:

1. Obiger Algorithmus ist für beliebige Matrizen $A \in \mathbb{R}^{n,n}$ durchführbar; falls A nicht invertierbar, so ist R nicht invertierbar und umgekehrt (da alle $H^{(i)}$ invertierbar).
2. Die Matrizen $H^{(i)}$ werden nicht explizit aufgebaut.
3. Speichertechnisch arbeitet man um $Ax = b$ zu lösen auf dem Feld $(A|b) \in \mathbb{R}^{n\times(n+1)} \to (R|Q^t b)$. Die anfallenden Nullen im linken unteren Teil müssen nicht explizit gesetzt werden.
4. Falls man sehr viele LGS mit übereinstimmender Matrix A lösen möchte, ist es sinnvoll, Q bzw. Q^t explizit zu berechnen. Dies macht man, indem man den Algorithmus auf das Feld $(A|Id) \in \mathbb{R}^{n\times 2n}$ anwendet, was zu $(R|Q^t)$ transformiert wird.
5. Wegen Lemma 2.4 Teil c bleiben im i-ten HOUSEHOLDER-Schritt die ersten $i-1$ Zeilen invariant.

Der Algorithmus der QR-Zerlegung lautet wie folgt:

Eingabe : Matrix $A \in \mathbb{R}^{n\times n}$, rechte Seite $b \in \mathbb{R}^n$

Ausgabe : Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$ und Vektor $Q^t b$ in Variante 1 bzw. Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$ und Matrix Q^t in Variante 2. (Wo die Ausgabe im Speicher zu finden ist: Siehe Punkte 3. und 4. in obiger Aufzählung).

Variante 1 : Zum Lösen eines linearen Gleichungssystems $Ax = b$:

```
 1:  for i := 1 to n do
 2:      a_{i,n+1} := b_i
 3:  for j := 1 to n - 1 do {
 4:      c := sign(a_jj) * sqrt( sum_{i=j}^{n} a_ij^2 )
 5:      if c ≠ 0 then {
 6:          a_jj := c + a_jj
 7:          d := 1/(a_jj)
 8:          for k := j + 1 to n + 1 do {
 9:              e := d * sum_{i=j}^{n} (a_ij * a_ik)
10:              for i := j to n do a_ik := a_ik - e * a_ij
11:          }
12:          a_jj := -c
13:      }
14:  }
```

Anschließend ist das Dreieckssystem $Rx = Q^t b$ (wobei der Vektor $Q^t b$ der $n+1$-ten Spalte des Arrays entnommen wird) per Rückwärtseinsetzen zu lösen.

Variante 2 : Zur Berechnung der Faktoren R und Q ersetze Zeile 2 durch

```
2':      for i := 1 to n do a_{i,n+j} := δ_ij
```

und „$n+1$“ in Zeile 8 durch „$2n$“.

Erläuterung :

- Zeilen 1-2 bzw. 1-2′ bauen das erweiterte Schema auf.
- j = Nummer des HOUSEHOLDER-Schritts.
- Zeile 5: Falls $c = 0$, so ist $H^{(j)} = Id$, d.h. es ist nichts zu tun.
- Zeile 8: Spaltenindex k startet erst bei $j+1$ wegen Lemma 2.4 Teil (b) und Anmerkung 3. auf Seite 23.
- Zeile 10: Zeilenindex i startet erst bei j wegen Lemma 2.4 Teil (c).
- Zeilen 6-11: Verwenden die Darstellung

$$H^{(j)}x = x - ev \quad \text{mit} \quad e = dv^t x, \quad d = \frac{2}{\|v\|_2^2} = \frac{1}{(c + a_{jj})c}$$

wobei das letzte “=” wie im Beweis von Lemma 2.4 Teil (a) folgt.

Aufwand der QR-Zerlegung (siehe Übung)

Variante 1: $\frac{2}{3}n^3 + O(n^2)$ Additionen + Multiplikationen,

Variante 2: $\frac{5}{3}n^3 + O(n^2)$ Additionen + Multiplikationen,

sowie jeweils $n-1$ Wurzeln. Das ist mehr Aufwand als für die LR-Zerlegung/Gauß ($\sim \frac{1}{3}n^3$), dieser kann aber ggf. gerechtfertigt sein wegen der Erhaltung der Kondition (→ Kapitel 2.1). Man sieht ferner: Aufwandsmäßig lohnt sich Variante 2 gegenüber 1, sobald man 3 oder mehr lineare Gleichungssysteme mit gleicher Matrix lösen will.

Lemma 2.5 *Für jede $n \times n$- Matrix A existiert eine Zerlegung $A = QR$ mit $Q \in \mathcal{O}^n$, R = rechte Dreiecksmatrix. Für invertierbares A sind die Faktoren Q und R eindeutig bestimmt in folgendem Sinn: Falls $A = QR$ und $A = \tilde{Q}\tilde{R}$ mit $Q, \tilde{Q} \in \mathcal{O}$, $R, \tilde{R}$ rechte Dreiecksmatrizen, so ist $\tilde{Q} = QD, \tilde{R} = DR$, wobei D eine Diagonalmatrix mit Einträgen ± 1 ist. D.h.: R und Q sind eindeutig bis auf Multiplikation von Zeilen von R und Spalten von Q mit (-1), Falls man also z.B. zusätzlich fordert $r_{ii} \geq 0 \ \forall \ i = 1, \ldots, n$ oder $r_{ii} \leq 0 \ \forall \ i = 1, \ldots, n$, so sind Q und R eindeutig.*

Beweis : Die Existenz ist klar nach obigem Algorithmus. Zur Eindeutigkeit: Sei

$$A = QR = \tilde{Q}\tilde{R} \quad \Rightarrow \quad \tilde{Q}^t Q = \tilde{R}R^{-1} \ .$$

Setzte

$$D := \tilde{Q}^t Q = \tilde{R}R^{-1} \ ,$$

da $\tilde{R}, R$ rechte Dreiecksmatrizen sind, ist auch D rechte Dreiecksmatrix. Da $\tilde{Q}, Q \in \mathcal{O}^n$ ist auch $D \in \mathcal{O}$, also $D^t = D^{-1}$. Da D rechte Dreiecksmatrix ist, ist D^{-1} rechte Dreiecksmatrix, und D^t ist linke Dreiecksmatrix $\Rightarrow D$ ist Diagonalmatrix. Da $D^t D = Id$ sind die Diagonaleinträge von D gleich ± 1; insbesondere $D^{-1} = D$ und nach Definition von D:

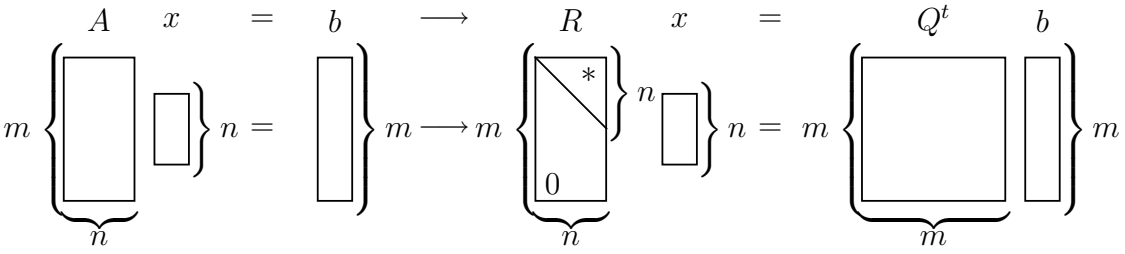
$$\tilde{R} = DR, \quad Q = \tilde{Q}D \Rightarrow \tilde{Q} = QD^{-1} = QD$$

□

Die Verallgemeinerung der QR-Zerlegung auf rechteckige Matrizen ist straight-forward:

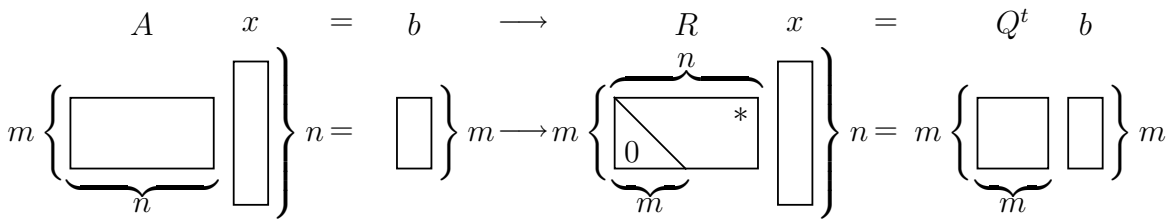
Eine gegebene Matrix $A \in \mathbb{R}^{m \times n}$ wird zerlegt in eine verallgemeinerte Dreiecksmatrix $R \in \mathbb{R}^{m \times n}$ und eine Orthogonalmatrix $Q \in \mathbb{R}^{m \times m}$.

- Im Fall $m > n$ ("überbestimmtes System "):



  Es sind n viele HOUSEHOLDER-Transformationen mit Matrix $H^{(i)} \in \mathbb{R}^{m \times m}, i = 1, \ldots, n$ nötig. Operationen $\sim 2n^2 m$ für $m \gg n$.

- Im Fall $m \leq n$ ("unterbestimmtes System"):



Es sind $m-1$ viele HOUSEHOLDER-Transformationen mit Matrix $H^{(i)} \in \mathbb{R}^{m\times m}, i = 1,\ldots,m-1$ nötig.

Es ist für $n, m \in \mathbb{N}$

$$rg(R) = rg(A),$$

da Q invertierbar.

2.3 Bestimmung des (numerischen) Ranges durch QR-Zerlegung mit Spaltenpivotisierung

Eine Variante des QR-Verfahrens besteht darin, eine Spalten-Pivot-Strategie einzuführen.
Im k-ten HOUSEHOLDER-Schritt: $H^{(k)}A^{(k)} = A^{(k+1)}$,

$$A^{(k)} = \left(\begin{array}{c|c} U^{(k)} & V^{(k)} \\ \hline 0 & W^{(k)} \end{array}\right)$$

mit $A^{(k)} \in \mathbb{R}^{m\times n}$, $U^{(k)} \in \mathbb{R}^{(k-1)\times(k-1)}$ (Dreiecksmatrix), $V^{(k)} \in \mathbb{R}^{(k-1)\times(n-k+1)}$,
$W^{(k)} \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$.
Die Einträge von $U^{(k)}, V^{(k)}$ ändern sich in den folgenden HOUSEHOLDER-Schritten nicht mehr und stellen Einträge der finalen Matrix R dar. (vgl. Lemma 2.4 Teil c).
Man ermittelt diejenige Spalte von $W^{(k)}$ mit größter Euklidischer Norm. Die entsprechende Spalte von $\begin{pmatrix} V^{(k)} \\ W^{(k)} \end{pmatrix}$ tauscht man mit der 1. Spalte von $\begin{pmatrix} V^{(k)} \\ W^{(k)} \end{pmatrix}$. Dann bestimmt man die zu dieser Spalte gehörige HOUSEHOLDER-Matrix $H^{(k)} = H(v(k,a))$ gemäß (2.4) und multipliziert $A^{(k)}$ von links mit ihr. Der rechte untere Eintrag von $U^{(k+1)}$, d.h. der Diagonaleintrag r_{kk} von R, ist bis aufs Vorzeichen gleich der Norm der erwählten Spalte von $W^{(k)}$, also größer gleich der Normen aller Spalten von $W^{(k)}$. Da sich die Spaltennorm bei Multiplikation mit orthogonalen Matrizen nicht ändert, und sich die Einträge im oberen Bereich von $A^{(k)}$ nicht mehr ändern, ist $|r_{kk}| \geq$ der Spaltennorm von $W^{(k+1)}$, also größer gleich der 1-Norm jeder Spalte von $W^{(k+1)}$.
Nach Ausführung der $\min\{n, m-1\}$ Householder-Schritte erhält man eine Zerlegung

$$AP = QR$$

mit Permutations-Matrix (Pivot-Matrix) $P \in \mathbb{R}^{n\times n}$ (vgl. L.A. I, Kapitel 2.4: Spaltenpivotsuche bei der LR-Zerlegung; die Matrix P wird natürlich nicht voll gespeichert, sondern nur eine Liste von Spaltenindizes j_k, die die Vertauschungen $k \leftrightarrow j_k$ dokumentieren).

Die Konstruktion hat zur Folge:

$$|r_{11}| \geq |r_{22}| \geq \ldots \geq |r_{n'n'}| \quad \text{mit} \quad n' = \min\{n, m\}$$

Insbesondere gilt: Falls ein k mit

$$r_{k+1,k+1} = 0 \tag{2.7}$$

auftritt, so ist

$$r_{k+1,k+1} = \ldots = r_{n',n'} = 0 \ .$$

Da $|r_{k+1,k+1}|$ größer gleich der 1-Norm aller Spalten von $W^{(k+1)}$ ist, folgt $W^{(k+1)} = 0$, sobald ein $r_{k+1,k+1} = 0$ ist. Man kann also, sobald ein $r_{k+1,k+1} = 0$ erreicht ist, die QR-Zerlegung (mit Spaltenpivotisierung) abbrechen und hat

$$R = \underbrace{m \left\{ \begin{pmatrix} \sharp & * & \ldots & \ldots & \ldots & * \\ & \ddots & \ddots & & & \vdots \\ & & \sharp & * & \ldots & * \\ & 0 & & & & \\ & & & & & \end{pmatrix} \right.}_{n}$$

wobei "$\sharp$" einen Eintrag, der sicher $\neq 0$ ist, bezeichnet. Somit ist $rg(A) = rg(R) = k$ $(\leq \min\{m, n\})$.

Da numerische Rechnungen immer rundungsfehlerbehaftet sind, ist es sinnvoll, das Kriterium (2.7) zu ersetzen durch

$$|r_{k+1,k+1}| < \varepsilon. \tag{2.8}$$

Man nennt das kleinste k, das (2.8) erfüllt, den " numerischen Rang" von A.
Mögliche Wahl: $\varepsilon := \delta \cdot |r_{11}|$, wobei δ die relative Genauigkeit der Einträge von A ist (z.B. $\delta = \tau$).

2.4 Givens-Rotationen

Diese stellen eine Alternative zu den Householder-Transformationen zur Berechnung der QR-Zerlegung dar.
Trivialerweise ist die Drehmatrix

$$G := \begin{pmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{pmatrix} (\vartheta = \text{Drehwinkel}),$$

oder gleichbedeutend

$$G := \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \text{ mit } c^2 + s^2 = 1 \tag{2.9}$$

eine Orthogonalmatrix.

Man kann eine gegebene $2 \times 2-$Matrix $A = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$ durch Multiplikation mit G auf Dreiecksform bringen.
Wie sind c,s zu wählen?

$$GA = \begin{pmatrix} * & * \\ -s\alpha + c\gamma & * \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} * & * \\ 0 & * \end{pmatrix} \Leftrightarrow s\alpha = c\gamma \tag{2.10}$$

Eine Lösung von (2.9) - (2.10) ist (für $(\alpha, \gamma) \neq (0, 0)$):

$$c(\alpha, \gamma) = \frac{\alpha}{\sqrt{\alpha^2 + \gamma^2}} \ , s(\alpha, \gamma) = \frac{\gamma}{\sqrt{\alpha^2 + \gamma^2}}.$$

Falls $(\alpha, \gamma) = (0, 0)$, so hat A schon Dreiecksform.
Dieses Prinzip lässt sich auf allgemeine $m \times n$- Matrizen A übertragen.

Definition 2.6 *Die $m \times m$ Matrix*

$$G(i,j,c,s) := \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c & & s & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ & & & -s & & c & \\ & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow\ j \\ \\ \\ \in \mathcal{O}^n \\ \\ \leftarrow\ i \\ \\ \\ \end{matrix} \qquad (2.11)$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & j & & i & & \end{matrix}$$

mit $i, j \in \{1, \ldots, m\}$, $i \neq j$ und $c^2 + s^2 = 1$ heißt GIVENS-*Rotation. Multiplikation einer $m \times n$-Matrix A (von links) mit ihr heißt* GIVENS-*Transformation.*
Bei der GIVENS*transformation ändern sich offensichtlich nur die i-te und die j-te Zeile von A.*
Wir durchlaufen nun nacheinander die Spalten $j = 1, \ldots, n-1$ und für jeden Eintrag $a_{ij} \neq 0$, $i > j$, führen wir eine GIVENS*-Rotation durch zur Elimination dieses Eintrags a_{ij}; die Rolle von α und γ spielen a_{jj} und a_{ij}*

$$c = \frac{a_{jj}}{\sqrt{a_{jj}^2 + a_{ij}^2}}, \quad s = \frac{a_{ij}}{\sqrt{a_{jj}^2 + a_{ij}^2}}, \quad G \text{ nach (2.11).} \qquad (2.12)$$

Beachte: $a_{jj} \neq 0$ ist nicht erforderlich. Bei der GIVENS*-Rotation wird die i-te Zeile von A durch das c-fache der i-ten minus das s-fache der j-ten Zeile, und die j-te Zeile wird durch das c-fache der j-ten Zeile plus das s-fache der i-ten Zeile ersetzt; im Fall $a_{jj} = 0$ bedeutet das eine einfache Vertauschung der i-ten mit der j-ten Zeile; im Fall $a_{ij} = 0$ bedeutet dies ein Unverändertlassen der Matrix.*

Um Under-/Overflow bei der Berechnung der Wurzel zu vermeiden (vgl. Bsp. in Kapitel 1.1) ist es sinnvoll, (2.11) zu ersetzen durch:

$$\text{Falls } |a_{jj}| \geq |a_{ij}|: \quad t := \frac{a_{ij}}{|a_{jj}|},\ c = \frac{sgn(a_{jj})}{\sqrt{1+t^2}},\ s = \frac{t}{\sqrt{1+t^2}}$$

$$\text{Falls } |a_{jj}| < |a_{ij}|: \quad t := \frac{a_{jj}}{|a_{ij}|},\ c = \frac{t}{\sqrt{1+t^2}},\ s = \frac{sgn(a_{ij})}{\sqrt{1+t^2}} \qquad (*)$$

$$\text{mit } sgn(a) = \begin{cases} -1, & a < 0, \\ 0, & a = 0, \\ 1, & a > 0 \end{cases}$$

(Skalierung mit $\max\{|a_{ij}|, |a_{jj}|\}$; immer ist $t \in [-1, 1]$)
Im allgemeinen Fall sind also $\frac{n}{2}(n-1)$ GIVENS-Rotationen ($A \in \mathbb{R}^{n\times n}$) (mit jeweils Aufwand $O(n)$) statt $n-1$ Householder-Transformationen (mit je Aufwand $O(n^2)$) nötig. Beliebt sind GIVENS-Rotationen, wenn A unterhalb der Hauptdiagonalen nur wenige Einträge $\neq 0$ hat, diese aber über viele Spalten verteilt sind, wie z.B. bei sogenannten Hessenberg-Matrizen

$$A = \begin{pmatrix} * & \dots & \dots & * \\ * & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ 0 & & * & * \end{pmatrix}$$

Für solche Matrizen sind nur $n-1$ GIVENS-Rotationen (Gesamtaufwand $O(n^2)$!) nötig, was effizienter ist als $n-1$ Householder-Transformationen (Gesamtaufwand $O(n^3)$).

2.5 QR- und LR-Zerlegung für Bandmatrizen

Sowohl die QR-Zerlegung als auch die LR-Zerlegung, wenn ohne Pivotisierung durchgeführt, bewahren in gewissem Sinne die Bandstruktur von A: Sei $A \in \mathbb{R}^{n,n}$ von der Bandstruktur

$$A = \begin{pmatrix} * & \dots & * & & & & & \\ * & \ddots & & \ddots & & & 0 & \\ \vdots & \ddots & \ddots & & \ddots & & & \\ * & & \ddots & \ddots & & \ddots & & \\ & \ddots & & \ddots & \ddots & & \ddots & \\ & & \ddots & & \ddots & \ddots & & * \\ & 0 & & \ddots & & \ddots & \ddots & \vdots \\ & & & \underbrace{*}_{p} & \dots & * & * & * \end{pmatrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \\ a \\ a \\ a \\ a \end{matrix}}\right\} q \text{ obere Bänder} \tag{2.13}$$

untere Bänder

Das heißt die Matrix A hat p untere Bänder und q obere Bänder. Beachte: Wir zählen die Hauptdiagonale zu den oberen Bändern).

Dann haben Q, R nach der QR-Zerlegung die Besetzungsstruktur

$$Q = \begin{pmatrix} * & \dots & \dots & \dots & \dots & \dots & \dots & * \\ * & \vdots & & & & & & \vdots \\ \vdots & \ddots & \ddots & & & * & & \vdots \\ * & & \ddots & \ddots & & & & \vdots \\ & \ddots & & \ddots & \ddots & & & \vdots \\ & & \ddots & & \ddots & \ddots & & \vdots \\ & 0 & & \ddots & & \ddots & \ddots & \vdots \\ & & & & * & \dots & * & * \end{pmatrix} \tag{2.14}$$

$$R = \begin{pmatrix} * & \cdots & * & & & & \\ & \ddots & & \ddots & & & \\ & & \ddots & * & \ddots & & \\ & & & \ddots & & \ddots & \\ & & & & \ddots & & * \\ & & & & & \ddots & \vdots \\ & & & & & & * \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \overbrace{\qquad}^{p} \\ \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix}\right\} p+q \tag{2.15}$$

Beweisskizze : Wir erhalten $R := A^{(n-1)}$, wobei $A^{(k+1)} := H^{(k)}A^{(k)}$, $A^{(1)} := A$; und $H^{(k)}$ gemäß der Darstellung (2.5)

$$H^{(k)} = Id - \frac{2}{\|v\|_2^2} \begin{pmatrix} v_1v_1 & \cdots & v_1v_n \\ \vdots & & \vdots \\ v_nv_1 & \cdots & v_nv_n \end{pmatrix} \text{ mit } v = v(k,a) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_k + c \\ \vdots \\ a_n \end{pmatrix}$$

Alle v_i mit $i < k$ sowie $i > k + p$ verschwinden wegen der Besetzungsstruktur von A.

$$\Rightarrow\ H^{(k)} = \left(\begin{array}{ccc} 1 & & \\ & \ddots & \\ & & 1 \\ \hline & * & \\ \hline & & \begin{matrix} 1 & \\ & \ddots \\ & & 1 \end{matrix} \end{array}\right)$$

$$\text{und insbesondere} \quad H^{(1)} = \left(\begin{array}{c|c} * & 0 \\ \hline 0 & \begin{matrix} 1 & \\ & \ddots \\ & & 1 \end{matrix} \end{array}\right)$$

Multiplikation von $H^{(1)}$ mit $A^{(1)}$ ergibt ein $A^{(2)}$ mit der Besetzungsstruktur

$$A^{(2)} = \begin{pmatrix} * & \dots & \dots & \dots & \dots & * & & \\ 0 & * & & & & \vdots & & \\ \vdots & \vdots & \ddots & & & \vdots & 0 & \\ \vdots & * & & \ddots & & * & & \\ \vdots & & \ddots & & \ddots & & \ddots & \\ \vdots & & & \ddots & & \ddots & & * \\ \vdots & & 0 & & \ddots & & \ddots & \vdots \\ 0 & & & & & * & \dots & * \end{pmatrix},$$

$$A^{(3)} = \begin{pmatrix} * & \dots & \dots & \dots & * & 0 & & \\ 0 & * & & & & * & & \\ \vdots & 0 & * & & & \vdots & 0 & \\ \vdots & \vdots & \vdots & \ddots & & * & & \\ \vdots & \vdots & * & & \ddots & & \ddots & \\ \vdots & \vdots & & \ddots & & \ddots & & * \\ \vdots & \vdots & & 0 & \ddots & & \ddots & \vdots \\ 0 & 0 & & & & * & \dots & * \end{pmatrix}$$

Man verfolgt die Struktur von $A^{(3)}, \dots, A^{(n-1)} = R$ und erhält so die Struktur (2.15) für R. R^{-1} ist, wie R, obere Dreiecksmatrix (allerdings *ohne* Bandstruktur). Aus $Q = AR^{-1}$ und der Struktur (2.13) von A folgt die Sturktur (2.14) von Q.

Noch besser bleibt die Bandstruktur bei der LR-Zerlegung erhalten: Eine Matrix A der Form (2.13) zerfällt in

$$L = \begin{pmatrix} 1 & & & & & \\ * & \ddots & & & 0 & \\ \vdots & \ddots & \ddots & & & \\ * & & \ddots & \ddots & & \\ & \ddots & & \ddots & \ddots & \\ 0 & & * & \dots & * & 1 \end{pmatrix}, \quad \text{und} \quad R = \begin{pmatrix} * & \dots & * & & & 0 \\ & \ddots & & \ddots & & \\ & & \ddots & & \ddots & \\ & & & \ddots & & * \\ & & 0 & & \ddots & \vdots \\ & & & & & * \end{pmatrix}$$

Der Aufwand ist für $p = q \ll n$ proportional zu $p^2 \cdot n$ ist also, für *festes* $p, q \ll n$ nur $O(n)$!

Beachte: Bei Verwendung von Pivot-Strategie vergrößert sich bei LR i.a. die Bandbreite von R zu $p + q$.

Für tridiagonale lineare Gleichungssysteme $Ax = d$ (vgl. Blatt 10, Aufgabe 52, L.A. I)

$$A = \begin{pmatrix} a_1 & b_1 & & & \\ c_1 & \ddots & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ 0 & & & c_{n-1} & a_n \end{pmatrix}$$

liefert der Ansatz

$$L = \begin{pmatrix} 1 & & & \\ l_1 & \ddots & & 0 \\ & \ddots & \ddots & \\ & & \ddots & \ddots \\ 0 & & l_{n-1} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} m_1 & r_1 & & \\ & \ddots & \ddots & 0 \\ & & \ddots & \ddots \\ & 0 & & \ddots & r_{n-1} \\ & & & & m_n \end{pmatrix}$$

die Gleichungen:

$$\begin{array}{rclcrclcl} & & & & a_1 & = & m_1, & & b_1 = r_1 \\ c_i & = & l_i m_i, & & a_{i+1} & = & l_i r_i + m_{i+1}, & & b_{i+1} = r_{i+1},\ i = 1, \ldots, n-2 \\ c_{n-1} & = & l_{n-1} m_{n-1}, & & a_n & = & l_{n-1} r_{n-1} + m_n & & \end{array}$$

Hieraus lassen sich rekursiv alle l_i, m_i, r_i berechnen:

$$r_i = b_i \ \forall i = 1, \ldots, n-1$$

Mit $m_1 := a_1$ erhält man:

```
for i := 1 to n-1 do {
    l_i = c_i / m_i
    m_{i+1} = a_{i+1} - l_i b_i
}
```

Falls $m_i = 0$ auftritt, hätte man Pivotisieren müssen. Anschließend wird die Vorwärtssubstitution $Ly = d$ ausgeführt. Sie lautet:

```
y_1 := d_1
for i := 2 to n do {
    y_i := d_i - l_{i-1} y_{i-1}
}
```

Danach nur noch Rückwärtseinsetzen $Rx = y$:

```
x_n := y_n / m_n
for i := n-1 to 1 do {
    x_i = (y_i - b_i x_{i+1}) / m_i
}
```

2.6 Lineare Ausgleichsprobleme

Wir betrachten lineare Gleichungssysteme $Ax = b$ mit $A \in \mathbb{R}^{m\times n},\ m > n,\ b \in \mathbb{R}^m$ ("überbestimmt")[1]. Diese haben im Allgemeinen keine Lösung. Daher betrachtet man stattdessen das sogenannte *lineare Ausgleichsproblem*

$$\boxed{\text{Finde } x \in \mathbb{R}^n \text{ so, dass } \|Ax - b\| \text{ minimal wird.}} \tag{2.16}$$

Meist wird $\|\cdot\| := \|\cdot\|_2$ benutzt (sog. „Methode der kleinsten Quadrate").

Vorkommen : Für einen zu bestimmenden funktionalen Zusammenhang $t \mapsto f(t),\ \mathbb{R} \to \mathbb{R}$ werden viele Messwerte $(t_i, y_i),\ i = 1, \dots m$ erhoben. Für f wird der Ansatz als Element eines niedrigdimensionalen Funktionenraums mit Basis $\{\varphi_1, \dots \varphi_n\} (n \le m)$ gemacht. Dann sucht man $x_1, \dots x_n \in \mathbb{R}$ so, dass $f(t) = \sum_{j=1}^n x_j \varphi_j(t)$ möglichst gut zu den Messwerten passt:

$$\sum_{j=1}^n x_j \varphi(t_i) \approx y_i \quad \forall i = 1, \dots, m$$

bzw. präziser

$$\sum_{i=1}^m \left(\sum_{j=1}^n \varphi_j(t_i) x_j - y_i \right)^2 \to \text{ minimal}$$

(„Methode der kleinsten Quadrate"). Dieses Problem hat die Form (2.16) mit $\|\cdot\| = \|\cdot\|_2$ und Matrix $A = (\varphi_j(t_i))_{\substack{i=1\dots m \\ j=1\dots n}},\quad b = (y_i)_{i=1,\dots m}$

Setze von nun an $\|\cdot\| := \|\cdot\|_2$ voraus. In L.A. I, (2.85)-(2.86), wurde (für $\|\cdot\| = \|\cdot\|_2$) gezeigt:

Lemma 2.7

(a) $x \in \mathbb{R}^n$ ist eine Lösung von (2.16) $\Leftrightarrow$ x erfüllt

$$A^t A x = A^t b \tag{2.17}$$

(b) Die Lösungsmenge ist nie leer. Die Lösung von (2.16)/(2.17) ist genau dann eindeutig, wenn A maximalen Rang hat, d.h $rg(A) = n$.

Beweis (Wdh.):

[1] Der Fall $m = n$ ist im Folgenden ebenfalls zulässig.

zu a) x erfüllt $\|Ax-b\| = \min\limits_{y\in\mathbb{R}^n}\{\|Ay-b\|\} = \min\limits_{v\in \text{Bild}(A)}\{\|v-b\|\}$

$\Leftrightarrow$ $u := Ax (\in \text{ Bild}A)$ erfüllt $\|u-b\| = \min\limits_{v\in \text{ Bild}(A)}\{\|v-b\|\}$

$\Leftrightarrow$ $u = Ax$ ist Orthogonalprojektion von b auf Bild(A) und ist damit eindeutig

$\Leftrightarrow$ $Ax-b \in \text{Bild}(A)^\perp \underset{\text{L.A. I}}{=} \text{Kern}(A^t)$, und das ist äquvialent zu $A^t(Ax-b)=0$

$\Leftrightarrow$ x erfüllt 2.17

[Beachte: Die Eindeutigkeit von u bedeutet nicht, dass x im allgemeinen eindeutig ist]

zu b) Da $u \in \text{Bild}(A)$ existiert immer mindestens eine Lösung x.

$$n = rg(A) \underset{(*)}{=} rg(A^tA) \Leftrightarrow A^tA \text{ invertierbar}$$
$$\Leftrightarrow \text{eindeutige Lösung } x = \underbrace{(A^tA)^{-1}A^t}_{=\text{ Pseudoinverse}}\, b$$

wobei $(*)$ nach (4.69) f. aus L.A. II oder Aufgabe 2.17, L.A. I □

Im folgenden beschäftigen wir uns zunächst mit dem Fall $rg(A) = n$. Wie können wir die (eindeutige) Lösung von (2.16) berechnen?
Erster Versuch: über das lineare Gleichungssystem $A^tAx = A^tb$ (wie in L.A. I S.110 vorgeschlagen). Nachteil: Diese Gleichung hat eine deutlich größere Fehlersensitivität als ein lineares Gleichungssystem mit Systemmatrix A.

