MATTHIAS GERDTS

EINFÜHRUNG IN DIE NUMERIK

Universität der Bundeswehr München
Wintertrimester 2014

Addresse des Autors:

Matthias Gerdts

Institut für Mathematik und Rechneranwendung
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
85577 Neubiberg

E-Mail: matthias.gerdts@unibw.de
WWW: www.unibw.de/lrt1/gerdts

Vorläufige Version: 9. Dezember 2014

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Was ist Numerische Mathematik?

Die Numerische Mathematik (kurz: Numerik) beschäftigt sich mit der Entwicklung, Analyse und Implementierung von Algorithmen zur Lösung von mathematischen Problemen. Die Problemstellungen sind dabei häufig durch konkrete Anwendungen motiviert.
Hierbei ist es wichtig, zwischen einem

(mathematischen) Problem

(z.B. Lösung eines linearen Gleichungssystems, Approximation einer Funktion, Berechnung von Eigenwerten) und einem

(numerischen) Algorithmus

zur Lösung des mathematischen Problems zu unterscheiden. Ein Algorithmus ist eine Verfahrensvorschrift zur konkreten Lösung eines mathematischen Problems, wobei es verschiedene Algorithmen zur Lösung desselben mathematischen Problems geben kann, etwa die Gauß'sche Eliminationsmethode und die Cramer'sche Regel zur Lösung linearer Gleichungssysteme.
Die meisten Probleme, die in der Numerik behandelt werden, können entweder nicht analytisch gelöst werden (d.h. es gibt keine geschlossene Lösungsdarstellung) und erfordern daher approximative Lösungsverfahren oder sie sind zu rechenintensiv, um Lösungen per Hand ausrechnen zu können (wer möchte schon lineare Gleichungssysteme mit tausenden von Gleichungen per Hand lösen?).

Beispiel 1.1.1

Für die folgenden Probleme kann die Lösung nicht explizit angegeben werden:

- *Die nichtlinearen Gleichungen*

$$x - \cos(x) = 0, \quad \exp(x) + x - 2 = 0, \quad x^5 - 3x^4 - \pi x^3 + x^2 - 43x + \sqrt{2} = 0$$

  können nicht geschlossen gelöst werden.

- *Die Werte der Integrale*

$$\int_1^5 \frac{\exp(x)}{x} dx, \quad \int_1^2 \frac{\sin(x)}{x} dx, \quad \int_0^1 \frac{1}{\sqrt{x^3+1}} dx$$

  können nicht analytisch berechnet werden.

- *Die nichtlineare Diffentialgleichung (Pendelgleichung)*

$$x''(t) = -\frac{3g}{2\ell}\sin(x(t))$$

  für x kann nicht geschlossen gelöst werden.

Es gibt sehr viele solcher Beispiele. Die meisten der in der Praxis auftretenden Beispiele sind nicht geschlossen lösbar.

Das Gebiet der Numerik bekam insbesondere durch die fortschreitende Entwicklung von leistungsfähigen Computern eine immer stärkere Bedeutung, da diese die Lösung immer komplexerer Aufgabenstellungen erlauben, z.B.

- Wettervorhersage mittels numerischer Berechnungsverfahren
- Simulation und Optimierung von technischen Systemen (Crashtests, Strömungssimulation, Baustatik, ...)
- Signalverarbeitung, Approximation von Funktionen, Bildverarbeitung (jpeg/mpeg Kompression)
- ...

Im Rahmen der Vorlesung werden verschiedene mathematische Problemstellungen und Lösungsalgorithmen behandelt. Dabei geht es nicht nur um die

Herleitung und Formulierung von Algorithmen,

sondern insbesondere auch um deren Eigenschaften. Zentrale Begriffe und Fragestellungen in der Numerik sind die folgenden:

- **Stabilität:** Wie reagiert ein Algorithmus auf Rundungsfehler, Störungen oder Parameteränderungen?
- **Konvergenz und Konvergenzgeschwindigkeit:** Konvergiert ein Algorithmus gegen die Lösung eines Problems und wenn ja, wie schnell?
- **Approximationsfehler und Approximationsordnung:** Wie gut approximiert ein Algorithmus die Lösung in Abhängigkeit von Diskretisierungsparametern?

Darüber hinaus besteht ein wesentlicher Teil der Numerischen Mathematik in der konkreten

Implementierung von Algorithmen auf dem Computer.

1.2 Ein einfaches Fahrzeugmodell: Differentialgleichungen, Stabilität und Eigenwerte

Ein stark vereinfachtes Fahrzeugmodell zur Untersuchung des Fahrkomforts eines Kraftfahrzeugs ist durch das Viertel-Fahrzeugmodell gegeben, vgl. Kortüm und Lugner [KL94] und Abbildung 1.1.

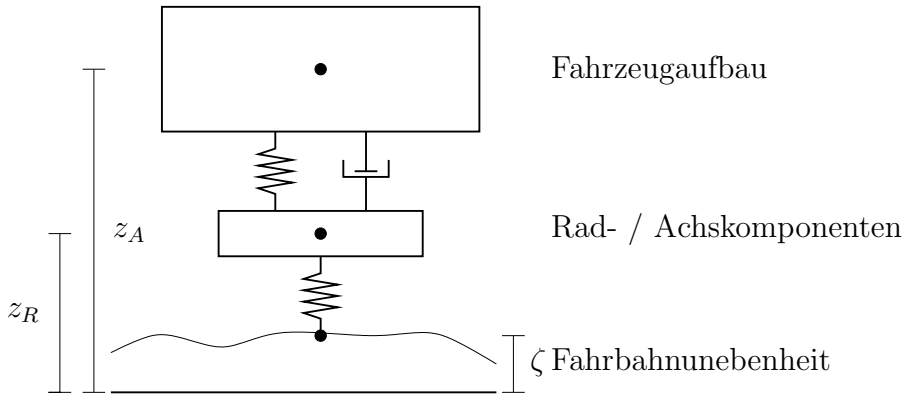


Abbildung 1.1: Viertel-Fahrzeugmodell zur Untersuchung des Fahrkomforts

Die Bewegungsgleichungen der Körperschwerpunkte z_A und z_R sind durch die Differentialgleichungen

$$\begin{aligned} m_A \ddot{z}_A &= -m_A g + F_D + F_F \\ m_R \ddot{z}_R &= -m_R g - F_D - F_F + F_S \end{aligned}$$

gegeben, wobei vereinfachend lineare Kraftgesetze für die Dämpferkraft

$$F_D = c_D(\dot{z}_R - \dot{z}_A),$$

die Federkraft

$$F_F = c_F(z_R - z_A) + F_F^0$$

und die Strassenanregung

$$F_S = c_S(\zeta - z_R) + F_S^0$$

angenommen werden.
Es bezeichnen m_A und m_R die Massen des Fahrzeugaufbaus bzw. der Rad- und Achskomponenten, g die Erdbeschleunigung, c_D und c_F die Dämpfer- bzw. Federkonstante der Radaufhängung und c_S die radiale Karkassensteifigkeit des Reifens, der in idealisierter Form als Federelement modelliert ist. Nachdem durch m_A bzw. m_R geteilt wurde, lässt

sich das Differentialgleichungssystem zweiter Ordnung dann als inhomogenes, lineares Differentialgleichungssystem erster Ordnung für den Zustand

$$x(t) = (z_A(t), z_R(t), v_A(t), v_R(t))^\top \in \mathbb{R}^4$$

in Matrixnotation schreiben:

$$\dot{x}(t) = Ax(t) + h(t)$$

mit

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{c_F}{m_A} & \frac{c_F}{m_A} & -\frac{c_D}{m_A} & \frac{c_D}{m_A} \\ \frac{c_F}{m_R} & -\frac{c_F+c_S}{m_R} & \frac{c_D}{m_R} & -\frac{c_D}{m_R} \end{pmatrix},$$

$$h(t) = \begin{pmatrix} 0 \\ 0 \\ -g + \frac{F_F^0}{m_A} \\ -g - \frac{F_F^0}{m_R} + \frac{F_S^0}{m_R} + \frac{c_S}{m_R}\zeta(t) \end{pmatrix}.$$

Wir wählen konkrete Werte

$$\begin{aligned} m_A &= 300.00 && [kg], \\ m_R &= 60.00 && [kg], \\ g &= 9.81 && [m/s^2], \\ c_F &= 300000.00 && [N/m], \\ c_D &= 30000.00 && [Ns/m], \\ c_S &= 30000.00 && [N/m], \\ F_F^0 &= 64743.00 && [N], \\ F_S^0 &= 14211.60 && [N]. \end{aligned}$$

Man erhält für diesen speziellen Fall

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1000 & 1000 & -100 & 100 \\ 5000 & -5500 & 500 & -500 \end{pmatrix},$$

$$h(t) = \begin{pmatrix} 0 \\ 0 \\ 206 \\ -852 + 500 \cdot \zeta(t) \end{pmatrix}.$$

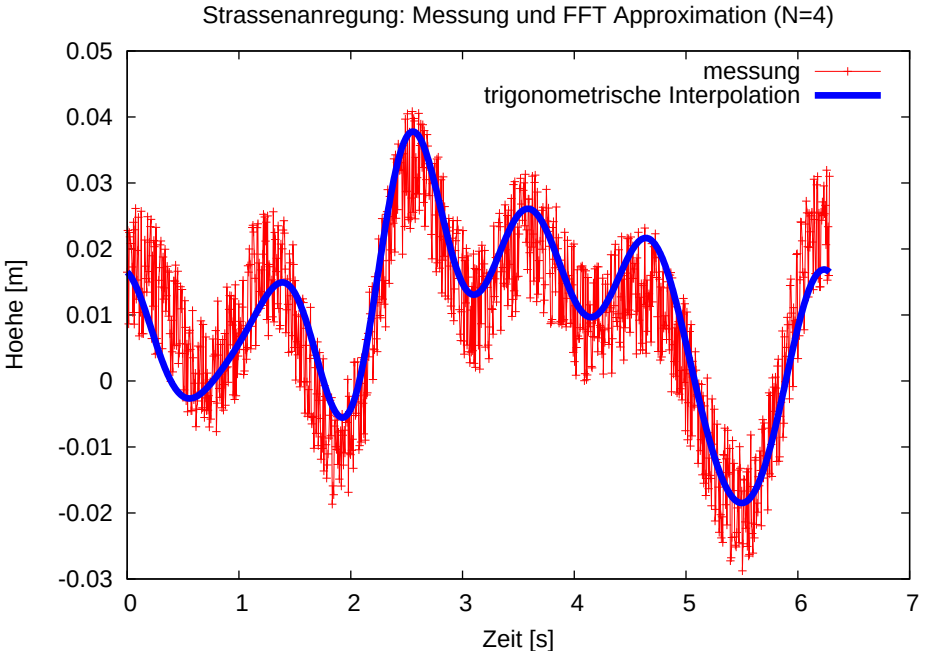
Folgende Fragestellungen können von Interesse sein:

Modellierung der Strassenanregung

Üblicherweise erhält man die Strassenanregung durch Messungen. Diese Messungen sind in der Regel bedingt durch Messungenauigkeiten verrauscht, vgl. Abbildung. Diese Messfehler möchte man in der Simulation möglichst nicht berücksichtigen und herausfiltern. Gesucht ist also ein Modell der Strassenanregung, welches die wesentlichen Schwingungen berücksichtigt. Ein gängiger Ansatz, der auch in der Datenkompression (JPEG, MP3) und der Signal- und Bildverarbeitung eine wichtige Rolle spielt, ist die **Interpolation** durch trigonometrische Polyonome. Hierbei erzeugt man für ein geeignetes m und geeignete Koeffizienten a_j, $j = 0, \dots, m-1$, b_j, $j = 1, \dots, m-1$, eine Funktion

$$p(t) = a_0 + \sum_{j=1}^{m-1} a_j \cos(jt) + b_j \sin(jt),$$

so dass $p(t_k) = \zeta_k$ für Messungen $k = 0, \dots, m-1$ gilt. Offenbar ist p eine Überlagerung von Sinus- und Cosinusfunktionen mit verschiedenen Frequenzen jt und Amplituden a_j, b_j.



Die Abbildung zeigt die Interpolierende p für $m = 8$.

Gleichgewichtslösung

Gesucht ist eine Gleichgewichtslösung bei ebener Fahrbahn mit $\zeta(t) = 0$, d.h. es sind Werte z_A und z_R gesucht, so dass $\ddot{z}_A(t) = \ddot{z}_R(t) = \dot{z}_A(t) = \dot{z}_R(t) = 0$ bzw. $\dot{v}_A(t) = \dot{v}_R(t) = v_A(t) = v_R(t) = 0$ gilt. Diese Fragestellung führt auf das **lineare Gleichungssystem für z_A und z_R**

$$\begin{aligned} 1000 z_A &- 1000 z_R &&= 206, \\ -5000 z_A &+ 5500 z_R &&= -852. \end{aligned}$$

Es hat die Lösung

$$z_A = 0.562, \quad z_R = 0.356.$$

Stabilität

Bei der Konstruktion eines Fahrzeugs ist insbesondere die Auswahl der Federn und Dämpfer (bzw. der entsprechenden Feder- und Dämpferkonstanten) derart vorzunehmen, dass das Fahrzeug ein **stabiles Fahrverhalten** aufweist. Beispielsweise soll das Fahrzeug nach dem Überfahren einer (nicht zu hohen) Bodenwelle nach einer gewissen Zeit wieder in die Gleichgewichtslage zurück gelangen und nicht in einen instabilen Fahrzustand gelangen. Die Begriffe werden genauer spezifiziert. Allgemein heißt eine Lösung $x(t)$ des allgemeinen Differentialgleichungssystems

$$x'(t) = f(t, x(t)), \quad t_0 \le t < \infty,$$

stabil, wenn zu jedem $\varepsilon > 0$ ein $\delta > 0$ existiert, so dass alle Lösungen $y(t)$ mit

$$\|y(t_0) - x(t_0)\| < \delta$$

für alle $t \ge t_0$ existieren und

$$\|y(t) - x(t)\| < \varepsilon$$

für alle $t \ge t_0$ erfüllen. Die Lösung $x(t)$ heißt **asymptotisch stabil**, wenn sie stabil ist und wenn $\delta > 0$ existiert, so dass für alle Lösungen $y(t)$ mit $\|y(t_0) - x(t_0)\| < \delta$ gilt

$$\lim_{t\to\infty} \|y(t) - x(t)\| = 0.$$

Eine Lösung $x(t)$ heißt **instabil**, wenn sie nicht stabil ist.
Es zeigt sich, dass die Eigenwerte der Matrix A eine wesentliche Rolle bei der Beurteilung der Stabilität eines Systems spielen. Für inhomogene, lineare Differentialgleichungssysteme der Form

$$\dot{x}(t) = Ax(t) + h(t, x(t))$$

kann gezeigt werden, vgl. Demailly [Dem91], dass jede Lösung $x(t)$ stabil ist, falls

- für sämtliche Eigenwerte λ_j der Matrix A gilt, dass der Realteil $Re\lambda_j < 0$ ist und
- falls die Inhomogenität h folgende Eigenschaften besitzt:
  - h ist stetig bzgl. beider Argumente,
  - es existiert eine stetige Funktion $k : [t_0, \infty) \to \mathbb{R}_+$ mit $\lim\limits_{t\to\infty} k(t) = 0$ und

    $$\|h(t, x_1) - h(t, x_2)\| \le k(t)\|x_1 - x_2\|$$

    für alle x_1, x_2 und $t_0 \le t < \infty$.

Zur Beurteilung der Stabilität des Fahrzeugs benötigen wir also die Eigenwerte der Matrix A. In unserem Beispiel ergeben sich die vier Eigenwerte

$$\lambda_1 \approx -589.11,\ \lambda_2 \approx -10.58,\ \lambda_3 = \bar{\lambda_4} \approx -0.15 + 8.95i.$$

Der Realteil sämtlicher Eigenwerte ist kleiner als Null, folglich hat unser Viertel-Fahrzeug ein stabiles Fahrverhalten, falls h noch obige Bedingungen erfüllt, vgl. auch Abbildung 1.2.

Simulation des Systems

Die Lösung des Differentialgleichungssystems kann nicht für beliebige Funktionen $\zeta(t)$ explizit angegeben werden. Ist man an der Lösung des Systems interessiert, benötigt man numerische Methoden, um diese zumindest näherungsweise berechnen zu können.
Wir modellieren das Überfahren einer Bodenwelle, indem das Fahrzeug zunächst zum Zeitpunkt $t = 0\ [s]$ aus der Gleichgewichtslage $z_A(0) = 0.562\ [m]$, $z_R(0) = 0.356\ [m]$, $v_A(0) = v_R(0) = 0\ [m/s]$ startend im Zeitintervall $[5, 5 + 2\pi]$ eine Bodenwelle gegeben durch

$$\zeta(t) = \begin{cases} 0.1 \cdot \sin(t-5), & \text{falls } 5 \le t \le 5 + 2\pi, \\ 0, & \text{sonst} \end{cases}$$

überfährt. Abbildung 1.2 zeigt das Ergebnis der Simulation.

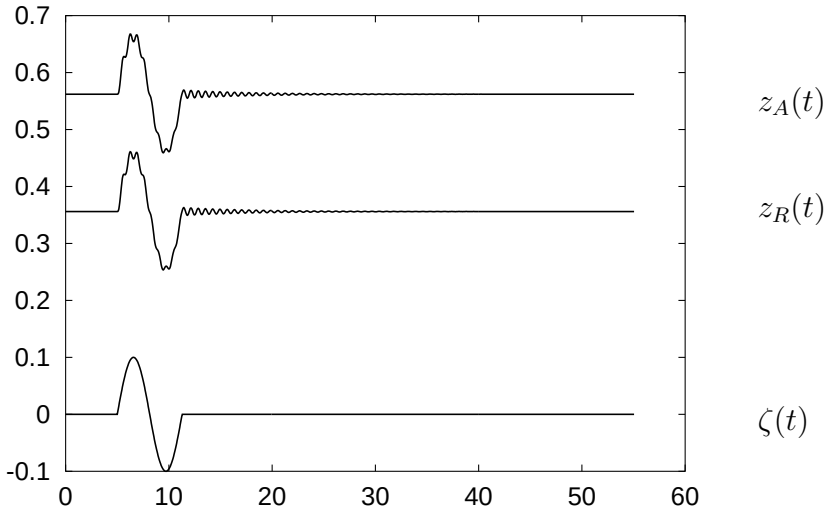


Abbildung 1.2: Simulation des Überfahrens einer Bodenwelle mit dem Viertel-Fahrzeugmodell.

Fazit

Was wurde in diesem Beispiel benötigt?

- Methoden zur Berechnung einer interpolierenden Funktion,

- Methoden zur Lösung von linearen Gleichungssystemen,
- Methoden zur Berechung von Eigenwerten,
- Methoden zur Lösung von Differentialgleichungssystemen.

1.3 Zahldarstellung und Gleitpunktarithmetik

Ein Computer hat nur eine begrenzte Anzahl von Speicherplätzen zur Verfügung. Die Anzahl der auf dem Computer darstellbaren Zahlen ist damit begrenzt auf solche Zahlen, die sich mit endlich vielen Ziffern vor oder nach dem Komma beschreiben lassen. Mathematisch entspricht dies einer Teilmenge der rationalen Zahlen $\mathbb{Q}$. Man nennt diese Zahlen auch **Maschinenzahlen**.
Die reelle Zahl 12345.678 kann geschrieben werden als

$$12345.678 = 1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} + 7 \cdot 10^{-2} + 8 \cdot 10^{-3}.$$

Allgemeiner kann eine Zahl (im Dezimalsystem) geschrieben werden als

$$\underbrace{a_n a_{n-1} \cdots a_1 a_0}_{\text{ganzzahliger Anteil}} \,.\, \underbrace{b_1 b_2 b_3 \cdots}_{\text{Bruchanteil}} = \underbrace{\sum_{k=0}^{n} a_k \cdot 10^k}_{\text{ganzzahliger Anteil}} + \underbrace{\sum_{k=1}^{\infty} b_k \cdot 10^{-k}}_{\text{Bruchanteil}}.$$

Tatsächlich benötigen wir mitunter einen unendlichen String von Ziffern rechts des Dezimalpunkts, um eine beliebige reelle Zahl darstellen zu können, z.B.

$$\begin{aligned} \frac{1}{3} &= 0.3333333\ldots, \\ \pi &= 3.1415926\ldots, \\ e &= 2.7182818\ldots, \\ \sqrt{2} &= 1.4142135\ldots. \end{aligned}$$

Die obigen Zahlen sind im Dezimalsystem zur Basis $B = 10$ dargestellt. Jedes andere $B \in \mathbb{N}$, $B > 1$, ist ebenfalls zulässig und man erhält die Darstellung

$$(a_n a_{n-1} \cdots a_1 a_0 . b_1 b_2 b_3 \cdots)_B = \sum_{k=0}^{n} a_k \cdot B^k + \sum_{k=1}^{\infty} b_k \cdot B^{-k},$$

wobei $a_i, b_j \in \{0, 1, \ldots, B-1\}$.
Im alltäglichen Leben sind wir es gewohnt im **Dezimalsystem** mit $B = 10$ zu rechnen. Bei der internen Darstellung im Computer wird hingegen zumeist eines der folgenden Systeme bevorzugt:

- **Binärsystem ($B = 2$).**
  Die Binärzahl 10010.1001 entspricht der Dezimalzahl 18.5625, denn

  $$(10010.1001)_2 = 1{\cdot}2^4{+}0{\cdot}2^3{+}0{\cdot}2^2{+}1{\cdot}2^1{+}0{\cdot}2^0{+}1{\cdot}2^{-1}{+}0{\cdot}2^{-2}{+}0{\cdot}2^{-3}{+}1{\cdot}2^{-4} = 18.5625.$$

- **Oktalsystem ($B = 8$).**

- **Hexadezimalsystem ($B = 16$).**
  Das Hexadezimalsystem verwendet die „Zahlen“ $a_i, b_j \in \{0, 1, \ldots, 9, A, B, C, D, E, F\}$.

Es kann ebenfalls passieren, dass die Darstellung einer Zahl in einem Zahlensystem endlich ist, also nur endlich viele Ziffern benötigt, während sie unendliche viele Ziffern zur Darstellung in einem anderen Zahlensystem benötigt, z.B.

$$(0.1)_{10} = (0.00011001100110011\ldots)_2.$$

Soll die reelle Zahl x im Computer realisiert werden, muss auf Grund der begrenzten Speicherkapazität die Anzahl der Ziffern zur Darstellung der Mantisse und des Exponenten auf eine gewisse Länge beschränkt werden, etwa in der Form

$$x = m \cdot B^e, \tag{1.1}$$

wobei

$$m = \pm(m_1 \cdot B^{-1} + m_2 \cdot B^{-2} + \ldots + m_\ell \cdot B^{-\ell}) \in \mathbb{R} \tag{1.2}$$

die **Mantisse**, $B \in \mathbb{N}$, $B > 1$ die **Basis** und

$$e = \pm(e_1 \cdot B^{n-1} + e_2 \cdot B^{n-2} + \ldots + e_{n-1} \cdot B + e_n) \in \mathbb{Z} \tag{1.3}$$

den **Exponenten** bezeichnen. Hierbei sind ℓ und n die jeweilige Länge der Mantisse bzw. des Exponenten. Für die Ziffern m_i, $i = 1, \ldots, \ell$, und e_i, $i = 1, \ldots, n$, gilt die Zusatzforderung

$$m_i, e_i \in \{0, 1, 2, \ldots, B-1\}.$$

Damit ist schon eine gewisse Normierung vorgenommen worden. Da $0 \le m_1 \le B - 1$ gefordert wurde, ist $|m|$ wegen

$$|m| = |\sum_{i=1}^{\ell} m_i \cdot B^{-i}| \le \sum_{i=1}^{\ell} \frac{B-1}{B^i} = \sum_{i=1}^{\ell} \left(\frac{1}{B^{i-1}} - \frac{1}{B^i} \right) = 1 - \frac{1}{B^\ell} < 1$$

stets kleiner als 1.

Definition 1.3.1 (Gleitpunktdarstellung)
Die Darstellung einer Zahl x in der Form

$$x = \pm 0.m_1 m_2 \ldots m_\ell \cdot B^{\pm e_1 e_2 \ldots e_n} \tag{1.4}$$

heißt **Gleitpunktdarstellung von** x. *x heißt dann auch* **Gleitpunktzahl**.

Die Gleitpunktdarstellung der Zahl x ist jedoch noch nicht eindeutig, wie das folgende Beispiel zeigt:

$$\begin{aligned} 12345.678 &= 0.12345678 \cdot 10^5 \\ &= 0.012345678 \cdot 10^6 \\ &= 0.0012345678 \cdot 10^7 \\ &\vdots \end{aligned}$$

Der Dezimalpunkt verschiebt sich durch Multiplikation mit entsprechenden Zehnerpotenzen. Um Zahlen auf dem Computer effizient verwalten zu können, benötigt man eindeutige Zahldarstellungen.
Eine **normierte Gleitpunktdarstellung von** x entsteht durch die Zusatzforderung, dass für $x \neq 0$ die erste Ziffer m_1 der Mantisse m ungleich 0 sei. Die normierte Gleitpunktdarstellung von x ist eindeutig.

Definition 1.3.2 **(Normierte Gleitpunktdarstellung)**
Die **normierte Gleitpunktdarstellung der reellen Zahl** $x \neq 0$ *ist definiert als*

$$x = \pm 0.m_1 m_2 m_3 \cdots m_\ell \ldots \cdot B^e = \pm m \cdot B^e$$

wobei $m_1 \neq 0$ gelte. Die Zahl $e = \pm e_1 e_2 \ldots e_n \in \mathbb{Z}$ heißt **Exponent**, *die Zahl m heißt* **normierte Mantisse**.

Die Menge aller normierten Gleitpunktdarstellungen liefert die auf dem Computer darstellbaren Zahlen, die sogenannten Maschinenzahlen.

Definition 1.3.3 **(Maschinenzahlen)**
Die Menge

$$\mathcal{M}_{\ell,n}^B := \{x \in \mathbb{R} \mid x = \pm 0.m_1 m_2 \ldots m_\ell \cdot B^{\pm e_1 e_2 \ldots e_n},\ m_1 \neq 0\} \cup \{0\},$$

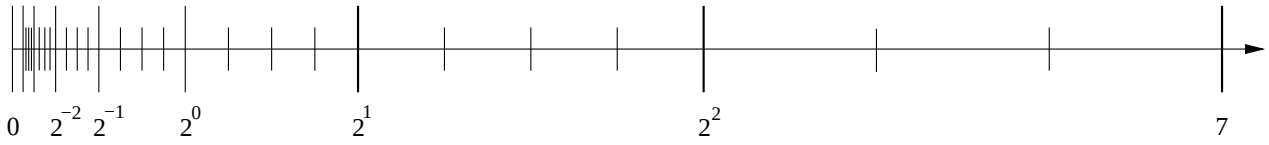
wobei die Länge des Bruchanteils der Mantisse auf $\ell \in \mathbb{N}$ Ziffern und die Länge des Exponenten auf $n \in \mathbb{N}$ Ziffern beschränkt sind, heißt **Menge der Maschinenzahlen zur Basis** B *(kurz: Maschinenzahlen).*

Auf Grund der beschränkten Länge des Exponenten gibt es einen größtmöglichen Exponenten e_{max} und einen kleinstmöglichen Exponenten e_{min} und $e \in [e_{min}, e_{max}]$. Die Zahl $\mu = B^{e_{min}-1}$ ist die kleinste positive Maschinenzahl und $\nu = B^{e_{max}} \cdot (1 - B^{-\ell})$ ist die größte Maschinenzahl.
Die Maschinenzahlen sind nicht gleichmäßig verteilt, wie das folgende Beispiel zeigt.

Beispiel 1.3.4 (Verteilung der Maschinenzahlen)
Betrachte die Maschinenzahlen $\mathcal{M}_{\ell,n}^B$ mit $\ell = 3$, $n = 2$ und Basis $B = 2$. Die folgende Tabelle listet die verfügbaren Mantissen und Exponenten auf:

Mantissen	*Exponenten*
$(0.100)_2 = \frac{1}{2}$	$(00)_2 = 0$
$(0.101)_2 = \frac{5}{8}$	$\pm(01)_2 = \pm 1$
$(0.110)_2 = \frac{3}{4}$	$\pm(10)_2 = \pm 2$
$(0.111)_2 = \frac{7}{8}$	$\pm(11)_2 = \pm 3$

Der größtmögliche Exponent ist $e_{max} = (11)_2 = 2^1 + 2^0 = 3$, der kleinstmögliche Exponent ist $e_{min} = -(11)_2 = -3$. Die kleinste positive Maschinenzahl ist $\mu = 2^{-4} = \frac{1}{16}$, die größte Maschinenzahl ist $\nu = 2^3(1 - 2^{-3}) = 7$. Die Verteilung der positiven Maschinenzahlen sieht wie folgt aus:



1.4 Rundung und Auslöschung

Rundungsfehler treten zwangsläufig auf, da z.B.

$$\begin{aligned} \frac{1}{3} &= 0.3333333\ldots, \\ \pi &= 3.1415926\ldots, \\ e &= 2.7182818\ldots, \\ \sqrt{2} &= 1.4142135\ldots, \end{aligned}$$

keine Maschinenzahlen sind und somit nicht korrekt auf einem Computer dargestellt werden können.
Die Umwandlung einer gegebenen reellen Zahl x in eine Maschinenzahl wird **Rundung** genannt. Ist x bereits eine Maschinenzahl, so soll sie invariant bzgl. Rundung sein. Mathematisch ist die Rundung eine Abbildung

$$rd : \mathbb{R} \to \mathcal{M}_{\ell,n}^B, \quad x \mapsto rd(x), \tag{1.5}$$

die eine reelle Zahl

$$x = \pm 0.m_1 m_2 \ldots m_k m_{k+1} \ldots m_\ell m_{\ell+1} \ldots \cdot B^e$$

gemäß der üblichen Vorschrift

$$rd(x) = \begin{cases} \pm 0.m_1 m_2 \dots m_\ell \cdot B^e, & \text{falls } 0 \le m_{\ell+1} < B/2, \\ \pm 0.m_1 m_2 \dots (m_k+1)0 \dots 0 \cdot B^e, & \text{falls } B/2 \le m_{\ell+1} \le B-1, \\ & m_k \neq B-1, \\ & m_{k+1} = \dots = m_\ell = B-1, \\ \pm 0.10 \dots 0 \cdot B^{e+1}, & \text{falls } B/2 \le m_{\ell+1} \le B-1, \\ & m_1 = \dots = m_\ell = B-1 \end{cases}$$

auf die nächstgelegene Maschinenzahl rundet. Die bei der Rundung auftretenden Fehler können abgeschätzt werden: Es gilt

$$|rd(x) - x| \le \frac{1}{2} \cdot B^{e-\ell}$$

für den absoluten Fehler und, falls $x \neq 0$ ist, gilt

$$\frac{|rd(x) - x|}{|x|} \le \frac{B}{2} \cdot B^{-\ell}$$

für den relativen Fehler. Die Zahl $rd(x)$ lässt sich zudem als

$$rd(x) = x(1+\varepsilon)$$

mit $|\varepsilon| \le \frac{B}{2} \cdot B^{-\ell}$ darstellen. Die Schranke

$$eps = \frac{B}{2} \cdot B^{-\ell}$$

für den relativen Rundungsfehler heißt **Maschinengenauigkeit**. Sie hängt von der Computerarchitektur ab.
Es können zwei Ausnahmesituationen auftreten:

- Ein **Overflow** tritt auf, falls $rd(x) = \pm 0.m_1 \dots m_\ell \dots \cdot B^e$ mit $e > e_{max}$ dargestellt werden soll. In diesem Fall generiert der Computer üblicherweise eine Fehlermeldung und bricht das aktuelle Programm ab.

- Ein **Underflow** tritt auf, falls $rd(x) = \pm 0.m_1 \dots m_\ell \dots \cdot B^e$ mit $e < e_{min}$ dargestellt werden soll. Der Computer behandelt diesen Fall üblicherweise automatisch, indem x auf Null gesetzt wird. War $x \neq 0$, so ist der relative Fehler immer 1, während der absolute Fehler kleiner als μ (kleinste positive Maschinenzahl) und damit sehr klein ist.

Beispiel 1.4.1 (Overflow)
Wir machen einen Test mit MATLAB. Die Befehlszeile

```
X = 1000.0; for i=1:7 X=X*X, end
```

liefert die Ausgabe

```
1000000 1.0000e+12 1.0000e+24 1.0000e+48 1.0000e+96 1.0000e+192 Inf
```

Bemerkung 1.4.2 (Runden durch Abschneiden)
Ein alternativer Weg des Rundens ist das Abschneiden (chopping). Für eine Zahl $x = \pm 0.m_1 m_2 \dots m_\ell d_{\ell+1} \dots \cdot B^e$ *wird* $rd(x)$ *definiert durch die Maschinenzahl* $rd(x) = \pm 0.m_1 m_2 \dots m_\ell \cdot B^e$*, d.h. die Ziffern* $m_{\ell+1} \dots$ *werden einfach vernachlässigt.*
Absoluter Fehler:

$$|rd(x) - x| \leq B^{e-\ell}.$$

Relativer Fehler für $x \neq 0$**:**

$$\frac{|rd(x) - x|}{|x|} \leq B^{-\ell+1}.$$

Beachte, dass diese Fehler doppelt so groß sind wie bei der exakten Rundung.

Für Maschinenzahlen x und y werden die Grundrechenarten $+$, $-$, $\cdot$, $/$ durch Operationen $\oplus$, $\ominus$, $\odot$, $\oslash$ gemäß

$$x \circledast y = rd(x * y), \qquad * \in \{+, -, \cdot, /\}$$

realisiert. Für diese Realisierung gilt

$$x \circledast y = (x * y)(1 + \varepsilon_*), \quad |\varepsilon_*| \leq eps, \quad * \in \{+, -, \cdot, /\}.$$

Insbesondere die Subtraktion zweier Zahlen x und y kann kritisch sein, wenn $x \approx y$. Dies kann wie folgt eingesehen werden:
Der Ansatz

$$z = rd(rd(x) + rd(y)) = (x(1 + \varepsilon_1) + y(1 + \varepsilon_2))(1 + \varepsilon_3)$$

liefert in erster Näherung (d.h. durch Vernachlässigung der Fehlerterme höherer Ordnung in der Taylorentwicklung)

$$\frac{|(x + y) - z|}{|(x + y)|} \doteq \left| \varepsilon_3 + \frac{x\varepsilon_1 + y\varepsilon_2}{(x + y)} \right|.$$

Speziell für $x \approx -y$ ist der relative Fehler im schlimmsten Fall unbeschränkt. Diese Beobachtung ist der Grund für schwerwiegende numerische Probleme, die in Berechnungen auftreten können, da ein Fehler in einem Zwischenergebnis durch weitere Rechenschritte vergrößert werden kann und der Gesamtfehler sich somit aufschaukelt. Dies kann zu völlig unbrauchbaren Ergebnissen führen. Dieses Phänomen heißt **Auslöschung**.

Beispiel 1.4.3

Seien $x = 0.3721448693$ *und* $y = 0.3720214371$ *gegeben. Angenommen, unser Computer besitzt 5 Stellen Genauigkeit. Dann gelten* $rd(x) = 0.37214$ *und* $rd(y) = 0.37202$. *Subtraktion liefert* $z = rd(rd(x) - rd(y)) = rd(0.37214 - 0.37202) = 0.00012$. *Das korrekte Resultat ist* $x - y = 0.0001234322$. *Der relative Fehler beträgt somit*

$$\frac{|(x-y)-z|}{|(x-y)|} = \frac{0.0000034322}{0.0001234322} \approx 3 \cdot 10^{-2}.$$

Dieser Fehler ist sehr groß im Vergleich zu den relativen Fehlern von $rd(x)$ *und* $rd(y)$, *die jeweils durch* $0.5 \cdot 10^{-4}$ *beschränkt sind. Durch Subtraktion zweier in etwa gleich großer Zahlen haben wir also 3 Stellen an Genauigkeit verloren!*

Gelegentlich ist es möglich, den Verlust an Genauigkeit zu vermeiden, indem alternative, analytisch äquivalente Rechenvorschriften verwendet werden.

Beispiel 1.4.4

Betrachte die Funktion $f(x) = \sqrt{x^2+1} - 1$. *Für* $x \approx 0$ *entsteht bei der Subtraktion ein Verlust an Genauigkeit, da dann* $\sqrt{x^2+1} \approx 1$ *gilt. Jedoch kann* f *dargestellt werden als*

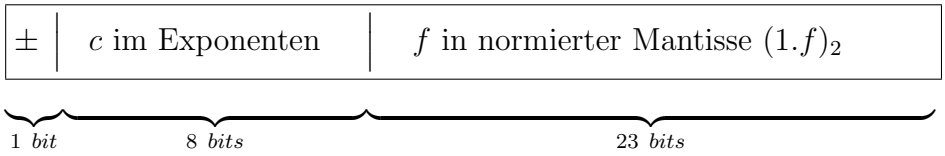
$$f(x) = \left(\sqrt{x^2+1} - 1\right) \cdot \frac{\sqrt{x^2+1}+1}{\sqrt{x^2+1}+1} = \frac{x^2}{\sqrt{x^2+1}+1}.$$

Wenn wir wiederum mit 5 Stellen Genauigkeit rechnen und $x = 10^{-3}$ *wählen, dann wird* $f(x)$ *gemäß der ersten Formel fälschlicherweise zu* 0 *berechnet, während die zweite Formel* $0.5 \cdot 10^{-6}$ *liefert. Der exakte Wert ist* $0.4999998750000625 \cdot 10^{-6}$. *Dies führt auf einen relativen Fehler von* 1 *für die erste Formel bzw. von ungefähr* $0.25 \cdot 10^{-6}$ *für die zweite Formel.*

Der IEEE Standard

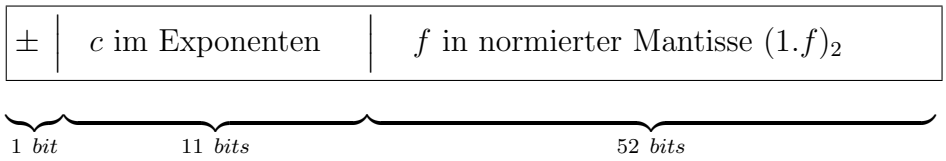
Die Zahldarstellung auf Computern orientiert sich in der Regel am IEEE Standard. Dieser Standard besitzt folgende Merkmale:

- Binärsystem, korrekte Rundung
- größte ganze Zahl: $2^{31} - 1 = 2147483647$
- Gleitpunktdarstellung in einfacher Genauigkeit (single precision) $x = \pm(1.f)_2 \cdot 2^{c-127}$



  Es gilt $0 < c < (11111111)_2 = 255$ und $-127 < c - 127 < 128$. Die Werte $c = 0$ und $c = 255$ sind reserviert für die Darstellung von u.a. ± 0 und $\pm\infty$.

- Gleitpunktdarstellung in doppelter Genauigkeit (double precision) $x = \pm(1.f)_2 \cdot 2^{c-1023}$



  Es gilt $0 < c < (11111111111)_2 = 2047$ und $-1023 < c - 1023 < 1024$. Die Werte $c = 0$ und $c = 2047$ sind reserviert für die Darstellung von u.a. ± 0 und $\pm\infty$.

	Single Precision	Double Precision
Länge Exponent	8 Bits	11 Bits
Länge Mantisse	23 Bits	52 Bits
Maschinengenauigkeit	$2^{-23} \approx 1.192 \cdot 10^{-7}$	$2^{-52} \approx 2.220 \cdot 10^{-16}$
kleinste positive Zahl	$2^{-126} \approx 1.175 \cdot 10^{-38}$	$2^{-1022} \approx 2.225 \cdot 10^{-308}$
größte Zahl	$(2 - 2^{-23})2^{127} \approx 3.403 \cdot 10^{38}$	$(2 - 2^{-52})2^{1023} \approx 1.798 \cdot 10^{308}$

1.5 Algorithmen, Kondition und Stabilität

Es ist wichtig zwischen folgenden Begriffen zu unterscheiden:

- **(Mathematisches) Problem**, das wir lösen möchten (Lösung einer linearen oder nichtlinearen Gleichung, Approximation einer Funktion, ...)

- **(Numerischer) Algorithmus** (oder **Prozedur**), der angewendet werden kann, um ein gegebenes Problem zu lösen (es kann verschiedene Algorithmen zur Lösung desselben Problems geben)

Ein **Problem** heißt

- **gut konditioniert**, falls kleine Störungen in den Problemdaten zu kleinen Änderungen in den Lösungen führen.

- **schlecht konditioniert**, falls kleine Störungen in den Problemdaten zu großen Änderungen in den Lösungen führen können.

Entsprechend heißt ein **(numerischer) Algorithmus**

- **vorwärtsstabil**, falls kleine Fehler in den Rechenschritten des Algorithmus zu kleinen Fehlern im Ergebnis führen.

- **instabil**, falls kleine Fehler in den Rechenschritten des Algorithmus zu großen Fehlern im Ergebnis führen können.

Natürlich ist man daran interessiert, stabile Algorithmen zu entwickeln.
Wir formalisieren die Begriffe.

Definition 1.5.1 **(mathematisches Problem)**
Unter einem **mathematischen Problem** *verstehen wir die Aufgabe, zu einem Eingabedatum x eine Lösung $y = \mathcal{A}(x)$ zu ermitteln. Darin ist $\mathcal{A}$ eine Abbildung*

$$\mathcal{A} : D \subseteq \mathbb{R}^n \to \mathbb{R}^m, \quad x \mapsto y = \mathcal{A}(x).$$

Es stellt sich die Frage, wie sich Fehler Δx im Eingabedatum x auf das Ergebnis y auswirken.

Definition 1.5.2 **(Kondition)**
Wir nennen das mathematische Problem $\mathcal{A}$ **gut konditioniert**, *falls kleine Fehler in x zu kleinen Abweichungen in y führen. Andernfalls heißt es* **schlecht konditioniert**.

Wir wollen den Begriff der Kondition genauer untersuchen und setzen voraus, dass die Abbildung $\mathcal{A}$ zweimal stetig differenzierbar ist. Taylorentwicklung bis zum linearen Glied und Vernachlässigung der Terme höherer Ordnung liefert die Darstellung

$$\|\Delta y\| = \|\mathcal{A}(x+\Delta x) - \mathcal{A}(x)\| \approx \left\| \frac{\partial \mathcal{A}}{\partial x}(x) \cdot \Delta x \right\| \le \left\| \frac{\partial \mathcal{A}}{\partial x}(x) \right\| \cdot \|\Delta x\| \tag{1.6}$$

für passende Normen.

Definition 1.5.3 **(absolute und relative Kondition)**

(a) Die Größe

$$\kappa_a(x) = \left\| \frac{\partial \mathcal{A}}{\partial x}(x) \right\| \tag{1.7}$$

heißt **absolute Kondition** *des Problems $\mathcal{A}$ in x.*

(b) Für $0 \neq y = \mathcal{A}(x)$ heißt die Zahl

$$\kappa_r(x) = \kappa_a(x) \cdot \frac{\|x\|}{\|y\|} = \left\| \frac{\partial \mathcal{A}}{\partial x}(x) \right\| \cdot \frac{\|x\|}{\|y\|}$$

relative Kondition *des Problems $\mathcal{A}$ in x.*

Für $y \neq 0$ hat der relative Fehler häufig eine höhere Aussagekraft als der absolute Fehler. Es gilt

$$\frac{\|\Delta y\|}{\|y\|} \le \kappa_a(x) \cdot \frac{\|\Delta x\|}{\|y\|} = \kappa_a(x) \cdot \frac{\|x\|}{\|y\|} \cdot \frac{\|\Delta x\|}{\|x\|} = \kappa_r(x) \cdot \frac{\|\Delta x\|}{\|x\|}.$$

An Hand der absoluten und relativen Kondition des Problems lässt sich ablesen, wie stark sich eine Störung in x auf den absoluten bzw. relativen Fehler in y auswirkt. Damit lässt sich der Begriff Kondition auch folgendermaßen formulieren, abhängig davon, ob man an dem absoluten oder relativen Fehler interessiert ist:

Das mathematische Problem $\mathcal{A}$ heißt **gut konditioniert**, falls κ_a (bzw. κ_r) klein ist (als Funktion von x).

Man beachte, dass die Kondition lediglich eine Aussage über die **Sensitivität** des mathematischen Problems macht. Rundungs- und Verfahrensfehler, die während der Lösung des Problems auf dem Rechner entstehen, sind nicht berücksichtigt. Um auch diese Fehler zu erfassen, müssen die einzelnen Rechenschritte, die zur Lösung des Problems durchgeführt werden, genauer spezifiziert werden.

Definition 1.5.4 (numerischer Algorithmus)
Eine konkrete Vorschrift, die die Abbildung $\mathcal{A}$ auf dem Rechner realisiert, wird **numerischer Algorithmus** *(kurz: Algorithmus) genannt. Das Endergebnis $y = \mathcal{A}(x)$ des numerischen Algorithmus wird darin über mehrere Rechenschritte r mit Zwischenergebnissen y_i, $i = 1, \ldots, r$, gemäß*

$$\begin{aligned} y_1 &= \mathcal{A}_1(x), \\ y_2 &= \mathcal{A}_2(y_1) \quad = \quad \mathcal{A}_2(\mathcal{A}_1(x)), \\ &\vdots \\ y = y_r &= \mathcal{A}_r(y_{r-1}) \quad = \quad \mathcal{A}_r(\mathcal{A}_{r-1}(\cdots(\mathcal{A}_1(x))\cdots)) \end{aligned}$$

erhalten. Der numerische Algorithmus $\mathcal{A}$ lässt sich damit auch als Verkettung

$$\mathcal{A} = \mathcal{A}_r \circ \mathcal{A}_{r-1} \circ \cdots \circ \mathcal{A}_1.$$

der Rechenschritte schreiben.

Im Allgemeinen kann in jedem Schritt des numerischen Algorithmus ein Rechenfehler bedingt durch Rundung auftreten. Die Zwischenergebnisse und auch das Eingabedatum x sind also verfälscht, etwa gemäß

$$\begin{aligned} \tilde{y}_0 &= x \cdot (1 + \varepsilon_0), \\ \tilde{y}_1 &= \mathcal{A}_1(\tilde{y}_0) \cdot (1 + \varepsilon_1), \\ \tilde{y}_2 &= \mathcal{A}_2(\tilde{y}_1) \cdot (1 + \varepsilon_2), \\ &\vdots \\ \tilde{y} = \tilde{y}_r &= \mathcal{A}_r(\tilde{y}_{r-1}) \cdot (1 + \varepsilon_r), \end{aligned}$$

wobei wir uns hier auf den skalarwertigen Fall beschränken. Die Zahlen ε_i, $i = 0, \ldots, r$, beschreiben die in der Realisierung auftretenden relativen Rundungsfehler und sind komponentenweise beschränkt durch *eps*.

Das verfälschte Ergebnis $\tilde{y}$ kann als Funktion der relativen Rundungsfehler aufgefasst werden, d.h. es ist

$$\tilde{y} = \tilde{y}(x, \varepsilon_0, \varepsilon_1, \dots, \varepsilon_r).$$

Man beachte, dass das exakte Ergebnis y für $\varepsilon_i = 0$, $i = 0, \dots, r$, angenommen wird. Andererseits gelten auch die Interpretationen

$$\tilde{y}_i = \tilde{y}_i(\tilde{y}_{i-1}, \varepsilon_i), \quad i = 1, \dots, r, \quad \tilde{y}_0 = \tilde{y}_0(x, \varepsilon_0).$$

Mit Hilfe der Taylorentwicklung unter Vernachlässigung der Terme höherer Ordnung folgt die Darstellung

$$\begin{aligned}
\tilde{y}(x, \varepsilon_0, \dots, \varepsilon_r) - \tilde{y}(x, 0, \dots, 0) &\approx \sum_{i=0}^{r} \frac{\partial \tilde{y}_r}{\partial \varepsilon_i}(x, 0, \dots, 0) \cdot \varepsilon_i \\
&= \frac{\partial \tilde{y}_r}{\partial \varepsilon_r}(x, 0, \dots, 0) \cdot \epsilon_r \\
&\quad + \sum_{i=1}^{r-1} \left(\frac{\partial \tilde{y}_r}{\partial \tilde{y}_{r-1}} \cdot \frac{\partial \tilde{y}_{r-1}}{\partial \tilde{y}_{r-2}} \cdots \frac{\partial \tilde{y}_{i+1}}{\partial \tilde{y}_i} \cdot \frac{\partial \tilde{y}_i}{\partial \varepsilon_i} \right) (x, 0, \dots, 0) \cdot \varepsilon_i \\
&\quad + \left(\frac{\partial \tilde{y}_r}{\partial \tilde{y}_{r-1}} \cdots \frac{\partial \tilde{y}_1}{\partial \tilde{y}_0} \cdot \frac{\partial \tilde{y}_0}{\partial \varepsilon_0} \right) (x, 0, \dots, 0) \cdot \varepsilon_0 \\
&= \mathcal{A}_r(y_{r-1}) \cdot \varepsilon_r \\
&\quad + \sum_{i=1}^{r-1} \frac{\partial \mathcal{A}_r}{\partial y_{r-1}}(y_{r-1}) \cdots \frac{\partial \mathcal{A}_{i+1}}{\partial y_i}(y_i) \cdot \mathcal{A}_i(y_{i-1}) \cdot \varepsilon_i \\
&\quad + \frac{\partial \mathcal{A}_r}{\partial y_{r-1}}(y_{r-1}) \cdots \frac{\partial \mathcal{A}_1}{\partial y_0}(y_0) \cdot x \cdot \varepsilon_0
\end{aligned}$$

für den absoluten Fehler. Darin bezeichnen y_i, $i = 0, \dots, r$, die exakten Zwischenergebnisse. Definiert man die absoluten Konditionszahlen für die Berechnung des Zwischenergebnisses y_i analog zu (1.7) durch

$$\kappa_a^i(y_{i-1}) = \left\| \frac{\partial \mathcal{A}_i}{\partial y_{i-1}}(y_{i-1}) \right\|,$$

erhält man die Fehlerabschätzung

$$\begin{aligned}
|\tilde{y} - y| &\overset{\cdot}{\leq} |\mathcal{A}_r(y_{r-1}) \cdot \varepsilon_r| \\
&\quad + \sum_{i=1}^{r-1} \kappa_a^r(y_{r-1}) \cdots \kappa_a^{i+1}(y_i) \cdot |\mathcal{A}_i(y_{i-1}) \cdot \varepsilon_i| \\
&\quad + \kappa_a^r(y_{r-1}) \cdots \kappa_a^1(y_0) \cdot |x \cdot \varepsilon_0| \\
&\leq eps \Bigg(|\mathcal{A}_r(y_{r-1}))| \\
&\quad + \sum_{i=1}^{r-1} \kappa_a^r(y_{r-1}) \cdots \kappa_a^{i+1}(y_i) \cdot |\mathcal{A}_i(y_{i-1}))| \\
&\quad + \kappa_a^r(y_{r-1}) \cdots \kappa_a^1(y_0) \cdot |x| \Bigg).
\end{aligned}$$

Damit lässt sich der Begriff Vorwärtsstabilität auch folgendermaßen formulieren:

Der numerische Algorithmus $\mathcal{A}$ heißt **vorwärtsstabil**, falls die Verstärkungsfaktoren $\kappa_a^r(y_{r-1}) \cdots \kappa_a^{i+1}(y_i)$, $i = 0, \ldots, r-1$, klein sind.

