Heinrich Heine
Universität
Düsseldorf

Mathematisches Institut
Lehrstuhl für Angewandte Mathematik
Prof. Dr. Christiane Helzel

Skript zur Vorlesung

Numerik 1

Vorwort

Die *numerische Mathematik* widmet sich der Entwicklung und Analyse von Verfahren zur näherungsweisen Lösung mathematischer Aufgaben.

In dieser ersten Vorlesung werden wir klassische numerische Verfahren kennenlernen, die zum Teil auf Wissenschaftler wie Gauß und Newton zurückgehen, die mit Hilfe dieser Verfahren tiefgreifende Fragestellungen der Naturwissenschaften von Hand quantifizieren und verstehen wollten. Durch den Einsatz von Computern können wir heutzutage numerische Verfahren für sehr komplexe Problemstellungen nutzen.

In der Numerik 1 beschäftigen wir uns mit der Interpolation, der numerischen Integration, der Lösung linearer und nichtlinearer Gleichungssysteme sowie mit der Lösung von Ausgleichsproblemen. Im ersten Kapitel besprechen wir grundlegende Konzepte, die bei der Lösung mathematischer Probleme unter Verwendung numerischer Verfahren zu beachten sind.

Wichtige Literatur bei der Ausarbeitung meiner Vorlesungen waren die Lehrbücher von Stoer/Bulirsch, *Numerische Mathematik 1*, Springer, P. Deuflhard, A. Hohmann, *Numerische Mathematik 1*, de Gruyter sowie das Vorlesungsskript *Numerik 0* von R. Rannacher. Einige Beispiele stammen aus dem Buch von Cleve B. Moler, *Numerical Computing with MATLAB*, SIAM. All diese Bücher sind natürlich auch als Begleitliteratur sehr gut geeignet.

Im Sommersemester 2020 werde ich verstärkt auf ebooks verweisen, die Ihnen über die HHU Bibliothek zur Verfügung stehen. Dazu gehört das bereits erwähnte Buch von Stoer und Bulirsch. Außerdem gibt es einige interessante Neuerscheinungen, z.B. T. Richter und T. Wick, *Einführung in die Numerische Mathematik* Begriffe, Konzepte und zahlreiche Anwendungen, Springer Spektrum 2017, S. Bartels, *Numerik* 3×9, Springer Spektrum 2016, A. Meister und T. Sonar, *Numerik* Eine lebendige und gut verständliche Einführung mit vielen Beispielen, Springer Spektrum 2019.

Für Ihre Hilfe bei der Erstellung früherer Versionen dieses Vorlesungsskripts danke ich Caroline Albrecht und Maximilian Schneiders.

Zuletzt bearbeitet: 19. Juni 2021

Inhaltsverzeichnis

1 Grundlegende Konzepte der Numerik

1.1 Typische Aufgabenstellungen in der Numerik

In der Numerik beschäftigen wir uns mit der praktischen Berechnung mathematischer Objekte, z.B.

- Berechnung von Integralen
$$\int_a^b f(x)dx$$
- Berechnung eines
$$\min_{x\in[a,b]} F(x)$$
- Lösung (nichtlinearer) Gleichungen / Gleichungssystemen
$$f(x) = 0$$
- Lösung linearer Gleichungssysteme
$$Ax = b$$
- Lösung von Eigenwertproblemen
$$Ax = \lambda x$$
- Lösung von Differentialgleichungen
$$y'(t) = f(t, y(t))$$

Abstrakt lassen sich diese Aufgabenstellungen wie folgt formulieren.

Definition 1.1. Eine *mathematische Aufgabe* besteht in der Auswertung einer Abbildung

$$\phi : X \to Y \quad \text{bei } x \in X.$$

Die Räume X und Y werden wir später spezifizieren.

Viele der oben erwähnten mathematischen Objekte sind nicht durch geschlossene Formeln definiert und können daher nur näherungsweise angegeben werden. Außerdem ist nicht jede beispielsweise reelle Zahl exakt auf einem Computer darstellbar. Dies führt unvermeidlich zu *Rundungsfehler*. Weitere Fehlerquellen sind *Modellfehler*, die bei der vereinfachten mathematischen Beschreibung komplexer Vorgänge auftreten, sowie *Datenfehler*, die durch (ungenaue) Messungen verursacht sein können.

In der Numerik beschäftigen wir uns mit den folgenden Fragestellungen:

- **Algorithmik:** Angabe von Berechnungsverfahren / Algorithmen zur (näherungsweisen) Lösung mathematischer Aufgaben
- **Konditionierung und Stabilität:** Einfluß von Störungen (Fehlern) auf das Ergebnis einer mathematischen Aufgabe
- **Konvergenz:** Abschätzung des Fehlers zwischen berechneter und exakter Lösung
- **Komplexität:** Aufwand der Verfahren

1.2 Gleitkommadarstellung

Ein Computer kann Zahlen nur mit endlich vielen Ziffern darstellen. Dies führt zwangsläufig zu Fehlern in numerischen Algorithmen. In diesem Kapitel beschäftigen wir uns mit der Darstellung von Zahlen in einem Computer sowie mit den dadurch verbundenen Problemen von numerischen Algorithmen.

Zahlen werden in einem Computer mit Hilfe von normalisierten Gleitkommazahlen realisiert. Es sei $x \in \mathbb{R}$ eine reelle Zahl. Dann lässt sich diese Zahl in der Form

$$x = \pm m \cdot b^{\pm e}$$

schreiben. Dabei bezeichen wir mit m die *Mantisse*, mit b die *Basis* (im Binärsystem ist $b = 2$ und im Dezimalsystem ist $b = 10$) und mit e den *Exponenten*. Zur Vereinfachung wird der Exponent e als $e = c - B$ mit einer positiven Zahl $c \in \mathbb{N}$ und dem Bias $B \in \mathbb{N}$ geschrieben. Der Biaswert B wird in einer konkreten Zahldarstellung fest gewählt und bestimmt somit den größtmöglichen negativen Exponenten. Der variable Exponentenanteil c wird selbst in Binärformat gespeichert.

Durch den ANSI/IEEE Standard (American National Standard Institute, Institute of Electrical and Electronics Engineers, seit 1985) werden im Computer Darstellungen für

binäre Gleitkommazahlen definiert, das heißt wir betrachten Zahlen der Form

$$x = \pm(1+f)2^{c-1023},$$

wobei die Binärdarstellung von f maximal 52 Bits benutzt, das heißt es gilt

$$f = f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + \cdots + f_{52} \cdot 2^{-52}$$

mit $f_i \in \{0,1\}$. Die Zahl c wird wie folgt charakterisiert: Es werden 11 Bits für den Exponenten vergeben, das heißt wir erhalten

$$c = c_0 \cdot 2^0 + c_1 \cdot 2^1 + \cdots + c_{10} \cdot 2^{10} \quad \text{mit} \quad c_i \in \{0,1\},$$

also $0 \leq c \leq 2^0 + 2^1 + \cdots + 2^{10} = 2047$ und damit

$$-1023 \leq e \leq 1024.$$

Die Werte $e = -1023$ und $e = 1024$ werden zur Speicherung besonderer „Zahlen“ verwendet:

- $e = 1024$ und $f = 0$: inf (infinity).
- $e = 1024$ und $f \neq 0$: nan (not a number).
- $e = -1023$ und $f = 0$, $c = 0$: Darstellung der Null.

Für alle anderen Zahlen gilt $-1022 \leq e \leq 1023$. Gleitkommazahlen (mit doppelter Genauigkeit) werden mit insgesamt 64 Bits gespeichert:

- 52 Bits für die Mantisse,
- 11 Bits für den Exponenten,
- 1 Bit für das Vorzeichen (0 entspricht +, 1 entspricht -) .

Bemerkung. Die Verwendung der Gleitkommadarstellung im numerischen Rechnen ist wesentlich, um Zahlen sehr unterschiedlicher Größe bearbeiten zu können. Des Weiteren liefert die Darstellung von m eine Beschränkung an die Genauigkeit und die Darstellung von e eine Beschränkung der Größenordnung.

Es gelte nun $-1022 \leq e \leq 1023$. Dann gilt für die größte/kleinste Gleitkommazahl

$$x_{\max,\min} = \pm[1 + 2^{-1} + 2^{-2} + \cdots + 2^{-52}] \cdot 2^{1023} \approx \pm 1{,}8 \cdot 10^{308}.$$

Für die kleinste positive/größte negative Gleitkommazahl erhalten wir

$$x_{\text{posmin}} = [1 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + \cdots + 0 \cdot 2^{-52}] \cdot 2^{-1022} \approx 2{,}2 \cdot 10^{-308},$$
$$x_{\text{negmax}} = -2^{-1022} \approx -2{,}2 \cdot 10^{-308}.$$

Bemerkung. Gleitkommazahlen sind nicht äquivalent verteilt. Insbesondere gibt es einen relativ großen Abstand zwischen Null und der ersten positiven Gleitkommazahl.