So z.B. ist im Fall $n = m$ und A invertierbar $\kappa_2(A^tA) = \kappa_2(A)^2$, also i.a. $\kappa_2(A^tA) \gg \kappa_2(A)$

Beweis : Per Definitoin ist
$$\begin{aligned} \kappa_2(A) &= \|A\|_2\|A^{-1}\|_2 = \frac{\sqrt{\lambda_{\max}(A^tA)}}{\sqrt{\lambda_{\min}(A^tA)}} \\ \kappa_2(A^tA) &= \|A^tA\|_2\|(A^tA)^{-1}\|_2 \underset{\text{sym.Matrizen}}{=} \\ &= \lambda_{\max}(A^tA)\cdot\lambda_{\max}((A^tA)^{-1}) = \frac{\lambda_{\max}(A^tA)}{\lambda_{\min}(A^tA)} \end{aligned}$$

□

Zweiter Versuch: Stattdessen folgende Überlegung: Wir führen die QR-Zerlegung $A = QR$ durch.

$$\varphi(x) = \|Ax-b\|_2^2 \underset{Lemma 2.2(a)}{=} \|Q^t(Ax-b)\|_2^2 \underset{Q^tA=R}{=} \|Rx-Q^tb\|_2^2 \tag{2.18}$$

Da wir im Fall $A, R \in \mathbb{R}^{m,n}$ mit $m \geq n$ sind, hat R die Form

$$R = \underbrace{\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}}_{n} \begin{matrix} \}n \\ \}m-n \end{matrix} \quad \text{mit } \tilde{R} \in \mathbb{R}^{n\times n} \text{ obere Dreiecksmatrix.}$$

$$\Rightarrow \quad \varphi(x) = \left\| \begin{pmatrix} \tilde{R}x \\ 0x \end{pmatrix} - \begin{pmatrix} (Q^tb)^{(1)} \\ (Q^tb)^{(2)} \end{pmatrix} \right\|_2^2$$

wobei $y \in \mathbb{R}^m$ zerlegt in $y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix}$, $y^{(1)} \in \mathbb{R}^n$, $y^{(2)} \in \mathbb{R}^{m-n}$ ist.

$$\Rightarrow \quad \varphi(x) = \|\tilde{R}x - (Q^t b)^{(1)}\|_2^2 + \|(Q^t b)^{(2)}\|_2^2$$

Wir sehen:

$$\varphi(x) \geq \|(Q^t b)^{(2)}\|_2^2 \; \forall \; x \text{ und } \varphi(x) = \|(Q^t b)^{(2)}\|_2^2 \Leftrightarrow \tilde{R}x = (Q^t b)^{(1)} \tag{2.19}$$

Im Fall $rg(A) = n$, d.h. $rg(\tilde{R}) = n$, kann $\tilde{R}x = (Q^t b)^{(1)}$ mittels Rückwärtseinsetzen gelöst werden. Also: Algorithmus für (2.16) mit $\|\cdot\| = \|\cdot\|_2$, $rg(A) = n$:

(1) QR-Zerlegung von $(A|b)$; [Q selbst muss nicht aufgebaut werden, nur R und $Q^t b$]

(2) Löse $\tilde{R}x = (Q^t b)^{(1)}$ durch Rückwärtseinsetzen.

Im allgemeineren Fall $p := rg(A) \leq n$ führen wir die QR-Zerlegung *mit Spaltenpivotisierung* (Kapitel 2.3) durch. Wir erhalten die Zerlegung

$$AP = QR, \text{ mit der Struktur } R = \begin{pmatrix} \overbrace{\tilde{R}_1}^{p} & \overbrace{\tilde{R}_2}^{n-p} \\ 0 & 0 \end{pmatrix} \begin{matrix} \}p \\ \}m-p \end{matrix} \tag{2.20}$$

wobei $\tilde{R}_1$ eine *nichtsinguläre* $p \times p$-obere Dreiecksmatrix ist. Analog zu (2.18) bekommen wir für alle $x \in \mathbb{R}^n$:

$$\begin{aligned} \varphi(x) &= \|Q^t Ax - Q^t b\|_2^2 \underset{Q^t A = RP^{-1}}{=} \|RP^{-1}x - Q^t b\|_2^2 = \\ &\underset{(2.20)}{=} \left\| \begin{pmatrix} \tilde{R}_1(P^{-1}x)^{(1)} + \tilde{R}_2(P^{-1}x)^{(2)} & -(Q^t b)^{(1)} \\ 0 & -(Q^t b)^{(2)} \end{pmatrix} \right\|_2^2 = \\ &= \|\tilde{R}_1(P^{-1}x)^{(1)} + \tilde{R}_2(P^{-1}x)^{(2)} - (Q^t b)^{(1)}\|_2^2 + \|(Q^t b)^{(2)}\|_2^2 \end{aligned}$$

wobei $y \in \mathbb{R}^N$ (für $N = n$ oder $N = m$) zerlegt wurde in

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix}, \; y^{(1)} \in \mathbb{R}^p, \; y^{(2)} \in \mathbb{R}^{N-p} \tag{2.21}$$

$\varphi(x)$ wird also minimal genau dann wenn

$$\tilde{R}_1(P^{-1}x)^{(1)} = (Q^t b)^{(1)} - \tilde{R}_2(P^{-1}x)^{(2)} \tag{2.22}$$

$\Rightarrow$ Algorithmus für (2.16) im Fall $\|\cdot\| = \|\cdot\|_2$ und $p := rg(A) \leq n$:

(1) QR-Zerlegung mit Spaltenpivotisierung liefert $AP = QR$, $RP^{-1}x = Q^t b$. Die Komponenten von $P^{-1}x$ werden zerlegt in $(P^{-1}x)^{(1)}$ und $(P^{-1}x)^{(2)}$ gemäß (2.21); $\tilde{R}_1$ und $\tilde{R}_2$ ergeben sich aus R nach (2.20).

(2) Die Komponenten von $(P^{-1}x)^{(2)}$ sind frei wählbar. Löse (2.22) mittels Rückwärtseinsetzten. Man erhält (siehe Übung) eine Basis des Lösungsraums, indem man (2.22) einmal mit $\tilde{x}^{(2)} := (P^{-1}x)^{(2)} = 0$ löst (Rückwärtseinsetzen): $\tilde{R}_1\tilde{x}^{(1)} = (Q^tb)^{(1)}$ bzw. anschließend für $\tilde{x}_i^{(2)} := e_{p+1}^{(2)},\ i = 1,\dots,n-p$ nacheinander

$$\tilde{R}_1\tilde{x}_i^{(1)} = -\tilde{R}_2\tilde{x}_i^{(2)} \qquad (\text{„homogenes Problem“})$$

löst. Die Lösungsmenge von (2.22) ist dann, wegen $x = P\tilde{x}$,

$$P\begin{pmatrix}\tilde{x}^{(1)}\\0\end{pmatrix} + span\left\{P\begin{pmatrix}\tilde{x}_1^{(1)}\\e_{p+1}^{(2)}\end{pmatrix},\dots,P\begin{pmatrix}\tilde{x}_{n-p}^{(1)}\\e_n^{(2)}\end{pmatrix}\right\}$$

Anstatt die Gesamtheit U der Lösungen wie oben zu berechnen, sucht man im Fall $rg(A) < n$ oft nur diejenige Lösung von (2.16), deren Norm minimal ist, also:

$$\begin{aligned}&\text{Löse } \|x\|_2^2 \to \min \quad \text{auf der Menge}\\ U = \{x \in \mathbb{R}^n \mid \|Ax-b\|_2^2 &= \min_{y\in\mathbb{R}^n}\{\|Ay-b\|_2^2\}\}\end{aligned} \tag{2.23}$$

Die Lösung von (2.23) ist *eindeutig*, da sie äquivalent ist zur orthogonalen Projektion des Nullvektors auf den affin-linearen Raum U (U ist affin-linearer Raum, als Lösungsmenge der (linearen) Normalengleichung).

Lemma 2.8 *Sei $p < n$, $V := \tilde{R}_1^{-1}\tilde{R}_2$ $(\in \mathbb{R}^{p\times(n-p)})$ und $u := \tilde{R}_1^{-1}(Q^tb)^{(1)}$ $[\in \mathbb{R}^p]$. Dann ist die eindeutig bestimmte Lösung von (2.23) gegeben durch*

$$\Big(\underbrace{Id + V^tV}_{s.p.d.\to invertierbar}\Big)(P^{-1}x)^{(2)} = V^tu \tag{2.24}$$

$$(P^{-1}x)^{(1)} := u - V(P^{-1}x)^{(2)} \tag{2.25}$$

[V^tV ist symmetrisch positiv semidefinit; $Id + V^tV$ ist s.p.d., somit invertierbar]

Beweis : Gleichung (2.22) mit $\tilde{R}_1^{-1}$ multipliziert und Definition von V, u verwendet ergibt (2.25). Mit der Schreibweise $\tilde{x}^{(1)} := P^{-1}x^{(1)}$, $\tilde{x}^{(2)} := P^{-1}x^{(2)}$, $\tilde{x} := P^{-1}x$ erhalten wir:

$$\begin{aligned}\|x\|_2^2 = \|\tilde{x}\|_2^2 &= \|\tilde{x}^{(1)}\|_2^2 + \|\tilde{x}^{(2)}\|_2^2\\ &\underset{(2.25)}{=} \|u - V\tilde{x}^{(2)}\|_2^2 + \|\tilde{x}^{(2)}\|_2^2\\ &= \|u\|_2^2 - 2\langle u, V\tilde{x}^{(2)}\rangle + \|V\tilde{x}^{(2)}\|_2^2 + \|\tilde{x}^{(2)}\|_2^2\\ &= \|u\|_2^2 - 2\langle V^tu, \tilde{x}^{(2)}\rangle + \langle\tilde{x}^{(2)}, (V^tV + Id)\tilde{x}^{(2)}\rangle =: \underbrace{\Psi(\tilde{x}^{(2)})}_{\overset{!}{\to}\min}\end{aligned}$$

Ψ ist quadratisches Funktional mit $\nabla\Psi(\tilde{x}^{(2)}) = -2V^tu + 2(V^tV + Id)\tilde{x}^{(2)} \overset{!}{=} 0$.

$$H\Psi(\tilde{x}^{(2)} \equiv 2(V^tV + Id)$$

Die Hesse-Matrix ist s.p.d., daher hat Ψ ein eindeutig bestimmtes Minimum. Dieses wird angenommen an der Stelle, die durch (2.24) beschrieben ist. □

Die Darstellung von Lemma 2.6 führt zu einen Algorithmus zur Lösung von (2.23):

(1) QR-Zerlegung mit Spaltenpivotisierung liefert $\tilde{R}_1, \tilde{R}_2, P, Q^t b$, siehe (2.26)

(2) $\tilde{R}_1 u = (Q^t b)^{(1)}$ lösen durch Rückwärtseinsetzen ($\rightsquigarrow u$), Bestimmung der Spalten von V aus $\tilde{R}_1 V = \tilde{R}_2$ durch Rückwärtseinsetzen.

(3) Löse (2.24) mit CHOLESKY-Zerlegung (s. Kapitel 2.7) (oder LR-Zerlegung), dann $\tilde{x}^{(1)}$ mittels (2.25), setze dazu $x = P\tilde{x}$.

2.7 Die Cholesky-Zerlegung

Die CHOLESKY-Zerlegung ist neben der LR- und der QR-Zerlegung das dritte wichtige "direkte" Verfahren zum Lösen von linearen Gleichungssystemen. Es ist schneller also LR und QR, kann jedoch nur auf symmetrischen, positiv definiten (s.p.d.-) Matrizen angewendet werden.

Definition 2.9 *Sei $A \in \mathbb{R}^{n\times n}$ s.p.d.. Eine Darstellung der Form*

$$A = LL^t \tag{2.26}$$

wobei L untere Dreiecksmatrix mit positiven Diagonalelementen $l_{ii} > 0$ *ist, heißt* CHOLESKY-Zerlegung *von A.*

Ist die CHOLESKY-Zerlegung von A bekannt, so kann man offensichtlich jedes lineare Gleichungssystem $Ax = b$ mittels Vorwärtseinsetzen $Ly = b$ und anschließenden Rückwärtseinsetzen $L^t x = y$ mit Aufwand $O(n^2)$ lösen (wie bei LR-Zerlegung).

Lemma 2.10 *Für jede s.p.d.-Matrix $A \in \mathbb{R}^{n\times n}$ existiert die Cholesky-Zerlegung und ist eindeutig bestimmt.*

Beweis:

(i) Eindeutigkeit: Seien $A = LL^t$ und $A = \tilde{L}\tilde{L}^t$ zwei CHOLESKY-Zerlegungen. Nach Voraussetzung ist A invertierbar.

$$\Rightarrow \quad det(A) \neq 0 \quad \text{mit} \quad det(A) = det(L) \cdot det(L^t) = (det(L))^2 \neq 0 \Rightarrow det(L) \neq 0$$

$$\Rightarrow \quad L \quad \text{invertierbar, analog} \quad \tilde{L}, L^t, \tilde{L}^t$$

$$LL^t = \tilde{L}\tilde{L}^t \quad \Rightarrow \quad \underbrace{L^t \tilde{L}^{-t}}_{=(\tilde{L}^{-1}L)^t} = L^{-1}\tilde{L}$$

Rechts steht als Produkt von linken Dreickesmatrizen eine linke Dreiecksmatrix, wohingegen links eine rechte Dreiecksmatrix steht.

$$\Rightarrow \quad (\tilde{L}^{-1}L)^t = L^{-1}\tilde{L} =: D \quad \text{ist Diagonalmatrix}$$

$$\Rightarrow \quad \tilde{L} = LD \Rightarrow LL^t = \tilde{L}\tilde{L}^t = LDD^tL^t = LD^2L^t \Rightarrow D^2 = Id$$

$$\Rightarrow \quad d_{ii} = \pm 1 \quad \forall i = 1, \ldots, n$$

Da $\tilde{l}_{ii} = l_{ii} \cdot d_{ii}$ und $\tilde{l}_{ii} > 0,\ l_{ii} > 0$ folgt $d_{ii} = 1 \quad \forall i = 1, ..., n.$

(ii) Existenz: Rekursive Betrachtung: $A \in \mathbb{R}^{1\times 1}$ positiv definit hat offensichtlich CHOLESKY-Zerlegung: $A = (a),\ a > 0,\ L = (\sqrt{a}),\ L^t = (\sqrt{a})$
Sei nun $\tilde{A} \in \mathbb{R}^{i\times i}$ s.p.d.- Matrix. Wir schreiben sie als

$$\tilde{A} = \begin{pmatrix} A & a \\ a^t & a_{ii} \end{pmatrix},\ A \in \mathbb{R}^{(i-1)\times(i-1)},\ a \in \mathbb{R}^{i-1},\ a_{ii} \in \mathbb{R}$$

A ist dann auch s.p.d. (Beweis siehe L.A. II, S. 18/19); es sei die CHOLESKY-Zerlegung $A = LL^t$ bekannt. Wir wollen daraus die CHOLESKY-Zerlegung von $\tilde{A} = \tilde{L}\tilde{L}^t$ berechnen. Ansatz:

$$\tilde{L} = \begin{pmatrix} L & 0 \\ l^t & l_{ii} \end{pmatrix} = (l_{jk})_{j,k=1,\ldots,i}$$

d.h.

$$\begin{pmatrix} A & a \\ a^t & a_{ii} \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} L & 0 \\ l^t & l_{ii} \end{pmatrix}\begin{pmatrix} L^t & l \\ 0 & l_{ii} \end{pmatrix} = \begin{pmatrix} LL^t & Ll \\ (Ll)^t & l^t l + l_{ii}^2 \end{pmatrix}$$

Wir wissen, dass L invertierbar ist. Es ist also l durch die Gleichung

$$Ll = a \tag{2.27}$$

bestimmt, d.h. l hat die Komponenten

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}}{l_{jj}},\ j = 1, \ldots i-1$$

(„Vorwärtseinsetzen“), und l_{ii} wird dann berechnet durch

$$a_{ii} - l^t l = l_{ii}^2,\ \text{d.h. } l_{ii} = \underset{(-)}{+}\sqrt{a_{ii} - l^t l}\ . \tag{2.28}$$

Also müssen wir noch zeigen, dass $a_{ii} - l^t l > 0$. Dazu betrachten wir den Vektor

$$x := \begin{pmatrix} L^{-t}l \\ -1 \end{pmatrix} \neq 0$$

$$\begin{aligned}
0 \underset{A \text{ s.p.d.}}{<} \langle x, \tilde{A}x\rangle &= \left\langle \begin{pmatrix} L^{-t}l \\ -1 \end{pmatrix}, \begin{pmatrix} A & a \\ a^t & a_{ii} \end{pmatrix}\begin{pmatrix} L^{-t}l \\ -1 \end{pmatrix} \right\rangle \\
&= \left\langle \begin{pmatrix} L^{-t}l \\ -1 \end{pmatrix}, \begin{pmatrix} AL^{-t}l - a \\ a^t L^{-t}l - a_{ii} \end{pmatrix} \right\rangle \\
&= \langle L^{-t}l, AL^{-t}l\rangle - \langle L^{-t}l, a\rangle - \overbrace{a^t L^{-t}}^{=(L^{-1}a)^t}\ l + a_{ii} \\
&= \langle l, \underbrace{L^{-1}AL^{-t}}_{=Id,\text{da}A=LL^t}\ l\rangle - \langle l, \underbrace{L^{-1}a}_{=l}\rangle - \langle \underbrace{L^{-1}a}_{=l}, l\rangle + a_{ii} \\
&= l^t l - l^t l - l^t l + a_{ii} = a_{ii} - l^t l
\end{aligned}$$

□

Dieser Beweis liefert praktischerweise den Algorithmus zur zeilenweise Berechnung von L:

```
1:  for i := 1 to n do {                          % (Zeilen)
2:      for j := 1 to n do {                      % (Spalten)
3:          if j < i                              % (links von der Hauptdiagonalen)
4:              l_ij := (a_ij - sum_{k=1}^{j-1} l_ik l_jk) / l_jj
5:          if j = i
6:              l_ii := sqrt(a_ii - sum_{k=1}^{i-1} l_ik^2)
7:          if j > i
8:              lij := 0
9:      }
10: }
```

Jedes a_{ij} wird offenbar nur einmal evaluiert und kann sofort danach überschrieben werden. Das heißt, dass A mit den Werten von L überschrieben werden kann. Es reichen offensichtlich insgesamt $\frac{n}{2}(n+1)$ Speicherplätze für A *und* L. (A symmetrisch!)

```
1:  for i := 1 to n do {
2:      for j := 1 to i-1 do
3:          a_ij := (a_ij - sum_{k=1}^{j-1} a_ik a_jk) / a_jj                (♯)
4:      a_ii := sqrt(a_ii - sum_{k=1}^{i-1} a_ik^2)                          (♯♯)
5:  }
```

Aufwand:

In (♯) und (♯♯) sind ca. $j-$ bzw. $i-$ viele Additionen und Multiplikationen auszuführen, also insgesamt

$$\sum_{i=1}^{n}(\sum_{j=1}^{i-1} j + i) \doteq \sum_{i=1}^{n}(\frac{i^2}{2} + i) \doteq \sum_{i=1}^{n} \frac{i^2}{2} \doteq \frac{n^3}{6}$$

Es sind also nur $\sim \frac{1}{6}n^3$ Additionen und Multiplikationen nötig, sowie n Wurzelberechnungen (vgl. LR: $\sim \frac{1}{3}n^3$). Dies liefert die "Existenzberechtigung" des CHOLESKY- Verfahrens. *Beachte auch:* CHOLESKY funktioniert für s.p.d.-Matrizen *immer ohne Pivotisierung*. LR funktioniert für beliebige invertierbare Matrizen im Allgemeinen nicht ohne Pivotisierung. Man kann leicht zeigen:

Für s.p.d Matrizen existiert auch die LR-Zerlegung ohne Pivotisierung.

($A = LL^t \Rightarrow A = LDD^{-1}L^t,\ \tilde{L} := LD,\ R := D^{-1}L^t$, wobei D Diagonalmatrix so dass LD Einsen auf der Hauptdiagonalen hat $\Rightarrow A = (LD)(D^{-1}L^t) = \tilde{L}R$ ist LR-Zerlegung von A; vgl. auch Satz 4.57, L.A. II)

3 Nichtlineare Gleichungen

Problemstellung:

Definition 3.1 *Sei* $(X, \|\cdot\|)$ *ein normierter Vektorraum,* $U \subset X,\ \Phi : U \to X,\ b \in X$.
Das Problem

$$\text{“Finde } x \in U \text{ mit } \Phi(x) = 0\text{”} \tag{3.1}$$

heißt (allgemeines, nichtlineares) Nullstellenproblem*; eine Lösung heißt* Nullstelle *von* Φ.
Das Problem

$$\text{“Finde } x \in U \text{ mit } \Phi(x) = x\text{”} \tag{3.2}$$

heißt Fixpunktproblem*; eine Lösung heißt* Fixpunkt *von* Φ.
Ferner betrachten wir noch die allgemeine nichtlineare Gleichung (“Gleichungssystem” im Fall $X = \mathbb{R}^n$*)*

$$\text{“Finde } x \in U \text{ mit } \Phi(x) = b\text{”} \tag{3.3}$$

Die Probleme (3.1), (3.2), (3.3) sind “gleich schwer” bzw. “äquivalent” in dem Sinne, dass jedes Problem des einen Typus in ein Problem jedes anderen Typus umgewandelt werden kann:

(3.1) $\to$ (3.2):	Setze z.B.: $\tilde{\Phi}(x) := \Phi(x) + x$
	(auch $\tilde{\Phi}(x) := \alpha\Phi(x) + x,\ \alpha \in \mathbb{R} \setminus \{0\}$ ist möglich)
(3.2) $\to$ (3.3):	Setze z.B.: $\tilde{\Phi}(x) := \Phi(x) - x + b$
(3.3) $\to$ (3.1):	Setze z.B.: $\tilde{\Phi}(x) := \Phi(x) - b$

Daher können wir uns im folgenden zunächst auf Fixpunktprobleme konzentrieren. Im allgemeinen Fall (wenn also f echt nichtlinear ist) gibt es keinen Algorithmus, der nach endlich vielen Schritten (3.2) löst. Wir werden stattdessen ein *iteratives* Lösungsverfahren der Form

$$x^{(n+1)} := \Phi(x^{(n)})$$

herleiten. Wichtigstest Hilfsmittel: Der Fixpunktsatz vom Banach.

3.1 Fixpunktiterationen

Satz 3.2 *(Fixpunktsatz von Banach, 1922) Sei* $(X, \|\cdot\|)$ *ein Banach-Raum, sei* $U \subset X$ *eine abgeschlossene Teilmenge; Sei* $\Phi : U \to X$ *mit*

$$\Phi(U) \subseteq U \quad [\Phi \textit{ heißt } \text{selbstabbildend}]. \tag{3.4}$$

Es existiere ein $0 < k < 1$ *so dass*

$$\|\Phi(x) - \Phi(y)\| \le k\|x - y\|\ \forall x, y \in U \tag{3.5}$$

[“Kontraktionseigenschaft”].
Dann gilt:

1. *Es gibt* genau einen *Fixpunkt* $x^* \in U$ *von* Φ.

2. *Für* beliebigen *Startwert* $x^{(0)} \in U$ *konvergiert die Folge*

$$x^{(n+1)} := \Phi(x^{(n)}) \tag{3.6}$$

gegen x^*.

3. *Es gelten*

$$\|x^{(n)} - x^*\| \leq \frac{k}{1-k}\|x^{(n)} - x^{(n-1)}\| \quad \textit{a posteriori-Fehlerabschätzung} \tag{3.7}$$

$$\leq \frac{k^n}{1-k}\|x^{(1)} - x^{(0)}\| \quad \textit{a priori-Fehlerabschätzung} \tag{3.8}$$

Bemerkung 3.3 *:*

- *(3.6) heißt* Fixpunktiteration*, k heißt* Kontraktionskonstante*; der Wert von k ist entscheidend für die Konvergenzgeschwindigkeit.*
- *Oft wird der Satz mit $U = X$ angewendet. (3.4) ist dann trivialerweise erfüllt.*
- *Der Satz lässt sich auf vollständige metrische Räume übertragen.*
- *Über die Probleme aus Definition 3.1 im $\mathbb{R}^n$ hinaus wird der Satz auch z.B. angewendet, um die Existenz und Eindeutigkeit von Lösungen von* Anfangswertproblemen von gewöhnlichen Differenzialgleichungen $y'(t) = f(t, y(t))$, $y(t) = y$*, bei Lipschitzstetigem f, zu zeigen; dabei* $(X, \|\cdot\|) = (\mathcal{C}([a,b]), \|\cdot\|_\infty)$

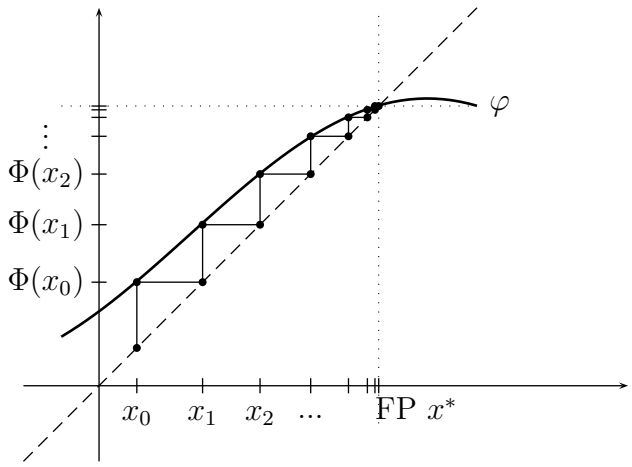


Abbildung 3: konvergente Fixpunktiteration, $X = \mathbb{R}$

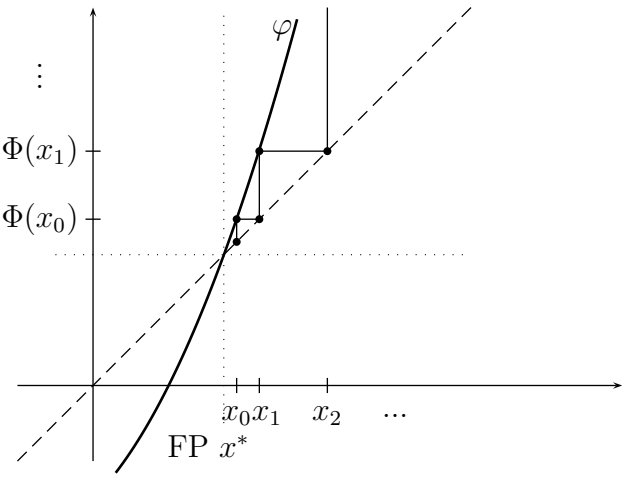


Abbildung 4: divergente Fixpunktiteration, $X = \mathbb{R}$

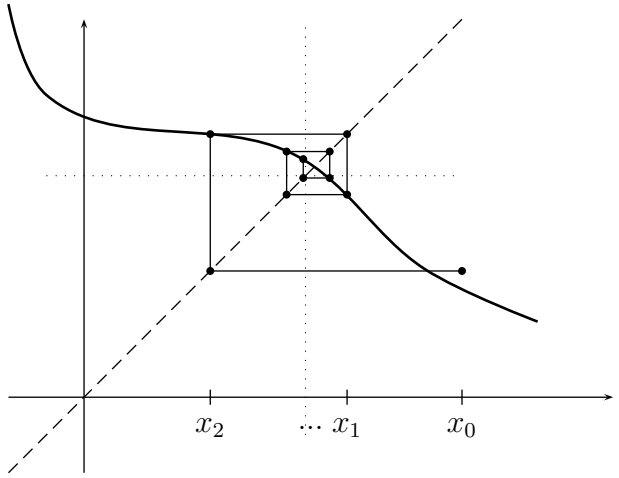


Abbildung 5: konvergente Fixpunktiteration, $X = \mathbb{R}$

Beweis : Die Folge $(x^{(n)})$ ist wegen (3.4) wohldefiniert. Wir zeigen: $(x^{(n)})_{n\in\mathbb{N}}$ ist CAUCHY-Folge:

$$
\begin{aligned}
\|x^{(n+1)} - x^{(n)}\| &\underset{\text{Def. Folge}}{=} \|\Phi(x^{(n)}) - \Phi(x^{(n-1)})\| \underset{\text{Kontraktion}}{\leq} k\|x^{(n)} - x^{(n-1)}\| \\
&\underset{\text{Def. Folge}}{\leq} k\|\Phi(x^{(n-1)}) - \Phi(x^{(n-2)})\| \underset{\text{Kontraktion}}{\leq} k^2\|x^{(n-1)} - x^{(n-2)}\| \\
&\ldots \\
&\leq k^n\|x^{(1)} - x^{(0)}\| \qquad (3.9) \\
\Rightarrow \|x^{(n+l)} - x^{(n)}\| &\underset{\text{Teleskopsumme}}{\leq} \|x^{(n+l)} - x^{(n+l-1)}\| + \ldots + \|x^{(n+1)} - x^{(n)}\| \\
&\underset{(3.9)}{\leq} (k^{n+l-1} + \ldots + k^n)\|x^{(1)} - x^{(0)}\| \\
&\leq k^n \left(\sum_{i=0}^{\infty} k^i\right) \|x^{(1)} - x^{(0)}\| \\
&= \underbrace{k^n \frac{1}{1-k}\|x^{(1)} - x^{(0)}\|}_{\leq \epsilon \text{ für } n \text{ hinreichend groß}} \Rightarrow \text{CAUCHY-Eigenschaft}
\end{aligned}
$$

Da $(X, \|\cdot\|)$ vollständig und $U \subset X$ abgeschlossen, ist $(x^{(n)})$ konvergent gegen ein $x^* \in U$. Wir zeigen: x^* ist Fixpunkt:

$$x^* = \lim_{n\to\infty} x^{(n)} \underset{\text{Def. Folge}}{=} \lim_{n\to\infty} \Phi(x^{(n-1)}) \underset{\Phi \text{ stetig}}{=} \Phi(\lim_{n\to\infty} x^{(n-1)}) = \Phi(x^*) \;\Rightarrow\; x^* \text{ist Fixpunkt}$$

Eindeutigkeit: Angenommen $x^*, x^{**} \in U$ seien Fixpunkte, dann:

$$\Rightarrow \quad \|x^* - x^{**}\| \underset{\text{Fixpunkt}}{=} \|\Phi(x^*) - \Phi(x^{**})\| \underset{\text{Kontraktion}}{\leq} k\|x^* - x^{**}\|$$

$$\underset{k<1}{\Rightarrow} \quad \|x^* - x^{**}\| = 0 \quad \Rightarrow \quad x^* = x^{**}$$

Zu den Abschätzungen (3.7) und (3.8):

$$\begin{aligned} \|x^{(n)} - x^*\| &\underset{\text{Def. Folge}}{=} \|\Phi(x^{(n-1)}) - \Phi(x^*)\| \underset{\text{Kontraktion}}{\leq} k\|x^{(n-1)} - x^*\| \qquad (3.10)\\ &\leq k \cdot (\|x^{(n-1)} - x^{(n)}\| + \|x^{(n)} - x^*\|) \\ \Rightarrow \|x^{(n)} - x^*\| &\leq \frac{k}{1-k}\|x^{(n-1)} - x^{(n)}\| \qquad \rightsquigarrow (3.7) \\ &\underset{(3.9)}{\leq} \frac{k^n}{1-k}\|x^{(1)} - x^{(0)}\| \qquad \rightsquigarrow (3.8) \end{aligned}$$

□

Lemma 3.4 *Eine stetig differenzierbare Abbildung $\Phi : U \to R$, $U \subset \mathbb{R}$ offenes Intervall, ist genau dann Lipschitz-stetig mit Lipschitz-Konstante $k > 0$, d.h.*

$$|\Phi(x) - \Phi(y)| \leq k|x - y| \quad \forall x, y \in U, \tag{3.11}$$

wenn

$$|\Phi'(x)| \leq k \quad \forall x \in U. \tag{3.12}$$

Beweis:

„⇐“ Es gelte (3.12). Der Mittelwertsatz liefert: $\Phi(x) = \Phi(y) + \Phi'(\zeta)(x - y)$, $\zeta \in U$.

$$\Rightarrow \quad |\Phi(x) - \Phi(y)| \leq |\Phi'(\zeta)| \cdot |x - y| \leq k \cdot |x - y|$$

„⇒“ Sei (3.12) falsch, es gibt also $x \in U$ mit $|\Phi'(x)| > k$. Dann gibt es (da f' stetig) eine zusammenhängende Umgebung $\tilde{U} \subseteq U$ von x, so dass $|\Phi'(\zeta)| > k \;\forall \zeta \in \tilde{U}$.

$$\Rightarrow \quad \text{für} \quad y \in \tilde{U},\ y \neq x \ |\Phi(x) - \Phi(y)| = |f'(\zeta)| \cdot |x - y| > k|x - y|$$

□

Für *stetig differenzierbares* $\Phi : U \to \mathbb{R}$ kann die Bedingung (3.5) also durch die einfacher überprüfbare Bedingung

$$k := \sup_{x\in U} |\Phi'(x)| < 1 \tag{3.13}$$

ersetzt werden.