1.5.1 Rückwärtsstabilität

Die Untersuchung der (relativen) Konditionszahlen in der Vorwärtsanalyse eines Problems führt häufig zu einer Überschätzung des Fehlers, da in jedem Schritt der schlechtestmögliche Fall betrachtet wird.

Neben der Vorwärtsstabilität, die eine Aussage über die Sensitivität eines Algorithmus (oder Problems) macht, gibt es noch den Begriff der Rückwärtsstabilität. Hierbei werden Störungen im Ergebnis des Algorithmus (oder Problems) betrachtet und dazu passende Argumente betrachtet. Genauer gilt folgende Definition.

Definition 1.5.5 **(Rückwärtsstabilität)**

Gegeben sei der numerische Algorithmus $y = \mathcal{A}(x)$. Der Algorithmus heißt **rückwärtsstabil in** *y_*, wenn es (von y_* abhängige) Konstanten $R > 0$ und $S > 0$ gibt, so dass die folgende Beziehung gilt:*

$$\forall x^1, x^2 \ : \ \|\mathcal{A}(x^i) - y_*\| \leq R, \ i = 1, 2 \quad \Rightarrow \quad \|x^1 - x^2\| \leq S \|\mathcal{A}(x^1) - \mathcal{A}(x^2)\|.$$

Beispiel 1.5.6

Die Funktionsauswertung

$$\mathcal{A}(x) = \begin{cases} (x+1)^2, & x < -1, \\ 0, & -1 \leq x \leq 1, \\ (x-1)^2, & x > 1 \end{cases}$$

ist vorwärtsstabil in $x = 0$ mit der absolution Konditionszahl

$$\kappa_a = |\mathcal{A}'(0)| = 0.$$

Sie ist nicht rückwärtsstabil in $y_* = 0$, da in der Nähe des Funktionswerts $0 = \mathcal{A}(0)$ für kleine Differenzen in den Funktionswerten nicht auf den Abstand der Argumente geschlossen werden kann. Dies sieht man wie folgt. Sei $R > 0$ beliebig. Wähle $x^1 = -1-\sqrt{R} < -1$ und $x^2 = 1 + \sqrt{R} > 1$. Dann gilt $|\mathcal{A}(x^i) - 0| = R$ für $i = 1, 2$, aber $|x^1 - x^2| \geq 2$.

Kapitel 2

Numerische Lösung linearer Gleichungsysteme

Das Folgende gilt sowohl für reellwertige als auch für komplexwertige Vektoren und Matrizen. Daher verwenden wir das Symbol $\mathbb{K}$, welches für $\mathbb{R}$ oder $\mathbb{C}$ steht.
Gesucht ist die Lösung $x = (x_1, \ldots, x_n)^\top \in \mathbb{K}^n$ des linearen Gleichungssystems

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n, \end{aligned} \tag{2.1}$$

wobei a_{ij}, $i, j = 1, \ldots, n$, und b_i, $i = 1, \ldots, n$, gegebene Zahlen sind. Mit

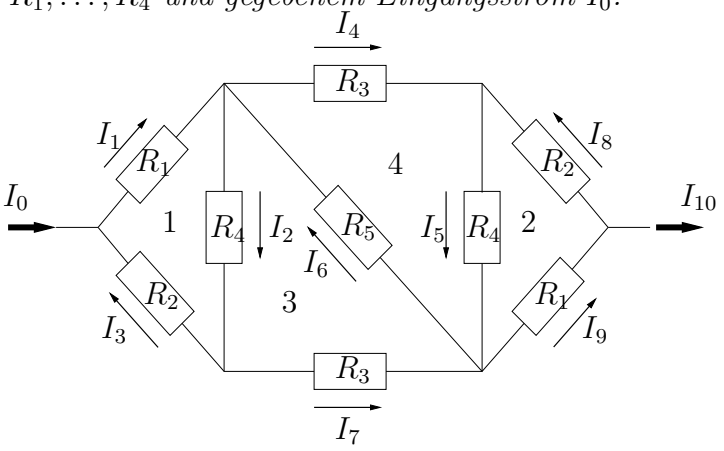
$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \; b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{2.2}$$

lautet das lineare Gleichungssystem in Matrixschreibweise

$$Ax = b \qquad (A \in \mathbb{K}^{n\times n},\; x, b \in \mathbb{K}^n). \tag{2.3}$$

Lineare Gleichungssysteme treten in nahezu allen Anwendungen und Verfahren auf.

Beispiel 2.0.1 (Elektrische Schaltkreise, Kirchhoffsches Gesetz und lineare Gleichungssysteme)
Gegeben sei das folgende stationäre elektronische Netzwerk mit gegebenen ohmschen Widerständen $R_1, \ldots, R_4$ und gegebenem Eingangsstrom I_0.



Das erste Kirchhoffsche Gesetz (Knotensatz)

An jedem Knotenpunkt ist die Summe aller zu- (positiven) und abfließenden (negativen) Ströme unter Beachtung der durch die Pfeile angegebenen Richtungen in jedem Zeitpunkt gleich Null

liefert die Beziehungen

$$\begin{aligned} -I_1 + I_3 &= -I_0, \\ I_1 - I_2 - I_4 + I_6 &= 0, \\ I_2 - I_3 - I_7 &= 0, \\ I_4 - I_5 + I_8 &= 0, \\ I_5 - I_6 + I_7 - I_9 &= 0, \\ -I_8 + I_9 - I_{10} &= 0. \end{aligned}$$

Aus dem Ohmschen Gesetz

$$U = R \cdot I$$

und dem zweiten Kirchhoffschen Gesetz (Maschensatz)

In einer Masche ist die Summe aller Teilspannungen in jedem Zeitpunkt gleich Null

folgen die Gleichungen

$$\begin{aligned} R_1 \cdot I_1 + R_4 \cdot I_2 + R_2 \cdot I_3 &= 0, \\ R_2 \cdot I_8 + R_4 \cdot I_5 + R_1 \cdot I_9 &= 0, \\ R_4 \cdot I_2 + R_5 \cdot I_6 + R_3 \cdot I_7 &= 0, \\ R_3 \cdot I_4 + R_4 \cdot I_5 + R_5 \cdot I_6 &= 0. \end{aligned}$$

Diese 10 Gleichungen für die Ströme $I_1, \dots, I_{10}$ können als **lineares Gleichungssystem** *geschrieben werden:*

$$\begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ R_1 & R_4 & R_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_4 & 0 & 0 & R_2 & R_1 & 0 \\ 0 & R_4 & 0 & 0 & 0 & R_5 & R_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_3 & R_4 & R_5 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \\ I_8 \\ I_9 \\ I_{10} \end{pmatrix} = \begin{pmatrix} -I_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Unser Ziel ist es, Verfahren zur Bestimmung einer Lösung x von (2.3) zu entwickeln und zu analysieren.

2.1 Grundlagen und Lösbarkeit

Im folgenden werden wir verschiedene Typen von Matrizen und einige Eigenschaften verwenden. Sei $A \in \mathbb{K}^{n\times n}$ eine Matrix. Sie heißt

- **Diagonalmatrix**, falls $a_{ij} = 0$ für alle $i \neq j$ gilt, d.h.

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

- **Tridiagonalmatrix**, falls $a_{ij} = 0$ für alle $i, j \in \{1, \dots, n\}$ mit $|i - j| > 1$ gilt, d.h.

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

- **untere Dreiecksmatrix**, falls $a_{ij} = 0$ für alle $i < j$ gilt, d.h.

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

- **obere Dreiecksmatrix**, falls $a_{ij} = 0$ für alle $i > j$ gilt, d.h.

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

- **normierte untere (obere) Dreiecksmatrix**, falls A eine untere (obere) Dreiecksmatrix ist, deren Diagonaleinträge alle gleich Eins sind.

Desweiteren heißt $A \in K^{n\times n}$

- **hermitesch**, wenn $A^* = A$ mit $A^* = \bar{A}^\top$ gilt, wobei $A^\top$ die transponierte Matrix von A bezeichnet und $\bar{A}$ die konjugiert komplexe Matrix von A ist.
- **symmetrisch**, wenn $\mathbb{K} = \mathbb{R}$ und $A^\top = A$ gilt.
- **unitär**, falls $A^*A = I$ gilt.
- **orthogonal**, falls $\mathbb{K} = \mathbb{R}$ und $A^\top A = I$ gilt.
- **positiv semidefinit**, falls $x^*Ax \geq 0$ für alle $x \in \mathbb{K}^n$ gilt.
- **positiv definit**, falls $x^*Ax > 0$ für alle $x \in \mathbb{K}^n$, $x \neq 0$, gilt.

Definition 2.1.1

Sei $A \in \mathbb{K}^{m\times n}$ eine Matrix. Dann heißen

$$\begin{aligned} \mathit{Kern}(A) &:= \{x \in \mathbb{K}^n \mid Ax = 0\} \qquad \mathit{Kern\ von\ } A, \\ \mathit{Bild}(A) &:= \{y \in \mathbb{K}^m \mid \exists x \in \mathbb{K}^n \ :\ y = Ax\} \qquad \mathit{Bild\ von\ } A, \\ \mathit{Rang}(A) &:= \mathit{dim}(\mathit{Bild}(A)) \qquad \mathit{Rang\ von\ } A. \end{aligned}$$

Kern(A) ist ein Unterraum des $\mathbb{K}^n$ und Bild(A) ist ein Unterraum von $\mathbb{K}^m$. Rang(A) ist die maximale Anzahl von linear unabhängigen Spalten von A, was gleich der Anzahl der linear unabhängigen Zeilen von A ist.
Es stellt sich die Frage, welche Voraussetzungen an A und b gestellt werden müssen, damit das lineare Gleichungssystem (2.3) überhaupt eine Lösung x besitzt. Zunächst ist klar, dass das lineare Gleichungssystem **genau eine Lösung** besitzt, falls die Matrix A invertierbar (regulär) ist. Die eindeutige Lösung des Gleichungssystems ist dann durch

$$x = A^{-1}b \tag{2.4}$$

gegeben. Es kann aber auch der Fall eintreten, dass (2.3) mehrere Lösungen oder gar keine Lösung besitzt. Offensichtlich kann die Matrix A dann nicht regulär sein.

Beispiel 2.1.2

Das folgende lineare Gleichungssystem besitzt keine Lösung:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Das folgende lineare Gleichungssystem besitzt genau eine Lösung:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Das folgende lineare Gleichungssystem besitzt unendlich viele Lösungen:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Es gilt das folgende Kriterium, welches die Frage der Lösbarkeit von (2.3) abschließend beantwortet.

Satz 2.1.3

Das lineare Gleichungssystem (2.3) besitzt genau dann **mindestens eine Lösung**, *falls gilt:*

$$Rang(A) = Rang(A|b).$$

Unter $(A|b)$ wird die um die rechte Seite b erweiterte Matrix

$$(A|b) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right)$$

verstanden.

Beweis: Offenbar besitzt das lineare Gleichungssystem genau dann eine Lösung, wenn $b \in \text{Bild}(A)$ gilt.
Sei nun $b \in \text{Bild}(A)$. Dann gilt auch $\text{Bild}(A) = \text{Bild}(A|b)$ und somit $\text{Rang}(A) = \text{Rang}(A|b)$. Gilt nun $\text{Rang}(A) = \text{Rang}(A|b)$, so lässt sich b als Linearkombination der Spalten von A schreiben. Also gilt $b \in \text{Bild}(A)$ und somit besitzt das Gleichungssystem eine Lösung. □

Die Struktur der Lösungsmenge eines linearen Gleichungssystems wird im folgenden Satz behandelt.

Satz 2.1.4

Seien $A \in \mathbb{K}^{n\times n}$ und $b \in \mathbb{K}^n$ gegeben. Sei $\hat{x}$ eine Lösung von $Ax = b$. Die gesamte Lösungsmenge von $Ax = b$ ist gegeben durch $\hat{x} + Kern(A)$.

Beweis: Sei $y \in \text{Kern}(A)$, d.h. $Ay = 0$. Dann gilt $A(\hat{x} + y) = A\hat{x} + Ay = b$ und $\hat{x} + y$ ist Lösung.
Sei x Lösung. Dann gilt $A(x - \hat{x}) = b - b = 0$ und somit $x - \hat{x} \in \text{Kern}(A)$ bzw. $x \in \hat{x} + \text{Kern}(A)$. □

Wir fassen für $A \in \mathbb{K}^{m\times n}$ noch einige aus der linearen Algebra bekannte Resultate zusammen:

- Für $A \in \mathbb{K}^{m\times n}$ gelten

$$\begin{aligned} n &= \dim(\mathrm{Kern}(A)) + \dim(\mathrm{Bild}(A)) \\ m &= \dim(\mathrm{Kern}(A^\top)) + \mathrm{Rang}(A) \end{aligned}$$

- Sind $A \in \mathbb{K}^{n\times n}$ und $B \in \mathbb{K}^{n\times n}$ invertierbar, so gelten

$$(A \cdot B)^{-1} = B^{-1} \cdot A^{-1} \quad \text{und} \quad (A^*)^{-1} = (A^{-1})^*.$$

- Sei $A \in \mathbb{K}^{n\times n}$. Die folgenden Aussagen sind äquivalent:
  - A ist regulär (invertierbar)
  - $\mathrm{Kern}(A) = \{0\}$
  - $\mathrm{Bild}(A) = \mathbb{K}^n$
  - $\mathrm{Rang}(A) = n$
  - $Ax = 0 \Leftrightarrow x = 0$
  - $Ax = b$ hat für jedes b genau eine Lösung
  - $\det(A) \neq 0$
  - alle Eigenwerte von A sind von Null verschieden

- Für $A \in \mathbb{K}^{m\times n}$ gelten die folgenden Beziehungen
  - $\mathrm{Kern}(A) = \mathrm{Bild}(A^*)^\perp$ und $\mathrm{Bild}(A^*) = \mathrm{Kern}(A)^\perp$
  - $\mathrm{Kern}(A) = \mathrm{Kern}(A^*A)$
  - $\mathrm{Bild}(A^*) = \mathrm{Bild}(A^*A)$

  Hierin bezeichnet $V^\perp = \{w \in \mathbb{K}^n \mid w^*v = 0 \ \forall v \in V\}$ das orthogonale Komplement des Unterraums $V \in \mathbb{K}^n$.

Um das lineare Gleichungssystem numerisch lösen zu können, gehen wir im folgenden davon aus, dass die Matrix A regulär ist und somit genau eine Lösung existiert. Die numerische Berechnung einer nicht eindeutigen Lösung im Falle einer singulären Matrix A ist komplizierter und wird hier nicht betrachtet.

2.2 Gauß'sches Eliminationsverfahren und LR-Zerlegung

Es werden Algorithmen zur numerischen Lösung des linearen Gleichungssystems (2.3) entwickelt. Es wird vorausgesetzt, dass die Matrix A regulär ist. Formal kennen wir die Lösung dann schon, sie ist durch (2.4) gegeben. Dort wird aber die Inverse von A benötigt, die i.a. nicht bekannt ist und für große Matrizen auch nicht in einfacher Weise berechnet werden kann.

2.2.1 Einfach zu lösende Gleichungssysteme

Zunächst widmen wir uns linearen Gleichungssystemen, die „einfach“ lösbar sind.
Wir betrachten insbesondere lineare Gleichungssysteme, in denen die Matrix A Dreiecksgestalt hat. Diese Gleichungssysteme werden später im Gauß'schen Eliminationsverfahren bzw. der LR-Zerlegung eine wichtige Rolle spielen.
Zunächst betrachten wir das lineare Gleichungssystem

$$Ly = b \tag{2.5}$$

für $y \in \mathbb{K}^n$ mit einem Vektor $b \in \mathbb{K}^n$ und einer linken unteren normierten Dreiecksmatrix

$$L = \begin{pmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{n,n-1} & 1 \end{pmatrix} \in \mathbb{K}^{n\times n}.$$

In Komponenten lautet es

$$\begin{aligned} y_1 &= b_1, \\ \ell_{21}y_1 + y_2 &= b_2, \\ \ell_{31}y_1 + \ell_{32}y_2 + y_3 &= b_3, \\ &\vdots \\ \ell_{n1}y_1 + \ell_{n2}y_2 + \ldots + \ell_{n-1,n}y_{n-1} + y_n &= b_n. \end{aligned}$$

Dieses Gleichungssystem können wir direkt lösen, da die Gleichungen von oben beginnend sukzessive nach den Variablen y_i, $i = 1, \ldots, n$, aufgelöst werden können:

$$\begin{aligned} y_1 &= b_1, \\ y_2 &= b_2 - \ell_{21}y_1, \\ y_3 &= b_3 - \ell_{31}y_1 - \ell_{32}y_2, \\ &\vdots \\ y_n &= b_n - \ell_{n1}y_1 - \ell_{n2}y_2 - \ldots - \ell_{n-1,n}y_{n-1} \end{aligned}$$

bzw.

$$y_i = b_i - \sum_{j=1}^{i-1} \ell_{ij}y_j, \quad i = 1, \ldots, n. \tag{2.6}$$

Bei der Berechnung von y_i werden nur die bereits bekannten Werte y_j, $j = 1, \ldots, i-1$, verwendet. Diesen Vorgang nennt man **Vorwärtssubstitution**.

Algorithmus 2.2.1 (Vorwärtssubstitution)

Input: untere Dreiecksmatrix L, rechte Seite b
Output: Lösung y von Ly = b.

```
y(1) = b(1);
for i = 2:n,
   y(i) = b(i);
   for j = 1:(i-1),
      y(i) = y(i) - L(i,j) * y(j);
   end
end
```

Der Aufwand der Vorwärtssubstitution beträgt offenbar $\mathcal{O}(n^2)$.
Entsprechend kann man auch bei der Lösung des linearen Gleichungssystems

$$Rx = c \tag{2.7}$$

für x mit rechter oberer Dreiecksmatrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ & r_{22} & r_{23} & \cdots & r_{2n} \\ & & r_{33} & \ddots & \vdots \\ & & & \ddots & r_{n-1,n} \\ & & & & r_{nn} \end{pmatrix}, \qquad r_{ii} \neq 0,\ i = 1, \ldots, n,$$

vorgehen. In diesem Fall kann man die Gleichungen

$$\begin{aligned} r_{11}x_1 + r_{12}x_2 + \ldots + r_{1n}x_n &= c_1, \\ r_{22}x_2 + \ldots + r_{2n}x_n &= c_2, \\ &\vdots \\ r_{n-1,n-1}x_{n-1} + r_{n-1,n}x_n &= c_{n-1}, \\ r_{nn}x_n &= c_n \end{aligned}$$

beginnend mit der letzten Zeile nach x_i, $i = n, n-1, \ldots, 1$, auflösen:

$$\begin{aligned} x_1 &= \tfrac{1}{r_{11}} \left(c_1 - r_{12}x_2 - r_{13}x_3 - \ldots - r_{1n}x_n \right), \\ x_2 &= \tfrac{1}{r_{22}} \left(c_2 - r_{23}x_3 - \ldots - r_{2n}x_n \right), \\ &\vdots \\ x_{n-1} &= \tfrac{1}{r_{n-1,n-1}} \left(c_{n-1} - r_{n-1,n}x_n \right), \\ x_n &= \tfrac{c_n}{r_{nn}} \end{aligned}$$

bzw.

$$x_i = \frac{1}{r_{ii}} \left(c_i - \sum_{j=i+1}^{n} r_{ij}x_j \right), \quad i = n, n-1, \ldots, 1.$$

Hier werden die Gleichungen „rückwärts“ aufgelöst, daher heißt der Vorgang auch **Rückwärtssubstitution**. Zu beachten ist, dass die Diagonalelemente von R nicht 0 sein dürfen, da in diesem Fall bei der Rückwärtssubstitution durch 0 geteilt werden würde.

Algorithmus 2.2.2 (Rückwärtssubstitution)

Input: obere Dreiecksmatrix R, rechte Seite c
Output: Lösung x von Rx = c.

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

Der Aufwand der Rückwärtssubstitution beträgt wiederum $\mathcal{O}(n^2)$.

2.2.2 Das Gauß'sche Eliminationsverfahren

Wir haben im vorhergehenden Abschnitt gesehen, dass lineare Gleichungssysteme mit Dreiecksstruktur direkt gelöst werden können. Die Idee des Gauß'schen Eliminationsverfahrens besteht darin, die Matrix A durch elementare Zeilenumformungen schrittweise in eine rechte obere Dreiecksmatrix R und die rechte Seite b in einen Vektor c zu überführen, so dass ein Gleichungssystem der Form (2.7) entsteht. Dieses kann dann mittels Rückwärtssubstitution gelöst werden. Wichtig ist hierbei, dass die Lösung des Ausgangsproblems mit der des transformierten Problems übereinstimmt. Dies ist bei der Verwendung elementarer Zeilenumformungen gewährleistet. Elementare Zeilenumformungen sind

- die Multiplikation einer Zeile mit einem Wert ungleich Null,
- die Addition bzw. Substraktion zweier Zeilen,
- sowie das Vertauschen zweier Zeilen.

Schematisch läuft der Gauß'sche Eliminationsalgorithmus wie folgt ab.

Gauß'scher Eliminationsalgorithmus:

(i) Setze $A^{(1)} := A$ und $b^{(1)} := b$.

(ii) Beginnend mit $A^{(1)}x = b^{(1)}$ führe $n-1$ Transformationsschritte durch bis ein äquivalentes lineares Gleichungssystem mit oberer Dreiecksstruktur erreicht ist:

$$A^{(1)}x = b^{(1)} \quad \Leftrightarrow \quad A^{(2)}x = b^{(2)} \quad \Leftrightarrow \ldots \Leftrightarrow \quad A^{(n)}x = b^{(n)}$$

(iii) Löse das lineare Gleichungssystem $A^{(n)}x = b^{(n)}$ durch Rückwärtssubstitution.

Beispiel 2.2.3

$$\underbrace{\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix}}_{=A^{(1)}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \underbrace{\begin{pmatrix} 12 \\ 34 \\ 27 \\ -38 \end{pmatrix}}_{=b^{(1)}}$$

Gauß'sche Elimination liefert folgendes Ergebnis:
Start:

$$\left[\begin{array}{c|c} A^{(1)} & b^{(1)} \end{array}\right] = \left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 12 \\ 12 & -8 & 6 & 10 & 34 \\ 3 & -13 & 9 & 3 & 27 \\ -6 & 4 & 1 & -18 & -38 \end{array}\right]$$

Subtraktion des 2*-fachen der 1. Zeile von der 2. Zeile und des* 0.5*-fachen der 1. Zeile von der 3. Zeile und des* -1*-fachen der 1. Zeile von der 4. Zeile liefert:*

$$\left[\begin{array}{c|c} A^{(2)} & b^{(2)} \end{array}\right] = \left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 12 \\ 0 & -4 & 2 & 2 & 10 \\ 0 & -12 & 8 & 1 & 21 \\ 0 & 2 & 3 & -14 & -26 \end{array}\right]$$

Subtraktion des 3*-fachen der 2. Zeile von der 3. Zeile und des* -0.5*-fachen der 2. Zeile von der 4. Zeile liefert:*

$$\left[\begin{array}{c|c} A^{(3)} & b^{(3)} \end{array}\right] = \left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 12 \\ 0 & -4 & 2 & 2 & 10 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 4 & -13 & -21 \end{array}\right]$$

Subtraktion des 2-fachen der 3. Zeile von der 4. Zeile liefert:

$$\left[\begin{array}{c|c} A^{(4)} & b^{(4)} \end{array}\right] = \left[\begin{array}{cccc|c} 6 & -2 & 2 & 4 & 12 \\ 0 & -4 & 2 & 2 & 10 \\ 0 & 0 & 2 & -5 & -9 \\ 0 & 0 & 0 & -3 & -3 \end{array}\right]$$

Rückwärtssubstitution:

$$x = (1, -3, -2, 1)^\top.$$

Im folgenden wird der Schritt von $i \mapsto i+1$ im Detail beschrieben. Der Algorithmus sei bis zum i-ten Schritt fortgeschritten mit

$$A^{(i)} = \begin{pmatrix} a_{11}^{(i)} & \cdots & a_{1i}^{(i)} & \cdots & a_{1n}^{(i)} \\ & \ddots & \vdots & \ddots & \vdots \\ & & a_{ii}^{(i)} & \cdots & a_{in}^{(i)} \\ & & a_{i+1,i}^{(i)} & \cdots & a_{i+1,n}^{(i)} \\ & & \vdots & \ddots & \vdots \\ & & a_{ni}^{(i)} & \cdots & a_{nn}^{(i)} \end{pmatrix}, \qquad b^{(i)} = \begin{pmatrix} b_1^{(i)} \\ \vdots \\ b_i^{(i)} \\ b_{i+1}^{(i)} \\ \vdots \\ b_n^{(i)} \end{pmatrix}.$$

Ziel ist es, die Elemente $a_{ji}^{(i)}$, $j = i+1, \ldots, n$, durch elementare Zeilenumformungen zu eliminieren. Es gelte $a_{ii}^{(i)} \neq 0$ für das sogenannte Pivotelement. Die Matrix $A^{(i+1)}$ erhält

man, indem das $\frac{a_{ji}^{(i)}}{a_{ii}^{(i)}}$-fache der i-ten Zeile von den Zeilen $j = i+1, \dots, n$ subtrahiert wird:

$$A^{(i+1)} = \begin{pmatrix} a_{11}^{(i+1)} & \cdots & a_{1,i+1}^{(i+1)} & \cdots & a_{1n}^{(i+1)} \\ & \ddots & \vdots & \ddots & \vdots \\ & & a_{i+1,i+1}^{(i+1)} & \cdots & a_{i+1,n}^{(i+1)} \\ & & a_{i+2,i+1}^{(i+1)} & \cdots & a_{i+2,n}^{(i+1)} \\ & & \vdots & \ddots & \vdots \\ & & a_{n,i+1}^{(i+1)} & \cdots & a_{nn}^{(i+1)} \end{pmatrix}$$

$$:= \begin{pmatrix} a_{11}^{(i)} & \cdots & a_{1i}^{(i)} & \cdots & a_{1n}^{(i)} \\ & \ddots & \vdots & \ddots & \vdots \\ & & a_{ii}^{(i)} & \cdots & a_{in}^{(i)} \\ & & a_{i+1,i}^{(i)} - \frac{a_{i+1,i}^{(i)}}{a_{ii}^{(i)}} a_{ii}^{(i)} & \cdots & a_{i+1,n}^{(i)} - \frac{a_{i+1,i}^{(i)}}{a_{ii}^{(i)}} a_{in}^{(i)} \\ & & \vdots & \ddots & \vdots \\ & & a_{ni}^{(i)} - \frac{a_{ni}^{(i)}}{a_{ii}^{(i)}} a_{ii}^{(i)} & \cdots & a_{nn}^{(i)} - \frac{a_{ni}^{(i)}}{a_{ii}^{(i)}} a_{in}^{(i)} \end{pmatrix}.$$

Die Matrix $A^{(i+1)}$ hat formal dieselbe Struktur wie $A^{(i)}$ mit dem Unterschied, dass in $A^{(i+1)}$ unterhalb der Hauptdiagonalen der i-ten Spalte, die in $A^{(i)}$ noch voll besetzt war, Nullen erzeugt wurden. Im nächsten Schritt würden dann die Elemente $a_{j,i+1}^{(i+1)}$, $j = i+2, \dots, n$ elimiert werden.

Entsprechend muss der Vektor $b^{(i)}$ transformiert werden:

$$b^{(i+1)} = \begin{pmatrix} b_1^{(i+1)} \\ \vdots \\ b_i^{(i+1)} \\ b_{i+1}^{(i+1)} \\ \vdots \\ b_n^{(i+1)} \end{pmatrix} := \begin{pmatrix} b_1^{(i)} \\ \vdots \\ b_i^{(i)} \\ b_{i+1}^{(i)} - \frac{a_{i+1,i}^{(i)}}{a_{ii}^{(i)}} b_i^{(i)} \\ \vdots \\ b_n^{(i)} - \frac{a_{ni}^{(i)}}{a_{ii}^{(i)}} b_i^{(i)} \end{pmatrix}.$$

Zusammenfassend lautet das Gauß'sche Eliminationsverfahren wie folgt (der Schleifendurchlauf für $j = k$ erzeugt nur Nulleinträge in A und kann ggf. weggelassen werden):

Algorithmus 2.2.4 (Gauß'scher Eliminationsalgorithmus)

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

Output: Lösung x mit $Ax = b$, A und b werden überschrieben.

%————————- Elimination ————————-
For *k = 1,. . . ,n-1:*
  For $i = k+1, \ldots, n$*:*
    $piv = a_{ik}/a_{kk}$
    For $j = k, \ldots, n$*:*
      $a_{ij} = a_{ij} - piv \cdot a_{kj}$
    end
    $b_i = b_i - piv \cdot b_k$
  end
end
%————– Backward substitution ————-
Führe Rückwärtssubstitution für A und b durch.

2.2.3 Pivoting

Der Gauß'sche Eliminationsalgorithmus ist durchführbar, solange die Pivotelemente $a_{ii}^{(i)} \neq 0$, $i = 1, \ldots, n-1$, erfüllen. Es kann gezeigt werden, dass dies für strikt diagonaldominante Matrizen und symmetrische, positiv definite Matrizen der Fall ist. Für andere Matrizen kann der Fall $a_{ii}^{(i)} = 0$ allerdings sehr schnell eintreten, wie das folgende Beispiel zeigt:

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

Darüber hinaus entstehen numerische Probleme bedingt durch Rundungsfehler und Fehlerfortpflanzung bereits für $a_{ii}^{(i)} \approx 0$ wie das folgende Beispiel zeigt.

Beispiel 2.2.5
Betrachte

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

wobei $\varepsilon \approx 0$*. Die exakte Lösung ist gegeben durch*

$$x_1 = \frac{1}{1-\varepsilon} \approx 1, \quad x_2 = \frac{1-2\varepsilon}{1-\varepsilon} \approx 1.$$

Der Gauß'sche Eliminationsalgorithmus liefert

$$\begin{pmatrix} \varepsilon & 1 \\ 0 & 1-\varepsilon^{-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2-\varepsilon^{-1} \end{pmatrix}.$$

Die Lösung dieses Systems lautet

$$x_2 = \frac{2-\varepsilon^{-1}}{1-\varepsilon^{-1}}, \quad x_1 = (1-x_2)\varepsilon^{-1}.$$

Für betragsmäßig kleine Werte von ε erhält man auf Grund von Rundungsfehlern $x_2 \approx 1$ und anschließend $x_1 \approx 0$, also ein völlig falsches Ergebnis.

Falls wir die Gleichungen vertauschen, d.h.

$$\begin{pmatrix} 1 & 1 \\ \varepsilon & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

und anschließend den Gauß'schen Algorithmus anwenden, erhalten wir

$$\begin{pmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1-2\varepsilon \end{pmatrix}$$

und die (beinahe) exakte Lösung

$$x_2 = \frac{1-2\varepsilon}{1-\varepsilon} \approx 1, \quad x_1 = 2 - x_2 \approx 1.$$

Zeilenvertauschungen haben im obigen Beispiel zum Erfolg geführt. Dies bedeutet, dass die natürliche Reihenfolge bei der Wahl der Pivotelemente geändert wurde. Ziel dabei ist es, im Schritt i durch Zeilenvertauschungen in $A^{(i)}$ **und** $b^{(i)}$ unter den Elementen $a_{ji}^{(i)}$, $j = i, \dots, n$, eines mit $a_{ki}^{(i)} \neq 0$ zu finden und anschließend Zeile i mit Zeile k zu vertauschen. Ein solches Element existiert stets; andernfalls wäre A singulär. Um die Division durch ein betragsmäßig sehr kleines Pivotelement zu vermeiden (numerische Probleme!), wird bei der sogenannten **Spaltenpivotsuche** unter den in Frage kommenden Elementen $a_{ji}^{(i)} \neq 0$ mit $i \leq j \leq n$ das betragsmäßig größte Element

$$|a_{ki}^{(i)}| = \max\{|a_{ji}^{(i)}| \mid i \leq j \leq n,\ a_{ji}^{(i)} \neq 0\} \tag{2.8}$$

als neues Pivotelement verwendet, d.h. die Zeilen i und k werden vertauscht. Der Gauß'sche Algorithmus zusammen mit der Spaltenpivotsuche ist für reguläre Matrizen A bei exakter Rechnung stets durchführbar.

Anstatt der Spaltenpivotsuche kann auch die **Totalpivotsuche** verwendet werden. Im Gegensatz zur Spaltenpivotsuche, die sich nur auf die i-te Spalte von $A^{(i)}$ beschränkt, ermittelt die Totalpivotsuche das Pivotelement $a_{pq}^{(i)}$ mit $i \leq p, q \leq n$ in der kompletten Teilmatrix

$$\begin{pmatrix} a_{ii}^{(i)} & \cdots & a_{in}^{(i)} \\ \vdots & \ddots & \vdots \\ a_{ni}^{(i)} & \cdots & a_{nn}^{(i)} \end{pmatrix}$$

von $A^{(i)}$. Anschließend werden Zeile p mit Zeile i **und** Spalte q mit Spalte i vertauscht. In der Praxis wird die Totalpivotsuche selten verwendet, da der Aufwand mit $\mathcal{O}(n^2)$ pro Eliminationsschritt in derselben Größenordnung liegt wie der Gauß-Algorithmus selbst.

Natürlich müssen Zeilenvertauschungen auch in $b^{(i)}$ und Spaltenvertauschungen in x berücksichtigt werden.
In einer effizienten Implementierung werden Zeilenvertauschungen nicht explizit durchgeführt, um zeitaufwändiges (und unnötiges) Umspeichern zu vermeiden. Anstattdessen werden die Zeilenvertauschungen in einem Indexvektor $l = [l_1, l_2, \ldots, l_n]$ protokolliert.
Die folgende Variante des Gauß'schen Eliminationsverfahrens verwendet eine skalierte Spaltenpivotsuche zur Bestimmung des Pivotelements.

Algorithmus 2.2.6 **(Gauß'sche Elimination mit skalierter Spaltenpivotsuche)**

(i) *Setze* $A^{(1)} := A$, $b^{(1)} := b$, $l := [l_1, l_2, \ldots, l_n] = [1, 2, \ldots, n]$ *und* $k = 1$.

(ii) *Berechne die* **Skalierungsfaktoren**

$$s_{l_i} := \max_{1 \le j \le n} |a^{(k)}_{l_i,j}| \qquad (k \le i \le n).$$

(iii) *Bestimme das Pivotelement* $a^{(k)}_{l_j,k}$ *mit* $k \le j \le n$ *gemäß*

$$\frac{a^{(k)}_{l_j,k}}{s_{l_j}} := \max_{k \le i \le n} \left| \frac{a^{(k)}_{l_i,k}}{s_{l_i}} \right|$$

(iv) *Vertausche Werte* l_k *und* l_j *in* l.

(v) *Verwende Zeile* l_k *von* $A^{(k)}$ *als Pivotzeile und das Element* $a^{(k)}_{l_k,k}$ *als Pivotelement und berechne* $A^{(k+1)}$ *und* $b^{(k+1)}$.

(vi) *Falls* $k < n - 1$, *ersetze* k *durch* $k + 1$ *und gehe zu (iii).*

(vii) *Löse* $A^{(n)}x = b^{(n)}$ *durch Rückwärtssubstitution in der Reihenfolge* $l_n, l_{n-1}, \ldots, l_1$.

Bemerkung 2.2.7

- *Es existiert eine Variante des Algorithmus, bei dem die Skalierungsfaktoren* s_i *in jedem Durchlauf neu berechnet werden.*
- *Es ist wichtig zu bemerken, dass lediglich die Einträge in dem Indexvektor* l *vertauscht werden. Dies vermeidet das kostenintensive Umspeichern von Zeilen.*
- *Die Berechnung von* $A^{(k+1)}$ *und* $b^{(k+1)}$ *in Schritt (v) erfolgt wie in (2.2.4), wobei der Zeilenindex* i *durch* l_i *ersetzt wird, d.h.* $a^{(k)}_{l_i,k}/a^{(k)}_{l_k,k}$ *multipliziert mit Zeile* l_k *wird von den Gleichungen* l_i, $i = k + 1, \ldots, n$, *subtrahiert, wohingegen die Zeilen* $l_1, \ldots, l_k$ *unverändert bleiben.*

Beispiel 2.2.8

Betrachte

$$\underbrace{\begin{pmatrix} 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \\ 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \end{pmatrix}}_{=A^{(1)}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \underbrace{\begin{pmatrix} -19 \\ -34 \\ 16 \\ 26 \end{pmatrix}}_{=b^{(1)}}.$$

Anwendung der Gauß'schen Elimination mit skalierter Spaltenpivotsuche liefert:

Init: $l = [1,2,3,4]$, $s = [13,18,6,12]$.

Schritt 1: $k = 1$. *Pivotelement (nicht eindeutig):*

$$\max_{1\le i\le 4} \left| \frac{a_{l_i,1}^{(1)}}{s_{l_i}} \right| = \max\left\{ \frac{3}{13}, \frac{6}{18}, \frac{6}{6}, \frac{12}{12} \right\} = \frac{6}{6} = \left| \frac{a_{3,1}^{(1)}}{s_3} \right|.$$

Zeile 3 ist Pivotzeile. Vertausche Zeilenindizes: $l = [3,2,1,4]$. *Elimination:*

$$A^{(2)} = \begin{pmatrix} 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \\ 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} -27 \\ -18 \\ 16 \\ -6 \end{pmatrix}.$$

Schritt 2: $k = 2$. *Pivotelement:*

$$\max_{2\le i\le 4} \left| \frac{a_{l_i,2}^{(2)}}{s_{l_i}} \right| = \max\left\{ \frac{2}{18}, \frac{12}{13}, \frac{4}{12} \right\} = \frac{12}{13} = \left| \frac{a_{1,2}^{(2)}}{s_1} \right|.$$

Zeile 1 ist Pivotzeile. Vertausche Zeilenindizes: $l = [3,1,2,4]$. *Elimination:*

$$A^{(3)} = \begin{pmatrix} 0 & -12 & 8 & 1 \\ 0 & 0 & 13/3 & -83/6 \\ 6 & -2 & 2 & 4 \\ 0 & 0 & -2/3 & 5/3 \end{pmatrix}, \quad b^{(3)} = \begin{pmatrix} -27 \\ -45/2 \\ 16 \\ 3 \end{pmatrix}.$$

Schritt 3: $k = 3$. *Pivotelement:*

$$\max_{3\le i\le 4} \left| \frac{a_{l_i,3}^{(3)}}{s_{l_i}} \right| = \max\left\{ \frac{13/3}{18}, \frac{2/3}{12} \right\} = \frac{13/3}{18} = \left| \frac{a_{2,3}^{(3)}}{s_2} \right|.$$

Zeile 2 ist Pivotzeile und $l = [3,1,2,4]$. *Elimination:*

$$A^{(4)} = \begin{pmatrix} 0 & -12 & 8 & 1 \\ 0 & 0 & 13/3 & -83/6 \\ 6 & -2 & 2 & 4 \\ 0 & 0 & 0 & -6/13 \end{pmatrix}, \quad b^{(4)} = \begin{pmatrix} -27 \\ -45/2 \\ 16 \\ -6/13 \end{pmatrix}.$$

Rückwärtssubstitution: $x_4 = 1$, $x_2 = 1$, $x_1 = 3$, $x_3 = -2$.

Der Aufwand berechnet sich zu

Satz 2.2.9 (Aufwand)
Der Gauß'sche Eliminationsalgorithmus mit skalierter Spaltenpivotsuche zur Lösung von $Ax = b$ mit $A \in \mathbb{K}^{n\times n}$, $x, b \in \mathbb{K}^n$, benötigt

$$\frac{1}{3}n^3 + \frac{3}{2}n^2 + \frac{1}{6}n - 1$$

wesentliche Rechenoperationen (Multiplikationen oder Divisionen).

Beweis: Der Aufwand zur Bestimmung der Skalierungsfaktoren enthält keine Multiplikationen oder Divisionen, welches die teueren Rechenoperationen sind, und wird daher nicht gerechnet.
Die Anzahl der Multiplikationen und Divisionen pro Hauptiteration $k = 1, \dots, n-1$ beträgt $n - k + 1$ für die Spaltenpivotsuche und $2(n-k) + (n-k)^2$ für die Elimination, insgesamt also

$$\begin{aligned}
& \sum_{k=1}^{n-1} \big(n - k + 1 + 2(n-k) + (n-k)^2\big) \\
= & \sum_{k=1}^{n-1} \big(3n + 1 + n^2 - (2n+3)k + k^2\big) \\
= & \; 3(n-1)n + n - 1 + (n-1)n^2 - (2n+3)\frac{(n-1)n}{2} + \frac{(n-1)n(2n-1)}{6} \\
= & \; \frac{1}{3}n^3 + n^2 - \frac{1}{3}n - 1.
\end{aligned}$$

Der Aufwand für die Rückwärtssubstitution beträgt

$$1 + n - 1 + \sum_{i=1}^{n-1}(n-i) = n + (n-1)n - \frac{(n-1)n}{2} = \frac{n^2}{2} + \frac{n}{2}.$$

Insgesamt beträgt der Aufwand $\frac{1}{3}n^3 + \frac{3}{2}n^2 + \frac{1}{6}n - 1$ Multiplikationen und Divisionen. □

2.2.4 Die LR-Zerlegung einer Matrix

In diesem Abschnitt werden wir sehen, dass das Gauß'sche Eliminationsverfahren nicht nur ein oberes Dreieckssystem produziert, sondern tatsächlich eine Faktorisierung

$$A = L \cdot R \tag{2.9}$$

der Matrix A erzeugt, wobei L eine untere Dreiecksmatrix und R eine obere Dreiecksmatrix ist.

Das lineare Gleichungssystem (2.3) kann dann gelöst werden, indem zunächst $y := Rx$ definiert wird und mit Hilfe der Vorwärtssubstitution

$$Ly = b$$

gelöst wird. Damit erhält man y. Im Anschluss wird mit der Rückwärtssubstitution

$$Rx = y$$

gelöst (y ist darin bereits bekannt). Dies liefert schließlich die Lösung x von (2.3). Dieser Vorgang wird **Vorwärts-Rückwärtssubstitution** genannt.
Diese Vorgehensweise ist besonders dann von Vorteil, falls das lineare Gleichungssystem (2.3) für viele verschiedene rechte Seiten b gelöst werden muss. Die $LR-$Zerlegung muss nur einmalig berechnet werden.

Beispiel 2.2.10 **(Berechnung der Inversen A^{-1})**
Die Inverse $X = A^{-1}$ einer invertierbaren Matrix A ist gegeben durch

$$A \cdot X = I.$$

Falls $x^{(i)}$ die i-te Spalte von X und $e^{(i)}$ den i-ten Einheitsvektor bezeichnen, müssen wir zur Bestimmung von X die folgenden n linearen Gleichungssysteme lösen:

$$Ax^{(i)} = e^{(i)}, \qquad i = 1, 2, \ldots, n.$$

Falls die $LR-$Zerlegung von A bekannt ist, müssen wir lediglich n Vorwärts-Rückwärts-Substitutionen durchführen. Der Gesamtaufwand zur Invertierung der Matrix beträgt damit $\mathcal{O}(n^3)$ (LR: $\mathcal{O}(n^3)$; FRS: $\mathcal{O}(n \cdot n^2)$). Würde das Gauß'sche Eliminationsverfahren für jedes der n Gleichungssysteme von Neuem angewendet werden, würde der Aufwand $\mathcal{O}(n^4)$ betragen.

Bemerkung 2.2.11
Es ist **niemals** *notwendig, die Inverse A^{-1} einer Matrix explizit zu berechnen, wenn wir die Lösung $x = A^{-1}b$ eines linearen Gleichungssystems berechnen wollen. Es genügt, die LR-Zerlegung von A zu berechnen und Vorwärts-Rückwärtssubstitution durchzuführen.*

Beispiel 2.2.12

$$A = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix}.$$

Jeder Schritt des Gauß'schen Eliminationsverfahrens entspricht der Multiplikation von links mit einer Matrix L_k, so dass

$$\underbrace{\underbrace{\underbrace{L_3 \cdot L_2 \cdot \underbrace{L_1 \cdot A}_{=A^{(2)}}}_{=A^{(3)}}}_{=A^{(4)}} = R}_{} \qquad \Rightarrow \qquad A = \underbrace{L_1^{-1} \cdot L_2^{-1} \cdot L_3^{-1}}_{=:L} \cdot R$$

wobei

$$L_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, L_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{pmatrix}, L_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix}.$$

Zusammenfassend erzeugt das Gauß'sche Eliminationsverfahren die LR-Zerlegung von A:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}}_{=L_1^{-1}} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & -\frac{1}{2} & 0 & 1 \end{pmatrix}}_{=L_2^{-1}} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}}_{=L_3^{-1}}$$

und

$$R = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix}.$$

(Verifiziere, dass tatsächlich $L \cdot R = A$ gilt!)

Der Transformationsschritt $i \mapsto i+1$ im Gauß'schen Eliminationsverfahren kann mit Hilfe von linksseitigen Matrixmultiplikationen beschrieben werden. Es gilt (nachprüfen!)

$$A^{(i+1)} = L_i \cdot A^{(i)}$$

mit

$$L_i = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{i+1,i} & 1 & \\ & & \vdots & & \ddots \\ & & l_{ni} & & 1 \end{pmatrix}, \qquad l_{ji} = -\frac{a_{ji}^{(i)}}{a_{ii}^{(i)}}, \quad j = i+1, \ldots, n. \tag{2.10}$$

Die Inverse L_i^{-1} zu L_i lässt sich leicht angeben (nachrechnen!):

$$L_i^{-1} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{i+1,i} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{ni} & & 1 \end{pmatrix}.$$

Das Gauß'sche Eliminationsverfahren liefert also nach $n-1$ Schritten die Beziehung

$$\begin{aligned} R &= A^{(n)} \\ &= L_{n-1} \cdot A^{(n-1)} \\ &= L_{n-1} \cdot L_{n-2} \cdot A^{(n-2)} \\ &\vdots \\ &= L_{n-1} \cdot L_{n-2} \cdots L_1 \cdot A^{(1)}. \end{aligned} \tag{2.11}$$

Mit $L := L_1^{-1} \cdots L_{n-2}^{-1} \cdot L_{n-1}^{-1}$ und $A^{(1)} = A$ folgt

$$A = L \cdot R.$$

Da das Produkt von zwei normierten linken unteren Dreiecksmatrizen wieder eine normierte linke untere Dreiecksmatrix ergibt, ist L normierte linke untere Dreiecksmatrix mit

$$L = \begin{pmatrix} 1 & & & \\ -l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ -l_{n1} & \cdots & -l_{n,n-1} & 1 \end{pmatrix}. \tag{2.12}$$

wobei die l_{ij}'s durch (2.10) gegeben sind (verifiziere dies!).
Um Speicherplatz zu sparen, speichern effiziente Computerimplementierungen L und R in A, d.h. die Elemente l_{ij} werden unterhalb der Diagonalen gespeichert und die Elemente von R werden auf und oberhalb der Diagonalen gespeichert.

Algorithmus 2.2.13 (LR-Zerlegung ohne Pivoting)
Input: Matrix $A \in \mathbb{K}^{n\times n}$, rechte Seite $b \in \mathbb{K}^n$
Output: Lösung $x \in \mathbb{K}^n$ des LGS $Ax = b$, A wird überschrieben und enthält unterhalb der Diagonalen die Matrix L und auf und oberhalb der Diagonalen die Matrix R

%————————- LR Zerlegung ————————
For *k = 1,. . . ,n-1:*

For $i = k+1, \ldots, n$*:*
  $piv = a_{ik}/a_{kk}$
  For $j = k+1, \ldots, n$*:*
    $a_{ij} = a_{ij} - piv \cdot a_{kj}$
  end
  $a_{ik} = piv$
end
end
%————– Vorwaertssubstitution Ly=b (Ergebnis y steht in x) ————-
$x_1 = b_1$
For *i=2,. . . ,n:*
  $x_i = b_i$
  For $j = 1, \ldots, i-1$*:*
    $x_i = x_i - a_{ij} \cdot x_j$
  end
end
%————– Rueckwaertssubstitution Rx=y ————-
$x_n = x_n/a_{nn}$
For *i=n-1,. . . ,1:*
  For $j = i+1, \ldots, n$*:*
    $x_i = x_i - a_{i,j} \cdot x_j$
  end
  $x_i = x_i/a_{ii}$
end

2.2.5 Pivoting in Matrixnotation

Die Spaltenpivotsuche lässt sich ebenfalls durch Matrixmultiplikationen darstellen. Die Multiplikation einer Matrix von links mit der Permutationsmatrix

$$P = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & 0 & & 1 & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ & & & 1 & & 0 & \\ & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \text{Zeile i} \\ \\ \\ \\ \text{Zeile j} \\ \\ \\ \end{matrix}$$

Spalte i Spalte j

vertauscht die **Zeilen** i und j, während die Multiplikation von rechts die **Spalten** i und j vertauscht. Zusätzlich gilt $P^\top P = I$ und $P = P^\top$. Die Spaltenpivotsuche im Schritt i entspricht daher der Multiplikation von $A^{(i)}$ von links mit einer Permutationsmatrix P_i bevor die Subdiagonalelemente eliminiert werden: $A^{(i+1)} = L_i \cdot P_i \cdot A^{(i)}$. Analog zu (2.11) erhält man

$$R = L_{n-1} \cdot P_{n-1} \cdot L_{n-2} \cdot P_{n-2} \cdots L_1 \cdot P_1 \cdot A. \tag{2.13}$$

Man kann zeigen dass für $i < k$ die folgende Beziehung gilt:

$$P_k \cdot L_i = \tilde{L}_i \cdot P_k.$$

Darin entsteht $\tilde{L}_i$ aus L_i durch Vertauschung der Elemente l_{ki} und l_{ji}, wobei j den Zeilenindex bezeichnet, der durch P_k mit k vertauscht wird. Damit lässt sich (2.13) zu

$$P \cdot A = \tilde{L} \cdot R$$

mit $P = P_{n-1} \cdot P_{n-2} \cdots P_1$ und $\tilde{L} = \tilde{L}_1^{-1} \cdots \tilde{L}_{n-2}^{-1} \cdot L_{n-1}^{-1}$ umformen. Beachte, dass $\tilde{L}_1, \ldots, \tilde{L}_{n-3}$ mehrfache Elementvertauschungen enthalten können.