Seien x_1 und x_2 die beiden kleinsten positiven Gleitkommazahlen in der oben beschriebenen Standarddarstellung. Dann gilt

$$x_1 = 2^{-1022},\ x_2 = (1 + 2^{-52})2^{-1022}$$

und somit

$$\frac{x_1 - 0}{x_2 - x_1} = 2^{52}.$$

Diese Lücke wird mit subnormalen Gleitkommazahlen gefüllt. Diese haben die Form

$$(0 + f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + \ldots + f_{52} \cdot 2^{-52})2^{-1022}.$$

Zur Speicherung dieser Zahlen benutzt man den Speicherplatz von $e = -1023$. Die Darstellung der Null ergibt sich für $f = 0$, $e = -1023$. Die kleinste positive, von Null verschiedene subnormale Gleitkommazahl ist

$$2^{-52} \cdot 2^{-1022} = 2^{-1074} \approx 5 \cdot 10^{-324}.$$

Mit $A = A(b, f, c) = A(b, \{f\}, \{c\}, B)$ bezeichnen wir das *numerische Gleitkommagitter*. Die Menge A ist die Menge der in der Form $x = \pm(1 + f) \cdot b^{c-1023}$ darstellbaren Maschinenzahlen. Dabei beschreibt f die Darstellung der Mantisse und c die Darstellung des Exponenten. Der zulässige Bereich wird definiert als

$$D := [x_{\min}, x_{\text{negmax}}] \cup \{0\} \cup [x_{\text{posmin}}, x_{\max}].$$

Wir definieren auf D die Rundungsoperation $\mathrm{rd} : D \to A$ mit

$$|x - \mathrm{rd}(x)| = \min_{y \in A}(|x - y|) \quad \text{für alle} \quad x \in D.$$

Beispiel:

$$\mathrm{rd}(x) = \mathrm{sign}(x) \cdot \begin{cases} (1 + f_1 \cdot 2^{-1} + \cdots + f_{52} \cdot 2^{-52}) \cdot 2^{c-1023}, & f_{53} = 0 \text{ oder} \\ & f_{53} = 1, f_{52} = 0 \\ (1 + f_1 \cdot 2^{-1} + \cdots + f_{52} \cdot 2^{-52} + 2^{-52}) \cdot 2^{c-1023}, & f_{53} = 1, f_{52} = 1. \end{cases}$$

(Man nimmt dabei an, dass der Rechner intern mit mehr Stellen arbeitet.) Dann gilt:

- absoluter Rundungsfehler: $|x - \mathrm{rd}(x)| \leq \frac{1}{2} \cdot 2^{-52} \cdot 2^e$,
- relativer Rundungsfehler: $\left|\frac{x - \mathrm{rd}(x)}{x}\right| \leq \frac{1}{2} \cdot \frac{2^{-52}}{(1+f)} \leq 2^{-53}$,

- maximaler Rundungsfehler $\max\limits_{x\in D, x\neq 0} \left|\frac{\mathrm{rd}(x)-x}{x}\right|$.

Die *Maschinengenauigkeit*, die mit eps bezeichnet wird, gibt den Abstand zwischen 1 und der nächstgrößeren Gleitkommazahl bzw. den relativen Abstand zwischen Gleitkommazahlen an. Im IEEE-Format gilt

$$\mathrm{eps} = 2^{-52} \approx 2{,}2204 \cdot 10^{-16}.$$

Listing 1: Python Programm zur Berechnung der Maschinengenauigkeit

```
myeps = 1.
while (1.+myeps)>1.:
    myeps = myeps/2.

print(2.*myeps)
```

```
2.220446049250313e-16
```

Für $x \in D$ gilt $\mathrm{rd}(x) = x(1+\varepsilon)$ mit $|\varepsilon| \leq \mathrm{eps}$. Die Grundoperationen $* \in \{+, -, \cdot, /\}$ werden im Computer durch Maschinenoperationen $\circledast \in \{\oplus, \ominus, \odot, \oslash\}$ ersetzt. Diese sind so definiert: Für $x, y \in A$ und $x * y \in D$ gilt $x \circledast y = (x * y)(1+\varepsilon)$ mit $|\varepsilon| \leq \mathrm{eps}$.

Bemerkung. Für $x, y, z \in A$ gilt:

(i) $(x \oplus y) \oplus z \neq x \oplus (y \oplus z)$ (kein Assoziativgesetz),

(ii) $(x \oplus y) \odot z \neq (x \odot z) \oplus (y \odot z)$ (kein Distributivgesetz),

(iii) $x \oplus y = x$ für $|y| \leq \frac{|x|}{2}\mathrm{eps}$.

Aus (iii) lässt sich die Größe der Maschinengenauigkeit eps experimentell bestimmen (vergleiche Python-Code 1).

1.3 Kondition und Stabilität

Wir betrachten zunächst ein Beispiel, das die Auswirkung von Störungen auf die Lösung eines Problems illustriert.

Beispiel. Betrachte zunächst das lineare Gleichungssystem

$$\begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix} x = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix}$$

mit der Lösung $x = (2, -2)^T$.

Durch geringe Störung der rechten Seite erhalten wir das System

$$\begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix} \tilde{x} = \begin{pmatrix} 0.86419999 \\ 0.14400001 \end{pmatrix}$$

mit der Lösung $x = (0.9911, -0.487)^T$.

Obwohl die Störung sehr klein ist unterscheiden sich die Lösungen x und $\tilde{x}$ sehr stark. Der Python-Code 2 zeigt die Implementation dieses Testproblems unter Verwendung von NumPy.

Listing 2: Beispiel für eine schlecht konditionierte Aufgabe

```
import numpy as np

A = np.array([[1.2969, 0.8648], [0.2161, 0.1441]])
b1 = np.array([0.8642,0.1440])
x1 = np.linalg.solve(A,b1)
#%%
b2 = np.array([0.86419999, 0.14400001])
x2 = np.linalg.solve(A,b2)
```

```
In [1]: x1
Out[1]: array([2.,-2.])

In [2]: x2
Out[2]: array([0.9911,-0.487])
```

Die Untersuchung des Einflußes von Störungen auf die Lösung eines mathematischen Problems führt uns auf den Begriff der Kondition. Wir definieren den Begriff zunächst für mathematische Aufgaben, die durch Abbildungen $\phi : \mathbb{R} \to \mathbb{R}$ beschrieben werden können.

Definition 1.2. Eine mathematische Aufgabe heißt *schlecht konditioniert*, wenn kleine Störungen in den Daten große relative Fehler in der Lösung verursachen. Anderenfalls heißt die Aufgabe *gut konditioniert*.

Eine Aufgabe $\phi : \mathbb{R} \to \mathbb{R}$ ist an der Stelle x schlecht konditioniert, wenn eine Störung $\tilde{x}$ existiert mit

$$\frac{|\phi(\tilde{x}) - \phi(x)|}{|\phi(x)|} \gg \frac{|\tilde{x} - x|}{|x|},$$

wobei $|x| \neq 0$ und $|\phi(x)| \neq 0$ gelte.

Die Relation $a \gg b$ bedeutet, dass a wesentlich größer als b ist (z.B. $a = 10^3 b$). Was genau als wesentlich größer angesehen wird ist allerdings problemabhängig und kann nicht genauer definiert werden.

Sei $\tilde{x} := x + \Delta x$ ein gestörter Eingabewert und $\phi(\tilde{x}) - \phi(x)$ der dadurch entstandene Fehler im Ergebnis. Dann gilt:

$$\frac{\phi(\tilde{x}) - \phi(x)}{\phi(x)} = \frac{\phi(x + \Delta x) - \phi(x)}{\phi(x)}$$

$$\underbrace{\frac{\phi(\tilde{x}) - \phi(x)}{\phi(x)}}_{\text{relativer Fehler im Ergebnis}} = \frac{\phi(x + \Delta x) - \phi(x)}{\Delta x} \frac{x}{\phi(x)} \qquad \underbrace{\frac{\Delta x}{x}}_{\text{relativer Eingabefehler}}$$

Für $\phi \in C^1(\mathbb{R}, \mathbb{R})$ und kleines Δx können wir den Differenzenquotienten durch $\phi'(x)$ ersetzen. Wir erhalten näherungsweise die Beziehung

$$\left| \frac{\phi(\tilde{x}) - \phi(x)}{\phi(x)} \right| \approx \left| \phi'(x) \frac{x}{\phi(x)} \right| \cdot \left| \frac{\Delta x}{x} \right|.$$

Die Größe

$$\kappa = \left| \phi'(x) \frac{x}{\phi(x)} \right|$$

heißt Konditionszahl. Sie beschreibt die relative Fehlerverstärkung von Eingabefehlern.

Im folgenden Satz werden wir die Kondition einer mathematische Aufgabe $\phi : \mathbb{R}^2 \to \mathbb{R}$ untersuchen.

Satz 1.3. *Die Aufgabe $\phi(x, y) = x \cdot y$ ist gut konditioniert.*

Beweis.

$$\begin{aligned}
\frac{|\phi(\tilde{x}, \tilde{y}) - \phi(x, y)|}{|\phi(x, y)|} &= \frac{|\tilde{x}\tilde{y} - xy|}{|xy|} \\
&= \frac{|(\tilde{x} - x)\tilde{y} + x(\tilde{y} - y)|}{|xy|} \\
&\leq \frac{|\tilde{x} - x|}{|x|} \cdot \frac{|\tilde{y} - y + y|}{|y|} + \frac{|\tilde{y} - y|}{|y|} \\
&\leq \frac{|\tilde{x} - x|}{|x|} \cdot \frac{|\tilde{y} - y|}{|y|} + \frac{|\tilde{x} - x|}{|x|} + \frac{|\tilde{y} - y|}{|y|}
\end{aligned}$$

Seien $\epsilon_x := \frac{|\tilde{x} - x|}{|x|}$, $\epsilon_y := \frac{|\tilde{y} - y|}{|y|}$ die relativen Fehler in x und y. Dann gilt für $\phi(x, y) = x \cdot y$:

$$\frac{|\phi(\tilde{x}, \tilde{y}) - \phi(x, y)|}{|\phi(x, y)|} \leq \epsilon_x + \epsilon_y + \epsilon_x \epsilon_y$$

Für kleine relative Fehler ϵ_x, ϵ_y ist auch der Fehler im Ergebnis klein. □

Bemerkung. Die gute Konditionierung einer mathematischen Aufgabe ist notwendig, um das Problem numerisch sinnvoll lösen zu können, da Rundungsfehler anderenfalls große Fehler verursachen können.

Definition 1.4. Ein *Verfahren* oder *Algorithmus* zur (nährungsweisen) Lösung einer mathematischen Aufgabe ϕ ist eine Abbildung $\tilde{\phi} : X \to Y$, die durch die Hintereinanderausführung endlich vieler oder abzählbar unendlich vieler elementarer möglicherweise rundungsfehlerbehafteter Rechenoperationen $\phi^{(k)}$, $k = 1, 2, \ldots$ definiert ist, d.h.

$$\tilde{\phi} = \ldots \circ \phi^{(2)} \circ \phi^{(1)}.$$

Bemerkung. Eine mathematische Aufgabe kann man oft mit verschiedenen Verfahren approximieren. Bei der Ausführung eines Algorithmus (auf einem Computer) treten in jedem Schritt Fehler auf (z.B. Rundungsfehler), die sich im Verlauf der Ausführung akkumulieren können. Von einem guten Algorithmus erwarten wir, dass die im Verlauf der Ausführung akkumulierten Fehler den durch die Kondition der Aufgabe bedingten unvermeidbaren Fehler nicht wesentlich übersteigen.