Beispiel 3.5 *Gesucht sind die Nullstellen von $f(x) = \cos x - 2x$ also $X = \mathbb{R}$, $U = \mathbb{R}$. Umwandlung in ein Fixpunktproblem:*

erster Versuch:

$$\begin{aligned} \cos x - 2x &\overset{!}{=} 0 \quad | \quad +x \\ \underbrace{\cos x - x}_{=:\Phi(x)} &= x \end{aligned}$$

Prüfe die Kontraktionseigenschaft. Wegen $\Phi \in \mathcal{C}^1(\mathbb{R}) \Rightarrow$ Lemma 3.4 ist anwendbar:

$$\Phi'(x) = -(\sin x + 1)$$

Erfüllt *nicht* (3.13)! Problem! Kann diese Problem durch Wahl von kleinerem $U \subsetneq \mathbb{R}$ behoben werden?
Wir erwarten Nullstelle von f im Bereich $[0, \frac{\pi}{2}]$, da $f(0) = 1 > 0,\ f(\frac{\pi}{2}) = -\pi < 0$. Aber selbst auf $[0, \frac{\pi}{2}] =: U$ ist $|\Phi'(x)| \geq 1$, da sin dort größer gleich 0 ist.

neuer Versuch:

$$f(x) = \cos x - 2x \overset{!}{=} 0 \qquad | \cdot \alpha,\ \alpha \neq 0,\ +x \quad \Leftrightarrow \quad \underbrace{\alpha(\cos x - 2x) + x}_{=:\Phi(x)} = x$$

Kontraktionseigenschaft :

$$\Phi'(x) = \overbrace{-\alpha \sin x}^{\in[-\alpha,\alpha]} - 2\alpha + 1 \in [1 - 3\alpha, 1 - \alpha] \overset{!}{\subseteq} (-1, 1)$$

Wähle zum Beispiel $\alpha := \frac{1}{2}$. Dann ist $\Phi'(x) \in [-\frac{1}{2}, \frac{1}{2}]$ und damit gilt für die Kontraktionskonstante

$$k = \sup_{x \in U} |\Phi'(x)| \leq \frac{1}{2}$$

($\alpha \in (0, \frac{2}{3})$, damit $k < 1$ gilt).
Der Fixpunktsatz von Banach liefert also: f hat genau eine Nullstelle x^*.
Die Folge $x^{(n+1)} := \Phi(x^{(n)}) = \frac{1}{2} \cos x^{(n)}$ konvergiert gegen x^* mit Kontraktionsrate $k = \frac{1}{2}$ ($\sim$ ca. 3 Iterationsschritte pro gewonnene Dezimalstelle), bei beliebigem Startwert $x^{(0)} \in U = \mathbb{R}$.

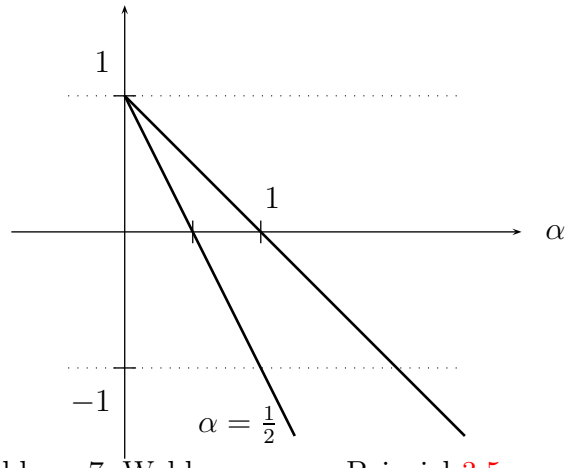


Abbildung 7: Wahl von α aus Beispiel 3.5, zweiter Versuch

3.2 Konvergenzordnung von Fixpunktverfahren und das Newton-Verfahren für skalare Nullstellenprobleme

Im Beispiel 3.5 haben wir das Nullstellenproblem $f(x) = 0$ umgewandelt in ein Fixpunktproblem $\Phi(x) = x$. Wir haben

$$\Phi(x) := x + \alpha f(x) \tag{3.14}$$

gewählt. Wie ist $\alpha \in \mathbb{R},\ \alpha \neq 0$, allgemein zu wählen? Für $\alpha := 1$ wäre die „Kontraktionsrate“

$$k := \sup_{x \in U = \mathbb{R}} |\Phi'(x)| > 1$$

$\Rightarrow$ Divergenz, für $\alpha := \frac{1}{2}$ ist $k = \frac{1}{2} < 1$. Welches α ist optimal? Oder, noch allgemeiner als (3.13): Wie soll Φ (in Abhängigkeit von f) gewählt werden? Bedingungen an Φ :

(a) **Konsistenz:** x^* soll genau dann Fixpunkt von Φ sein, wenn x^* Nullstelle von f ist.

(b) **schnelle Konvergenz:** $k := \sup\limits_{x \in U} |\Phi'(x)|$, wobei U eine Umgebung von x^* ist, sollte möglichst klein sein, bzw.

(b^*) **asymptotisch** für $x^{(k)} \approx x^*$, sollte $k^* := |\Phi'(x^*)|$, die sogenannte asymptotische Kontraktionskonstante, möglichst klein sein.

Wenn wir Φ von der Form (3.14) wählen, so ist (a) erfüllt; und für (b) bzw. (b^*) wäre $\alpha := \frac{1}{f'(x^*)}$ günstig (falls $f'(x^*) \neq 0$), da dann

$$\Phi(x) = x - \frac{1}{f'(x^*)} f(x) \quad \Rightarrow \quad \Phi'(x^*) = 1 - \frac{f'(x^*)}{f'(x^*)} = 0$$

(optimale asymptotische Kontraktionskonstante).

[Wir können erwarten, dass $\Phi'(x)$ auch in einer Umgebung U von x^* *klein* ist und damit die Kontraktionskonstante k klein ist.]

Aber: x^* ist ja a priori unbekannt, daher ist die Wahl $\alpha := -\frac{1}{f'(x^*)}$ unpraktikabel. Eine praktisch durchführbare Wahl, die um so besser ist, je näher $x^{(n)}$ bei x^* liegt, ist es, bei der Berechnung von $x^{(n+1)}$ aus $x^{(n)}$,

$$\alpha = \alpha(n) := -\frac{1}{f'(x^{(n)})}$$

zu wählen, also:

$$\begin{aligned} \Phi(x) &:= x - \frac{f(x)}{f'(x)} \\ x^{(n+1)} &:= \Phi(x^{(n)}) = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})} \end{aligned} \tag{3.15}$$

Dann ist

$$\Phi'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2},$$

und damit $\Phi'(x^*) = 0$ (falls $f'(x^*) \neq 0$).

Diese Iteration zur Bestimmung von Nullstellen von f heißt NEWTON-*Iteration*/ NEWTON-*Verfahren*. Nach obigen Überlegungen erwarten wir für das NEWTON-Verfahren besonders gute Konvergenzeigenschaften. Dies werden wir im folgenden genauer untersuchen (Def. 3.6 - Satz 3.10).

Zuvor noch eine *geometrische* Interpretation des NEWTON-Verfahrens (für skalare Probleme):
Sei $x^{(n)}$ eine Nährung für die Nullstelle x^* von f. Anstatt die Nullstelle von f direkt zu berechnen, wird f approximiert durch eine Funktion $\tilde{f}$, deren Nullstelle $\tilde{x}^*$ man *leicht* berechnen kann. $\tilde{x}^*$ wird als neue, bessere Approximation $x^{(n+1)}$ an die Nullstelle x^* von f verwendet. Als $\tilde{f}$ nehmen wir die *Tangente* (=Linearisierung von f) an f in der Stelle $x^{(n)}$ (setzt $f \in \mathcal{C}^1$ voraus, außerdem $f'(x^{(n)}) \neq 0$). Steigungsdreieck: $f'(x^{(n)}) = \frac{f(x^{(n)})-0}{x^{(n)}-x^{(n+1)}} \Leftrightarrow \quad x^{(n)} - x^{(n+1)} = \frac{f(x^{(n)})}{f'(x^{(n)})} \quad \Leftrightarrow$ (3.15)

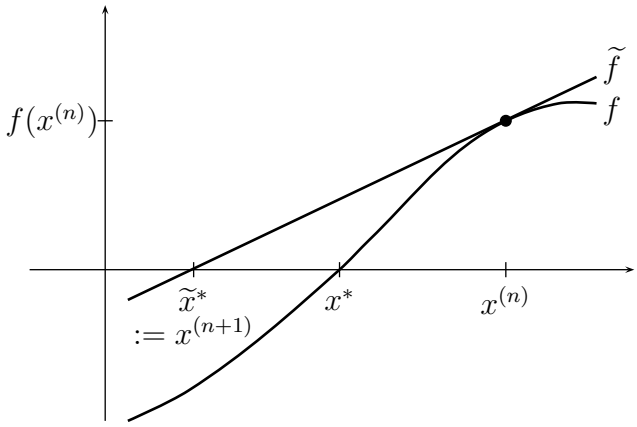


Abbildung 8: geometrische Interpretation des NEWTON-Verfahrens

Um das Konvergenzverhalten des NEWTON-Verfahrens beschreiben zu können, sind folgende Definitionen nützlich:

Definition 3.6 *Sei $(X, \|\cdot\|)$ ein normierter Vektorraum, $U \subset X$ offen. Eine Folge $x^{(n+1)} := \Phi(x^{(n)})$, $\Phi : U \to U$ heißt* lokal konvergent *gegen $x^* \in U$, falls es eine Umgebung $V \subseteq U$ von x^* gibt, so dass für* beliebigen *Startwert $x^{(0)} \in V$ die Folge gegen x^* konvergiert. Die Folge heißt* global konvergent*, falls für* beliebigen *Startwert $x^{(0)} \in U$ die Folge gegen x^* konvergiert.*

Wir wissen (Satz 3.2): Unter den Vorraussetzungen des Fixpunktsatzes von Banach ist die Fixpunktiteration *global konvergent*.

Definition 3.7 *Sei $(X, \|\cdot\|)$ ein normierter Vektorraum, $U \subseteq X$ offen, $x^* \in U$, $x^{(0)} \in U$, $x^{(n+1)} := \Phi(x^{(n)}), p \in \mathbb{R}, p \geq 1$. Falls es ein $c > 0$ gibt mit*

$$\|x^{(n+1)} - x^*\| \leq c\|x^{(n)} - x^*\|^p \quad \forall n \in \mathbb{N} \tag{3.16}$$

und, nur im Fall $p = 1$, zusätzlich $c < 1$ gilt, so heißt die Folge $(x^{(n)})$ (lokal) konvergent von der Ordnung p.

Im Fall $p = 1$ sprechen wir von (lokal) ***linearer*** *Konvergenz.*
Im Fall $p = 2$ sprechen wir von (lokal) ***quadratischer*** *Konvergenz.*
Im Fall $p = 3$ sprechen wir von (lokal) ***kubischer*** *Konvergenz.*

Bemerkung: In Büchern wird das Wort "lokal" in obiger Definition meist weggelassen. Das ist aber irreführend, denn die Bedingung (3.16) impliziert im Allgemeinen nicht für beliebige Startwerte $x^{(0)} \in U$ Konvergenz der Folge $(x^{(n)})$ (=„globale Konvergenz"), sondern nur für $x^{(0)}$ hinreichend nahe bei x^* (= „lokale Konvergenz") (z.B. für $\|x^{(0)} - x^*\| < 1$, wenn $c \leq 1$) siehe dazu auch das folgende Lemma 3.8.

Wir wissen (siehe (3.10) im Beweis des Fixpunktsatzes von Banach): Unter den Vorraussetzungen des Fixpunktsatzes ist die Fixpunktiteration global konvergent *von der Ordnung mindestens* 1.

Lemma 3.8 *Sei $(X, \|\cdot\|)$ normierter Vektorraum, $U \subset X$ offen, $\Phi : U \to U$ stetig, $x^{(0)} \in U$ gegeben, $x^{(n+1)} := \Phi(x^{(n)})$, $x^* \in U$, $p \geq 1$, $c > 0$, sowie $c < 1$ falls $p = 1$. Es gelte*

$$\|\Phi(x) - x^*\| \leq c\|x - x^*\|^p \quad \forall x \in U \tag{3.17}$$

Dann konvergiert die Iteration lokal *(d.h. es gibt eine Umgebung $\tilde{U} \subset U$ von x^* und für alle $x^{(0)} \in \tilde{U}$ konvergiert sie) gegen x^* mit Konvergenzordnung mindestens p, und x^* ist Fixpunkt von Φ.*

Beweis : Wähle $r > 0$ so dass die Kugel $B_r(x^*) \subseteq U$ und

$$c \cdot r^{p-1} =: k < 1 \tag{3.18}$$

Es ist $x^{(n)} \in U$ $\forall n \in \mathbb{N}$, und für $x^{(n)} \in B_r(x^*)$ gilt

$$d^{(n+1)} := \|x^{(n+1)} - x^*\| = \|\Phi(x^{(n)}) - x^*\| \underset{(3.17)}{\leq} c\|x^{(n)} - x^*\|^p \underset{x^{(n)} \in B_r(x^*)}{\leq} cr^p \underset{(3.18)}{<} r \tag{3.19}$$

d.h. auch $x^{(n+1)} \in B_r(x^*)$. Falls also der Anfangswert $x^{(0)} \in B_r(x^*) =: \tilde{U}$ gewählt wird, ist die gesamte Folge $(x^{(n)})$ in $\tilde{U}$. Weiter gilt

$$d^{(n+1)} \underset{(3.19)}{\leq} c \cdot \underbrace{\|x^{(n)} - x^*\|^{p-1}}_{\leq r \text{ da } x^{(n)} \in \tilde{U}} \underbrace{\|x^{(n)} - x^*\|}_{= d^{(n)}} \leq cr^{p-1} d^{(n)} \underset{(3.18)}{=} kd^{(n)}, \; k < 1$$

$\Rightarrow x^{(n)} \to x^*$, $x^* = \lim\limits_n x^{(n)} = \lim\limits_n \Phi(x^{(n-1)}) = \Phi(\lim\limits_n x^{(n-1)}) = \Phi(x^*)$
Die Konvergenzordnung ist (mindestens) p nach (3.19). □

Wir wissen: Fixpunktiterationen konvergieren unter den Voraussetzungen von Satz 3.2 *mindestens linear*. Aus Lemma 3.8 folgt der folgende Satz 3.9, der uns sagt, unter welcher Voraussetzungen an Φ wir sogar Konvergenz p-ter Ordnung der Folge $x^{(n+1)} = \Phi(x^{(n)})$ haben; vgl. die Konstruktionsversuche von Φ am Anfang von Kapitel 3.2, wo wir versucht haben, möglichst gute Konvergenz zu erreichen und so die Wahl $\Phi(x) := x - \frac{f(x)}{f'(x)}$ zur Lösung von $f(x) = 0$ motiviert haben.

Satz 3.9 *(Konvergenzordnung von Fixpunktverfahren)*
Sei $X = \mathbb{R}$, $U \subset \mathbb{R}$ offen, $\Phi : U \to U$ p-mal stetig differenzierbar. Sei $x^* \in U$ Fixpunkt von Φ, sei $p \in \mathbb{N}$. Ist

$$\Phi'(x^*) = \ldots = \Phi^{(p-1)}(x^*) = 0, \; \Phi^{(p)}(x^*) \neq 0 \quad \textit{im Fall } p > 1$$

bzw.

$$\Phi'(x^*) \neq 0, \; |\Phi'(x^*)| < 1 \quad \textit{im Fall } p = 1$$

dann konvergiert die durch Φ definierte Fixpunktiteration $x^{(n+1)} = \Phi(x^{(n)})$ lokal *gegen x^** genau mit Ordnung *p, und x^* ist Fixpunkt von Φ.*

Beweis : TAYLOR-Entwicklung von Φ um x^*:

$$\Phi(x) = \underbrace{\Phi(x^*)}_{=x^*} + \frac{\Phi^{(p)}(\zeta)}{p!}(x-x^*)^p \quad \forall x \in U \text{ mit } \zeta = \zeta(x) \in U \quad | - x^* \tag{3.20}$$

$\Rightarrow \quad |\Phi(x) - x^*| \leq c|x-x^*|^p$ mit $c := \max\limits_{\zeta \in \tilde{U}} \frac{|\Phi^{(p)}(\zeta)|}{p!}$, $\tilde{U} \subset U$ kompakte Umgebung von x^*

$\Rightarrow$ Lemma 3.8 liefert lokale Konvergenz der Ordnung mindestens p. Angenommen die Ordnung sei höher als p. Dann müsste (3.17) für ein $\tilde{p} > p$ statt p gelten:

$$|\Phi(x) - x^*| \leq c|x-x^*|^{\tilde{p}} \quad \forall x \in U \tag{3.21}$$

Dies kann nicht sein, da

$$\Phi^{(p)}(x^*) \underset{\text{nach Vor.}}{\neq} 0, \text{ also } \Phi^{(p)}(\zeta) \neq 0$$

in einer Umgebung $\tilde{U}$ von x^* in (3.20), also

$$|\Phi(x) - x^*| \geq \min_{\zeta \in \tilde{U}} |\Phi^{(p)}(\zeta)| \; |x-x^*|^p \; \forall x \in \tilde{U},$$

was ein Widerspruch zu (3.21) ist. □

Satz 3.9 im Fall $p = 2$ besagt: Wir bekommen *quadratische* Konvergenz (als Verbesserung der linearen Konvergenz, die allgemein in der Situation des Fixpunktsatzes von Banach gilt) genau dann wenn

$$\Phi'(x^*) = 0 \quad , \quad \Phi''(x^*) \neq 0.$$

Dies wird in der Tat vom NEWTON-Verfahren

$$\Phi(x) = x - \frac{f(x)}{f'(x)}, \; x^{(n+1)} = \Phi(x^{(n)}), \tag{3.22}$$

wenn $f'(x^*) = 0$, erfüllt:

Satz 3.10 *Sei $f \in \mathcal{C}^3(\mathbb{R})$ und x^* eine* einfache *Nullstelle von f (also $f'(x^*) \neq 0$). Dann ist das* NEWTON*-Verfahren lokal konvergent, mindestens von Ordnung 2.*

Beweis : Wegen der Stetigkeit existiert eine offene Umgebung V von x^*, auf der $f'(x) \neq 0$ ist, d.h Φ aus (3.22) ist wohldefiniert und stetig auf V, es ist

$$\Phi(x^*) = x^* - \frac{\overbrace{f(x^*)}^{=0}}{f'(x^*)} = x^*$$

(*Konsistenz* des Nullstellenproblems $f(x) = 0$ mit dem Fixpunktproblem $\Phi(x) = x$), es ist

$$\Phi'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}, \text{ also } \Phi'(x^*) = 0,$$

und $\Phi''(x)$ existiert und ist stetig.
Nach Satz 3.9 folgt die Behauptung. □

Zur Vorgehensweise bei *mehr*facher Nullstelle (siehe Übung): Setze $\Phi(x) := x - \gamma\frac{f(x)}{f'(x)}$ wobei $\gamma = p$ die Vielfachheit der Nullstelle ist. Hier für $p = 2$ (= doppelte Nullstellen), also $f(x^*) = f'(x^*) = 0, f''(x^*) \neq 0$ hergeleitet:

$$\begin{aligned}\Phi(x) &:= x - \gamma\frac{f(x)}{f'(x)}, \quad x \neq x^* \\ \lim_{x\to x^*}\Phi(x) &= x - \gamma\lim_{x\to x^*}\frac{f'(x)}{f''(x)} = x^*\end{aligned}$$

$\Rightarrow$ Mit der Setzung $\Phi(x^*) := x^*$ wird die Definitionslücke von Φ stetig geschlossen.

$$\Phi'(x) = 1 - \gamma\frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = 1 - \gamma + \gamma\frac{f(x)f''(x)}{f'(x)^2}, \quad x \neq x^*$$

Es folgt mit L'Hopital:

$$\begin{aligned}\Phi'(x^*) &= \lim_{x\to x^*}\frac{f(x) - f(x^*)}{x - x^*} = \lim_{x\to x^*}\frac{x - \gamma\frac{f(x)}{f'(x)} - x^*}{x - x^*} = 1 - \gamma\lim_{x\to x^*}\frac{f(x)}{f'(x)(x - x^*)} \\ &= 1 - \frac{\gamma}{\lim\limits_{x\to x^*}\frac{f'(x)(x-x^*)}{f(x)}} = 1 - \frac{\gamma}{\lim\limits_{x\to x^*}\frac{f''(x)(x-x^*)+f'(x)}{f'(x)}} \\ &= 1 - \frac{\gamma}{1 + f''(x^*)\lim\limits_{x\to x^*}\frac{x-x^*}{f'(x)}} = 1 - \frac{\gamma}{1 + f''(x^*)\lim\limits_{x\to x^*}\frac{1}{f''(x)}} \\ &= 1 - \frac{\gamma}{2}\end{aligned}$$

sowie:

$$\begin{aligned}\lim_{x\to x^*}\Phi'(x) &= 1 - \gamma + \gamma f''(x^*)\lim_{x\to x^*}\frac{f(x)}{f'^2(x)} = 1 - \gamma + \gamma f''(x^*)\cdot\lim\frac{f'(x)}{2f'(x)f''(x)} \\ &= 1 - \gamma + \gamma f''(x^*)\cdot\frac{1}{2f''(x^*)} = 1 - \frac{\gamma}{2}.\end{aligned}$$

Also: Φ ist stetig differenzierbar an der Stelle $x = x^*$, und die Forderung

$$\Phi'(x^*) = 1 - \frac{\gamma}{2} \stackrel{!}{=} 0$$

erfordert $\gamma := 2$.

Fazit soweit: Das Newton-Verfahren zur Nullstellenbestimmung konvergiert nur dann sicher gegen eine Nullstelle x^*, wenn der Startwert hinreichend nahe bei x^* liegt. Der Konvergenzbereich kann sehr klein sein, Insbesondere wenn Extrem- und Wendepunkte in der Nähe von x^* (zwischen x^* und $x^{(0)}$) liegen: Um sich eine bessere Startlösung $x^{(0)}$ fürs Newton-Verfahren zu verschaffen, kann man zunächst einige Schritte eines anderen Verfahrens (z.B. Bisektion, s. Kapitel 3.4) vorab durchführen, oder gegebenenfalls ein "gedämpftes Newton-Verfahren" durchführen (s. Kapitel 3.3).

Wenn das Newton-Verfahren konvergiert, dann extrem schnell. (**lokal quadratische** Konvergenz)

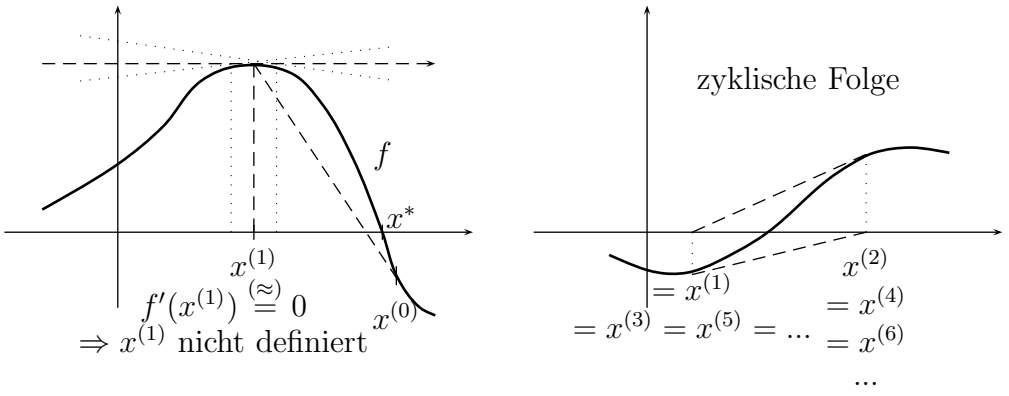


Abbildung 9: NEWTON-Verfahren bei Extrem- und Wendepunkten

3.3 Das Bisektions- und das Sekantenverfahren

Als Alternative zum NEWTON-Verfahren für skalare Nullstellenprobleme gibt es das *Bisektionsverfahren* (= *Intervallhalbierungsverfahren*) sowie das *Sekantenfahren*.

Bisektionsverfahren : Sei $f : \mathbb{R} \to \mathbb{R}$ stetig, sei $x^{(0)} < y^{(0)}$ bekannt mit

$$f(x^{(0)}) \cdot f(y^{(0)}) \leq 0$$

$\Rightarrow f$ besitzt eine Nullstelle im Intervall $I^{(0)} := [x^{(0)}, y^{(0)}]$. Man zerteilt das Intervall in

$$[x^{(0)}, m^{(0)}] \text{ und } [m^{(0)}, y^{(0)}],\ m^{(0)} := \frac{x^{(0)} + y^{(0)}}{2}\ .$$

In mindestens einem der beiden Intervalle muss eine Nullstelle liegen, man testet also: Falls

$$f(x^{(0)}) \cdot f(m^{(0)}) < 0,$$

dann

$$x^{(1)} := x^{(0)},\ y^{(1)} := m^{(0)},$$

andernfalls

$$x^{(1)} := m^{(0)}\ y^{(1)} := y^{(0)},$$

und setzt

$$I^{(1)} := [x^{(1)}, y^{(1)}]$$

dies wird iteriert. $\Rightarrow$ Jedes der Intervalle $I^{(n)}$ enthält eine Nullstelle, es ist

$$|I^{(n)}| = |y^{(n)} - x^{(n)}| = \left(\frac{1}{2}\right)^n \cdot |I^{(0)}|$$

d.h. wir wissen

$$\begin{aligned} |x^* - x^{(n)}| &\leq \left(\frac{1}{2}\right)^n |I^0|, \\ |x^* - y^{(n)}| &\leq \left(\frac{1}{2}\right)^n |I^0|. \end{aligned}$$

Deswegen ist das Verfahren (nur) *linear* konvergent. Es wird *Bisektions-* oder *Intervallhalbierungsverfahren* genannt.

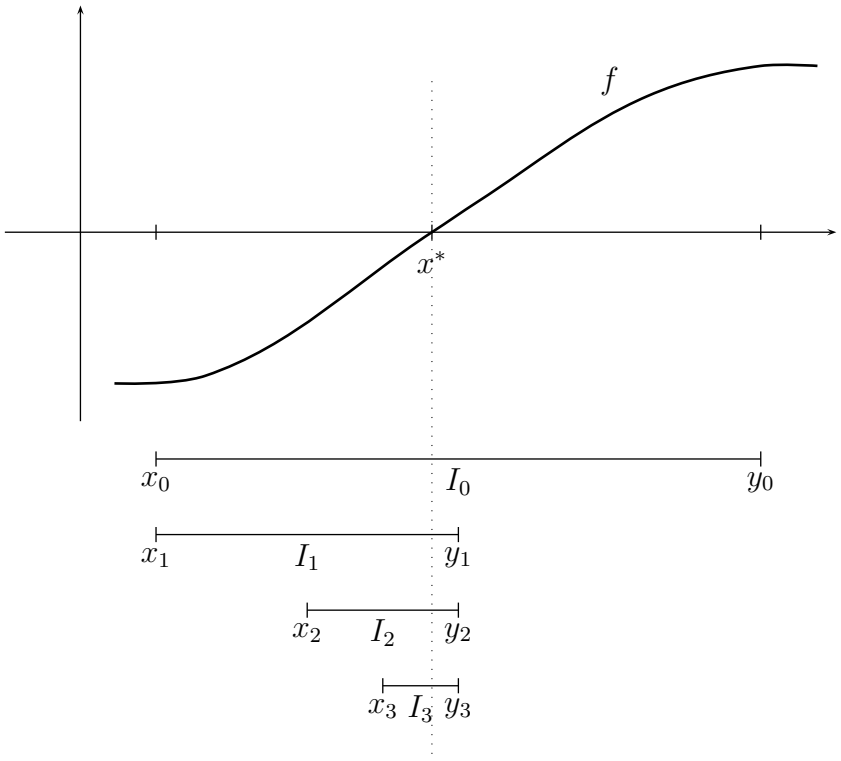


Abbildung 10: Bisektionsverfahren

Vorteile:

- Konvergiert *immer* (*globale* Konvergenz), sobald $x^{(0)}, y^{(0)}$ mit $f(x^{(0)}) \cdot f(y^{(0)}) < 0$ bekannt.
- Braucht kein f'; Stetigkeit von f reicht.
- Liefert *Einschließung* der Lösung.