Bei der Totalpivotsuche werden zusätzlich noch Spaltenvertauschungen notwendig: $A^{(i+1)} = L_i \cdot P_i \cdot A^{(i)} \cdot Q_i$. Analoges Vorgehen liefert

$$P \cdot A \cdot Q = \tilde{L} \cdot R.$$

P und $\tilde{L}$ sind wie oben erklärt. Q ergibt sich zu $Q = Q_1 \cdot Q_{n-2} \cdots Q_{n-1}$.

Satz 2.2.14 (LR-Zerlegung mit Pivoting)
Sei $A \in \mathbb{K}^{n\times n}$ invertierbar. Dann definiert das Gauß'sche Eliminationsverfahren mit Spaltenpivotsuche eine Zerlegung $PA = LR$ mit einer rechten oberen Dreiecksmatrix R, einer normierten linken unteren Dreiecksmatrix L und einer Permutationsmatrix P.

Beweis: Bricht das Gauß'sche Eliminationsverfahren mit Spaltenpivotsuche nicht zusammen, d.h. sind alle Pivotelemente $a_{ki}^{(i)} \neq 0$ für alle $i = 1, \ldots, n$, so liefern die vorangegangenen Betrachtungen die Behauptung.
Zu klären ist, dass das Verfahren für invertierbare Matrizen tatsächlich nicht zusammenbricht, d.h. dass alle Pivotelemente von Null verschieden sind. Angenommen, das Pivotelement im i-ten Teilschritt wäre Null, dann gilt gemäß der Auswahlregel (2.8) des Pivotelements $a_{ji}^{(i)} = 0$ für alle $i \leq j \leq n$ und $A^{(i)}$ hätte folgende Gestalt:

$$A^{(i)} = \left(\begin{array}{ccc|ccc} a_{11}^{(i)} & \cdots & & a_{1i}^{(i)} & \ldots & a_{1n}^{(i)} \\ & \ddots & & \vdots & \ddots & \vdots \\ \hline & & & 0 & \ldots & a_{in}^{(i)} \\ & & & 0 & \ldots & a_{i+1,n}^{(i)} \\ & & & \vdots & \ddots & \vdots \\ & & & 0 & \ldots & a_{nn}^{(i)} \end{array}\right).$$

Die Determinante des rechten unteren Teilblocks ist folglich 0 und somit ist $\det(A^{(i)}) = 0$. Wegen

$$\det(A^{(i)}) = \det(L_{i-1} \cdot P_{i-1} \cdots L_1 \cdot P_1 \cdot A) = \left(\prod_{j=1}^{i-1} \det(L_j)\det(P_j)\right)\det(A)$$

und $\det(L_j) = 1$, $\det(P_j) = \pm 1$, $j = 1, \ldots, i-1$, folgt dann $\det(A) = 0$ im Widerspruch zur Invertierbarkeit von A. □
Ohne Beweis sei erwähnt, dass

- die LR-Zerlegung ohne Pivoting für allgemeine invertierbare Matrizen **nicht** vorwärtsstabil ist.
- die LR-Zerlegung mit Spaltenpivoting für allgemeine invertierbare Matrizen **vorwärtsstabil** ist.
- die LR-Zerlegung mit totaler Pivotsuche für allgemeine invertierbare Matrizen **rückwärtsstabil** ist.

Beispiel 2.2.15

Wir lösen das lineare Gleichungssystem

$$Ax = b \quad mit \quad A = \begin{pmatrix} 2 & 18 & 9 \\ 4 & 4 & 4 \\ 1 & 33 & 9 \end{pmatrix}, \quad b = \begin{pmatrix} 21 \\ 4 \\ 9 \end{pmatrix},$$

indem wir die LR-Zerlegung mit Spaltenpivotsuche berechnen.

1.Schritt: *Pivotelement* $a_{21} = 4$.

$$P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad L_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{4} & 0 & 1 \end{pmatrix}, \quad A^{(2)} = \begin{pmatrix} 4 & 4 & 4 \\ 0 & 16 & 7 \\ 0 & 32 & 8 \end{pmatrix}$$

2.Schritt: *Pivotelement* $a_{32}^{(2)} = 32$.

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{pmatrix}, \quad A^{(3)} = \begin{pmatrix} 4 & 4 & 4 \\ 0 & 32 & 8 \\ 0 & 0 & 3 \end{pmatrix}$$

Insgesamt:

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \tilde{L} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 4 & 4 & 4 \\ 0 & 32 & 8 \\ 0 & 0 & 3 \end{pmatrix}$$

Lösung: *Es ist* $\tilde{b} := P \cdot b = P \cdot Ax = \tilde{L} \cdot Rx$. *Es wird also* $\tilde{L} \cdot Rx = \tilde{b} = (4, 9, 21)^\top$ *mit Vorwärts-Rückwärtssubstitution gelöst.*

$$\begin{aligned} \textbf{Vorwärtssubstitution:} \quad & \tilde{L}y = \tilde{b} \quad \Rightarrow \quad y = (4, 8, 15)^\top \\ \textbf{Rückwärtssubstitution:} \quad & Rx = y \quad \Rightarrow \quad x = (-3, -1, 5)^\top \end{aligned}$$

2.3 Cholesky-Zerlegung

Treten bei einer Anwendung ausschließlich lineare Gleichungssysteme mit hermitescher, positiv definiter Matrix $A \in \mathbb{K}^{n \times n}$ auf, sollte die Cholesky-Zerlegung an Stelle der LR-Zerlegung verwendet werden, da diese nur etwa den halben Aufwand erfordert.

Definition 2.3.1 **(hermitesche Matrix)**

Eine Matrix $A \in \mathbb{K}^{n \times n}$ *heißt* **hermitesch**, *wenn* $A^* = A$ *gilt, wobei* $A^* = \bar{A}^\top$ *gilt.*

Definition 2.3.2 **(positive Definitheit)**

Eine Matrix $A \in \mathbb{K}^{n \times n}$ *heißt* **positiv definit**, *wenn* $x^* Ax > 0$ *für alle* $x \neq 0$, $x \in \mathbb{K}^n$,

gilt.

Definition 2.3.3 **(Cholesky-Zerlegung)**
Sei $A \in \mathbb{K}^{n\times n}$. Eine Zerlegung der Form

$$A = L \cdot L^*$$

mit linker unterer (nicht normierter!) Dreiecksmatrix L und positiven Diagonalelementen heißt **Cholesky-Zerlegung von** *A.*

Es gilt folgender Satz.

Satz 2.3.4
Sei $A \in \mathbb{K}^{n\times n}$ hermitesch. A besitzt eine Cholesky-Zerlegung genau dann, wenn A positiv definit ist.

Diese Äquivalenz von Cholesky-Zerlegung und positiver Definitheit einer hermiteschen Matrix liefert einen praktisch durchführbaren Algorithmus, um eine gegebene hermitesche Matrix auf positive Definitheit zu testen. Man muss lediglich die Cholesky-Zerlegung berechnen. Falls sie existiert (d.h. es gilt $\ell_{kk} > 0$ für alle k), ist A positiv definit, andernfalls nicht.
Die konkrete Berechnung von L erfolgt durch zeilenweisen Koeffizientenvergleich im Ansatz

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} \ell_{11} & & & \\ \ell_{21} & \ell_{22} & & \\ \vdots & \vdots & \ddots & \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nn} \end{pmatrix} \cdot \begin{pmatrix} \bar{\ell}_{11} & \bar{\ell}_{21} & \cdots & \bar{\ell}_{n1} \\ & \bar{\ell}_{22} & \cdots & \bar{\ell}_{n2} \\ & & \ddots & \vdots \\ & & & \bar{\ell}_{nn} \end{pmatrix}.$$

Es folgt (beachte: für reelle Matrizen, soll eine reelle Zerlegung entstehen):

$$\begin{aligned}
a_{11} = \ell_{11}\bar{\ell}_{11} = |\ell_{11}|^2 &\Rightarrow \ell_{11} = \sqrt{a_{11}}, \\
a_{21} = \ell_{21}\bar{\ell}_{11} &\Rightarrow \ell_{21} = a_{21}/\bar{\ell}_{11}, \\
a_{22} = \ell_{21}\bar{\ell}_{21} + \ell_{22}\bar{\ell}_{22} &\Rightarrow \ell_{22} = \sqrt{a_{22} - |\ell_{21}|^2}, \\
a_{31} = \ell_{31}\bar{\ell}_{11} &\Rightarrow \ell_{31} = a_{31}/\bar{\ell}_{11}, \\
a_{32} = \ell_{31}\bar{\ell}_{21} + \ell_{32}\bar{\ell}_{22} &\Rightarrow \ell_{32} = (a_{32} - \ell_{31}\bar{\ell}_{21})/\bar{\ell}_{22}, \\
a_{33} = \ell_{31}\bar{\ell}_{31} + \ell_{32}\bar{\ell}_{32} + \ell_{33}\bar{\ell}_{33} &\Rightarrow \ell_{33} = \sqrt{a_{33} - |\ell_{31}|^2 - |\ell_{32}|^2}, \\
&\vdots \quad \vdots \quad \vdots \\
a_{ij} = \sum_{k=1}^{j-1} \ell_{ik}\bar{\ell}_{jk} + \ell_{ij}\bar{\ell}_{jj} &\Rightarrow \ell_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\bar{\ell}_{jk}\right)/\bar{\ell}_{jj}, \qquad j = 1, \ldots, i-1, \\
a_{ii} = \sum_{k=1}^{i-1} \ell_{ik}\bar{\ell}_{ik} + \ell_{ii}\bar{\ell}_{ii} &\Rightarrow \ell_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} \ell_{ik}\bar{\ell}_{ik}}
\end{aligned}$$

Insgesamt entstehen die Formeln

$$\begin{aligned}\ell_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} |\ell_{ik}|^2}, \quad i = 1, \dots, n,\\ \ell_{ij} &= \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\bar{\ell}_{jk}\right) / \bar{\ell}_{jj}, \quad j = 1, \dots, i-1.\end{aligned}$$

Diese Formeln sind für positiv definite Matrizen wohldefiniert, da die Existenz einer Cholesky-Zerlegung bereits durch Satz 2.3.4 garantiert ist. Die obigen Formeln sind auch eindeutig unter der Forderung, dass für relle Matrizen eine reelle Zerlegung entstehen soll.

Beispiel 2.3.5

Cholesky-Zerlegung:

$$A = \begin{pmatrix} 10 & 2 & -1 \\ 2 & 5 & 0 \\ -1 & 0 & 4 \end{pmatrix} \approx \begin{pmatrix} 3.1623 & 0 & 0 \\ 0.6325 & 2.1448 & 0 \\ -0.3162 & 0.0933 & 1.9726 \end{pmatrix} \begin{pmatrix} 3.1623 & 0.6325 & -0.3162 \\ 0 & 2.1448 & 0.0933 \\ 0 & 0 & 1.9726 \end{pmatrix}$$

Da A symmetrisch ist und die Cholesky-Zerlegung existiert ($l_{kk} > 0$ für alle k), folgt, dass A positiv definit ist.

Der Aufwand der Cholesky-Zerlegung beträgt

$$\sum_{i=1}^{n} \left(i - 1 + \sum_{j=1}^{i-1} (j-1) + 1 \right) = \frac{1}{6}n^3 + \frac{1}{2}n^2 - \frac{2}{3}n = \frac{1}{6}n^3 + \mathcal{O}(n^2)$$

Multiplikationen und Divisionen und ist damit in etwa halb so groß wie bei der LR-Zerlegung.

2.4 Fehlereinfluss und Kondition

Gegenstand dieses Abschnitts ist die Untersuchung der Kondition eines linearen Gleichungssystems. Dazu benötigen wir zusätzlich zum Normbegriff für Vektoren auch den Normbegriff für Matrizen.

Beispiel 2.4.1 (Vandermonde-Matrix)

Wir versuchen, das lineare Gleichungssystem

$$\begin{pmatrix} 1 & 2 & 4 & 8 & \cdots & 2^{n-1} \\ 1 & 3 & 9 & 27 & \cdots & 3^{n-1} \\ 1 & 4 & 16 & 64 & \cdots & 4^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & n+1 & (n+1)^2 & (n+1)^3 & \cdots & (n+1)^{n-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} (2^n-1)/1 \\ (3^n-1)/2 \\ (4^n-1)/3 \\ \vdots \\ ((n+1)^n-1)/n \end{pmatrix}$$

für verschiedene Werte von n mittels Gauß'scher Elimination zu lösen. Die auftretende Matrix heißt **Vandermonde-Matrix**. *Die exakte Lösung ist durch $x_i = 1$ für $i = 1, 2, \ldots, n$ gegeben. Auf einem Computer mit ca. 16 Stellen Genauigkeit erhalten wir das exakte Resultat für $n \leq 14$. Für $n = 15$ jedoch erhalten wir plötzlich folgendes Ergebnis (gerundet auf 2 Stellen):*

$$x_1 \approx -894.97, x_2 \approx 2074.34, x_3 \approx -2138.14, x_4 \approx 1309.97, x_5 \approx -531.35, x_6 \approx 153.62,$$
$$x_7 \approx -30.88, x_8 \approx 5.94, x_9 \approx 0.43, x_{10} \approx 1.05, x_{11} \approx \ldots \approx x_{15} \approx 1.00$$

Dieses Ergebnis ist völlig wertlos und hat nichts mit der Lösung zu tun. Für $n > 15$ wird es noch schlimmer.

Was ist der Grund für dieses seltsame Verhalten in Beispiel 2.4.1? Offenbar spielen Rundungsfehler eine Rolle, da ansonsten bei exakter Rechnung die exakte Lösung gefunden werden müsste.
Wir interessieren uns nun für die Frage, wie Fehler in A und b (etwa Rundungsfehler, die beim Gauß'schen Eliminationsverfahren auftreten) die Lösung x von $Ax = b$ beeinflussen. Wir werden sehen, dass die sogenannte Konditionszahl einer Matrix eine entscheidende Rolle spielt.

Definition 2.4.2 **(Vektornorm)**
Sei V ein Vektorraum über $\mathbb{K}$. Die Abbildung $\|\cdot\| : V \to \mathbb{R}$ heißt Norm, wenn sie die folgenden Eigenschaften besitzt:

1. *$\|x\| = 0$ gilt genau dann, falls $x = 0$ ist,*
2. *$\|cx\| = |c| \, \|x\|$ für alle $x \in V$ und $c \in \mathbb{K}$,*
3. *$\|x + y\| \leq \|x\| + \|y\|$ für alle $x, y \in V$.*

Beispiel 2.4.3 **(Vektornormen)**
Häufig verwendete Vektornormen auf $\mathbb{K}^n$ mit $x = (x_1, \ldots, x_n)^\top$ sind

(i) die Maximumnorm: $\|x\|_\infty := \max\limits_{1 \leq i \leq n} |x_i|$;

(ii) die Betragssummennorm: $\|x\|_1 := \sum\limits_{i=1}^{n} |x_i|$;

(iii) die Euklidische Norm: $\|x\|_2 := \sqrt{\sum\limits_{i=1}^{n} |x_i|^2}$.

(iv) p-Norm: Für $p \in \mathbb{N}$ ist $\|x\|_p := \left(\sum\limits_{i=1}^{n} |x_i|^p \right)^{1/p}$ eine Norm.

Bemerkung 2.4.4

In endlichdimensionalen Vektorräumen sind alle Normen äquivalent, d.h. für beliebige Normen $\|\cdot\|_{(1)}$ und $\|\cdot\|_{(2)}$ über demselben Vektorraum V gibt es Konstanten c_1, c_2, so dass für alle $x \in V$ gilt

$$c_1\|x\|_{(2)} \le \|x\|_{(1)} \le c_2\|x\|_{(2)}.$$

Für unendlichdimensionale Vektorräume gilt dies nicht!

Formal kann man einer Vektornorm $\|\cdot\|_V$ eine „passende“ Matrixnorm $\|\cdot\|_M$ zuordnen.

Definition 2.4.5 (Matrixnorm)

Sei $\|\cdot\|_V$ eine Vektornorm. Die durch $\|\cdot\|_V$ induzierte Matrixnorm $\|\cdot\|_M$ auf dem Raum aller linearen Abbildungen $A : \mathbb{K}^n \to \mathbb{K}^m$ ist definiert als

$$\|A\|_M := \sup_{\|x\|_V \neq 0} \frac{\|Ax\|_V}{\|x\|_V} \left(= \sup_{\|x\|_V = 1} \|Ax\|_V = \sup_{\|x\|_V \le 1} \|Ax\|_V \right), \tag{2.14}$$

wobei $x \in \mathbb{K}^n$ und $A \in \mathbb{K}^{m\times n}$ gilt

Beachte: In Definition 2.4.5 wird in $\mathbb{K}^n$ und $\mathbb{K}^m$ dieselbe Vektornorm verwendet! Allgemeiner könnte man auch unterschiedliche Vektornormen verwenden, was wir aus Gründen der Übersichtlichkeit hier nicht tun. Die Wohldefiniertheit der Matrixnorm ergibt sich aus der Stetigkeit der Abbildung $x \mapsto \|Ax\|_V$ und der Kompaktheit der Menge $\{x \in \mathbb{K}^n \mid \|x\|_V = 1\}$. Dass $\|\cdot\|_M$ tatsächlich eine Norm ist, ergibt sich direkt aus den Normeigenschaften von $\|\cdot\|_V$ und kann leicht nachgerechnet werden.
Die so definierte Matrixnorm hat zwei schöne Eigenschaften:

Satz 2.4.6

Sei $\|\cdot\|_V$ eine Vektornorm und $\|\cdot\|_M$ die induzierte Matrixnorm. Dann gelten:

(a) Verträglichkeit, d.h. es gilt $\|Ax\|_V \le \|A\|_M \cdot \|x\|_V$ für alle $x \in \mathbb{K}^n$, $A \in \mathbb{K}^{m\times n}$.

(b) Submultiplikativität, d.h. es gilt $\|A \cdot B\|_M \le \|A\|_M \cdot \|B\|_M$ für alle $A \in \mathbb{K}^{m\times n}$, $B \in \mathbb{K}^{n\times q}$.

Beweis:

(a) Für $x = 0$ ist die Behauptung klar. Für $\|x\|_V \neq 0$ gilt $\|Ax\|_V = \|A\frac{x}{\|x\|_V}\| \cdot \|x\|_V$ und somit

$$\|Ax\|_V \le \left(\sup_{\|x\|_V \neq 0} \|A\frac{x}{\|x\|_V}\| \right) \cdot \|x\|_V = \left(\sup_{\|x\|_V \neq 0} \frac{\|Ax\|_V}{\|x\|_V} \right) \cdot \|x\|_V = \|A\|_M \cdot \|x\|_V.$$

(b) Nach (a) gilt $\|A \cdot Bx\|_V \leq \|A\|_M \cdot \|Bx\|_V$ und somit

$$\|A \cdot B\|_M = \sup_{\|x\|_V \neq 0} \frac{\|A \cdot Bx\|_V}{\|x\|_V} \leq \|A\|_M \cdot \sup_{\|x\|_V \neq 0} \frac{\|Bx\|_V}{\|x\|_V} = \|A\|_M \cdot \|B\|_M.$$

□

Die konkrete Berechnung einer induzierten Matrixnorm für eine gegebene Vektornorm ist mitunter kompliziert. Wir fassen gänge Matrixnormen zusammen.

Satz 2.4.7 **(Zeilensummennorm, Spaltensummennorm, Spektralnorm)**

(a) Die durch die Vektornorm $\|\cdot\|_\infty$ induzierte Matrixnorm ist die sogenannte **Zeilensummennorm** $\|A\|_\infty := \max\limits_{1 \leq i \leq m} \sum\limits_{j=1}^{n} |a_{ij}|$.

(b) Die durch die Betragssummennorm $\|\cdot\|_1$ induzierte Matrixnorm ist die sogenannte **Spaltensummennorm** $\|A\|_1 := \max\limits_{1 \leq j \leq n} \sum\limits_{i=1}^{m} |a_{ij}|$.

(c) Die durch die euklidische Vekotrnorm $\|\cdot\|_2$ induzierte Matrixnorm ist die **Spektralnorm**

$$\|A\|_2 = \sqrt{\rho(A^*A)},$$

wobei $\rho(A^*A)$ den Spektralradius von A^*A bezeichnet, d.h.

$$\rho(A^*A) = \max\{|\lambda| \mid \lambda \text{ ist Eigenwert von } A^*A\}.$$

Bemerkung 2.4.8

Es gibt weitere Matrixnormen, die aber i.a. nicht durch Vektornormen induziert sind:

- *Die Matrixnorm $\|A\| = \max\limits_{1 \leq i \leq m, 1 \leq j \leq n} |a_{ij}|$ ist durch keine Vektornorm induziert, da sie nicht submultiplikativ ist wie folgendes Beispiel zeigt:*

$$A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad A \cdot B = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}, \quad 2 = \|A \cdot B\| > \|A\| \cdot \|B\| = 1.$$

- *Die* **Frobenius-Schur-Norm**

$$\|A\|_F := \sqrt{\sum_{\substack{i=1,\ldots,m \\ j=1,\ldots n}} |a_{ij}|^2},$$

ist verträglich und submultiplikativ bezüglich der euklidischen Vektornorm, aber sie wird nicht durch diese induziert.

Mit diesen Vorbereitungen untersuchen wir den Fehlereinfluss in linearen Gleichungssystemen. Hierbei sei x die Lösung von $Ax = b$.

Störungen in der rechten Seite b:
Wie wirken sich Störungen Δb in der rechten Seite b in (2.3) auf die Lösung $x + \Delta x$ des gestörten Gleichungssystems

$$A(x + \Delta x) = b + \Delta b$$

aus?
Für die Abweichung der gestörten Lösung von der exakten Lösung gilt bezüglich einer Vektornorm $\|\cdot\|_V$ und der zugeordneten Matrixnorm $\|\cdot\|_M$ die Beziehung

$$\begin{aligned} \|\Delta x\|_V &= \|x + \Delta x - x\|_V \\ &= \|A^{-1}(b + \Delta b) - A^{-1}b\|_V \\ &= \|A^{-1}(b + \Delta b - b)\|_V \\ &\leq \|A^{-1}\|_M \cdot \|\Delta b\|_V. \end{aligned}$$

Mit Hilfe der **absoluten Konditionszahl** $\kappa_a = \|A^{-1}\|_M$ (vgl. Abschnitt 1.5) ergibt sich für den absoluten Fehler die Abschätzung

$$\|\Delta x\|_V \leq \kappa_a \cdot \|\Delta b\|_V. \tag{2.15}$$

Mit (2.15) und der Verträglichkeit folgt für $\|x\|_V, \|b\|_V \neq 0$ die Abschätzung

$$\|A^{-1}\|_M \cdot \frac{\|\Delta b\|_V}{\|b\|_V} = \|A^{-1}\|_M \cdot \frac{\|\Delta b\|_V}{\|Ax\|_V} \geq \frac{\|\Delta x\|_V}{\|Ax\|_V} \geq \frac{\|\Delta x\|_V}{\|A\|_M \cdot \|x\|_V}.$$

Umformen ergibt eine Abschätzung des relativen Fehlers

$$\frac{\|\Delta x\|_V}{\|x\|_V} \leq \|A\|_M \cdot \|A^{-1}\|_M \cdot \frac{\|\Delta b\|_V}{\|b\|_V}. \tag{2.16}$$

Definition 2.4.9 **(Kondition einer Matrix)**
Die Zahl

$$\kappa_M(A) := \|A\|_M \cdot \|A^{-1}\|_M$$

heißt **Konditionszahl** *der Matrix A bzgl. der Matrixnorm $\|\cdot\|_M$. A heißt* **schlecht konditioniert**, *falls $\kappa_M(A)$ sehr groß ist.*

Die Kondition erfüllt wegen

$$1 = \|I\|_M = \|A^{-1}A\|_M \leq \|A^{-1}\|_M \cdot \|A\|_M = \kappa_M(A)$$

stets $\kappa_M(A) \geq 1$.
Mit dieser Definition lässt sich (2.16) schreiben als

$$\frac{\|\Delta x\|_V}{\|x\|_V} \leq \kappa_M(A) \cdot \frac{\|\Delta b\|_V}{\|b\|_V}.$$

Die Kondition $\kappa_M(A)$ einer Matrix A macht also eine Aussage darüber, wie stark sich Störungen in der rechten Seite auf das Ergebnis auswirken können. Insbesondere bei schlecht konditionierten Matrizen mit $\kappa_M(A) >> 1$ können sich bereits kleine Störungen in b sehr stark auf das Ergebnis auswirken.
Vergleich mit Abschnitt 1.5: dort ist $\kappa_r = \kappa_a \cdot \|b\|_V/\|x\|_V$; hier ist $\tilde{\kappa}_r = \|A\|_M \cdot \kappa_a$ mit $\tilde{\kappa}_r \geq \kappa_r$ gewählt, denn $\|b\|_V/\|x\|_V = \|Ax\|_V/\|x\|_V \leq \|A\|_M$.

Störungen in der Matrix A und der rechten Seite b:
Wir untersuchen nun die Auswirkungen von Störungen ΔA in A und Δb in b in (2.3) auf die Lösung $x + \Delta x$ des gestörten Gleichungssystems

$$(A + \Delta A)(x + \Delta x) = b + \Delta b.$$

Es gilt folgender Satz, der hier nicht bewiesen werden soll.

Satz 2.4.10

Es gelte $Ax = b$ und $(A + \Delta A)(x + \Delta x) = b + \Delta b$. Dann gilt

$$\frac{\|\Delta x\|_V}{\|x\|_V} \leq \frac{\kappa_M(A)}{1 - \kappa_M(A) \cdot \frac{\|\Delta A\|_M}{\|A\|_M}} \cdot \left(\frac{\|\Delta A\|_M}{\|A\|_M} + \frac{\|\Delta b\|_V}{\|b\|_V} \right),$$

falls $\kappa_M(A) \cdot \frac{\|\Delta A\|_M}{\|A\|_M} < 1$ und $\|x\|_V \neq 0$ ist.

Beispiel 2.4.11 (Vandermonde-Matrix, Revisited)

Nun können wir den Grund für das seltsame Verhalten in Beispiel 2.4.1 angeben. Die Vandermonde-Matrix ist ein Beispiel für eine extrem schlecht konditionierte Matrix.
Faustregel: *Falls $\kappa_M(A) = 10^k$, dann muss man mit dem Verlust von mindestens k Stellen Genauigkeit bei der Lösung von $Ax = b$ rechnen. Für die Vandermonde-Matrix A mit $n = 15$ gilt*

$$\kappa_M(A) = 1024224393669250854080 > 10^{21}.$$

Wir untersuchen noch einige Eigenschaften der Kondition bzgl. der Spektralnorm $\|\cdot\|_2$. Diese Norm bietet einige Vorteile, da für unitäre Matrizen $Q \in \mathbb{K}^{n\times n}$ für beliebige $x \in \mathbb{K}^n$ die Beziehung

$$\|Qx\|_2^2 = x^*Q^*Qx = x^*x = \|x\|_2^2 \quad \text{bzw.} \quad \|Qx\|_2 = \|x\|_2$$

gilt. Da mit Q auch Q^{-1} unitär ist, folgt ebenso $\|Q^{-1}x\|_2 = \|x\|_2$. Aus der Definition einer Matrixnorm folgt damit $\|Q\|_2 = \|Q^{-1}\|_2 = 1$ und somit

$$\kappa_2(Q) = 1.$$

Unitäre Matrizen haben also eine bzgl. der Spektralnorm optimale Kondition. Dies ist der Grund, warum wir später mit orthogonalen Householder-Matrizen arbeiten werden, vgl. Abschnitt 3.2.3.

2.5 Iterative Lösung

Der Aufwand zur Berechnung der LR-Zerlegung und der Cholesky-Zerlegung (und später der QR-Zerlegung) für quadratische Matrizen ist jeweils von der Ordnung $\mathcal{O}(n^3)$. Der Aufwand wächst also kubisch mit der Dimension n des Gleichungssystems. Für sehr große Gleichungssysteme mit teilweise mehr als einer Milliarde Gleichungen, wie sie in praktischen Anwendungen durchaus entstehen können, ist der Aufwand damit zu groß. Es werden alternative Methoden benötigt, die weniger aufwendig sind.

Beispiel 2.5.1 (Diskretisierung einer partiellen Differentialgleichung)
Große, dünn besetzte Gleichungssysteme entstehen bei Diskretisierungsverfahren zur approximativen Lösung partieller Differentialgleichungen. Als Beispiel sei die 2-dimensionale partielle Differentialgleichung

$$\begin{aligned} -u_{xx}(x,y) - u_{yy}(x,y) &= f(x,y), \qquad (x,y) \in \Omega := (0,1)\times(0,1), \\ u(x,y) &= 0, \qquad (x,y) \in \partial\Omega \end{aligned}$$

genannt. Hierbei ist eine Funktion $u : \Omega \to \mathbb{R}$ gesucht, die die obige partielle Differentialgleichung erfüllt. Eine gängige Vorgehensweise besteht in der Diskretisierung der Differentialgleichung auf dem äquidistanten Gitter

$$G := \{(x_i, y_j) \mid x_i = ih,\ y_j = jh,\ 0 \le i,j \le N\}, \quad h = 1/N.$$

Approximation der zweiten partiellen Ableitungen durch

$$\begin{aligned} u_{xx}(x_i,y_j) &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}, \quad 1 \le i,j \le N-1, \\ u_{yy}(x_i,y_j) &\approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}, \quad 1 \le i,j \le N-1, \end{aligned}$$

mit $u_{i,j} \approx u(x_i,y_j)$ liefert das lineare Gleichungssystem $Au = b$ für den Vektor

$$u = (u_{1,1}, u_{1,2}, \dots, u_{1,N-1}, u_{2,1}, u_{2,2}, \dots, u_{2,N-1}, \dots, u_{N-1,1}, u_{N-1,2}, \dots, u_{N-1,N-1})^\top \in \mathbb{R}^{(N-1)^2},$$

mit

$$b = h^2(f_{1,1}, f_{1,2}, \dots, f_{1,N-1}, f_{2,1}, f_{2,2}, \dots, f_{2,N-1}, \dots, f_{N-1,1}, f_{N-1,2}, \dots, f_{N-1,N-1})^\top \in \mathbb{R}^{(N-1)^2},$$

$f_{i,j} := f(x_i, y_j)$,

$$A = \begin{pmatrix} M_1 & D_2 & & & \\ D_2 & M_2 & D_3 & & \\ & \ddots & \ddots & \ddots & \\ & & D_{N-2} & M_{N-2} & D_{N-1} \\ & & & D_{N-1} & M_{N-1} \end{pmatrix}, \quad M_i = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{(N-1)^2}$$

und $D_i = -I \in \mathbb{R}^{(N-1)\times(N-1)}$. Die Matrix A ist sehr groß (Dimension wächst quadratisch mit N), aber sehr dünn besetzt (pro Zeile sind maximal 5 Einträge $\neq 0$). Zudem ist sie positiv definit und symmetrisch (Beweis?).
In der Praxis werden nur die von Null verschiedenen Einträge der Matrix A und die zugehörigen Spalten- und Zeilenpositionen gespeichert. Damit kann man Multiplikationen der Form $A \cdot d$ mit einem Vektor d sehr effizient berechnen (der Aufwand hängt linear von der Anzahl der nichtverschwindenden Einträge in A ab). Wir werden im Folgenden sehen, dass die sogenannten iterativen Verfahren lediglich Matrix-Vektor Multiplikationen benötigen.

Die Idee der **iterativen Verfahren** zur Lösung von linearen Gleichungssystemen (2.3) basiert auf der geschickten Zerlegung der Matrix A in

$$A = M - N \tag{2.17}$$

mit Matrizen M und N, die noch genauer spezifiziert werden müssen. Es wird angenommen, dass M regulär ist und „leicht“ invertiert werden kann. Dann lässt sich das lineare Gleichungssystem

$$b = Ax = (M - N)x = Mx - Nx$$

umformen zu

$$x = M^{-1}Nx + M^{-1}b. \tag{2.18}$$

Mit $T := M^{-1}N$ und $c := M^{-1}b$ ist dies eine **Fixpunktgleichung** für x:

$$x = Tx + c. \tag{2.19}$$

Damit ist jede Lösung der Fixpunktgleichung (2.19) auch Lösung des linearen Gleichungssystems (2.3) und umgekehrt.

Definition 2.5.2 **(Fixpunkt)**
$\hat{x}$ heißt **Fixpunkt** *der Abbildung $g : \mathbb{K}^n \to \mathbb{K}^n$, $x \mapsto g(x)$, wenn*

$$\hat{x} = g(\hat{x})$$

gilt.

Eine einfache Idee zur Berechnung eines Fixpunkts $\hat{x}$ von g besteht darin, ausgehend von einem Startwert $x^{(0)}$ eine **Fixpunktiteration**

$$x^{(i+1)} = g(x^{(i)}), \quad i = 0, 1, 2, \ldots$$

durchzuführen, indem das Ergebnis immer wieder in die Funktion eingesetzt wird. Auf diese Weise erhält man eine Folge $\left\{x^{(i)}\right\}$. Es stellt sich natürlich die Frage, ob die Folge auch tatsächlich gegen den gesuchten Fixpunkt $\hat{x}$ konvergiert.

Beispiel 2.5.3

(i) *Wir wollen einen Fixpunkt $\hat{x}$ mit $\hat{x} = \cos(\hat{x})$ bestimmen. Wir führen die Fixpunktiteration ausgehend von $x^{(0)} = 1$ für die Funktion $g(x) = \cos(x)$ durch. Die Fixpunktiteration lässt sich ganz einfach mit dem Taschenrechner durchführen, indem 1 eingetippt wird und anschließend immer wieder auf* cos *gedrückt wird (im Bogenmaß rechnen!). Nach mehrmaligem Drücken der* cos*-Taste erhält man mit fünf Nachkommastellen $\hat{x} = 0.739085$. Weiteres Drücken der* cos*-Taste ändert das Ergebnis nicht mehr wesentlich. Die Folge konvergiert offensichtlich.*

(ii) *Jetzt wollen wir einen Fixpunkt von $g(x) = \exp(x)$ mit Startwert $x^{(0)} = 1$ berechnen. Hier das Ergebnis:*

$$\begin{aligned} x^{(1)} &= 2.71828183 \\ x^{(2)} &= 15.15426223 \\ x^{(3)} &= 0.38142791 \cdot 10^{7} \\ x^{(4)} &= 0.22316774 \cdot 10^{1656521} \end{aligned}$$

Der nächste Iterationsschritt liefert einen Exponentenüberlauf. Die Folge konvergiert offensichtlich nicht.

Beispiel 2.5.3 zeigt, dass die Fixpunktiteration nicht immer sinnvolle Resultate liefert. Offensichtlich hängt die Konvergenz von Eigenschaften der Funktion g ab. Ein hinreichendes Kriterium für die Konvergenz der Fixpunktfolge liefert

Satz 2.5.4 (Banachscher Fixpunktsatz)
Sei eine **Selbstabbildung** $g : D \to D$ *mit* **abgeschlossener Menge** $D \subseteq \mathbb{K}^n$ *gegeben. Für alle $x, y \in D$ sei die* **Kontraktionsbedingung**

$$\|g(x) - g(y)\| \leq q\|x - y\| \quad \text{mit} \quad q < 1 \tag{2.20}$$

für eine geeignete Norm $\|\cdot\|$ erfüllt. Dann gilt:

(a) g besitzt genau einen Fixpunkt $\hat{x}$ in D.

(b) Die Fixpunktiteration $x^{(i+1)} = g(x^{(i)})$, $i = 0, 1, 2, \ldots$ konvergiert für jeden Startwert $x^{(0)}$ aus D gegen den Fixpunkt $\hat{x}$.

(c) Es gilt die a-priori Fehlerabschätzung

$$\|x^{(i)} - \hat{x}\| \leq \frac{q^i}{1-q}\|x^{(1)} - x^{(0)}\|.$$

Beweis: Auf Grund der Selbstabbildungseigenschaft ist die Fixpunktiteration wohldefiniert.

(i) Existenz eines Fixpunkts:

Sei $x^{(0)} \in D$ beliebig. Wir zeigen, dass die Folge eine Cauchyfolge ist.

Die Kontraktionseigenschaft von g liefert für jedes $i \in \mathbb{N}$ die Beziehung

$$\|x^{(i+1)} - x^{(i)}\| \leq q\|x^{(i)} - x^{(i-1)}\| \leq \ldots \leq q^i\|x^{(1)} - x^{(0)}\|.$$

Mit Hilfe der Dreiecksungleichung folgt dann für jedes $k \in \mathbb{N}$ die Abschätzung

$$\begin{aligned}\|x^{(i+k)} - x^{(i)}\| &\leq \|x^{(i+k)} - x^{(i+k-1)}\| + \|x^{(i+k-1)} - x^{(i+k-2)}\| + \ldots + \|x^{(i+1)} - x^{(i)}\| \\ &\leq \left(q^{i+k-1} + q^{i+k-2} + \ldots + q^i\right)\|x^{(1)} - x^{(0)}\| \\ &\leq \frac{q^i}{1-q}\|x^{(1)} - x^{(0)}\|.\end{aligned}$$

Wegen $q < 1$ ist $\{x^{(i)}\}$ eine Cauchyfolge in D. Da D abgeschlossen ist, konvergiert die Folge dann gegen ein $\hat{x} \in D$.

Gemäß der Kontraktionsbedingung ist g lipschitz-stetig und insbesondere stetig. Damit gilt

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

d.h. $\hat{x}$ ist Fixpunkt.

(ii) Eindeutigkeit:

Angenommen, es gäbe einen weiteren Fixpunkt $\tilde{x} \neq \hat{x}$ in D. Dann gilt

$$\|\tilde{x} - \hat{x}\| = \|g(\tilde{x}) - g(\hat{x})\| \leq q\|\tilde{x} - \hat{x}\|.$$

Wegen $q < 1$ kann dies nur für $\|\tilde{x} - \hat{x}\| = 0$ gelten im Widerspruch zu $\tilde{x} \neq \hat{x}$.

(iii) Die Fehlerabschätzung folgt mit (i) aus

$$\|x^{(i)} - \hat{x}\| \leq \|x^{(i+1)} - \hat{x}\| + \|x^{(i+1)} - x^{(i)}\| \leq q\|x^{(i)} - \hat{x}\| + q^i\|x^{(1)} - x^{(0)}\|.$$

□

Anwendung der Fixpunktiteration auf die Fixpunktgleichung (2.19) liefert das folgende iterative Verfahren zur Lösung des linearen Gleichungssystems (2.3):

Algorithmus 2.5.5 (Iteratives Lösungsverfahren für (2.3))

(i) Zerlege die Matrix A gemäß (2.17) mit einer invertierbaren Matrix M, wähle einen beliebigen Startvektor $x^{(0)}$ und setze $k = 0$.

(ii) Falls $r_k = \|b - Ax^{(k)}\|_2$ hinreichend klein, STOP.

(iii) Berechne

$$Mx^{(k+1)} = Nx^{(k)} + b \tag{2.21}$$

(iv) Setze $k := k + 1$ und gehe zu (ii).

Bemerkung 2.5.6

- *(2.21) ist unter den bisherigen Annahmen äquivalent mit (2.19).*
- *Konvergiert die Folge $\{x^{(k)}\}$ gegen $\hat{x}$, dann gilt $M\hat{x} = N\hat{x} + b = (M - A)\hat{x} + b = M\hat{x} - A\hat{x} + b$ und somit löst $\hat{x}$ das lineare Gleichungssystem $A\hat{x} = b$.*

Unter Anwendung des Banachschen Fixpunktsatzes lautet ein hinreichendes Konvergenzkriterium wie folgt.

Satz 2.5.7 (Hinreichendes Konvergenzkriterium)

Es gelte

$$\|T\|_M < 1 \tag{2.22}$$

bezüglich einer mit der Vektornorm $\|\cdot\|_V$ verträglichen Matrixnorm $\|\cdot\|_M$. Dann existiert genau ein Fixpunkt $\hat{x}$ der Funktion $g(x) = Tx + c$ und die Fixpunktiteration

$$x^{(i+1)} = Tx^{(i)} + c, \qquad i = 0, 1, 2, \ldots \tag{2.23}$$

konvergiert für jeden Startwert $x^{(0)} \in \mathbb{K}^n$ gegen $\hat{x}$. Insbesondere ist $\hat{x}$ Lösung des Gleichungssystems (2.3).

Beweis: Wegen $\|T\|_M < 1$ gilt

$$\|Tx + c - (Ty + c)\|_V = \|T(x - y)\|_V \leq \|T\|_M \cdot \|x - y\|_V$$

und die Kontraktionseigenschaft (2.20) ist nachgewiesen. Da die Abbildung $x \mapsto Tx + c$ affin linear ist, liegt eine Selbstabbildung mit $D = \mathbb{K}^n$ vor, so dass die Voraussetzungen des Banachschen Fixpunktsatzes erfüllt sind. Die Behauptungen ergeben sich aus demselben. □

Die bisherige Konvergenzaussage hängt noch von den gewählten Normen $\|\cdot\|_V$ und $\|\cdot\|_M$ ab und setzt voraus, dass solche Normen mit (2.22) gefunden werden können. Ein notwendiges und hinreichendes Kriterium für die Konvergenz von (2.23) liefert

Satz 2.5.8

Sei $A = M - N$ invertierbar und $T = M^{-1} \cdot N$. Dann konvergiert (2.23) genau dann für jeden Startwert $x^{(0)} \in \mathbb{R}^n$ gegen die Lösung $\hat{x}$ von (2.3), wenn für den Spektralradius $\rho(T)$ von T die Ungleichung $\rho(T) < 1$ gilt.

Es stellt sich noch die Frage, wie schnell die so konstruierten Näherungen $x^{(i)}$ gegen den Fixpunkt $\hat{x}$ konvergieren.
Wegen

$$\|x^{(i+1)} - \hat{x}\|_V = \|Tx^{(i)} + c - (T\hat{x} + c)\|_V = \|T(x^{(i)} - \hat{x})\|_V \leq \|T\|_M \cdot \|x^{(i)} - \hat{x}\|_V$$

mit $\|T\|_M < 1$ erhalten wir i.a. nur eine sogenannte **lineare Konvergenzgeschwindigkeit** für die Fixpunktiteration, da sich der Fehler im schlechtesten Fall in jeder Iteration nur um den Faktor $\|T\|_M$ verringert. Insbesondere kann die Konvergenz für $\|T\|_M \approx 1$ sehr langsam sein.

2.5.1 Spezielle Verfahren

Es werden spezielle Verfahren durch geeignete Wahl von M und N konstruiert. Die Fixpunktiteration (2.23) ist besonders dann günstig, falls M leicht invertiert werden kann und die auftretenden Multiplikationen $M^{-1}N$ und $M^{-1}b$ billig sind, wie es häufig bei dünn besetzten Matrizen der Fall ist.
Wir zerlegen A in

$$A = D - L - R$$

wobei D die Diagonalelemente von A, L die Elemente unterhalb der Diagonalen von $-A$ und R die Elemente oberhalb der Diagonalen von $-A$ enthalten.

2.5.1.1 Gesamtschrittverfahren (Jacobi-Verfahren)

Wir wählen in (2.17)

$$M = D, \quad N = L + R$$

und erhalten den **Gesamtschrittoperator**

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

Die Fixpunktiteration (2.23) für das Gesamtschrittverfahren lautet somit

$$x^{(i+1)} = D^{-1}\left((L+R)x^{(i)} + b\right). \tag{2.24}$$

Die Inverse von $D = diag(a_{11}, a_{22}, \ldots, a_{nn})$ ist sehr leicht zu berechnen, da D eine Diagonalmatrix ist. Es ist $D^{-1} = diag(1/a_{11}, 1/a_{22}, \ldots, 1/a_{nn})$. Um das Verfahren durchführen zu können, müssen die Diagonalelemente ungleich Null sein.

Algorithmus 2.5.9 **(Jacobi-Verfahren)**

(i) Wähle $M = diag(a_{11}, a_{22}, \ldots, a_{nn})$.

(ii) Löse (2.21) bzgl. $x^{(i+1)}$*:*

$$x_j^{(i+1)} = \frac{1}{a_{jj}}\left(b_j - \sum_{k\neq j} a_{jk}x_k^{(i)}\right), \qquad j = 1, \ldots, n.$$

Ein einfach überprüfbares Kriterium zur Konvergenz des Verfahrens liefert der folgende Satz.

Satz 2.5.10 **(Konvergenz des Gesamtschrittverfahrens)**
Die Matrix $A \in \mathbb{K}^{n\times n}$ *sei* **strikt diagonaldominant**, *d.h. es gelte*

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

Dann konvergiert das Gesamtschrittverfahren für jeden Startvektor $x^{(0)} \in \mathbb{R}^n$ *gegen die Lösung* $\hat{x}$ *von* $Ax = b$ *und es gilt die Fehlerabschätzung*

$$\|x^{(i)} - \hat{x}\|_\infty \leq \frac{\|T\|_\infty^i}{1 - \|T\|_\infty}\|x^{(1)} - x^{(0)}\|_\infty, \quad i = 1, 2, \ldots,$$

wobei $T = D^{-1}(L+R)$ *den Gesamtschrittoperator bezeichnet.*

Beweis: Die strikte Diagonaldominanz impliziert $\|T\|_\infty < 1$, wobei $\|\cdot\|_\infty$ die Zeilensummennorm bezeichnet. Die Konvergenz folgt dann aus Satz 2.5.7. Die Fehlerabschätzung folgt aus dem Banach'schen Fixpunktsatz. □

Ein analoger Satz gilt für die Spaltensummennorm $\|\cdot\|_1$, wenn gefordert wird, dass

$$|a_{jj}| > \sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}|, \qquad j = 1, \ldots, n.$$

2.5.1.2 Einzelschrittverfahren (Gauß-Seidel-Verfahren)

Wir wählen in (2.17)

$$M = D - L, \quad N = R$$

und erhalten den **Einzelschrittoperator**

$$T = (D - L)^{-1} R.$$

Die Fixpunktiteration (2.23) für das Einzelschrittverfahren lautet somit

$$x^{(i+1)} = (D - L)^{-1} \left(R x^{(i)} + b \right). \tag{2.25}$$

Algorithmus 2.5.11 **(Gauß-Seidel-Verfahren)**

(i) Wähle M als den unteren Dreiecksanteil von A, einschließlich Diagonale.

(ii) Löse (2.21) bzgl. $x^{(i+1)}$ mittels Vorwärtssubstitution:

$$x_j^{(i+1)} = \frac{1}{a_{jj}} \left(b_j - \sum_{k=1}^{j-1} a_{jk} x_k^{(i+1)} - \sum_{k=j+1}^{n} a_{jk} x_k^{(i)} \right), \qquad j = 1, \ldots, n.$$

Man beachte, dass die auf der rechten Seite auftretenden Komponenten $x_k^{(i+1)}$, $k < j$ der neuen Iterierten bereits zur Berechnung von $x_j^{(i+1)}$ verwendet werden.
Ein einfach überprüfbares Kriterium zur Konvergenz des Verfahrens liefert der folgende Satz.

Satz 2.5.12 **(Konvergenz des Einzelschrittverfahrens)**
Die Matrix $A \in \mathbb{K}^{n \times n}$ erfülle das **Sassenfeld-Kriterium**:

(i) Sämtliche Diagonalelemente von A sind $\neq 0$.

(ii) Es gilt $\beta = \max\limits_{i=1,\ldots,n} \beta_i < 1$ mit

$$\begin{aligned} \beta_1 &= \frac{1}{|a_{11}|} \sum_{j=2}^{n} |a_{1j}|, \\ \beta_i &= \frac{1}{|a_{ii}|} \left(\sum_{j=1}^{i-1} |a_{ij}| \beta_j + \sum_{j=i+1}^{n} |a_{ij}| \right), \quad i = 2, \ldots, n. \end{aligned}$$

Dann konvergiert das Einzelschrittverfahren für jeden Startvektor $x^{(0)} \in \mathbb{K}^n$ gegen die Lösung $\hat{x}$ von $Ax = b$ und es gilt die Fehlerabschätzung

$$\|x^{(i)} - \hat{x}\|_\infty \leq \frac{\beta^i}{1 - \beta} \|x^{(1)} - x^{(0)}\|_\infty, \quad i = 1, 2, \ldots.$$

Beweis: Wir zeigen, dass $\|T\|_\infty \leq \beta$ für den Einzelschrittoperator $T = (D-L)^{-1}R$ gilt.