Definition 1.5. Ein Algorithmus $\tilde{\phi}$ heißt *instabil*, wenn es eine Störung $\tilde{x}$ von x gibt, so dass der durch Rundungsfehler und Störungen verursachte relative Fehler erheblich größer ist als der nur durch die Störung verursachte Fehler, d.h., falls $\phi(x) \neq 0$ und

$$\frac{|\tilde{\phi}(\tilde{x}) - \phi(x)|}{|\phi(x)|} \gg \frac{|\phi(\tilde{x}) - \phi(x)|}{|\phi(x)|}.$$

Ein Algorithmus heißt *stabil*, wenn er nicht instabil ist.

Beispiel. Wir betrachten die mathematische Aufgabe

$$\phi(x) = \frac{1}{x(x+1)}.$$

Zunächst zeigen wir, das diese Aufgabe für große Zahlen $x \in \mathbb{R}$, $x \gg 0$ gut konditioniert ist. Dazu betrachten wir (gemäß Def. 1.2)

$$\frac{|\phi(x) - \phi(\tilde{x})|}{|\phi(x)|}$$

für kleine Störungen $\tilde{x} = x(1 + \epsilon)$, $\epsilon \in \mathbb{R}$, $|\epsilon| \ll 1$.

Es gilt:

$$\begin{aligned}
\left|\frac{\phi(x)-\phi(\tilde{x})}{\phi(x)}\right| &= \left|1-\frac{x(x+1)}{x(1+\epsilon)\left(x(1+\epsilon)+1\right)}\right| \\
&= \left|1-\frac{x+1}{x(1+\epsilon)^2+(1+\epsilon)}\right| \\
&= \left|\frac{x(1+\epsilon)^2+(1+\epsilon)-x-1}{x(1+\epsilon^2)+(1+\epsilon)}\right| \\
&= \left|\frac{2\epsilon x+\epsilon^2 x+\epsilon}{x+\epsilon^2 x+1+\epsilon}\right| \\
&< \left|\frac{2\epsilon x+\epsilon^2 x+\epsilon}{x}\right| \\
&< 2|\epsilon|+\epsilon^2+\frac{|\epsilon|}{x} \\
&< 4|\epsilon|
\end{aligned}$$

Aufgrund der Gleichheit

$$\frac{1}{x(x+1)} = \frac{1}{x} - \frac{1}{x+1}$$

bieten sich zwei mögliche Algorithmen zur Berechnung dieser mathematischen Aufgabe an:

$$\tilde{\phi}_1(x) := \frac{1}{(x(x+1))}, \quad \tilde{\phi}_2(x) := \left(\frac{1}{x}\right) - \left(\frac{1}{x+1}\right).$$

Die Klammern beschreiben die Reihenfolge der Ausführung der elementaren Operationen. Beide Formeln lassen sich leicht in Python (oder einer anderen Umgebung) auswerten. In Tabelle 1.1 sind die Ergebnisse solch einer Rechnung für verschiedene x Werte dargestellt. Die x Werte wurden so gewählt, dass man sich leicht von der Genauigkeit der mittels Algorithmus $\tilde{\phi}_1$ erzielten Ergebnisse überzeugen kann, während Algorithmus $\tilde{\phi}_2$ ungenaue Ergebnisse liefert.

x	$\frac{1}{(x(x+1))}$	$\left(\frac{1}{x}\right) - \left(\frac{1}{x+1}\right)$
10	0.009090909090909	0.009090909090909
10^3	$9.990009990009991 \cdot 10^{-7}$	$9.990009990010207 \cdot 10^{-7}$
10^5	$9.999900000999990 \cdot 10^{-11}$	$9.999900001006240 \cdot 10^{-11}$
10^7	$9.999999000000100 \cdot 10^{-15}$	$9.999998990718643 \cdot 10^{-15}$

Tabelle 1.1: Numerische Resultate für die Auswertung der beiden Formeln.

Algorithmus $\tilde{\phi}_2$ ist instabil anfgrund von *Auslöschungseffekten*, die bei der Subtraktion nahezu gleich großer Zahlen auftreten können. Diesen Effekt werden wir in Abschnitt 1.6 genauer analysieren.

Bemerkung. Der durch Rundung und näherungsweises Lösen verursachte Fehler bei der Lösung einer Aufgabe lässt sich mittels des Fehlers aufgrund der Kondition der mathematischen Aufgabe und des Fehlers aufgrund der Wahl des Verfahrens abschätzen:

$$|\phi(x) - \tilde{\phi}(\tilde{x})| \leq \underbrace{|\phi(x) - \phi(\tilde{x})|}_{\text{Kondition}} + \underbrace{|\phi(\tilde{x}) - \tilde{\phi}(\tilde{x})|}_{\text{Verfahrenswahl}}$$

In Abschnitt 1.6 werden wir die Kondition für vektorwertige Abbildungen $\phi : \mathbb{R}^m \to \mathbb{R}^n$ untersuchen. Dabei werden wir auch eine neue Notation, die in Abschnitt 1.5 eingeführt wird, verwenden.

1.4 Komplexität numerischer Verfahren

Neben der Stabilität ist der Rechenaufwand (die Komplexität) eine wichtige Größe bei der Bewertung numerischer Verfahren.

Definition 1.6. Für eine Aufgabe $\phi : X \to Y$ und ein zugehöriges Verfahren $\tilde{\phi} : X \to Y$ ist der *Aufwand* die Anzahl der benötigten Rechenoperationen von $\tilde{\phi}$.

Eine exakte Bestimmung des Aufwands ist nicht notwendig. Stattdessen wird der Aufwand in Abhängigkeit relevanter Systemgrößen angegeben.

Beispiel. Bei der Lösung eines linearen Gleichungssystems

$$Ax = b, \quad A \in \mathbb{R}^{n \times n},\, b \in \mathbb{R}^n$$

ist man am Aufwand in Abhängigkeit von n interessiert.

Zur Beschreibung der Komplexität von Algorithmen verwendet man die Landau Notation.

Definition 1.7. Die Folge $(a_n)_{n \in \mathbb{N}}$ ist (asymptotisch) von der Ordnung der Folge $(b_n)_{n \in \mathbb{N}}$, falls Zahlen $c > 0$ und $N \in \mathbb{N}$ existieren, sodass $|a_n| \leq c|b_n|$ für alle $n \geq N$ gilt. In diesem Fall verwenden wir die Landau Notation $a_n = \mathcal{O}(b_n)$.

Beispiel. Zur Berechnung des Skalarproduktes zweier Vektoren $x, y \in \mathbb{R}^n$ führen wir n Multiplikationen und $n - 1$ Additionen durch. Der Aufwand für diese Rechnung ist also $\mathcal{O}(n)$.

Die Multiplikation einer Matrix $A \in \mathbb{R}^{n \times n}$ mit einem Vektor $x \in \mathbb{R}^n$ kann mit Aufwand $\mathcal{O}(n^2)$ berechnet werden.

1.5 Landausymbole und Genauigkeit

Bei der Beschreibung und Abschätzung von Rundungs- und Verfahrensfehlern und deren Fortpflanzung in Algorithmen benutzt man ebenfalls Landausymbole. Die Landausymbole für Funktionen sind wie folgt definiert.

Definition 1.8. Es seien $f, g : D \subset \mathbb{R}^n \to \mathbb{R}$ zwei Funktionen und $x, x_0 \in D$.

(1) Die Funktion f wächst für $x \to x_0$ langsamer als g, symbolisch geschrieben

$$f = \mathrm{o}(g), \quad x \to x_0$$

(gesprochen: f ist ein klein o von g),

wenn

$$\lim_{x \to x_0} \frac{|f(x)|}{|g(x)|} = 0$$

gilt.

(2) Die Funktion f wächst für $x \to x_0$ nicht wesentlich schneller als g, symbolisch geschrieben

$$f = \mathcal{O}(g), \quad x \to x_0$$

(gesprochen: f ist ein groß O von g),

wenn

$$\exists c > 0, \exists \epsilon > 0 : |f(x)| \leq c|g(x)|$$

für alle x mit $|x - x_0| < \epsilon$ gilt.

(3) $f = \mathrm{o}(g)$, $x \to \infty$ und $f = \mathcal{O}(g)$, $x \to \infty$ werden analog definiert, spielen aber für unsere Fehlerbetrachtungen keine Rolle.

Beispiele. Es gilt:

$$e^x = 1 + x + \frac{x^2}{2!} + \mathcal{O}(x^3), \quad x \to 0,$$

denn für $c = \frac{1}{3}$ gilt

$$\left| \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots \right| \leq c|x^3|, \quad x \to 0.$$

Außerdem gilt

$$e^x = 1 + x + \frac{x^2}{2!} + \mathrm{o}(x^2), \quad x \to 0,$$

denn

$$\lim_{x \to 0} \frac{|\frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \ldots|}{|x^2|} = 0.$$

1.5.1 Rechenregeln für die Landausymbole

Für Funktionen $f, f_1, f_2, g, g_1, g_2 : D \subset \mathbb{R} \to \mathbb{R}$ und $x, x_0 \in D$ gelten die folgenden Regeln:

(1) $f = \mathrm{o}(g) \Rightarrow f = \mathcal{O}(g)$

(2) $f = \mathcal{O}(1) \Leftrightarrow f$ ist beschränkt

(3) $f = \mathrm{o}(1)$ für $x \to x_0 \Leftrightarrow \lim_{x \to x_0} f(x) = 0$

(4) $f_1 = \mathcal{O}(g), f_2 = \mathcal{O}(g) \Rightarrow f_1 + f_2 = \mathcal{O}(g)$

(5) $f_1 = \mathrm{o}(g), f_2 = \mathrm{o}(g) \Rightarrow f_1 + f_2 = \mathrm{o}(g)$

(6) $f_1 = \mathcal{O}(g_1),\ f_2 = \mathcal{O}(g_2) \Rightarrow f_1 \cdot f_2 = \mathcal{O}(g_1 \cdot g_2)$

(7) $f_1 = \mathcal{O}(g_1),\ f_2 = \mathrm{o}(g_2) \Rightarrow f_1 \cdot f_2 = \mathrm{o}(g_1 \cdot g_2)$

(8) $f_1 = \mathcal{O}(g_1), f_2 = \mathcal{O}(g_2) \Rightarrow f_1 + f_2 = \mathcal{O}\left(\max\{|g_1|, |g_2|\}\right)$

(9) $f = \mathcal{O}(g), c \in \mathbb{R} \Rightarrow cf = \mathcal{O}(g)$

(10) $f = \mathrm{o}(g), c \in \mathbb{R} \Rightarrow cf = \mathrm{o}(g)$

Die Beweise folgen aus den Definitionen der Landausymbole. Meister & Sonar (S. 14) beweisen exemplarisch (4).