Nachteile:

- Nur linear konvergent.
- Nicht auf $X = \mathbb{R}^n$ zu verallgemeinern (vgl. Kapitel 3.4: NEWTON-Verfahren im $\mathbb{R}^n$)

Das Sekantenverfahren : Man ersetzt im NEWTON-Verfahren die Ableitung $f'(x^{(n)})$ durch einen Differenzenquotienten $\frac{f(x^{(n)})-f(x^{(n-1)})}{x^{(n)}-x^{(n-1)}}$, also

$$\begin{aligned} x^{(n+1)} &= x^{(n)} - \frac{f(x^{(n)})}{\frac{f(x^{(n)})-f(x^{(n-1)})}{x^{(n)}-x^{(n-1)}}} = \frac{x^{(n)}(f(x^n)) - f(x^{(n-1)})}{f(x^{(n)}) - f(x^{(n-1)})} - \frac{f(x^{(n)})(x^{(n)} - x^{(n-1)})}{f(x^{(n)} - f(x^{(n-1)}))} = \\ &= \frac{x^{(n-1)} f(x^{(n)}) - x^{(n)} f(x^{(n-1)})}{f(x^{(n)}) - f(x^{(n-1)})} \end{aligned} \tag{3.23}$$

Es hängt $x^{(n+1)}$ also nicht nur von $x^{(n)}$, sondern auch von $x^{(n-1)}$ ab es handelt sich um ein *Zweischrittverfahren*. Ein solches hat allgemein die Form:

$$x^{(n+1)} = \Phi(x^{(n-1)}, x^{(n)}), \ \Phi : \mathbb{R}^2 \to \mathbb{R} \text{ (bzw. } \Phi : X^2 \to X \text{ im allgemeinen Fall)}$$

Man benötigt also 2 Startwerte $x^{(0)}, x^{(1)}$. Beim Sekantenverfahren ist offensichtlich $f(x^{(0)}) \neq f(x^{(1)})$ erforderlich. Rein formal kann man jedes Zweischrittverfahren in $\mathbb{R}$ (bzw X) als ein *Ein*schrittverfahren in $\mathbb{R}^2$ (bzw. X^2) auffassen, indem man jeweils Paare von aufeinanderfolgenden Iterationen zusammenfasst und den Übergang

$$\tilde{x}^{(n)} = (x^{(n-1)}, x^{(n)}) \to (x^{(n)}, x^{(n+1)}) = \tilde{x}^{(n+1)}$$

durch ein

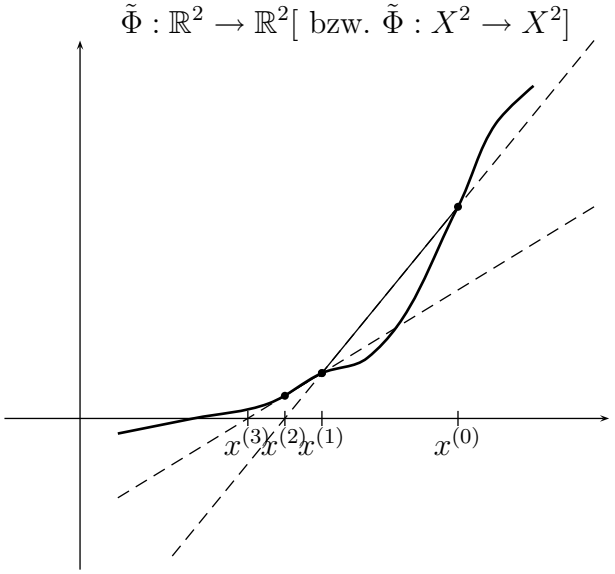


Abbildung 11: geometrische Interpretation des Sekantenverfahrens

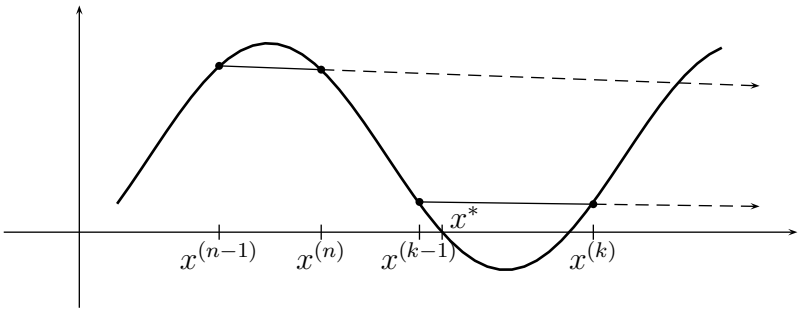


Abbildung 12: Probleme bei lokalen Extrema

Lemma 3.11 *Sei $f \in \mathcal{C}^2(U)$, $U \subset \mathbb{R}$ offen, $x^* \in U$ einfache Nullstelle, $x^{(0)} \neq x^{(1)}$. Dann ist das Sekantenverfahren lokal konvergent gegen x^* mit Konvergenzordnung*

$$p = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618 \qquad (\text{„goldener Schnitt“, Lösung von } p^2 = p + 1) \, .$$

Beweisskizze : (nur für Konvergenz*ordnung* und Wohldefiniertheit)

Da $f'(x^*) \neq 0$, ist f in einer Umgebung $\tilde{U} \subseteq U$ von x^* streng monoton, also injektiv. Wir gehen davon aus, dass wir ein $n_0 \in \mathbb{N}$ haben mit $x^{(n)} \in \tilde{U}$ $\forall n \geq n_0$ (ohne Beweis). Aus

$$x^{(n)} \neq x^{(n-1)}$$

folgt

$$x^{(n+1)} = x^{(n)} - \frac{\overbrace{(x^{(n)} - x^{(n-1)})}^{\neq 0} f(x^{(n)})}{f(x^{(n)}) - f(x^{(n-1)})} \neq x^{(n)}$$

(es sei denn, $f(x^{(n)}) = 0$, d.h. $x^{(n)} = x^*$, in welchem Falle man abbrechen kann). Per Induktion folgt:

$$x^{(n+1)} \neq x^{(n)} \ \forall n \geq n_0$$

(bis ggf. zum Abbruch $x^{(n)} = x^*$).
Sei $e^{(n)} := x^{(n)} - x^*$ der Fehler. Subtraktion von x^* von (3.23) ergibt:

$$\begin{aligned} e^{(n+1)} &= x^{(n+1)} - x^* = \frac{x^{(n-1)} f(x^{(n)}) - x^{(n)} f(x^{(n-1)})}{f(x^{(n)}) - f(x^{(n-1)})} - x^* \\ &= \frac{e^{(n-1)} f(x^{(n)}) - e^{(n)} f(x^{(n-1)})}{f(x^{(n)}) - f(x^{(n-1)})}. \end{aligned} \tag{3.24}$$

Unter der Annahme $|e^{(n)}|,\ |e^{(n-1)}| \ll 1$ bekommen wir näherungsweise mittels Taylor-Entwicklung:

$$\begin{aligned} \text{Nenner} &\approx f(x^*) + f'(x^*)(x^{(n)} - x^*) - [f(x^*) + f'(x^*)(x^{(n-1)} - x^*)] \\ &= f'(x^*)(e^{(n)} - e^{(n-1)}) \\ \text{Zähler} &\approx e^{(n-1)} \cdot \left[\underbrace{f(x^*)}_{=0} + f'(x^*) e^{(n)} + f''(x^*) \frac{e^{(n)^2}}{2} \right] \\ &\quad - e^{(n)} \left[\underbrace{f(x^*)}_{=0} + f'(x^*) e^{(n-1)} + f''(x^*) \frac{e^{(n-1)^2}}{2} \right] \\ &= \frac{1}{2} e^{(n-1)} e^{(n)} (e^{(n)} - e^{(n-1)}) f''(x^*) \qquad [\text{Annahme: } f''(x^*) \neq 0] \\ \Rightarrow \quad e^{(n+1)} &\approx \frac{f''(x^*)}{2f'(x^*)} e^{(n-1)} e^{(n)} \end{aligned}$$

Insbesondere gilt

$$e^{(n+1)} \leq \underbrace{\left(\left| \frac{f''(x^*)}{2f'(x^*)} \right| + \epsilon \right)}_{=:c} e^{(n-1)} e^{(n)} \ .$$

Sei $E_{n_0} \geq |e^{(n_0)}|,\ E_{n_0+1} \geq |e^{(n_0+1)}|,\ E_{n+1} \underset{(*)}{:=} c E_n E_{n-1}$ $(\Rightarrow |e^{(n)}| \leq E_n \ \forall n \geq n_0$, d.h. es reicht

E_n abzuschätzen)[2]. Der Ansatz $E_n \stackrel{!}{=} k \cdot E_{n-1}^p$ wird eingesetzt in die Rekursionsgleichung $(*)$:

$$\left.\begin{array}{r} \text{links } E_{n+1} = k \cdot E_n^p = k \cdot (kE_{n-1}^p)^p \\ \text{rechts } cE_nE_{n-1} = ckE_{n-1}^pE_{n-1} \end{array}\right\} \stackrel{!}{=} \quad \Rightarrow k^p E_{n-1}^{p^2} = cE_{n-1}^{p+1} \; \forall n \geq n_0.$$

Dies wird gelöst durch $k = \sqrt[p]{c}$, $p^2 = p + 1 \Rightarrow \; p = \frac{1}{2}(1 \pm \sqrt{5})$. Die negative Wurzel kann man ausschließen, indem man die "Anfangswerte" E_{n_0+1} und E_{n_0} so wählt, dass

$$E_{n_0+1} = \sqrt[\frac{1}{2}(1+\sqrt{5})]{c} E_{n_0}^{\frac{1}{2}(1+\sqrt{5})}.$$

□

Effizienzanalyse :

Durch Zwischenspeicherung der Werte $f(x^{(n)})$ kann man das Sekantenverfahren so formulieren, dass in jedem Iterationsschritt (außer dem ersten) nur *eine* Funktionsauswertung nötig ist. Beim NEWTON-Verfahren dagegen sind Auswertung von f und f' nötig.
Unter der Annahme, dass daher die Durchführung eines NEWTON-Schritts in etwa so lange dauert wie die Durchführung von zwei Schritten des Sekantenverfahrens ($x^{(n)} \to x^{(n+2)}$), ergibt sich: Fehlerordung eines Doppel-Schritts des Sekantenverfahrens: $p^2 = p + 1 \approx 2.61 > 2$!
In diesem Sinne kann das Sekantenverfahren durchaus effizienter sein als das NEWTON-Verfahren!

Bewertung :

+ sehr schnell;
+ braucht nur f, nicht f';
+ realistische a priori-Abschätzung verfügbar (Stummel & Hainer: Praktische Mathematik);
- keine globale Konvergenz, Probleme bei Extrema;
- Probleme bei mehrfachen Nullstellen möglich (siehe Übung);
- schwierig auf mehrdimensionale Probleme ($X = \mathbb{R}^n$) zu verallgemeinern.

Weitere Verfahren:

- **Regula Falsi:** s. Stummel & Hainer: Praktische Mathematik, Knabner-Skript;
  Kann als "Mischung" aus Sekanten und Bisektionsverfahren betrachtet werden. Startet mit $x^{(0)}, x^{(1)}$ so dass $f(x^{(0)}) \cdot f(x^{(1)}) < 0$, $x^{(2)}$ sei Nullstelle der Sekante durch $x^{(0)}, x^{(1)}$; liefert *Einschließung* der Lösung;
  aber Vorsicht: im Allgemeinen $|I^{(n)}| = |x^{(n+1)} - x^{(n)}| \not\to 0$, sondern nur $dist(x^*, \partial I^{(n)}) \to 0$ für $n \to \infty$, im Allgemeinen von erster Ordnung; nur historisch interessant;

[2]Rekursionsgleichungen der Form $x_n = b + \alpha_1 x_{n-1} + \ldots + \alpha_k x_{n-k}$ ($b, \alpha_1, \ldots, \alpha_{n-k}$ gegeben) werden auch Differenzengleichungen genannt; für diese gibt es eine Lösungstheorie, die die Struktur der Lösungsmenge charakterisiert. Gleichung $(*)$ hat, wenn logarithmiert, diese Struktur bzgl. der Unbekannten $\ln E_n$. Wir verwenden einen *Ansatz*, um Lösungen auch ohne theoretische Betrachtung zu bestimmen.

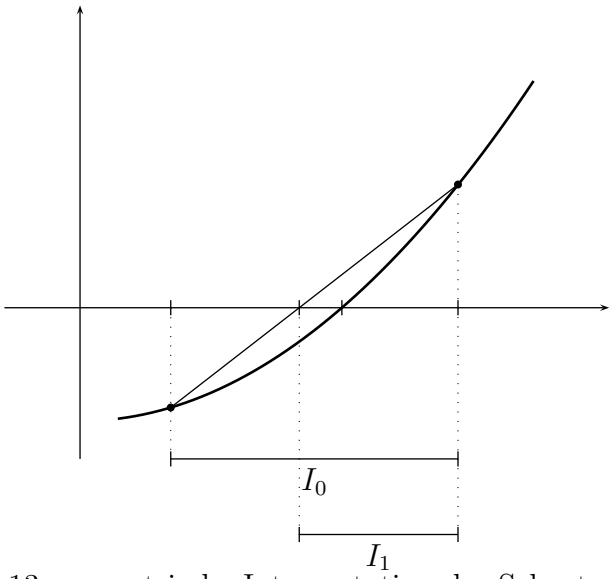


Abbildung 13: geometrische Interpretation des Sekantenverfahrens

- **Hybridverfahren**, wie z.B.: Starte mit einschließendem Intervall $x^* \in I^{(0)} = [a^{(0)}, b^{(0)}]$, probiere NEWTON-Schritt; falls das Ergebnis $x^{(1)} \in I^{(0)}$, so akzeptiere; bilde je nach Vorzeichen von $f(x^{(1)})$ das Intervall $I^{(1)} = [a^{(0)}, x^{(1)}]$ bzw. $I^{(1)} = [x^{(1)}, b^{(0)}]$; falls dagegen $x^{(1)} \notin I^{(0)}$, so verwerfe das Ergebnis des NEWTON-Schritts und führe Intervallhalbierungsschritt stattdessen durch.

3.4 Das Newton-Verfahren im $\mathbb{R}^m$

Für $f : \mathbb{R} \to \mathbb{R}$ hatten wir das Newton-Verfahren durch *Linearisierung* gewonnen: Die TAYLOR-Entwicklung von f um $x^{(n)}$ (wenn wir uns an die geometrische Konstruktion auf S. 46 erinnern):

$$f(x) = \underbrace{f(x^{(n)}) + f'(x^{(n)}) \cdot (x - x^{(n)})}_{\text{lineare Approximation an } f \text{ (=Tangente!)}} + O(|x - x^{(n)}|^2)$$

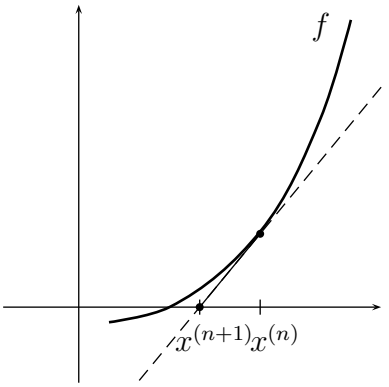


Abbildung 14: Linearisierung von f

$x^{(n+1)}$ wurde durch *Nullsetzen* der *Linearisierung*

$$f(x^{(n)}) + f'(x^{(n)}) \cdot (x^{(n+1)} - x^{(n)}) \stackrel{!}{=} 0$$

$$\Leftrightarrow x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$$

gewonnen. Naheliegend im Fall $f : \mathbb{R}^m \to \mathbb{R}^m$ ist: Wir verwenden die TAYLOR-Entwicklung

$$f(x) = \underbrace{f(x^{(n)}) + (Jf)(x^{(n)})(x - x^{(n)})}_{\textit{lineare} \text{ Approximation an } f} + O(\|x - x^{(n)}\|^2),$$

und $x^{(n+1)}$ sei wieder die Nullstelle der Linearisierung:

$$f(x^{(n)}) + Jf(x^{(n)})(x^{(n+1)} - x^{(n)}) \stackrel{!}{=} 0 \quad \Leftrightarrow \quad \boxed{x^{(n+1)} = x^{(n)} - Jf(x^{(n)})^{-1} f(x^{(n)})} \tag{3.25}$$

NEWTON-Verfahren im$\mathbb{R}^m$

Erforderlich: Die Matrix $Jf(x^{(n)})$ muss invertierbar sein. Nach (3.25) scheint es, das $(Jf(x^{(n)}))^{-1}$ explizit berechnet werden muss. Folgende Umformulierung zeigt, dass es stattdessen reicht, ein lineares Gleichungssystem zu lösen:

Setze $\Delta x^{(n)} := x^{(n+1)} - x^{(n)}$. Dann ist (3.25) $\Leftrightarrow$ $Jf(x^{(n)})\Delta x^{(n)} = -f(x^{(n)})$. Der NEWTON-Schritt (3.25) wurde so äquivalent umgeschrieben zu:

$$\begin{array}{ll} (1) & \text{Löse das Lineare Gleichungssystem } Jf(x^{(n)})\Delta x^{(n)} = -f(x^{(n)}) \\ (2) & \text{Setze } x^{(n+1)} := x^{(n)} + \Delta x^{(n)} \end{array} \tag{3.26}$$

$\Delta x^{(n)}$ bzw. Teilschritt (2) heißt auch *Newton-Update*.

Das nichtlineare Problem $f(x) = 0$ wurde ersetzt durch eine *Folge* von *linearen* Problemen.

Aufwandsvergleich: Bei Verwendung von LR erfordert (3.26)

$$\underbrace{\frac{1}{3}m^3}_{LR\text{-Zerlegung}} + \underbrace{\frac{m^2}{2} + \frac{m^2}{2}}_{\text{Vor- und Rückwärtseinsetzen}} \doteq \frac{1}{3}m^3 \text{ Additionen und Multiplikationen}$$

Bei (3.25) mit LR brauchen wir ebenfalls eine LR-Zerlegung für $A := Jf(x^{(n)})$, allerdings brauchen wir zum Aufstellen von A^{-1} m-maliges Vor-/Rückwärtseinsetzen:

$$LRu_j = e_j; \{e_1, \ldots, e_n\} = \text{ Standard-Basis}$$

Die u_j bilden die Spalten von A^{-1}, insgesamt $\frac{1}{3}m^3 + m \cdot m^2 = \frac{4}{3}m^3$ Operationen!

(3.25) ist also viermal so teuer wie (3.26)! Man kann auch im vektoriellen Fall $X = \mathbb{R}^m$ zeigen, dass das NEWTON-Verfahren (3.25) bzw. (3.26) *lokal quadratisch* konvergent ist; man kann sogar konkrete Angaben über die *Mindest*größe des Konvergenzbereichs machen (die aber in der Praxis mühsam zu überprüfen sind):

Satz 3.12 *(Konvergenz Newton-Verfahren für $X = \mathbb{R}^m$)*
Sei $U \subset \mathbb{R}^m$ offen und konvex, sei $f : U \to \mathbb{R}^m$ differenzierbar, sei $x^{(0)} \in U$. Es gebe Zahlen $\alpha, \beta, \gamma, h, r > 0$ mit

$$h := \frac{\alpha\beta\gamma}{2} < 1 \quad , \quad r := \frac{\alpha}{1-h} \quad , \quad \overline{B_r}(x^{(0)}) \subseteq U.$$

Ferner sei vorausgesetzt:

(a) $\|Jf(x) - Jf(y)\| \leq \gamma\|x - y\|$ $\forall x, y \in \tilde{U} := B_{r+\epsilon}(x^{(0)}) \subset U$ *für ein* $\epsilon > 0$, *("Jf ist Lipschitz-stetig".)*[3]

(b) Für alle $x \in \overline{B_r}(x^{(0)})$ *existiert* $(Jf(x))^{-1}$ *und* $\|Jf(x)^{-1}\| \leq \beta$

(c) $\|Jf(x^{(0)})^{-1} f(x^{(0)})\| \leq \alpha$

Dann gilt:

(1) Die Newton-Iteration (3.25)/(3.26) ist wohldefiniert und $x^{(n)} \in B_r(x^{(0)})$ $\forall n \in \mathbb{N}$

[3]Dies ist eine Abschwächung der Voraussetzung "$f \in \mathcal{C}^3$" aus Satz 3.10.

(2) $(x^{(n)})$ *konvergiert gegen eine Nullstelle* x^* *von* f*.*

(3) $\|x^{(n+1)} - x^*\| \leq \frac{\beta\gamma}{2}\|x^{(n)} - x^*\|^2$ *(d.h. quadratische Konvergenz)*
sowie $\|x^{(n)} - x^*\| \leq \alpha\frac{h^{2^n-1}}{1-h^{2n}}$ $\forall n \in \mathbb{N}$ *(a priori-Abschätzung)*

Beweis: s. Knabner-Skript S. 107-109

Bemerkung:

- Die Aussagen gelten natürlich erst recht im skalaren Fall $X = \mathbb{R}$.
- Durch die Wahl von $x^{(0)}$ hinreichend nahe bei einer Nullstelle x^* (also $f(x^{(0)})$ hinreichend nahe bei 0) kann $\alpha > 0$ beliebig klein gemacht werden (vgl. (c)), so dass die Bedingungen des Satzes erfüllt werden ($\rightarrow$ *lokale* Konvergenz).
- Stellt man die Frage nach der *Größe* des Konvergenzbereichs *nicht*, so lässt sich Satz 3.12 vereinfachen zu:

  f differenzierbar auf einer Umgebung U von x^* mit $f(x^*) = 0$ und $Jf(x^*)$ invertierbar, Jf L-stetig auf U. Dann ist das Newton-Verfahren *lokal quadratisch* konvergent.

  Beweis: (sowie weitere Infos:)
  P. Deuflhard, Newton methods for nonlinear problems, Springer, 2004, 430 Seiten.

In der Praxis wird man, um das Verfahren auf Konvergenz/Divergenz zu prüfen, weniger die Voraussetzung von Satz 3.12 überprüfen, sondern eher einen sogenannten *Monotonietest* durchführen:

Das Nullstellenproblem $f(x) = 0$ ist offensichtlich äquivalent zum Lösen von:

$$\text{minimiere } g(x) := \|f(x)\|^2 \quad \text{für } x \in \mathbb{R}^n \tag{3.27}$$

Man könnte also einen "Abstieg" $g(x^{(n+1)}) \overset{!}{\leq} g(x^{(n)})$ in diesem Funktional erwarten und daher

$$\|f(x^{(n+1)})\| \leq \Theta\|f(x^{(n)})\| \quad \text{für ein } \Theta \in (0,1) \tag{3.28}$$

testen. Falls (3.28) nicht erfüllt ist, wird für die neue Newton-Iterierte $x^{(n+1)} = x^{(n)} + \Delta x^{(n)}$, so kann man den Newton-Schritt "dämpfen" d.h. man setzt

$$x^{(n+1)} := x^{(n)} + t\Delta x^{(n)} \quad \text{mit } t \in (0,1], \tag{3.29}$$

wobei ein t so zu finden ist, dass (3.28) erfüllt ist oder besser

$$\|f(x^{(n)} + t\Delta x^{(n)})\| \leq (1 - \Theta t)\|f(x^{(n)})\|$$

erfüllt ist („Armijo-Regel").
In der Praxis probiert man nacheinander $t = 1$, $t = \gamma$, $t = \gamma^2$, ... für ein festes $\gamma \in (0,1)$, z.B.: $\gamma = 0,5$, aus, bis (3.28) erfüllt wird:

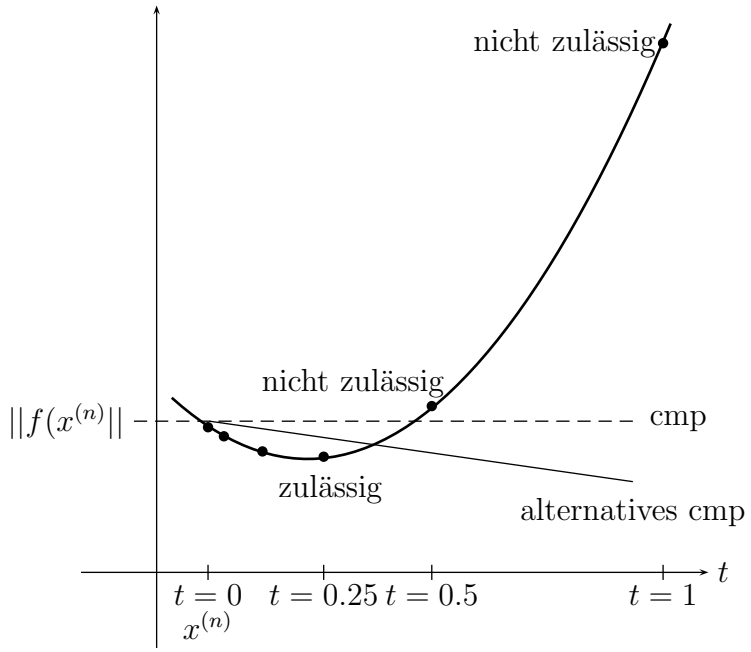


Abbildung 15: "Newton-Armijo-Regel"

gegeben: Startwert x, Parameter γ, $\tilde{\Theta} \in (0, 1]$, $l_{\max} \in \mathbb{N}$;

```
1:  wiederhole {
2:      t := 1
3:      loese Jf(x) Δ x = -f(x)
4:      l := 0
5:      solange || f(x + t Δ x) || > (1 - t Θ̃) || f(x) ||
        und l ≤ l_max
6:          t := t · γ; l := l + 1
7:      falls l > l_max: Abbruch
8:      x := x + t Δ x
9:  } bis Abbruchkriterium erfuellt
```

Dieser Algorithmus (gedämpftes Newton-Verfahren nach Armijo) hat im allgemeinen einen größeren Konvergenzbereich als das reine Newton-Verfahren.

Literatur: Kelley: Iterative methods for linear and nonlinear problems

Die Suche nach der geeigneten "Schrittweite" t nennt man auch *Line Search*, da die neue Iteration entlang der durch (3.29) beschriebenen *Geraden* gesucht wird.

Zur Übung: Der Line Search sollte, zumindest für $\Theta \approx 1$, bzw. $\tilde{\Theta} = 0$ sofern $Jf(x)$ immer invertierbar und $x \mapsto Jf(x)$ stetig differenzierbar ist, irgendwann erfolgreich ein t finden, denn die Richtung

$$r := \Delta x^{(n)} = -(Jf(x^{(n)})^{-1} f(x^{(n)})$$

stellt eine *Abstiegs*richtung für das Funktional $g(x) = \|f(x)\|^2$ dar, sofern $\|\cdot\| = \|\cdot\|_2$:

$$\begin{aligned}\frac{\partial g(x^{(n)})}{\partial r} = \langle \nabla g(x^{(n)}), \Delta x^{(n)} \rangle &= \langle 2Jf(x^{(n)})^t f(x), -Jf(x^{(n)})^{-1} f(x^{(n)}) \rangle \\ &= -2\langle f(x^{(n)}), \underbrace{Jf(x^{(n)}) Jf(x^{(n)})^{-1}}_{=Id} f(x^{(n)}) \rangle \\ &= -2\|f(x^{(n)})\|^2 \leq 0\end{aligned}$$

(sogar < 0 solange $f(x^{(n)}) \neq 0$).

Um bei Nichtexistenz von $(Jf(x^{(n)}))^{-1}$ einen Abbruch des Verfahrens zu vermeiden, kann man in diesem Fall die Suchrichtung $\Delta x^{(n)} = (Jf(x^{(n)}))^{-1} f(x^{(n)})$ durch die Richtung des stärksten Abstiegs des Funktionals g, also $-\nabla g(x^{(n)}) = -2Jf(x^{(n)})^t f(x^{(n)})$, multipliziert mit einer geeigneten Schrittweite t, ersetzen (s. Hanke-Bourgeois).

Vereinfachtes Newton-Verfahren :

Um Rechenzeiten zu sparen, kann man alle (oder zumindest jeweils 2 oder 3) Newton-Schritte mit ein und derselben Jacobi-Matrix $Jf(x^{(0)})$ durchführen;

$$x^{(n+1)} = x^{(n)} - (Jf(x^{(0)})^{-1} f(x^{(n)}). \tag{3.30}$$

Diese LR-Zerlegung der Jacobi-Matrix ist dann nur einmal durchzuführen. Diesen Rechenzeitvorteil erkauft man sich allerdings mit dem Verlust der quadratischen Konvergenz; die Konvergenz ist nur noch linear (wenn auch mit kleiner Kontraktionskonstanten k, falls $x^{(0)}$ nahe bei x^* ist). Beachte:
(3.30) ist von der Form (3.14), d.h.

$$\Phi(x) = x + \alpha f(x), \text{ mit } \alpha = -Df(x^{(0)})^{-1} \text{ bzw. } \alpha = -\frac{1}{f'(x^{(0)})} \text{ im } 1-D\text{-Fall.}$$

Eine weitere Vereinfachungsmöglichkeit ist es, beim Assemblieren (=Aufstellen) der linearen Gleichungssysteme die Ableitung $\frac{\partial f_i}{\partial x_j}$ durch diskrete Differenzenquotienten (vgl. Kapitel 1.5) zu ersetzen, um das Assemblieren zu beschleunigen, da die Ableitungen oft kompliziertere Funktionen sind als die f_i selbst.

4 Iterative Verfahren für lineare Gleichungssysteme. Teil I: Fixpunktverfahren

4.1 Einführung

Wir betrachten das lineare Gleichungssystem $Ax = b,\ A \in \mathbb{C}^{n\times n}$, invertierbar, $b \in \mathbb{C}^n$. Wir wollen dieses Problem auf *Fixpunktgestalt* bringen und per Fixpunktverfahren iterativ lösen. Die Konvergenz wird mit dem Fixpunktsatz von Banach untersucht.

Mögliche elementare Ansätze:

1. (In Anlehnung an den nichtlinearen Fall Kapitel 3.2)

$$\begin{aligned} Ax - b &= 0 && | \cdot(-\omega),\ \omega \neq 0 \\ (-\omega)(Ax-b) &= 0 && | + x \\ \underbrace{(Id - \omega A)x + \omega b}_{=:\Phi(x)} &= x && \text{Fixpunktgleichung} \end{aligned}$$

   Fixpunktiteration: $x^{(k+1)} := (Id - \omega A)x^{(k)} + \omega b$.

2. Statt mit $\omega \in \mathbb{R}$ kann man in 1. auch mit einer nichtsingulären Matrix $-B \in \mathbb{C}^{n\times n}$ multiplizieren. Man erhält die Fixpunktgleichung

$$\underbrace{(Id - BA)x + Bb}_{=:\Phi(x)} = x$$

   und die Fixpunktiteration $x^{(k+1)} := (Id - BA)x^{(k)} + Bb$.

3. Man zerlegt A in einen "wesentlichen Anteil" W und einen "Rest" S:

$$A = W + S. \tag{4.1}$$

   W soll dabei leicht invertierbar sein (z.B. bei diagonaldominantem A: $W := \mathrm{diag}(a_{ii})$ und $S := A - W$):

$$Ax = b \Leftrightarrow Wx = -Sx + b \Leftrightarrow x = \underbrace{-W^{-1}Sx + W^{-1}b}_{=:\Phi(x)} \tag{4.2}$$

In allen Fällen hat Φ die Form

$$\Phi(x) = Mx + \tilde{b} \tag{4.3}$$

wobei $M, \tilde{b}$ derart, dass

$$\Phi(x) = x \Leftrightarrow Ax = b$$

("*Konsistenz*" des linearen Gleichungssystems und des Fixpunktproblems) (mit $M = Id - \omega A$ bzw. $M = Id - BA$ bzw. $M = -W^{-1}S$). Wann liefert der Fixpunktsatz von Banach Konvergenz der Fixpunktiteration

$$x^{(k+1)} = \Phi(x^{(k)})\ ? \tag{4.4}$$

Wir wollen eine Matrixnorm $\|\cdot\|$ als *Operatornorm* bezeichnen, falls es eine zugehörige Vektornorm $\|\cdot\|$ gibt mit

$$\|A\| = \sup_{0\neq x\in\mathbb{C}^n} \frac{\|Ax\|}{\|x\|} \ \forall A \in \mathbb{C}^{n\times n}.$$

Lemma 4.1 *Sei $x^{(0)} \in \mathbb{C}^n$, sei $\|\cdot\|$ eine beliebige Norm des $\mathbb{C}^n$ sowie eine verträgliche Matrixnorm (z.B. die zugeordnete Matrixnorm/Operatornorm). Falls $\|M\| < 1$, so ist die Iteration (4.4)/(4.3) konvergent (sogar in* jeder *beliebigen Norm des $\mathbb{C}^n$).*

Beweis: $\Phi : \mathbb{C}^n \to \mathbb{C}^n$ ist trivialerweise selbstabbildend.

$$\|\Phi(x) - \Phi(y)\| = \|M(x-y)\| \underset{\text{verträglich}}{\leq} \underbrace{\|M\|}_{<1} \cdot \|x-y\| \quad \forall x, y \in \mathbb{C}^n$$

$$\overset{\text{Fixpunktsatz}}{\underset{\text{von Banach}}{\Rightarrow}} \text{ Behauptung}$$

□

Es kann vorkommen, dass für *eine* Matrix M *eine* Matrixnorm $\|M\|_a > 1$, und eine *andere* Matrixnorm $\|M\|_b < 1$ erfüllt (in diesem Fall hätten wir nach Lemma 4.1 Konvergenz!). Welche Norm ist zu prüfen? Kann man (für festes M)

$$\inf\{\|M\| \mid \|\cdot\| \text{ ist zu einer Vektornorm des } \mathbb{C}^n \text{ verträgliche Matrixnorm}\} \tag{4.5}$$

charakerisieren? (Prüfe dann, ob dies < 1 ist!)
Es reicht in (4.5) nur die *Operator*normen zu betrachten, denn für eine *beliebige* verträgliche Matrixnorm $\|\cdot\|$, d.h. $\|Ax\| \leq \|A\| \cdot \|x\| \ \forall A, x$, also $\|A\| \geq \frac{\|Ax\|}{\|x\|} \ \forall A, \ x \neq 0$, gibt es immer eine Operatornorm $|||\cdot|||$ mit $|||A||| \leq \|A\|$, nämlich $|||A||| := \sup\limits_{x\neq 0} \frac{\|Ax\|}{\|x\|} \leq \|A\|$. Also gilt:

$$(4.5) = \inf\{\|M\| \mid \|\cdot\| \text{ ist eine Operatornorm}\}$$

Lemma 4.2 *Sei $\varrho(M) = \max\{|\lambda| \mid \lambda \in \mathbb{C}$ ist Eigenwert von $M\}$ der Spektralradius von $M \in \mathbb{C}^{n\times n}$. Dann gilt*

(a) Für jede *Operatornorm $\|M\| = \sup\limits_{0\neq x\in\mathbb{C}^n} \frac{\|Ax\|}{\|x\|}$ gilt:*

$$\varrho(M) \leq \|M\|$$

(d.h. $\varrho(M)$ist eine untere Schranke für die Menge (4.5)).

(b) Für jedes $\epsilon > 0$ und jede Matrix $M \in \mathbb{C}^{n\times n}$ gibt es eine Operatornorm

$$\|M\|_M = \sup_{0\neq x\in\mathbb{C}^n} \frac{\|Mx\|_M}{\|x\|_M}, \tag{4.6}$$

so dass

$$\|M\|_M \leq \varrho(M) + \epsilon.$$

Also: $\varrho(M)$ ist der gesuchte Wert (4.5)!

Bemerkung: Wir wissen: Für *symmetrisches* M können wir in (b) $\|\cdot\|_M := \|\cdot\|_2$ wählen (für beliebiges $\epsilon > 0$); es gilt $\|M\|_2 = \varrho(M)$.

Beweis von Lemma 4.2:

(a) Mit $Mx = \lambda x, x \neq 0$, folgt $\|M\| \underset{\text{Operatornorm}}{\geq} \frac{\|Mx\|}{\|x\|} = |\lambda| \Rightarrow \|M\| \geq \varrho(M)$.

(b) siehe Stoer & Bulirsch, Satz 6.8.2 oder Schwarz, 3. Auflage, Satz 11.4.