Sei $x \in \mathbb{K}^n$ mit $\|x\|_\infty \leq 1$ und $y = Tx$, d.h. $(D-L)y = Rx$. Dann folgt

$$a_{11}y_1 = -\sum_{j=2}^{n} a_{1j}x_j \quad \Rightarrow \quad |y_1| \leq \frac{1}{|a_{11}|}\sum_{j=2}^{n}|a_{1j}| = \beta_1.$$

Die Behauptung folgt nun per Induktion. Sei also $|y_i| \leq \beta_i$ für $k = 1, 2, \dots, j-1$ bewiesen. Dann folgt

$$\sum_{k=1}^{j-1} a_{jk}y_k + a_{jj}y_j = -\sum_{k=j+1}^{n} a_{jk}x_k \quad \Rightarrow \quad |y_j| \leq \frac{1}{|a_{jj}|}\left(\sum_{k=1}^{j-1}|a_{jk}|\beta_k + \sum_{k=j+1}^{n}|a_{jk}|\right) = \beta_j.$$

Somit gilt also $|y_i| \leq \beta_i$ für $i = 1, 2, \dots, n$. Damit gilt $\|T\|_\infty = \sup_{\|x\|_\infty \leq 1} \|Tx\|_\infty \leq \beta$. Die Konvergenz folgt dann aus Satz 2.5.7. Die Fehlerabschätzung folgt aus dem Banach'schen Fixpunktsatz. □

Beispiel 2.5.13 (Diskretisierung einer partiellen Differentialgleichung, Teil II)
Wir betrachten wiederum Beispiel 2.5.1 mit $N = 50$ *und*

$$f(x,y) = 1000 \cdot \sin((x-0.5)\cdot(y-0.5)).$$

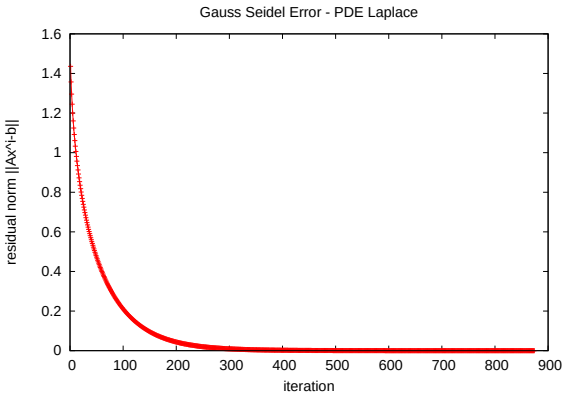
Für $N = 50$ *hat* A *im Gleichungssystem* $Au = b$ *die Dimension* $49^2 = 2401$ *und von den* $2401^2 = 5764801$ *Einträgen sind lediglich* 11809 *Einträge von Null verschieden.*

Wir lösen das Gleichungssystem mit dem Gauß-Seidel-Verfahren, wählen $u^{(0)} = 0$ *als Startwert und brechen die Iteration ab, wenn das relative Residuum*

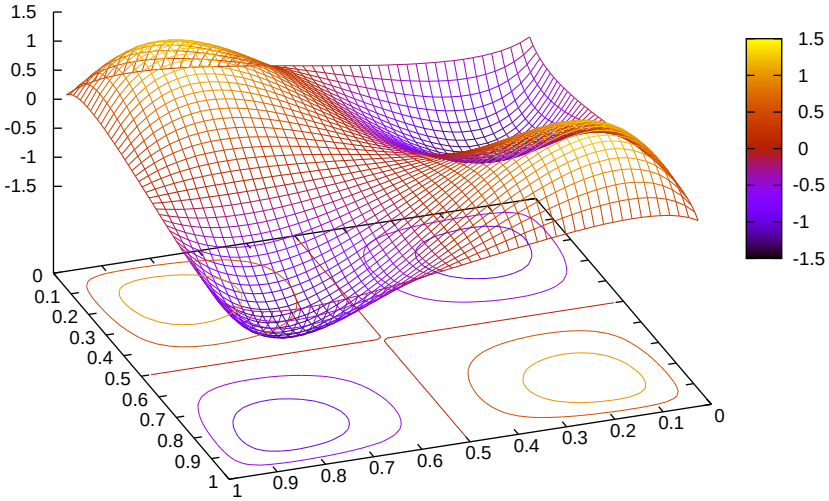
$$\frac{r_k}{r_0} = \frac{\|b - Au^{(k)}\|_2}{\|b - Au^{(0)}\|_2}, \quad k = 0, 1, 2, \dots.$$

kleiner oder gleich der Toleranz 10^{-5} *ist.*

```
ITER  RESIDUUM
-------------
 1  1.43604533
 2  1.35716494
 3  1.29587804
 4  1.24465554
 5  1.20013794
 6  1.16048560
 7  1.12456585
...
 870  1.61266564E-05
 871  1.60113515E-05
 872  1.58971717E-05
 873  1.57841049E-05
 874  1.56721393E-05
 ITERATION SUCCESSFUL:
 INITIAL ERROR:   1.56259882
 FINAL ERROR:     1.55612629E-05
 ITERATIONS:     875
```



Obige Abbildung zeigt das Residuum $\|b - Ax\|_2$ der Iterierten.
Die folgende Abbildung zeigt die mit dem Gauß-Seidel-Verfahren berechnete numerische Lösung u der partiellen Differentialgleichung:



Beispiel 2.5.14

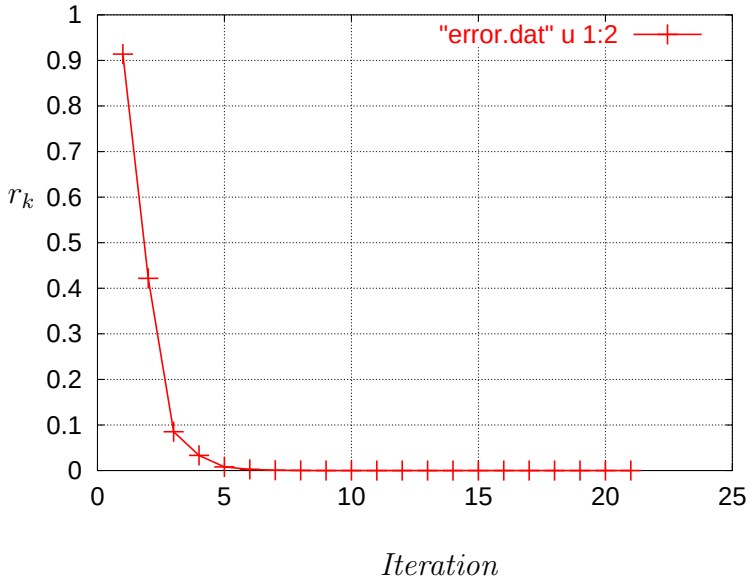
Wir lösen das lineare Gleichungssystem

$$\begin{pmatrix} 2 & -1 & 0 \\ 1 & 6 & -2 \\ 4 & -3 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -4 \\ 5 \end{pmatrix}$$

mit dem Gauß-Seidel-Verfahren und erhalten den folgenden Output:

```
IT  NORM RESIDUUM
---------------
```

```
1   0.913821342
2   0.421932152
3   0.0851292437
4   0.033203378
5   0.00789585657
6   0.00256967289
7   0.000723273381
8   0.000195688235
9   6.52208342E-05
10  1.46979182E-05
11  5.78429145E-06
12  1.09568644E-06
13  5.04543055E-07
14  8.21821862E-08
15  4.32969302E-08
16  6.34648404E-09
17  3.65635959E-09
18  5.17553689E-10
19  3.03904484E-10
20  4.49659492E-11
21  2.48600765E-11
ITERATION SUCCESSFUL:
INITIAL ERROR:   6.70820393
FINAL ERROR:     4.09426575E-12
ITERATIONS:     22
APPROXIMATE SOLUTION:   0.62 -0.76  0.03
```



Die Abbildung zeigt das Residuum

$$r_k = \|b - Ax^{(k)}\|_2^2, \quad k = 0, 1, 2, \ldots.$$

Man erkennt sehr gut, dass der Fortschritt in den ersten Iterationen sehr groß ist, die Konvergenz des Verfahrens am Ende allerdings sehr langsam ist.

Häufig beobachtet man in der Praxis, dass das Einzelschrittverfahren schneller konvergiert als das Gesamtschrittverfahren.

2.6 Das konjugierte Gradientenverfahren

Ein gänzlich anderer Ansatz zur Lösung des linearen Gleichungssystems (2.3) wird beim **konjugierten Gradientenverfahren** verfolgt. Wir beschränken uns hier auf den reellen Fall $\mathbb{K} = \mathbb{R}$. An Stelle des Gleichungssystems $Ax = b$ wird das Minimierungsproblem

$$\text{Minimiere } f(x) = \frac{1}{2} x^\top Ax - b^\top x \tag{2.26}$$

betrachtet. Eine Lösung des Minimierungsproblems ist stets auch Lösung von (2.3), denn für ein Minimum von f ist die Bedingung

$$0 = \nabla f(x) = Ax - b$$

notwendig. Ist A positiv definit und symmetrisch, so sind beide Problemstellungen äquivalent. Für symmetrische, positiv definite Matrizen gilt das konjugierte Gradientenverfahren als eines der effizientesten Verfahren.
Das Verfahren arbeitet ebenfalls iterativ, lässt sich aber nicht als Fixpunktiteration darstellen. Im folgenden wird vorausgesetzt, dass A symmetrisch und positiv definit ist.

Ausgehend von einer Startschätzung $x^{(0)}$ wird die nächste Iterierte

$$x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)}, \quad i = 0, 1, 2, \ldots$$

mit Hilfe einer **Suchrichtung** $d^{(i)}$ und einer **Schrittweite** α_i berechnet. Dabei werden Suchrichtung und Schrittweite so bestimmt, dass die Zielfunktion in jeder Iteration abnimmt.

2.6.1 Bestimmung der Schrittweite

Für eine gegebene Suchrichtung $d^{(i)}$ wird die Schrittweite α_i bestimmt, indem f in Richtung $d^{(i)}$ minimiert wird, vgl. Abbildung 2.1.

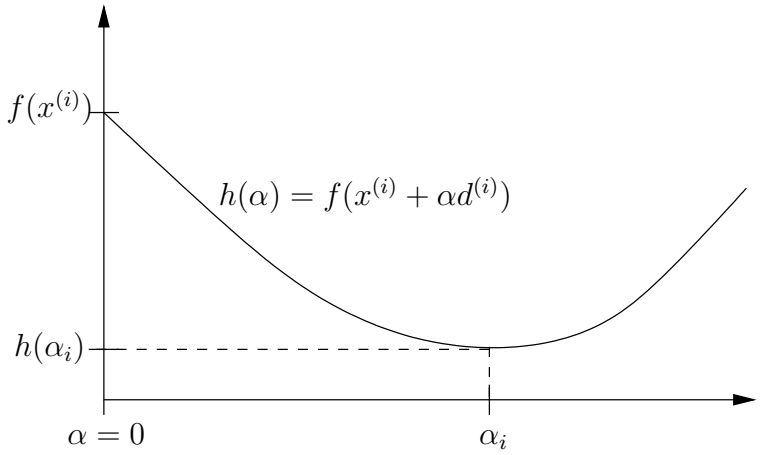


Abbildung 2.1: Liniensuche zur Bestimmung der Schrittweite α_i.

Dieser Vorgang wird **Liniensuche** genannt. Die Aufgabe besteht also darin, die Funktion

$$h(\alpha) := f(x^{(i)} + \alpha d^{(i)})$$

bezüglich α zu minimieren. Dies ist eine eindimensionale Optimierungsaufgabe mit eindeutiger Lösung, da A positiv definit ist. Berücksichtigt man die Struktur von f, kann man die optimale Schrittweite

$$\alpha_i = \frac{(b - Ax^{(i)})^\top d^{(i)}}{(d^{(i)})^\top A d^{(i)}} \tag{2.27}$$

ausrechnen, wobei der Nenner wegen der positiven Definitheit von A ungleich Null ist.

2.6.2 Bestimmung der Suchrichtung

Im ersten Schritt des Verfahrens wird die Richtung des steilsten Abstiegs verwendet. Bekanntlich zeigt der Gradient einer Funktion in Richtung des steilsten Anstiegs der

Funktion. Der negative Gradient zeigt somit in Richtung des steilsten Abstiegs. Daher wird

$$d^{(0)} = -\nabla f(x^{(0)}) = -(Ax^{(0)} - b) = b - Ax^{(0)}$$

als erste Suchrichtung gewählt. Die weiteren Suchrichtungen werden so konstruiert, dass sie bzgl. des Skalarprodukts $\langle x, y\rangle_A := x^\top Ay$ orthogonal zu den bisherigen Suchrichtungen sind.

Definition 2.6.1

Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Die Vektoren $s_1, \dots, s_m \in \mathbb{R}^n$, $s_i \neq 0$, $i = 1, \dots, m$, heißen **A-orthogonal** *(oder* **A-konjugiert***), falls gilt*

$$\langle s_i, s_j\rangle_A = 0 \quad \textit{für} \quad i \neq j.$$

Beachte, dass A-orthogonale Richtungen linear unabhängig sind, denn

$$\sum_{i=1}^m \alpha_i s_i = 0 \quad \Rightarrow \quad 0 = s_k^\top A^\top \left(\sum_{i=1}^m \alpha_i s_i\right) = \alpha_k \underbrace{s_k^\top A^\top s_k}_{>0} \quad \Rightarrow \quad \alpha_k = 0.$$

2.6.3 Konvergenz

Damit ist das Verfahren vollständig beschrieben. Insbesondere bilden n A-orthogonale Richtungen eine Basis des $\mathbb{R}^n$ und jeder Punkt des $\mathbb{R}^n$ kann als Linearkombination von (maximal) n A-orthogonalen Richtungen dargestellt werden. Insbesondere besitzt auch das Minimum $\hat{x}$ der Funktion f eine solche Darstellung. Der folgende Satz zeigt, dass nach spätestens n Schritten das Minimum gefunden wird.

Satz 2.6.2

Sei f gegeben durch (2.26) und $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Die Vektoren $d^{(0)}, \dots, d^{(n-1)}$ seien A-orthogonal und $x^{(0)} \in \mathbb{R}^n$ sei beliebig. Desweiteren gelte

$$x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)}, \qquad i = 0, \dots, n-1$$

mit exakter Schrittweite gemäß (2.27). Dann gilt

$$f(x^{(n)}) = \min_{x\in\mathbb{R}^n} f(x).$$

Beweis: (vgl. Jarre und Stoer [JS04])

Jeder Vektor $v \in \mathbb{R}^n$ lässt sich darstellen als

$$v = \sum_{j=0}^{n-1} \gamma_j d^{(j)} \quad \Rightarrow \quad \left(d^{(i)}\right)^\top Av = \gamma_i \underbrace{\left(d^{(i)}\right)^\top Ad^{(i)}}_{>0} \quad \Rightarrow \quad \gamma_i = \frac{\left(d^{(i)}\right)^\top Av}{\left(d^{(i)}\right)^\top Ad^{(i)}}.$$

Damit folgt

$$v = \sum_{j=0}^{n-1} \frac{\left(d^{(j)}\right)^\top A v}{\left(d^{(j)}\right)^\top A d^{(j)}} d^{(j)}. \tag{2.28}$$

Nach Wahl von α_i gemäß (2.27) folgt

$$\begin{aligned}
\alpha_i &= \frac{\left(b - Ax^{(i)}\right)^\top d^{(i)}}{\left(d^{(i)}\right)^\top A d^{(i)}} \\
&= \frac{\left(b - A\left(x^{(0)} + \sum_{j=0}^{i-1} \alpha_j d^{(j)}\right)\right)^\top d^{(i)}}{\left(d^{(i)}\right)^\top A d^{(i)}} \\
&= \frac{\left(b - Ax^{(0)}\right)^\top d^{(i)} - \sum_{j=0}^{i-1} \alpha_j \left(d^{(j)}\right)^\top A d^{(i)}}{\left(d^{(i)}\right)^\top A d^{(i)}} \\
&\overset{\langle d^{(j)}, d^{(i)}\rangle_A = 0}{=} -\frac{\left(d^{(i)}\right)^\top \left(Ax^{(0)} - b\right)}{\left(d^{(i)}\right)^\top A d^{(i)}}.
\end{aligned}$$

Wegen $x^{(n)} = x^{(0)} + \sum_{i=0}^{n-1} \alpha_i d^{(i)}$ folgt dann

$$x^{(n)} = x^{(0)} - \sum_{i=0}^{n-1} \frac{\left(d^{(i)}\right)^\top \left(Ax^{[0]} - b\right)}{\left(d^{(i)}\right)^\top A d^{(i)}} d^{(i)} = x^{(0)} - \sum_{i=0}^{n-1} \frac{\left(d^{(i)}\right)^\top A \overbrace{\left(x^{(0)} - A^{-1}b\right)}^{=:v}}{\left(d^{(i)}\right)^\top A d^{(i)}} d^{(i)}.$$

Definition von $v := x^{(0)} - A^{-1}b$ zusammen mit der Darstellung von v in (2.28) liefert

$$x^{(n)} = x^{(0)} - \left(x^{(0)} - A^{-1}b\right) = A^{-1}b = \arg\min_{x \in \mathbb{R}^n} f(x).$$

□

Der Satz liefert mit anderen Worten die Darstellung

$$\hat{x} = x^{(0)} + \sum_{k=0}^{n-1} \alpha_k d^{(k)}$$

für eine Lösung $\hat{x}$ von $Ax = b$.
Es bleibt die Frage, wie A-orthogonale Richtungen berechnet werden können.
Wir erzeugen die Richtungen schrittweise und achten darauf, dass Abstiegsrichtungen entstehen (d.h. in diese Richtungen soll die Zielfunktion fallen). Zur Abkürzung sei ab jetzt

$$g^{(i+1)} := \nabla f(x^{(i+1)}) = Ax^{(i+1)} - b = A(x^{(i)} + \alpha_i d^{(i)}) - b. \tag{2.29}$$

Wir starten mit $d^{(0)} = -\nabla f(x^{(0)}) = -g^{(0)}$ und gehen davon aus, dass bereits A-orthogonale Richtungen $d^{(0)}, \ldots, d^{(i)}$ vorliegen. Zusätzlich sei $g^{(j)} \neq 0$, $0 \leq j \leq i+1$ (ansonsten wären

wir fertig). Die Schrittweite α_i sei durch (2.27) gegeben und kann geschrieben werden als

$$\alpha_i = -\frac{(g^{(i)})^\top d^{(i)}}{(d^{(i)})^\top A d^{(i)}}. \tag{2.30}$$

Durch Einsetzen von α_i und $g^{(i+1)}$ zeigt man leicht, dass

$$(g^{(i+1)})^\top d^{(i)} = \left(Ax^{(i)} + \alpha_i A d^{(i)} - b\right)^\top d^{(i)} = 0. \tag{2.31}$$

(i) Aus $g^{(i+1)} - g^{(i)} = \alpha_i A d^{(i)}$ folgt für $j < i$ die Beziehung

$$\left(g^{(i+1)} - g^{(i)}\right)^\top d^{(j)} = \alpha_i (d^{(i)})^\top A d^{(j)} = 0 \tag{2.32}$$

und damit auch

$$(g^{(i+1)})^\top d^{(j)} \quad = \quad (g^{(j+1)})^\top d^{(j)} + \sum_{k=j+1}^{i} \left(g^{(k+1)} - g^{(k)}\right)^\top d^{(j)}$$
$$\overset{(2.31),(2.32)}{=} 0.$$

Insgesamt folgt also

$$(g^{(i+1)})^\top d^{(j)} = 0 \qquad \forall j \le i. \tag{2.33}$$

(ii) Gilt $g^{(i+1)} \neq 0$, so bedeutet (2.33), dass

$$g^{(i+1)} \perp \operatorname{span}(d^{(0)}, \dots, d^{(i)}). \tag{2.34}$$

Wir wählen den Ansatz

$$d^{(i+1)} = -g^{(i+1)} + \sum_{j=0}^{i} \frac{\langle d^{(j)}, g^{(i+1)} \rangle_A}{\langle d^{(j)}, d^{(j)} \rangle_A} d^{(j)}. \tag{2.35}$$

In der Tat folgt aus der A-Orthogonalität von $\{d^{(0)}, \dots, d^{(i)}\}$ die Beziehung $\langle d^{(i+1)}, d^{(j)} \rangle_A = 0$ für $j = 0, \dots, i$. Somit ist auch $\{d^{(0)}, \dots, d^{(i+1)}\}$ A-orthogonal.

Zudem ist $d^{(i+1)}$ auch Abstiegsrichtung, denn Multiplikation von links mit $(g^{(i+1)})^\top$ liefert zusammen mit (2.33) die Beziehung

$$(g^{(i+1)})^\top d^{(i+1)} = -\|g^{(i+1)}\|^2 < 0. \tag{2.36}$$

Beachte, dass (2.30) $\alpha_{i+1} > 0$ liefert.

(iii) Wir zeigen, dass in (2.35) alle Summanden mit $j < i$ verschwinden. Wegen (2.35) ist

$$g^{(i+1)} \in \operatorname{span}(d^{(0)}, \dots, d^{(i+1)}) \quad \text{bzw.} \quad g^{(j)} \in \operatorname{span}(d^{(0)}, \dots, d^{(j)}),\ 0 \le j \le i+1.$$

Wegen (2.34) folgt

$$(g^{(i+1)})^\top g^{(j)} = 0 \qquad \forall j \le i. \tag{2.37}$$

Für $j < i$ folgt dann

$$(g^{(i+1)})^\top A d^{(j)} = \frac{1}{\alpha_j}(g^{(i+1)})^\top \left(g^{(j+1)} - g^{(j)}\right) = 0.$$

(2.35) reduziert sich somit zu

$$d^{(i+1)} = -g^{(i+1)} + \frac{\langle d^{(i)}, g^{(i+1)}\rangle_A}{\langle d^{(i)}, d^{(i)}\rangle_A} d^{(i)}. \tag{2.38}$$

Schließlich folgt mit $Ad^{(i)} = \frac{1}{\alpha_i}\left(g^{(i+1)} - g^{(i)}\right)$:

$$\begin{aligned}
(g^{(i+1)})^\top A d^{(i)} &= \frac{1}{\alpha_i}\left(\|g^{(i+1)}\|^2 - (g^{(i+1)})^\top g^{(i)}\right) \\
&\overset{(2.37)}{=} \frac{1}{\alpha_i}\|g^{(i+1)}\|^2, \\
(d^{(i)})^\top A d^{(i)} &\overset{(2.30)}{=} -\frac{1}{\alpha_i}(g^{(i)})^\top d^{(i)} \\
&\overset{(2.36)}{=} \frac{1}{\alpha_i}\|g^{(i)}\|^2.
\end{aligned}$$

Einsetzen in (2.38) liefert

$$d^{(i+1)} = -g^{(i+1)} + \frac{\|g^{(i+1)}\|^2}{\|g^{(i)}\|^2} d^{(i)}. \tag{2.39}$$

Die obigen Betrachtungen zusammen mit Satz 2.6.2 und der Tatsache, dass maximal n A-orthogonale Richtungen existieren, beweisen

Satz 2.6.3

Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann endet das obige Verfahren für einen beliebigen Startwert $x^{(0)} \in \mathbb{R}^n$ nach $m \le n$ Schritten mit $Ax^{(m)} = b$.
Für das kleinste $m \le n$ mit $Ax^{(m)} = b$ gelten für alle $l \le m$ die Orthogonalitätsrelationen:

$$\begin{aligned}
(d^{(i)})^\top A d^{(j)} &= 0, \qquad 0 \le j < i \le l, \\
(g^{(i)})^\top g^{(j)} &= 0, \qquad 0 \le j < i \le l, \ \textit{vgl. (2.37)} \\
(g^{(i)})^\top d^{(j)} &= 0, \qquad 0 \le j < i \le l, \ \textit{vgl. (2.33)} \\
(g^{(i)})^\top d^{(i)} &= -\|g^{(i)}\|^2, \qquad 0 \le i \le l, \textit{vgl. (2.36)}.
\end{aligned}$$

Die obigen Betrachtungen liefern ein konstruktives Verfahren zur Berechnung der A-orthogonalen Suchrichtungen. Zusammen mit Satz 2.6.2 folgt daraus die Endlichkeit des

folgenden Algorithmus:

Algorithmus 2.6.4 (CG-Verfahren)

(i) *Wähle einen Startvektor* $x^{(0)} \in \mathbb{R}^n$, $\varepsilon \geq 0$, *berechne* $g^{(0)} = Ax^{(0)} - b$ *und setze* $d^{(0)} = -g^{(0)}$, $i = 0$.

(ii) *Falls* $\|g^{(i)}\| \leq \varepsilon$, *STOP.*

(iii) *Berechne*

$$\begin{aligned} \alpha_i &= \frac{\|g^{(i)}\|^2}{(d^{(i)})^\top A d^{(i)}}, \\ x^{(i+1)} &= x^{(i)} + \alpha_i d^{(i)}, \\ g^{(i+1)} &= g^{(i)} + \alpha_i A d^{(i)}, \\ \beta_i &= \frac{\|g^{(i+1)}\|^2}{\|g^{(i)}\|^2}, \\ d^{(i+1)} &= -g^{(i+1)} + \beta_i d^{(i)}. \end{aligned}$$

(iv) *Setze* $i := i + 1$, *und gehe zu (ii).*

In der Praxis führt man häufig aufgrund von auftretenden Rundungsfehlern nach einer gewissen Anzahl von Iterationen einen Restart des Verfahrens durch.

Beispiel 2.6.5 (Diskretisierung einer partiellen Differentialgleichung, Teil III)
Wir betrachten wiederum Beispiel 2.5.1 mit $N = 100$ *und*

$$f(x, y) = 1000 \cdot \sin((x - 0.5) \cdot (y - 0.5)).$$

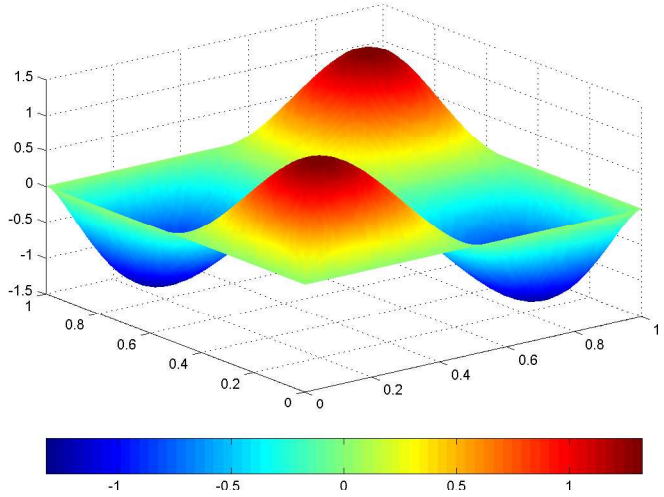
Die partielle Differentialgleichung wird wie zuvor diskretisiert und führt auf das lineare Gleichungssystem $Au = b$. *Anschließend wird das äquivalente Problem*

$$\frac{1}{2} u^\top A u - b^\top u \to \min$$

mit dem CG-Verfahren gelöst. Dabei werden nur die von Null verschiedenen Einträge von A *mit einer entsprechenden Indizierung gespeichert, so dass Multiplikationen* Ad *sehr effizient durchgeführt werden können. Für* $N = 100$ *hat* A *die Dimension* $99^2 = 9801$ *und von den* $9801^2 = 96059601$ *Einträgen sind lediglich* 48609 *Einträge von Null verschieden. Als Startwert wählen wir* $u^{(0)} = 0$. *Für* $N = 100$ *und* $\varepsilon = 10^{-8}$ *ergibt sich folgende Ausgabe:*

ITER	||GRAD||	||DX||	ALPHA	BETA
0	0.8055956E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
1	0.2271664E+01	0.6796640E+01	0.8287775E+01	0.7951592E+01
3	0.2274125E+01	0.7564077E+01	0.7990598E+00	0.1092265E+01
5	0.2100902E+01	0.7055373E+01	0.5045940E+00	0.8358480E+00
7	0.1932038E+01	0.6517664E+01	0.4768494E+00	0.8424729E+00
9	0.1767712E+01	0.5963891E+01	0.4706184E+00	0.8557311E+00
11	0.1611152E+01	0.5421028E+01	0.4690171E+00	0.8656038E+00
13	0.1463812E+01	0.4902665E+01	0.4686974E+00	0.8722568E+00
15	0.1326138E+01	0.4414901E+01	0.4687291E+00	0.8765485E+00
17	0.1198085E+01	0.3960077E+01	0.4688082E+00	0.8791410E+00
19	0.1079372E+01	0.3538615E+01	0.4688279E+00	0.8804796E+00
21	0.9696041E+00	0.3149930E+01	0.4687504E+00	0.8808559E+00
23	0.8683385E+00	0.2792889E+01	0.4685635E+00	0.8804600E+00
25	0.7751144E+00	0.2466078E+01	0.4682637E+00	0.8794144E+00
...				
135	0.3018280E-06	0.5503303E-06	0.4188414E+00	0.6769555E+00
137	0.2309873E-06	0.4445599E-06	0.4462299E+00	0.7570287E+00
139	0.1747984E-06	0.3426776E-06	0.4463246E+00	0.7220239E+00
141	0.1093353E-06	0.1928130E-06	0.4093333E+00	0.6062993E+00
143	0.6981008E-07	0.1201987E-06	0.4214496E+00	0.6967370E+00
145	0.5219784E-07	0.9674677E-07	0.4486677E+00	0.7779052E+00
147	0.3285968E-07	0.5685132E-07	0.3985365E+00	0.6319190E+00
149	0.2182741E-07	0.3802979E-07	0.4169753E+00	0.7148059E+00
151	0.1323888E-07	0.2172648E-07	0.3884909E+00	0.5765313E+00
153	0.8588212E-08	0.1449120E-07	0.4170621E+00	0.7137157E+00

Die folgende Abbildung zeigt die numerische Lösung u der partiellen Differentialgleichung:



Bemerkung 2.6.6

Kelley [Kel95], Th. 2.2.3, S. 15, zeigt, dass das CG-Verfahren das Minimum in höchstens k Schritten liefert, wobei k die Anzahl der verschiedenen Eigenwerte von A bezeichnet.

Obwohl das CG-Verfahren (zumindest theoretisch) nach maximal n Schritten terminiert, ist es in der Praxis aus Aufwandsgründen oft nicht möglich, n Iterationen tatsächlich durchzuführen (n kann sehr groß sein, etwa 1 Milliarde).
Daher ist es sinnvoll, das CG-Verfahren als iteratives Verfahren zu betrachten. Es stellt sich dann die Frage nach der Konvergenzgeschwindigkeit. Es lässt sich folgende Abschätzung beweisen (vgl. Kelley [Kel95, S. 17]:

$$\|x^{(i)} - \hat{x}\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^i \|x^{(0)} - \hat{x}\|_A, \tag{2.40}$$

wobei $\kappa(A) = \lambda_{max}/\lambda_{min}$ die Kondition der Matrix A bzgl. der Norm $\|\cdot\|_2$ bezeichnet.
Die Fehlerabschätzung zeigt, dass das Verfahren umso schneller konvergiert, je näher die

Kondition $\kappa(A)$ bei Eins liegt. Umgekehrt ist die Konvergenz umso langsamer, je größer die Kondition von A ist. Es ist daher erstrebenswert, eine möglichst gut konditionierte Matrix A zu haben, was durch die sogenannte **Vorkonditionierung** (oder **Präkonditionierung**) erreicht werden kann.
Hierbei wird eine Variablentransformation durchgeführt:

$$\begin{aligned} x &:= Sz, \quad S \in \mathbb{R}^{n\times n} \text{ regulär}, \\ \hat{f}(z) &:= f(Sz) = \frac{1}{2} z^\top S^\top A S z - (S^\top b)^\top z. \end{aligned}$$

Natürlich sollte S so gewählt werden, dass die Kondition von $S^\top AS$ kleiner ist als die von A. Nun wenden wir das übliche CG-Verfahren auf $\hat{f}$ an und erhalten die Größen

$$\hat{x}^{(i)}, \quad \hat{g}^{(i)}, \quad \hat{d}^{(i)}, \quad \hat{\alpha}_i, \quad \hat{\beta}_i.$$

Rücksubstitution liefert

$$x^{(i)} = S\hat{x}^{(i)}, \quad S^\top g^{(i)} = \hat{g}^{(i)}, \quad d^{(i)} = S\hat{d}^{(i)}.$$

Unter Berücksichtigung der Rücksubstitution ergibt sich:

Algorithmus 2.6.7 (Präkonditioniertes CG-Verfahren)

(i) *Wähle eine symmetrische und positiv definite Matrix $B \in \mathbb{R}^{n\times n}$, einen Startvektor $x^{(0)} \in \mathbb{R}^n$, $\varepsilon \geq 0$, berechne $g^{(0)} = Ax^{(0)} - b$ und setze $d^{(0)} = -Bg^{(0)}$, $i = 0$.*

(ii) *Falls $\|g^{(i)}\| \leq \varepsilon$, STOP.*

(iii) *Berechne*

$$\begin{aligned} \hat{\alpha}_i &= \frac{(g^{(i)})^\top B g^{(i)}}{(d^{(i)})^\top A d^{(i)}}, \\ x^{(i+1)} &= x^{(i)} + \hat{\alpha}_i d^{(i)}, \\ g^{(i+1)} &= g^{(i)} + \hat{\alpha}_i A d^{(i)}, \\ \hat{\beta}_i &= \frac{(g^{(i+1)})^\top B g^{(i+1)}}{(g^{(i)})^\top B g^{(i)}}, \\ d^{(i+1)} &= -Bg^{(i+1)} + \hat{\beta}_i d^{(i)}. \end{aligned}$$

(iv) *Setze $i := i + 1$, und gehe zu (ii).*

Im Algorithmus tritt nur noch die symmetrische und positiv definite Matrix $B = SS^\top$ auf. Wir wählen B als Approximation an A^{-1}. Dann ist $BA \approx I$. Die Eigenwerte von BA liegen dann nahe bei 1. Wegen

$$S^{-1}BAS = S^{-1}SS^\top AS = S^\top AS$$

ist BA ähnlich zu $S^\top AS$ und folglich besitzen BA und $S^\top AS$ dieselben Eigenwerte. Da $S^\top AS$ symmetrisch ist, gilt die Formel $\kappa_2(S^\top AS) = \lambda_{max}/\lambda_{min}$, wobei λ_{max} und λ_{min} den betragsmäßig größten bzw. kleinsten Eigenwert von $S^\top AS$ bezeichnen. Da die Eigenwerte nahe bei 1 liegen ist auch die Kondition nahe bei 1.
Desweiteren sollte B so gewählt werden, dass die Multiplikationen $g \mapsto Bg$ kostengünstig ausgewertet werden können.
Mögliche Ansätze für B sind:

(a) $B = D^{-1}$, wobei D die Diagonale von A bezeichnet.

(b) Berechne die Cholesky-Zerlegung $A = L \cdot L^\top$, approximiere L durch Weglassen kleiner Elemente durch $\hat{L}$ und setze $B = \hat{L}^{-\top}\hat{L}^{-1}$. Dann gilt $BA = \hat{L}^{-\top}\hat{L}^{-1}LL^\top \approx I$. Dieses Verfahren ist brauchbar für dünn besetzte Matrizen.

Kapitel 3

Lineare Ausgleichsprobleme

Häufig gilt es, über- oder unterbestimmte Gleichungssysteme zu lösen.

Beispiel 3.0.1 (Lineares Ausgleichsproblem)

Die Messung eines physikalischen Vorgangs hat an den Ortspunkten $p_i = (p_{1,i}, p_{2,i})$, $i = 1, \ldots, 9$, *die folgenden Messwerte* y_i, $i = 1, \ldots, 9$, *ergeben:*

Messpunkt $i =$	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
$p_{1,i}$	*6*	*7*	*9*	*11*	*13*	*15*	*17*	*18*	*19*
$p_{2,i}$	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
y_i	*96*	*189*	*283*	*373*	*467*	*553*	*647*	*733*	*832*

Es wird ein affin-linearer Zusammenhang zwischen den Messpunkten und den Messwerten vermutet und folgender Ansatz gewählt

$$y(p_1, p_2) = x_1 + x_2 p_1 + x_3 p_2,$$

wobei $x = (x_1, x_2, x_3)^\top \in \mathbb{R}^3$ *unbekannt ist. Ein naheliegender Versuch,* x *zu bestimmen, besteht darin, das lineare Gleichungssystem*

$$y_i = y(p_{1,i}, p_{2,i}) = x_1 + x_2 p_{1,i} + x_3 p_{2,i}, \qquad i = 1, \ldots, 9,$$

zu lösen. In Matrixschreibweise lautet dies $Ax = b$ *mit*

$$b = \begin{pmatrix} 96 \\ 189 \\ 283 \\ 373 \\ 467 \\ 553 \\ 647 \\ 733 \\ 832 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 6 & 0 \\ 1 & 7 & 1 \\ 1 & 9 & 2 \\ 1 & 11 & 3 \\ 1 & 13 & 4 \\ 1 & 15 & 5 \\ 1 & 17 & 6 \\ 1 & 18 & 7 \\ 1 & 19 & 8 \end{pmatrix}.$$

Offenbar ist dieses lineare Gleichungssystem überbestimmt, da die Anzahl der Gleichungen die Anzahl der Unbekannten übersteigt. Leider stellt sich bei näherer Betrachtung heraus, dass das Gleichungssystem **keine Lösung** *besitzt. Der Versuch, den Vektor* x *über die Lösung des Gleichungssystems* $Ax = b$ *zu bestimmen, schlägt also fehl.*

Ein sinnvoller Zugang ist es daher, den Vektor x so zu wählen, dass der Fehler im Gleichungssystem $\|Ax-b\|_2^2$ möglichst klein wird. Dies führt auf das **Minimierungsproblem**

$$\|Ax - b\|_2^2 \to \min .$$

Dieses Problem ist auch als **lineares Ausgleichsproblem** *oder* **Least-Squares-Problem** *bekannt, welches bereits auf Gauß zurück geht.*

Johann Carl Friedrich Gauss
Born: 30.4.1777 in Braunschweig (Germany)
Died: 23.2.1855 in Göttingen (Germany)

In diesem Kapitel beschäftigen wir uns mit dem folgenden Problem.

Problem 3.0.2 **(Lineares Ausgleichsproblem)**
Gegeben seien eine Matrix $A \in \mathbb{R}^{m\times n}$ und ein Vektor $b \in \mathbb{R}^m$ mit $m, n \in \mathbb{N}$. Bestimme die Lösung $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ des Minimierungsproblems

$$\text{Minimiere} \qquad f(x) := \frac{1}{2}\|Ax - b\|_2^2 \tag{3.1}$$

Bemerkung 3.0.3

- *Im Fall $m = n$ und A invertierbar, ist die Lösung eindeutig durch das lineare Gleichungssystem $Ax = b$ bestimmt.*
- *Interessant ist insbesondere der Fall $m > n$, der in praktischen Anwendungen besonders häufig auftritt, zumeist ist m dort wesentlich grösser als n.*
- *Der Fall $m < n$ kann ebenfalls auftreten. In diesem Fall ist das Gleichungssystem $Ax = b$ unterbestimmt, es kann jedoch unlösbar sein, wenn $Rang(A) \neq Rang(A|b)$.*
- *Anstatt der Norm $\|\cdot\|_2$ können auch andere Normen verwendet werden, etwa $\|\cdot\|_\infty$ oder $\|\cdot\|_1$, jedoch führt dies i.a. auf andere Lösungen und die Berechnung von*

Lösungen gestaltet sich mitunter schwierig, da die Funktion f i.a. nicht mehr differenzierbar ist.

3.1 Gauß'sche Normalengleichungen

Wir betrachten das lineare Ausgleichsproblem von einem geometrischen Standpunkt. Dazu definieren wir den linearen Vektorraum

$$V := \{y \in \mathbb{R}^m \mid y = Ax,\ x \in \mathbb{R}^n\} = Bild(A).$$

V ist ein Teilraum des $\mathbb{R}^m$ mit Dimension $dim(V) = Rang(A)$. Das lineare Ausgleichsproblem lautet somit:
Finde $\hat{y} \in V$, so dass

$$\|b - \hat{y}\|_2 \leq \|b - y\|_2 \qquad \forall y \in V. \tag{3.2}$$

Definition 3.1.1

Sei $(X, \|\cdot\|)$ ein normierter Vektorraum, $V \subset X$ ein Teilraum und $b \in X$. $\hat{y} \in V$ heißt **Bestapproximierende an b in V**, *wenn (3.2) gilt.*

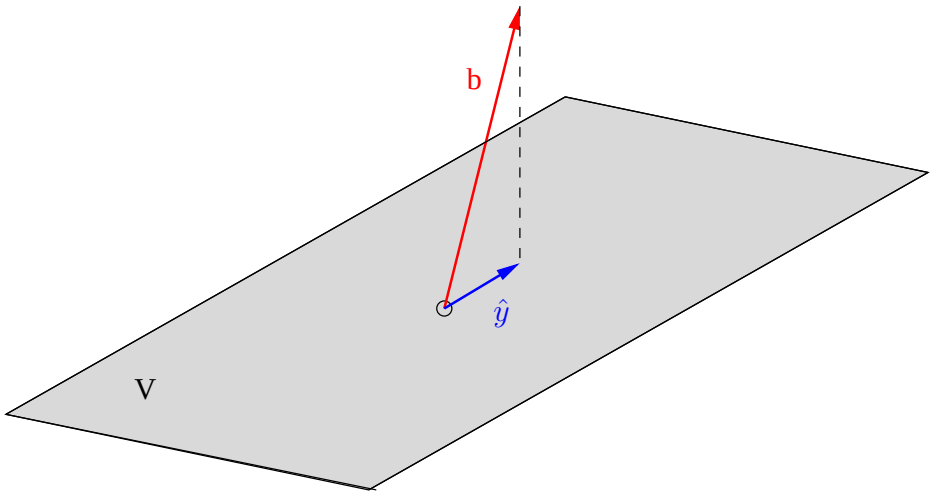


Abbildung 3.1: Bestapproximierende $\hat{y}$ an b in V.

Aus der geometrischen Anschauung ist klar, dass $\hat{y}$ genau dann Bestapproximierende an b in V ist, wenn $b - \hat{y}$ senkrecht auf V steht, vgl. Abbildung 3.1. Dies besagt der folgende Satz, der in recht allgemeiner Form gilt.

Satz 3.1.2 (Projektionssatz)

Sei X ein Vektorraum über $\mathbb{K} = \mathbb{R}$ oder $\mathbb{K} = \mathbb{C}$ und $\langle \cdot, \cdot \rangle$ ein Skalarprodukt auf $X \times X$. Sei $b \in X$ und $V \subset X$ ein Teilraum. Dann gelten:

(i) Es existiert höchstens eine Bestapproximierende $\hat{y}$ an b in V.

(ii) $\hat{y} \in V$ ist Bestapproximierende an b in V genau dann, wenn

$$\langle b - \hat{y}, v \rangle = 0 \qquad \forall v \in V. \tag{3.3}$$

Beweis: Wir zeigen zunächst (ii). Teil (i) ergibt sich unterwegs.

$\Rightarrow$: Sei $\hat{y}$ Bestapproximierende an b in V. Angenommen, (3.3) gilt nicht. Dann gibt es ein $v \in V$ mit $\langle b - \hat{y}, v \rangle = a \neq 0$. Insbesondere ist $v \neq 0$. Setze $w = \hat{y} + a\frac{v}{\|v\|^2} \in V$. Es folgt

$$\|b - w\|^2 = \langle b - \hat{y} - \frac{av}{\|v\|^2}, b - \hat{y} - \frac{av}{\|v\|^2} \rangle = \|b - \hat{y}\|^2 - \frac{|a|^2}{\|v\|^2} < \|b - \hat{y}\|^2.$$

Dies ist ein Widerspruch dazu, dass $\hat{y}$ Bestapproximierende ist.

$\Leftarrow$: Es gelte (3.3). Dann gilt für alle $v \in V$ mit $v \neq \hat{y}$,

$$\begin{aligned} \|b - v\|^2 &= \langle b - v, b - v \rangle \\ &= \langle b - \hat{y} + \hat{y} - v, b - \hat{y} + \hat{y} - v \rangle \\ &= \|b - \hat{y}\|^2 + \|\hat{y} - v\|^2 + \langle b - \hat{y}, \hat{y} - v \rangle + \langle \hat{y} - v, b - \hat{y} \rangle \\ &= \|b - \hat{y}\|^2 + \|\hat{y} - v\|^2 \\ &> \|b - \hat{y}\|^2. \end{aligned}$$

Damit ist $\hat{y}$ Bestapproximierende. (i) ist hiermit ebenfalls gezeigt, da obige Abschätzung auf einen Widerspruch führt, wenn man annimmt, dass es zwei unterschiedliche Bestapproximierende gibt.

□

Anwendung des Projektionssatzes 3.1.2 auf das lineare Ausgleichsproblem liefert folgenden Satz:

Satz 3.1.3 (Gauß'sche Normalengleichungen)
$\hat{x} \in \mathbb{R}^n$ löst das lineare Ausgleichsproblem genau dann, wenn die **Gauß'schen Normalengleichungen**

$$A^\top A\hat{x} = A^\top b$$

gelten.

Beweis: Mit $\hat{y} = A\hat{x}$ und $v = Ax$ lautet (3.3):

$$0 = \langle b - A\hat{x}, Ax \rangle = (b - A\hat{x})^\top Ax = \left(A^\top b - A^\top A\hat{x}\right)^\top x.$$

Da diese Gleichung für alle $x \in \mathbb{R}^n$ gelten muss, muss zwangsläufig $A^\top A\hat{x} - A^\top b = 0$ gelten. □

Die Gauß'schen Normalengleichungen charakterisieren also die Lösung des linearen Ausgleichproblems. Es gilt zu klären, wann die Normalengleichungen (bzw. das lineare Ausgleichsproblem) eine eindeutige Lösung besitzen. Offenbar ist $A^\top A$ symmetrisch und positiv semidefinit.

Satz 3.1.4

Die Gauß'schen Normalengleichungen besitzen genau dann eine eindeutige Lösung, wenn $Rang(A) = n$ ist.

Beweis: Zu zeigen ist, dass $A^\top A \in \mathbb{R}^{n\times n}$ positiv definit und somit invertierbar ist. Es gilt $Rang(A) = Rang(A^\top A)$. Beweis: Wegen $Ax = 0 \Rightarrow A^\top Ax = 0$ gilt $Kern(A) \subseteq Kern(A^\top A)$. Sei $x \in Kern(A^\top A)$, also $A^\top Ax = 0$. Dann ist $\langle A^\top Ax, x\rangle = \langle Ax, Ax\rangle = \|Ax\|^2 = 0$, also $x \in Kern(A)$. Insgesamt: $Kern(A) = Kern(A^\top A)$. Aus der linearen Algebra weiss man: $Rang(A^\top A) = n - \dim(Kern(A^\top A)) = n - \dim(Kern(A)) = n - (n - Rang(A)) = Rang(A)$. Damit gilt $Rang(A^\top A) = n$ ganau dann, wenn $Rang(A) = n$ gilt. □

Beispiel 3.1.5 (vgl. Beispiel 3.0.1)

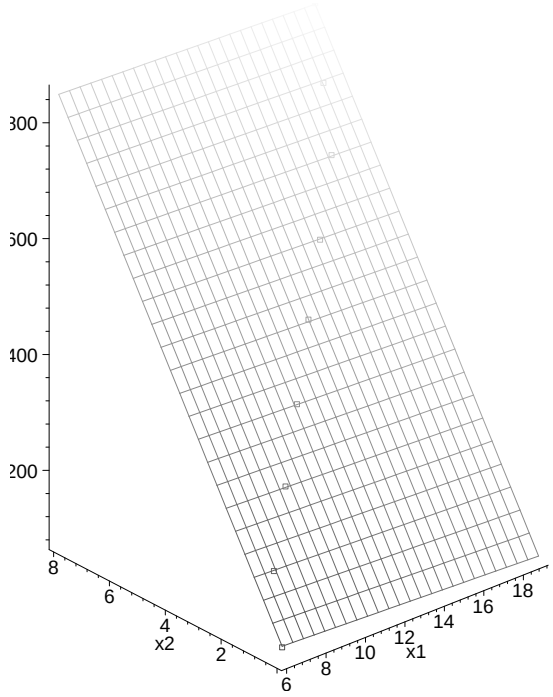
Wir betrachten wiederum Beispiel 3.0.1 und lösen das lineare Ausgleichsproblem (3.1) mit Hilfe der Normalengleichungen $A^\top Ax = A^\top b$. Es gilt

$$A^\top A = \begin{pmatrix} 9 & 115 & 36 \\ 115 & 1655 & 565 \\ 36 & 565 & 204 \end{pmatrix}, \quad A^\top b = \begin{pmatrix} 4173 \\ 62916 \\ 22176 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Die Lösung der Normalengleichungen lautet

$$x_1 = \frac{533}{5} = 106.6, \quad x_2 = -\frac{96}{65} \approx -1.476923077, \quad x_3 = \frac{6109}{65} \approx 93.98461538.$$

Grafisch ergibt sich folgendes:



Ein Nachteil der Gauß'schen Normalengleichungen liegt in der möglicherweise schlechten Kondition der Matrix $A^\top A$ im Vergleich zur Matrix A, da für quadratische Matrizen $\kappa_2(A^\top A) = \kappa_2(A)^2$ gilt. Aus diesem Grund erfolgt die numerische Lösung des Ausgleichsproblems in der Regel nicht über die Normalengleichungen, sondern wird mit Hilfe einer QR-Zerlegung der Matrix A durchgeführt. Eigenschaften und Verfahren zur Berechnung einer QR-Zerlegung werden in den folgenden Abschnitten untersucht.

3.2 QR-Zerlegung einer Matrix

Ähnlich zur LR-Zerlegung beim Gauß'schen Eliminationsverfahren ist es unser Ziel, eine nicht notwendig quadratische Matrix $A \in \mathbb{R}^{m\times n}$ zu faktorisieren in $A = Q \cdot R$.

Definition 3.2.1 (QR-Zerlegung)
Sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ gegeben. Eine Zerlegung der Form $A = Q \cdot R$ mit $Q \in R^{m\times m}$ und $R \in \mathbb{R}^{m\times n}$, wobei Q orthogonal ist und R die Gestalt

$$R = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}$$

mit einer invertierbaren rechten oberen Dreiecksmatrix $\bar{R} \in \mathbb{R}^{n\times n}$ hat, heißt **QR-Zerlegung von** A.

Wir werden sehen, dass eine solche QR-Zerlegung stets möglich ist, wenn $Rang(A) = n$ gilt. Eine QR-Zerlegung ist sehr nützlich für

- die Lösung von linearen Gleichungssystemen;
- die Lösung von linearen Ausgleichsproblemen;
- die numerische Rangbestimmung einer Matrix.

Wir beschränken die Darstellung auf den Fall $\mathbb{K} = \mathbb{R}$. Im komplexen Fall gelten die Beziehungen analog.

3.2.1 QR-Zerlegung zur Lösung von linearen Gleichungssystemen

Ist eine QR-Zerlegung von A bekannt, so kann das lineare Gleichungssystem (2.3) im Fall $n = m$ gelöst werden, indem die Orthogonalität von Q ausgenutzt wird:

$$b = Ax = Q \cdot Rx \qquad \Leftrightarrow \qquad c := Q^\top b = Rx.$$

Das lineare Gleichungssystem $Rx = c$ kann wiederum mit Rückwärtssubstitution gelöst werden. Der Hauptvorteil der QR-Zerlegung im Vergleich zur LR-Zerlegung liegt in der Tatsache begründet, dass orthogonale Transformationen invariant bzgl. der Spektralnorm sind. Für einen beliebigen Vektor x gilt

$$\|Qx\|_2 = \|x\|_2, \quad \|Q\|_2 = 1.$$

Speziell gilt $\|Q^\top b\|_2 = \|b\|_2$ und somit werden durch Multiplikation mit $Q^\top$ möglicherweise vorhandene Fehler in b nicht verstärkt. Für den relativen Fehler in (2.16) erhält man für die LR-Zerlegung wegen $\kappa_2(A) = \kappa_2(L \cdot R) \leq \kappa_2(L) \cdot \kappa_2(R)$ die im Vergleich zu (2.16) „schlechtere“ Abschätzung

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \kappa_2(L) \cdot \kappa_2(R) \cdot \frac{\|\Delta b\|_2}{\|b\|_2}.$$

Bei Verwendung der QR-Zerlegung gilt $\kappa_2(A) = \kappa_2(Q \cdot R) = \kappa_2(R)$ und somit folgt

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \kappa_2(R) \cdot \frac{\|\Delta b\|_2}{\|b\|_2}.$$

Fazit: Die QR-Zerlegung ist numerisch stabiler als die LR-Zerlegung. Der Nachteil der QR-Zerlegung liegt im etwa doppelt so hohen Aufwand im Vergleich zur LR-Zerlegung.