1.5.2 Fehlerabschätzung in numerischen Verfahren

In der Numerik werden die Landausymbole oft verwendet um die Genauigkeit von Nährungsverfahren zu beschreiben. Dies wollen wir uns anhand eines Beispiels anschauen.

Beispiel. Sei $u : \mathbb{R} \to \mathbb{R}$, $u \in C^\infty(\mathbb{R})$. Wir möchten näherungsweise die erste Ableitung der Funktion u an einem vorgegebenen Punkt $\bar{x}$ bestimmen. Dazu benutzen wir die folgenden drei Verfahren:

(1) $$D_+ u(\bar{x}) := \frac{u(\bar{x} + h) - u(\bar{x})}{h}$$

(2) $$D_- u(\bar{x}) := \frac{u(\bar{x}) - u(\bar{x} - h)}{h}$$

(3) $$D_0 u(\bar{x}) := \frac{u(\bar{x} + h) - u(\bar{x} - h)}{2h}$$

Den Parameter h können wir dabei beliebig wählen. Wir erwarten, dass der Fehler kleiner wird, wenn wir h verkleinern. Für $u(x) = \sin(x)$, $\bar{x} = 1/2$, $u'(\bar{x}) = \cos(\bar{x}) = 0.877582\ldots$ erhalten wir die Näherungswerte aus Tabelle 1.2. Die Tabelle 1.3 gibt die Fehler der verschiedenen Verfahren in Abhängigkeit von h an. Wir beobachten, dass

h	$D_+u(\bar{x})$	$D_-u(\bar{x})$	$D_0u(\bar{x})$
$\frac{1}{2}$	0.724091	0.958851	0.841471
$\frac{1}{4}$	0.808852	0.928086	0.868469
$\frac{1}{8}$	0.845373	0.905224	0.875298
$\frac{1}{16}$	0.862034	0.891988	0.877011

Tabelle 1.2: Approximationen an $u'(\bar{x})$ für $u(x) = \sin(x)$, $\bar{x} = 1/2$.

h	$\lvert D_+u(\bar{x}) - u'(\bar{x})\rvert$	$\lvert D_-u(\bar{x}) - u'(\bar{x})\rvert$	$\lvert D_0u(\bar{x}) - u'(\bar{x})\rvert$
$\frac{1}{2}$	0.153492	0.081268	0.036111
$\frac{1}{4}$	0.068729	0.050503	0.009112
$\frac{1}{8}$	0.032208	0.027641	0.002283
$\frac{1}{16}$	0.015548	0.014405	0.000571

Tabelle 1.3: Absoluter Fehler der Differenzenverfahren.

sich der absolute Fehler bei Verwendung von $D_+u(\bar{x})$ und $D_-u(\bar{x})$ etwa halbiert, wenn h halbiert wird. Daher vermuten wir

$$|D_+u(\bar{x}) - u'(\bar{x})| = \mathcal{O}(h) \quad \text{für } h \to 0$$
$$|D_-u(\bar{x}) - u'(\bar{x})| = \mathcal{O}(h) \quad \text{für } h \to 0$$

Das Verfahren $D_0u(\bar{x})$ liefert den kleinsten Fehler. Halbiert man h, so verringert sich der Fehler etwa um einen Faktor vier. Wir vermuten also eine Beziehung der Form

$$|D_0u(\bar{x}) - u'(\bar{x})| = \mathcal{O}(h^2) \quad \text{für } h \to 0.$$

Diese Abschätzungen für den Fehler wollen wir nun beweisen. Es gilt (Taylorentwicklung):

$$u(\bar{x} + h) = u(\bar{x}) + u'(\bar{x})h + \frac{1}{2}u''(\bar{x})h^2 + \mathcal{O}(h^3)$$
$$u(\bar{x} - h) = u(\bar{x}) - u'(\bar{x})h + \frac{1}{2}u''(\bar{x})h^2 + \mathcal{O}(h^3)$$

Damit erhalten wir

$$\begin{aligned}
\frac{u(\bar{x}+h)-u(\bar{x})}{h} &= \frac{u(\bar{x})+u'(\bar{x})h+\mathcal{O}(h^2)-u(\bar{x})}{h} \\
&= u'(\bar{x})+\mathcal{O}(h) \\
\frac{u(\bar{x}+h)-u(\bar{x})}{h} &= u'(\bar{x})+\mathcal{O}(h) \\
\frac{u(\bar{x}+h)-u(\bar{x}-h)}{2h} &= \frac{u(\bar{x})+u'(\bar{x})h+\frac{1}{2}u''(\bar{x})h^2-u(\bar{x})+u'(\bar{x})h-\frac{1}{2}u''(\bar{x})+\mathcal{O}(h^3)}{2h} \\
&= \frac{2u'(\bar{x})h+\mathcal{O}(h^3)}{2h} \\
&= u'(\bar{x})+\mathcal{O}(h^2)
\end{aligned}$$

1.6 Differentielle Fehleranalyse

Wir beschäftigen uns nun noch einmal mit der Frage, wie sich Störungen in den Eingabedaten, unabhängig vom gewählten Algorithmus, auf das Resultat auswirken. Jetzt betrachten wir Aufgaben, die sich als Abbildungen $\phi : \mathbb{R}^m \to \mathbb{R}^n$ interpretieren lassen. D.h., unter einer numerischen Aufgabe verstehen wir nun die Berechnung endlich vieler Größen y_i $(i = 1, \ldots, n)$ aus Größen x_j $(j = 1, \ldots, m)$ mittels einer funktionalen Vorschrift $y_i = \phi_i(x_1, \ldots, x_m)$. Schreibweise: $y = \phi(x)$, $x = (x_1, \ldots, x_m)^T$, $y = (y_1, \ldots, y_m)^T$, $\phi = (\phi_1, \ldots, \phi_m)^T$.

Es sei $|\Delta x_j| \ll |x_j|$ (relativ kleiner Datenfehler) und $\phi_i = \phi_i(x_1, \ldots, x_m)$ zweimal stetig partiell differentierbar nach den Argumenten x_j. Dann gilt mit Taylorentwicklung:

$$\Delta y_i = \phi_i(x+\Delta x) - \phi_i(x) = \sum_{j=1}^{m} \frac{\partial \phi_i}{\partial x_j}(x)\Delta x_j + R_i^{\phi}(x, \Delta x), \quad i = 1, \ldots, m, \qquad (*)$$

wobei $R_i^{\phi}(x, \Delta x) = \mathcal{O}(|\Delta x|^2)$ das Restglied ist. Aus $(*)$ folgt in erster Näherung

$$\Delta y_i \approx \sum_{j=1}^{m} \frac{\partial \phi_i}{\partial x_j}(x)\Delta x_j,$$

das heißt Gleichheit gilt in erster Näherung bis auf einen Term der Ordnung $\mathcal{O}\left(|\Delta x|^2\right)$. Für den relativen Fehler (komponentenweise) folgt mit $y_i = \phi_i(x)$ also

$$\frac{\Delta y_i}{y_i} \approx \sum_{j=1}^{m} \frac{\partial \phi_i}{\partial x_j}(x)\frac{\Delta x_j}{y_i} = \sum_{j=1}^{m} \frac{\partial \phi_i}{\partial x_j}(x)\frac{x_j}{\phi_i(x)}\frac{\Delta x_j}{x_j}.$$

Definition 1.9. Die Größen

$$K_{i,j}(x) := \frac{\partial \phi_i}{\partial x_j}(x)\frac{x_j}{\phi_i(x)} \quad i = 1, \ldots, n, j = 1, \ldots, m$$

heißen *relative Konditionszahlen* der Funktion ϕ im Punkt x. Sie sind ein Maß dafür, wie sich kleine relative Fehler in den Ausgangsdaten im Ergebnis auswirken. Man nennt die Aufgabe, $y = \phi(x)$ aus x zu berechnen, *schlecht konditioniert*, wenn es ein i, j gibt mit $|K_{i,j}(x)| \gg 1$ ist, andernfalls *gut konditioniert*.

Grundoperationen

1) Wir betrachten für $x_1, x_2 \in \mathbb{R}$, $x_1 \cdot x_2 \neq 0$ die Addition $y = \phi(x_1, x_2) = x_1 + x_2$. Dann gilt

$$K_1 = \frac{\partial \phi}{\partial x_1} \cdot \frac{x_1}{\phi} = 1 \cdot \frac{x_1}{x_1 + x_2} = \frac{1}{1 + \frac{x_2}{x_1}},$$
$$K_2 = \frac{\partial \phi}{\partial x_2} \cdot \frac{x_2}{\phi} = 1 \cdot \frac{x_2}{x_1 + x_2} = \frac{1}{1 + \frac{x_1}{x_2}}.$$

Für $\frac{x_1}{x_2} \approx -1$ (also für die Subtraktion zweier fast gleich großer Zahlen) ist die Addition schlecht konditioniert.

Definition 1.10 (Auslöschung)**.** Unter *Auslöschung* versteht man den Verlust an wesentlichen Dezimalstellen bei der Subtraktion von Zahlen gleichen Vorzeichens. Dies ist gefährlich, falls eine oder beide Zahlen keine Maschinenzahlen sind und diese vor Ausführung der Operation gerundet werden.