□

Lemma 4.2 besagt: $\varrho(M) < 1$ ist notwendig und hinreichend für die Existenz einer Operatornorm $\|\cdot\|_M$, so dass $\|M\|_M < 1$ ist. Damit folgt:

Lemma 4.3 *Sei $M \in \mathbb{C}^{n\times n}$, $\tilde{b} \in \mathbb{C}^n$. Die Fixpunktiteration $x^{(k+1)} = \Phi(x^{(n)})$, mit $\Phi(x) = Mx + \tilde{b}$, ist genau dann für beliebige Startwerte $x^{(0)}$ konvergent gegen den eindeutig bestimmten Fixpunkt x^* von Φ (d.h. $Mx^* + \tilde{b} = x^*$), wenn $\varrho(M) < 1$.*

Beweis:

"$\varrho(M) < 1 \Rightarrow$ Konvergenz": Aus $\varrho(M) < 1$ folgt mit Lemma 4.2 (b) die Existenz einer Vektornorm $\|\cdot\|_M$, so dass die zugehörige Matrixnorm $\|M\|_M < 1$ erfüllt. Die Behauptung folgt mit Lemma 4.1 (d.h. Fixpunktsatz von BANACH in $(\mathbb{C}^n, \|\cdot\|_M)$)

"$\varrho(M) \geq 1 \Rightarrow$ keine Konvergenz für beliebige $x^{(0)}$" : Sei $\varrho(M) \geq 1$. Angenommen $x^{(k)} \to x^*$ (x^* Fixpunkt von Φ), für beliebigen Startpunkt $x^{(0)} \in \mathbb{C}^n$. Sei x Eigenvektor von M zu einem Eigenwert $|\lambda| \geq 1$: Setze $x^{(0)} := x + x^*$

$$\begin{aligned}
\Rightarrow x^{(1)} &= Mx^{(0)} + \tilde{b} = Mx + \underbrace{Mx^* + \tilde{b}}_{=x^*} = \lambda x + x^* \\
x^{(2)} &= Mx^{(1)} + \tilde{b} = \lambda Mx + \underbrace{Mx^* + \tilde{b}}_{=x^*} = \lambda^2 x + x^* \\
&\vdots \\
\|x^{(k)} - x^*\| &= \|\lambda^k x\| = \underbrace{|\lambda|^k}_{\geq 1} \|x\| \geq \underbrace{\|x\|}_{\neq 0} \qquad \forall k \in \mathbb{N} \\
&\Rightarrow \quad x^{(k)} \nrightarrow x^*
\end{aligned}$$

□

Für die Verfahrensklassen (1) - (3) ist also sicherzustellen:

$$\varrho(Id - \omega A) < 1 \quad \text{bzw.} \quad \varrho(Id - BA) < 1 \quad \text{bzw.} \quad \varrho(W^{-1}S) < 1.$$

Im Folgenden konzentrieren wir uns auf Verfahren der Kategorie (3). Zu Kategorie (1) siehe Übung. Kategorie (2) und (3) lassen sich ineinander umwandeln (siehe ebenfalls Übung) durch:

$$B := (Id + W^{-1}S)A^{-1} \quad \text{bzw.} \quad W := B^{-1}, \quad S := A - B^{-1}.$$

4.2 Jacobi- und Gauß-Seidel-Verfahren

Man zerlegt $A = L + D + R$, wobei L und R eine strikte (linke bzw. rechte) Dreiecksmatrix und $D = \mathrm{diag}(a_{ii})$ eine Diagonalmatrix ist. Wir setzen voraus, dass $a_{ii} \neq 0\ \forall i$, d.h. D^{-1} existiert. In der Notation von Verfahrensklasse (3) in Kapitel 4.1 setzt man $W := D$ und $S = L + R$, d.h.:

$$\begin{aligned} Ax = b \quad &\Leftrightarrow \quad Dx &&= -(L+R)x + b \\ &\Leftrightarrow \quad x &&= \underbrace{-D^{-1}(L+R)x + D^{-1}b}_{\Phi(x)} \end{aligned}$$

Die zugehörige Fixpunktiteration

$$\begin{aligned} x^{(k+1)} = \Phi(x^{(k)}) &= M_{\mathrm{J}} x^{(k)} + \tilde{b} \\ \text{mit} \quad M_{\mathrm{J}} &= -D^{-1}(L+R), \quad \tilde{b} = D^{-1}b \end{aligned} \tag{4.7}$$

heißt Jacobi-*Verfahren* (Carl Jacobi, 1804-51, Berlin) oder *Gesamtschrittverfahren*. Komponentenweise geschrieben:

$$x_i^{(k+1)} := \frac{1}{a_{ii}}(-\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j^{(k)} + b_i), \quad i = 1, \ldots, n.$$

Das Update des i-ten Vektoreintrags verwendet also die Komponenten $x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)}$ der alten Iterierten $x^{(k)}$. Naheliegend ist die Vermutung, dass man schnellere Konvergenz bekommt, wenn man stattdessen die bereits berechneten Werte $x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ (sowie weiterhin $x_{i+1}^{(k)}, \ldots, x_n^{(k)}$) benutzt:

$$x_i^{(k+1)} := \frac{1}{a_{ii}}(-\sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} + b_i), \ i = 1, \ldots, n. \tag{4.8}$$

(4.8) heißt Gauss-Seidel-*Verfahren* oder *Einzelschrittverfahren* (Seidel: 1821-96, München). Im Computer kann (4.8) einfach mittels Überschreiben implementiert werden:

$$x_i := \frac{1}{a_{ii}}(-\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j + b_i) \tag{4.9}$$

Um die Iterationsmatrix M_{GS} des Gauß-Seidel-Verfahrens (4.8) zu identifizieren, schreiben wir es als

$$x^{(k+1)} = D^{-1}(-Lx^{(k+1)} - Rx^{(k)} + b) \qquad |D\cdot \tag{4.10}$$

$$\Leftrightarrow \quad (D+L)x^{(k+1)} = -Rx^{(k)} + b$$

$$\Leftrightarrow \quad x^{(k+1)} = -(D+L)^{-1}Rx^{(k)} + (D+L)^{-1}b, \tag{4.10a}$$

$$\text{also} \quad M_{GS} = -(D+L)^{-1}R, \quad \tilde{b} = (D+L)^{-1}b.$$

Die Invertierung von $D + L$ in (4.10) muss nur *scheinbar* erfolgen, siehe Darstellungen (4.8), (4.9). Ein Schritt des Jacobi- und des Gauss-Seidel-Verfahrens sind gleich aufwändig.

Satz 4.4 *Sei A (zeilenweise)* diagonaldominant*, d.h.*

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| < |a_{ii}| \ \forall i = 1, \ldots, n \ .$$

Dann gilt:

(a) $\|M_J\|_\infty < 1$.

(b) $\|M_{GS}\|_\infty \leq \|M_J\|_\infty \ \ (< 1)$.

d.h. das Jacobi- *und das* Gauss-Seidel-*Verfahren sind konvergent.* [4]

Bemerkung: Ist A *spaltenweise* diagonaldominant, so gelten in Satz 4.4 die entsprechenden Aussagen für $\|\cdot\|_1$ statt $\|\cdot\|_\infty$.
Beweis:

(a) $\|M_J\|_\infty = \|D^{-1}(L+R)\|_\infty = \max\limits_i \sum\limits_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1.$

(b) Wir zeigen:

$$\|\underbrace{\overbrace{(D+L)^{-1}R}^{-M_{GS}}x}_{=y}\|_\infty \leq \|\overbrace{D^{-1}(L+R)}^{-M_J}\|_\infty \ \forall x \in \mathbb{C}^n \ \text{ mit } \|x\|_\infty = 1. \tag{4.11}$$

daraus folgt durch sup-Bildung die Behauptung.

Setze $y := \overset{x}{(D+L)^{-1}}Rx$, also $(D+L)y = Rx, \ Dy = -Ly + Rx$; komponentenweise heißt das:

$$a_{ii}y_i = -\sum_{j<i} a_{ij}y_i + \sum_{j>i} a_{ij}x_j \ \ \forall i = 1, \ldots, n. \tag{4.12}$$

Wir zeigen per Induktion nach i, dass

$$|y_i| \leq \|D^{-1}(L+R)\|_\infty, \ \ \forall i = 1, \ldots, n \tag{4.13}$$

(woraus dann $\|y\|_\infty \leq \|D^{-1}(L+R)\|_\infty$, also (4.11), und somit die Behauptung, folgt)

$\underline{i=1}$: $|y_1| \underset{(4.12)}{=} \left|\frac{1}{a_{11}}\sum\limits_{j=2}^{n} a_{1j}x_j\right| \leq \sum\limits_{j=2}^{n} \left|\frac{a_{1j}}{a_{11}}\right| \cdot \underbrace{|x_j|}_{\leq 1} \leq \sum\limits_{j=2}^{n} \left|\frac{a_{1j}}{a_{11}}\right|$ = erste Zeilensumme von $D^{-1}(L+R)$,
also $\leq \|D^{-1}(L+R)\|_\infty$.

[4] Aussage (b) "passt" zu der zuvor geäußerten Intuition, dass das Gauß-Seidel-Verfahren schneller sein sollte als das Jacobi-Verfahren, ist aber *kein* strenger Beweis einer solchen Behauptung. Vergleiche auch S.76, L.A. II.

$\underline{\{1,\dots,i-1\} \to i}$: $\|y_i\| \underset{(4.12)}{\leq} \sum_{j<i} \left|\frac{a_{ij}}{a_{ii}}\right| \cdot \underbrace{|x_j|}_{<1} + \sum_{j>i} \left|\frac{a_{ij}}{a_{ii}}\right| \cdot \underbrace{|x_j|}_{\leq 1}$.

Nach Induktionsvorraussetzung ist

$$\begin{aligned} |y_j| &\underset{\text{I.V. für } j<i}{\leq} \|D^{-1}(L+R)\|_\infty \underset{\text{(a)}}{<} 1 \text{ für } j<i \\ \Rightarrow |y_i| &\leq \sum_{j\neq i} \left|\frac{a_{ij}}{a_{ii}}\right| = i\text{-te Zeilensumme von } D^{-1}(L+R) \\ &\leq \|D^{-1}(L+R)\|_\infty \end{aligned}$$

Also: (4.13) gilt für alle $i = 1,\dots,n$.
$\Rightarrow \|(D+L)^{-1}R\|_\infty = \max_{\|x\|_\infty=1} \| \underbrace{(D+L)^{-1}Rx}_{=y} \|_\infty \underset{(4.13)}{\leq} \|D^{-1}(L+R)\|_\infty$

□

Die Forderung der Diagonaldominanz in Satz 4.4 ist in der Praxis oft zu restriktiv. Siehe zum Beispiel die Diskretisierung der POISSON-Gleichung

$$-\Delta u = f \quad (\text{bzw. } -u'' = f \text{ in 1-D})$$

[vgl. L.A.I Blatt 7 sowie L.A. II Seite 75 f., L.A. II Blatt 8]. Dort hat man Matrizen A, die nur

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j\neq n}} |a_{ij}| \quad \forall i = 1,\dots,n \tag{4.14}$$

und

$$\exists\, i \in \{1,\dots,n\} \quad |a_{ii}| > \sum_{\substack{j=1 \\ j\neq n}}^{n} |a_{ij}| \tag{4.15}$$

erfüllen.

Eine Matrix $A \in \mathbb{C}^{n\times n}$, die (4.14) und (4.15) erfüllt, heißt *schwach diagonaldominant*. Man kann zeigen, dass Satz 4.4 auch für schwach diagonaldominante Matrizen gültig ist, sofern zusätzlich vorausgesetzt wird, dass A *irreduzibel* ist. Eine Matrix $A \in \mathbb{C}^{n\times n}$ heißt *irreduzibel*, falls es für jede disjunkte Zerlegung der Indexmenge

$$\{1,\dots,n\} = I_1 \cup I_2,\ I_1, I_2 \neq \emptyset,\ I_1 \cap I_2 = \emptyset,$$

stets Indizes $i \in I_1,\ j \in I_2$ gibt mit $a_{ij} \neq 0$. Äquivalent dazu ist:
Es gibt keine Permutationsmatrix P, so dass

$$P^{-1}AP = \begin{pmatrix} B_1 & 0 \\ B_2 & B_3 \end{pmatrix},$$

wobei B_1 und B_3 quadratische Blöcke sind. Es gibt einen grafentheoretischen Algorithmus zum Testen der Irreduzibilität (s. Schwarz & Köckler, Seite 497 f.). Ist A *reduzibel*, so kann das Lösen von $Ax = b$ offensichtlich in zwei (oder mehr) getrennte lineare Gleichungssysteme, deren Matrix jeweils irreduzibel ist, gesplittet werden, d.h. die Forderung der Irreduzibilität ist keine große Einschränkung. Zum Beweis der Verallgemeinerung von Satz 4.4 auf schwach diagonaldominante irreduzible Matrizen siehe z.B. Schwarz & Köckler.

4.3 Relaxation für Gauß-Seidel- und Jacobi-Verfahren

Iterative Verfahren lohnen sich im Allgemeinen dann, wenn nicht allzuviele Iterationsschritte nötig sind[5]. Daher ist es aus Effizienzgründen erforderlich, dass die Kontraktionsrate $\varrho(T)$ hinreichend klein ist. Für lineare Gleichungssysteme, die aus der Diskretisierung von PDEs entstehen, hat man oft

$$\varrho(M_{\mathrm{GS}}) \to 1, \varrho(M_{\mathrm{J}}) \to 1$$

für Diskretisierungsparameter $h \to 0$ (d.h. $n \to \infty$), d.h. asymptotisch immer schlechter werdende Konvergenzraten (s. Übung). Eine Möglichkeit, die Konvergenzrate zu verbessern, bietet oft die *Relaxation*. Beim JACOBI-Verfahren

$$\begin{aligned} x^{(k+1)} &:= -D^{-1}(L+R)x^{(k)} + D^{-1}b \\ &= x^{(k)} \underbrace{-D^{-1}(L+D+R)x^{(k)} + D^{-1}b}_{=:\Delta x^{(k)}} \end{aligned}$$

ersetzt man einfach das Update $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ durch $\omega \cdot \Delta x^{(k)}$, $\omega \in \mathbb{R}$, d.h.

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \omega \Delta x^{(k)} \\ &= (1-\omega)x^{(k)} - \omega D^{-1}(L+R)x^{(k)} + \omega D^{-1}b. \end{aligned} \tag{4.16}$$

(4.16) heißt *relaxiertes* JACOBI-*Verfahren*, $\omega \in \mathbb{R}$ heißt *Relaxationsparameter*. Für $\omega > 1$ spricht man von *Überrelaxation*, für $\omega < 1$ von *Unterrelaxation*. Nichtsdestotrotz hat sich für (4.16), $\omega \in \mathbb{R}$, der Begriff *JOR-Verfahren* (Jacobi overrelaxation) eingebürgert. Das JOR-Verfahren hat die Iterationsmatrix

$$\begin{aligned} M_{\mathrm{JOR}}(\omega) &= (1-\omega)Id - \omega D^{-1}(L+R) \\ &= (1-\omega)Id + \omega M_{\mathrm{J}}, \\ M_{\mathrm{JOR}}(1) &= M_{\mathrm{J}}. \end{aligned} \tag{4.17}$$

Beim GAUSS-SEIDEL-Verfahren geht man ähnlich vor. Die Darstellungen (4.8)/(4.10) (*nicht* Darstellung (4.10a)!) werden verwendet, um $\Delta x^{(k)}$ zu erklären:

$$\begin{aligned} x^{(k+1)} &= D^{-1}(-Lx^{(k+1)} - Rx^{(k)} + b) \\ &= x^{(k)} \underbrace{-D^{-1}Lx^{(k+1)} - D^{-1}(D+R)x^{(k)} + D^{-1}b}_{=:\Delta x^{(k)}} \end{aligned}$$

Dies wird modifiziert zu

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \omega \Delta x^{(k)} \\ &= (1-\omega)x^{(k)} - \omega D^{-1}(Lx^{(k+1)} + Rx^{(k)}) + \omega D^{-1}b. \end{aligned} \tag{4.18}$$

Für dieses Verfahren hat sich der Name *SOR-Verfahren* (successive overrelaxation) eingebürgert. Die Darstellung (4.18) des SOR-Verfahrens ist ideal für die Implementierung. Zur Ermittlung der

[5] z.B. wenn die Anzahl der Schritte $\approx n^{\alpha}$ mit $\alpha < 1$ für $n \times n$ Matrizen, die aus der Diskretisierung von partiellen Differentialgleichungen hervorgehen.

Iterationsmatrix formen wir um:

$$\begin{aligned}(4.18) \quad &\Leftrightarrow \quad (Id + \omega D^{-1}L)x^{(k+1)} = (1-\omega)x^{(k)} - \omega D^{-1}Rx^{(k)} + \omega D^{-1}b \quad |D\cdot \\ &\Leftrightarrow \quad (D+\omega L)x^{(k+1)} = (1-\omega)D^{(k)} - \omega R x^{(k)} + \omega b \\ &\Leftrightarrow \quad x^{(k+1)} = (D+\omega L)^{-1}[(1-\omega)D - \omega R]x^{(k)} + \omega(D+\omega L)^{-1}b,\end{aligned}$$

die Iterationsmatrix lautet also:

$$\boxed{M_{\mathrm{SOR}}(\omega) = (D+\omega L)^{-1}[(1-\omega)D - \omega R]}$$

Lemma 4.5 *Falls das* JACOBI-*Verfahren konvergent ist, so ist auch das JOR-Verfahren konvergent für* $0 < \omega \leq 1$.

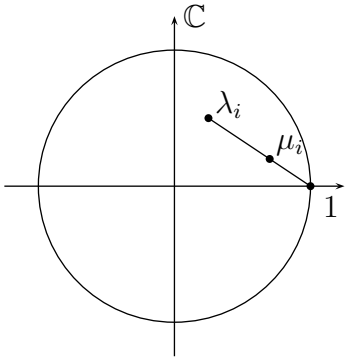


Abbildung 16: Konvexkombination

Beweis:

Nach Voraussetzung erfüllen alle Eigenwerte $\lambda_i \in \mathbb{C}$ von M_J : $|\lambda_i| < 1$. Nach (4.17) lauten die Eigenwerte von $M_{\mathrm{JOR}}(\omega)$: $\mu_i = (1-\omega)\cdot 1 + \omega\lambda_i$.
Das heißt $\mu_i \in \mathbb{C}$ ist Konvexkombination von λ_i und 1 (wobei der Koeffizient von 1 nicht null ist). Daraus folgt $|\mu_i| < 1$.

□

Lemma 4.6 *Es sei* $A \in \mathbb{R}^{n\times n}$ *s.p.d., und das* JACOBI-*Verfahren sei konvergent. Dann ist das JOR-Verfahren konvergent für alle* $\omega \in \mathbb{R}$ *mit*

$$0 < \omega < \frac{2}{1-\mu_{\min}} \left(\underset{(a)}{\leq} 2 \right),$$

wobei $\mu_{\min}$ *der kleinste Eigenwert von* M_J *ist; dieser ist insbesondere kleiner gleich 0.*

Beweis : A symmetrisch $\Rightarrow$ $L+R$ symmetrisch und hat nur reelle Eigenwerte, A positiv definit $\Rightarrow a_{ii} > 0 \Rightarrow D^{\frac{1}{2}} := \mathrm{diag}(\sqrt{a_{ii}})$ existiert.

$$M_J = -D^{-1}(L+R)$$

ist *ähnlich* zur Matrix (d.h. hat die gleichen Eigenwerte wie)

$$S := -D^{\frac{1}{2}} M_{\mathrm{J}} D^{-\frac{1}{2}} = -D^{-\frac{1}{2}}(L+R)D^{-\frac{1}{2}},$$

welche *symmetrisch* ist und daher nur reelle Eigenwerte hat. S hat Spur $= 0$, die Spur ist die Summe aller Eigenwerte $\Rightarrow$ S (also auch M_{J}) hat kleinsten Eigenwert $\mu_{\min} \leq 0$. Wegen der vorausgesetzten Konvergenz des Jacobi-Verfahrens hat M_{J} Eigenwerte $-1 < \mu_i < 1$. $M_{\mathrm{JOR}}(\omega)$ hat (nach (4.17)) Eigenwerte $\nu_j = 1 - \omega + \omega\mu_j$. Notwendige und hinreichende Bedingung für die Konvergenz des JOR-Verfahrens ist also

$$\begin{aligned} -1 < \quad & 1-\omega+\omega\mu_j \quad < 1 \qquad && |-1 \\ -2 < \quad & -\omega+\omega\mu_j \quad < 0 \qquad && |\cdot(-1) \\ 2 > \quad & \omega(1-\mu_j) \quad > 0 \qquad && |\cdot \frac{1}{1-\mu_j} \end{aligned}$$

Da nach Voraussetzung das Jacobi-Verfahren konvergent ist, ist $1-\mu_j > 0 \ \ \forall j$, also

$$\underbrace{\frac{2}{1-\mu_j}}_{\text{wird minimal für} \mu_j = \mu_{\min}} > \omega > 0 \quad \forall \text{ Eigenwerte } \mu_j \text{ von } M_{\text{J}}$$

Dies ist äquivalent zu

$$\frac{2}{1-\mu_{\min}} > \omega > 0.$$

□

Lemma 4.7 *(optimale Wahl des Relaxationsparameters für JOR)*
Sei A symmetrisch (M_J ist dann ähnlich zu einer symmetrischen Matrix, s. Beweis Lemma 4.6) und die (dann reellen) Eigenwerte von M_J erfüllen

$$\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n < 1$$

(ist z.B. erfüllt, wenn das Jacobi-Verfahren konvergent ist; z.B. wenn A diagonal dominant ist). Dann ist die optimale Wahl von ω, d.h. ω_{opt} so, dass

$$\varrho(M_{JOR}(\omega_{opt})) = \min_{\omega \in \mathbb{R}} \varrho(M_{JOR}(\omega)),$$

gegeben durch

$$\omega_{opt} = \frac{2}{2-\mu_n-\mu_1}.$$

Es ist dann $\varrho(M_{JOR}(\omega_{opt})) = \frac{\mu_n-\mu_1}{2-\mu_n-\mu_1}$.

Bemerkung: Leider kennt man im Allgemeinen die Eigenwerte von M_{J} nicht (noch nicht einmal die von A), d.h. man bestimmt ein "gutes" ω durch Ausprobieren.

Beweis : Es ist $\mu_1 \leq 0, \mu_n \geq 0$ (wie im Beweis von Lemma 4.7). Es ist

$$M_{\text{JOR}}(\omega) = (1-\omega)Id + \omega M_{\text{J}},$$

d.h. M_{JOR} hat die Eigenwerte $\nu_i = (1-\omega) + \omega\mu_i$. Offensichtlich (siehe grafische Darstellung, Abbildung 17) kommen nur $\omega > 0$ in Frage. Für $\omega > 0$ hängen die ν_i monoton von den μ_i ab:

$$\begin{gathered} \nu_1 \leq \ldots \leq \nu_n < 1 \\ \Rightarrow \varrho(M_{\text{JOR}}(\omega)) = \max\{|\nu_1|, \ldots, |\nu_n|\} = \max\{|\nu_1|, |\nu_n|\}. \end{gathered}$$

Dieser Wert wird *minimal*, wenn

$$\begin{aligned} & -\nu_1 = \nu_n \\ \text{d.h.} \quad & -1+\omega-\omega\mu_1 = 1-\omega+\omega\mu_n \\ \Leftrightarrow \quad & 2-2\omega+\omega(\mu_1+\mu_n) = 0 \\ \Leftrightarrow \quad & 2 = \omega(2-\mu_1-\mu_n) \\ \Leftrightarrow \quad & \omega = \frac{2}{2-\mu_1-\mu_n}. \end{aligned}$$

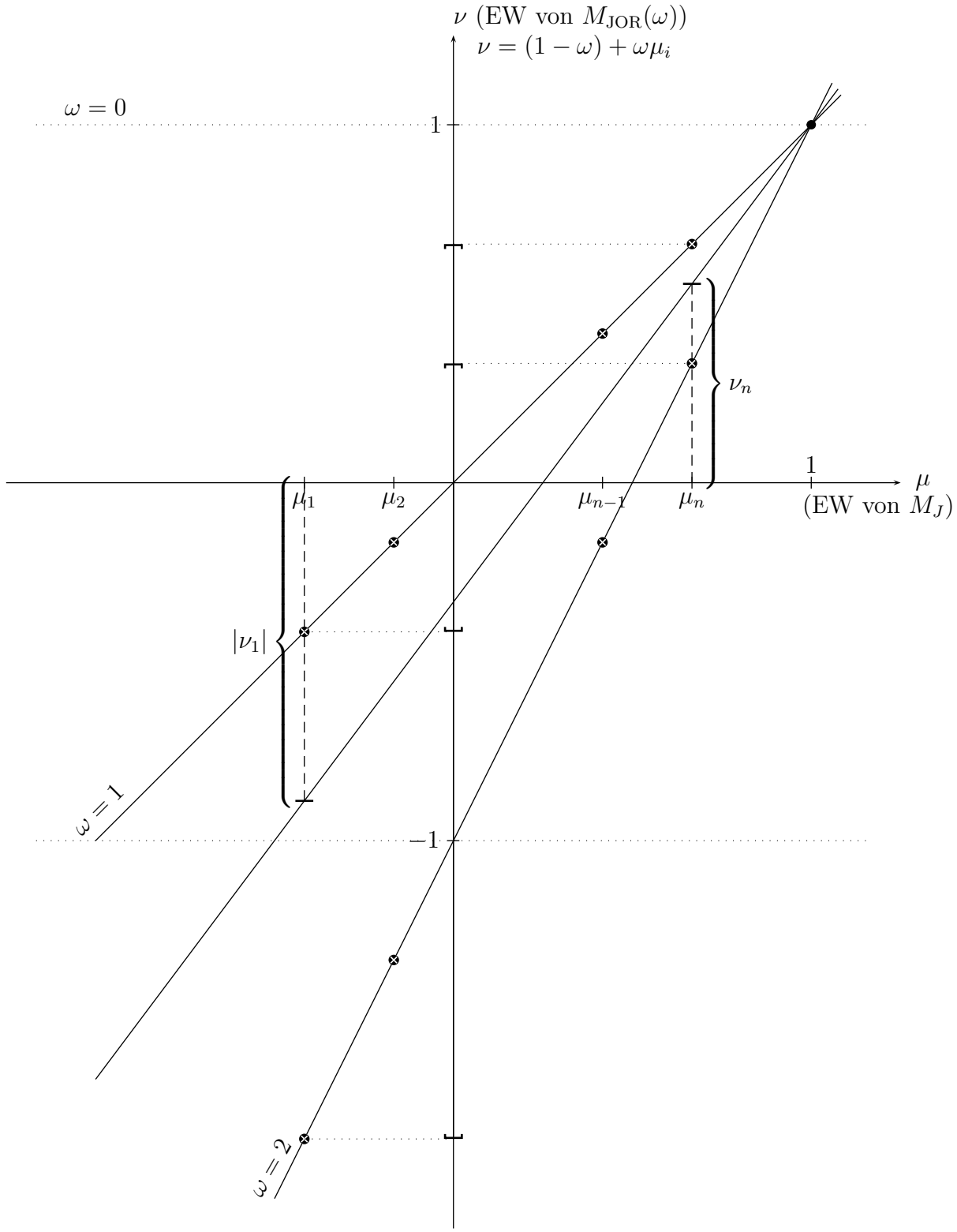


Abbildung 17: Eigenwerte von M_J und M_{JOR} für verschiedene ω

Für dieses ω ist

$$\varrho(M_{\text{JOR}}(\omega)) = -\nu_1 = +\nu_n = 1 + \omega \cdot (\mu_n - 1) = 1 + \frac{2}{2 - \mu_1 - \mu_n} \cdot (\mu_n - 1) = \frac{\mu_n - \mu_1}{2 - \mu_1 - \mu_n}.$$

□

Bemerkung:
Falls $\mu_1 = -\mu_n$, was bei der Diskretisierung der LAPLACE-Gleichung zum Beispiel der Fall ist, so ist $\omega_{\text{opt}} = 1$, d.h. die Relaxation bringt keine Verbesserung.

Auch für das *SOR*-Verfahren lassen sich Konvergenzsätze sowie eine Formel für die optimale Wahl von ω angeben. Die Analyse des SOR-Verfahrens ist allerdings schwieriger als die des JOR-Verfahrens (Lemma 4.5 - 4.7), da bei SOR die Iterationsmatrix

$$M_{\text{SOR}}(\omega) = (D + \omega L)^{-1}[(1 - \omega)D - \omega R]$$

nichtlinear von ω abhängt und außerdem auch bei symmetrisch positiv definitem A im Allgemeinen nicht ähnlich zu einer spd-Matrix ist.
Hier einige Resultate:

Lemma 4.8

a) Es gilt $\varrho(M_{SOR}(\omega)) \geq |1 - \omega|$,
d.h. für $\omega \leq 0$ und für $\omega \geq 2$ ist das SOR-Verfahren divergent.

b) Für diagonaldominantes A (auch für schwach diagonaldominantes und irreduzibles A) ist das SOR-Verfahren für $0 < \omega \leq 1$ konvergent.

c) Für A s.p.d. ist das SOR-Verfahren für $0 < \omega < 2$ konvergent.

Beweis:

a)

$$detM_{\text{SOR}}(\omega) \underset{\text{Det. Prod. Satz}}{=} \frac{det[(1 - \omega)D - \omega R]}{det(D + \omega L)} \underset{det \text{ von } \Delta-\text{Matrizen}}{=} \frac{(1 - \omega)^n detD}{detD} = (1 - \omega)^n$$

Da Determinante = Produkt der Eigenwerte $\in \mathbb{C}$:

$$\prod_{i=1}^{n} \lambda_i = (1 - \omega)^n$$

Ferner

$$\varrho(M_{\text{SOR}}(\omega))^n = \max_{i=1\ldots n} |\lambda_i|^n \geq \prod_{i=1}^{n} |\lambda_i| = |1 - \omega|^n$$

$\Rightarrow$ Behauptung

b) und c) : siehe Schwarz & Köckler Satz 11.13, 11.15

□

Definition 4.9 *Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt* T - Matrix*, falls sie die blockweise Tridiagonalgestalt*

$$A = \begin{pmatrix} D_1 & R_1 & & & & \\ U_1 & \ddots & \ddots & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & R_{s-1} \\ & & & & U_{s-1} & D_s \end{pmatrix}$$

hat, wobei die D_i quadratische *Diagonalmatrizen sind (die R_i, U_i sind im Allgemeinen rechteckig).*

Bemerkung: Matrizen, die bei der Diskretisierung von partiellen Differentialgleichungen mittels sogenannter finiter Differenzen entstehen, sind, bei geeigneter Nummerierung der Gleichungen/Unbekannten, oft T-Matrizen[6]. T-Matrizen haben die Eigenschaft, dass $det(\alpha L + \beta D + \frac{1}{\alpha}R)$ unabhängig von α ist. Diese Eigenschaft kann genutzt werden, um die Eigenwerte der komplizierten Matrix $M_{\mathrm{SOR}}(\omega)$ mit denen der einfach aufgebauten Matrix M_{J} in Beziehung setzen. So lässt sich beweisen (s. Schwarz & Köckler, Kapitel 11.2.3):

Lemma 4.10 *(Varga '62, Young '71)*
Ist A eine T-Matrix mit $a_{ii} \neq 0$, und besitzt $M_J = -D^{-1}(L+R)$ nur reelle Eigenwerte μ_j, und gilt $\varrho(M_J) < 1$, dann ist der optimale Relaxationsparameter ω_{opt} des SOR-Verfahrens gegeben durch

$$\omega_{opt} = \frac{2}{1+\sqrt{1-\varrho(M_J)^2}} \in [1,2)$$

und der zugehörige optimale Spektralradius ist

$$\varrho(M_{SOR}(\omega_{opt})) = \omega_{opt} - 1 \in [0,1).$$

$\varrho(M_{\mathrm{J}})$	ω_{opt}	$\varrho(M_{\mathrm{SOR}}(\omega_{\mathrm{opt}}))$
0.9	1.393	0.393
0.99	1.753	0.753
0.999	1.914	0.914

Bemerkung: Die Abbildung $\omega \to \varrho(M_{\mathrm{SOR}}(\omega))$ hat die Form

d.h. es ist besser ein etwas zu großes als ein etwas zu kleines ω zu wählen. In der Praxis: ω durch Ausprobieren bestimmen.

Allgemeine Bemerkung zu Fixpunktverfahen:

[6]Die Gitterpunkte werden schachbrettartig schwarz oder weiß "eingefärbt", so dass jeder Gitterpunkt von vier Gitterpunkten der jeweils anderen Farbe umgeben ist; dann startet man mit den "weißen" Gleichungen/Unbekannten und endet mit den "schwarzen". Es ergibt sich die obige Struktur mit $s = 2$.