Bemerkung 3.2.2

Die rechte obere Dreicksmatrix R in der LR-Zerlegung von A ist nicht zu verwechseln mit der rechten oberen Dreiecksmatrix R, die bei der QR-Zerlegung berechnet wird. Die jeweiligen Matrizen sind im allgemeinen verschieden.

3.2.2 Lösung des linearen Ausgleichsproblems mittels QR-Zerlegung

Wir wenden uns wieder dem linearen Ausgleichsproblem (3.1) zu und möchten die Funktion $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ minimieren, wobei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und $b \in \mathbb{R}^m$ gegeben sind.

Wir setzen voraus, dass eine QR-Zerlegung von A gegeben ist gemäß

$$A = Q \cdot R, \qquad R = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \qquad Q \in \mathbb{R}^{m\times m}, R \in \mathbb{R}^{m\times n}.$$

Hierin sei $\bar{R} \in \mathbb{R}^{n\times n}$ eine invertierbare rechte obere Dreiecksmatrix und Q eine orthogonale Matrix. Wegen der Invarianz der Euklidnorm gilt

$$\|Ax - b\|_2^2 = \|Q^\top(Ax - b)\|_2^2 = \|Rx - Q^\top b\|_2^2.$$

Definiere

$$c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} := Q^\top b, \qquad c_1 \in \mathbb{R}^n,\ c_2 \in \mathbb{R}^{m-n}.$$

Damit folgt

$$\|Rx - Q^\top b\|_2^2 = \|\bar{R}x - c_1\|_2^2 + \|c_2\|_2^2.$$

Dieser Ausdruck wird genau dann minimal, wenn $\bar{R}x = c_1$ gilt, welches ein lineares Gleichungssystem für x darstellt und mittels Rückwärtssubstitution gelöst werden kann. Die Lösung dieses Gleichungssystems

$$\hat{x} = \bar{R}^{-1}c_1$$

ist auch die Lösung der Minimierungsaufgabe und somit des linearen Ausgleichsproblems. Der Wert des Zielfunktionals kann ebenfalls direkt abgelesen werden und lautet

$$f(\hat{x}) = \frac{1}{2}\|c_2\|_2^2.$$

Diese Vorgehensweise zur Lösung des Ausgleichsproblems ist dem Weg über die Gauß'schen Normalengleichungen aus Stabilitätsgründen vorzuziehen, da die Kondition der Matrix $A^\top A$ häufig sehr schlecht ist, während bei Verwendung der QR-Zerlegung bzgl. der Euklidnorm keine zusätzliche Fehlerverstärkung durch den Faktor Q auftritt. In den folgenden Abschnitten werden Algorithmen zur Berechnung einer QR-Zerlegung diskutiert.

3.2.3 QR-Zerlegung nach Householder

In diesem Abschnitt wird die QR-Zerlegung nach Householder besprochen, die eine numerisch stabile Methode zur Berechnung einer QR-Zerlegung $A = Q\cdot R$ gemäß Definition 3.2.1 darstellt.

Das Verfahren arbeitet mit Spiegelungen, welche durch Multiplikation mit der sogenannten Householder-Matrix H realisiert werden.

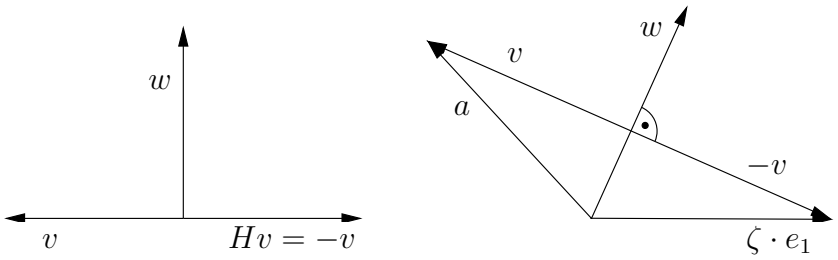


Abbildung 3.2: Householder Transformationen: Spiegelung (links) und Abbildung auf Vielfaches des Einheitsvektors bei spezieller Wahl des Vektors v (rechts).

Für einen Vektor v ist die Householder-Matrix H definiert als

$$H = I - \frac{2}{v^\top v} v v^\top. \tag{3.4}$$

Sie hat folgende Eigenschaften, vgl. auch Abbildung 3.2, die man leicht nachrechnet:

Satz 3.2.3

Für H gemäß (3.4) gelten folgende Aussagen:

(i) H ist symmetrisch ($H^\top = H$) und orthogonal ($H^\top \cdot H = I$).

(ii) H beschreibt eine Spiegelung des Vektors v gemäß $Hv = -v$.

(iii) $Hw = w$ für jeden Vektor w mit $w^\top v = 0$ (w und v stehen senkrecht aufeinander).

Durch geschickte Wahl des Vektors v ist es möglich, einen gegebenen Vektor a (z.B. die Spalte einer Matrix) auf ein Vielfaches des Einheitsvektors abzubilden. Auf dieser Idee basiert die QR-Zerlegung nach Householder.
Analog zur Berechnung der LR-Zerlegung wird ausgehend von A eine Folge von Matrizen erzeugt bis eine rechte obere Dreiecksmatrix entsteht:

$$A = A^{(1)} \to A^{(2)} \to A^{(3)} \to \ldots \to A^{(\ell+1)} = R \qquad (\ell = \min\{m-1, n\})$$

In jedem Schritt des Verfahrens wird eine speziell konstruierte Householder-Matrix $H^{(i)}$ von links an $A^{(i)}$ heranmultipliziert, um die nächste Iterierte $A^{(i+1)} = H^{(i)} \cdot A^{(i)}$ zu erhalten.

Insgesamt erhält man so

$$\begin{aligned} R &= A^{(\ell+1)} \\ &= H^{(\ell)} \cdot A^{(\ell)} \\ &= H^{(\ell)} \cdot H^{(\ell-1)} \cdot A^{(\ell-1)} \\ &\vdots \\ &= H^{(\ell)} \cdot H^{(\ell-1)} \cdots H^{(1)} \cdot A^{(1)}. \end{aligned}$$

Mit

$$Q := \left(H^{(1)}\right)^\top \cdots \left(H^{(\ell-1)}\right)^\top \cdot \left(H^{(\ell)}\right)^\top$$

(Q ist als Produkt von orthogonalen Matrizen ebenfalls orthogonal!) folgt

$$A = Q \cdot R.$$

Im folgenden wird die Konstruktion der Matrix $H^{(i)}$ beschrieben. Das Verfahren sei bis zur Matrix

$$A^{(i)} = \left(\begin{array}{ccc|ccc} a_{11}^{(i)} & \cdots & & a_{1i}^{(i)} & \cdots & a_{1n}^{(i)} \\ & \ddots & & \vdots & \ddots & \vdots \\ \hline & & & a_{ii}^{(i)} & \cdots & a_{in}^{(i)} \\ & & & a_{i+1,i}^{(i)} & \cdots & a_{i+1,n}^{(i)} \\ & & & \vdots & \ddots & \vdots \\ & & & a_{mi}^{(i)} & \cdots & a_{mn}^{(i)} \end{array}\right) =: \left(\begin{array}{c|c} R_{11}^{(i)} & A_{12}^{(i)} \\ \hline & A_{22}^{(i)} \end{array}\right) \tag{3.5}$$

fortgeschritten, so dass die ersten $i-1$ Spalten bereits in Dreiecksgestalt vorliegen. Da wir eine rechte obere Dreiecksmatrix konstruieren wollen, müssen die Elemente $a_{ji}^{(i)}$, $j = i+1, \ldots, m$, eliminiert werden. Dazu werden wir nun eine Householdertransformation derart konstruieren, dass der Vektor

$$a^{(i)} := \begin{pmatrix} a_{ii}^{(i)} \\ a_{i+1,i}^{(i)} \\ \vdots \\ a_{mi}^{(i)} \end{pmatrix} \in \mathbb{R}^{m-i+1} \tag{3.6}$$

auf ein Vielfaches $\zeta \cdot e_i$ des Einheitsvektors $e_i = (1, 0, \ldots, 0)^\top \in \mathbb{R}^{m-i+1}$ abgebildet wird, siehe Abbildung 3.2. Dies geht natürlich nur dann, wenn $\|a^{(i)}\|_2 \neq 0$ gilt. Ist $\|a^{(i)}\|_2 \neq 0$, so kann $H_i = I$ gewählt werden, allerdings ist dann $r_{ii} = 0$ in der resultierenden Zerlegung und $\bar{R}$ in Definition 3.2.1 ist nicht invertierbar.

In der folgenden Herleitung gelte $\|a^{(i)}\|_2 \neq 0$. Es gilt, einen Vektor v_i zu konstruieren, so dass

$$H_i \cdot a^{(i)} = \left(I - \frac{2}{v_i^\top v_i} v_i v_i^\top \right) \cdot a^{(i)} = \zeta \cdot e_i \tag{3.7}$$

mit einer Konstanten $\zeta \neq 0$ gilt. Mit $H_i \in \mathbb{R}^{(m-i+1)\times(m-i+1)}$ ist dann auch

$$H^{(i)} := \left(\begin{array}{c|c} I & \\ \hline & H_i \end{array} \right) \in \mathbb{R}^{m\times m}$$

orthogonal und wir erhalten

$$A^{(i+1)} = H^{(i)} \cdot A^{(i)} = \left(\begin{array}{c|c} I & \\ \hline & H_i \end{array} \right) \cdot \left(\begin{array}{c|c} R_{11}^{(i)} & A_{12}^{(i)} \\ \hline & A_{22}^{(i)} \end{array} \right) = \left(\begin{array}{c|c} R_{11}^{(i)} & A_{12}^{(i)} \\ \hline & H_i \cdot A_{22}^{(i)} \end{array} \right),$$

wobei die ersten $i-1$ Zeilen von $A^{(i)}$ durch die Multiplikation mit $H^{(i)}$ unverändert bleiben, da diese bereits in Dreiecksgestalt vorliegen.
Da H_i orthogonal ist, folgt aus (3.7) die Bedingung

$$\|H_i \cdot a^{(i)}\|_2 = \|a^{(i)}\|_2 = |\zeta|.$$

Der Ansatz

$$v_i = \frac{1}{\|a^{(i)}\|_2} \left(a^{(i)} + \alpha e_i \right) \tag{3.8}$$

liefert nach kurzer Rechnung

$$\begin{aligned}
v_i^\top a^{(i)} &= \|a^{(i)}\|_2 + \frac{\alpha a_{ii}^{(i)}}{\|a^{(i)}\|_2}, \\
v_i^\top v_i &= \frac{\|a^{(i)} + \alpha e_i\|_2^2}{\|a^{(i)}\|_2^2} = 1 + \frac{2\alpha a_{ii}^{(i)}}{\|a^{(i)}\|_2^2} + \frac{\alpha^2}{\|a^{(i)}\|_2^2}, \\
H_i \cdot a^{(i)} &= a^{(i)} - \frac{2\left(\|a^{(i)}\|_2^2 + \alpha a_{ii}^{(i)} \right)}{\|a^{(i)}\|_2^2 + 2\alpha a_{ii}^{(i)} + \alpha^2} a^{(i)} - \frac{2\alpha \left(\|a^{(i)}\|_2^2 + \alpha a_{ii}^{(i)} \right)}{\|a^{(i)}\|_2^2 + 2\alpha a_{ii}^{(i)} + \alpha^2} e_i.
\end{aligned} \tag{3.9}$$

Um $H_i a^{(i)} = \zeta e_i$ zu erfüllen, wählen wir α so, dass die ersten beiden Terme in (3.9) wegfallen, also

$$1 - \frac{2\left(\|a^{(i)}\|_2^2 + \alpha a_{ii}^{(i)} \right)}{\|a^{(i)}\|_2^2 + 2\alpha a_{ii}^{(i)} + \alpha^2} = 0.$$

Auflösen nach α liefert

$$\alpha^2 = \|a^{(i)}\|_2^2 \qquad \Rightarrow \qquad \alpha = \pm \|a^{(i)}\|_2.$$

Einsetzen von α in (3.9) liefert $H_i \cdot a^{(i)} = -\alpha e_i$. Durch die Wahl des Vorzeichens von α haben wir nun zwei Möglichkeiten, v_i in (3.8) zu wählen. Um Auslöschung in der ersten Komponente von v_i zu vermeiden, wählen wir α so, dass

$$\operatorname{sign}(\alpha) = \operatorname{sign}(a_{ii}^{(i)}) \qquad \text{bzw.} \qquad \alpha = \operatorname{sign}(a_{ii}^{(i)}) \|a^{(i)}\|_2$$

gilt und erhalten somit aus (3.8)

$$v_i = \frac{1}{\|a^{(i)}\|_2} \cdot a^{(i)} + sign(a_{ii}^{(i)}) \cdot e_i.$$

Diese Wahl von v_i leistet das Gewünschte mit

$$H_i \cdot a^{(i)} = -\text{sign}(a_{ii}^{(i)}) \|a^{(i)}\|_2 \cdot e_i.$$

Bei der Implementierung ist darauf zu achten, dass die Matrixmultiplikation $H_i \cdot A_{22}^{(i)}$ niemals explizit ausgeführt wird. Anstattdessen wird ausgenutzt, dass für eine Matrix B die Darstellung

$$H \cdot B = B - \frac{2}{v^\top v} v w^\top$$

mit $w = B^\top v$ gilt. Die Berechnung von w und $vw^\top$ benötigt jeweils nur $m \cdot n$ Operationen. Die explizite Berechnung von $H \cdot B$ benötigt $\mathcal{O}(m^2 n)$ Operationen. Zusammenfassend erhalten wir folgenden Algorithmus.

Algorithmus 3.2.4 (QR-Zerlegung nach Householder)
Gegeben: Matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$. Setze $A^{(1)} := A$.

Für $i = 1, 2, \ldots, \min\{m-1, n\}$:

Seien $a^{(i)}$ und $A_{22}^{(i)}$ definiert gemäß (3.6) und (3.5).

Falls $\|a^{(i)}\|_2 = 0$, setze $H_i = I$.

Andernfalls setze

$$\begin{aligned} v &= \frac{a^{(i)}}{\|a^{(i)}\|_2} + sign(a_{ii}^{(i)}) e_i, \\ \beta &= \frac{2}{v^\top v} = \left(1 + \frac{|a_{ii}^{(i)}|}{\|a^{(i)}\|_2}\right)^{-1}, \\ w &= \beta (A_{22}^{(i)})^\top v \end{aligned}$$

und aktualisiere $A_{22}^{(i)} := A_{22}^{(i)} - v \cdot w^\top$.

Der Aufwand (Multiplikationen und Divisionen) im i-ten Schleifendurchlauf beträgt

$$\begin{aligned} v &: 2(m-i+1) \\ \beta &: 2 \\ w &: (n-i+1)(m-i+1) + (n-i+1) \\ A_{22}^{(i)} &: (n-i+1)(m-i+1) \end{aligned}$$

Insgesamt ergibt sich ein Aufwand von

$$\sum_{i=1}^{n} ((m-i+1)(2+n-i+1+n-i+1)+(n-i+1)+2) = mn^2 - \frac{1}{3}n^3 + \mathcal{O}(mn+n^2).$$

Im Fall $m = n$ ist daher der Aufwand mit $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ in etwa doppelt so teuer wie bei der Gauß'schen LR-Zerlegung.

Beispiel 3.2.5

Gesucht ist die QR-Zerlegung der Matrix

$$A = \begin{pmatrix} 5 & 7 \\ -5 & -7 \\ -1 & 1 \end{pmatrix}.$$

Es gilt $A = Q \cdot R$ mit (gerundete Werte)

$$R \approx \begin{pmatrix} -7.140 & -9.660 \\ 0 & 2.380 \\ 0 & 0 \end{pmatrix}, \qquad Q \approx \begin{pmatrix} -0.700 & 0.099 & 0.707 \\ 0.700 & -0.099 & 0.707 \\ 0.140 & 0.990 & 0 \end{pmatrix}$$

Bemerkung 3.2.6

Die QR-Zerlegung von $A \in \mathbb{R}^{m\times n}$, $m \geq n$, $Rang(A) = n$ liefert eine Zerlegung der Form

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad Q = (Y, Z) \in \mathbb{R}^{m\times m},\ Y \in \mathbb{R}^{m\times n},\ Z \in \mathbb{R}^{m\times(m-n)},\ R \in \mathbb{R}^{n\times n}.$$

Hierin ist Y ist eine Orthonormalbasis des Bildes von A, Z ist eine Orthonormalbasis des orthogonalen Komplements von $Bild(A)$, d.h. des Kerns von $A^\top$. Jedes $x \in \mathbb{R}^n$ kann als $x = Yx_Y + Zx_z$ mit eindeutig bestimmten Vektoren $x_Y \in \mathbb{R}^n$ und $x_z \in \mathbb{R}^{(m-n)}$ dargestellt werden.

Kapitel 4

Nichtlineare Gleichungen

In vielen Problemstellungen ist es erforderlich, eine

nichtlineare Gleichung

$$F(x) = 0, \quad F : \mathbb{R} \to \mathbb{R},\ x \in \mathbb{R}, \tag{4.1}$$

oder ein

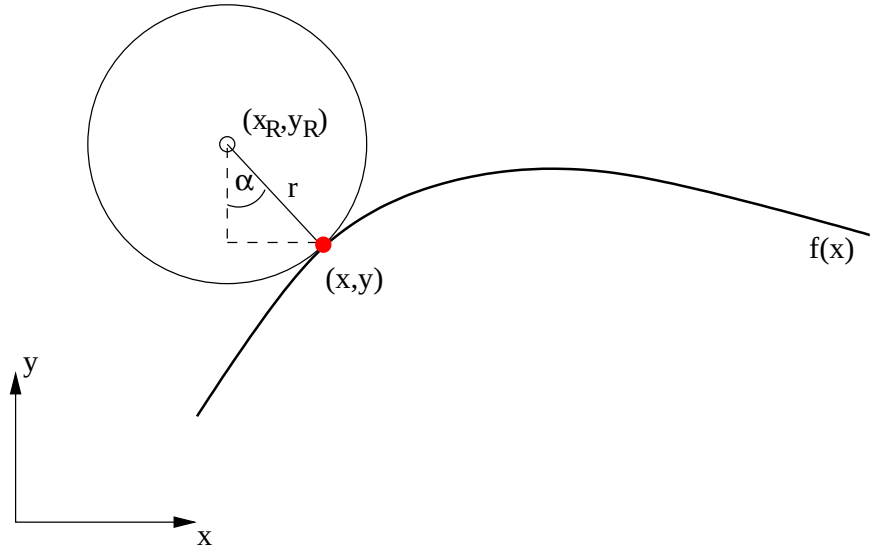
nichtlineares Gleichungssystem

$$F(x) = \begin{pmatrix} F_1(x_1, \ldots, x_n) \\ \vdots \\ F_n(x_1, \ldots, x_n) \end{pmatrix} = 0, \quad F : \mathbb{R}^n \to \mathbb{R}^n,\ x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n \tag{4.2}$$

zu lösen. Offenbar ist (4.1) als Spezialfall mit $n = 1$ in (4.2) enthalten. Es gibt jedoch spezielle Verfahren für den eindimensionalen Fall $n = 1$. Nichtlineare Gleichungssysteme können meist nicht analytisch gelöst werden, sondern erfordern numerische Verfahren, wie das Beispiel $F(x) = x - \cos(x)$ zeigt. In industriellen Anwendungen treten häufig Probleme mit mehreren tausend Variablen auf.

Beispiel 4.0.1

Bei der Simulation eines Autos tritt das Problem auf, den Kontaktpunkt (x, y) eines Autoreifens mit Mittelpunkt (x_R, y_R) und Radius r auf einer unebenen Fahrbahn zu bestimmen, vgl. Abbildung. Das Fahrbahnprofil sei durch die nichtlineare Funktion $f : \mathbb{R} \to \mathbb{R}$ beschrieben.



Der Kontaktpunkt ist implizit gegeben durch die nichtlineare Gleichung

$$f(x_R + r\sin\alpha) = y_R - r\cos\alpha,$$

die den Winkel α festlegt.

Beispiel 4.0.2 (Reaktionsgeschwindigkeiten)

Wir betrachten ein Beispiel aus der Biochemie. Gegeben sind n Proteine und ein Peptid P. Die n Proteine $1,\dots,n$ reagieren mit dem Peptid P und bilden feste Komplexe $1P,\dots,nP$ mit Konzentrationen $[1P],\dots,[nP]$. Die totalen Konzentrationen $[P]_t,[1]_t,\dots,[n]_t$ und die Reaktionsgeschwindigkeiten $k_1,\dots,k_n$ für die Reaktionen $i+P\to iP$, $i=1,\dots,n$, sind bekannt.

Es bestehen die folgenden Zusammenhänge für die Konzentrationen $[P]_f$ und $[i]_f$, $i=1,\dots,n$, der noch nicht gebundenen (freien) Peptide und Proteine:

$$\begin{aligned} [P]_f &= [P]_t - ([1P] + ... + [nP]), \\ [i]_f &= [i]_t - [iP], \qquad i = 1, ..., n, \\ [iP]\cdot k_i &= [i]_f \cdot [P]_f, \qquad i = 1, ..., n \end{aligned}$$

Gesucht sind die Konzentrationen der Komplexe $[iP]$, $i=1,...,n$, und die Konzentration $[P]_f$.

Einsetzen der obigen Beziehungen führt auf das nichtlineare Gleichungssystem

$$[iP]\cdot\left(k_i + [P]_t - \sum_{j=1}^{n}[jP]\right) - [i]_t\cdot\left([P]_t - \sum_{j=1}^{n}[jP]\right)) = 0, \qquad i = 1, ..., n.$$

Dieses nichtlineare System ist für Konzentrationen $[iP]$, $i=1,\dots,n$, zu lösen. Im allgemeinen besitzt das System mehrere Lösungen. Interessant sind dabei nur nicht-negative

Konzentrationen.

Weitere Anwendungen für nichtlineare Gleichungssysteme:

- Diskretisierung von Anfangswertproblemen mittels impliziter Integrationsverfahren
- Diskretisierung von nichtlinearen partiellen Differentialgleichungen und Randwertproblemen
- Technische Anwendungen, z.B. Bestimmung des Kontaktpunkts eines Autoreifens auf einer unebenen Fahrbahn, Bestimmung von Gleichgewichtslösungen einer Differentialgleichung, ...
- Lagrange-Newton-Verfahren in der restringierten Optimierung
- Newtonverfahren in der unrestringierten Optimierung
- Bestimmung von Polynomnullstellen (Eigenwerte sind Nullstellen des charakteristischen Polynoms)

Im folgenden werden Algorithmen zur approximativen Bestimmung von zumindest einer Nullstelle von F untersucht, wobei vorausgesetzt wird, dass eine Nullstelle existiert.

Definition 4.0.3 **(Nullstelle)**
$\hat{x} \in \mathbb{R}^n$ heißt Nullstelle von (4.2), wenn $F(\hat{x}) = 0$ gilt.

Die Verfahren in diesem Kapitel haben die Gemeinsamkeit, dass sie **iterativ** arbeiten, d.h. es wird eine Folge $\{x^{(k)}\}$ mit

$$x^{(k+1)} = g(x^{(k)}), \qquad k = 0, 1, 2, \ldots$$

berechnet, wobei g eine geeignete Funktion ist. In jedem Iterationsschritt muss dabei ein einfach zu lösendes Problem gelöst werden. Das Hauptaugenmerk wird auf folgenden Fragestellungen liegen:

- Unter welchen Voraussetzungen konvergierte die Folge $\{x^{(k)}\}$ gegen $\hat{x}$?
- Wie schnell konvergiert $\{x^{(k)}\}$ gegen $\hat{x}$?

Die Konvergenzgeschwindigkeit einer Folge ist wie folgt definiert, wobei eine Folge umso schneller konvergiert, je höher die Konvergenzordnung ist.

Definition 4.0.4 (lineare Konvergenz, quadratische Konvergenz, superlineare Konvergenz, Konvergenzordnung)
Die Folge $\{x^{(i)}\}$ konvergiere gegen $\hat{x}$.

(a) Die Folge $\{x^{(i)}\}$ heißt **linear konvergent**, *falls es eine Konstante $0 \le C < 1$ und eine Zahl i_0 gibt mit*

$$\|x^{(i+1)} - \hat{x}\| \le C \cdot \|x^{(i)} - \hat{x}\| \quad \forall i \ge i_0.$$

(b) Die Folge $\{x^{(i)}\}$ heißt **quadratisch konvergent**, *falls es eine Konstante C und eine Zahl i_0 gibt mit*

$$\|x^{(i+1)} - \hat{x}\| \le C \cdot \|x^{(i)} - \hat{x}\|^2 \quad \forall i \ge i_0.$$

(c) Die Folge $\{x^{(i)}\}$ heißt **konvergent mit Ordnung** $p > 1$, *falls es eine Konstante C und eine Zahl i_0 gibt mit*

$$\|x^{(i+1)} - \hat{x}\| \le C \cdot \|x^{(i)} - \hat{x}\|^p \quad \forall i \ge i_0.$$

(d) Die Folge $\{x^{(i)}\}$ heißt **superlinear konvergent**, *falls es eine Folge $\{C_i\}$ mit $\lim_{i\to\infty} C_i = 0$ und eine Zahl i_0 gibt mit*

$$\|x^{(i+1)} - \hat{x}\| \le C_i \cdot \|x^{(i)} - \hat{x}\| \quad \forall i \ge i_0.$$

Beachte, dass die Konvergenz der Folge $\{x^{(i)}\}$ in der Definition explizit vorausgesetzt wurde. Im Fall der linearen und superlinearen Konvergenz ist dies nicht nötig, da lineare und superlineare Konvergenz automatisch die Konvergenz der Folge nach sich ziehen. Für die übrigen Konvergenzbegriffe gilt dies nicht, so dass beim Beweis der Konvergenz mit Ordnung $p > 1$ stets zunächst die Konvergenz der Folge gezeigt werden muss.

4.1 Bisektion (Intervallschachtelungsverfahren)

Im eindimensionalen Fall $n = 1$ wird unter der Voraussetzung, dass F stetig ist, häufig das Bisketionsverfahren zur Bestimmung **einer** Nullstelle verwendet. Ausgehend von einem Startintervall $[a, b]$, $a < b$, mit

$$F(a) \cdot F(b) < 0 \tag{4.3}$$

berechnet das Bisketionsverfahren in jedem Schritt ein neues Intervall mit halber Länge, in dem eine Nullstelle von F enthalten ist. Man erhält so eine beliebig genaue Einschachtelung einer Nullstelle. Das Bisektionsverfahren kann u.a. angewendet werden, um Eigenwerte zu berechnen, indem eine Nullstelle des charakteristischen Polynoms gesucht wird.

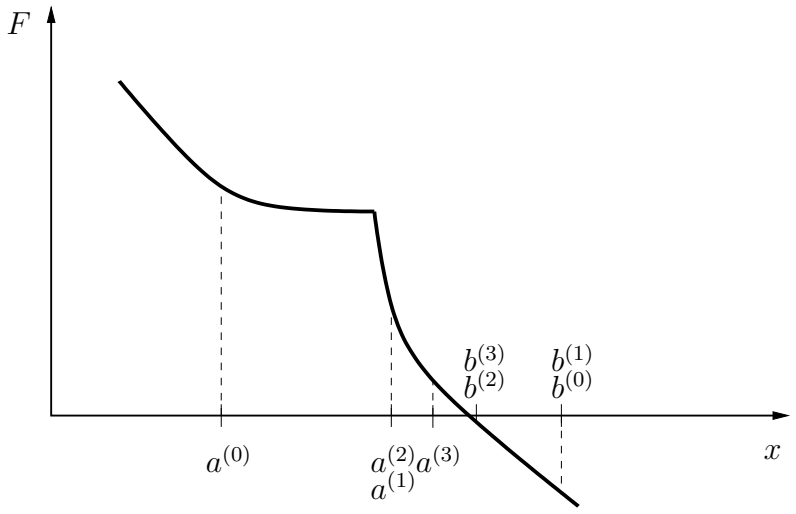


Abbildung 4.1: Bisketion: Iterierte Intervallhalbierung.

Bedingung (4.3) besagt, dass $F(a)$ und $F(b)$ unterschiedliche Vorzeichen besitzen. In diesem Fall garantiert der **Zwischenwertsatz** 1.1.1, dass im Intervall (a, b) (mindestens) eine Nullstelle von F existiert.
Durch iterierte Intervallschachtelung, vgl. Abbildung 4.1, erlaubt das Bisektionsverfahren die beliebig genaue Approximation einer Nullstelle in $[a, b]$ (es ist aber nicht gesagt, welche Nullstelle man bekommt).

Algorithmus 4.1.1 (Bisektionsverfahren)

(0) Input:

- *a, b mit $a < b$ und $F(a) \cdot F(b) < 0$*
- *Toleranz $tol > 0$*
- *maximale Iterationszahl n_{max}*

(1) Setze $i = 0$, $a^{(0)} = a$, $b^{(0)} = b$.

(2) Setze $x^{(i)} = (a^{(i)} + b^{(i)})/2$ und $w = F(x^{(i)})$.
Falls $|b^{(i)} - a^{(i)}| \leq tol$ oder $i > n_{max}$, STOP mit Ausgabe $x^{(i)}$ und w.
Falls $w = 0$, dann ist $x^{(i)}$ eine Nullstelle, STOP mit Ausgabe $x^{(i)}$ und w.
Falls $w \cdot F(a^{(i)}) < 0$, setze $a^{(i+1)} = a^{(i)}$, $b^{(i+1)} = x^{(i)}$.
Falls $w \cdot F(a^{(i)}) > 0$, setze $a^{(i+1)} = x^{(i)}$, $b^{(i+1)} = b^{(i)}$.

(3) Setze $i = i + 1$ und gehe zu (2).

In jeder Iteration $i = 0, 1, 2, \ldots$ des Bisektionsverfahrens wird ein Intervall $[a^{(i)}, b^{(i)}]$ konstruiert. Per Konstruktion des Intervalls wissen wir nach dem Zwischenwertsatz, dass mindestens eine Nullstelle von F im Intervall $[a^{(i)}, b^{(i)}]$ für alle $i = 0, 1, 2, \ldots$ liegen muss. Entsprechend ist die beste Approximation für alle Nullstellen von F im Intervall $[a^{(i)}, b^{(i)}]$ in der i-ten Iteration gegeben durch

$$x^{(i)} = \frac{a^{(i)} + b^{(i)}}{2}.$$

Seit $\hat{x}^{(i)}$ eine beliebige Nullstelle von F im Intervall $[a^{(i)}, b^{(i)}]$. Dann erhalten wir die Fehlerabschätzung

$$|x^{(i)} - \hat{x}^{(i)}| \leq \frac{b^{(i)} - a^{(i)}}{2} \quad \forall i = 0, 1, 2, \ldots.$$

Da $b^{(i)} - a^{(i)} = (b^{(i-1)} - a^{(i-1)})/2$ gilt, folgt durch Induktion, dass

$$|x^{(i)} - \hat{x}^{(i)}| \leq \frac{b^{(0)} - a^{(0)}}{2^{i+1}} = \frac{b - a}{2^{i+1}} \quad \forall i = 0, 1, 2, \ldots.$$

Dies ist gerade die Konvergenz des Verfahren gegen eine Nullstelle von F, da $(b-a)/2^{i+1} \to 0$ für $i \to \infty$. Zusammenfassend erhalten wir

Satz 4.1.2

Sei F stetig in $[a, b]$, $a < b$, mit $F(a) \cdot F(b) < 0$.
Nach i Iterationen des Bisektionsverfahrens gilt für die Approximation $x^{(i)}$ einer beliebigen Nullstelle $\hat{x}^{(i)} \in [a^{(i)}, b^{(i)}]$ die Fehlerabschätzung

$$|x^{(i)} - \hat{x}^{(i)}| \leq \frac{b - a}{2^{i+1}}.$$

Häufig ist man daran interessiert, eine Nullstelle mit einer vorgegebenen Toleranz $\varepsilon > 0$ zu approximieren. Die dafür notwendige Anzahl an Iterationen bei exakter Rechnung ergibt sich aus der Fehlerabschätzung

$$|x^{(i)} - \hat{x}^{(i)}| \leq \frac{b - a}{2^{i+1}} < \varepsilon$$

und berechnet sich zu

$$i > \log_2 \left(\frac{b - a}{2\varepsilon} \right) = \frac{\log\left(\frac{b-a}{2\varepsilon}\right)}{\log 2}.$$

Beachte, dass die Folge $\{x^{(i)}\}$ i.a. **nicht** linear konvergent im Sinne von Definition 4.0.4 ist. Aber da $b^{(i)} - a^{(i)} = (b^{(i-1)} - a^{(i-1)})/2$ gilt, ist die Folge $\left\{(b^{(i)} - a^{(i)})/2\right\}$ der Fehlerschranken linear konvergent mit Konstante $C = 1/2$. Daher betrachten wir das Bisektionsverfahren ebenfalls als linear konvergent.

Beispiel 4.1.3

Berechnung von $\sqrt{2}$ als (positive) Nullstelle von $F(x) = x^2 - 2$.
Ausgabe des Bisektionsverfahrens mit Startintervall $[1, 1.5]$:

```
Iteration 1 :  x= 1.5000000000 in [1.0000000000, 1.5000000000], error: 0.2500000000
Iteration 2 :  x= 1.2500000000 in [1.2500000000, 1.5000000000], error: 0.1250000000
Iteration 3 :  x= 1.3750000000 in [1.3750000000, 1.5000000000], error: 0.0625000000
Iteration 4 :  x= 1.4375000000 in [1.3750000000, 1.4375000000], error: 0.0312500000
Iteration 5 :  x= 1.4062500000 in [1.4062500000, 1.4375000000], error: 0.0156250000
Iteration 6 :  x= 1.4218750000 in [1.4062500000, 1.4218750000], error: 0.0078125000
Iteration 7 :  x= 1.4140625000 in [1.4140625000, 1.4218750000], error: 0.0039062500
Iteration 8 :  x= 1.4179687500 in [1.4140625000, 1.4179687500], error: 0.0019531250
Iteration 9 :  x= 1.4160156250 in [1.4140625000, 1.4160156250], error: 0.0009765625
Iteration 10:  x= 1.4150390625 in [1.4140625000, 1.4150390625], error: 0.0004882812
Iteration 11:  x= 1.4145507812 in [1.4140625000, 1.4145507812], error: 0.0002441406
Iteration 12:  x= 1.4143066406 in [1.4140625000, 1.4143066406], error: 0.0001220703
Iteration 13:  x= 1.4141845703 in [1.4141845703, 1.4143066406], error: 0.0000610352
Iteration 14 : x= 1.4142456055 in [1.4141845703, 1.4142456055], error: 0.0000305176
Iteration 15 : x= 1.4142150879 in [1.4141845703, 1.4142150879], error: 0.0000152588
Iteration 16 : x= 1.4141998291 in [1.4141998291, 1.4142150879], error: 0.0000076294
Iteration 17 : x= 1.4142074585 in [1.4142074585, 1.4142150879], error: 0.0000038147

x =  1.4142 f = -1.7264e-05 a =  1.4142 b =  1.4142
```

Das Bisketionsverfahren bietet folgende **Vorteile:**

- Es werden nur Funktionsauswertungen benötigt und das Verfahren ist sehr einfach zu implementieren.
- Die Funktion muss lediglich stetig sein. Differenzierbarkeit wird nicht benötigt.
- Es ist sehr genau und konvergiert stets, wenn die Anfangswerte richtig gewählt wurden.

Leider hat das Bisektionsverfahren auch einige **Nachteile:**

- Das Verfahren lässt sich nur für $n = 1$ anwenden. Eine Erweiterung auf $n > 1$ ist schwierig.
- Das Verfahren liefert nur eine lineare Konvergenzordnung mit Konstante $C = 0.5$.
- Es muss ein Startintervall $[a, b]$ mit $F(a) \cdot F(b) < 0$ bestimmt werden.

4.2 Fixpunktiteration

Wir haben im Zusammenhang mit iterativen Verfahren zur Lösung von linearen Gleichungssystemen bereits den Banachschen Fixpunktsatz 2.5.4 zur iterativen Berechnung eines Fixpunkts der Gleichung

$$x = g(x)$$

kennengelernt. Die Aufgabenstellung (4.2) lässt sich nun leicht auf „Fixpunktgestalt“ transformieren, etwa in der Form

$$F(x) = 0 \qquad \Leftrightarrow \qquad x = x + \rho(x) \cdot F(x) =: g(x),$$

wobei $\rho(x) \in \mathbb{R}^{n\times n}$ eine reguläre, problemabhängige Matrix ist, die dazu dienen soll, die Voraussetzungen des Banachschen Fixpunktsatzes zu erfüllen. Der naive Ansatz $\rho(x) = \pm I$ für alle x führt nur selten zum Erfolg. Anstelle des Nullstellenproblems (4.2) wird also ein Fixpunkt $\hat{x}$ der Funktion g gesucht. Unter der Annahme, dass $\rho(\hat{x})$ regulär ist, ist der Fixpunkt $\hat{x}$ wegen

$$\hat{x} = \hat{x} + \rho(\hat{x}) \cdot F(\hat{x}) \quad \Leftrightarrow \quad \rho(\hat{x}) \cdot F(\hat{x}) = 0 \quad \Leftrightarrow \quad F(\hat{x}) = 0$$

auch Nullstelle von F.
Für einen geeigneten Startwert $x^{(0)}$ wird die Fixpunktiteration

$$x^{(i+1)} = g(x^{(i)}) = x^{(i)} + \rho(x^{(i)}) \cdot F(x^{(i)}), \quad i = 0, 1, \ldots$$

zur Approximation von $\hat{x}$ durchgeführt, vgl. Abbildung 4.2.

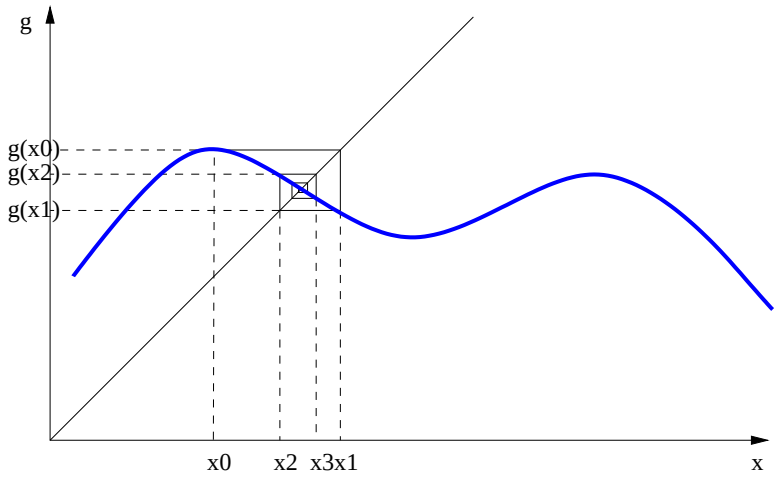


Abbildung 4.2: Schematische Darstellung der Fixpunktiteration.

Diese Methode ist auch dann anwendbar, wenn F nicht differenzierbar ist. Um das Konvergenzresultat im Banachschen Fixpunktsatz anwenden zu können, müssen folgende Voraussetzungen erfüllt sein, vgl. Satz 2.5.4:

- **Selbstabbildungseigenschaft**, d.h. es muss eine abgeschlossene Menge $D \subseteq \mathbb{R}^n$ existieren, so dass $g(D) \subseteq D$ gilt.
- g muss die **Kontraktionsbedingung**

  $$\|g(x) - g(y)\| \leq q\|x - y\|$$

  für ein $q < 1$ und alle $x, y \in D$ erfüllen.

Kennt man die Kontraktionszahl $q < 1$, so folgt aus dem Banachschen Fixpunktsatz die Fehlerabschätzung

$$\|x^{(i)} - \hat{x}\| \leq \frac{q^i}{1-q}\|x^{(1)} - x^{(0)}\|.$$

Der Nachteil der Fixpunktiteration liegt in der i.a. nur linearen Konvergenzgeschwindigkeit.
Ist g stetig differenzierbar, so kann die Kontraktionsbedingung (2.20) mit Hilfe der Ableitung von g überprüft werden.

Satz 4.2.1

Es sei $D \subseteq \mathbb{R}^n$ abgeschlossen und konvex und $g : D \to D$ stetig differenzierbar. Gibt es eine Konstante $q < 1$ mit $\|g'(x)\| \leq q$ für alle $x \in D$, so erfüllt g in D die Kontraktionsbedingung (2.20).

Beweis: Seien $x, y \in D$ gegeben. Anwendung des Mittelwertsatzes in Integralform (vgl. Satz 1.1.4) liefert

$$g(x) - g(y) = \int_0^1 g'(x + t(y - x))(y - x)dt.$$

Wir schätzen mit einer geeigneten Vektornorm und einer verträglichen Matrixnorm ab:

$$\begin{aligned} \|g(x) - g(y)\| &\leq \int_0^1 \|g'(x + t(y - x))(y - x)\|dt \\ &\leq \int_0^1 \|g'(x + t(y - x))\| \cdot \|y - x\|dt \\ &\leq \max_{z \in D} \|g'(z)\| \cdot \int_0^1 \|y - x\|dt \\ &\leq q\|x - y\|. \end{aligned}$$

□

Bemerkung 4.2.2

Es ist möglich, die Kontraktionszahl q numerisch zu schätzen, sobald einige Iterierte zur Verfügung stehen. Man betrachte die Quotienten

$$q \approx \frac{|x^{(i+1)} - x^{(i)}|}{|x^{(i)} - x^{(i-1)}|}, \quad i = 1, 2, \ldots.$$

Hintergrund: Aus der Kontraktivität von g folgt

$$|x^{(i+1)} - x^{(i)}| = |g(x^{(i)}) - g(x^{(i-1)})| \leq q|x^{(i)} - x^{(i-1)}|.$$

Das folgende Beispiel zeigt einerseits, dass die Fixpunktiteration nicht immer konvergiert, andererseits wird der Einfluss von ρ verdeutlicht.

Beispiel 4.2.3

Berechnung von $\sqrt{2}$ als (positive) Nullstelle von $F(x) = x^2 - 2$. Ausgehend vom Startwert $x^{(0)} = 2$ wird zunächst mit $\rho(x) = 1$ für alle x gerechnet. Ausgabe

```
Iteration 1 :      4.0000000000, Fehler:      2.5857864376
Iteration 2 :     18.0000000000, Fehler:     16.5857864376
Iteration 3 :    340.0000000000, Fehler:    338.5857864376
Iteration 4 : 115938.0000000000, Fehler: 115936.5857864376
```

Die Fixpunktiteration divergiert. Hier ist $g(x) = x + x^2 - 2$ *und* $g'(x) = 1 + 2x$. *Damit ist* $|g'(x)| \geq 1$ *für* $x \notin (-1, 0)$. *In einer Umgebung des Startwerts* $x^{(0)} = 2$ *ist die Kontraktionsbedingung nicht erfüllt. Wird bei gleichem Startwert hingegen* $\rho(x) = -1/4$ *gewählt, erhält man die Ausgabe*

```
Iteration 1 :    1.5000000000, Fehler:    0.0857864376, Rate:    0.1464466094
Iteration 2 :    1.4375000000, Fehler:    0.0232864376, Rate:    0.2714466094
Iteration 3 :    1.4208984375, Fehler:    0.0066848751, Rate:    0.2870716094
Iteration 4 :    1.4161603451, Fehler:    0.0019467827, Rate:    0.2912220000
Iteration 5 :    1.4147828143, Fehler:    0.0005692520, Rate:    0.2924065231
Iteration 6 :    1.4143802114, Fehler:    0.0001666490, Rate:    0.2927509058
Iteration 7 :    1.4142623658, Fehler:    0.0000488034, Rate:    0.2928515566
Iteration 8 :    1.4142278560, Fehler:    0.0000142936, Rate:    0.2928810180
Iteration 9 :    1.4142177488, Fehler:    0.0000041864, Rate:    0.2928896454
Iteration 10 :   1.4142147886, Fehler:    0.0000012262, Rate:    0.2928921722
Iteration 11 :   1.4142139215, Fehler:    0.0000003591, Rate:    0.2928929122
Iteration 12 :   1.4142136676, Fehler:    0.0000001052, Rate:    0.2928931292
Iteration 13 :   1.4142135932, Fehler:    0.0000000308, Rate:    0.2928931913
Iteration 14 :   1.4142135714, Fehler:    0.0000000090, Rate:    0.2928932135
Iteration 15 :   1.4142135650, Fehler:    0.0000000026, Rate:    0.2928932004
```

Die Fixpunktiteration konvergiert offensichtlich. Es ist $g(x) = x - (x^2 - 2)/4$ *und* $g'(x) = 1 - x/2$. *Damit ist* $|g'(x)| < 1$ *für* $x \in (0, 4)$. *Damit ist* g *kontrahierend in einer Umgebung des Startwerts* $x^{(0)} = 2$. *Zu beachten ist noch, dass die Iterationsfolge im Intervall* $(0, 4)$ *verbleibt. Somit sind die Voraussetzungen des Banachschen Fixpunktsatzes erfüllt und es folgt die Konvergenz. Die Konvergenzkonstante beträgt hier ca.* 0.29289.

Die Wahl von ρ *ist entscheidend und leider problemabhängig.*

Bemerkung 4.2.4 **(Wahl von ρ)**

Wir werden später sehen, dass die Wahl $\rho(x) = -F'(x)^{-1}$ *sehr geeignet ist und auf das sogenannte Newtonverfahren*

$$x^{(i+1)} = x^{(i)} - F'(x^{(i)})^{-1} F(x^{(i)}), \quad i = 0, 1, 2, \ldots$$

führt. Insbesondere kann das Newtonverfahren also als Fixpunktiteration mit

$$g(x) = x - F'(x)^{-1} F(x)$$

interpretiert werden.

4.3 Newtonverfahren

Die grundsätzliche Idee für das Newtonverfahren wird zunächst im eindimensionalen Fall (4.1) erläutert und später auf den mehrdimensionalen Fall (4.2) übertragen. In jedem Fall wird vorausgesetzt, dass die Funktion F mindestens **stetig differenzierbar** ist.

4.3.1 Der eindimensionale Fall

Die Idee des Newtonverfahrens ist es, die Funktion F lokal im Punkt $x^{(i)}$ durch ihre **Tangente** $T(x)$ zu approximieren, d.h.

$$F(x) \approx T(x) := F(x^{(i)}) + F'(x^{(i)})(x - x^{(i)}),$$

vgl. Abbildung 4.3.

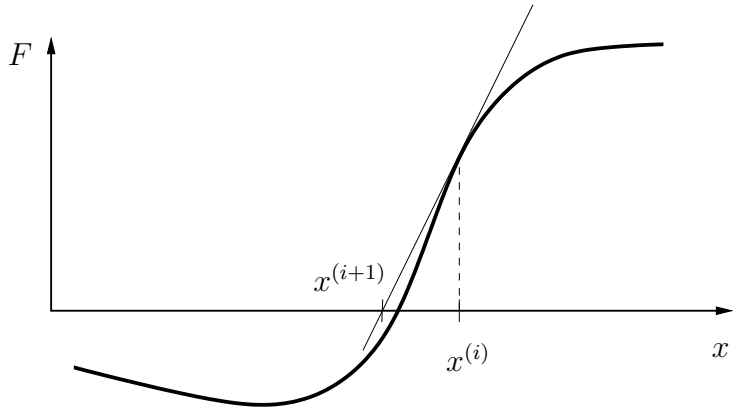


Abbildung 4.3: Newtonverfahren: Lokale Approximation von F durch Tangente im Punkt $x^{(i)}$. Die Nullstelle der Tangente definiert die neue Iterierte $x^{(i+1)}$.

Natürlich ist die Tangente i.a. lediglich eine Approximation an F. Ist F zweimal stetig differenzierbar, so gilt nach dem Taylor'schen Satz die Beziehung

$$F(x) = \underbrace{F(x^{(i)}) + F'(x^{(i)})(x - x^{(i)})}_{=T(x)} + \frac{1}{2}F''(\xi)(x - x^{(i)})^2,$$

wobei ξ zwischen x und $x^{(i)}$ liegt. Der große Vorteil der Approximation durch T besteht darin, dass die Nullstelle von T leicht berechnet werden kann, während dies für F nicht möglich ist. Auf Grund der Taylorentwicklung kann man auch erwarten, dass die Nullstelle von T eine hinreichend gute Approximation für eine Nullstelle $\hat{x}$ von F darstellt, falls die Linearisierung in der Nähe von $\hat{x}$ erfolgt.
Es sei $F'(x^{(i)}) \neq 0$. Dann besitzt T eine Nullstelle, die mit $x^{(i+1)}$ bezeichnet wird und sich berechnet zu

$$T(x^{(i+1)}) = 0 \qquad \Leftrightarrow \qquad x^{(i+1)} = x^{(i)} - \frac{F(x^{(i)})}{F'(x^{(i)})}.$$

$x^{(i+1)}$ wird als neue Approximation der Nullstelle gewählt. Iterative Anwendung dieser Regel liefert das lokale Newtonverfahren.

Algorithmus 4.3.1 ((Lokales) Newtonverfahren)

(i) *Wähle* $x^{(0)} \in \mathbb{R}$, $tol > 0$ *und setze* $i = 0$.

(ii) *Falls* $|F(x^{(i)})| \leq tol$, *STOP.*

(iii) *Berechne*

$$x^{(i+1)} = x^{(i)} - \frac{F(x^{(i)})}{F'(x^{(i)})}$$

(iv) *Setze* $i = i + 1$ *und gehe zu (ii).*

Bemerkung 4.3.2

Weitere sinnvolle Abbruchbedingungen mit geeignet gewählten Konstanten $\delta, \varepsilon, \gamma > 0$*lauten*

$$\begin{aligned} |F'(x^{(i)})| &\leq \delta \\ |F(x^{(i)})| &\leq \varepsilon |F'(x^{(i)})| \\ |x^{(i+1)} - x^{(i)}| &\leq \gamma(1 + |F(x^{(i)})|) \end{aligned}$$

Es stellt sich die Frage, inwiefern das oben konstruierte Verfahren sinnvoll ist und wann es konvergiert.