Beispiel (Dezimale Gleitpunktrechnung mit vier signifikanten Ziffern). Gegeben seien

$$x_1 = 0{,}11258762 \cdot 10^2, \quad x_2 = 0{,}11244891 \cdot 10^2.$$

Da wir nur vier signifikante Ziffern betrachten, erhalten wir

$$\mathrm{rd}(x_1) = 0{,}1126 \cdot 10^2, \quad \mathrm{rd}(x_2) = 0{,}1124 \cdot 10^2$$

sowie jeweils einen absoluten Fehler von

$$|\Delta x_1| = 0{,}001238, \quad |\Delta x_2| = 0{,}004891.$$

Der relative Fehler in der Eingabe beträgt somit

$$\left|\frac{\Delta x_1}{x_1}\right| = 1{,}099 \cdot 10^{-4}, \quad \left|\frac{\Delta x_2}{x_2}\right| = 4{,}3495 \cdot 10^{-4}.$$

Nun gilt

$$x_1 + x_2 = 0{,}22503653 \cdot 10^2, \quad \mathrm{rd}(x_1) + \mathrm{rd}(x_2) = 0{,}2250 \cdot 10^2$$

und

$$x_1 - x_2 = 0{,}13871 \cdot 10^{-1}, \quad \mathrm{rd}(x_1) - \mathrm{rd}(x_2) = 0{,}2000 \cdot 10^{-1}.$$

Wir berechnen für den relativen Fehler der Subtraktion

$$\left|\frac{(x_1 - x_2) - (\mathrm{rd}(x_1) - \mathrm{rd}(x_2))}{x_1 - x_2}\right| = 0{,}44185.$$

Mit $\phi(x_1, x_2) = x_1 - x_2$ erhalten wir für die relativen Konditionszahlen die Werte

$$K_1 = \left|\frac{\partial\phi}{x_1} \cdot \frac{x_1}{\phi}\right| = \left|\frac{x_1}{x_1 - x_2}\right| = 811{,}67, \quad K_2 = \left|\frac{\partial\phi}{x_2} \cdot \frac{x_2}{\phi}\right| = \left|-\frac{x_2}{x_1 - x_2}\right| = 810{,}67,$$

also

$$K_1 \left|\frac{\Delta x_1}{x_1}\right| + K_2 \left|\frac{\Delta x_2}{x_2}\right| = 0{,}44139.$$

Die Subtraktion fast gleich großer Zahlen ist also sehr schlecht konditioniert. Im Gegensatz dazu gilt für die Addition

$$K_1 = 0{,}5003, \quad K_2 = 0{,}4997.$$

2) Betrachten wir nun noch einmal die Multiplikation $y = \phi(x_1, x_2) = x_1 \cdot x_2$. Dann gilt

$$K_1 = \frac{\partial\phi}{\partial x_1} \cdot \frac{x_1}{\phi} = \frac{x_1 x_2}{x_1 x_2} = 1,$$
$$K_2 = \frac{\partial\phi}{\partial x_2} \cdot \frac{x_2}{\phi} = \frac{x_2 x_1}{x_2 x_1} = 1.$$

Das heißt die Multiplikation ist gut konditioniert. Damit ist auch die Division gut konditioniert.

Wir betrachten nun noch ein weiteres Beispiel für ungenaue Berechnungen aufgrund von Auslöschung.

Beispiel. Wir betrachten das Polynom

$$\begin{aligned} p(x) &= (x-1)^7 \\ &= x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1. \end{aligned}$$

Abbildung 1.1 zeigt den Plot von p in einer kleinen Umgebung der 7-fachen Nullstelle $x = 1$. Während die Auswertung der Faktorisierung $p(x) = (x-1)^7$ einen stabilen Algorithmus liefert, ist der Algorithmus, der das Polynom in der Form

$$p(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$

auswertet, sehr instabil.

Listing 3: Berechnung eines Polynoms siebten Grades auf zwei verschiedene Weisen.

```
import matplotlib.pyplot as plt
```

```
import numpy as np

x = np.linspace(0.988,1.012,1000)
y= x**7 - 7*x**6 + 21*x**5 - 35*x**4 + 35*x**3 - 21*x**2 + 7*x -1

plt.plot(x,y,'b')
plt.plot(x,(x-1)**7,'k')
```

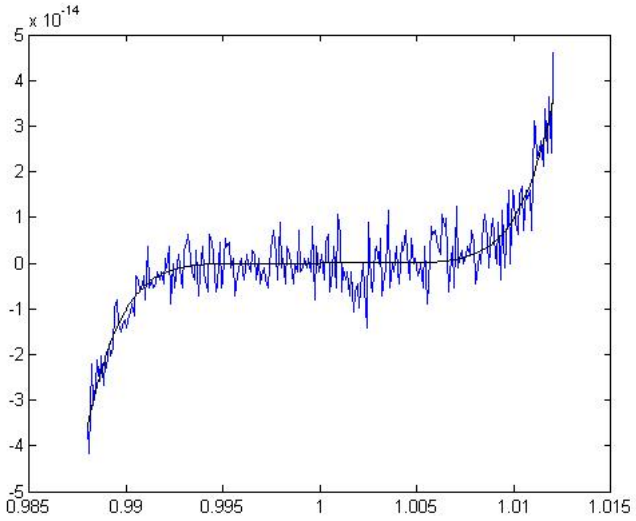


Abbildung 1.1: Plot eines Polynoms siebten Grades, das auf zwei verschiedene Weisen berechnet wurde.

1.7 Integral-Rekursion

Abschließend wollen wir uns noch ein weiteres Beispiel für einen stabilen und einen instabilen Algorithmus anschauen.

Die mathematische Aufgabe besteht in der Berechnung von

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \,\mathrm{d}x \quad \text{für} \quad n = 0, 1, \ldots, 30.$$

Partielle Integration liefert uns

$$\begin{aligned} I_n &= \frac{1}{e} \int_0^1 x^n e^x \, \mathrm{d}x \\ &= \frac{1}{e} \left[x^n e^x\right]_0^1 - \int_0^1 n x^{n-1} e^x \, \mathrm{d}x \\ &= \frac{1}{e} \left(e - n \int_0^1 x^{n-1} e^x \, \mathrm{d}x \right) \\ &= 1 - n I_{n-1}. \end{aligned}$$

Damit erhalten wir die Zwei-Term-Rekursionsformel

$$I_n = 1 - n I_{n-1}.$$

Als Startwert berechnen wir

$$I_0 = \frac{1}{e} \int_0^1 e^x \, \mathrm{d}x = \frac{1}{e}(e-1) \approx 0{,}63212\ldots$$

Ausgehend von I_0 lassen sich aus der Rekursionsformel beliebige Integrale I_n für $n > 0$ berechnen.

```
I(0)  =  6.3212e-001
I(5)  =  1.4553e-001
I(10) =  8.3877e-002
I(15) =  5.9034e-002
I(16) =  5.5459e-002
I(17) =  5.7192e-002
I(18) = -2.9454e-002
I(19) =  1.5596e+000
I(20) = -3.0192e+001
I(21) =  6.3504e+002
I(22) = -1.3970e+004
```

Einige Integrale der Integral-Rekursion. Ab I_{18} bekommt man völlig falsche Werte.

Wir wollen uns nun überlegen, ob die berechneten Werte korrekt sein können. Es gilt

$$I_n = \frac{1}{e} \int_0^1 x^n e^x \, \mathrm{d}x \geq 0, \quad \text{da} \quad x^n e^x > 0 \quad \text{für alle} \quad x \in (0,1).$$

Wegen $e^x < e$ für alle $x \in (0,1)$ können wir weiterhin

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \,\mathrm{d}x \leq \frac{1}{e} e \int_0^1 x^n \,\mathrm{d}x = \left[\frac{x^{n+1}}{n+1}\right]_0^1 = \frac{1}{n+1}$$

abschätzen und bekommen somit $0 \leq I_n \leq \frac{1}{n+1}$ für alle $n > 0$. Damit sehen wir, dass die Rekursionsformel ab I_{18} deutlich falsche Ergebnisse liefert.

Es sei nun I_n der exakte Wert und $\tilde{I}_n$ ein mit Rundungsfehler behafteter Wert. Weiterhin sei $\Delta I_0 = \tilde{I}_0 - I_0$ der Fehler im Startwert, etwa $|\Delta I_0| \approx 10^{-16}$. Dann gilt $I_n = 1 - nI_{n-1}$ und $\tilde{I}_n = 1 - n\tilde{I}_{n-1}$. Daraus folgt $\tilde{I}_n - I_n = -n\left(\tilde{I}_{n-1} - I_{n-1}\right)$, also $\Delta I_n = -n\Delta I_{n-1}$. Sukzessives Einsetzen liefert

$$\Delta I_n = -n\Delta I_{n-1} = -n(-(-n-1))\Delta I_{n-2} = \cdots = (-1)^n n!\Delta I_0$$

und wir erhalten die Fehlerformel $\Delta I_n = (-1)^n n!\Delta I_0$. Dies erklärt das alternierende Vorzeichen für großes n. Für $n = 18$ haben wir $18!10^{-16} = 0{,}64023$ und für $n = 19$ haben wir bereits $19!10^{-16} = 12{,}16$. Der Algorithmus ist also instabil.

Eine Alternative zur Berechnung der Integrale ist die Rückwärtsrekursion $I_{n-1} = \frac{1-I_n}{n}$. Das Problem hier: Der Startwert ist unbekannt. Wir können aber $I_n = \frac{1}{n+1}$ benutzen. Die Fehlerformel liefert $\Delta I_0 = \frac{1}{(-1)^n \cdot n!}\Delta I_n$. Auf diese Weise liefert Python den Wert $I_0 \approx 0{,}632120558828555$, wohingegen der „exakte“ Wert $\frac{e-1}{e} = 0{,}632120558828558$ ist. Der Algorithmus ist also stabil.

Listing 4: Vorwärts- und Rückwärtsrekursion für die Integral-Rekursion in Python.

```
# Integralrekursion, Teil 1

I = np.zeros(31)
I[0] = (np.exp(1)-1.)/np.exp(1)
for k in range(30):
    I[k+1] = 1. - (k+1)*I[k]
    print(k,I[k])

# Integralrekursion, Teil 2

I = np.zeros(31)
I[30] = 1./(31)

for k in range(29,-1,-1):
    I[k] = (1.-I[k+1])/(k+1.)
    print(k,I[k])
```

2 Interpolation

Häufig tritt in der numerischen Mathematik die Situation auf, dass statt einer Funktion $f : \mathbb{R} \to \mathbb{R}$ nur einige diskrete Funktionswerte $f(x_i)$ und eventuell noch Ableitungen $f^{(j)}(x_i)$ an endlich vielen Punkten x_i gegeben sind. Zum Beispiel, wenn f nur in Form von experimentellen Daten $f(x_0), \ldots, f(x_n)$ vorliegt und man an weiteren Funktionswerten zwischen den tabellierten Werten interessiert ist. Man möchte dann aus den gegebenen Daten eine Funktion φ konstruieren, die

$$\varphi^{(j)}(x_i) = f^{(j)}(x_i)$$

für alle i und j erfüllt und sich möglichst wenig von f unterscheidet. Letztere Eigenschaft muss natürlich noch genauer spezifiziert werden. Außerdem sollte φ leicht auswertbar sein und mit einem stabilen Algorithmus berechenbar sein.