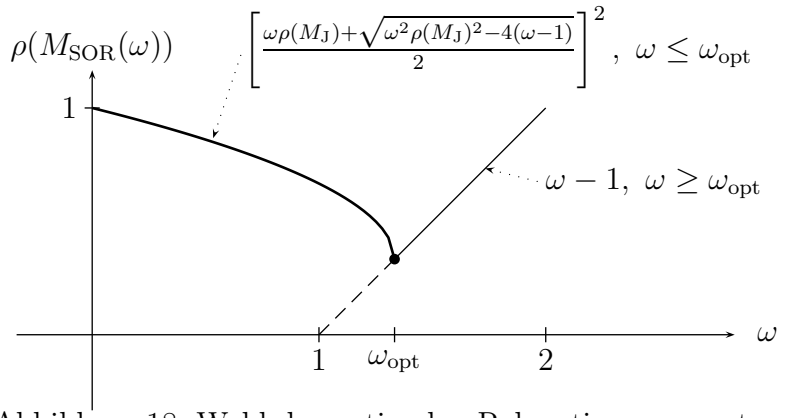


Abbildung 18: Wahl des optimalen Relaxationsparameters.

- Ein prinzipieller Vorteil von Fixpunktverfahren zum Lösen von linearen Gleichungssystemen gegenüber direkten Verfahren ist, dass sich Rundungsfehler nicht akkumulieren können!
- In der Praxis kennt man die Eigenwerte von M_{J} nicht (nicht einmal die von A)
  $\rightarrow$ gutes ω für SOR und JOR durch *Ausprobieren* finden!

5 Interpolation

Problemstellung:
gegeben:

$$n+1 \text{ Wertepaare } (x_i, y_i) \in \mathbb{R} \times \mathbb{R},\ i = 0, \ldots, n$$

gesucht

$$\text{Eine Funktion } \Phi \text{ mit } \Phi(x_i) = y_i,\ i = 0, \ldots, n \tag{5.1}$$

Natürlich gibt es unendlich viele solche Funktionen Φ Man fordert daher außerdem, dass Φ von einer bestimmten "Bauart" ist, d.h. dass Φ von $n+1$ Parametern abhängt, die durch die $n+1$ Bedingungen (5.1) eindeutig festgelegt sind; üblicherweise gibt man einen Funktionenraum V_n mit dim $V_n = n+1$ vor und fordert zuätzlich zu (5.1):

$$\Phi \in V_n.$$

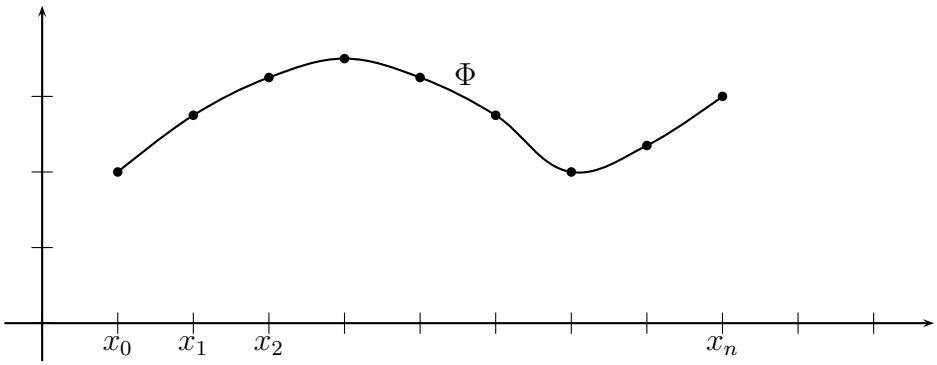


Abbildung 19: Grundidee der Interpolation

Anwendungen der Interpolation:

- z.B. in der Computergrafik:
  (a) Landkarte: Höhenangabe auf Gitternetz bekannt; Fläche soll nach Höhe eingefärbt werden;
  (b) Objekt (Randkörper) wird durch endlich viele Randpunkte gespeichert, soll visualisiert werden.

  [beachte: Es ist auch $(i, x_i),\ (i, y_i)$, d.h. $\Phi_1(i) \stackrel{!}{=} x_i,\ \Phi_2(i) \stackrel{!}{=} y_i$ möglich]

- Oft sind die Wertepaare (x_i, y_i) Funktionswerte einer *unbekannten* (oder komplizierten) Funktion f, d.h. $y_i = f(x_i)$; die Interpolierende kann als Näherung an (Ersatz für) f verwendet werden;

- Als innermathematisches Werkzeug:
  Um Näherungsformeln für Integralberechnung herzuleiten, kann eine vorgegebene, schwierig zu integrierende Funktion f durch eine leicht zu integrierende Funktion Φ mit $\Phi(x_i) = f(x_i)$ ersetzt (approximiert) werden, und als Näherung an $\int f\ dx$ wird $\int \Phi\ dx$ berechnet ($\rightarrow$ Beispiel 0.1 und Kapitel 6)

- Insbesondere bei *fehlerbehafteten* Daten ist es oft sinnvoll, statt eines Interpolationsproblems ein Approximationsproblem zu lösen (d.h. $dim V_n < \sharp$ Wertepaare statt "=" und die Erfüllung von (5.1) wird nicht exakt gefordert).

Beispiele zur Interpolation

1. Polynominterpolation

$$\begin{aligned} V_n &= \Pi_n := \{p \text{ reelles Polynom vom Grad } \leq n\} \\ \Phi(x) &= \sum_{j=0}^{n} a_j x^j \end{aligned}$$

2. Trigonometrische Interpolation

$$\begin{aligned} V_{2n} &= \operatorname{span}\{1, \cos x, \ldots, \cos nx, \sin x, \ldots, \sin nx\} \\ \Phi(x) &= a_o + \sum_{j=1}^{n} a_{2j} \cos ix + \sum_{j=1}^{n} a_{2j-1} \sin ix \end{aligned}$$

3. Spline-Interpolation
   $\Phi|_{[x_i, x_{i+1}]}$ ist Polynom eines gewissen Grades, und an den Übergangsstellen x_i hat Φ eine gewisse Regularität

4. Rationale Interpolation (siehe Stoer, Band 1, Kapitel 2.2)

$$\Phi(x) = \frac{\sum_{j=0}^{n} a_j x^j}{\sum_{j=0}^{n} b_j x^j}$$

Definition 5.1 *(*lineare *Interpolationsaufgabe)*
Gegeben sei ein $n+1$-dimensionaler Vektorraum $V_n = \operatorname{span}\{\varphi_0, \ldots, \varphi_n\}$ mit Basis $\varphi_0, \ldots, \varphi_n$, und Funktionswerte y_i zu paarweise verschiedenen *Stützstellen x_i, $i = 0, \ldots, n$.*
Gesucht sind die Parameter a_i, $i = 0, \ldots, n$, so, dass

$$\Phi(x) := \sum_{j=0}^{n} a_j \varphi_j(x)$$

die Interpolationsbedingung

$$[\Phi(x_i) =] \sum_{j=0}^{n} a_j \varphi_j(x_i) = y_i \quad \forall i = 0, \ldots, n, \tag{5.2}$$

erfüllt. Das heißt

$$a = \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} \in \mathbb{R}^{n+1}$$

ist Lösung des linearen Gleichungssystems

$$Aa = y, \ \textit{mit } y = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n+1}, \ A = (a_{ij}) \in \mathbb{R}^{(n+1)\times(n+1)}, \ a_{ij} = \varphi_j(x_i) \ .$$

Bemerkung 5.2

(a) Die Beispiele 1. - 3. sind lineare *Interpolationsaufgaben; 4. nicht, da die Funktionen* Φ *einen Untervektorraum bilden;*

(b) Existenz einer Lösung ist, da das System (5.2) quadratisch ist, äquivalent zur Eindeutigkeit *einer Lösung.*
Das System (5.2) ist also genau dann eindeutig lösbar, wenn $Kern(A) = 0$*, d.h. wenn:*

$$y_0 = \ldots = y_n = 0 \ \Rightarrow \ a_0 = \ldots = a_n = 0 \ [\Leftrightarrow \Phi \equiv 0]$$

also insbesondere $\Phi \equiv 0$*. Diese Eigenschaft "*$y_0 = \ldots = y_n = 0 \Rightarrow \Phi \equiv 0$*" der Interpolierenden heißt* Unisolvenz*.*

5.1 Polynominterpolation

Wähle $V_n = \Pi_n$ = Raum der Polynome vom Grad kleiner gleich n.

Satz 5.3 *Zu beliebigen paarweise verschiedenen Stützstellen* $x_0, \ldots, x_n \in \mathbb{R}$ *und Daten* $y_0, \ldots, y_n \in \mathbb{R}$ *gibt es genau ein Polynom* $\Phi \in \Pi_n$ *mit*

$$\Phi(x_i) = y_i \quad \forall i = 0, \ldots, n. \tag{5.3}$$

Beweis : Nach Bermekung 5.2 reicht es die Unisolvenz zu zeigen. Sei $y_0 = \ldots = y_n = 0$. Sei $\Phi \in \Pi_n$ und es gelte (5.3). Nach dem Hauptsatz der Algebra hat Φ höchstens n Nullstellen, oder $\Phi \equiv 0$. Nach (5.3) hat Φ mindestens $n+1$ Nullstellen, also $\Phi \equiv 0$. □

anderer Beweis : Wähle als Basis von V_n die Monome

$$\varphi_j(x) := x^j \quad , \quad j = 0, \ldots, n$$

Dann hat die Matrix A die Einträge $\varphi_j(x_i) = x_i^j$. Dies ist die sogenannte VANDERMONDE-*Matrix*, deren Determinate nicht Null ist:

$$\det A = \prod_{j>i} x_j - x_i \neq 0 \qquad \text{[siehe L.A. I, Blatt 11]}$$

$\Rightarrow$ Behauptung. □

Die VANDERMONDE-Matrix ist voll besetzt und schlecht konditioniert (z.B. bei $n = 20$, äquidistante Punkte: $\kappa_\infty \simeq 11000$). Das Lösen des linearen Gleichungssystems (5.2) mit oben vorgeschlagener Monom-Basis φ_j ist daher unpraktikabel.
Stattdessen sollte man versuchen die Basis $\varphi_0, \dots \varphi_j$ so zu wählen, dass das zugehörige lineare Gleichungssystem (5.2) leicht zu lösen ist[7]. Wir können sogar $A = Id$ erreichen! Die Bedingung dazu lautet:

$$\varphi_j(x_i) = \delta_{ij} \quad \forall i, j = 0, \dots, n. \tag{5.4}$$

Eine Basis $\varphi_0, \dots, \varphi_n$ von Π_n, die die Eigenschaft (5.4) besitzt, lautet:

$$\varphi_j(x) := \frac{\prod\limits_{\substack{i=0 \\ i \neq j}} (x - x_i)}{\prod\limits_{\substack{i=0 \\ i \neq j}} (x_j - x_i)} \quad , \quad j = 0, \dots, n. \tag{5.5}$$

Definition und Satz 5.4 *Die Polynome φ_j aus (5.5) heißen* LAGRANGE-Polynome *zu den (paarweise verschiedenen) Stützstellen $x_0, \dots, x_n$ (*LAGRANGE*: 1736-1813). Sie erfüllen offensichtlich $\varphi_j \in \Pi_n$ sowie (5.4), also $A = Id$. Sie sind linear unabhängig*[8]*. Die Interpolationsaufgabe 5.1 wird, da $A = Id$, gelöst durch $a_i = y_i \ \forall i = 0, \dots, n$, also*

$$\Phi(x) = \sum_{j=0}^{n} y_j \varphi_j(x) \tag{5.6}$$

mit φ_j aus (5.5) ist die eindeutig bestimmte Lösung der Polynominterpolationsaufgabe in LAGRANGE-Darstellung. *Die Darstellung (5.5)/(5.6) heißt* LAGRANGEsche Interpolationsformel.

Einschub - Horner-Schema :

Möchte man ein Polynom der Form

$$\Phi(x) = \sum_{i=0}^{n} \alpha_i x^i \tag{5.7}$$

auswerten, so sind $\sum\limits_{i=1}^{n} i = \frac{n}{2}(n+1)$ Multiplikationen (und n Additionen) nötig. Es geht sehr viel effizient in der Form

$$\Phi(x) = \sum_{i=0}^{n} \alpha_i x^i = \alpha_0 + x(\alpha_1 + x(\alpha_2 + \dots + x(\alpha_{n-1} + x\alpha_n) \dots)) \tag{5.8}$$

z.B. $5x^3 + 7x^2 - 3x + 2 = 2 + x(-3 + x(7 + x \cdot 5))$

[7] Die resultierende Interpolierende Φ ist natürlich unabhängig von der Wahl der Basis von Π_n; die Schwierigkeit der Berechnung kann aber durchaus von der Wahl der Basis abhängen!

[8] denn wären sie linear abhängig, also

$$\exists a \neq 0 : \ \sum_{j=0}^{n} a_j \varphi_j \equiv 0 \ \Rightarrow \ \sum_{i=0}^{n} a_j \varphi_j(x_i) = 0 \ \forall i \ (\Rightarrow a \in Kern(A) = Kern(Id) \lightning)$$

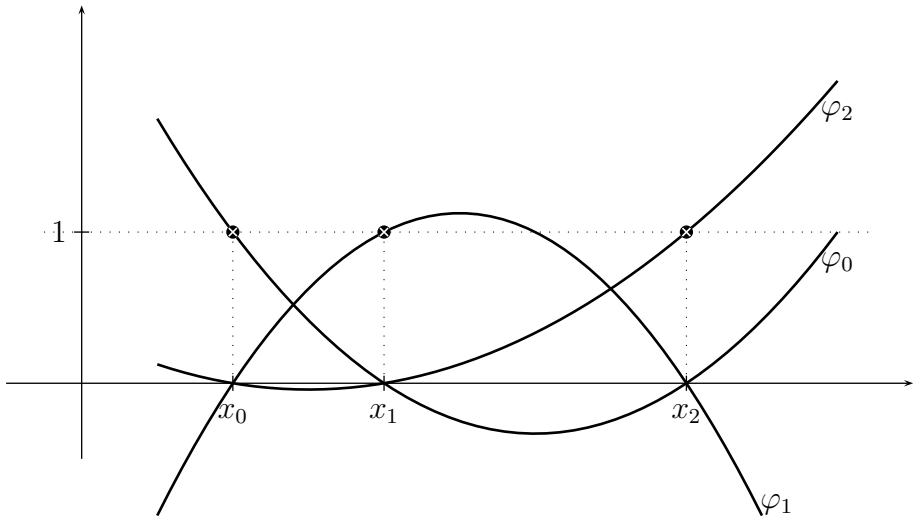


Abbildung 20: LAGRANGE-Basis

Die Darstellung (5.8) heißt HORNER-*Schema*. Die Evaluation erfordert nur n Multiplikationen und n Additionen. Der Algorithmus

```
1:  p := α_n
2:  for i := n-1 to 0 do {
3:      p := p * x + α_i
4:  }
```

liefert $p = \Phi(x)$.

Polynominterpolation nach NEWTON

Nachteile der Polynominterpolation nach LAGRANGE:

- Die Darstellung (5.6)/(5.5) ist ungünstig in der Hinsicht, dass *jede* Evaluation des Interpolationspolynoms $O(n^2)$ Operationen erfordert. Dieser Nachteil kann abgeschwächt werden, indem man die Darstellung (5.6)/(5.5) umrechnet mit Aufwand $O(n^2)$ in eine Darstellung, deren Evaluation mit $O(n)$ möglich ist; siehe z.B. Oevel S.328-329.
- Bei Hinzufügen eines neuen Punktes (x_{n+1}, y_{n+1}) muss alles neu berechnet werden, da jedes Φ_j und jedes a_j von *allen* (x_i, y_i) abhängt[9].

Wir nehmen statt der Basis-Funktionen (5.5) nun eine Basis, bei der die j-te Funktion nur von den ersten j Stützstellen $(x_0, \ldots, x_{j-1})$ abhängt:

$$\begin{aligned} \varphi_0(x) &:\equiv 1 \\ \varphi_1(x) &:= x - x_0 \\ \varphi_2(x) &:= (x - x_0)(x - x_1) \\ &\vdots \\ \varphi_n(x) &:= (x - x_0) \cdot \ldots \cdot (x - x_{n-1}) \end{aligned} \tag{5.9}$$

[9]Hinzufügen von Punkten: ggf. in der Computergrafik: näherkommende Objekte detaillierter darstellen

Nimmt man ein x_{n+1}/φ_{n+1} hinzu, ändern sich die $\varphi_0, \ldots, \varphi_n$ nicht. Man erhält mit (5.9) die Darstellung

$$\Phi(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0) \cdot \ldots \cdot (x - x_{n-1}) \tag{5.10}$$

des Interpolationspolynoms.
Die Matrix $A = (\varphi_j(x_i))_{i,j}$ aus (5.2) gebildet mit (5.9) ist offensichtlich eine Dreiecksmatrix (denn $\varphi_j(x_i) = 0$ für $i < j$):

$$\begin{pmatrix} 1 & & & & 0 \\ 1 & x_1 - x_0 & & & \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & & \\ \vdots & \vdots & & \ddots & \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \ldots & \prod_{i=0}^{n-1}(x_n - x_i) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ \vdots \\ \vdots \\ y_n \end{pmatrix} \tag{5.11}$$

Dies kann per Vorwärtseinsetzen mit $O(n^2)$ Operationen gelöst werden:

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

$\Rightarrow$ Jedes a_i hängt nur von den ersten i Wertepaaren ab. Für die a_i kann man eine *Rekursionsformel* angeben (siehe Satz 5.6). Dazu definieren wir:

Definition 5.5 *Wir setzen*

$$[x_i] := y_i$$

und rekursiv

$$[x_{i_0}, x_{i_1}, \ldots, x_{i_k}] := \frac{[x_{i_0}, x_{i_2}, \ldots, x_{i_k}] - [x_{i_1}, x_{i_2}, \ldots, x_{i_k}]}{x_{i_0} - x_{i_1}}, \tag{5.12}$$

wobei $i_0, \ldots, i_n$ paarweise verschieden seien. Diese Terme heißen k-te (NEWTONsche) dividierte Differenzen.

Bemerkung: (5.12) weicht ab von der Standard-Darstellung; siehe später.

Satz 5.6 *(Newton-Interpolation)*
Die Koeffizienten a_i der Darstellung (5.10) des Interpolationspolynoms Φ durch die Stützwerte $(x_0, y_0), \ldots, (x_n, y_n)$ berechnen sich zu

$$a_i = [x_i, \ldots, x_0],$$

also

$$\begin{aligned} \Phi(x) &= [x_0] + [x_1, x_0](x - x_0) + [x_2, x_1, x_0](x - x_0)(x - x_1) + \ldots \\ &\ldots + [x_n, \ldots, x_0](x - x_0) \cdot \ldots \cdot (x - x_{n-1}). \end{aligned} \tag{5.13}$$

Beweis (per Induktion nach n):

Sei Φ_n das Interpolationspolynom durch $(x_0, y_0), \ldots, (x_n, y_n)$

$n = 0$**:**

$$\Pi_0 \ni \Phi_0 \equiv y_0 \overset{(5.12)}{=} [x_0] \;\Rightarrow\; (5.13) \text{ gilt für } n = 0.$$

$n - 1 \to n$**:** Es habe Φ_{n-1} die Darstellung

$$\Phi_{n-1}(x) = [x_0] + [x_1, x_0](x - x_0) + \ldots + [x_{n-1}, \ldots, x_0](x - x_0) \cdot \ldots \cdot (x - x_{n-2})$$

Wir zeigen, dass

$$\Phi_{n-1}(x) + [x_n, \ldots, x_0](x - x_0) \cdot \ldots \cdot (x - x_{n-1}) \tag{5.14}$$

das Interpolationspolynom durch $(x_0, y_0), \ldots, (x_n, y_n)$, also gleich $\Phi_n(x)$, ist[10]. Für $j = 0, \ldots, n-1$ erfüllt es die Interpolationseigenschaft (5.1):

$$(5.14)|_{x=x_j} = \Phi_{n-1}(x_j) + [x_n, \ldots, x_0] \cdot 0 \underset{\text{I.V.}}{=} y_i \quad \forall j = 0, \ldots, n-1$$

Es bleibt (5.1) für $j = n$ zu prüfen, also

$$(5.13) \overset{x=x_n}{\Rightarrow} \Phi_{n-1}(x_n) + [x_n, \ldots, x_0] \prod_{i=0}^{n-1}(x_n - x_i) \overset{!}{=} y_n$$

$$\begin{aligned}
y_n - \Phi_{n-1}(x_n) \overset{\text{I.V.}}{=} & \underbrace{\underbrace{y_n}_{=[x_n]} - [x_0]}_{=[x_n, x_0] \cdot (x_n - x_0)} - [x_1, x_0](x_n - x_0) - [x_2, x_1, x_0](x_n - x_0)(x_n - x_1) - \ldots \\
& \ldots - [x_{n-1}, \ldots, x_0] \prod_{i=0}^{n-2}(x_n - x_i) \\
= & \underbrace{([x_n, x_0] - [x_1, x_0])}_{=[x_n, x_1, x_0] \cdot (x_n - x_1)} \cdot (x_n - x_0) - [x_2, x_1, x_0](x_n - x_0)(x_n - x_1) - \ldots \\
& \ldots - [x_{n-1}, \ldots, x_0] \prod_{i=0}^{n-2}(x_n - x_i) \\
= & \underbrace{([x_n, x_1, x_0] - [x_2, x_1, x_0])}_{=[x_n, x_2, x_1, x_0] \cdot (x_n - x_2)}(x_n - x_0)(x_n - x_1) - \ldots \\
& \ldots - [x_{n-1}, \ldots, x_0] \prod_{i=0}^{n-2}(x_n - x_i) \\
= & ([x_n, x_{n-2}, x_{n-3}, \ldots, x_1, x_0] - [x_{n-1}, x_{n-2}, \ldots, x_0]) \prod_{i=0}^{n-2}(x_n - x_i) \\
= & [x_n, \ldots, x_0] \prod_{i=0}^{n-1}(x_n - x_i)
\end{aligned}$$

[10] Einsetzen von $\Phi_{n-1}(x)$ nach (5.13) in (5.14) liefert die Induktionsbehauptung. (5.14) ist Polynom vom Grad kleiner gleich n.

□

Folgerung 5.7 *Die Reihenfolge der Argumente spielt bei den dividierten Differenzen (5.12) keine Rolle, d.h.*

$$[x_{p(0)}, \ldots, x_{p(k)}] = [x_0, \ldots, x_k]$$

für jede Permutation p auf $\{0, \ldots, k\}$; insbesondere ist

$$[x_{i_0}, \ldots, x_{i_k}] = \frac{[x_{i_0}, \ldots, x_{i_{k-1}}] - [x_{i_1}, \ldots, x_{i_k}]}{x_{i_0} - x_{i_k}};$$

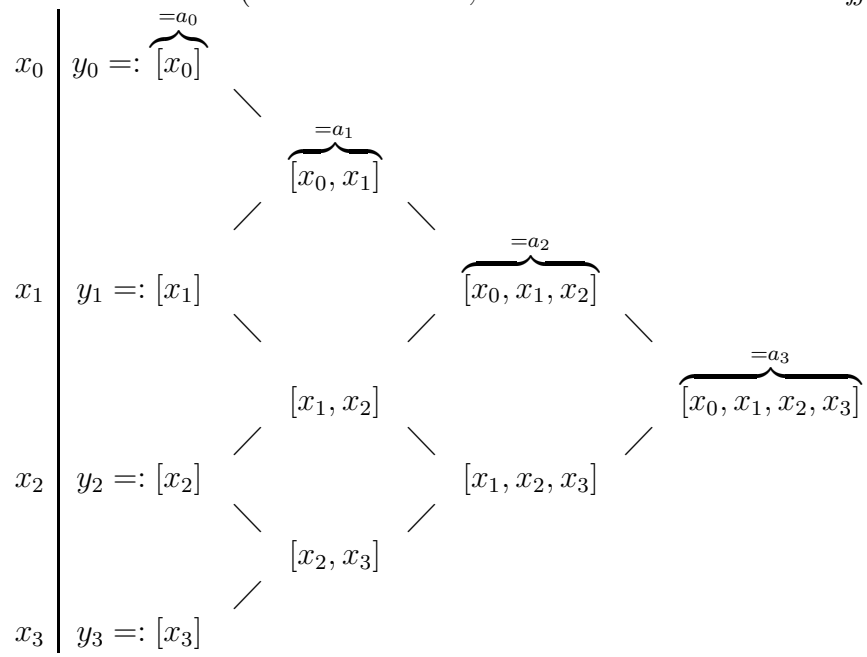
dies *(und nicht (5.12)) ist die allgemein übliche Definition der dividierten Differenzen.*

Beweis : Nach Satz 5.6 ist $[x_{i_0}, \ldots, x_{i_k}]$ einer der Koeffizienten des eindeutig bestimmten Interpolationspolynoms vom Grad kleiner gleich k durch $(x_{i_0}, y_{i_0}), \ldots, (x_{i_k}, y_{i_k})$. Dieses ändert sich nicht bei Permutation der Stützwerte! Also

$$\begin{aligned} [x_{i_0}, \ldots, x_{i_k}] &= [x_{i_0}, x_{i_k}, x_{i_1}, \ldots, x_{i_{k-1}}] \underset{\text{Def.(5.12)}}{=} \frac{[x_{i_0}, x_{i_1}, \ldots, x_{i_{k-1}}] - [x_{i_k}, x_{i_1}, \ldots, x_{i_{k-1}}]}{x_{i_0} - x_{i_k}} \\ &= \frac{[x_{i_0}, \ldots, x_{i_{k-1}}] - [x_{i_1}, \ldots x_{i_k}]}{x_{i_0} - x_{i_k}} \end{aligned}$$

□

Rechenschema (*Newton-Schema, Schema der dividierten Differenzen*):



Bemerkung 5.8

(a) Wiederholte Differenzbildung kann zu Auslöschung führen (für $n \gtrsim 20$, oder $n \gtrsim 30$ bei double precision können *Ergebnisse unbrauchbar sein).*

(b) Die Evaluation der Newton'schen Form (5.10)/(5.13) von $\Phi(x)$ muss natürlich nicht mit Aufwand $O(n^2)$ erfolgen; in Analogie zum Horner-Schema kann man

$$\Phi(x) = c_0 + (x - x_0) \cdot [c_1 + (x - x_1) \cdot [c_2 + (x - x_2) \cdot [\ldots [c_{n-1} + c_n \cdot (x - x_{n-1})] \ldots]]]$$

verwenden $\rightarrow O(n)$ Aufwand.

Fehlerabschätzung bei der Polynominterpolation

Bisher: $(x_0, y_0), \ldots (x_n, y_n)$ sind "diskrete" gegebene Wertepaare.
Nun: Die Werte (x_i, y_i) sind Wertepaare einer unbekannten Funktion $f : y_i = f(x_i)$.
Frage: Wie groß ist der Abstand zwischen f und dem Interpolationspolynom Φ?

Satz 5.9 *Seien $x_0, \ldots, x_n \in [a, b]$ paarweise verschiedene Stützstellen, sei $f \in \mathcal{C}^{n+1}([a, b])$. Dann existiert zu jedem $x \in [a, b]$ ein $\xi = \xi(x) \in I$, wobei I_x das kleinste Intervall ist, das $x, x_0, \ldots, x_n$ enthält, so dass*

$$f(x) - \Phi(x) = \frac{w(x)}{(n+1)!} f^{(n+1)}(\xi), \tag{5.15}$$

wobei

$$w(x) := \prod_{i=0}^{n} (x - x_i).$$

Insbesondere ist also

$$|f(x) - \Phi(x)| \leq \frac{|w(x)|}{(n+1)!} \|f^{(n+1)}\|_\infty \tag{5.16}$$

$$\|f - \Phi\|_\infty \leq \frac{\|w\|_\infty \|f^{(n+1)}\|_\infty}{(n+1)!} \tag{5.17}$$

wobei $\|\cdot\|_\infty$ die supremums-Norm auf $[a, b]$ ist.

Beweis : Es reicht (5.15) zu zeigen. Für $x = x_i$ ist (5.15) trivialerweise erfüllt ("$0 = 0$"). Sei also $x \neq x_i \ \forall i$ fest gewählt. Definiere

$$h(z) := w(x) \cdot [f(z) - \Phi(z)] - w(z) \cdot [f(x) - \Phi(x)] \tag{5.18}$$

Es ist dann $h(x) = 0$ sowie

$$h(x_i) = w(x) \cdot [\underbrace{f(x_i) - \Phi(x_i)}_{=0}] - \underbrace{w(x_i)}_{=0} \cdot [f(x) - \Phi(x)] = 0 \quad \forall i = 0, \ldots, n$$

$\Rightarrow h$ hat mindestens $n + 2$ paarweise verschiedene Nullstellen.

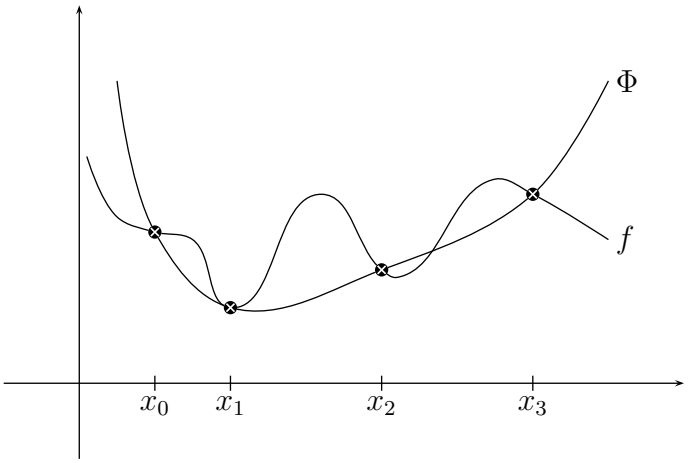


Abbildung 21: Funktion f mit Interpolierender Φ.

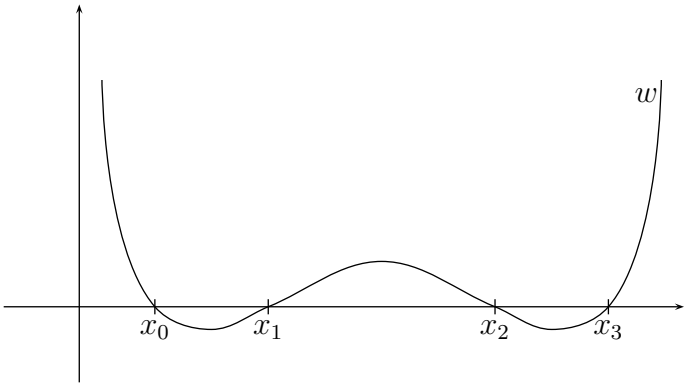


Abbildung 22: Gestalt der Funktion $w(x)$ bei vier Stützstellen $x_0, ..., x_3$.

Satz von Rolle: h' hat mindestens $n+1$ verschiedene Nullstellen,
h'' hat mindestens n verschiedene Nullstellen,
$\vdots$
$h^{(n+1)}$ hat mindestens 1 Nullstelle.