Satz 4.3.3

Sei F *zweimal stetig differenzierbar auf* $[a, b]$. F *habe eine Nullstelle* $\hat{x} \in (a, b)$ *mit* $F'(\hat{x}) \neq 0$. *Dann existiert ein* $r > 0$, *so dass das Newtonverfahren für alle Startwerte* $x^{(0)}$ *mit* $|x^{(0)} - \hat{x}| \leq r$ *mindestens quadratisch gegen* $\hat{x}$ *konvergiert.*

Beweis:

(i) Konvergenz:
Das Newtonverfahren wird als Fixpunktiteration mit

$$g(x) = x - \frac{F(x)}{F'(x)}$$

aufgefasst, d.h. $\rho(x) = -1/F'(x)$. Wegen $F(\hat{x}) = 0$ folgt in $\hat{x}$

$$g'(\hat{x}) = 1 - \frac{F'(\hat{x})^2 - F(\hat{x})F''(\hat{x})}{F'(\hat{x})^2} = \frac{F(\hat{x})F''(\hat{x})}{F'(\hat{x})^2} = 0.$$

Da $F'(\hat{x}) \neq 0$ vorausgesetzt ist und da $g'(\hat{x}) = 0$ gilt, folgt aus der Stetigkeit von F' und g', dass es ein $r > 0$ mit

$$|F'(x)| \geq \mu > 0, \quad |g'(x)| \leq q < 1$$

für alle $x \in [\hat{x} - r, \hat{x} + r] \subseteq [a, b]$ gibt. Also ist g eine kontrahierende Abbildung auf $[\hat{x} - r, \hat{x} + r]$. Es bleibt noch zu zeigen, dass die Fixpunktiteration nicht aus dem Intervall $[\hat{x} - r, \hat{x} + r]$ hinausführt. Zunächst sei bemerkt, dass $\hat{x}$ Fixpunkt von g ist, denn wegen $F(\hat{x}) = 0$ folgt $g(\hat{x}) = \hat{x} - F(\hat{x})/F'(\hat{x}) = \hat{x}$. Sei nun $x \in [\hat{x} - r, \hat{x} + r]$ beliebig, d.h. es gilt $|x - \hat{x}| \leq r$. Zu zeigen ist, dass dann auch $g(x) \in [\hat{x} - r, \hat{x} + r]$ bzw. $|g(x) - \hat{x}| \leq r$ gilt. Dies folgt aus

$$|g(x) - \hat{x}| = |g(x) - \underbrace{g(\hat{x})}_{=\hat{x}}| \leq \underbrace{q}_{<1} \underbrace{|x - \hat{x}|}_{\leq r} < r.$$

Damit sind die Voraussetzungen des Fixpunktsatzes bewiesen und es folgt die Konvergenz des Newtonverfahrens für jeden Startwert $x^{(0)} \in [\hat{x} - r, \hat{x} + r]$.

(ii) Konvergenzordnung:
Taylorentwicklung um die Iterierte $x^{(i)}$ liefert

$$0 = F(\hat{x}) = F(x^{(i)}) + F'(x^{(i)})(\hat{x} - x^{(i)}) + \frac{1}{2}F''(\xi)(\hat{x} - x^{(i)})^2, \quad \xi \in [\hat{x} - r, \hat{x} + r].$$

Damit folgt

$$x^{(i+1)} - \hat{x} = x^{(i)} - \frac{F(x^{(i)})}{F'(x^{(i)})} - \hat{x} = \frac{1}{2}\frac{F''(\xi)}{F'(x^{(i)})}(\hat{x} - x^{(i)})^2.$$

Da F'' beschränkt ist auf der kompakten Menge $[\hat{x} - r, \hat{x} + r]$ (folgt aus der Stetigkeit von F'') folgt für alle Startwerte $x^{(0)} \in [\hat{x} - r, \hat{x} + r]$

$$|x^{(i+1)} - \hat{x}| \leq \underbrace{\frac{1}{2\mu} \max_{\xi \in [a,b]} |F''(\xi)|}_{=:C} \cdot |\hat{x} - x^{(i)}|^2, \quad i = 0, 1, 2, \ldots.$$

Dies ist gerade die quadratische Konvergenz.

□

Bemerkung 4.3.4

Beachte, dass das Newtonverfahren nur für solche Anfangswerte $x^{(0)}$ mit $|\hat{x} - x^{(0)}| \leq r$ konvergiert. Für Anfangswerte außerhalb dieser Umgebung kann das Verfahren divergieren. Man spricht hier von **lokaler Konvergenz**.

Beispiel 4.3.5

Wir betrachten erneut das Problem, $\sqrt{2}$ als (positive) Nullstelle von $F(x) = x^2 - 2$ zu berechnen. Das Newtonverfahren liefert die folgende Ausgabe:

```
Iteration 1 :    1.5000000000, Fehler:    0.0857864376
Iteration 2 :    1.4166666667, Fehler:    0.0024531043
Iteration 3 :    1.4142156863, Fehler:    0.0000021239
Iteration 4 :    1.4142135624, Fehler:    0.0000000000
```

Hier wird die Stärke des Newtonverfahrens im Vergleich zum Bisektionsverfahren deutlich. Während das Bisektionsverfahren hier 17 Iterationen benötigte, konvergiert das Newtonverfahren in nur 4 Iterationen, wobei die Anzahl der führenden Nullen im Fehler sich von Iteration zu Iteration in etwa verdoppeln. Dies ist die quadratische Konvergenz des Newtonverfahrens.

4.3.2 Der n-dimensionale Fall

Die Idee des Newtonverfahrens lässt sich problemlos auf den n-dimensionalen Fall übertragen. Das Newtonverfahren basiert auf lokaler Taylorentwicklung der vektorwertigen Funktion $F = (F_1, \dots, F_n)^\top$ um die Iterierte $x^{(i)}$:

$$F(x) = F(x^{(i)}) + F'(x^{(i)}) \cdot (x - x^{(i)}) + o(\|x - x^{(i)}\|).$$

Darin bezeichnet

$$F'(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1}(x) & \cdots & \frac{\partial F_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1}(x) & \cdots & \frac{\partial F_n}{\partial x_n}(x) \end{pmatrix}$$

die **Jacobimatrix** von F and der Stelle $x = (x_1, \dots, x_n)^\top$. Vernachlässigt man den Term $o(\|x-x^{(i)}\|)$ in der Taylorentwicklung von F, so wird F lokal durch ihre Richtungstangente T im Punkt $x^{(i)}$ in Richtung $x - x^{(i)}$ mit

$$T(x) = F(x^{(i)}) + F'(x^{(i)}) \cdot (x - x^{(i)})$$

approximiert (F wird in $x^{(i)}$ linearisiert.). Die neue Iterierte $x^{(i+1)}$ ergibt aus der Nullstelle der Richtungstangente und führt auf die Darstellung

$$x^{(i+1)} = x^{(i)} - F'(x^{(i)})^{-1} \cdot F(x^{(i)}). \tag{4.4}$$

In der Praxis wird die Inverse der Jacobimatrix $F'(x)^{-1}$ **niemals** explizit berechnet, da hierfür n lineare Gleichungssysteme der Dimension $n \times n$ gelöst werden müssten. Stattdessen wird (4.4) in der Form

$$\begin{aligned} F'(x^{(i)}) \cdot d^{(i)} &= -F(x^{(i)}), \\ x^{(i+1)} &= x^{(i)} + d^{(i)}, \quad i = 0, 1, 2, \dots \end{aligned}$$

gelöst. Diese Variante benötigt nicht mehr die Inverse der Jacobimatrix und erfordert nur die Lösung **eines** linearen Gleichungssystems für $d^{(i)}$. Zusammenfassend erhalten wir

Algorithmus 4.3.6 **((Lokales) Newtonverfahren in $\mathbb{R}^n$)**

(i) *Wähle $x^{(0)} \in \mathbb{R}^n$, $tol > 0$ und setze $i = 0$.*

(ii) *Falls $\|F(x^{(i)})\| \leq tol$, STOP.*

(iii) *Löse (z.B. mit der LR-Zerlegung) das lineare Gleichungssystem*

$$F'(x^{(i)}) \cdot d^{(i)} = -F(x^{(i)})$$

und setze

$$x^{(i+1)} = x^{(i)} + d^{(i)}.$$

(iv) *Setze $i = i + 1$ und gehe zu (ii).*

Bemerkung 4.3.7 **(Newtonverfahren als Fixpunktiteration)**
Man kann das Newtonverfahren als Fixpunktiteration

$$x^{(i+1)} = g(x^{(i)}), \qquad i = 0, 1, 2, \ldots \tag{4.5}$$

mit der Fixpunktfunktion

$$g(x) := x - F'(x)^{-1} \cdot F(x) \tag{4.6}$$

interpretieren.

Satz 4.3.8 **(Konvergenzsatz für das Newtonverfahren)**
Es sei $D \subseteq \mathbb{R}^n$ offen und $F : D \to \mathbb{R}^n$ besitze eine Nullstelle $\hat{x}$ in D. F sei stetig differenzierbar auf D und $F'(\hat{x})$ sei invertierbar. Dann gelten folgende Aussagen:

(i) *Es gibt ein $r > 0$, so dass das lokale Newton-Verfahren für alle Startwerte $x^{(0)} \in U_r(\hat{x})$ wohldefiniert ist und die Folge $\{x^{(i)}\}$ konvergiert gegen $\hat{x}$.*

(ii) *Die Konvergenzrate ist superlinear.*

(iii) *$\hat{x}$ ist die einzige Nullstelle in $U_r(\hat{x})$.*

(iv) *Erfüllt F' zusätzlich noch die Lipschitz-Bedingung*

$$\|F'(x) - F'(y)\| \leq L \cdot \|x - y\| \qquad \forall x, y \in D,$$

so ist die Folge $\{x^{(i)}\}$ sogar quadratisch konvergent.

Beweis: Wir zeigen hier nur die quadratische Konvergenzgeschwindigkeit des Newtonverfahrens. Die superlineare Konvergenz und die lokale Eindeutigkeit der Nullstelle kann gezeigt werden, indem man das Newtonverfahren als Fixpunktiteration auffasst.
Zur quadratischen Konvergenz: Anwendung des Mittelwertsatzes in Integralform (siehe Anhang)und Ausnutzung der Lipschitz-Bedingung liefert

$$\begin{aligned}
\|x^{(i+1)} - \hat{x}\| &= \|x^{(i)} - \hat{x} - F'(x^{(i)})^{-1} F(x^{(i)})\| \\
&= \|F'(x^{(i)})^{-1} \left(F(\hat{x}) - F(x^{(i)}) - F'(x^{(i)})(\hat{x} - x^{(i)}) \right) \| \\
&\leq \|F'(x^{(i)})^{-1}\| \cdot \left\| \int_0^1 F'(x^{(i)} + t(\hat{x} - x^{(i+1)}))(\hat{x} - x^{(i)}) dt - F'(x^{(i)})(\hat{x} - x^{(i)}) \right\| \\
&= \|F'(x^{(i)})^{-1}\| \cdot \left\| \int_0^1 \left(F'(x^{(i)} + t(\hat{x} - x^{(i)})) - F'(x^{(i)}) \right) \cdot (\hat{x} - x^{(i)}) dt \right\| \\
&\leq \|F'(x^{(i)})^{-1}\| \cdot \int_0^1 Lt \|\hat{x} - x^{(i)}\|^2 dt \\
&= \|F'(x^{(i)})^{-1}\| \cdot L \cdot \|\hat{x} - x^{(i)}\|^2 \cdot \int_0^1 t dt \\
&= \|F'(x^{(i)})^{-1}\| \cdot \frac{L}{2} \cdot \|\hat{x} - x^{(i)}\|^2.
\end{aligned}$$

Da $F'(\hat{x})$ invertierbar ist, existiert auch $F'(x^{(i)})^{-1}$ in einer Umgebung von $\hat{x}$ und ist dort beschränkt. Damit ist die quadratische Konvergenz gezeigt. □

Beispiel 4.3.9 (Funktion von Himmelblau)

Wir betrachten ein Beispiel aus der unrestringierten Optimierung. Gesucht ist ein lokales Minimum der Funktion

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2.$$

Aus der Theorie der nichtlinearen Optimierung ist bekannt, dass ein lokales Minimum $(\hat{x}, \hat{y})$ von f notwendig die Bedingung $\nabla f(\hat{x}, \hat{y}) = 0$ erfüllt, d.h. wir suchen eine Nullstelle der nichtlinearen Funktion

$$F(x, y) := \nabla f(x, y) = \begin{pmatrix} 4x(x^2 + y - 11) + 2(x + y^2 - 7) \\ 2(x^2 + y - 11) + 4y(x + y^2 - 7) \end{pmatrix},$$

welche die Jacobimatrix

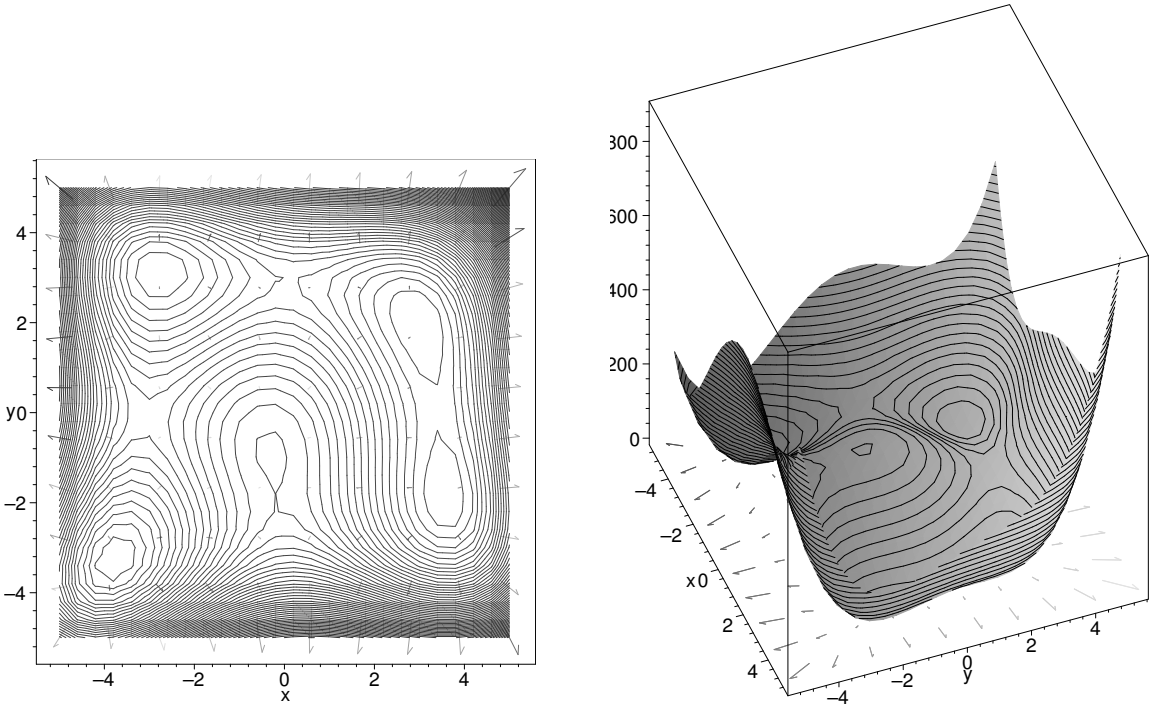
$$F'(x, y) = \nabla^2 f(x, y) = \begin{pmatrix} 12x^2 + 4y - 42 & 4(x + y) \\ 4(x + y) & 4x + 12y^2 - 26 \end{pmatrix}$$

besitzt. Ein Blick auf den Graphen von f zeigt, dass es in diesem Beispiel mehrere Nullstellen gibt, nämlich

- *4 lokale Minimalstellen (zugleich global) mit Funktionswert 0; darunter der Punkt* $(3,2)^\top$.

- *4 Sattelpunkte*

- *ein lokales Maximum in* $(-0.270845, -0.923039)^\top$



Als Abbruchkriterium des Newtonverfahrens verwenden wir $\|F(x,y)\| \leq 10^{-13}$ *und als Startpunkt* $(x^{(0)}, y^{(0)}) = (4, 2.5)$.

Das Newtonverfahren liefert das folgende Ergebnis:

ITER	X(1)	X(2)	||F(X)||	||DX||	P=1	P=2
0	0.4000000E+01	0.2500000E+01	0.1351240E+03	0.0000000E+00	0.0000000E+00	0.0000000E+00
1	0.3281417E+01	0.2056664E+01	0.2617493E+02	0.8443389E+00	0.2567589E+00	0.2296521E+00
2	0.3035131E+01	0.1988137E+01	0.2424694E+01	0.2556422E+00	0.1291689E+00	0.4499638E+00
3	0.3000634E+01	0.1999744E+01	0.4203618E-01	0.3639711E-01	0.1844451E-01	0.4974264E+00
4	0.3000000E+01	0.2000000E+01	0.1406811E-04	0.6837008E-03	0.3217815E-03	0.4704954E+00
5	0.3000000E+01	0.2000000E+01	0.1500787E-11	0.2200729E-06	0.1109167E-06	0.5039996E+00
6	0.3000000E+01	0.2000000E+01	0.0000000E+00	0.2440976E-13	0.0000000E+00	0.0000000E+00

Die letzten beiden Spalten testen die Folge $(x^{(i)}, y^{(i)})^\top$, $i = 0, 1, 2, \ldots$, *auf lineare (P=1 strebt gegen Wert* $\neq 0$, *P=2 explodiert), superlineare (P=1 strebt gegen* 0, *P=2 explodiert) und quadratische Konvergenz (P=1 strebt gegen* 0, *P=2 strebt gegen Wert* $\neq 0$*).*

Die Rechnungen zeigen eine quadratische Konvergenz des Newtonverfahrens gegen das Minimum $(3,2)^\top$.

Für andere Startwerte konvergiert das Newtonverfahren mitunter gegen andere Nullstellen von F.

In der Praxis stellt sich die Frage, wann das Newtonverfahren angehalten werden kann. Das Kriterium $\|F(x^{(i)})\| \leq tol$ ist zwar naheliegend, aber nicht skalierungsinvariant, d.h. durch Multiplikation von F mit einer Konstanten, lässt sich die Bedingung stets erfüllen und gibt mitunter keine Auskunft über die Güte der Iterierten $x^{(i)}$.
Als geeigneteres Konvergenzkriterium hat sich beispielsweise der **natürliche Monotonietest** bewährt, vgl. Deuflhard und Hohmann [DH91]. In jedem Iterationsschritt wird dabei geprüft, ob

$$\|\bar{d}^{(i+1)}\| \leq \Theta \|d^{(i)}\|$$

gilt, wobei meist $\Theta = 0.5$ gewählt wird. Darin bezeichnet $d^{(i)}$ die Newtonrichtung und $\bar{d}^{(i+1)}$ ist die Lösung des linearen Gleichungssystems

$$F'(x^{(i)})\bar{d}^{(i+1)} = -F(x^{(i+1)}).$$

Die Lösung dieses Gleichungssystems erfordert nur eine Vorwärts-Rückwärtssubstitution, da die LR-Zerlegung von $F'(x^{(i)})$ bereits bei der Berechnung von $x^{(i+1)}$ berechnet wurde.

Bemerkung 4.3.10 (Varianten des Newtonverfahrens)

In der Praxis werden häufig Varianten des Newtonverfahrens verwendet, die hauptsächlich versuchen, die Vorteile des Newtonverfahrens (superlineare bzw. quadratische Konvergenzgeschwindigkeit, Anwendbarkeit für $n > 1$) zu erhalten und die Nachteile des Newtonverfahrens zu umgehen. Die Hauptnachteile des Verfahrens bestehen in der i.a. nur lokalen Konvergenz des Verfahrens, d.h. sobald der Startwert nicht hinreichend nahe an der Nullstelle liegt, besteht die Gefahr der Divergenz des Verfahrens. Desweiteren wird die Jacobimatrix von F benötigt. Die Berechnung derselben kann sehr aufwändig sein und ist oft nur numerisch möglich, mitunter ist sie auch nicht einmal invertierbar.

- **Vereinfachtes Newtonverfahren:** *Um den Rechenaufwand zu reduzieren wird die Jacobimatrix F' nicht in jedem Iterationsschritt neu berechnet, sondern es wird vereinfachend die konstante Matrix $A_0 := F'(x^{(0)})$ für alle Iterationen (bzw. für eine bestimmte Anzahl von Iterationen) verwendet. Dadurch entsteht ein linear konvergentes Verfahren.*

- **Finite-Differenzen-Approximation:** *Die Jacobimatrix wird durch Differenzenquotienten approximiert, etwa durch den Vorwärtsdifferenzenquotienten*

$$\frac{\partial F_j}{\partial x_i}(x) \approx \frac{F_j(x + he_i) - F_j(x)}{h}$$

mit dem i-ten Einheitsvektor $e_i \in \mathbb{R}^n$.

- **Quasi-Newton-Verfahren:** *Quasi-Newton-Verfahren ersetzen die Jacobimatrix $F'(x^{(i)})$ im Newtonverfahren durch eine Matrix B_i, die eine geeignete Approximation an $F'(x^{(i)})$ darstellen soll. Bei Quasi-Newton-Verfahren wird die Approximation B_{i+1} durch eine geeignete Update-Formel aus B_i berechnet. Man erhält so ein superlinear konvergentes Verfahren.*

- **Globalisierung des Newtonverfahrens:** *Ein Hauptnachteil des Newtonverfahrens ist die nur lokale Konvergenz. Es gibt jedoch Ansätze, die eine Konvergenz des Newtonverfahrens für beliebige Startwerte garantieren (natürlich nur unter bestimmten Voraussetzungen). Die Globalisierung des Newtonverfahrens besteht in der Einführung einer Schrittweite $\alpha_i > 0$ bei der Berechnung von $x^{(i+1)}$ gemäß*

$$x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)}.$$

Die Schrittweite α_i wird hierbei so bestimmt, dass das Residuum $\|F\|$ beim Übergang von $x^{(i)}$ zu $x^{(i+1)}$ hinreichend stark abnimmt. Diese Technik ist aus der Theorie der nichtlinearen Optimierung bekannt und kann z.B. durch eine sogenannte Liniensuche für die eindimensionale Funktion

$$\varphi(\alpha) := \|F(x^{(i)} + \alpha d^{(i)})\|_2^2$$

realisert werden. Details finden sich z.B. in Geiger und Kanzow [GK99] und Kosmol [Kos93]. Globalisierte Newtonverfahren sind in der Regel so konstruiert, dass die Schrittweite $\alpha_i = 1$ akzeptiert wird, sobald $x^{(i)}$ den lokalen Konvergenzeinzugsbereich des lokalen Newtonverfahrens erreicht. Insofern erben die globalisierten Varianten die lokalen Konvergenzeigenschaften des Newtonverfahrens.

4.4 Sekantenverfahren

Wiederum im eindimensionalen Fall $n = 1$ verwendet man häufig auch das Sekantenverfahren. Sind zwei Näherungen $x^{(i-1)} \neq x^{(i)}$ gegeben, approximiert man die Funktion F lokal durch die Sekante $S(x)$, die durch die Punkte $(x^{(i-1)}, F(x^{(i-1)}))$ und $(x^{(i)}, F(x^{(i)}))$ verläuft, vgl. Abbildung 4.4. Die Sekante ist durch

$$S(x) = F(x^{(i-1)}) + \frac{x - x^{(i-1)}}{x^{(i)} - x^{(i-1)}}(F(x^{(i)}) - F(x^{(i-1)}))$$

gegeben. Die nächste Näherung $x^{(i+1)}$ ist durch die Nullstelle der Sekante gegeben: $S(x^{(i+1)}) = 0$. Diese lässt sich leicht ausrechnen:

$$\begin{aligned} x^{(i+1)} &= x^{(i)} - \left(\frac{x^{(i)} - x^{(i-1)}}{F(x^{(i)}) - F(x^{(i-1)})}\right) F(x^{(i)}) \\ &= \frac{x^{(i-1)}F(x^{(i)}) - x^{(i)}F(x^{(i-1)})}{F(x^{(i)}) - F(x^{(i-1)})}, \quad i = 1, 2, 3, \ldots \end{aligned}$$

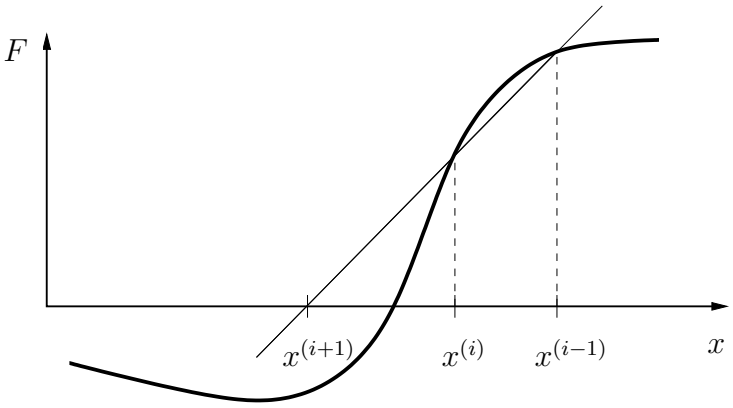


Abbildung 4.4: Sekantenverfahren: Ersetzung der Tangente durch Sekante.

Zum Start des Verfahrens werden **zwei** Anfangswerte $x^{(0)}$ und $x^{(1)}$ mit $F(x^{(0)}) \neq F(x^{(1)})$ benötigt. Ein Problem entsteht, wenn während der Iteration der Fall $F(x^{(i)}) \approx F(x^{(i-1)})$ auftritt. Deshalb sollte das Verfahren abgebrochen werden, wenn

$$|F(x^{(i)}) - F(x^{(i-1)}| \leq \varepsilon |F(x^{(i)})|$$

mit einer gegebenen Toleranz ε gilt.
Es gilt

Satz 4.4.1

Sei $\hat{x} \in (a,b)$ eine Nullstelle von F. Sei F zweimal stetig differenzierbar auf $[a,b]$ und $F'(\hat{x}) \neq 0$.
Dann gilt: Das Sekantenverfahren konvergiert **lokal** *gegen $\hat{x}$. Die Konvergenzordnung ist $p = (1+\sqrt{5})/2 \approx 1.618$.*

Beweis: siehe Lempio [Lem98, S.140] □

Gemäß Definition 4.0.4 ist das Sekantenverfahren also lokal superlinear konvergent.
Vorteile:

- Es werden nur Funktionsauswertungen benötigt (sehr einfach zu implementieren).
- Die Konvergenzordnung ist $p = (1+\sqrt{5})/2 \approx 1.618 > 1$. Die Konvergenzordnung ist formal zwar kleiner als die des Newtonverfahrens, berücksichtigt man jedoch den Aufwand zur Durchführung eines Iterationsschrittes, zeigt sich, dass das Sekantenverfahren nur eine Funktionsauswertung benötigt, während das Newtonverfahren eine Funktionsauswertung und eine Ableitung benötigt, die i.a. teurer als eine Funktionsauswertung ist. Um den gleichen Aufwand zu erhalten, können zwei Schritte des Sekantenverfahrens bei nur einem Newtonschritt durchgeführt werden. Damit erhielte man die Konvergenzordnung $p^2 \approx 2.618$.

Nachteile:

- Nur für $n = 1$ anwendbar.
- Das Verfahren konvergiert nur lokal und es ist instabil, falls $F(x^{(i)}) \approx F(x^{(i-1)})$ gilt.

Beispiel 4.4.2

Wir betrachten erneut das Problem, $\sqrt{2}$ als (positive) Nullstelle von $F(x) = x^2 - 2$ zu berechnen. Ausgabe:

```
Iteration 1 :    1.3333333333, Fehler:    0.0808802290
Iteration 2 :    1.4000000000, Fehler:    0.0142135624
Iteration 3 :    1.4146341463, Fehler:    0.0004205840
Iteration 4 :    1.4142114385, Fehler:    0.0000021239
Iteration 5 :    1.4142135621, Fehler:    0.0000000003
Iteration 6 :    1.4142135624, Fehler:    0.0000000000
```

Kapitel 5

Interpolation

Interpolationsaufgaben beschäftigen sich mit der Konstruktion einer Funktion, die an endlich vielen Stellen mit einer i.a. unbekannten oder schwer zu berechnenden Funktion übereinstimmt.
Eine typische Aufgabenstellung ist die Messung eines physikalischen oder technischen Vorgangs, welche $n+1 \in \mathbb{N}$ verschiedene reellwertige Stützwerte

$$\begin{array}{rcc|c|c|c} x & : & x_0 & x_1 & \cdots & x_n \\ \hline f(x) & : & f_0 & f_1 & \cdots & f_n \end{array} \tag{5.1}$$

ergebe. Die Funktion f, die dem Vorgang zu Grunde liegt, ist häufig nicht explizit bekannt und liegt somit nur an den Stützstellen $x_0, \ldots, x_n$ mit Funktionswerten $f_i = f(x_i), i = 0, \ldots, n$, vor. Um trotzdem mit Funktionswerten **zwischen** den Stützstellen arbeiten zu können und nicht jedesmal ein mitunter aufwändiges Experiment durchführren zu müssen, wird eine Approximation an die unbekannte Funktion f konstruiert, die zumindest die folgende Interpolationsaufgabe löst und ggf. weitere sinnvolle Eigenschaften besitzt (z.B. hinreichend hohe Glattheit).

Problem 5.0.1 **(Interpolationsaufgabe)**
Bestimme eine **geeignete** *Funktion* $g : \mathbb{R} \to \mathbb{R}$*, die die* **Interpolationsbedingungen**

$$g(x_i) = f_i, \quad i = 0, \ldots, n, \tag{5.2}$$

erfüllt, vgl. auch Abbildung 5.1.

Eine Funktion g, die die Interpolationsbedingungen (5.2) erfüllt, heißt **interpolierende Funktion**. Man sagt auch, g **interpoliert die Stützwerte (5.1)**.
Interpolationsaufgaben treten auch in der Computergrafik (zeichnen einer geeigneten Kurve durch vorgegebene Punkte), in technischen Geräten (z.B. Scanner oder Digitalkameras: echte Auflösung 600×1200 dpi, interpolierte Auflösung 9600×9600 dpi), in der Bildverarbeitung (Kompressionsalgorithmen) und in der Signalverarbeitung (Approximation von Signalen) auf.
Interpolierende Funktionen können auch zur Approximation von Funktionen dienen, deren Auswertung sehr kostenintensiv ist. In diesem Fall stellt sich die Frage, wie gut die Appro-

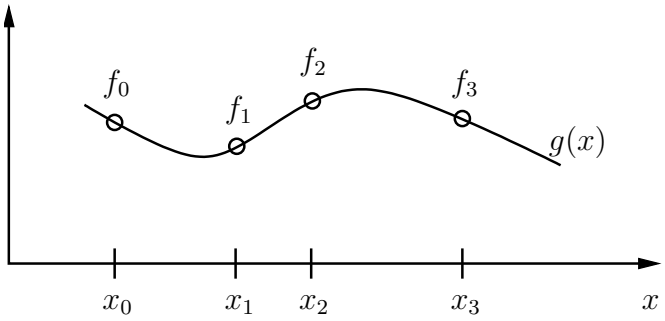


Abbildung 5.1: Interpolation: Stützwerte und interpolierende Funktion g.

ximation durch die interpolierende Funktion ist, d.h. wie groß der Approximationsfehler ist.

Beispiel 5.0.2

Die Funktion $f(x) = \exp(x)$ soll mittels Interpolation von Funktionswerten approximiert werden. Insbesondere soll f an der Stelle $x = 0.4$ approximiert werden.

(a) Wir verwenden die Funktionswerte $1 = f(0) = \exp(0)$ und $e = f(1) = \exp(1)$ und berechnen aus den Interpolationsbedingungen

$$\begin{aligned} 1 &= p(0) = a_0, \\ \exp(1) &= p(1) = a_0 + a_1, \end{aligned}$$

das interpolierende Polynom ersten Grades $p(x) = a_0 + a_1 x = 1 + (exp(1) - 1)x$. Damit ergibt sich an der Stelle $x = 0.4$ der Approximationsfehler

$$|p(0.4) - f(0.4)| \approx |1.6873127 - 1.4918247| = 0.1954880.$$

Dieser Fehler ist verhältnismäßig groß.

(b) Wir verwenden nun die Funktionswerte $1 = f(0) = \exp(0)$, $1.6487213 \approx f(0.5) = \exp(0.5)$ und $e = f(1) = \exp(1)$ und berechnen aus den Interpolationsbedingungen

$$\begin{aligned} 1 &= p(0) = a_0, \\ \exp(0.5) &= p(0.5) = a_0 + \frac{a_1}{2} + \frac{a_2}{4}, \\ \exp(1) &= p(1) = a_0 + a_1 + a_2, \end{aligned}$$

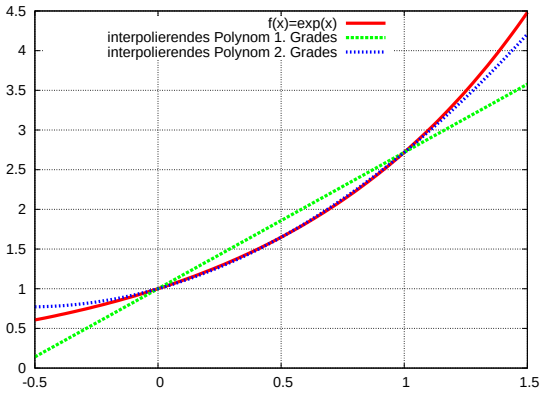
das interpolierende Polynom $p(x) = a_0 + a_1 x + a_2 x^2$ zweiten Grades mit

$$a_0 = 1, \quad a_1 = 4\exp(0.5) - exp(1) - 3, \quad a_2 = 2\exp(1) + 2 - 4\exp(0.5).$$

Damit ergibt sich an der Stelle $x = 0.4$ der Approximationsfehler

$$|p(0.4) - f(0.4)| \approx |1.4853099 - 1.4918247| = 0.0065148.$$

Dieser Fehler ist deutlich kleiner als in (a), vgl. nachstehende Abbildung.



5.1 Polynominterpolation

Bei der **Polynominterpolation** werden Polynome als interpolierende Funktion g gewählt. Ein Polynom p vom Höchstgrad r kann allgemein als

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_r x^r = \sum_{i=0}^{r} a_i x^i \tag{5.3}$$

mit den $r+1$ Koeffizienten $a_0, a_1, \ldots, a_r$ dargestellt werden. r bezeichnet des Grad des Polynoms. Unser Ziel ist es, ein **interpolierendes Polynom** p mit

$$p(x_i) = f_i, \quad i = 0, \ldots, n,$$

zu finden. Folgende Fragestellungen sind dabei relevant:

- Existenz und Eindeutigkeit eines interpolierenden Polynoms?
- Wie kann das (bzw. ein) Interpolationspolynom berechnet werden?
- Wie groß ist der Interpolationsfehler zwischen den Stützstellen?

Wir weisen die **Existenz eines interpolierenden Polynoms** konstruktiv nach, indem wir explizit ein interpolierendes Polynom konstruieren. Dabei setzen dabei voraus, dass die Knoten x_i, $i = 0, \ldots, n$, in (5.1) paarweise verschieden sind.

Definition 5.1.1 **(Lagrange Polynom)**
Für paarweise verschiedene Knoten x_i, $i = 0, \ldots, n$, ist für $k = 0, \ldots, n$ das k-te **Lagrange'sche Polynom** *definiert durch*

$$L_k(x) := \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}.$$

Für paarweise verschiedene Knoten x_i, $i = 0, \ldots, n$, sind die Lagrange'schen Polynome wohldefiniert, da der Nenner niemals Null wird. Man sieht ebenfalls sofort, dass $L_k, k = 0, \ldots, n$, Polynome vom Grad n sind, da jedes L_k sich aus n Produkten von Polynomen ersten Grades zusammensetzt. Desweiteren gilt für $k = 0, \ldots, n$,

$$L_k(x_j) = \begin{cases} 1 & \text{falls } j = k, \\ 0 & \text{falls } j \neq k. \end{cases}$$

Mit Hilfe dieser schönen Eigenschaft kann nun ein Interpolationspolynom angegeben werden:

$$p(x) = f_0 L_0(x) + f_1 L_1(x) + \ldots + f_n L_n(x) = \sum_{j=0}^{n} f_j L_j(x). \tag{5.4}$$

(Machen Sie sich klar, dass $p(x_k) = f_k$ für $k = 0, 1, \ldots, n$ gilt!) Mithin haben wir konstruktiv die Existenz einer Lösung nachgewiesen. Weiter gilt

Satz 5.1.2

Sind die Stützstellen $x_i, i = 0, \ldots, n$, paarweise verschieden, so gibt es genau ein interpolierendes Polynom vom Höchstgrad n.

Beweis: Die Existenz eines interpolierenden Polynoms vom Höchstgrad n wurde bereits durch das Polynom p in (5.4) konstruktiv nachgewiesen. Es verbleibt, die Eindeutigkeit nachzuweisen. Es seien also p_1 und p_2 interpolierende Polynome vom Höchstgrad n. Dann ist das Polynom $p = p_1 - p_2$ ebenfalls ein Polynom mit Höchstgrad n und es hat mindestens $n + 1$ verschiedene Nullstellen

$$p(x_i) = p_1(x_i) - p_2(x_i) = f_i - f_i = 0, \quad i = 0, \ldots, n.$$

Da ein Polynom mit Höchstgrad n, welches verschieden vom Nullpolynom ist, maximal n reelle Nullstellen haben kann (Fundamentalsatz der Algebra), muss $p = p_1 - p_2$ das Nullpolynom sein. Also gilt $p_1 = p_2$ und die Eindeutigkeit ist gezeigt. □

Zumindest theoretisch kann das interpolierende Polynom vom Höchstgrad n gemäß (5.4) berechnet werden. Es zeigt sich jedoch, dass die Berechnung gemäß (5.4) für gegebenes x sehr rundungsfehleranfällig und somit numerisch instabil ist. Deshalb sollte man den Wert eines Interpolationspolynoms an der Stelle x in der Praxis nicht über (5.4) berechnen, sondern auf die Newton'schen dividierten Differenzen des nächsten Abschnitts zurück greifen. Ein weiterer Nachteil von (5.4) ist, dass das Interpolationspolynom komplett neu berechnet werden muss, wenn nachträglich weitere Stützwerte hinzugefügt werden.

Bemerkung 5.1.3 **(Interpolation und Vandermonde-Matrix)**
Wertet man die Interpolationsbedingungen (5.2) mit dem Polynom (5.3) aus, folgt das **lineare Gleichungssystem**

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^r \\ 1 & x_1 & x_1^2 & \cdots & x_1^r \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_n & x_n^2 & \cdots & x_n^r \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_r \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix} \tag{5.5}$$

für die Koeffizienten $a_0, \ldots, a_r$. Die Matrix nennt man auch **Vandermonde-Matrix**. *Die Frage der Lösbarkeit der Interpolationsaufgabe ist also gleichbedeutend mit der Frage nach der Lösbarkeit des linearen Gleichungssystems. Falls der Grad r des Polynoms kleiner als n ist, besitzt das Gleichungssystem i.a. keine Lösung ($\rightarrow$ Gegenbeispiel?). Für $r = n$ und verschiedene Stützstellen ist die Vandermonde-Matrix regulär, da wir die Existenz und Eindeutigkeit eines interpolierenden Polynoms bereits bewiesen haben. Es empfiehlt sich nicht, das lineare Gleichungsystem zu lösen, da die Vandermonde-Matrix extrem schlecht konditioniert ist, wie wir bereits in Beispiel 2.4.1 gesehen haben. Für $r > n$ ist die Interpolationsaufgabe ebenfalls stets lösbar, allerdings nicht eindeutig.*

5.1.1 Newton'sche dividierte Differenzen

Wir leiten ein effizientes Verfahren zur Bestimmung des interpolierenden Polynoms her und stellen das Interpolationspolynom p hierfür in der Form

$$p(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \ldots + \alpha_n(x - x_0)\cdots(x - x_{n-1}) \tag{5.6}$$

dar. Beachte, dass die α_i, $i = 0, \ldots, n$, i.a. verschieden von den a_i, $i = 0, \ldots, n$, in der Darstellung (5.3) sind. Auswertung der Interpolationsbedingungen (5.2) für p führt auf das lineare Gleichungssystem

$$\begin{aligned} f_0 &= \alpha_0 \\ f_1 &= \alpha_0 + \alpha_1(x_1 - x_0) \\ f_2 &= \alpha_0 + \alpha_1(x_2 - x_0) + \alpha_2(x_2 - x_0)(x_2 - x_1) \\ &\vdots \\ f_n &= \alpha_0 + \alpha_1(x_n - x_0) + \ldots + \alpha_n(x_n - x_0)\cdots(x_n - x_{n-1}). \end{aligned}$$

Dieses Gleichungssystem hat Dreiecksgestalt und besitzt, falls die Stützstellen verschieden sind, eine eindeutige Lösung für $\alpha_0, \ldots, \alpha_n$. Die Lösung kann mit Hilfe der **Newton'schen dividierten Differenzen** $f[x_i, \ldots, x_{i+k}]$ angegeben werden, welche wir im folgenden herleiten möchten.

Es bezeichne $p_{x_i x_{i+1} \cdots x_{i+k}}$ das Polynom vom Höchstgrad k, welches die Stützwerte (x_j, f_j), $j = i, \ldots, i+k$, interpoliert. Nach Satz 5.1.2 ist dieses eindeutig bestimmt. Der folgende Satz gibt eine Rekursionsformel zur Berechnung an.

Satz 5.1.4 (Rekursionsformel von Neville)

Für die Interpolationspolynome $p_{x_i x_{i+1} \cdots x_{i+k}}$ gilt für $k \geq 1$ die Rekursionsformel

$$p_{x_i x_{i+1} \cdots x_{i+k}}(x) = \frac{(x - x_i) p_{x_{i+1} \cdots x_{i+k}}(x) - (x - x_{i+k}) p_{x_i x_{i+1} \cdots x_{i+k-1}}(x)}{x_{i+k} - x_i}.$$

Beweis: Der Beweis wird induktiv nach dem Grad k der interpolierenden Polynome geführt. Für $k = 0$ ist $p_{x_i}(x) = f_i$ das interpolierende Polynom.
Sei die Behauptung richtig für Polynome vom Grad $k-1$ ($k \geq 1$). Definiere das Polynom

$$q(x) := \frac{(x - x_i) p_{x_{i+1} \cdots x_{i+k}}(x) - (x - x_{i+k}) p_{x_i x_{i+1} \cdots x_{i+k-1}}(x)}{x_{i+k} - x_i},$$

welches offenbar den Grad k hat. Nach Induktionsvoraussetzung gilt:

$$\begin{aligned}
q(x_i) &= p_{x_i x_{i+1} \cdots x_{i+k-1}}(x_i) = f_i, \\
q(x_j) &= \frac{(x_j - x_i) p_{x_{i+1} \cdots x_{i+k}}(x_j) - (x_j - x_{i+k}) p_{x_i x_{i+1} \cdots x_{i+k-1}}(x_j)}{x_{i+k} - x_i} \\
&= \frac{(x_j - x_i) f_j - (x_j - x_{i+k}) f_j}{x_{i+k} - x_i} = f_j, \quad j = i+1, \ldots, i+k-1, \\
q(x_{i+k}) &= p_{x_{i+1} \cdots x_{i+k}}(x_{i+k}) = f_{i+k}.
\end{aligned}$$

Also interpoliert q genauso wie $p_{x_i x_{i+1} \cdots x_{i+k}}$ die Stützstellen (x_j, f_j), $j = i, \ldots, i+k$. Da das Interpolationspolynom nach Satz 5.1.2 eindeutig ist, gilt $q \equiv p_{x_i x_{i+1} \cdots x_{i+k}}$. □

Definition 5.1.5 (Newton'sche dividierte Differenz)

Der Koeffizient vor x^k in $p_{x_i x_{i+1} \cdots x_{i+k}}$ heißt Newton'sche dividierte Differenz und wird mit $f[x_i, \ldots, x_{i+k}]$ bezeichnet.

Mit Hilfe von Satz 5.1.4 erhalten wir eine Rekursionsformel für die Newton'schen dividierten Differenzen.

Satz 5.1.6 (Rekursionsformel für Newton'sche dividierte Differenzen)

Die Newtonschen dividierten Differenzen sind rekursiv durch

$$\begin{aligned}
f[x_i] &:= f_i, \\
f[x_i, x_{i+1}, \ldots, x_{i+k}] &:= \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}
\end{aligned}$$

gegeben und können nach folgendem Schema spaltenweise berechnet werden:

$$\begin{array}{lllllll} f_0 & = & f[x_0] & & & & \\ f_1 & = & f[x_1] & f[x_0,x_1] & & & \\ f_2 & = & f[x_2] & f[x_1,x_2] & f[x_0,x_1,x_2] & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ f_n & = & f[x_n] & f[x_{n-1},x_n] & f[x_{n-2},x_{n-1},x_n] & \cdots & f[x_0,\ldots,x_n] \end{array}$$

Der Aufwand des Schemas beträgt $\frac{1}{2}n(n+1)$ *Divisionen und* $n(n+1)$ *Subtraktionen.*

Beweis: Folgt direkt aus Satz 5.1.4 durch Koeffizientenvergleich. □

Ein wesentlicher Vorteil der dividierten Differenzen ist, dass zusätzliche Knoten leicht hinzugefügt werden können. Hierfür muss das Schema lediglich um eine weitere Zeile ergänzt werden.

Der folgende Satz klärt den Zusammenhang zwischen dem Interpolationspolynom in (5.6) und den Newton'schen dividierten Differenzen.

Satz 5.1.7

Das Interpolationspolynom p *zu den Stützstellen* (x_i, f_i), $i = 0,\ldots,n$, *ist durch (5.6) mit*

$$\alpha_i = f[x_0,\ldots,x_i], \quad i = 0,1,\ldots,n$$

gegeben.

Beweis: Es gilt die Darstellung

$$\begin{aligned} p &= p_{x_0\cdots x_n} \\ &= p_{x_0} + (p_{x_0x_1} - p_{x_0}) + (p_{x_0x_1x_2} - p_{x_0x_1}) + \ldots + (p_{x_0x_1\cdots x_n} - p_{x_0x_1\cdots x_{n-1}}). \end{aligned}$$

Das Polynom $p_{x_0x_1\cdots x_{k+1}} - p_{x_0x_1\cdots x_k}$ hat den Höchstgrad $k+1$, die $k+1$ Nullstellen $x_0,\ldots,x_k$, sowie per Definition den Koeffizienten $f[x_0,\ldots,x_{k+1}]$ für x^{k+1}. Da $p_{x_0x_1\cdots x_k}$ ein Polynom vom Grad k ist, gilt

$$p_{x_0x_1\cdots x_{k+1}}(x) - p_{x_0x_1\cdots x_k}(x) = f[x_0,\ldots,x_{k+1}](x-x_0)(x-x_1)\cdots(x-x_k),$$

woraus die Behauptung folgt. □

Das Polynom p in (5.6) kann mit Hilfe des **Horner-Algorithmus** effizient an einer Stelle x ausgewertet werden. Dazu wird p in geschachtelter Form

$$p(x) = \alpha_0 + (x-x_0)\left\{\alpha_1 + (x-x_1)\left[\alpha_2 + (\ldots + (\alpha_{n-1} + (x-x_{n-1})\alpha_n)\ldots)\right]\right\} \qquad (5.7)$$

geschrieben. Diese geschachtelte Form kann für ein gegebenes x effizient durch den nachstehenden Algorithmus berechnet werden:

Algorithmus 5.1.8 **(Horner Algorithmus)**

Gegeben: x, α_i, $i = 0, \ldots, n$.

Resultat: $v = p(x)$.

Setze $v = \alpha_n$.

for $i = n-1, \ldots, 0$*:*

$v := \alpha_i + v(x - x_i)$

end

Beispiel 5.1.9

Wir betrachten zwei Interpolationsaufgaben

(i)

i	1	2	3	4	5
x_i	-2	-1	0	1	2
f_i	0	0	10	0	0

(ii)

i	1	2	3	4	5	6	7	8	9	10	11	12	13
x_i	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
f_i	0	0	0	0	0	0	10	0	0	0	0	0	0

Abbildung 5.2 zeigt die zugehörigen Interpolationspolynome. Beispiel (ii) verdeutlicht eine grundsätzliche Problematik bei der Interpolation mit Polynomen: Mit wachsender Anzahl der Stützwerte wächst auch der Polynomgrad und das Interpolationspolynom weist starke Oszillationen auf, die zum Rand hin immer stärker werden. In der Praxis interpoliert man die gegebenen Stützwerte daher häufig **abschnittsweise** *durch Polynome mit nicht zu hohem Grad und „heftet“ die abschnittsweise definierten Polynome aneinander.*

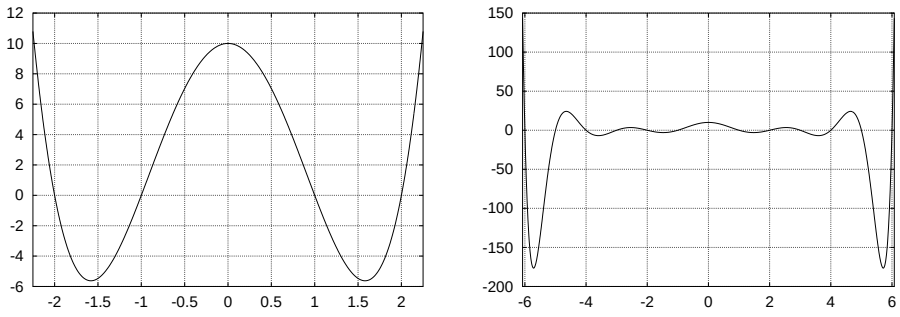


Abbildung 5.2: Interpolationspolynome für die Daten (i) (links) und (ii) (rechts).

Ist man nur am Wert des Interpolationspolynoms an einer festen Stelle x interessiert, kommt der **Algorithmus von Neville** zum Einsatz. Im Gegensatz zu den Newton'schen

dividierten Differenzen werden nicht die Koeffizienten des Interpolationspolynoms berechnet, sondern nur der Funktionswert an der Stelle x. Es wird wiederum vorausgesetzt, dass die Stützstellen x_i, $i = 0, \dots, n$, verschieden sind. Das Neville-Schema verwendet die Darstellung des Interpolationspolynoms aus Satz 5.1.4.

Algorithmus 5.1.10 **(Algorithmus von Neville)**

(0) Sei die feste Stelle $x \in \mathbb{R}$ gegeben. Setze

$$p_{x_i} := f_i, \quad i = 0, \dots, n.$$

(1) Berechne rekursiv die Werte

$$p_{x_i \cdots x_{i+k}} := \frac{(x - x_i) p_{x_{i+1} \cdots x_{i+k}} - (x - x_{i+k}) p_{x_i \cdots x_{i+k-1}}}{x_{i+k} - x_i}.$$

Die Berechnungsvorschrift kann in Tableauform („Neville-Schema“) zusammengefasst werden

$$\begin{array}{lllllll} f_0 & = & p_{x_0} & & & & \\ f_1 & = & p_{x_1} & p_{x_0 x_1} & & & \\ f_2 & = & p_{x_2} & p_{x_1 x_2} & p_{x_0 x_1 x_2} & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ f_n & = & p_{x_n} & p_{x_{n-1} x_n} & p_{x_{n-2} x_{n-1} x_n} & \cdots & p_{x_0 x_1 \cdots x_n} \end{array}$$

Der gesuchte Funktionswert des Interpolationspolynoms ist $p_{x_0 x_1 \cdots x_n}$.