In Abbildung 2.1 zeigen wir zwei interpolierende Funktionen, die durch sieben gegebene Datenpunkte (markiert durch blaue Kreise) verlaufen. Bei der roten Kurve handelt es sich um ein Interpolationspolynom. Die türkise Kurve zeigt einen interpolierenden Spline. Beide Arten der Interpolation werden wir in diesem Kapitel kennenlernen.

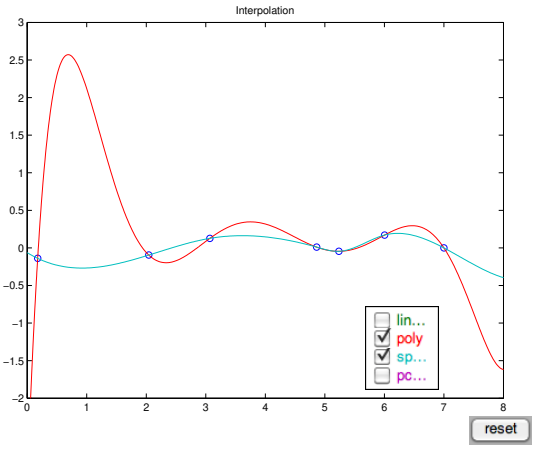


Abbildung 2.1: Zwei interpolierende Funktionen. Die rote Kurve zeigt ein interpolierendes Polynom, die türkise Kurve zeigt einen interpolierenden Spline. Die blauen Kreise markieren die gegebenen Datenpunkte.

Die Interpolation von gegebenen Daten gehört zu den wichtigsten Grundaufgaben der Numerik und ist gleichzeitig deren älteste Teilaufgabe. Bereits im 16. Jahrhundert wur-

den erste Interpolationsalgorithmen verwendet um Zwischenwerte von tabellarisch gegebenen Daten zu berechnen. Heutzutage benutzt man mehrdimensionale Splines um beispielsweise die Formen von Autokarosserien zu beschreiben.

Wir betrachten zunächst Interpolationsprobleme, bei denen zu gegebenen Daten eine stetige Funktion bestimmt werden soll, die durch diese Daten verläuft. In Abschnitt Hermite Interpolation betrachten wir dann Interpolationsprobleme, bei denen die interpolierende Funktion zusätzlich vorgegebene Werte der Ableitung erfüllt.

Definition 2.1. Unter einer Interpolationsaufgabe verstehen wir

- zu paarweise verschiedenen Punkten x_j und dazu gehörigen vorgegebenen Werten y_j, $j = 0, 1, \ldots, n$, die zu einer bekannten oder unbekannten Funktion $f(x)$ mit $f(x_j) = y_j$ gehören,
- und gegebenen Ansatzfunktionen $g_0(x), \ldots, g_n(x)$

Koeffizienten $c_0, c_1, \ldots, c_n$ zu finden, so dass

$$\varphi(x_j) = \sum_{k=0}^{n} c_k g_k(x_j) = y_j, \quad \text{für } j = 0, 1, \ldots, n$$

gilt.

Die $n + 1$ Interpolationsbedingungen lassen sich durch ein lineares Gleichungssystem der Form

$$\begin{pmatrix} g_0(x_0) & g_1(x_0) & \ldots & g_n(x_0) \\ g_0(x_1) & g_1(x_1) & \ldots & g_n(x_1) \\ \vdots & & & \vdots \\ g_0(x_n) & g_1(x_n) & \ldots & g_n(x_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

ausdrücken. Durch Lösen dieses Gleichungssystems erhalten wir die Koeffizienten $c_0, \ldots, c_n$ und somit die gesuchte Funktion

$$\varphi(x) = \sum_{k=0}^{n} c_k g_k(x),$$

die als Linearkombination der Ansatzfunktionen die Interpolationsbedingungen erfüllt.

Als einfache Ansatzfunktionen betrachtet man beispielsweise die Monome $g_k(x) = x^k$ oder trigonometrische Funktionen $g_k(x) = \cos(kx)$ oder $g_k(x) = \sin(kx)$ jeweils für $k = 0, \ldots, n$.

In der Computergraphik benutzt man radiale Basisfunktionen als Ansatzfunktionen für mehrdimensionale Rekonstruktionen.

2.1 Polynominterpolation

Wir betrachten zunächst die Interpolation durch Polynome. Unter einem Polynom vom Grad n verstehen wir eine Funktion der Form

$$p(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n$$

mit reellen Koeffizienten a_i, $i = 0, \ldots, n$. Der Koeffizient $a_n \neq 0$ heißt Hauptkoeffizient des Polynoms. Es sei

$$\mathbb{P}_n = \{p(x) = a_0 + a_1 x + \cdots + a_n x^n \mid a_i \in \mathbb{R}, i = 0, \ldots, n\}$$

die Menge der Polynome vom Grad kleiner gleich n. Die Menge $\mathbb{P}_n$ ist mit der üblichen Addition

$$p(x) + q(x) = \sum_{i=0}^{n} a_i x^i + \sum_{i=0}^{n} b_i x^i = \sum_{i=0}^{n} (a_i + b_i) x^i$$

und der Skalarmultiplikation

$$\alpha p(x) = \alpha \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} (\alpha a_i) x^i$$

für alle $p, q \in \mathbb{P}_n$ und alle $\alpha \in \mathbb{R}$ ein Vektorraum. Basis dieses Vektorraums sind die Monome $\{1, x \ldots, n^n\}$.

Definition 2.2. Die Polynom-Interpolationsaufgabe besteht darin, zu $n + 1$ paarweise verschiedenen Stützstellen (auch Knoten genannt) $x_0, \ldots, x_n \in \mathbb{R}$ und gegebenen Knotenwerten $y_0, \ldots, y_n \in \mathbb{R}$, ein Polynom $p \in \mathbb{P}_n$ mit der Eigenschaft

$$p(x_i) = y_i, \quad i = 0, \ldots, n \qquad \text{(Interpolationsbedingung)}$$

zu bestimmen.

Satz 2.3. *Die Polynom-Interpolationsaufgabe ist eindeutig lösbar.*

Beweis. Eindeutigkeit: Seien $p, q \in \mathbb{P}_n$ zwei interpolierende Polynome mit

$$p(x_i) = q(x_i), \quad i = 0, \ldots, n.$$

Dann ist $p - q \in \mathbb{P}_n$ ein Polynom mit den $n + 1$ Nullstellen $x_0, \ldots, x_n$, da alle Knoten paarweise verschieden sind. Aus dem Satz von Rolle folgt damit sofort, dass $p - q$ das Nullpolynom ist, also $p = q$.

Existenz: Wir wollen ein Polynom $p(x) = a_n x^n + \cdots + a_1 x + a_0$ konstruieren, das die Interpolationsaufgabe erfüllt. Die Koeffizienten erhalten wir als Lösung des linearen Gleichungssystems

$$\underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}}_{=:V_n} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Dieses lineare Gleichungssystem hat genau dann eine eindeutige Lösung, wenn $\det(V_n) \neq 0$. Man kann zeigen (siehe Übung), dass

$$\det(V_n) = \prod_{i=0}^{n} \prod_{j=i+1}^{n} (x_j - x_i),$$

also $\det(V_n) \neq 0$ genau dann, wenn die Knoten paarweise verschieden sind. □

Bemerkung. Für größere Gleichungssysteme ist die Lösung eines linearen Gleichungssystems mit Vandermonde-Matrix eine schlecht konditionierte mathematische Aufgabe. Daher ist dieser Ansatz zur Berechnung des Interpolationspolynoms im Allgemeinen ungeeignet.

Wir werden uns nun mit anderen Ansätzen zur Berechnung des (eindeutig bestimmten) Interpolationspolynoms beschäftigen.

2.1.1 Lagrangesche Form des Interpolationspolynoms

Es seien $x_0 < x_1 < \cdots < x_n$ paarweise verschiedene Knoten.

Satz 2.4 (Lagrange-Darstellung)**.** *Das (eindeutige) Interpolationspolynom*

$$p_n := p(f|x_0, \ldots, x_n) \in \mathbb{P}_n \quad \textit{mit} \quad p_n(x_j) = f(x_j), \quad j = 0, \ldots, n$$

lässt sich in der Form

$$p_n(x) = \sum_{j=0}^{n} f(x_j) l_{jn}(x) \quad \textit{mit} \quad l_{jn}(x) = \prod_{k=0, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}, \quad j = 0, \ldots, n$$

schreiben. Die l_{jn} heißen Lagrange-Fundamentalpolynome.

Beweis. Es gilt $l_{jn} \in \mathbb{P}_n$ für alle j und somit $p_n \in \mathbb{P}_n$. Weiterhin ist $l_{jn}(x_i) = \delta_{ij}$, wobei δ_{ij} das Kronecker-Symbol ist. Und schließlich folgt damit

$$p_n(x_i) = \sum_{j=0}^{n} f(x_j) l_{jn}(x_i) = f(x_i).$$

Also löst p_n die Interpolationsaufgabe. Die Eindeutigkeit folgt mit Satz 2.3. □

In Abbildung 2.2 (links) sind die Basispolynome zu den Knoten $0, 2, 4, 6, 8, 10$ dargestellt. In der rechten Abbildung sieht man die Basispolynome zu den Knoten $0, 2, 5, 10$.

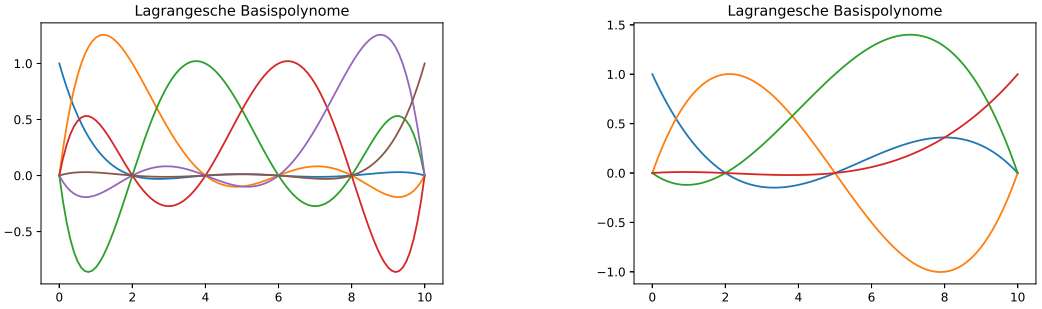


Abbildung 2.2: Lagrangesche Basispolynome zu den Knoten (links) $0, 2, 4, 6, 8, 10$ und (rechts) $0, 2, 5, 10$

Der Vorteil der Lagrangeschen Darstellung liegt in der Nützlichkeit für theoretische Betrachtungen. Der Nachteil dieser Darstellung hingegen ist, dass bei Hinzunahme einer neuen Stützstelle die gesamte Darstellung neu berechnet werden muss. Außerdem ist für Stützstellen $x_i \approx x_j$ diese Darstellung sehr ungenau. Dies führt uns zu einer alternativen Darstellung, der sogenannten Newton-Darstellung, die wir im folgenden Abschnitt herleiten.