Sei ξ eine Nullstelle von $h^{(n+1)}$.

$$
\begin{aligned}
0 &= h^{(n+1)}(\xi) \underset{\substack{(5.18)\\ n+1\text{-mal abgeleitet}}}{=} \\
&= w(x) \cdot [f^{(n+1)}(\xi) - \underbrace{\Phi^{(n+1)}(\xi)}_{=0,\ \text{da } \Phi \in \Pi_n}] - \underbrace{w^{(n+1)}(\xi)}_{\substack{=(n+1)! \\ \text{da } w \in \Pi_{n+1} \text{ mit Leitkoeff. 1}}} [f(x) - \Phi(x)] \\
&= w(x) f^{(n+1)}(\xi) - (n+1)! [f(x) - \Phi(x)] \\
&\Rightarrow \quad (5.15)
\end{aligned}
$$

Folgerung 5.10 *Falls die Stützstellen äquidistant liegen,*

$$x_i - x_j = (i-j)\cdot h,$$

so ist auf $[a,b] = [x_0, x_n]$*:*

$$\|f - \Phi\|_\infty \le \frac{h^{n+1}}{n+1}\|f^{(n+1)}\|_\infty. \tag{5.19}$$

Beweis : Ist $x \in [x_{i_0}, x_{i_0+1}]$, so ist

$$\begin{aligned} |x - x_{i_0}|, |x - x_{i_0+1}| &\le h, \\ |x - x_{i_0-1}|, |x - x_{i_0+2}| &\le 2h, \text{ etc.} \end{aligned}$$

Das Produkt dieser Schranken für $i = 0, \ldots, n$ wird am größten, wenn $x \in [x_0, x_1]$ oder $x \in [x_{n-1}, x_n]$; es gilt folglich

$$|w(x)| \le 1h \cdot 1h \cdot 2h \cdot 3h \cdot \ldots \cdot nh = n!\, h^{n+1}.$$

Mit (5.16) folgt die Behauptung. □

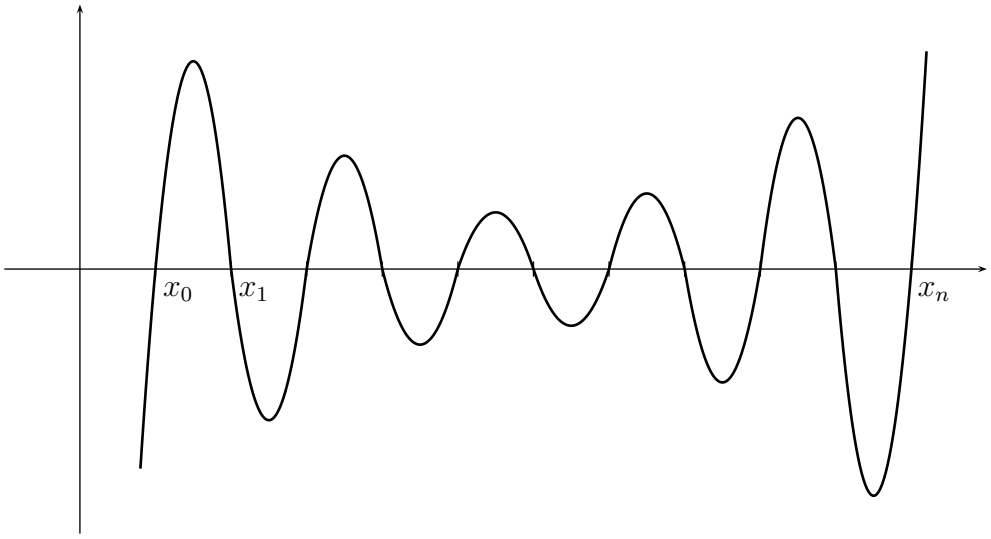


Abbildung 23: Verhalten von ω bei vielen äquidistanten Stützstellen

Es stellt sich die Frage, ob für $n \to \infty$ der Fehler $\|f - \Phi_n\|_\infty$, oder zumindest punktweise $f(x) - \Phi_n(x)$ für jedes $x \in [a,b]$, gegen Null geht. Dazu gibt es das berühmte *Runge-Beispiel* (Runge 1901):

$$f(x) = \frac{1}{1+x^2}$$

auf $[a,b] = [-5,5]$, äquidistante Stützstellen $x_i = -5 + \frac{10i}{n}$ $i = 0, \ldots, n$: *Keine* Punktweise Konvergenz von Φ_n gegen f auf $[a,b]$ (s. Übung). Hintergrund (vgl. (5.15)/(5.16)): $f^{(n+1)}$ wächst hier schneller als $\frac{w(x)}{(n+1)!}$ bzw. $\frac{h^{n+1}}{n+1}$ fällt! Für die Runge-Funktion f und n gerade ist z.B. $\|f^{(n+1)}\|_\infty = (n+1)!$ Es gilt sogar:

Satz 5.11 *(Satz von Faber, 1914)*
Zu jeder Folge von Intervallzerlegungen von $[a, b]$ kann man eine Funktion $f \in \mathcal{C}([a, b])$ finden, so dass die zugehörige Folge von Interpolationspolynomen nicht gleichmäßig gegen f konvergiert.
Zitiert nach: Stoer, Band 1, Satz 2.1.4.4
Weitere Infos: Quarteroni, Sacco, Saleri, Numerische Mathematik 2, Kapitel 8.1

Maßnahmen, falls Oszillationen auftreten:

- Knotenpunkte zum Rand des Intervalls $[a, b]$ hin verdichten, z.B.: $x_i = \frac{(2i+1)\pi}{2n+2}$ (Tschebyscheff - Knoten) oder $x_i = \cos\frac{\pi i}{n}, i = 0, \ldots, n$ (Gauß-Lobatto-Knoten) für $[a, b] = [-1, 1] \Rightarrow \|w\|_n$ wird minimal bei dieser Wahl. *Stückweise* Polynomielle Interpolation mit *festem* Polynomgrad (Abschätzung (5.18) liefert für Polynomgrad n *fest*, $h \to 0, \|f - \Phi\|_\infty \to 0$, aber: Φ hat dann global nur $\mathcal{C}^0$-Regularität).
- noch besser: Spline Interpolation, s. Kapitel 5.2

5.2 Spline-Interpolation

Wir haben gesehen: Für $n \to \infty$ kann die Polynominterpolation zu Oszillationen neigen. Stückweise polynomielle Interpolation liefert eine Interpolationsfunktion, die an den Übergangsstellen lediglich $\mathcal{C}^0$-Regularität hat. Dieser Mangel kann behoben werden durch *Spline-Interpolation*:

Definition 5.12 *Gegeben sei eine Zerlegung* $\Delta : a = x_0 < x_1 < \ldots < x_n = b$ *des Intervalls* $[a, b]$ *mit Feinheit* $|\Delta| := \max\limits_{i=0,\ldots,n-1} |x_{i+1} - x_i|$. *Der Raum*

$$S_k(\Delta) := \{\Phi \in \mathcal{C}^{k-1}([a, b]) \mid \Phi|_{[x_i, x_{i+1}]} \in \Pi_k \ \forall i = 0, \ldots, n-1\},$$

$k \in \mathbb{N}$, *heißt* Raum der Spline-Funktionen vom Grad k.
Im Fall $k = 1, 2, 3$ *heißt* $S_k(\Delta)$ Raum der linearen, quadratischen, kubischen Splines.

Lineare Splines:

Eine Basis des Raums $S_1(\Delta)$ bilden offensichtlich die sogenannten *Hütchen-Funktionen* (engl.: hat functions) $\varphi_j : [a, b] \to \mathbb{R}, j = 0, \ldots, n$, die gegeben sind durch

$$\varphi_j(x_i) = \delta_{ij}, \quad \varphi_j|_{[x_i, x_{i+1}]} \in \Pi_1 \ \forall\, i = 0, ..., n-1. \tag{5.20}$$

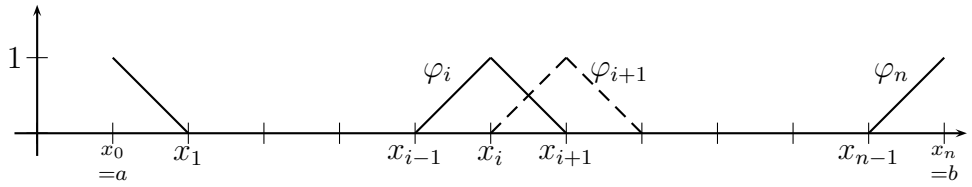


Abbildung 24: Hütchen-Funktionen: Basis von $S_1(\Delta)$.

Diese Basis des Interpolationsraumes ist, anders als diejenigen aus Kapitel 5.1, *lokal*, was heißen soll, dass die Träger $\text{supp}\varphi_i = \{x \in [a, b] \mid \varphi_i(x) \neq 0\}$ deutlich kleiner als $[a, b]$ sind und sich nur "wenige" Träger überlappen: $\text{supp}\varphi_i \cap \text{supp}\varphi_j = \emptyset \forall i, j$ mit $|i - j| > 1$.

Nach (5.20) ist die Systemmmatrix $A = (\varphi_j(x_i))$ die Einheitsmatrix! Die Interpolationsaufgabe

$$\text{"Finde } \Phi \in S_1(\Delta) \text{ mit } \Phi(x_i) = y_i \ \forall i = 0, \ldots, n\text{"}$$

ist offensichtlich eindeutig lösbar durch

$$\Phi(x) = \sum_{j=0}^{n} y_j \varphi_j(x), \text{ d.h. } a_j = y_j.$$

Satz 5.13 *(Fehlerabschätzung)*
Sei $f \in \mathcal{C}^2([a, b])$. *Die Lösung* $\Phi \in S_1(\Delta)$ *der obigen Interpolationsaufgabe erfüllt*

$$|f(x) - \Phi(x)| \leq \frac{(x_{i+1} - x_i)^2}{8} \max_{\xi \in [x_i, x_{i+1}]} |f''(\xi)| \ \forall x \in [x_i, x_{i+1}],$$

also

$$\|f - \Phi\|_\infty \leq \frac{1}{8} |\Delta|^2 \|f''\|_\infty.$$

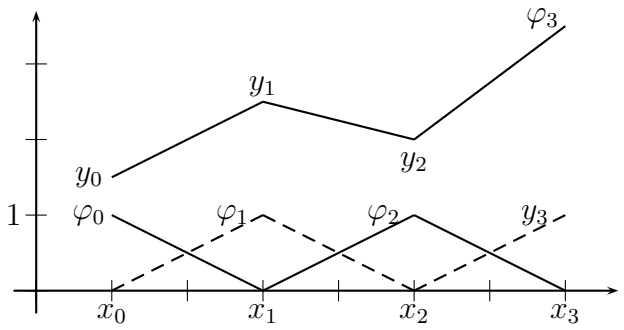


Abbildung 25: Lösung der Interpolationsaufgabe durch die Hütchen-Funktionen.

Beweis : Auf jedem Teilintervall $[x_i, x_{i+1}]$ kann Satz 5.9 mit $n = 1$ angewendet werden.

$$\Rightarrow \quad f(x) - \Phi(x) = \frac{(x - x_i)(x - x_{i+1})}{2!} f''(\xi), \ \xi \in [x_i, x_{i+1}].$$

$|(x - x_i)(x - x_{i+1})|$ wird maximal auf $[x_i, x_{i+1}]$ maximal an der Stelle $x = \frac{x_i + x_{i+1}}{2}$, dort beträgt sein Wert $-\left(\frac{x_{i+1} - x_i}{2}\right)^2$. $\Rightarrow$ Behauptung □

Kubische Splines:

Wir wollen die Interpolationsaufgabe

$$\text{“Finde } \Phi \in S_3(\Delta) \text{ mit } \Phi(x_i) = y_i \ \forall i = 0, \ldots, n\text{”} \tag{5.21}$$

lösen. Mit dem Ansatz

$$\Phi|_{[x_i, x_{i+1}]}(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \ i = 0, \ldots, n - 1 \tag{5.22}$$

haben wir auf jedem der n Teilintervalle vier Freiheitsgrade. Es sollen an jedem der $n - 1$ inneren Knoten die drei Übergangsbedingungen

$$\Phi^{(j)}(x_i - 0) = \Phi^{(j)}(x_i + 0), \ j = 0, 1, 2; \ i = 1, \ldots, n - 1,$$

erfüllt werden, sowie die $n + 1$ Interpolationsbedingungen aus (5.21).

$$\underbrace{4n}_{\text{Freiheitsgrade}} \quad -\underbrace{(3 \cdot (n - 1) + (n + 1))}_{\text{Bedingungen}} = 2,$$

Das heißt wir müssen noch zwei weitere Bedingungen stellen, wenn wir die eindeutige Existenz der Lösung erreichen wollen. Es gibt mehrere Möglichkeiten, sogenannte Randbedingungen zu fordern:

1. Periodische Randbedingungen:

   $$\Phi'(x_n) = \Phi'(x_0), \ \Phi''(x_n) = \Phi''(x_0).$$

   Dies ist sinnvoll, falls f periodisch.

2. Vollständige Randbedingungen:

$$\Phi'(x_0) = f'_0 := f'(x_0), \Phi'(x_n) = f'_n := f'(x_n),$$

falls $f'(x_0), f'(x_n)$ exisitieren und bekannt sind.

3. Natürliche Randbedingungen:

$$\Phi''(x_0) = \Phi''(x_n) = 0$$

($\rightarrow$ "natürliche Splines").

Zur Bedeutung der kubischen Splines:

Man stelle sich einen biegsamen Stab (engl.: Spline) im $\mathbb{R}^2$ vor, der an den Stellen

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

fixiert ist. Wie ist die Form $f(x)$ des Stabes? Der Stab versucht seine Deformationsenergie

$$E(f) = \int_a^b \frac{f''(x)^2}{1 + f'(x)^2}\ dx$$

zu minimieren. Falls $|f'(x)|$ "klein" ist, ist

$$E(f) \approx \int_a^b f''(x)^2\ dx =: E_{\text{approx}}(f).$$

Man kann zeigen: Die kubische Spline-Interpolierende mit Randbedingung 1 (oder 2 oder 3) ist gerade die Lösung des Optimierungsproblems $E_{\text{approx}}(f) \rightarrow \min$ (Oevel Kapitel 8.2.5 oder Knabner-Skript) im Raum der $\mathcal{C}^2$-Funktionen die $\Phi(x_i) = y_i$, sowie die Randbedingung 1 (bzw. 2 bzw. 3) erfüllen.
Ein biegsamer Stab wird insbesondere $\Phi''(x) = 0$ auf $\mathbb{R} \setminus (a, b)$ erfüllen, also Randbedingung 3 ($\Rightarrow$ "natürliche Randbedinung").
Berechnung der Lösung von (5.21) (mit natürlichen Randbedingungen):

Ansatz (5.22), $h_i := x_{i+1} - x_i$.

$$\text{Bedingung} \quad \Phi(x_i) = y_i \ \Rightarrow\ a_i = y_i, \, {\scriptstyle i=0,\ldots,n-1,} \tag{5.23}$$

$$\Phi(x_{i+1}) = y_{i+1} \ \Rightarrow\ a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1}, \, {\scriptstyle i=0,\ldots,n-1,} \tag{5.24}$$

$$\Phi'(x_i - 0) = \Phi'(x_i + 0) \ \Rightarrow\ b_{i-1} + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2 = b_i, \, {\scriptstyle i=1,\ldots,n-1,} \tag{5.25}$$

$$\Phi''(x_{i+1} - 0) = \Phi''(x_{i+1} + 0) \ \Rightarrow\ \begin{array}{l} 2c_i + 6d_i h_i = 2c_{i+1}, \, {\scriptstyle i=0,\ldots,n-2,} \\ 2c_0 = 0,\ 2c_{n-1} + 6d_{n-1}h_{n-1} = 0 \text{ (nat. RB)} \end{array} \tag{5.26}$$

für die $4n$ Unbekannten $a_i, b_i, c_i, d_i,\ i = 0, \ldots, n-1$. Setze $c_n = 0 \Rightarrow$ (5.26) gilt für $i = 0, \ldots, n-1$.

$$(5.26) \quad \Rightarrow \quad d_i = \frac{c_{i+1} - c_i}{3h_i},\ i = 0, \ldots, n-1 \tag{5.27}$$

Dies in (5.24) eingesetzt:

$$\Rightarrow \quad b_i = \frac{y_{i+1}-y_i}{h_i} - c_i h_i - \frac{c_{i+1}-c_i}{3h_i}\cdot h_i^2 = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{3}(c_{i+1}+2c_i),\ i=0,\dots,n-1. \tag{5.28}$$

Nun b_i, b_{i-1} gem. (5.28) und d_i nach (5.27) in (5.25) einsetzen:

$$\begin{aligned}
&\Rightarrow \quad \frac{y_i-y_{i-1}}{h_{i-1}} - \frac{h_{i-1}}{3}(c_i+2c_{i-1}) + 2c_{i-1}h_{i-1} + h_{i-1}(c_i-c_{i-1}) = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{3}(c_{i+1}+2c_i)\\
&\Leftrightarrow \quad \frac{1}{3}h_{i-1}c_{i-1} + \frac{2(h_i+h_{i-1})}{3}c_i + \frac{1}{3}h_i c_{i+1} = \frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}} =: \frac{1}{3}\delta_i \quad |\cdot 3
\end{aligned}$$

für $i = 1,\dots,n-1$. Dies liefert (unter Beachtung von $c_0 = c_n = 0$) das lineare $(n-1)\times(n-1)$ Gleichungssystem

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & h_{n-2} \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ \vdots \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \vdots \\ \vdots \\ \vdots \\ \delta_{n-1} \end{pmatrix}.$$

Die Matrix ist (stark) diagonaldominant, das lineare Gleichungssystem ist daher eindeutig lösbar und kann also per LR-Zerlegung ohne Pivotisierung gelöst werden. Aufwand nur $O(n)$. Kondition: $\kappa_2 \le 3$ für *beliebiges* $n \in \mathbb{N}$ mit äquidistanten Punkten, wie später gezeigt werden kann. Die Größen a_i, b_i, d_i köennen über (5.23),(5.27),(5.28) unter Beachtung von $c_n = 0$ berechnet werden. Siehe Oevel S. 348-351 für weitere Randbedingungen.

Satz 5.14 *(Fehlerabschätzung)*
Sei $f \in \mathcal{C}^4([a,b])$ und sei $\Phi \in S_3(\Delta)$ der interpolierende Spline, mit einer der drei Randbedingungen. Dann gilt

$$\|\Phi^{(k)} - f^{(k)}\|_\infty \le c|\Delta|^{4-k}\|f^{(4)}\|_\infty \quad \textit{für } k = 0,1,2.$$

Beweis: Oevel, Stoer Band 1, Seite 86 (ähnlicher Satz)

Also: Für Stützstellenzahl $n \to \infty$ z.B. im äqudistanten Fall $|\Delta| = \frac{b-a}{n}$ konvergiert die Spline-Interpolation gegen f von 4. Ordnung; bei der üblichen Polynominterpolation (Kapitel 5.1) ist Konvergenz nicht gesichert.

5.3 Trigonometrische Interpolation

Ziel: Interpolation von Daten (x_i, y_i) durch *periodische* Funktionen Φ. Dies ist sinnvoll, wenn die Daten y_i Werte einer periodischen Funktion $y_i = f(x_i)$ sind. Wir setzen voraus, dass die Periodenlänge 2π ist (andernfalls: falls f die Periode Länge $L > 0$ hat, so hat $\tilde{f}(x) := f\left(\frac{Lx}{2\pi}\right)$ Periode 2π).

Im ganzen Kapitel 5.3 gehen wir von *äquidistanten Stützstellen*

$$x_l = \frac{2\pi l}{n} \quad , \quad l = 0,\dots.n-1 \tag{5.29}$$

aus (ggf. $l \in \mathbb{Z}$, da f periodisch vorausgesetzt).
Am einfachsten stellt sich die trigonometrische Interpolationsaufgabe in $\mathbb{C}$ dar:

Satz 5.15 *(komplexe trigonometrische Interpolation)*
Seien Daten $y_0, \dots, y_{n-1} \in \mathbb{C}$ zu den Stützstellen (5.29) gegeben, $n \in \mathbb{N}$. Dann gibt es genau eine Funktion $\Phi : \mathbb{R} \to \mathbb{C}$ aus dem $\mathbb{C}$-Vektorraum

$$V_n := span\{e^{ikx},\ k = 0, \dots, n-1\}$$

$$\begin{aligned} mit\ \Phi(x_l) &= y_l,\ l = 0, \dots, n-1, && (5.30)\\ und\ zwar\ \Phi(x) &= \sum_{k=0}^{n-1} d_k e^{ikx}, && (5.31)\\ mit\ d_k &= \frac{1}{n}\sum_{j=0}^{n-1} y_j e^{-2\pi i \frac{jk}{n}} \in \mathbb{C},\ k = 0, \dots, n-1. && (5.32) \end{aligned}$$

(siehe auch Modifikation $\tilde{V}_n, \tilde{\Phi}$ statt V_n, Φ am Ende von Kapitel 5.3)

[Für späteren Gebrauch wollen wir d_k nach Formel (5.32) für alle $k \in \mathbb{Z}$ setzen]

Beweis : Wir zeigen, dass (5.31)-(5.32) eine Lösung von (5.30) ist:

$$\Phi(x_l) \underset{\substack{(5.29),(5.31)\\(5.32)}}{=} \frac{1}{n}\sum_{k=0}^{n-1}\sum_{j=0}^{n-1} y_j e^{-2\pi i\frac{jk}{n}} e^{2\pi i\frac{kl}{n}} = \frac{1}{n}\sum_{j=0}^{n-1} y_j \sum_{k=0}^{n-1}\left(e^{2\pi i\frac{l-j}{n}}\right)^k$$

Für $r \in \mathbb{Z}$ ist

$$\sum_{k=0}^{n-1}\left(e^{2\pi i\frac{r}{n}}\right)^k = \begin{cases} \frac{1-e^{2\pi i\frac{r}{n}n}}{1-e^{2\pi i\frac{r}{n}}} = 0, & \text{falls } e^{2\pi i\frac{r}{n}} \neq 1,\ \text{also } \frac{r}{n} \notin \mathbb{Z} \\ \sum_{k=0}^{n-1} 1^k = n, & \text{falls } \frac{r}{n} \in \mathbb{Z} \end{cases}$$

also

$$\sum_{k=0}^{n-1}(e^{2\pi i\frac{l-j}{n}})^k = n \cdot \delta_{lj} \quad \text{, für } 0 \le l, j \le n-1$$

also

$$\Phi(x_l) = \frac{1}{n} \cdot \sum_{j=0}^{n-1} y_j \cdot n \cdot \delta_{lj} = y_l \ \Rightarrow\ (5.30) \text{ wird erfüllt}$$

Das Problem: „Finde zum Datenvektor $y \in \mathbb{C}^n$ einen Vektor $d \in \mathbb{C}^n$, so dass (5.30) gilt“, kann als lineares Gleichungssystem (vgl. (5.2))

$$\begin{pmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \dots & \omega^{0\cdot(n-1)} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & & \omega^{1\cdot(n-1)} \\ \vdots & & & \vdots \\ \omega^{(n-1)\cdot 0} & \omega^{(n-1)\cdot 1} & \dots & \omega^{(n-1)\cdot(n-1)} \end{pmatrix} \cdot \begin{pmatrix} d_0 \\ \vdots \\ \vdots \\ d_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ \vdots \\ y_{n-1} \end{pmatrix} \qquad (5.33)$$

in $\mathbb{C}$ mit Matrix $A = (\varphi_k(x_j))$ geschrieben werden, wobei

$$\omega := e^{\frac{2\pi i}{n}},$$

denn

$$\varphi_k(x_j) = e^{ikx_j} = e^{\frac{i2\pi kj}{n}} = \omega^{kj}.$$

Wir haben gezeigt, dass dieses lineare Gleichungssystem für *jede* rechte Seite $y \in \mathbb{C}^n$ (mindestens) eine Lösung $d \in \mathbb{C}^n$ hat. Da das lineare Gleichungssystem quadratisch ist, folgt hieraus die Eindeutigkeit der Lösung. □

Bemerkung 5.16

- *(5.32) besagt, dass die Lösung $d = A^{-1}y \in \mathbb{R}^n$ expilizit als*

$$d_k = \frac{1}{n}\sum_{j=0}^{n-1} y_j \omega^{-jk}$$

angegeben werden kann, also

$$\begin{pmatrix} d_0 \\ \vdots \\ d_{n-1} \end{pmatrix} = \underbrace{\frac{1}{n} \cdot \begin{pmatrix} \omega^{-0\cdot 0} & \dots & \omega^{-0\cdot(n-1)} \\ \vdots & & \vdots \\ \omega^{-(n-1)\cdot 0} & \dots & \omega^{-(n-1)\cdot(n-1)} \end{pmatrix}}_{=A^{-1}} \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix} \tag{5.34}$$

d.h wir haben eine explizite Darstellung der Matrix A^{-1}.

- *Aufwand: $O(n^2)$ (später dazu mehr; Man folgert leicht: $\frac{1}{\sqrt{n}}$ A ist unitär).*

Nun zur Übertragung auf den "*reellen*" Fall. Die Interpolierende soll eine Linerakombination von Sinus- und Kosinusschwingungen $\sin\frac{kx}{n}, \cos\frac{kx}{n}$ sein:

Satz 5.17 *Sei $n \in \mathbb{N}$ ungerade, $n = 2\tilde{n} + 1$. Seien Daten $y_0, \dots, y_{n-1} \in \mathbb{C}$ (ggf.: $\in \mathbb{R}$) zu den Stützstellen (5.29) gegeben.*
Dann gibt es genau eine Funktion $\Phi : \mathbb{R} \to \mathbb{C}$ aus dem $\mathbb{C}$-Vektorraum

$$V_n := span\{\cos kx, k = 0, \dots, \tilde{n}\} \times span\{\sin kx, k = 1, \dots, \tilde{n}\}$$

die die Interpolationsbedingung (5.30) erfüllt, und zwar

$$\tilde{\Phi}(x) = a_0 + \sum_{k=1}^{\tilde{n}} a_k \cos kx + \sum_{k=1}^{\tilde{n}} b_k \sin kx \tag{5.35}$$

mit

$$\begin{aligned} a_0 &= d_0, \\ a_k &= d_k + d_{n-k}, \ k = 1, \dots, \tilde{n}, \\ b_k &= i(d_k - d_{n-k}), k = 1, \dots, \tilde{n}, \end{aligned} \tag{5.36}$$

wobei $d_0, \dots, d_n$ wie in (5.32).

Bemerkung: Zu den n Daten $y_0, \dots, y_{n-1}$ suchen wir n Freiheitsgrade $a_0, \dots, a_{\tilde{n}}, b_1, \dots, b_{\tilde{n}}$. Die Berechnung der a_k, b_k wird auf (5.32)/(5.34) zurückgeführt! Der Aufwand beträgt offensichtlich $O(n^2)$ (Matrix-Vektor-Multiplikation (5.34))

Beweis : Wir wissen aus Satz 5.14, dass

$$\Phi(x) = \sum_{k=0}^{n-1} d_k e^{ikx} \text{ mit } d_0, \dots, d_{n-1} \text{ aus (5.32)}$$

die Interpolationsbedingung (5.30) erfüllt, also

$$\begin{aligned}
y_l = \Phi(x_l) &= \sum_{k=0}^{n-1} d_k e^{ikx_l} = \sum_{k=0}^{\tilde{n}} d_k e^{2\pi i \frac{kl}{n}} + \sum_{k=\tilde{n}+1}^{2\tilde{n}} d_k e^{2\pi i \frac{kl}{n}} \\
&= \sum_{k=0}^{\tilde{n}} d_k e^{2\pi i \frac{kl}{n}} + \sum_{k=\tilde{n}+1}^{2\tilde{n}} d_k e^{2\pi i \frac{(k-n)l}{n}} \\
&= \sum_{k=0}^{\tilde{n}} d_k e^{2\pi i \frac{kl}{n}} + \sum_{\tilde{k}=1}^{\tilde{n}} d_{n-\tilde{k}} e^{-2\pi i \frac{\tilde{k}l}{n}} \\
&= d_0 e^0 + \sum_{k=1}^{\tilde{n}} (d_k e^{2\pi i \frac{kl}{n}} + d_{n-k} e^{-2\pi i \frac{kl}{n}}) \\
&= d_0 + \sum_{k=1}^{\tilde{n}} \left[d_k \left(\cos \frac{2\pi kl}{n} + i \sin \frac{2\pi kl}{n} \right) + d_{n-k} \left(\cos \frac{2\pi kl}{n} - i \sin \frac{2\pi kl}{n} \right) \right] \\
&= \underbrace{d_0}_{=:a_0} + \sum_{k=1}^{\tilde{n}} \underbrace{(d_k + d_{n-k})}_{=:a_k} \cos kx_l + \underbrace{i(d_k - d_{n-k})}_{=:b_k} \sin kx_l = \tilde{\Phi}(x_l) \\
&\Rightarrow \text{ (5.34)/(5.35) ist eine Lösung des Interpolationsproblems}
\end{aligned}$$

Da die Aufgabe linear ist und genau so viele Freiheitsgrade wie Daten hat, folgt aus der Lösbarkeit für beliebige rechte Seite die Eindeutigkeit der Lösung. □

Folgerung 5.18 *Die a_k, b_k aus (5.35)/(5.36) haben die Darstellung*

$$\begin{aligned}
a_0 &= \frac{1}{n} \sum_{j=0}^{n-1} y_j, \\
a_k &= \frac{2}{n} \sum_{j=0}^{n-1} y_j \cos \frac{2\pi jk}{n}, \\
b_n &= \frac{2}{n} \sum_{j=0}^{n-1} y_j \sin \frac{2\pi jk}{n}.
\end{aligned} \tag{5.37}$$

Insbesondere sind, falls die y_j reell sind, alle a_k, b_k reell.

Beweis :

$$\begin{aligned} d_k \pm d_{n-k} &\underset{(5.32)}{=} \frac{1}{n}\sum_{j=0}^{n-1} y_j (e^{-2\pi i \frac{jk}{n}} \pm \underbrace{e^{-2\pi i \frac{j(n-k)}{n}}}_{=e^{+2\pi i \frac{jk}{n}}}) \\ &= \frac{1}{n}\sum_{j=0}^{n-1} y_j \left[\left(\cos\frac{2\pi jk}{n} - i \sin\frac{2\pi jk}{n} \right) \pm \left(\cos\frac{2\pi jk}{n} + i \sin\frac{2\pi jk}{n} \right) \right] \\ &\Rightarrow \quad \text{Behauptung} \end{aligned}$$

□

Definition 5.19 *Die Abbildung*

$$y \in \mathbb{C}^n \to d \in \mathbb{C}^n \quad \textit{gemäß (5.32)/(5.33)}$$

heißt komplexe diskrete FOURIER-Transformation*, die Abbildung*

$$y \in \mathbb{R}^n \to (a,b) \in \mathbb{R}^n \quad \textit{gemäß (5.37)}$$

heißt reelle diskrete FOURIER-Transformation. *Die Umkehrabbildungen heißen* diskrete komplexe *bzw.* reelle FOURIER-Synthese *oder auch* FOURIER-Rücktransformation.