5.1.2 Fehlerdarstellung interpolierender Polynome

Der folgende Satz gibt Auskunft über die Approximationsgüte des interpolierenden Polynoms vom Höchstgrad n an eine Funktion f zwischen den Stützstellen in (5.1).

Satz 5.1.11

Seien $f : [a, b] \to \mathbb{R}$ $(n+1)$-mal stetig differenzierbar und die Stützstellen $x_i \in [a, b]$, $i = 0, \dots, n$, verschieden. Für das Interpolationspolynom p höchstens n-ten Grades mit $p(x_i) = f(x_i)$, $i = 0, \dots, n$, gilt

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \cdot \prod_{i=0}^{n} (x - x_i) \tag{5.8}$$

mit einer von x abhängigen Stelle ξ mit

$$\min\{x_0, \dots, x_n, x\} < \xi < \max\{x_0, \dots, x_n, x\}.$$

Die Fehlerdarstellung in (5.8) hängt von der Funktion f und dem Polynom

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

ab, wobei letzteres nur von den Stützstellen x_i, $i = 0, \ldots, n$, abhängt. Insbesondere folgt aus ihr die Abschätzung

$$\begin{aligned} |f(x) - p(x)| &\leq \frac{1}{(n+1)!}\|f^{(n+1)}\|_\infty \cdot |\omega(x)| \\ &\leq \frac{1}{(n+1)!}\|f^{(n+1)}\|_\infty \cdot \|\omega\|_\infty \end{aligned} \tag{5.9}$$

für alle $x \in I := [\min\{x_0, \ldots, x_n\}, \max\{x_0, \ldots, x_n\}]$. Hierin bezeichnet $\|\cdot\|_\infty$ die Supremumsnorm, die für eine stetige Funktion $g : [a, b] \to \mathbb{R}$ definiert ist durch

$$\|g\|_\infty := \max_{a \leq x \leq b} |g(x)|.$$

Falls die Stützstellen x_i, $i = 0, \ldots, n$, frei wählbar sind, kann der Approximationsfehler in (5.9) für gegebenes f minimiert werden, indem die Stützstellen x_i, $i = 0, \ldots, n$, optimal gewählt werden. Optimal bedeutet in diesem Zusammenhang, dass

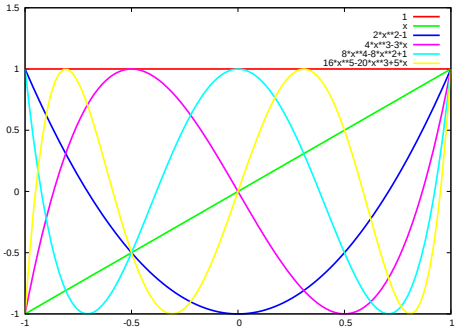
$$\|\omega\|_\infty = \max_{\min\{x_0,\ldots,x_n\} \leq x \leq \max\{x_0,\ldots,x_n\}} |\omega(x)|$$

durch geeignete Wahl von paarweise verschiedenen Stütztellen x_i, $i = 0, \ldots, n$, minimal wird. Es gilt also, paarweise verschiedene Stützstellen x_i, $i = 0, \ldots, n$, zu bestimmen, die ω bzgl. der Supremumsnorm minimieren. Es zeigt sich, dass die Nullstellen der sogenannten **Tschebyscheff-Polynome**

$$T_n(x) := \cos(n \cdot \arccos(x)), \qquad n = 0, 1, 2, \ldots,$$

genau dies leisten (wenn man das Problem zuvor auf das Intervall $[-1, 1]$ transformiert). Die Tschebyscheff-Polynome bis zum Grad 5 lauten wie folgt:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_5(x) &= 16x^5 - 20x^3 + 5x \end{aligned}$$



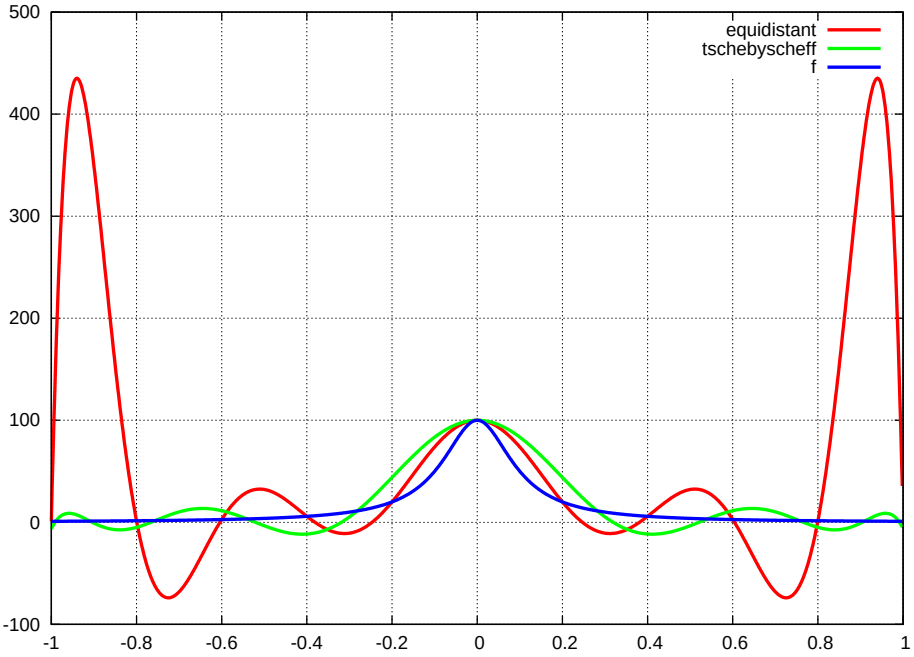
Beispiel 5.1.12

Betrachte

$$f(x) = \frac{1}{x^2 + 0.01}.$$

Die folgende Abbildung zeigt das interpolierende Polynom p_{eq} zu den äquidistanten Stützstellen $x_i = -1 + i \cdot h$, $i = 0, \dots, 10$, mit $h = 2/10$, sowie das interpolierende Polynom zu den Tschebyscheff-Stützstellen

$$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \qquad i = 0, \dots, 10.$$



Es ist deutlich zu sehen, dass das auf Tschebyscheff-Stützstellen basierende Interpolationspolynom die Funktion f besser (im Sinne der Supremumsnorm) approximiert als das auf äquidistanten Stützstellen basierende Interpolationspolynom. Insbesondere am Rand oszilliert letzteres stark.

5.2 Splineinterpolation

Im Computer Aided Design (CAD) oder in anderen Interpolationsaufgaben sind vor allem „visuell ansprechende“ interpolierende Funktionen erwünscht. Wir haben in Beispiel 5.1.9 gesehen, dass Polynominterpolation hier nur bedingt geeignet ist, da mit wachsendem Polynomgrad starke Oszillationen auftreten können. Polynome mit niedrigem Polynomgrad hingegen sehen angenehm aus. Dies motiviert die Verwendung von sogenannten **Splines**. Ein Spline besteht dabei aus abschnittsweise definierten Polynomen, die an den Stützstellen unter Beachtung von Glattheitsanforderungen zusammengesetzt werden.

Definition 5.2.1 **(Spline vom Grad k)**

Eine Funktion s heißt **Spline vom Grad k auf** $[a,b]$, $k \in \mathbb{N}$, *falls s folgende Bedingungen erfüllt:*

- *s ist $(k-1)$-mal stetig differenzierbar in $[a,b]$.*
- *Es gibt eine Partition $a = x_0 < x_1 < \ldots < x_n = b$ von $[a,b]$, so dass s in jedem Teilintervall $[x_i, x_{i+1}]$, $0 \le i \le n-1$, ein Polynom vom Höchstgrad k ist.*

Für $\mathbb{G} = \{x_0 < x_1 < \ldots < x_n\}$ bezeichnet

$$\mathcal{S}_k(\mathbb{G}) := \{s : [x_0, x_n] \to \mathbb{R} \mid s \text{ ist Spline vom Grad } k \text{ auf } [x_0, x_n]\}$$

den Raum der Splines vom Grad k mit Stützstellen in $\mathbb{G}$.
Ein **interpolierender Spline vom Grad** k *für die Daten (x_i, f_i), $i = 0, \ldots, n$, ist ein Spline s vom Grad k, der die Interpolationsbedingungen $s(x_i) = f_i$, $i = 0, \ldots, n$, erfüllt.*

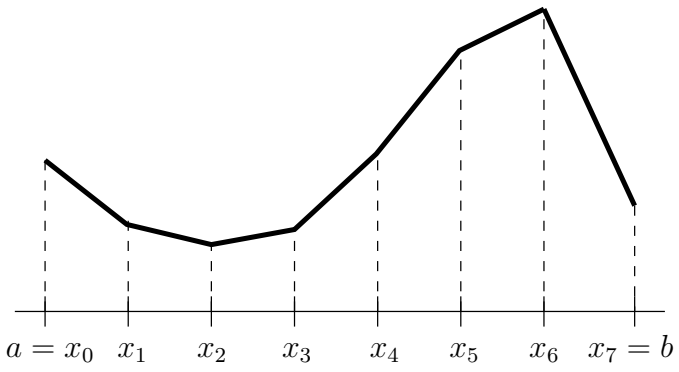
Beispiel 5.2.2 **(Spline vom Grad 1)**

Eine stetige, stückweise lineare Funktion auf $[a,b]$ ist ein Spline vom Grad 1. Offenbar ist der interpolierende Spline s vom Grad 1 für (x_i, f_i), $i = 0, \ldots, n$, eindeutig bestimmt durch

$$s(x) = \begin{cases} s_0(x), & x_0 \le x \le x_1, \\ s_1(x), & x_1 \le x \le x_2, \\ \vdots & \\ s_{n-1}(x), & x_{n-1} \le x \le x_n \end{cases}$$

mit

$$s_i(x) = s\big|_{[x_i, x_{i+1}]}(x) = f_i + \frac{x - x_i}{x_{i+1} - x_i}(f_{i+1} - f_i), \quad i = 0, \ldots, n-1.$$



Im folgenden werden wir uns speziell mit Splines vom Grad 3, den sogenannten **kubischen Splines**, beschäftigen. Kubische Splines werden insbesondere in der Computergrafik verwendet, um Funktionsverläufe zu erzeugen, die für das menschliche Auge „glatt“ erscheinen. Der Hintergrund ist, dass das menschliche Auge Unstetigkeiten in der Krümmung

(also i. W. in der zweiten Ableitung) noch wahrnehmen kann, während Funktionen mit stetiger Krümmung (also stetiger zweiter Ableitung) als glatt aufgefasst werden.

Zur Motivation betrachten wir folgendes Problem:
Ein Balken soll an den Stellen (5.1) durch Lager so eingespannt werden, dass dort nur Kräfte senkrecht zur Biegelinie wirken. Er wird dann eine Lage annehmen, die durch eine minimale Biegeenergie $E(s)$ charakterisiert ist, wobei die Funktion s die Biegekurve bezeichnet. Die Biegeenergie ist durch

$$E(s) = c \int_{x_0}^{x_n} \frac{s''(x)^2}{\sqrt{1+s'(x)^2}^5} dx$$

mit einer Konstanten c gegeben. Die Funktion

$$\kappa(x) = \frac{s''(x)}{\sqrt{1+s'(x)^2}^3}$$

beschreibt gerade die **Krümmung der Funktion** s. Für $|s'(x)| << 1$ gilt näherungsweise

$$E(s) \approx c \int_{x_0}^{x_n} s''(x)^2 dx. \tag{5.10}$$

Die Aufgabe besteht nun darin, eine Funktion s zu finden, die das Integral in (5.10) und damit auch (näherungsweise) die Biegeenergie unter den Nebenbedingungen $s(x_i) = f_i, i = 0, \dots, n$ minimiert.

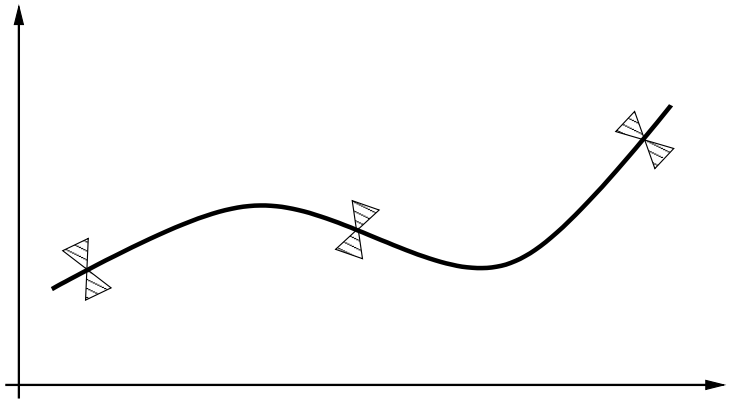


Abbildung 5.3: Spline vom Grad 3 (kubischer Spline): Kurve mit minimaler Krümmung durch gegebene Punkte.

Dies leisten sogenannte Splines, wie der folgende Satz aussagt.

Satz 5.2.3 **(Minimaleigenschaft interpolierender kubischer Splines)**
Ein interpolierender kubischer Spline s erfüllt

$$\int_{x_0}^{x_n} s''(x)^2 dx \le \int_{x_0}^{x_n} g''(x)^2 dx \qquad \forall g \in \mathcal{F},$$

mit

$$\mathcal{F} := \{g : [x_0, x_n] \to \mathbb{R} \mid g \text{ ist stetig differenzierbare interpolierende Funktion, deren zweite Ableitung in jedem Intervall } (x_i, x_{i+1}) \text{ stetig mit existierenden Grenzwerten } \lim_{x\to x_i^+} f''(x) \text{ und } \lim_{x\to x_{i+1}^-} f''(x) \text{ ist}\},$$

wenn noch **eine** *der folgenden Zusatzbedingungen gilt:*

(i) $s''(x_0) = s''(x_n) = 0$ *(s heißt* **natürlicher Spline***); (geometrisch: die Krümmung in a und b ist Null, daher kann der natürliche Spline außerhalb von* $[a, b]$ *linear fortgesetzt werden);*

(ii) s und g sind periodisch mit Periode $x_n - x_0$ *(s heißt* **periodischer Spline***);*

(iii) $f'_0 = g'(x_0) = s'(x_0)$, $f'_n = g'(x_n) = s'(x_n)$ *für gegebenes* f'_0 *und* f'_n *(s heißt* **Spline mit hermitescher Datenvorgabe***);*

s ist eindeutig bestimmt.

Konstruktion interpolierender kubischer Splines

Ziel ist es nun, einen die Stützstellen 5.1 interpolierenden kubischen Spline zu konstruieren. Nach Definition ist ein interpolierender kubischer Spline s für $i = 0, \ldots, n-1$ im Intervall $[x_i, x_{i+1}]$ ein Polynom s_i vom Höchstgrad 3, etwa

$$s(x) = \begin{cases} s_0(x), & x_0 \le x \le x_1, \\ s_1(x), & x_1 \le x \le x_2, \\ \vdots & \\ s_{n-1}(x), & x_{n-1} \le x \le x_n \end{cases} \tag{5.11}$$

mit

$$s_i(x) = s\big|_{[x_i, x_{i+1}]}(x) = \alpha_i + \beta_i(x - x_i) + \frac{1}{2}\gamma_i(x - x_i)^2 + \frac{1}{6}\delta_i(x - x_i)^3. \tag{5.12}$$

Weiterhin müssen die folgenden Bedingungen erfüllt sein:

$$\begin{aligned} s_0(x_0) &= f_0, \\ s_i(x_{i+1}) = s_{i+1}(x_{i+1}) &= f_{i+1}, \qquad i = 0, 1, \ldots, n-2, \\ s_{n-1}(x_n) &= f_n, \\ s'_i(x_{i+1}) &= s'_{i+1}(x_{i+1}), \qquad i = 0, 1, \ldots, n-2, \\ s''_i(x_{i+1}) &= s''_{i+1}(x_{i+1}), \qquad i = 0, 1, \ldots, n-2. \end{aligned}$$

Aus der Darstellung (5.12) folgt

$$\gamma_i = s_i''(x_i) = s''(x_i), \qquad i = 0, 1, \dots, n-1.$$

Zusätzlich definieren wir $\gamma_n := s''(x_n) = s_{n-1}''(x_n)$ und

$$h_{i+1} := x_{i+1} - x_i, \qquad i = 0, 1, \dots, n-1.$$

Zunächst wird gezeigt, wie s_i (und damit auch s) berechnet werden kann, wenn die sogenannten **Momente** γ_i, $i = 0, \dots, n$, bekannt sind.

(a) Aus $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}) = \gamma_{i+1}$, $i = 0, \dots, n-1$, folgt

$$s_i''(x_{i+1}) = \gamma_i + \delta_i h_{i+1} = \gamma_{i+1} = s_{i+1}''(x_{i+1}).$$

Auflösen nach δ_i liefert die Darstellung

$$\delta_i = \frac{\gamma_{i+1} - \gamma_i}{h_{i+1}}, \qquad i = 0, \dots, n-1. \tag{5.13}$$

(b) Aus $s_i(x_i) = f_i$, $i = 0, \dots, n-1$, folgt sofort

$$\alpha_i = f_i, \qquad i = 0, \dots, n-1.$$

Aus $s_i(x_{i+1}) = f_{i+1}$, $i = 0, \dots, n-1$, folgt

$$s_i(x_{i+1}) = f_i + \beta_i h_{i+1} + \frac{\gamma_i}{2} h_{i+1}^2 + \frac{\gamma_{i+1} - \gamma_i}{6 h_{i+1}} h_{i+1}^3 = f_{i+1}.$$

Auflösen nach β_i liefert die Darstellung

$$\beta_i = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{2\gamma_i + \gamma_{i+1}}{6} h_{i+1}, \qquad i = 0, \dots, n-1. \tag{5.14}$$

Zusammenfassend sind die Koeffizienten α_i, β_i und δ_i in (5.12) nach (5.13) und (5.14) gegeben durch

$$\begin{aligned}
h_{i+1} &= x_{i+1} - x_i, && i = 0, \dots, n-1,\\
\alpha_i &= f_i, && i = 0, \dots, n-1,\\
\beta_i &= \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{2\gamma_i + \gamma_{i+1}}{6} h_{i+1}, && i = 0, \dots, n-1,\\
\delta_i &= \frac{\gamma_{i+1} - \gamma_i}{h_{i+1}}, && i = 0, \dots, n-1,
\end{aligned}$$

falls die Momente γ_i, $i = 0, \dots, n$, bekannt sind. Es verbleibt, die Momente γ_i, $i = 0, \dots, n$, zu berechnen. Hierzu werden die Bedingungen $s_i'(x_i) = s_{i-1}'(x_i)$, $i = 1, \dots, n-1$, ausgenutzt. Diese führen auf

$$\beta_i = \beta_{i-1} + \gamma_{i-1} h_i + \frac{1}{2} \delta_{i-1} h_i^2, \qquad i = 1, \dots, n-1.$$

Einsetzen von β_i, β_{i-1} und δ_{i-1} aus (5.13) und (5.14) liefert nach etwas Rechnung

$$\frac{h_i}{h_i+h_{i+1}}\gamma_{i-1} + 2\gamma_i + \frac{h_{i+1}}{h_i+h_{i+1}}\gamma_{i+1} = \frac{6}{h_i+h_{i+1}}\left(\frac{f_{i+1}-f_i}{h_{i+1}} - \frac{f_i-f_{i-1}}{h_i}\right) \tag{5.15}$$

für $i = 1, 2, \ldots, n-1$. (5.15) definiert insgesamt $n-1$ lineare Gleichungen für die $n+1$ Momente γ_i, $i = 0, \ldots, n$. Wir benötigen also noch zwei weitere Bedingungen, um eine eindeutige Lösung zu ermöglichen. Hier kommt **eine** der Bedingungen (i), (ii) oder (iii) aus Satz 5.2.3 ins Spiel und je nachdem, welche Bedingung gewählt wird, entsteht ein natürlicher Spline, ein periodischer Spline oder ein Spline mit hermitescher Datenvorgabe. In jedem Fall führt die Auswertung von (5.15) zusammen mit einer der Bedingungen (i), (ii) oder (iii) aus Satz 5.2.3 jeweils auf ein lineares Gleichungssystem der Form

$$A \cdot \gamma = d, \quad \gamma := (\gamma_0, \ldots, \gamma_n)^\top,\ d = (d_0, \ldots, d_n)^\top \tag{5.16}$$

für die Momente γ_i, $i = 0, \ldots, n$. Wir diskutieren die einzelnen Fälle und die daraus entstehenden linearen Gleichungssysteme.

Natürlicher Spline

Es wird $s''(x_0) = s''(x_n) = 0$ gefordert. Per Definition der Momente führt dies auf die beiden Gleichungen $\gamma_0 = 0$ und $\gamma_n = 0$. Zusammen mit (5.15) ist das resultierende Gleichungssystem (5.16) dann gegeben durch

$$\begin{aligned}
A &= \begin{pmatrix}
1 & & & & & & \\
\mu_1 & 2 & \lambda_1 & & & & \\
 & \mu_2 & 2 & \lambda_2 & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \mu_{n-2} & 2 & \lambda_{n-2} & \\
 & & & & \mu_{n-1} & 2 & \lambda_{n-1} \\
 & & & & & & 1
\end{pmatrix}, \\
d_i &= \begin{cases}
0, & \text{für } i = 0 \text{ und } i = n, \\
\dfrac{6}{h_i+h_{i+1}}\left[\dfrac{f_{i+1}-f_i}{h_{i+1}} - \dfrac{f_i-f_{i-1}}{h_i}\right], & \text{für } i = 1, \ldots, n-1,
\end{cases} \\
\lambda_i &= \frac{h_{i+1}}{h_i+h_{i+1}}, \quad i = 1, \ldots, n-1, \\
\mu_i &= 1 - \lambda_i, \qquad i = 1, \ldots, n-1.
\end{aligned}$$

Periodischer Spline

Für einen periodischen Spline muss zunächst $f_0 = f_n$ gelten. Weiter wird gefordert, dass $s'(x_0) = s'(x_n)$ und $s''(x_0) = s''(x_n)$ gelten. Mit den obigen Bezeichnungen führt dies auf die beiden Gleichungen

$$0 = s''(x_0) - s''(x_n) = \gamma_0 - \gamma_n$$

und

$$0 = s'(x_0) - s'(x_n) = \beta_0 - (\beta_{n-1} + \gamma_{n-1}h_n + \frac{1}{2}\delta_{n-1}h_n^2).$$

Einsetzen von β_0, β_{n-1} und δ_{n-1} aus (5.13) und (5.14) liefert nach etwas Rechnung die Gleichung

$$2\frac{h_1}{h_1+h_n}\gamma_0 + \frac{h_1}{h_1+h_n}\gamma_1 + \frac{h_n}{h_1+h_n}\gamma_{n-1} + 2\frac{h_n}{h_1+h_n}\gamma_n = \frac{6}{h_1+h_n}\left(\frac{f_1-f_0}{h_1} - \frac{f_n-f_{n-1}}{h_n}\right).$$

Zusammen mit (5.15) ist das resultierende Gleichungssystem (5.16) dann gegeben durch

$$\begin{aligned}
A &= \begin{pmatrix}
1 & & & & & & -1 \\
\mu_1 & 2 & \lambda_1 & & & & \\
 & \mu_2 & 2 & \lambda_2 & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \mu_{n-2} & 2 & \lambda_{n-2} & \\
 & & & & \mu_{n-1} & 2 & \lambda_{n-1} \\
2\lambda_n & \lambda_n & & & & \mu_n & 2\mu_n
\end{pmatrix}, \\
d_i &= \begin{cases}
0, & \text{für } i = 0, \\
\dfrac{6}{h_i+h_{i+1}}\left[\dfrac{f_{i+1}-f_i}{h_{i+1}} - \dfrac{f_i-f_{i-1}}{h_i}\right], & \text{für } i = 1,\dots,n-1, \\
\dfrac{6}{h_1+h_n}\left[\dfrac{f_1-f_0}{h_1} - \dfrac{f_n-f_{n-1}}{h_n}\right], & \text{für } i = n,
\end{cases} \\
\lambda_i &= \frac{h_{i+1}}{h_i+h_{i+1}}, \quad i = 1,\dots,n-1, \quad \lambda_n = \frac{h_1}{h_1+h_n}, \\
\mu_i &= 1 - \lambda_i, \qquad i = 1,\dots,n.
\end{aligned}$$

Spline mit hermitescher Datenvorgabe

In x_0 und x_n werden neben den Funktionswerten $s(x_0) = f_0$ und $s(x_n) = f_n$ auch Ableitungen $s'(x_0) = f_0'$ und $s'(x_n) = f_n'$ vorgegeben. Mit den obigen Bezeichnungen führt dies auf die beiden Gleichungen

$$f_0' = s'(x_0) = \beta_0 = \frac{f_1-f_0}{h_1} - \frac{2\gamma_0+\gamma_1}{6}h_1$$

und

$$f_n' = s'(x_n) = \beta_{n-1} + \gamma_{n-1}h_n + \frac{1}{2}\delta_{n-1}h_n^2.$$

Einsetzen von β_0, β_{n-1} und δ_{n-1} aus (5.13) und (5.14) liefert nach etwas Rechnung die Gleichung

$$\gamma_{n-1} + 2\gamma_n = \frac{6}{h_n}\left(f_n' - \frac{f_n-f_{n-1}}{h_n}\right).$$

Zusammen mit (5.15) ist das resultierende Gleichungssystem (5.16) dann gegeben durch

$$
\begin{aligned}
A &= \begin{pmatrix} 2 & 1 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & 1 & 2 \end{pmatrix}, \\
d_i &= \begin{cases} \dfrac{6}{h_1}\left[\dfrac{f_1 - f_0}{h_1} - f_0'\right], & \text{für } i = 0, \\ \dfrac{6}{h_i + h_{i+1}}\left[\dfrac{f_{i+1} - f_i}{h_{i+1}} - \dfrac{f_i - f_{i-1}}{h_i}\right], & \text{für } i = 1, \ldots, n-1, \\ \dfrac{6}{h_n}\left[f_n' - \dfrac{f_n - f_{n-1}}{h_n}\right], & \text{für } i = n, \end{cases} \\
\lambda_i &= \frac{h_{i+1}}{h_i + h_{i+1}}, \quad i = 1, \ldots, n-1, \\
\mu_i &= 1 - \lambda_i, \qquad i = 1, \ldots, n-1.
\end{aligned}
$$

Im Falle des natürlichen Splines und des Splines mit hermitescher Datenvorgabe hat die Matrix A eine **tridiagonale Struktur** und ist wegen $\lambda_i + \mu_i = 1 < 2$ **strikt diagonaldominant**. Bei der numerischen Lösung wird diese tridiagonale Struktur ausgenutzt, um Rechenzeit zu sparen.

5.2.1 Fehlerdarstellung interpolierender kubischer Splines

Wie bei der Polynominterpolation interessieren wir uns für Approximationseigenschaften von kubischen Splines bezüglich einer gegebenen Funktion f. Zusätzlich zur Supremumsnorm $\|\cdot\|_\infty$ definieren wir die L_2-Norm

$$\|g\|_2 := \left(\int_a^b |g(t)|^2 dt\right)^{1/2}$$

für eine quadratintegrierbare Funktion $g : [a, b] \to \mathbb{R}$. Es gilt folgender Satz, der hier nicht bewiesen werden soll:

Satz 5.2.4

Gegeben seien $a = x_0 < x_1 < \ldots < x_n = b$ und eine zweimal stetig differenzierbare Funktion $f : [a, b] \to \mathbb{R}$.

Der interpolierende kubische Spline s mit

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

und einer der Bedingungen (i), (ii) oder (iii) aus Satz 5.2.3 erfüllt die Fehlerabschätzungen

$$\begin{aligned} \|f-s\|_2 &\leq \frac{1}{4}\|f''\|_2 h_{max}^2, \\ \|f-s\|_\infty &\leq \frac{1}{2}\|f''\|_2 h_{max}^{3/2}, \end{aligned}$$

wobei $h_{max} = \max\limits_{i=0,\dots,n-1}(x_{i+1}-x_i)$ *sei.*

Kapitel 6

Numerische Integration

Viele praktisch relevante Anwendungen, wie z.B. die Lösung einer elliptischen partiellen Differentialgleichung mit der Finite-Element-Methode oder die Fourieranalyse, erfordern die Auswertung von bestimmten Integralen der Form

$$I[f] := \int_a^b f(x)dx, \quad a, b \in \mathbb{R}, a < b. \tag{6.1}$$

Da diese Integrale oftmals nicht exakt berechnet werden können, betrachte z.B.

$$\int_0^{\pi/2} \left[1 - \left(\frac{x}{r}\right)^2 \sin^2\theta\right]^{1/2} d\theta,$$

werden numerische Methoden zur approximativen Berechnung von (6.1) benötigt.
Zur numerischen Berechnung des Integrals (6.1) verwendet man in der Regel den Ansatz

$$\int_a^b f(x)dx \approx \sum_{i=0}^{N} w_i f(x_i)$$

mit $N \in \mathbb{N}_0$, Gewichten w_i und Stützstellen

$$a \leq x_0 \leq x_1 \leq \ldots \leq x_N \leq b.$$

Die Aufgabe besteht nun darin, geeignete Werte für w_i und x_i zu finden, so dass die sogenannte **Quadraturformel**

$$Q[f] := \sum_{i=0}^{N} w_i f(x_i) \tag{6.2}$$

den Wert des Integrals möglichst gut approximiert.

Beispiel 6.0.1 (Trapezregel)
Die Trapezregel basiert auf der linearen Interpolation der Stützstellen $(a, f(a))$ *und* $(b, f(b))$*, vgl. Abbildung 6.1.*

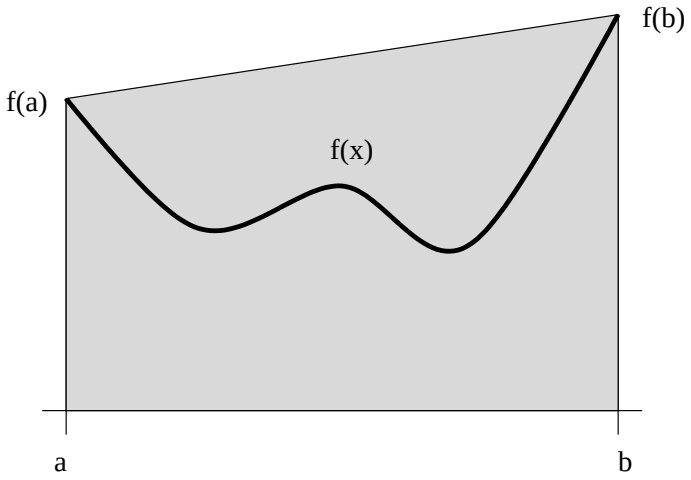


Abbildung 6.1: Trapezregel

Der Integrand f wird dann durch das lineare Interpolationspolynom

$$p_1(x) = f(a) + \frac{x-a}{b-a}(f(b) - f(a)) \tag{6.3}$$

ersetzt. Das resultierende Integral über p_1 kann berechnet werden und lautet

$$\int_a^b f(x)dx \approx \int_a^b p_1(x)dx = \frac{b-a}{2}(f(a) + f(b)) =: T_1[f].$$

Geometrisch ist dies gerade der Flächeninhalt des Trapezes, welches durch die Punkte $(a,0), (a,f(a)), (b,f(b)), (b,0)$ gegeben ist, vgl. Abbildung 6.1. Die Trapezregel $T_1[f]$ ist also eine spezielle Quadraturformel (6.2) mit

$$N = 1,\ w_0 = w_1 = \frac{b-a}{2},\ x_0 = a,\ x_1 = b.$$

Ein Maß für die Güte der Approximation, welches später noch häufiger auftreten wird, ist u.a. der Exaktheitsgrad einer Quadraturformel.

Definition 6.0.2 **(Exaktheitsgrad einer Quadraturformel)**
Eine Quadraturformel $Q[f]$ hat **Exaktheitsgrad** *p, falls alle Polynome bis zum Grad p exakt integriert werden.*

Durch einfaches Nachrechnen zeigt sich, dass die Trapezregel den Exaktheitsgrad eins hat. Polynome ersten Grades werden also durch die Trapezregel exakt integriert.
Wie Abbildung 6.1 bereits erahnen lässt, ist die Approximation von f durch p_1 im Falle der Trapezregel i.a. nicht sehr genau. Häufig verwendet man daher sogenannte **zusammengesetzte Quadraturformeln**. Dabei wird das Intervall $[a,b]$ (in der Regel) äquidistant

mit Schrittweite $h = (b-a)/n$ für $n \in \mathbb{N}$ unterteilt und in jedem Teilintervall $[x_i, x_{i+1}]$ mit $x_i = a + ih$, $i = 0, \ldots, n-1$, wird dann eine Quadraturformel der Form (6.2) auf

$$I_i[f] = \int_{x_i}^{x_{i+1}} f(x)dx, \qquad i = 0, \ldots, n-1,$$

angewendet. Eine Approximation von $I[f]$ ergibt sich dann durch Summation über alle Teilintervalle:

$$I[f] = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx \approx \sum_{i=0}^{n-1} Q[f|_{[x_i,x_{i+1}]}] = Q_n[f].$$

Beispiel 6.0.3 **(zusammengesetzte Trapezregel)**
Das Intervall $[a, b]$ wird durch Einführung von äquidistanten Stützstellen

$$x_i = a + ih,\ i = 0, \ldots, n,\ h = \frac{b-a}{n} \tag{6.4}$$

in Teilintervalle $[x_i, x_{i+1}]$ zerlegt. Die Trapezregel wird dann auf jedes Teilintervall angewendet, vgl. Abbildung 6.2.

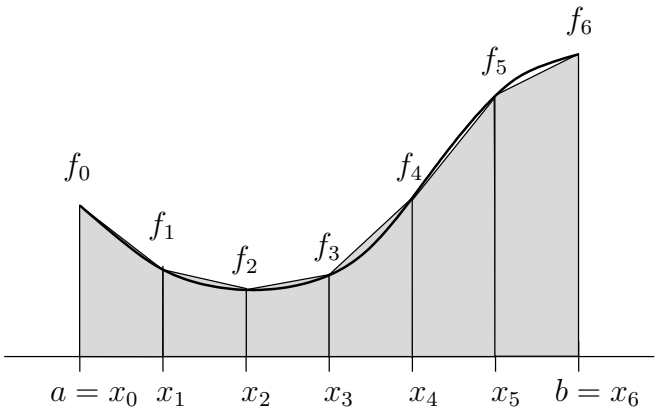


Abbildung 6.2: Zusammengesetzte Trapezregel für $n = 6$.

Dies liefert die **zusammengesetzte Trapezregel** *oder* **iterierte Trapezregel**

$$T_n[f] := \frac{h}{2}f(a) + h\sum_{i=1}^{n-1} f(x_i) + \frac{h}{2}f(b).$$

Die zusammengesetzte Trapezregel entspricht der Approximation von f durch eine stetige, stückweise lineare Funktion. Sie ist ebenfalls eine Quadraturformel der Form (6.2) mit

$$N = n,\ w_0 = w_n = \frac{h}{2},\ w_i = h,\ i = 1, \ldots, n-1,\ x_i = a + ih,\ i = 0, \ldots, n.$$

Bei zusammengesetzten Quadraturformeln spielt die Konvergenzordnung bzgl. der Schrittweite h eine große Rolle.

Definition 6.0.4 **(Konvergenzordnung zusammengesetzte Quadraturformel)**
Eine zusammengesetzte Quadraturformel $Q_n[f]$ hat die **Konvergenzordnung** p *(***ist von Ordnung** p*), falls*

$$|Q_n[f] - I[f]| = \mathcal{O}(h^p)$$

mit $h = (b-a)/n$ gilt.

6.1 Interpolatorische Quadraturformeln

Die Idee der interpolatorischen Quadraturformeln besteht darin, die Funktion f im Integranden von $I[f]$ durch ein interpolierendes Polynom mit Höchstgrad N zu ersetzen, welches f an den Stützstellen $a \leq x_0 < x1 < \ldots < x_{N-1} < x_N \leq b$ interpoliert. Für dieses interpolierende Polynom kann das Integral dann leicht ausgewertet werden. Als Spezialfall haben wir die Trapezregel bereits kennen gelernt.
Allgemeiner wird nun ein Polynom p_N höchstens N-ten Grades zur Interpolation der $N+1$ Stützpunkte $(x_i, f(x_i))$, $i = 0, \ldots, N$, verwendet. Dieses Interpolationspolynom ist nach Satz 5.1.2 eindeutig bestimmt und besitzt nach (5.4) die Darstellung

$$p_N(x) = \sum_{k=0}^{N} f(x_k) L_k(x),$$

wobei $L_k(x)$ das k-te Lagrange'sche Polynom bezeichnet.
Ersetzen von f durch p_N in $I[f]$ in (5.1) liefert die Approximation

$$I[f] = \int_a^b f(x)dx \approx \int_a^b p_N(x)dx,$$

wobei das rechts stehende Integral explizit berechnet werden kann. Es gilt

$$\int_a^b p_N(x)dx = \sum_{k=0}^{N} f(x_k) \underbrace{\int_a^b L_k(x)dx}_{=:w_k} = \sum_{k=0}^{N} w_k f(x_k)$$

mit den Koeffizienten

$$w_k = \int_a^b L_k(x)dx.$$

Insgesamt erhalten wir daraus die interpolatorischen Quadraturformeln.

Definition 6.1.1 **(interpolatorische Quadraturformeln)**
Seien $a < b$, $N \in \mathbb{N}_0$ und Stützstellen x_i, $i = 0, \ldots, N$, mit

$$a \leq x_0 < x1 < \ldots < x_{N-1} < x_N \leq b$$

gegeben. Die Quadraturformel

$$Q[f] = \sum_{k=0}^{N} w_k f(x_k), \qquad w_k = \int_a^b L_k(x)dx \tag{6.5}$$

heißt **interpolatorische Quadraturformel**.
Im Fall äquidistanter Stützstellen

$$x_i = a + ih, \quad i = 0, \ldots, N, \quad h = \frac{b-a}{N}$$

heißt eine interpolatorische Quadraturformel **geschlossenes Newton-Cotes-Verfahren**, *wobei hierbei $(a, f(a))$ und $(b, f(b))$ interpoliert werden.*
Im Fall äquidistanter Stützstellen

$$x_i = a + (i+1)h, \quad i = 0, \ldots, N, \quad h = \frac{b-a}{N+2}$$

heißt eine interpolatorische Quadraturformel **offenes Newton-Cotes-Verfahren**, *wobei hierbei $(a, f(a))$ und $(b, f(b))$ nicht interpoliert werden.*

Gemäß Definition 6.1.1 ist die Trapezregel also ein geschlossenes Newton-Cotes-Verfahren. Ein Beispiel für ein offenes Newton-Cotes/-Verfahren ist die folgende Mittelpunktsregel.

Beispiel 6.1.2 **(Mittelpunktsregel)**
Sei $[a, b]$, $a < b$, gegeben. Wähle $N = 0$ und $x_0 = (a + b)/2$ (Mittelpunkt des Intervalls). Das interpolierende Polynom 0-ten Grades zum Stützwert $(x_0, f(x_0))$ ist gegeben durch $p_0(x) = f(x_0)$. Damit wird das Integral $I[f]$ in (6.1) approximiert durch die sogenannte **Mittelpunktsregel**

$$I[f] \approx M_0[f] = \int_a^b p_0(x)dx = (b-a) f\left(\frac{a+b}{2}\right),$$

vgl. Abbildung 6.3.

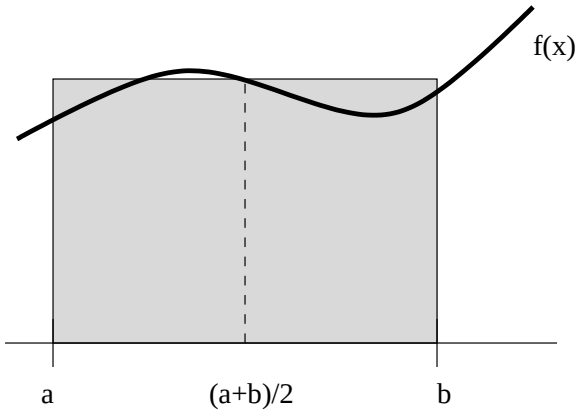


Abbildung 6.3: Mittelpunkstregel

Dies ist ein offenes Newton-Cotes-Verfahren der Form (6.2) mit

$$N = 0,\ w_0 = b - a,\ x_0 = (a+b)/2.$$

Die daraus resultierende zusammengesetze Quadraturformel lautet

$$M_n[f] = \sum_{i=0}^{n-1} (x_{i+1} - x_i) f\left(\frac{x_i + x_{i+1}}{2}\right) = h \sum_{i=0}^{n-1} f\left(\frac{x_i + x_{i+1}}{2}\right)$$

mit $h = (b-a)/n$, vgl. Abbildung 6.4.

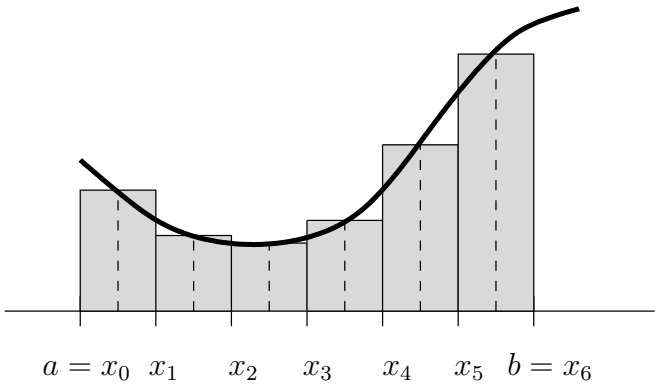


Abbildung 6.4: Iterierte Mittelpunktsregel für $n = 6$.

Wir interessieren uns nun für den Approximationsfehler interpolatorischer Quadraturformeln, wofür wir auf Satz 5.1.11 zurück greifen.

Satz 6.1.3

Seien $f : [a, b] \to \mathbb{R}$ $(N+1)$-mal stetig differenzierbar und $a \leq x_0 < x_1 < \ldots < x_N \leq b$ gegeben. Dann gilt

$$|I[f] - Q[f]| \leq \frac{1}{(N+1)!} \max_{x \in [a,b]} \left| f^{(N+1)}(x) \right| \cdot \int_a^b \prod_{i=0}^{N} |x - x_i| dx$$

für die interpolatorische Quadraturformel $Q[f]$ in (6.5).

Insbesondere besitzt die interpolatorische Quadraturformel $Q[f]$ (mindestens) den Exaktheitsgrad N.

Beweis: Es gilt

$$|I[f] - Q[f]| = |\int_a^b f(x) - p_N(x) dx| \leq \int_a^b |f(x) - p_N(x)| dx.$$

Die Fehlerdarstellung des interpolierendes Polynoms p_N aus Satz 5.1.11 liefert

$$|f(x) - p_N(x)| \leq \frac{1}{(N+1)!} \max_{x \in [a,b]} |f^{(N+1)}(x)| \cdot \prod_{i=0}^{N} |x - x_i|$$

für alle $x \in [a, b]$. Integration liefert die erste Behauptung.

Da $f^{(N+1)} \equiv 0$ für alle Polynome mit Höchstgrad N gilt, folgt aus der Fehlerdarstellung sofort $|I[f] - Q[f]| = 0$, so dass Polynome bis zum Höchstgrad N exakt integriert werden. □

Der maximale Exaktheitsgrad von $Q[f]$ kann größer als N ausfallen. So besitzt die Simpsonregel (geschlossenes Newton-Cotes-Verfahren mit $N = 2$) sogar den Exaktheitsgrad $N + 1 = 3$.

Tabelle 6.1 fasst einige gebräuchliche geschlossene Newton-Cotes-Verfahren und deren Exaktheitsgrad zusammen. Hierbei gilt $x_i = a + ih$, $i = 0, \ldots, N$, $h = (b - a)/N$.

Tabelle 6.1: Einige gebräuchliche geschlossene Newton-Cotes-Verfahren

N	Gewichte	Name	Fehler	E.-Grad
1	$w_0 = w_1 = h/2$	Trapezregel	$-\frac{h^3}{12} f^{(2)}(\xi)$	1
2	$w_0 = w_2 = h/3$, $w_1 = 4h/3$	Simpsonregel	$-\frac{h^5}{90} f^{(4)}(\xi)$	3
3	$w_0 = w_3 = 3h/8$, $w_1 = w_2 = 9h/8$	3/8-Regel (pulcherrima)	$-\frac{3h^5}{80} f^{(4)}(\xi)$	3
4	$w_0 = w_4 = 28h/90$, $w_1 = w_3 = 128h/90$, $w_2 = 48h/90$	Milneregel	$-\frac{8h^7}{945} f^{(6)}(\xi)$	5

Tabelle 6.2 fasst einige gebräuchliche offene Newton-Cotes-Verfahren und deren Exaktheitsgrad zusammen. Hierbei gilt $x_i = a + (i+1)h$, $i = 0, \dots, N$, $h = (b-a)/(N+2)$.

Tabelle 6.2: Einige gebräuchliche offene Newton-Cotes-Verfahren

N	Gewichte	Name	Fehler	E.-Grad
0	$w_0 = 2h$	Mittelpunktsregel	$\frac{h^3}{3} f^{(2)}(\xi)$	1
1	$w_0 = w_1 = 3h/2$,		$\frac{3h^3}{4} f^{(2)}(\xi)$	1
2	$w_0 = w_2 = 8h/3$, $w_1 = -4h/3$		$\frac{28h^5}{90} f^{(4)}(\xi)$	3
3	$w_0 = w_3 = 55h/24$, $w_1 = w_2 = 5h/24$,		$\frac{95h^5}{144} f^{(4)}(\xi)$	3

Für jede Regel gilt jeweils $\xi \in [a,b]$ mit eigenem ξ.

6.1.1 Zusammengesetzte Trapezregel und Euler-MacLaurin'sche Summenformel

Für jede interpolatorische Quadraturformel lässt sich eine zugehörige zusammengesetzte interpolatorische Quadraturformel entwickeln. Auf Grund der besonderen Wichtigkeit der zusammengesetzten Trapezregel für die Romberg-Extrapolation, die im folgenden Abschnitt besprochen wird, betrachten wir hier stellvertretend für andere zusammengesetzte Quadraturformeln die zusammengesetzte Trapezregel

$$T_n[f] := \frac{h}{2} f(a) + h \sum_{i=1}^{n-1} f(x_i) + \frac{h}{2} f(b) \tag{6.6}$$

mit $x_i = a + ih$, $i = 0, \ldots, n$, $h = (b-a)/n$, $n \in \mathbb{N}$, zur Approximation von $I[f]$. Insbesondere interessieren wir uns für die Konvergenzordnung und die Darstellung des Approximationsfehlers.
Für diese zusammengesetzte Trapezregel kann man mithilfe der sogenannten Euler-MacLaurin'schen Summenformel die folgende Fehlerdarstellung beweisen.

Satz 6.1.4 (asymptotische Fehlerdarstellung der zusammengesetzten Trapezregel)
Sei $f : [a,b] \to \mathbb{R}$ $(2m+2)$-mal stetig differenzierbar. Dann gilt für die zusammengesetzte Trapezregel $T_n[f]$ die **asymptotische Fehlerdarstellung**

$$T_n[f] - I[f] = \gamma_1 h^2 + \gamma_2 h^4 + \ldots + \gamma_m h^{2m} + r(h) h^{2m+2}$$

mit Konstanten γ_i, $i = 1, \ldots, m$, und beschränktem Restgliedkoeffizienten $|r(h)| \leq K$.

Beispiel 6.1.5
Betrachte

$$f(x) = \tan(x), \quad I[f] = \int_0^1 f(x)dx = [-\ln(\cos(x))]_0^1 = -\ln(\cos(1)).$$

Wir wenden die iterierte Trapezregel an und schätzen numerisch die Ordnung der Approximation. Hierzu wird angenommen, dass $|T_n[f] - I[f]| \approx Ch^p$ mit $h = 1/n$ und einer Konstanten C gilt. Dann gilt $|T_{2n}[f] - I[f]| \approx C(h/2)^p$ und somit

$$\frac{|T_n[f] - I[f]|}{|T_{2n}[f] - I[f]|} = \frac{Ch^p}{C(h/2)^p} = 2^p$$

und daraus folgt

$$p = \ln\left(\frac{|T_n[f] - I[f]|}{|T_{2n}[f] - I[f]|}\right) / \ln(2).$$

Ergebnisse der iterierten Trapezregel:

N	*Wert*	*Fehler*	*rel. Fehler*	*Ordnung*
4	0.6279861835	0.0123597131	0.02007664338	
8	0.6187665645	0.0031400941	0.00510064828	1.97
16	0.6164148747	0.0007884043	0.00128065367	1.99

Die numerischen Werte deuten auf die Ordnung 2 *hin.*

Das Besondere an der Fehlerdarstellung der zusammengesetzten Trapezregel ist die Tatsache, dass ausschließlich gerade Potenzen von h auftreten. Dieser Umstand wird von sogenannten Extrapolationsverfahren ausgenutzt.

6.2 Romberg-Verfahren und Extrapolation

Die Idee des Romberg-Verfahrens besteht darin, für verschiedene Schrittweiten h_i Näherungen für $I[f]$ mit der zusammengesetzten Trapezregel zu berechnen und diese Näherungen geeignet zu kombinieren, um bessere Näherungen zu erhalten. Konkret ergibt sich folgender

Algorithmus 6.2.1 (Romberg-Verfahren)

(0) Wähle eine Folge von Zahlen $n_i \in \mathbb{N}$, $i = 0, 1, \dots, m$ *mit*

$$n_0 < n_1 < n_2 < \dots < n_m$$

und berechne die zugehörigen Schrittweiten $h_i = (b-a)/n_i$, $i = 0, 1, 2, \dots, m$.

(1) Berechne für $i = 0, 1, 2, \dots, m$ *die zusammengesetzten Trapezsummen*

$$T_{i,0} := T_{n_i}[f].$$

(2) Berechne für $k = 1, \dots, i$, $k \le m$ *rekursiv die Werte*

$$T_{i,k} := T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left(\frac{h_{i-k}}{h_i}\right)^2 - 1}.$$

Die Berechnungsvorschrift kann in Tableauform („Neville-Schema“) zusammengefasst werden

$$\begin{array}{lcccccc} T_{n_0}[f] & = & T_{0,0} & & & & \\ T_{n_1}[f] & = & T_{1,0} & T_{1,1} & & & \\ T_{n_2}[f] & = & T_{2,0} & T_{2,1} & T_{2,2} & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\ T_{n_m}[f] & = & T_{m,0} & T_{m,1} & T_{m,2} & \cdots & T_{m,m} \end{array}$$

In der Praxis werden die **Romberg-Folge** (Schrittweitenhalbierung)

$$\{n_0, n_1, n_2, n_3, \ldots, n_i, \ldots\} = \{1, 2, 4, 8, \ldots, 2n_{i-1}, \ldots\}$$

oder die **Bulirsch-Folge**

$$\{n_0, n_1, n_2, n_3, n_4, n_5, \ldots, n_i, \ldots\} = \{1, 2, 3, 4, 6, 8, \ldots, 2n_{i-2}, \ldots\} \quad (i >= 3)$$

verwendet.