2.1.2 Newtonsche Form des Interpolationspolynoms

Lemma 2.5. *Für die Lagrange-Interpolationspolynome*

$$p_{n-1} = p(f|x_0, \ldots, x_{n-1}) \in \mathbb{P}_{n-1} \quad \textit{und} \quad p_n = p(f|x_0, \ldots, x_n) \in \mathbb{P}_n$$

gilt

$$p_n(x) = p_{n-1}(x) + \delta_n w_n(x)$$

mit

$$w_n(x) = \prod_{i=0}^{n-1} (x - x_i) \in \mathbb{P}_n \quad \textit{und} \quad \delta_n = \frac{f(x_n) - p_{n-1}(x_n)}{w_n(x_n)}.$$

Hierbei nennt man $\{w_i \mid i = 0, \ldots, n\}$ *die* Newton-Basis *von* $\mathbb{P}_n$ *mit*

$$w_0 = 1, w_1 = (x - x_0), w_2 = (x - x_0)(x - x_1), \ldots$$

Beweis. Per Definition gilt $p_{n-1} \in \mathbb{P}_{n-1}$ und $w_n \in \mathbb{P}_n$. Damit folgt

$$q_n(x) = p_{n-1}(x) + \delta_n w_n(x) \in \mathbb{P}_n.$$

Es bleibt zu zeigen, dass q die Interpolationsbedingung an den Stützstellen $x_0, \ldots, x_n$ erfüllt. Für $i = 0, \ldots, n-1$ gilt

$$q_n(x_i) = p_{n-1}(x_i) + \delta_n w_n(x_i) = p_{n-1}(x_i) = f(x_i),$$

da $w_n(x_i) = 0$ nach Definition. Es sei nun $i = n$. Dann ist

$$\begin{aligned} q_n(x_n) &= p_{n-1}(x_n) + \delta_n w_n(x_n) \\ &= p_{n-1}(x_n) + \left(\frac{f(x_n) - p_{n-1}(x_n)}{w_n(x_n)} \right) w_n(x_n) \\ &= f(x_n). \end{aligned}$$

Falls $\delta_n = 0$ ist, so folgt bereits $p_{n-1}(x_n) = f(x_n)$ und Satz 2.3 liefert schließlich $q_n = p_n$ und damit die Eindeutigkeit des Interpolationspolynoms von f an den Stützstellen $x_0, \ldots, x_n$. □

Der Koeffizient δ_n hängt von f und den Stützstellen x_i ab. Daher hat sich auch die Schreibweise $\delta_n = [x_0, \ldots, x_n]f$ oder $f[x_0, \ldots, x_n]$ eingebürgert. Eine wiederholte Anwendung der Argumentation in Lemma 2.5 liefert uns den folgenden Satz.

Satz 2.6 (Newtonsche Darstellung der Interpolationspolynoms)**.** *Das Interpolationspolynom zu* f *und Knoten* $x_0, \ldots, x_n$ *ist gegeben durch*

$$p(f|x_0, \ldots, x_n) = \sum_{i=0}^{n} ([x_0, \ldots, x_i]f) w_i(x) \quad \textit{mit} \quad w_i(x) = \prod_{j=0}^{i-1} (x - x_j),$$

also

$$p(f|x_0, \ldots, x_n) = [x_0]f + [x_0, x_1]f w_1(x) + \cdots + [x_0, \ldots, x_n]f w_n(x).$$

Wir wollen nun ein Verfahren zur effizienten Berechnung der $[x_0, \ldots, x_i]f$, $i = 0, \ldots, n$ herleiten.

Lemma 2.7 (Aitken)**.** *Es gilt*

$$p(f|x_0, \ldots, x_n)(x) = \frac{x - x_0}{x_n - x_0} p(f|x_1, \ldots, x_n)(x) + \frac{x_n - x}{x_n - x_0} p(f|x_0, \ldots, x_{n-1})(x).$$

Die Interpolierende an $x_0, \ldots, x_n$ *ist also eine lineare Interpolation zwischen zwei Interpolationspolynomen auf* $x_1, \ldots, x_n$ *und* $x_0, \ldots, x_{n-1}$.

Beweis. Wir setzen x_i in die rechte Seite ein und schauen uns die Terme für verschiedene Werte von i getrennt an. Es sei also zunächst $0 < i < n$. Dann gilt:

$$\begin{aligned}&\frac{x_i - x_0}{x_n - x_0} p(f|x_1, \ldots, x_n)(x_i) + \frac{x_n - x_i}{x_n - x_0} p(f|x_0, \ldots, x_{n-1})(x_i) \\ &= \frac{x_i - x_0}{x_n - x_0} f(x_i) + \frac{x_n - x_i}{x_n - x_0} f(x_i) = f(x_i) = p(f|x_0, \ldots, x_n)(x_i).\end{aligned}$$

Für den Fall $i = 0$ fällt der erste Summand weg und wir erhalten

$$\frac{x_n - x_0}{x_n - x_0} p(f|x_0, \ldots, x_{n-1})(x_0) = p(f|x_0, \ldots, x_{n-1})(x_0) = f(x_0).$$

Analog fällt für $i = n$ der zweite Summand weg und es folgt

$$\frac{x_n - x_0}{x_n - x_0} p(f|x_1, \ldots, x_n)(x_0) = p(f|x_1, \ldots, x_n)(x_0) = f(x_0).$$

Also stimmen beide Seiten für alle $x_0, \ldots, x_n$ überein und sind eindeutiges Interpolationspolynom vom Grad kleiner oder gleich n. □

Bemerkung 2.8. i) Für paarweise verschiedene x_i gilt

$$[x_0, \ldots, x_n]f = \frac{[x_1, \ldots, x_n]f - [x_0, \ldots, x_{n-1}]f}{x_n - x_0}.$$

Diesen Ausdruck nennt man *dividierte Differenz* der Ordnung n von f.

ii) Die dividierte Differenz $[x_0, \ldots, x_n]f$ ist der führende Koeffizient a_n des Interpolationspolynoms

$$p(f|x_0, \ldots, x_n)(x) = a_n x^n + \cdots + a_1 x + a_0.$$

Beweis. Zu i): Nach Lemma 2.7 gilt

$$p(f|x_0, \ldots, x_n)(x) = \frac{x - x_0}{x_n - x_0} p(f|x_1, \ldots, x_n)(x) + \frac{x_n - x}{x_n - x_0} p(f|x_0, \ldots, x_{n-1})(x).$$

Setze die Newton-Darstellung für das Interpolationspolynom ein und führe Koeffizientenvergleich für den führenden Koeffizienten durch. Die Newton-Darstellung ist

$$p(f|x_0, \ldots, x_n)(x) = \sum_{i=0}^{n} ([x_0, \ldots, x_i]f) w_i(x) \quad \text{mit} \quad w_i(x) = \prod_{j=0}^{i-1} (x - x_j).$$

Koeffizientenvergleich liefert uns jetzt

$$[x_0, \ldots, x_n]f = \frac{1}{x_n - x_0}[x_1, \ldots, x_n]f - \frac{1}{x_n - x_0}[x_0, \ldots, x_{n-1}]f.$$

Zu ii): Folgt aus Lemma 2.5. □

Schema zur Berechnung der dividierten Differenzen für gegebene Startwerte $[x_i]f = f(x_i)$ für $i = 0, \ldots, n$:

$$
\begin{array}{llll}
x_0 & [x_0]f & & \\
 & & [x_0, x_1]f & \\
x_1 & [x_1]f & & [x_0, x_1, x_2]f \\
 & & [x_1, x_2]f & \\
x_2 & [x_2]f & & [x_1, x_2, x_3]f \\
 & & [x_2, x_3]f & \\
x_3 & [x_3]f & &
\end{array}
$$

Algorithmus 2.9 (Berechnung dividierter Differenzen)**.**

geg.: Knoten $x_0, \ldots, x_n$; Daten $f_0, \ldots, f_n$

ges.: $\Delta_{i,k} := [x_i, \ldots, x_{i+k}]f$, $k = 0, \ldots, n$, $i = 0, \ldots, n-k$

Schritt 1: Für $i = 0, \ldots, n$

setze $\Delta_{i,0} := f_i$

Schritt 2: Für $k = 1, \ldots, n$ und $i = 0, \ldots, n-k$

berechne $\Delta_{i,k} = (\Delta_{i+1,k-1} - \Delta_{i,k-1})/(x_{i+k} - x_i)$

Beachte: Zur effizienten algorithmischen Beschreibung wird ein doppelter Index benutzt.

Für eine effiziente Implementation der Newtonschen Form des Interpolationspolynoms nutzen wir die folgende Beziehung:

$$
\begin{aligned}
&P(f|x_0, \ldots, x_n)(x^*) = \Delta_{0,0} + \Delta_{0,1} w_1(x^*) + \ldots + \Delta_{0,n} w_n(x^*) \\
&= \Delta_{0,0} + (x^* - x_0)\,(\Delta_{0,1} + (x^* - x_1)\,(\Delta_{0,2} + (x^* - x_2)\,(\Delta_{0,3} + \ldots + (x^* - x_{n-1})\Delta_{0,n})\ldots))
\end{aligned}
$$

Algorithmus 2.10 (Auswertung der Newton-Interpolationsformel (Horner-Schema))**.**

geg.: $x_0, \ldots, x_n$; $\Delta_{0,0}, \ldots, \Delta_{0,n}$; Auswertungspunkt x^*

ges.: $y^* := P(f|x_0, \ldots, x_n)(x^*)$

Schritt 1: $y := \Delta_{0,n}$

Schritt 2: Für $k = n-1, \ldots, 0$

berechne $y := \Delta_{0,k} + (x^* - x_k)y$

Schritt 3: $y^* = y$

Aufwand: Es werden $n^2/2$ Divisionen zur Berechnung von $\Delta_{0,i}$ (wird einmal durchgeführt) und n Multiplikationen zur Berechnung von y^* (wird für jeden Auswertungspunkt x^* durchgeführt) benötigt.