Satz 5.15 deckt leider nur den Fall n ungerade ab. Im Fall $n = 2\tilde{n}$ *gerade* ist Satz 5.15 folgendermaßen abzuwandeln:

$$V_n = span\{\cos kx, k = 0, \ldots, \tilde{n}\} \times span\{\sin kx, k = 1, \ldots, \tilde{n}-1\} \tag{5.38}$$

(beachte: dim $V_n = (\tilde{n}+1) + (\tilde{n}-1) = n = \sharp$ Daten),

$$\tilde{\Phi}(x) = a_0 + \sum_{k=1}^{\tilde{n}} a_k \cos kx + \sum_{k=1}^{\tilde{n}-1} b_k \sin kx \tag{5.39}$$

mit

$$\begin{aligned} a_0 &= d_0, \; a_{\tilde{n}} = d_{\tilde{n}} \\ a_k &= d_k + d_{n-k}, \; k = 1, \ldots, \tilde{n}-1 \\ b_k &= i(d_k - d_{n-k}), \; k = 1, \ldots, \tilde{n}-1 \end{aligned} \tag{5.40}$$

Beweis : Φ aus (5.31) erfüllt

$$\begin{aligned} y_l = \Phi(x_l) &= \sum_{k=0}^{n-1} d_k e^{ikx_l} = \sum_{k=0}^{\tilde{n}-1} d_k e^{ikx_l} + \sum_{k=\tilde{n}}^{2\tilde{n}-1} d_k e^{i(k-n)x_l} \\ &= \sum_{k=0}^{\tilde{n}-1} d_k e^{ikx_l} + \sum_{\tilde{k}=1}^{\tilde{n}} d_{n-\tilde{k}} e^{-\tilde{k}x_l} \qquad (\tilde{k} = n-k) \\ &= d_0 + \sum_{k=1}^{\tilde{n}-1} (d_k e^{ikx_l} + d_{n-k} e^{-ikx_l}) + d_{\tilde{n}} e^{-i\tilde{n}x_l} \\ &= d_0 + \sum_{k=1}^{\tilde{n}-1} (d_k + d_{n-k}) \cos kx_l + \sum_{k=1}^{\tilde{n}-1} i(d_k - d_{n-k}) \sin kx_l + d_{\tilde{n}} \cos \tilde{n}x_l \\ &\quad -d_{\tilde{n}} i \underbrace{\sin \tilde{n}x_l}_{=\sin \frac{2\pi l \tilde{n}}{n} = \sin \pi l = 0} \\ &=: \tilde{\Phi}(x_l). \end{aligned}$$

Die Funktion

$$\tilde{\Phi}(x) := d_0 + \sum_{k=1}^{\tilde{n}-1} (d_k + d_{n-k}) \cos kx + \sum_{k=1}^{\tilde{n}-1} i(d_k - d_{n-k}) \sin kx + d_{\tilde{n}} \cos \tilde{n}x$$

löst also das Interpolationsproblem, liegt (anders als Φ) im Raum (5.38). □

Die Schnelle FOURIER-Transformation (Fast FOURIER Transform, FFT):

[Geht zurück auf Good '58 und Cooley & Tukey '65]
Die "naive" Berechnung der komplexen diskreten FOURIER-Koeffizienten mittels (5.32)/(5.34) erfordert offensichtlich $O(n^2)$ Operationen (Matrix-Vektor-Multiplikation). Die reelle diskrete FOURIER-Transformation ebenfalls. Man kann stattdessen jedoch *rekursiv* die Berechnung der diskreten FOURIER-Transformation eines Datensatzes der Länge $2m$ auf die Berechnung der FOURIER-Koeffizienten von zwei Datensetzen der Länge jeweils m zurückführen:

Satz 5.20 *Es sei*

$$d_k^{(2m)} = d_k(y_0, y_1, \ldots, y_{2m-1}) \in \mathbb{C}, k = 0, \ldots 2m-1,$$

der Satz der diskreten FOURIER*-Koeffizienten des Datensatzes*

$$y_0, \ldots, y_{2m-1} \in \mathbb{C}.$$

Es seien $d_k(y_0, y_2, \ldots, y_{2m-2})$ und $d_k(y_1, y_3, \ldots, y_{2m-1}), k = 0, \ldots m-1$ die FOURIER*-Koeffizienten der Datensätze $y_0, y_2, \ldots, y_{2m-2}$, bzw. $y_1, y_3, \ldots, y_{2m-1}$.*
Dann gilt:

$$\begin{aligned} d_k^{(2m)} &= \frac{1}{2}[d_k(y_0, y_2, \ldots, y_{2m-2}) + e^{-i\pi\frac{k}{m}} d_k(y_1, y_3, \ldots, y_{2m-1})] \\ d_{k+m}^{(2m)} &= \frac{1}{2}[d_k(y_0, y_2, \ldots, y_{2m-2}) - e^{-i\pi\frac{k}{m}} d_k(y_1, y_3, \ldots, y_{m-1})] \end{aligned}$$

für $k = 0, \ldots, m-1$

Beweis :

$$\begin{aligned}
d_k^{(2m)} &= \frac{1}{2m}\left[\sum_{\substack{j=0\\ j \text{ gerade}}}^{2m-1} y_j e^{-2\pi i \frac{jk}{2m}} + \sum_{\substack{j=0\\ j \text{ ungerade}}}^{2m-1} y_j e^{-2\pi i \frac{jk}{2m}}\right] \\
&= \frac{1}{2m}\left[\sum_{j=0}^{m-1} y_{2j} e^{-2\pi i \frac{jk}{m}} + \sum_{j=0}^{m-1} y_{2j+1} e^{-2\pi i \frac{(j+\frac{1}{2})k}{m}}\right] \\
&= \frac{1}{2}\left[\frac{1}{m}\sum_{j=0}^{m-1} y_{2j} e^{-2\pi i \frac{jk}{m}} + e^{-\pi i \frac{k}{m}} \frac{1}{m}\sum_{j=0}^{m-1} y_{2j+1} e^{-2\pi i \frac{jk}{m}}\right] \quad \forall k = 0, \ldots, 2m-1 \qquad (5.41) \\
&= \frac{1}{2}\left[d_k(y_0, y_1, \ldots, y_{2m-2}) + e^{-\pi i \frac{k}{m}} d_k(y_1, y_2, \ldots, y_{2m-1})\right] \text{ für } k = 0, \ldots, m-1
\end{aligned}$$

Die $d_k^{(2m)}$ für $k \geq m$ formen wir weiter um: Für $k = 0, \ldots, m-1$ ist

$$\begin{aligned}
d_{k+m}^{(2m)} &\underset{(5.35)}{=} \frac{1}{2}\left[\frac{1}{m}\sum_{j=0}^{m-1} y_{2j} e^{-2\pi i \frac{j(k+m)}{m}} + \underbrace{e^{-\pi i \frac{k+m}{m}}}_{=-e^{\pi i k}} \frac{1}{m}\sum_{j=0}^{m-1} y_{2j+1} e^{-2\pi i \frac{j(k+m)}{m}}\right] \\
&= \frac{1}{2}\left[\frac{1}{m}\sum_{j=0}^{m-1} y_{2j} e^{-2\pi i \frac{jk}{m}} - e^{-\pi i \frac{k}{m}} \frac{1}{m}\sum_{j=0}^{m-1} y_{2j+1} e^{-2\pi i \frac{jk}{m}}\right] \\
&= \frac{1}{2}[d_k(y_0, y_2, \ldots, y_{2m-2}) - e^{-\pi i \frac{k}{m}} d_k(y_1, y_3, \ldots, y_{2m-1})], \; k = 0, \ldots, m-1
\end{aligned}$$

□

Bemerkung:

- Analog geht man mit der *Rück*transformation (FOURIER-Synthese) um (positives statt negatives Vorzeichen im Exponenten; Vorfaktor "$\frac{1}{2}$" weglassen)
- Analog kann man die diskrete FOURIER-Transformation von Datensätzen der Länge $p \cdot m$ auf die diskrete FOURIER-Transformation von p Datensätzen der Länge m zurückführen
- Ist $n = 2^N$ Zweierpotenz, so kann man mittels Satz 5.20 *rekursiv* die diskrete FOURIER-Transformation auf die triviale diskrete FOURIER-Transformation von Datensätzen der Länge 1 ($d_0 = y_0$) zurückführen. Man nennt die diskrete FOURIER-Transformation dann *schnelle* FOURIER-*Transformation (FFT).*

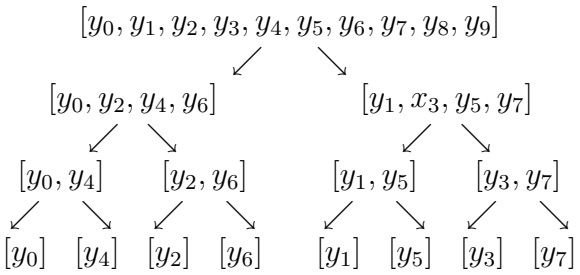
Aufwand der FFT:

Sie $A(n)$ der Aufwand für die FFT eines Datensatzes der Länge $2^N = n$. Offensichtlich gilt $A(0) = 0$, $A(N+1) = 2 \cdot A(N) + 2^{N+1}$. Per Induktion folgt:

$$A(N) = N \cdot 2^N = n \log_2 n$$

Als *rekursiver* Algorithmus kann die FFT offensichtlich leicht implementiert werden. Etwas effizienter (Speicherplatz, Rechenzeit) ist es, die FFT mittels Schleifen zu implemetieren. Dazu veranschaulichen wir uns die Vorgänge zunächst an einem Datensatz der Länge $n = 8$:

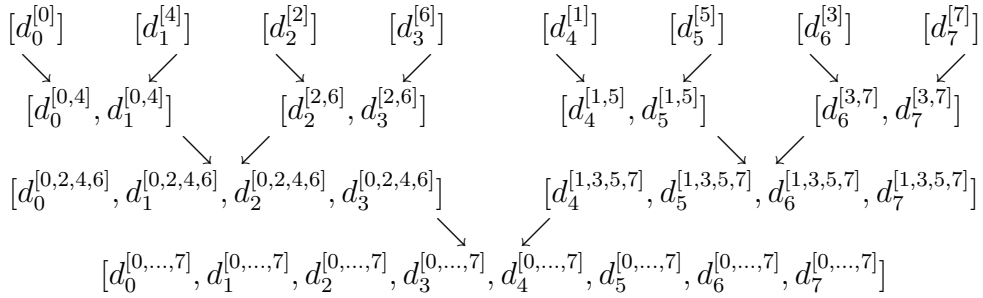
(1) Wiederholte Zerlegung des Datensatzes:



Sei $d_k^{[j_1,\ldots,j_k]}$ der k-te diskrete FOURIER-Koeffizient des Datensatzes $[y_{j_1}, ..., y_{j_m}]$, $k = 0, ..., m-1$.

(2) FOURIER-Transformation der Datensätze der Länge 1: $d_0^{[j]} := y_j$

(3)



In Schritt (1) ist offensichtlich eine *Permutation* $\sigma_N : \{0, \ldots, 2^N - 1\} \to \{0, \ldots, 2^N - 1\}$ der Daten durchzuführen (für $N = 3 : \sigma(0) = 0, \sigma(1) = 4, \sigma(2) = 2, \sigma(3) = 6, \ldots$). Diese Permutation ist leicht durchzuführen, wenn man die Indizes $0, \ldots, 2^N - 1$ in Binärdarstellung betrachtet:

$$\begin{aligned} 000 &\leftrightarrow 000 \\ 001 &\leftrightarrow 100 \\ 010 &\leftrightarrow 010 \\ 011 &\leftrightarrow 110 \\ \vdots &\leftrightarrow \vdots \end{aligned}$$

Die Folge von Bits ist einfach *umzukehren*! Grund: Im ersten Schritt werden alle geraden Zahlen, d.h. alle Zahlen der Form $b_0 b_1 \ldots b_{N-2} 0$ auf Zahlen $< \frac{n}{2}$, d.h. auf die Zahlen $0 b_0 b_1 \ldots b_{N-2}$ geschickt, ungerade Zahlen $b_0 b_1 \ldots b_{N-2} 1$ werden auf Zahlen $\geq \frac{n}{2}$, also auf $1 b_0 b_1 \ldots b_{N-2}$ geschickt. Allgemein:

$$b_0 \ldots b_{N-1} \to b_{N-1} b_0 b_1 \ldots b_{N-2}.$$

Im zweiten Schritt bleibt das führende Bit unverändert und das Bit b_{N-2} wandert an die zweite Stelle.

$$b_{N-1} b_0 b_1 \ldots b_{N-2} \to b_{N-1} b_{N-2} b_0 b_1 \ldots b_{N-3}$$

usw. bis $b_{N-1} b_{N-2} ... b_1 b_0$ erreicht ist. Die Permutation $\sigma_N : \{0, \ldots, 2^N - 1\} \to \{0, \ldots, 2^N - 1\}$,

$$\sum_{k=0}^{N-1} b_k 2^k \mapsto \sum_{k=0}^{N-1} b_{N-1-k} 2^k$$

wobei alle $b_k \in \{0,1\}$, heißt auch *bit-reverse*. Manche Programmiersprachen beinhalten einen Befehl zur Durchführung eines Bit-Reverse. Andernfalls kann σ wie folgt implementiert werden:

```
1:  σ_N[0] := 0
2:  for k = 0 to N-1 do
3:      for j = 0 to 2^k-1 do
4:          σ_N[j + 2^k] := σ_N[j] + 2^(N-k-1)
```

Die Werte für σ_N werden vorab berechnet und in einem Integer-Array gespeichert. Der Algorithmus FFT(N,Y,D) lautet dann wie folgt:

```
1:  Eingabe: N ∈ ℕ_0, Datensatz Y = (y_0, y_1, ..., y_{2^N-1}) ∈ ℂ^{2^N},
2:  Ausgabe: Datensatz D = (d_0, d_1, ..., d_{2^N-1}) ∈ ℂ^{2^N}
3:  Berechne Permutation σ_N (s.o.)
4:  for k = 0 to 2^N-1 do D[k] := Y[σ_N[k]] / 2^N
5:  for r = 1 N do {
6:      for k = 0 to 2^(r-1)-1 do {
7:          for j = 0 to 2^(N-r)-1 do {
8:              t := e^(-2π i k/2^r) · D[j2^r + 2^(r-1) + k]
9:              D[j2^r + k] := D[j2^r + k] + t
10:             D[j2^r + 2^(r-1) + k] := D[j2^r + k] - t
11:         }
12:     }
13: }
```

Kommentare:

- Zeile 4: Permutation
- Zeile 5: r = Rekursionstiefe, $r = 1 \triangleq$ "unten"
- Zeile 6: 2^{r-1} = halbe Länge eines Datensatzes auf Stufe r
- Zeile 7: 2^{N-r} = Anzahl der Datensätze der Länge 2^r auf Stufe r

Bemerkung:

- Die Werte $e^{-\frac{2\pi i k}{2^r}}, k = 0, \ldots, 2^{r-1}$ sowie die Werte $2^j, j = 0, \ldots, N$ können zuvor tabelliert werden.
- Die *reelle* diskrete FOURIER-Transformation kann mittels (5.40) auf obige FFT zurückgeführt werden.
- Schnelle Rücktransformation: siehe Übung.

Anwendungen: Filtern von Daten, Datenkompression, Bildbearbeitung,...

Mehrdimensionale FFT (hier 2-dimensional):
gegeben: Datensatz

$$y_{k_1k_2}, k_1 = 0, \dots n_1 - 1, k_2 = 0, \dots, n_2 - 1$$

gesucht:

$$d_{k_1k_2} = \frac{1}{n_1} \sum_{j_1=0}^{n_1-1} [\frac{1}{n_2} \sum_{j_2=0}^{n_2-1} y_{j_1j_2} e^{-2\pi i \frac{j_2k_2}{n_2}}] e^{-2\pi i \frac{j_1k_1}{n_1}}$$

Wenn n_1 und n_2 Zweierpotenzen sind ($n_1 = 2^{N_1}, n_2 = 2^{N_2}$), kann die schnelle FOURIER-Transformation in $2D$ folgendermaßen implementiert werden:

Für jedes feste $j_1 = 0, \dots, n_1 - 1$ führe die 1-dimensionale FFT auf dem Datensatz $(y_{j_1j_2})_{j_2=0,\dots,n_2-1}$ durch. Auf dem Ergebnisfeld $(d^*_{j_1j_2})_{\substack{j_1=0,\dots,n_1-1 \\ j_2=0,\dots,n_2-1}}$ führe für jedes feste $j_2 = 0, \dots, n_2-1$ die 1-dimensionale FFT auf dem Datensatz $(y_{j_1j_2})_{j_1=0,\dots,n_1}$ (= Spalte) durch.
Wahlweise auch umgekehrt, denn es gilt auch

$$d_{k_1k_2} = \frac{1}{n_2} \sum_{j_2=0}^{n_2-1} [\frac{1}{n_1} \sum_{j_1=0}^{n_1-1} y_{j_1j_2} e^{-2\pi i \frac{j_1k_1}{n_1}}] e^{-2\pi i \frac{j_2k_2}{n_2}}$$

Aufwand: $n_2 \cdot n_1 \log_2 n_1 + n_1 \cdot n_2 \log_2 n_2 = n_1 n_2 \log_2(n_1 n_2) = n \log_2 n$, wobei $n = n_1 \cdot n_2 =$ Größe des Datensatzes. **Anwendung:** Bildverarbeitung

Neben der diskreten FOURIER-Transformation gibt es außerdem noch die *(exakte, eigentliche) Foruier-Reihe* (vgl. Analysis Vorlesung):
Für eine gegebene Funktion

$$f : [0, L] \to \mathbb{C} \text{ (oder } \mathbb{R}), \quad [\text{oder periodisches } f : \mathbb{R} \to \mathbb{C} \text{ mit Periode } L > 0]$$

berechnet man die (eigentlichen, exakten) FOURIER-*Koeffizienten*

$$c_k := \frac{1}{L} \int_0^L f(x) e^{-2\pi i \frac{kx}{L}} dx, \quad k \in \mathbb{Z} \tag{5.42}$$

bzw.

$$\begin{aligned} \alpha_k &:= \frac{2}{L} \int_0^L f(x) \cos \frac{2\pi kx}{L} dx, \quad k \in \mathbb{N}_0 \\ \beta_k &:= \frac{2}{L} \int_0^L f(x) \sin \frac{2\pi kx}{L} dx, \quad k \in \mathbb{N} \end{aligned} \tag{5.43}$$

Man kann zeigen (vgl. mit (5.36)/(5.40))

$$\begin{aligned} \alpha_k &= c_k + c_{-k}, \\ \beta_k &= i(c_k - c_{-k}), \quad \text{sowie, falls } f \text{ reell ist:} \\ c_{-k} &= \overline{c_k} \quad (\Rightarrow \alpha_k = c_k + c_{-k} = 2\text{Re}(c_k) \in \mathbb{R}, \beta_k = 2\text{Im}(c_k) \in \mathbb{R}. \end{aligned} \tag{5.44}$$

Man betrachtet die sog. FOURIER-Reihe (F_n)

$$F_n(x) := \sum_{k=-n}^{n} c_k e^{2\pi i \frac{kx}{L}} \tag{5.45}$$

bzw.

$$\frac{\alpha_0}{2} + \sum_{k=1}^{n} \alpha_k \cos\frac{2\pi kx}{L} + \beta_k \sin\frac{2\pi kx}{L}. \tag{5.46}$$

Unter gewissen Voraussetzungen an f konvergieren die Reihen (5.45), (5.46) gegen f:

Satz 5.21
Es sei $f : \mathbb{R} \to \mathbb{R}$ eine L-periodische, stückweise stetige Funktion mit stückweise stetiger erster Ableitung mit $f(x) = \frac{f(x+0)+f(x-0)}{2}$. Dann konvergieren die FOURIER*-Reihen (5.45) und (5.46) punktweise gegen f.*

Beweis: Siehe z.B. Heuser, Band 2, Satz 136.2

Bemerkung : Unter sehr schwachen Voraussetzungen ("$f \in L^2(0, L)$") kann man zeigen, dass (F_n) gegen f in der sog. L^2-Norm $\|u\|_{L^2(0,L)} := (\int_0^L |f(x)|^2 dx)^{\frac{1}{2}}$ konvergiert. (Heuser, Kapitel 141, Band 2). [Dabei ist das Intergral das sog. LEBESGUE-Intergral, eine Verallgemeinerung des RIEMANN-Integrals]

Für *glatte* Funktionen fallen die FOURIER-Koeffizienten schnell ab:

Satz 5.22 *Die L-periodische Funktion $f : \mathbb{R} \to \mathbb{C}$ sei p-fach stetig differenzierbar. Dann gilt für die* FOURIER*-Koeffizienten (5.42):*

$$|c_k| \leq \left(\frac{L}{2\pi|k|}\right)^p \max_{\xi\in[0,L]} |f^{(p)}(\xi)| \quad \forall k \in \mathbb{N}$$

Beweis : p-fache partielle Integration von (5.42) liefert

$$c_k = \frac{1}{L} \cdot \left(\frac{L}{2\pi ki}\right)^p \int_0^L f^{(p)}(x) \underbrace{e^{-2\pi i \frac{kx}{L}}}_{|\cdot|=1} dx,$$

wobei sich alle Randterme wegen der Periodizität $f^{(j)}(0) = f^{(j)}(L)\ \forall j = 0, \ldots, p$ wegheben.
$\Rightarrow$ Behauptung $\square$

Zusammenhang zwischen den FOURIER-Koeffizienten und den diskreten FOURIER-Koeffizienten:

Diskretisiert man die Formel (5.42) für c_k bezüglich äquidistanter Stützstellen $x_j = \frac{Lj}{n}$ mittels der Trapez-Regel (s. Beispiel **??**, sowie später Kapitel **??**), so bekommt man

$$c_k \overset{(5.42)}{=} \frac{1}{L}\int_0^L f(x)e^{-2\pi i\frac{kx}{L}} \approx \frac{1}{L}\frac{L}{n}\sum_{j=0}^{n-1} f(x_j)e^{-2\pi i\frac{kLj}{Ln}} \underset{(5.32)}{=} d_k \tag{5.47}$$

d.h. wir erwarten, dass die diskreten FOURIER-Koeffizienten d_k *Approximationen an die* FOURIER-*Koeffizienten* c_k *sind*! Dies ist richtig, sofern f hinreichend glatt und $|k|$ "klein" ist (denn andernfalls kann der Intergrand von (5.42) stark oszillierend sein, was die Genauigkeit der Trapezregel in (5.47) herabsetzt. Man kann die Definition (5.32) der d_k offensichtlich von $k = 0 \ldots, n-1$ auf alle $k \in \mathbb{Z}$ ausdehnen, wobei dann $(d_k)_{k\in\mathbb{Z}}$ offensichtlich n-periodisch ist, wohingegen (c_k) nach Satz 5.22 für $|k| \to \infty$ fällt. Das typische Verhalten der FOURIER-Koeffizienten und der diskreten FOURIER-Koeffizienten sieht daher (für "glattes" f) so aus:

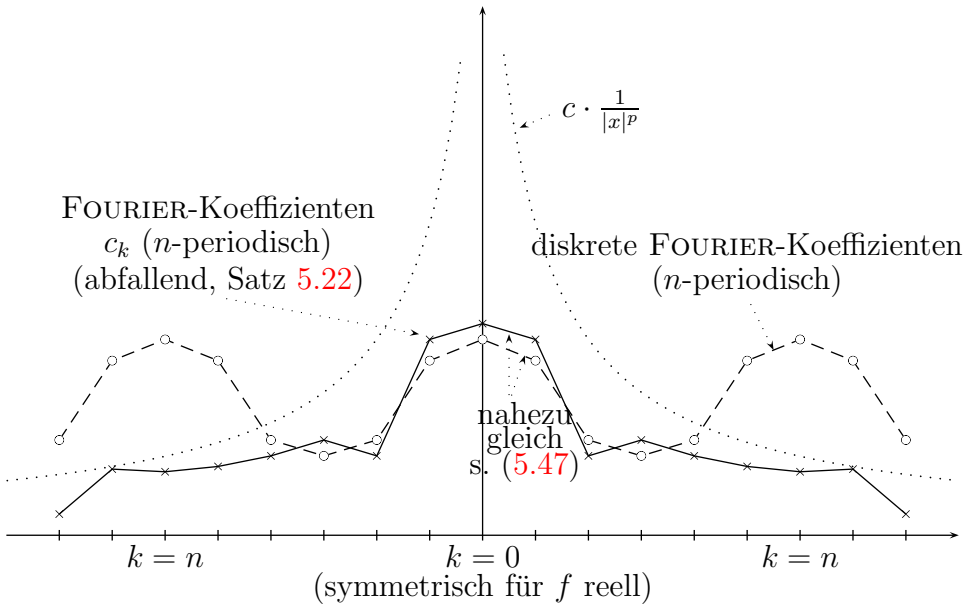


Abbildung 26: Typisches Verhalten der FOURIER-Koeffizienten.

Die Behauptung (5.47), dass $d_k \approx c_k$ für $|k|$ klein und f glatt, lässt sich genauer untersuchen bzw. präzisieren:

Satz 5.23 *(Zusammehang zwischen exakten und diskreten* FOURIER-*Koeffizienten)*
Sei $f : [0, L] \to \mathbb{C}$ *dauert, dass ihre* FOURIER-*Reihe (5.45)/(5.42) punktweise gegen* f *konvergiert, seien die* $x_j = \frac{jk}{N}$ *äqudistant. Dann gilt*

$$d_k = c_k + \underbrace{\sum_{m=1}^{\infty}(c_{k+mn} + c_{k-mn})}_{= \text{"Approximationsfehler"}} = c_k + \lim_{k\to\infty} \underbrace{\sum_{m=-k}^{k} c_{k+mn}}_{\text{abschätzen nach Satz 5.22}}$$

Beweis : Nach Voraussetzung an f gilt

$$\begin{aligned} f(x_j) &= \lim_{k\to\infty} \sum_{\tilde{k}=-k}^{k} c_{\tilde{k}} e^{2\pi \frac{j\tilde{k}}{n}}, \ j = 0, \ldots, n-1, \\ \Rightarrow d_k &= \frac{1}{n}\sum_{j=0}^{n-1} f(x_j) e^{-2\pi i \frac{kj}{n}} = \lim_{k\to\infty} \sum_{\tilde{k}=-k}^{k} c_{\tilde{k}} \frac{1}{n} \underbrace{\sum_{j=0}^{n-1} e^{2\pi i \frac{j(\tilde{k}-k)}{n}}}_{\text{geometrische Reihe}} \end{aligned}$$

Summieren der geometrischen Reihe:

$$\sum_{j=0}^{n-1}\left(e^{2\pi i\frac{(\tilde{k}-k)}{n}}\right)^j = \begin{cases} \sum_{j=0}^{n-1} 1^j = n, & \text{falls } \frac{\tilde{k}-k}{n} \in \mathbb{Z} \\ \frac{e^{2\pi i(\tilde{k}-k)\frac{n}{n}}-1}{e^{2\pi i\frac{(\tilde{k}-k)}{n}}-1} = 0, & \text{sonst.} \end{cases}$$

$\Rightarrow$ Behauptung $\square$

Bemerkung : (Bedeutung von Satz 5.23 und 5.22)
Die Differenz zwischen exakten und diskreten FOURIER-Koeffizienten lässt sich, für $|k| \ll n$, darstellen als Summe von FOURIER-Koeffizienten mit "großem" Index. Solche FOURIER-Koeffizienten mit großem Index nehmen, falls f "glatt" ist, nach Satz 5.22 *kleine* Werte an, in diesem Sinne gilt dann die Näherung (5.47)

Es steht noch aus, eine Abschätzung für den Interpolationsfehler $\Phi(x) - f(x)$ anzugeben. Da die Basisfunktionen

$$\frac{1}{\sqrt{L}}e^{2\pi i\frac{kx}{L}}, k \in \mathbb{Z},$$

(analog: $1, \sqrt{\frac{2}{L}}\cos\frac{2\pi kx}{L}(k \in \mathbb{N}), \sqrt{\frac{2}{L}}\sin\frac{2\pi kx}{L}(k \in \mathbb{N})$) ein Orthonormalsystem im HILBERTraum $L^2(0,L)$ mit Skalarprodukt

$$\langle u, v\rangle := \int_0^L \overline{u(x)}v(x)dx$$

bilden, ist es am Praktischsten, als Norm die L^2-Norm

$$\|u\|_{L^2(0,x)} = \langle u,u\rangle^{\frac{1}{2}} = \left(\int_0^L |u(x)|^2 dx\right)^{\frac{1}{2}}$$

zu verwenden.

Das trigonometrische Interpolationspolynom

$$\Phi \in V_n = span\{e^{2\pi i\frac{kx}{L}}, k = 0,\ldots,n-1\}$$

das bisher betrachtet wurde für das Problem

$$\Phi(x_l) \stackrel{!}{=} f(x_l)\ \forall l = 0,\ldots,n-1,\ x_l = L\frac{l}{n},$$

liefert keine gute Approximationseigenschaft an f. Stattdessen ist es besser,

$$\tilde{\Phi} \in \tilde{V}_n := span\{e^{2\pi i\frac{kx}{L}}, k = -\tilde{n},\ldots,\tilde{n}\}\ n = 2\tilde{n}+1 \text{ ungerade}$$

[lässt sich auch auf gerade n übertragen, dann $k = -\tilde{n}+1,\ldots,\tilde{n};\ n = 2\tilde{n}$] zu betrachten.
Glücklicherweise brauchen die Koeffizienten $\tilde{d}_k$, $k = -\tilde{n},\ldots,\tilde{n}$, von

$$\tilde{\Phi}(x) = \sum_{k=-\tilde{n}}^{\tilde{n}} \tilde{d}_k e^{2\pi i\frac{kx}{L}}$$

nicht neu berechnet zu werden, denn

$$\begin{aligned}\Phi(x_l) &= \sum_{k=0}^{n-1} d_k e^{2\pi i \frac{kl}{n}} \underset{\substack{\text{wie im Beweis}\\ \text{von Satz 5.15}}}{=} \sum_{k=0}^{\tilde{n}} d_k e^{2\pi i \frac{kl}{n}} + \sum_{k=1}^{\tilde{n}} \underbrace{d_{n-k}}_{\substack{=d_{-k} \text{ wegen der}\\ \text{Periodizität der } d_k}} e^{-2\pi i \frac{kl}{n}} \\ &= \sum_{k=-\tilde{n}}^{\tilde{n}} d_k e^{2\pi i \frac{kl}{n}} = \tilde{\Phi}(x_l),\end{aligned}$$

also $\tilde{d_k} = d_k = d_{n+k} \forall k = -\tilde{n}, \ldots, \tilde{n}$, sofern wir

$$d_k := \sum_{l=0}^{n-1} f(x_l) e^{2\pi i \frac{kl}{n}}$$

nicht nur für $k = 0, \ldots, n-1$, sondern für alle $k \in \mathbb{Z}$ definieren.
[Vorsicht : $\Phi(x_l) = f(x_l) = \tilde{\Phi}(x_l)$ $\forall$ Gitterpunkte $l = 0, \ldots, n-1$, aber i.a. $\Phi(x) \neq \tilde{\Phi}(x)$.]

Satz 5.24 *(Fehler der trigonometrischen Interpolation)*

Sei $f : [0, L] \to \mathbb{C}$ eine quadratintegrable Funktion (d.h. $f \in L^2(0, L)$, d.h. $\int_0^L |f(x)|^2 dx < \infty$ existiert), seien $y_j = f(x_j)$ die Auswertungen von f an äqudistanten Stützstellen $x_j = \frac{jx}{n}$, wobei $n = 2\tilde{n} + 1$ ungerade sei. Dann gilt

$$\|\tilde{\Phi} - f\|^2_{L^2(0,L)} \leq L \cdot \sum_{|k| \leq \tilde{n}} |c_k - d_k|^2 + L \sum_{|k| \geq \tilde{n}} |c_k|^2,$$

wobei die c_k die exakten Fourier-*Koeffizienten gemäß (5.42) und die d_k die diskreten* Fourier-*Koeffizienten gemäß (5.32), $k \in \mathbb{Z}$, sind.*

Beweis : Für $f \in L^2(0, L)$ gilt

$$f(x) = \lim_{K \to \infty} \sum_{k=-K}^{K} c_k e^{2\pi i \frac{kx}{L}} \quad \text{im } L^2(0, L) - \text{Sinne}$$

Ferner

$$\begin{aligned}\tilde{\Phi}(x) &= \sum_{k=-\tilde{n}}^{\tilde{n}} d_k e^{2\pi i \frac{kx}{L}} \\ \Rightarrow f(x) - \tilde{\Phi}(x) &= \sum_{k=-\tilde{n}}^{\tilde{n}} (c_k - d_k) \frac{\sqrt{L}}{\sqrt{L}} e^{2\pi i \frac{kx}{L}} + \sum_{|k| > \tilde{n}} c_k e^{2\pi i \frac{kx}{L}} \cdot \frac{\sqrt{L}}{\sqrt{L}}\end{aligned}$$

d.h. die Zahlen $(c_k - d_k) \cdot \sqrt{L}$ für $|k| \leq \tilde{n}$ und $c_k \sqrt{L}$ für $|k| > \tilde{n}$ sind die Fourier-Koeffizienten der Funktion $f - \tilde{\Phi}$ bzgl. der Schauder-*ONB* $e^{2\pi i \frac{kx}{L}}, k \in \mathbb{Z}$

$$\Rightarrow \|f - \tilde{\Phi}\|^2_{L^2(0,L)} = L \sum_{k=\tilde{n}}^{\tilde{n}} |c_k - d_k|^2 + L \sum_{|k| > \tilde{n}} |c_k|^2 \tag{5.48}$$

□

Bemerkung 5.25

(1) Die Werte $|c_k - d_k|$ und $|c_k|$ können mit den Sätzen 5.23, 5.22 weiter abgeschätzt werden.

(2) Nimmt man Φ statt $\tilde{\Phi}$ also Interpolierende, so bekommt man

$$\|f - \Phi\|^2_{L^2(0,L)} = L \sum_{k=0}^{n} \underbrace{|c_k - d_k|^2}_{(**)} + L \sum_{k \in \mathbb{Z} \setminus \{0,\dots,n\}} \underbrace{|c_k|^2}_{(*)}$$

wobei in $(*)$ für $k = -1, -2, \dots$ und in $(**)$ für $k \approx n$ durchaus große Werte stehen können; $(*), (**)$ sind schlecht mit 5.22,5.23 abzuschätzen

(3) Besser als Approximation mit Φ oder $\tilde{\Phi}$ ist die Approximation mit F_n (ist aber keine Interpolierende!) gemäß (5.39) [erfordert allerdings das Wissen um f statt nur $f(x_l)$ und Berechnung von Integralen statt Summen/FFT]. Es ist offensichtlich

$$\|f - F_n\|^2_{L^2(0,L)} = L \sum_{|k| > \tilde{n}} |c_k|^2;$$

Vergleiche mit (5.48).