Beispiel 6.2.2

Betrachte

$$f(x) = \tan(x), \quad \int_0^1 f(x)dx = [-\ln(\cos(x))]_0^1 = -\ln(\cos(1)).$$

Das Romberg-Verfahren liefert folgende Ausgabe:

```
EXAKTER WERT:   0.6156264704E+00
ERGEBNIS DER ROMBERGEXTRAPOLATION:
 0.7787038623E+00
 0.6625031761E+00  0.6237696140E+00
 0.6279861833E+00  0.6164805191E+00  0.6159945794E+00
 0.6187665645E+00  0.6156933582E+00  0.6156408808E+00  0.6156352666E+00
 0.6164148747E+00  0.6156309781E+00  0.6156268194E+00  0.6156265962E+00  0.6156265622E+00
FEHLER DER ROMBERGEXTRAPOLATION:
 0.1630773919E+00
 0.4687670570E-01  0.8143143619E-02
 0.1235971295E-01  0.8540486976E-03  0.3681090361E-03
 0.3140094122E-02  0.6688784635E-04  0.1441045627E-04  0.8796193098E-05
 0.7884043243E-03  0.4507725118E-05  0.3490503687E-06  0.1258534497E-06  0.9185211769E-07
```

Eine genauere Betrachtung der Fehler in Spalte 1 zeigt, dass sich der Fehler bei jeder Schritthalbierung viertelt, was auf die Approximationsordnung 2 *der iterierten Trapezregel schließen lässt. In der zweiten Spalte erkennt man (mit etwas gutem Willen), dass sich der Fehler bei Schrittweitenhalbierung etwa um den Faktor* 1/16 *verringert, was auf die Ordnung* 4 *hindeutet.*

Es stellt sich die Frage, was es mit den in Schritt (2) des Algorithmus berechneten Werten $T_{i,k}$ auf sich hat. Für eine $(2m+2)$-mal stetig differenzierbare Funktion f gilt nach Satz 6.1.4 die asymptotische Fehlerdarstellung

$$T_n[f] - I[f] = \gamma_1 h^2 + \gamma_2 h^4 + \ldots + \gamma_m h^{2m} + r(h)h^{2m+2}$$

für die zusammengesetzte Trapezregel, wobei die γ_i's Konstanten und $|r(h)| \leq K$ sind.

Zur Vereinfachung betrachten wir die Romberg-Folge $h_i = h_{i-1}/2$. Für zwei konkrete Schrittweiten $h_i = h_{i-1}/2$ gelten die Fehlerdarstellungen

$$\begin{aligned} T_{i-1,0} - I[f] &= \sum_{j=1}^{m} \gamma_j h_{i-1}^{2j} + r(h_{i-1}) h_{i-1}^{2m+2}, && (6.7)\\ T_{i,0} - I[f] &= \sum_{j=1}^{m} \gamma_j h_i^{2j} + r(h_i) h_i^{2m+2} && (6.8)\\ &= \sum_{j=1}^{m} \gamma_j \left(\frac{h_{i-1}}{2}\right)^{2j} + r\left(\frac{h_{i-1}}{2}\right)\left(\frac{h_{i-1}}{2}\right)^{2m+2}. && (6.9) \end{aligned}$$

Betrachte das Interpolationspolynom p der Form

$$p(h) = \alpha_0 + \alpha_1 h^2$$

in h, welches die Werte $(h_{i-1}, T_{i-1,0})$ und $(h_i, T_{i,0})$ interpoliert. Es folgt

$$\alpha_0 = \frac{{h_i}^2 T_{i-1,0} - {h_{i-1}}^2 T_{i,0}}{{h_i}^2 - {h_{i-1}}^2}, \quad \alpha_1 = \frac{T_{i,0} - T_{i-1,0}}{{h_i}^2 - {h_{i-1}}^2}.$$

Auswertung des Polynoms an der Stelle $h = 0$ (**Extrapolation**) liefert mit einigen Umformungen

$$p(0) = \frac{{h_i}^2 T_{i-1,0} - {h_{i-1}}^2 T_{i,0}}{{h_i}^2 - {h_{i-1}}^2} = T_{i,0} + \frac{T_{i,0} - T_{i-1,0}}{\left(\frac{h_{i-1}}{h_i}\right)^2 - 1} = T_{i,1}. \tag{6.10}$$

Für $T_{i,1}$ folgt durch Einsetzen von (6.7) und (6.8) die Fehlerdarstellung

$$\begin{aligned} T_{i,1} - I[f] &= \sum_{j=2}^{m} \tilde{\gamma}_j h_{i-1}^{2j} + \tilde{r}(h_{i-1}) h_{i-1}^{2m+2}\\ &= \tilde{\gamma}_2 h_{i-1}^4 + \tilde{\gamma}_3 h_{i-1}^6 + \ldots + \tilde{\gamma}_m h_{i-1}^{2m} + \tilde{r}(h_{i-1}) h_{i-1}^{2m+2} \end{aligned}$$

mit

$$\tilde{\gamma}_j = \frac{\gamma_j}{3}\left(\frac{1}{2^{2j-2}} - 1\right), \quad \tilde{r}(h_{i-1}) = \frac{4}{3}\left(\frac{r(h_{i-1}/2)}{2^{2m+2}} - 4r(h_{i-1})\right).$$

Also hat $T_{i,1}$ genau wie $T_{i-1,0}$ und $T_{i,0}$ eine asymptotische Darstellung mit beschränktem Restglied $\tilde{r}$. Während der führende Term in den Fehlerdarstellungen (6.7) und (6.8) noch h_{i-1}^2 enthielt, lautet dieser für $T_{i,1}$ aber $\tilde{\gamma}_2 h_{i-1}^4$. Wir haben also durch eine geeignete Linearkombination von Approximationen $T_{i-1,0}$ und $T_{i,0}$ der Ordnung 2 gemäß (6.10) eine bessere Approximation $T_{i,1}$ der Ordnung 4 berechnet, indem wir ein geeignetes interpolierendes Polynom an der Stelle $h = 0$ ausgewertet haben (**Extrapolationsvorgang**). Betrachte das Neville-Schema. Die Werte $T_{i-1,0}$ und $T_{i,0}$ stehen in der ersten Spalte, die Werte $T_{i,1}$ stehen in der zweiten Spalte. Beim Übergang von der ersten Spalte zur zweiten Spalte gewinnen wir also zwei Potenzen in h an Genauigkeit. Die Vermutung liegt

nahe, dass beim Übergang von der zweiten zur dritten Spalte wieder zwei Potenzen in h gewonnen werden. Allgemein sei

$$p_{ik}(h) = \alpha_0 + \alpha_1 h^2 + \ldots + \alpha_k h^{2k}$$

dasjenige Interpolationspolynom, welches die Werte $(h_{i-k}, T_{i-k,0})$, $(h_{i-k+1}, T_{i-k+1,0}), \ldots, (h_i, T_{i,0})$ interpoliert, vgl. Abbildung 6.5, d.h.

$$p_{ik}(h_j) = T_{j,0} = T_{n_j}[f], \quad j = i-k, i-k+1, \ldots, i.$$

Dann gilt

$$T_{i,k} = p_{ik}(0).$$

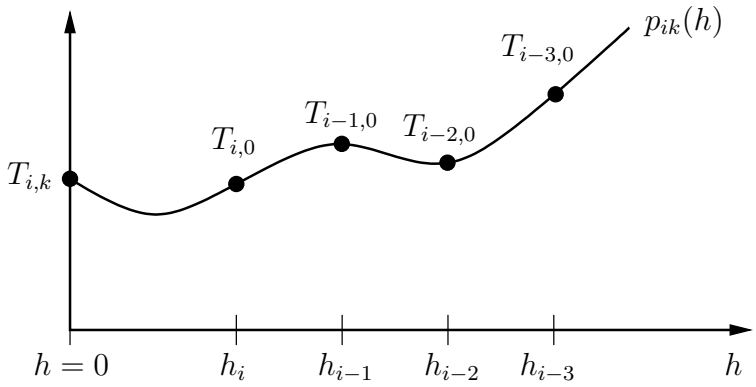


Abbildung 6.5: Prinzip der Extrapolation für $k = 3$.

Für die Romberg-Folge ist die Fehlerordnung

$$T_{i,k} - I[f] = \mathcal{O}(h_{i-k}^{2k+2}).$$

Mit Hilfe des Extrapolationsverfahrens können sehr schnell extrem genaue Approximationen berechnet werden. Der Zusammenhang des Tableaus mit dem Neville-Schema zur Berechnung des Wertes eines Interpolationspolynoms wird deutlich, wenn p_{ik} als Polynom in h^2 aufgefasst wird:

$$p_{ik}(h) = \alpha_0 + \alpha_1 (h^2)^1 + \alpha_2 (h^2)^2 + \ldots + \alpha_k (h^2)^k.$$

Grundvoraussetzung zur Durchführbarkeit der Extrapolation ist die Existenz einer asymptotischen Entwicklung. Diese muss nicht notwendig nur gerade Potenzen von h enthalten, sondern kann für eine Funktion T, die das exakte Ergebnis γ_0 approximiert, allgemein wie folgt aussehen:

$$T(h) = \gamma_0 + \gamma_1 h^{\tau_1} + \ldots + \gamma_m h^{\tau_m} + r(h) h^{\tau_{m+1}}$$

mit Konstanten $\gamma_0, \dots, \gamma_m$, $0 < \tau_1 < \tau_2 < \dots < \tau_{m+1}$ und beschränkter Funktion r. Die Beschränktheit von r ist wichtig, da ansonsten nicht gewährleistet ist, dass der Fehler für $h \to 0$ beschränkt bleibt. Der Nachweis der Existenz einer asymptotischen Entwicklung für ein konkretes Approximationsverfahren T ist keinesfalls trivial. Desweiteren besitzt nicht jedes Approximationsverfahren eine asymptotische Entwicklung.
Die Existenz einer asymptotischen Entwicklung ermöglicht auch eine Schrittweitensteuerung. Die Idee ist, dass aus zwei gegebenen Näherungen eine neue Schrittweite berechnet wird, so dass der Fehler der zugehörigen Approximation eine gegebene Toleranz nicht übersteigt.

6.3 Gauß'sche Quadraturformeln

Ziel ist es nun, Quadraturformeln der Form (6.2) mit **maximalem Exaktheitsgrad** herzuleiten, wobei neben den Gewichten w_i auch die Stützstellen x_i **frei wählbar** sind. Wir betrachten hier etwas allgemeinere Integrale der Form

$$I_\alpha[f] := \int_a^b \alpha(x) f(x) dx$$

mit einer gegebenen stetigen und positiven Gewichtsfunktion $\alpha : [a,b] \to \mathbb{R}_+$, die durch die Quadraturformel

$$Q_N[f] = \sum_{k=0}^{N} w_k f(x_k)$$

approximiert werden sollen. Als Spezialfall kann (und wird) stets $\alpha \equiv 1$ betrachtet werden. Wir haben im Abschnitt über interpolatorische Quadraturformeln gesehen, dass bei fest gewählten Knoten x_k, $k = 0, \dots, N$, Exaktheitsgrad N erreicht werden kann. Es ist daher zu vermuten, dass durch optimale Wahl der $N+1$ Knoten, der Exaktheitsgrad $2N+1$ erreicht werden kann. Ein noch höherer Exaktheitsgrad kann nicht erreicht werden, wie das Gegenbeispiel

$$p(x) = \prod_{k=0}^{N} (x - x_k)^2$$

vom Grad $2N+2$ zeigt. Es ist nichtnegativ und verschwindet nicht identisch, daher gilt wegen $\alpha(x) > 0$ für alle $x \in [a,b]$ auch $I_\alpha[p] > 0$. Andererseits gilt aber $Q_N[p] = 0$ für beliebige Wahl der Stützstellen x_k, da diese gerade die Nullstellen von p sind. Also beträgt der maximale Exaktheitsgrad von $Q_N[f]$ höchstens $2N+1$.
Der folgende Satz liefert ein hinreichendes Kriterium für den Exaktheitsgrad $2N+1$.

Satz 6.3.1 (Gauß-Quadratur)
Seien α eine stetige nichtnegative Gewichtungsfunktion und q ein nichttriviales Polynom

vom Grad $N+1$ mit

$$\int_a^b x^k \alpha(x) q(x) dx = 0 \qquad (0 \le k \le N), \tag{6.11}$$

d.h. q ist α-orthogonal zu allen Polynomen vom Höchstgrad N. Seien $x_0, x_1, \ldots, x_N$ die Nullstellen von q. Dann ist die Quadraturformel

$$Q_N[f] = \sum_{k=0}^{N} w_k f(x_k) \qquad \textit{mit} \qquad w_k = \int_a^b \alpha(x) L_k(x) dx \tag{6.12}$$

exakt für alle Polynome vom Höchstgrad $2N+1$.

Beweis: Sei f ein Polynom vom Grad $2N+1$. Division mit Rest von f durch q liefert $f = qp + r$ mit Polynomen p und r vom Höchstgrad N. Nach Definition von q gilt $f(x_k) = r(x_k)$, $k = 0, \ldots, N$. Weiter gilt wegen (6.11)

$$I_\alpha[f] = \int_a^b \alpha(x) q(x) p(x) + \alpha(x) r(x) dx = \int_a^b \alpha(x) r(x) dx.$$

Analog zu Satz 6.1.3 kann gezeigt werden, dass die Quadraturformel (6.12) exakt ist für alle Polynome bis zum Höchstgrad N, so dass

$$\int_a^b \alpha(x) r(x) dx = \sum_{k=0}^{N} w_k r(x_k) = \sum_{k=0}^{N} w_k f(x_k)$$

gilt, woraus die Behauptung folgt. □

Gelingt es, α-orthogonale Polynome zu konstruieren, so besitzen die Nullstellen dieser Polynome und die Gewichte weitere schöne Eigenschaften.

Satz 6.3.2

Sei α eine positive und stetige Gewichtsfunktion und q ein Polynom $N+1$-ten Grades mit (6.11). Für die Nullstellen $x_0, \ldots, x_N$ von q und die Gewichte $w_0, \ldots, w_N$ gemäß (6.12) gelten folgende Aussagen:

(i) $x_k \in (a,b)$ für alle $k = 0, \ldots, N$ und die Nullstellen sind reell und paarweise verschieden.

(ii) $w_k > 0$ für alle $k = 0, \ldots, N$ und $\sum_{k=0}^{N} w_k = \int_a^b \alpha(x) dx$.

Nach Satz 6.3.1 gilt es, α-orthogonale Polynome q mit (6.11) zu konstruieren. Dies kann wie im Abschnitt über das konjugierte Gradientenverfahren mit Hilfe des **Gram-Schmidtschen Orthogonalisierungsverfahrens** bzgl. des Skalarprodukts

$$\langle f,g\rangle_\alpha := \int_a^b \alpha(x)f(x)g(x)dx$$

geschehen. Die Polynome $\{1,x,x^2,\dots,x^k,\dots\}$ sind linear unabhängig, so dass α-orthogonale Polynome durch die Vorschrift

$$q_k = x^k - \sum_{j=0}^{k-1} \frac{\langle q_j, x^k\rangle_\alpha}{\langle q_j,q_j\rangle_\alpha} q_j, \qquad k=1,2,\dots,$$

berechnet werden können.

Wir beschränken uns zur Illustration auf das Intervall $[a,b]=[-1,1]$. Dies ist keine wirkliche Einschränkung, da das Integral (6.1) durch die lineare Transformation $x(t) = (a(1-t)+b(1+t))/2$ auf das Intervall $[-1,1]$ transformiert werden kann:

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^1 f\left((a(1-t)+b(1+t))/2\right)dt$$

Ausserdem betrachten wir nur den Fall $\alpha\equiv 1$. Für andere Gewichtsfunktionen ergeben sich andere α-orthogonale Polynome und damit auch andere Quadraturformeln.

Beispiel 6.3.3

Startend mit $q_0(x)\equiv 1$ liefert das Gram-Schmidtsche Orthogonalisierungsverfahren:

$$\begin{aligned} q_1(x) &= x - \frac{\int_{-1}^1 1\cdot x dx}{\int_{-1}^1 1^2 dx}\cdot 1 = x, \\ q_2(x) &= x^2 - \frac{\int_{-1}^1 1\cdot x^2 dx}{\int_{-1}^1 1^2 dx}\cdot 1 - \frac{\int_{-1}^1 x\cdot x^2 dx}{\int_{-1}^1 x^2 dx}\cdot x = x^2 - \frac{1}{3}. \end{aligned}$$

Jedes Vielfache ungleich Null dieser Polynome ist ebenfalls zulässig, so dass noch eine Normierung vorgenommen werden könnte.

Wir konstruieren die dazu passende Gauß'sche Quadraturformel für $N=1$. Dazu benötigen wir die Nullstellen des Polynoms $q_{N+1}=q_2$, welche durch

$$x_0 = -\sqrt{\frac{1}{3}}, \quad x_1 = \sqrt{\frac{1}{3}}$$

gegeben sind. Daraus ergeben sich folgende Gewichte:

$$\begin{aligned} w_0 &= \int_{-1}^{1} L_0(x)dx \\ &= \int_{-1}^{1} \frac{x-x_1}{x_0-x_1}dx \\ &= 1, \\ w_1 &= \int_{-1}^{1} L_1(x)dx \\ &= \int_{-1}^{1} \frac{x-x_0}{x_1-x_0}dx \\ &= 1. \end{aligned}$$

Die Gauß'sche Quadraturformel für $N = 1$ und $[a, b] = [-1, 1]$ lautet also

$$\int_{-1}^{1} f(x)dx \approx w_0 f(x_0) + w_1 f(x_1) = f(-\sqrt{1/3}) + f(\sqrt{1/3}).$$

Allgemein gilt folgender Satz zur Konstruktion α-orthogonaler Polynome.

Satz 6.3.4 **(α-orthogonale Polynome)**
Die durch die Rekursion

$$\begin{aligned} q_0(x) &= 1, \\ q_1(x) &= x - a_1, \\ q_k(x) &= (x-a_k)q_{k-1}(x) - b_k q_{k-2}(x), \quad k = 2, 3, \ldots, \end{aligned}$$

mit

$$\begin{aligned} a_k &= \frac{\langle xq_{k-1}, q_{k-1}\rangle_\alpha}{\langle q_{k-1}, q_{k-1}\rangle_\alpha}, \\ b_k &= \frac{\langle xq_{k-1}, q_{k-2}\rangle_\alpha}{\langle q_{k-2}, q_{k-2}\rangle_\alpha} \end{aligned}$$

definierten Polynome sind α-orthogonal.

Beweis: Der Beweis kann per Induktion geführt werden.

Für $[a, b] = [-1, 1]$ und $\alpha \equiv 1$ ergeben sich aus der Rekursion die sogenannten **Legendre-**

Polynome, deren erste Vertreter wie folgt lauten:

$$\begin{aligned} q_0(x) &= 1, \\ q_1(x) &= x, \\ q_2(x) &= x^2 - \frac{1}{3}, \\ q_3(x) &= x^3 - \frac{3}{5}x, \\ q_4(x) &= x^4 - \frac{6}{7}x^2 + \frac{3}{35}. \end{aligned}$$

Tabelle 6.3 fasst Knoten und Gewichte von einigen Gauß-Quadraturformeln für $\alpha \equiv 1$ und $[a,b] = [-1,1]$ zusammen. Zu beachten ist, dass die angegebenen Stützstellen und Koeffizienten ausschließlich für das Intervall $[-1,1]$ und die Gewichtsfunktion $\alpha \equiv 1$ gültig sind.

Tabelle 6.3: Einige Gauß Quadraturformeln für $\alpha \equiv 1$ und $[a,b] = [-1,1]$.

N	Knoten x_i	Gewichte A_i
0	$x_0 = 0$	$w_0 = 2$
1	$x_0 = -\sqrt{1/3}$	$w_0 = 1$
	$x_1 = +\sqrt{1/3}$	$w_1 = 1$
2	$x_0 = -\sqrt{3/5}$	$w_0 = 5/9$
	$x_1 = 0$	$w_1 = 8/9$
	$x_2 = +\sqrt{3/5}$	$w_2 = 5/9$
3	$x_0 = -\sqrt{\frac{3}{7} + \frac{1}{7}\sqrt{\frac{24}{5}}}$	$w_0 = \frac{18-\sqrt{30}}{36}$
	$x_1 = -\sqrt{\frac{3}{7} - \frac{1}{7}\sqrt{\frac{24}{5}}}$	$w_1 = \frac{18+\sqrt{30}}{36}$
	$x_2 = +\sqrt{\frac{3}{7} - \frac{1}{7}\sqrt{\frac{24}{5}}}$	$w_2 = \frac{18+\sqrt{30}}{36}$
	$x_3 = +\sqrt{\frac{3}{7} + \frac{1}{7}\sqrt{\frac{24}{5}}}$	$w_3 = \frac{18-\sqrt{30}}{36}$

Wir betrachten noch ein Beispiel.

Beispiel 6.3.5

Approxiere das Integral

$$I = \int_{-2}^{2} \exp(-t^2) dt.$$

Da das Intervall auf $[a,b] := [-2,2]$ *und nicht auf* $[-1,1]$ *definiert ist, muss es zunächst*

auf $[-1,1]$ *transformiert werden. Dies wird erreicht durch die lineare Transformation*

$$t(x) = \frac{a(1-x)+b(1+x)}{2} = \frac{-2(1-x)+2(1+x)}{2} = 2x \qquad \Rightarrow \qquad dt = 2dx.$$

Transformation des Integrals:

$$\int_{-2}^{2} \exp(-t^2)dt = \int_{-1}^{1} exp(-(2x)^2)\cdot 2dx = \int_{-1}^{1} 2\exp(-4x^2)dx.$$

Anwendung der Formeln:

$$\int_{-2}^{2} \exp(-t^2)dt = \int_{-1}^{1} \underbrace{2\exp(-4x^2)}_{=f(x)}\,dx \approx \sum_{k=0}^{N} w_k f(x_k).$$

Für $N=1$ *folgt*

$$\begin{aligned}
\int_{-2}^{2} \exp(-t^2)dt &= \int_{-1}^{1} \underbrace{2\exp(-4x^2)}_{=f(x)}\,dx \\
&\approx w_0 f(x_0) + w_1 f(x_1) \\
&= 1\cdot f(-\sqrt{1/3}) + 1\cdot f(\sqrt{1/3}) \\
&= 2\exp(-4/3) + 2\exp(-4/3) \\
&= 4\exp(-4/3) \approx 1.054388553.
\end{aligned}$$

Für $N=2$ *erhalten wir den Wert* 1.979373230 *und für* $N=3$ 1.714546068. *Der exakte Wert des Integrals beträgt ungefähr* $I \approx 1.764162782$.

Bemerkung 6.3.6

Wie bereits erwähnt, ergeben sich für verschiedene Intervalle $[a,b]$ *und Gewichtsfunktionen* α *verschiedene Gauß-Quadraturformeln.*

- *Für die Gewichtsfunktion* $\alpha(x) = \frac{1}{\sqrt{1-x^2}}$ *und* $[a,b] = [-1,1]$ *ergeben sich die* α*-orthogonalen Tschebyscheff-Polynome.*
- *Für die Gewichtsfunktion* $\alpha(x) = \exp(-x)$ *und* $[a,b] = [0,\infty)$ *ergeben sich die sogenannten* **Laguerre-Polynome**, *deren erste Glieder wie folgt lauten:*

$$1, \quad 1-x, \quad 2-4x+x^2, \quad 6-18x+9x^2-x^3$$

- *Für die Gewichtsfunktion* $\alpha(x) = \exp(-x^2)$ *und* $[a,b] = (-\infty,\infty)$ *ergeben sich die sogenannten* **Hermite-Polynome**, *deren erste Glieder wie folgt lauten:*

$$1, \quad x, \quad x^2 - \frac{1}{2}, \quad x^3 - \frac{3}{2}x.$$

Abschließend zitieren wir noch ein Konvergenzresultat für die Gauß-Quadratur für den Fall $N \to \infty$.

Satz 6.3.7

Sei f stetig in $[a,b]$. Dann gilt

$$\lim_{N\to\infty} Q_N[f] = I[f]$$

für die Gauß-Quadratur.

6.4 Adaptive Verfahren am Beispiel der Simpson-Regel

In den vorangehenden Abschnitten wurden stets eine feste Unterteilung des Intervalls $[a,b]$ gewählt. In diesem Abschnitt wird diese Unterteilung nicht a priori festgelegt, sondern **adaptiv** in Abhängigkeit vom Approximationsfehler gewählt. Der Grund hierfür ist, dass in einigen Abschnitten von $[a,b]$ mitunter (z.B. wenn f dort stark variiert) eine sehr feine Unterteilung notwendig ist, um eine gute Approximation zu erreichen, während in anderen Bereichen (z.B. weil f sich dort kaum ändert) eine grobe Unterteilung ausreicht, vgl. Abbildung 6.6.

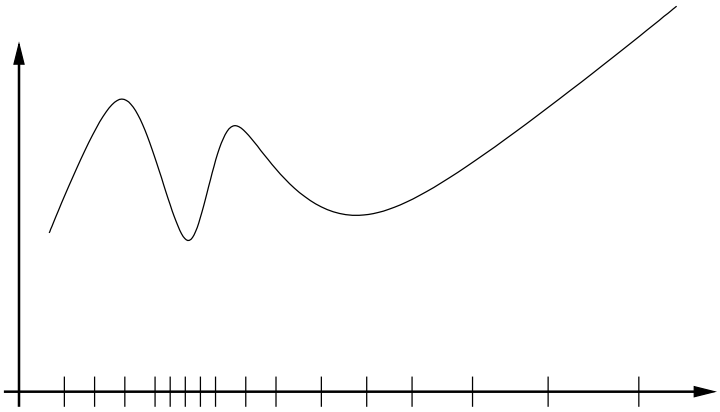


Abbildung 6.6: Idee eines adaptiven Verfahrens: Die Schrittweite wird lokal adaptiv an die Eigenschaften von f angepasst.

Zur Demonstration wählen wir die

elementare Simpson-Regel:

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

Falls f in $[a,b]$ 4-mal stetig differenzierbar ist, dann besitzt die Simpson-Regel folgende Fehlerdarstellung:

Fehler:

$$I[f] - S(a,b) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi) \qquad (\xi \in (a,b)).$$

Die adaptive Simpson-Regel basiert auf einer Methode zur numerischen Schätzung des Interpolationsfehlers:

Adaptive Simpson-Regel::

(i) Seien das Intervall $[a,b]$ und eine Toleranz $\varepsilon > 0$ gegeben.

(ii) Wende die Simpson-Regel auf $[a,b]$ an und berechne $S^{(1)} := S(a,b)$.

(iii) Wende die elementare Simpsonregel auf $[a,c]$ und $[c,b]$ mit $c = (a+b)/2$ an und berechne $S^{(2)} := S(a,c) + S(c,b)$ (**zusammengesetzte Simpson-Regel!**).

(iv) Falls die Ungleichung

$$|S^{(2)} - S^{(1)}| < 15\varepsilon \tag{6.13}$$

erfüllt ist, STOP (der Approximationsfehler ist hinreichend klein) und verwende

$$S^{(2)} + \frac{1}{15}\left(S^{(2)} - S^{(1)}\right) \tag{6.14}$$

als Approximation an $\int_a^b f(x)dx$.
Falls die Ungleichung **nicht** erfüllt ist:

- Teile $[a,b]$ in zwei Teilintervalle $[a,c]$ und $[c,b]$ auf,
- Wende diesen Algorithmus rekursiv auf $[a,c]$ **und** $[c,b]$ mit ε ersetzt durch $\varepsilon/2$ an.

Hintergrund der Fehlerschätzung (6.13):
Seien $h = b - a$ und $C := f^{(4)}(c)$[1]. Mit Hilfe der Fehlerformel der elementaren Simpson-

[1]Falls f 5-mal stetig differenzierbar ist, liefert Taylorentwicklung $f^{(4)}(\xi) = f^{(4)}(c) + \frac{1}{2}f^{(5)}(\zeta)(\xi - c) = f^{(4)}(c) + \mathcal{O}(h)$ für jedes $\xi \in (a,b)$ und $\zeta \in (c,\xi)$.

Regel folgt

$$E^{(1)} := I[f] - S^{(1)} = -\frac{1}{90}\left(\frac{h}{2}\right)^5 C + \mathcal{O}(h^6)) \tag{6.15}$$

$$\begin{aligned} E^{(2)} := I[f] - S^{(2)} &= -\frac{1}{90}\left(\frac{h/2}{2}\right)^5 C - \frac{1}{90}\left(\frac{h/2}{2}\right)^5 C + \mathcal{O}(h^6) \\ &= \frac{1}{16}\left(-\frac{1}{90}\left(\frac{h}{2}\right)^5\right) C + \mathcal{O}(h^6)). \end{aligned} \tag{6.16}$$

Ein Vergleich der rechten Seiten liefert

$$\begin{aligned} I[f] - S^{(1)} &\approx 16E^{(2)}, \\ I[f] - S^{(2)} &= E^{(2)}. \end{aligned}$$

Subtraktion der zweiten von der ersten Gleichung liefert

$$S^{(2)} - S^{(1)} \approx 15E^{(2)},$$

und daher

$$I[f] - S^{(2)} = E^{(2)} \approx \frac{1}{15}\left(S^{(2)} - S^{(1)}\right).$$

Deshalb kann der Wert

$$\frac{1}{15}\left(S^{(2)} - S^{(1)}\right) \tag{6.17}$$

als Approximation des Fehlers $I[f] - S^{(2)}$ verwendet werden. Beachte, dass $E^{(2)}$ nicht bekannt ist, aber wir können (6.17) berechnen. Gleichung (6.13) sichert (approximativ), dass $E^{(2)}$ kleiner als ε ist. Es ist wichtig zu bemerken, dass die Approximation in (6.14) den Fehler $\mathcal{O}(h^6)$ hat. Dies folgt aus den Fehlerdarstellungen (6.15) und (6.16) (mulitpliziere die erste Gleichung mit $-1/15$, die zweite mit $16/15$ und addiere beide Gleichungen).
Die Division von ε durch zwei in Schritt (iv) soll garantieren, dass der Fehler in jedem der zwei Teilintervalle kleiner als $\varepsilon/2$ verbleibt, so dass der Fehler des gesamten Intervalls kleiner als $\varepsilon/2 + \varepsilon/2 = \varepsilon$ ist.

6.5 Mehrdimensionale Integrale

Ziel dieses Abschnitts ist es, n-dimensionale Integrale der Form

$$I[f] = \int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} f(x_1, \ldots, x_n) dx_n \cdots dx_1$$

zu approximieren. Zur Vereinfachung sei im folgenden $n = 2$ gewählt, d.h.

$$I[f] = \int_a^b \int_c^d f(x, y) dx dy. \tag{6.18}$$

Man kann leicht Quadraturformeln entwickeln, indem die Quadraturformeln für eindimensionale Integrale verwendet werden. Dies soll am Beispiel der Trapezregel verdeutlicht werden. Zunächst wird das innere Integral

$$\int_c^d f(x,y)dx$$

für festes y durch die zusammengesetzte Trapezregel approximiert:

$$\int_c^d f(x,y)dx \approx h_1 \left(\frac{f(x_0)}{2} + \sum_{i=1}^{n_1-1} f(x_i,y) + \frac{f(x_{n_1})}{2} \right) =: \sum_{i=0}^{n_1} A_i f(x_i,y)$$

wobei $h_1 = (d-c)/n_1$ und $x_i = c + ih_1$, $i = 0, 1, \ldots, n_1$. Dann wird diese Approximation in das ursprüngliche Integral eingesetzt und die iterierte Trapezregel wird angewendet auf das äußere Integral mit $h_2 = (b-a)/n_2$, $y_j = a + jh_2$, $j = 0, 1, \ldots, n_2$:

$$\begin{aligned}
\int_a^b \int_c^d f(x,y)dxdy &\approx \int_a^b \sum_{i=0}^{n_1} A_i f(x_i,y)dy \\
&= \sum_{i=0}^{n_1} A_i \left(\int_a^b f(x_i,y)dy \right) \\
&\approx \sum_{i=0}^{n_1} A_i \left(\sum_{j=0}^{n_2} B_j f(x_i,y_j) \right) \\
&= \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} A_i B_j f(x_i,y_j).
\end{aligned}$$

Entsprechend kann für $n > 2$ verfahren werden. Ebenso können auch Newton-Cotes-Formeln oder zusammengesetzte Verfahren verwendet werden. Es gibt auch Verfahren, die auf der Gauß-Quadratur basieren.

Kapitel 7

Numerische Differentiation

In diesem Kapitel beschäftigen wir uns mit der Approximation von Ableitungen einer hinreichend glatten Funktion f an einem Punkt x.

7.1 Approximation der ersten Ableitung mittels Taylorentwicklung

Ziel ist es, die erste Ableitung $f'(x)$ zu approximieren.

7.1.1 Finite Differenzen Verfahren erster Ordnung

Sei f zweimal stetig differenzierbar in x. Der Satz von Taylor liefert dann

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi).$$

Umsortierung führt auf

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(\xi).$$

Vernachlässigung des Fehlerterms $-\frac{1}{2}hf''(\xi)$ liefert

Finite Differenzen Approximation erster Ordnung an $f'(x)$:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{7.1}$$

Fehlerabschätzung:

$$\left| f'(x) - \frac{f(x+h) - f(x)}{h} \right| \leq C \cdot h = \mathcal{O}(h), \qquad C := \frac{1}{2} \max_{\zeta \in [x, x+h]} |f''(\zeta)|.$$

Bemerkung 7.1.1

In der Praxis kann h auf Grund von Rundungsfehlereinfluss nicht beliebig klein gewählt werden. Andererseits sollte h auch nicht zu groß gewählt werden, da dann der Approximationsfehler dominiert. In der Praxis hat sich $h = \sqrt{eps}\max\{1, |x|\}$ bewährt, wobei eps die Maschinengenauigkeit bezeichnet.

7.1.2 Finite Differenzen zweiter Ordnung

Sei f dreimal stetig differenzierbar in x. Taylorentwicklung liefert

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi_1), \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(\xi_2). \end{aligned}$$

Subtraktion liefert

$$f(x+h) - f(x-h) = 2hf'(x) + \underbrace{\frac{1}{6}h^3(f'''(\xi_1) + f'''(\xi_2))}_{\text{error term}}.$$

Umsortieren und Vernachlässigen des Fehlerterms führt auf

Finite Differenzen Approximation 2. Ordnung an $f'(x)$:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{7.2}$$

Fehlerabschätzung:

$$\left| f'(x) - \frac{f(x+h) - f(x-h)}{2h} \right| \leq C \cdot h^2 = \mathcal{O}(h^2), \qquad C := \frac{1}{6} \max_{\zeta \in [x-h, x+h]} |f'''(\zeta)|.$$

7.2 Approximation zweiter Ableitungen mittels Taylorentwicklung

Wir möchten $f''(x)$ approximieren.
Sei f viermal stetig differenzierbar in x. Dann gilt

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \frac{1}{24}h^4 f^{(4)}(\xi_1), \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + \frac{1}{24}h^4 f^{(4)}(\xi_2). \end{aligned}$$

Addition liefert

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + \underbrace{\frac{1}{24}h^4(f^{(4)}(\xi_1) + f^{(4)}(\xi_2))}_{\text{Fehler}}.$$

Umsortierung und Vernachlässigung des Fehlers führt auf

Finite Differenzen Approximation zweiter Ordnung an $f''(x)$:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{7.3}$$

Fehlerabschätzung:

$$\left| f''(x) - \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \right| \le C \cdot h^2 = \mathcal{O}(h^2), \qquad C := \frac{1}{12} \max_{\zeta \in [x-h, x+h]} |f^{(4)}(\zeta)|.$$

7.3 Richardson Extrapolation

Für festes f und x definiere das Verfahren $\varphi(h)$ durch

$$\varphi(h) := \frac{f(x+h) - f(x-h)}{2h}.$$

Unter der Annahme, dass f beliebig hohe Ableitungen besitzt, liefert Taylorentwicklung

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2!}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \frac{1}{4!}h^4 f^{(4)}(x) + \dots, \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2!}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + \frac{1}{4!}h^4 f^{(4)}(x) - \dots. \end{aligned}$$

Subtraktion und Umsortierung liefert

$$\begin{aligned} \varphi(h) - f'(x) &= \frac{1}{3!}h^2 f'''(x) + \frac{1}{5!}h^4 f^{(5)}(x) + \dots \\ &=: -a_2 h^2 - a_4 h^4 - a_6 h^6 - \dots \\ &= -\sum_{i=1}^{\infty} a_{2i} h^{2i}. \end{aligned} \tag{7.4}$$

Die Koeffizienten a_{2i} hängen von x aber nicht von h ab!

Bemerkung 7.3.1 (Asymptotische Fehlerdarstellung)
Gleichung 7.4 ist eine **asymptotische Fehlerdarstellung**. *Die Existenz dieser Darstellung ist die Basis für die folgende Richardson Extrapolation (vgl. auch die zusammengesetzte Trapezregel zur Integration von Funktionen).*

Betrachte

$$\begin{aligned} \varphi(h) - f'(x) &= -a_2 h^2 - a_4 h^4 - a_6 h^6 - \dots, \\ \varphi\left(\frac{h}{2}\right) - f'(x) &= -a_2 \left(\frac{h}{2}\right)^2 - a_4 \left(\frac{h}{2}\right)^4 - a_6 \left(\frac{h}{2}\right)^6 - \dots. \end{aligned}$$

Beachte, dass

$$\varphi\left(\frac{h}{2}\right) + \frac{1}{3}\left[\varphi\left(\frac{h}{2}\right) - \varphi(h)\right] - f'(x) = \frac{1}{4}a_4 h^4 + \frac{5}{16}a_6 h^6 + \dots.$$

Durch eine geeignete Kombination der beiden Approximationen $\varphi(h)$ und $\varphi(h/2)$, wobei jede den Fehler $\mathcal{O}(h^2)$ besitzt, erhalten wir eine Approximation der Ordnung $\mathcal{O}(h^4)$. Wir haben also zwei Ordnungen gewonnen.

Dieses Verfahren heißt **Richardson Extrapolation**.

Algorithmus 7.3.2 (Richardson Extrapolation)

(i) Wähle N und h.

(ii) Für $i = 0, 1, \ldots, N$, berechne $D(i,0) := \varphi\left(\frac{h}{2^i}\right)$.

(iii) Für $j = 1, \ldots, i$, $i = 1, \ldots, N$, berechne

$$D(i,j) = D(i,j-1) + \frac{D(i,j-1) - D(i-1,j-1)}{4^j - 1}.$$

Der Algorithmus führt auf das folgende Schema („Neville Schema“):

		$\mathcal{O}(h^2)$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^6)$	$\cdots$	$\mathcal{O}(h^{2(N+1)})$
$\varphi(h)$	$=$	$D(0,0)$				
$\varphi(h/2)$	$=$	$D(1,0)$	$D(1,1)$			
$\varphi(h/4)$	$=$	$D(2,0)$	$D(2,1)$	$D(2,2)$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
$\varphi(h/2^N)$	$=$	$D(N,0)$	$D(N,1)$	$D(N,2)$	$\cdots$	$D(N,N)$

Es gilt

Satz 7.3.3 (Richardson Extrapolation)

Sei f beliebig oft differenzierbar. Für $D(n,m)$ gilt

$$D(n,m) - f'(x) = \sum_{k=m+1}^{\infty} A(k,m) \left(\frac{h}{2^n}\right)^{2k} \qquad (0 \le m \le n)$$

mit Konstanten $A(k,m)$, die nicht von h abhängen.

Kapitel A

Hilfsmittel

A.1 Einige Hilfsmittel

Es werden einige Hilfsmittel aus der Analysis zusammengefasst, die im Laufe der Vorlesung benötigt werden.
Der Zwischenwertsatz besagt, dass eine stetige Funktion jeden Wert zwischen zwei Funktionswerten $f(a)$ und $f(b)$ annimmt.

Satz 1.1.1 (Zwischenwertsatz)

Sei $f : [a, b] \to \mathbb{R}$ stetig in dem abgeschlossenen Intervall $[a, b]$ und sei y eine beliebige Zahl mit $f(a) \leq y \leq f(b)$ oder $f(b) \leq y \leq f(a)$. Dann existiert ein Punkt $c \in [a, b]$ mit $f(c) = y$.

Der Mittelwertsatz erlaubt es, $f'(x)$ in (a, b) durch

$$f'(x) \approx f'(\xi) = \frac{f(b) - f(a)}{b - a}$$

zu approximieren.

Satz 1.1.2 (Mittelwertsatz)

Sei $f : [a, b] \to \mathbb{R}$ stetig und stetig differenzierbar im offenen Intervall (a, b). Dann gilt

$$f(b) = f(a) + (b - a)f'(\xi)$$

für ein $\xi \in (a, b)$.

Ist $f = (f_1, \ldots, f_m)^\top$ vektorwertig, so kann der Mittelwertsatz zwar auf jede Komponente von f einzeln angewendet werden, jedoch mit unterschiedlichen Zwischenstellen ξ_i, $i = 1, \ldots, m$. Eine alternative Darstellung für vektorwertige Funktionen f liefert der Mittelwertsatz in Integralform, welcher die Jacobimatrix von f verwendet.

Definition 1.1.3 (Jacobimatrix)

Exisitieren für $f : \mathbb{R}^n \to \mathbb{R}^m$ in $x \in \mathbb{R}^n$ die partiellen Ableitungen $\frac{\partial f_j}{\partial x_i}$ für $i = 1, \ldots, n$, $j = 1, \ldots, m$, so heißt

$$f'(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{pmatrix}$$

die **Jacobimatrix von** f **in** x.

Satz 1.1.4 **(Mittelwertsatz in Integralform)**

Sei $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^m$ *stetig differenzierbar in der offenen Menge* D *und sei* $x \in D$. *Sei* $h \in \mathbb{R}^n$ *ein Vektor derart, dass* $x + th \in D$ *für alle* $0 \leq t \leq 1$.
Dann gilt

$$f(x+h) - f(x) = \int_0^1 f'(x+th)h dt.$$

Neben den Mittelwertsätzen ist der Satz von Taylor in der Numerik sehr wichtig. Dieser gilt analog auch im $\mathbb{R}^n$.

Satz 1.1.5 **(Satz von Taylor)**

Sei $f : [a,b] \to \mathbb{R}$ $(n+1)$*-mal stetig differenzierbar in* $[a,b]$. *Dann gilt für jedes* $x, x_0 \in [a,b]$

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k + E_{n+1}(x;x_0).$$

Es gelten folgende Restglieddarstellungen:

(i)

$$E_{n+1}(x;x_0) = \int_{x_0}^{x} \frac{(x-t)^n}{n!} f^{(n+1)}(t) dt.$$

(ii) Restglieddarstellung nach Lagrange:

$$E_{n+1}(x;x_0) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1},$$

wobei $\xi = \xi(x,x_0)$ *zwischen* x *und* x_0 *liegt.*

(iii) Restglieddarstellung nach Cauchy:

$$E_{n+1}(x;x_0) = \frac{f^{(n+1)}(\xi)}{n!}(x-\xi)^n(x-x_0),$$

wobei $\xi = \xi(x,x_0)$ *zwischen* x *und* x_0 *liegt.*

Definition 1.1.6 **(Taylorpolynom)**

Das Polynom

$$T_n(x;x_0) := \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

heißt n**-tes Taylorpolynom von** f **in** x_0.

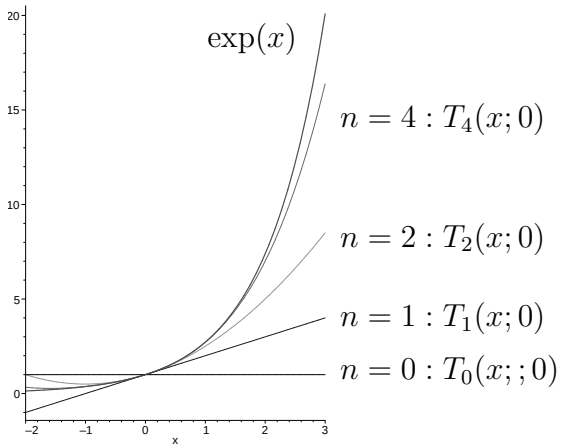
Falls f $(n+1)$-mal stetig differenzierbar in $[a,b]$ ist, kann f in $x \in [a,b]$ durch das n-te Taylorpolynom in $x_0 \in (a,b)$ approximiert werden. Der Approximationsfehler in x kann abgeschätzt werden durch

$$\begin{aligned} |f(x) - T_n(x;x_0)| &= |E_{n+1}(x;x_0)| \\ &= \left|\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}\right| \\ &\le \max_{\xi\in[a,b]}\left|\frac{f^{(n+1)}(\xi)}{(n+1)!}\right| \cdot |x-x_0|^{n+1} \\ &= C \cdot |x-x_0|^{n+1}. \end{aligned} \tag{A.1}$$

Beispiel 1.1.7

Approximation von $f(x) = \exp(x)$ *in* $x_0 = 0$ *durch* $T_n(x;0)$*:*

$$\begin{aligned} T_0(x;0) &= 1, \\ T_1(x;0) &= 1 + x, \\ T_2(x;0) &= 1 + x + \frac{x^2}{2!}, \\ &\vdots \\ T_n(x;0) &= \sum_{i=0}^{n} \frac{x^i}{i!}. \end{aligned}$$



Bemerkung 1.1.8

Formal kann die **Taylorreihe** *definiert werden als*

$$f(x) \sim \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k,$$

falls die Ableitungen existieren. Allerdings gilt i.a. **nicht**, *dass die Reihe tatsächlich existiert und gleich $f(x)$ ist! Eine hinreichende Bedingung für Gleichheit ist, dass der Fehler $E_{n+1}(x; x_0)$ gegen Null strebt für $n \to \infty$.*

A.1.1 $\mathcal{O}$ und o Notation

Häufig wird es notwendig sein, zwei Funktionen $f(x)$ und $g(x)$ miteinander zu vergleichen, wenn x gegen einen Wert x_0 strebt.

Definition 1.1.9

Seien $f, g : D \subseteq \mathbb{R}^n \to \mathbb{R}$ gegeben.

(a) Wir schreiben

$$f(x) = \mathcal{O}(g(x)) \qquad \text{für } x \to x_0,$$

falls

$$\limsup_{x \to x_0} \frac{|f(x)|}{|g(x)|} < \infty.$$

(b) Wir schreiben

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

falls

$$\lim_{x \to x_0} \frac{f(x)}{g(x)} = 0.$$

Mit Hilfe der $\mathcal{O}$-Notation können wir (A.1) schreiben als

$$f(x) - T_n(x; x_0) = \mathcal{O}(|x - x_0|^{n+1}) \qquad \text{für } x \to x_0.$$

Ist f stetig differenzierbar, so gilt

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + o(\|x - x_0\|)$$

mit $\lim_{x \to x_0} \frac{o(\|x-x_0\|)}{\|x-x_0\|} = 0$.

Kapitel B

Software

Available software in the world wide web:

- SCILAB: A Free Scientific Software Package; http://www.scilab.org
- GNU OCTAVE: A high-level language, primarily intended for numerical computations; http://www.octave.org/octave.html
- GAMS: Guide to Available Mathematical Software; http://gams.nist.gov/
- NETLIB: collection of mathematical software, papers, and databases; http://www.netlib.org/
- NEOS GUIDE: www-fp.mcs.anl.gov/otc/Guide

Empfohlene Literatur für Numerische Mathematik I:

- Deuflhard, P. und Hohmann, A. *Numerische Mathematik*. de Gruyter, Berlin, 1991.
- Hämmerlin, G. und Hoffmann, K.-H. *Numerische Mathematik*. Springer, Berlin-Heidelberg-New York, 1994, 4. Auflage.
- Schaback, R. und Werner, H. *Numerische Mathematik*. Springer, Berlin-Heidelberg-New York, 1992, 4. Auflage.
- Stoer, J. *Numerische Mathematik 1*. Springer, Berlin-Heidelberg-New York, 1993, 6. Auflage.
- Opfer, G. *Numerische Mathematik für Anfänger*. Vieweg, 2008, 5. Auflage.
- Kincaid, D. und Cheney, W. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole Series in Advanced Mathematics, Thomson Learning, 3rd Edition, 2002.
- Lempio, F. *Numerische Mathematik I und II*. Bayreuther Mathematische Schriften, 51 und 55, 1997 und 1998.

Literaturverzeichnis

[DB94] Deuflhard, P. and Bornemann, F. *Numerische Mathematik II*. de Gruyter, Berlin, 1994.

[Dem91] Demailly, J.-P. *Gewöhnliche Differentialgleichungen*. Vieweg, Braunschweig, 1991.

[DH91] Deuflhard, P. and Hohmann, A. *Numerische Mathematik*. de Gruyter, Berlin, 1991.

[GK99] Geiger, C. and Kanzow, C. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer, Berlin-Heidelberg-New York, 1999.

[HH94] Hämmerlin, G. and Hoffmann, K.-H. *Numerische Mathematik*. Springer, Berlin-Heidelberg-New York, 4th edition, 1994.

[JS04] Jarre, F. and Stoer, J. *Optimierung*. Springer, Berlin-Heidelberg-New York, 2004.

[Kan07] Kanzow, C. *Numerische Mathematik I*. Vorlesungsskript, Institut für Mathematik, Universität Würzburg, 2007.

[KC02] Kincaid, D. and Cheney, W. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole–Thomson Learning, Pacific Grove, CA, 3rd edition, 2002.

[Kel95] Kelley, C. T. *Iterative Methods for Solving Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1995.

[KL94] Kortüm, W. and Lugner, P. *Systemdynamik und Regelung von Fahrzeugen*. Springer, Berlin-Heidelberg-New York, 1994.

[Kos93] Kosmol, P. *Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben*. Teubner, Stuttgart, 2nd edition, 1993.

[Lem97] Lempio, F. *Numerische Mathematik I – Methoden der linearen Algebra*. volume 51 of *Bayreuther Mathematische Schriften*. Bayreuth, 1997.

[Lem98] Lempio, F. *Numerische Mathematik II – Methoden der Analysis*. volume 55 of *Bayreuther Mathematische Schriften*. Bayreuth, 1998.

[Opf08] Opfer, G. *Numerische Mathematik für Anfänger*. Vieweg, 5th edition, 2008.

[Sto93] Stoer, J. *Numerische Mathematik I*. Springer, Berlin-Heidelberg-New York, 6th edition, 1993.

[SW92] Schaback, R. and Werner, H. *Numerische Mathematik*. Springer, Berlin-Heidelberg-New York, 4th edition, 1992.