Bemerkung 2.11. Für die dividierten Differenzen gilt:

(i) $[x_0, \ldots, x_n]f$ ist unabhängig von der Reihenfolge der x_i.

(ii) Ist $f \in C^n(I, \mathbb{R})$, $I = [a, b]$, $a \leq x_0 < x_1 < \cdots < x_n \leq b$, so gibt es ein $\eta \in I$ mit $[x_0, \ldots, x_n]f = \frac{1}{n!} f^{(n)}(\eta)$.

(iii) Ist $p \in \mathbb{P}_{n-1}$, so gilt $[x_0, \ldots, x_n]p = 0$

Beweis. Zu i): Folgt aus der Newton-Darstellung des Interpolationspolynoms.

Zu ii): Sei $g := f - p(f|x_0, \ldots, x_n) \in C^n(I, \mathbb{R})$. Dann hat g die $n+1$ Nullstellen $x_0, \ldots, x_n$. Aus dem Satz von Rolle folgt die Existenz eines $\eta \in I$ mit

$$0 = g^{(n)}(\eta) = f^{(n)}(\eta) - p(f|x_0, \ldots x_n)^{(n)}(\eta) = f^{(n)}(\eta) - n![x_0, \ldots, x_n]f.$$

Zu iii): Ist klar, denn der n-te Koeffizient eines Polynoms vom Grad kleiner oder gleich $n-1$ verschwindet. □

2.1.3 Verfahrensfehler der Polynominterpolation

Wir stellen uns die Frage, wie stark das Interpolationspolynom von der zu interpolierenden Funktion abweicht. In der Abbildung 2.3 zeigen wir Intepolationspolynome der Sinusfunktion mit 4-10 Knoten. Bei Vergrößerung der Anzahl der Knoten verbessert sich die Approximation an die Sinusfunktion. Kann man dieses Verhalten für beliebige Funktionen erwarten? Mit dieser Frage wollen wir uns in diesem Abschnitt beschäftigen.

Das Interpolationspolynom erkennt nicht, von welcher Funktion die Daten kommen. Für festes $\overline{x} \in [x_0, x_n]$ gilt aber immer

$$f(\overline{x}) = p(f|x_0, \ldots, x_n, \overline{x})(\overline{x}) \quad \text{mit} \quad p(f|x_0, \ldots, x_n, \overline{x}) \in \mathbb{P}_{n+1}.$$

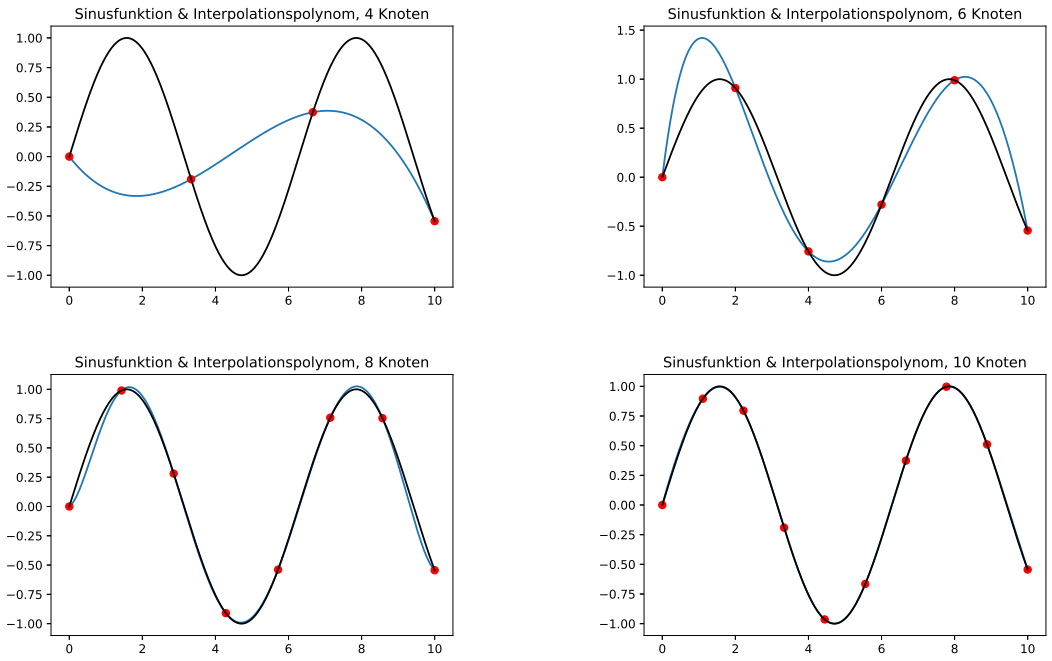


Abbildung 2.3: Interpolationspolynome der Sinusfunktion für 4, 6, 8 und 10 äquidistante Knoten auf dem Intervall $[0, 10]$.

Daraus folgt mit Lemma 2.5

$$\begin{aligned} f(x) - p(f|x_0, \ldots, x_n)(x) &= p(f|x_0, \ldots, x_n, x)(x) - p(f|x_0, \ldots, x_n)(x) \\ &= [x_0, \ldots, x_n, x]f \cdot w_{n+1}(x) \end{aligned}$$

mit

$$w_{n+1}(x) = \prod_{i=0}^{n}(x - x_i).$$

Wir wollen die linke Seite abschätzen. Nun gibt es nach Bemerkung 2.11 (ii) ein $\xi \in I = [a, b]$, $a \leq x_0$, $b \geq x_n$ mit

$$[x_0, \ldots, x_n, x]f = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

falls $f \in C^{n+1}(I, \mathbb{R})$. Damit erhalten wir die folgende Abschätzung für den Interpolationsfehler:

Satz 2.12. *Sei $f \in C^{n+1}([x_0, x_n], \mathbb{R})$. Dann gibt es zu jedem $x \in [x_0, x_n]$ ein $\xi \in [x_0, x_n]$, sodass*

$$f(x) - p(f|x_0, \ldots, x_n)(x) = w_{n+1}(x)\frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Insbesondere gilt:

$$\|f - p(f|x_0, \dots x_n)\|_\infty \leq \|w_{n+1}\|_\infty \frac{\|f^{(n+1)}\|_\infty}{(n+1)!},$$

wobei $\|f\|_\infty = \max\limits_{x \in [x_0, x_n]} |f(x)|$ *ist.*

Bemerkung. (i) Der Faktor $\|f^{(n+1)}\|_\infty$ hängt von f, aber nicht von den x_i ab,

(ii) und $w_{n+1} = \prod\limits_{j=0}^{n} (x - x_j)$ hängt von den Stützstellen ab, nicht jedoch von f.

Eine grobe Abschätzung erhält man durch $|w_{n+1}(x)| \leq (b-a)^{n+1}$ für alle $x \in [a, b]$. Hierbei geht jedoch der Einfluss der Wahl der Sütztstellen verloren.

Als Folgerung von Satz 2.12 erhalten wir die folgende Aussage.

Korollar 2.13. *Sei $f \in C^\infty[a, b]$ und es existiere ein $M > 0$, sodass für alle $n \in \mathbb{N}$*

$$|f^{(n)}(x)| \leq M \quad \forall x \in [a, b] \tag{2.1}$$

gilt. Dann konvergiert die Folge der Interpolationspolynome auf $[a, b]$ gleichmäßig gegen die Funktion f.

Beweis. Die Behauptung folgt aus

$$\|f - p_n\|_{\infty,[a,b]} \leq \frac{M}{(n+1)!}(b-a)^{n+1} \to 0,\ n \to \infty.$$

□

Die Bedingung (2.1) ist eine sehr starke Anforderung an die zu interpolierende Funktion. Sie ist z.B. für $\sin(x), \cos(x), e^x$ und für Polynome erfüllt. (Siehe auch noch einmal Abbildung 2.3.)

Für die einfache rationale Funktion $f(x) = 1/x$ mit $f^{(n)}(x) = (-1)^n n!/x^{n+1}$ gilt (2.1) etwa auf dem Intervall $[1, 2]$ schon nicht mehr. Durch Erhöhung der Anzahl der Stützstellen n kann man also nicht automatisch eine beliebig genaue Approximation an f bezüglich der $\|\cdot\|_\infty$-Norm erreichen. Runge hat hierfür ein Gegenbeispiel angegeben.

Betrachte

$$f(x) = \frac{1}{1+x^2} \in C^\infty, \quad x \in [-c, c], \quad c > 0$$

mit äquidistanten Stützstellen

$$x_i = -c + \frac{2c}{n} i, \quad i = 0, \dots, n.$$

Es gilt:

- $\|f - P_n\|_\infty \to \infty$ für $n \to \infty$, falls $c > \frac{e}{2}$,
- $\|f - P_n\|_\infty \to 0$ für $n \to \infty$, falls $c \leq \frac{e}{2}$.

Im divergenten Fall treten insbesondere an den Intervallrändern immer stärkere Oszillationen auf, siehe Abbildung 2.4. Dieses oszillatorische Verhalten in der Nähe des Randes bezeichnet man auch als *Runge-Phänomen*.

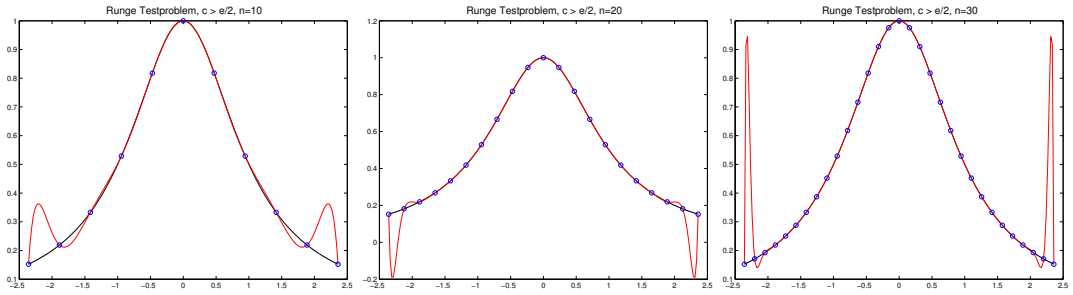


Abbildung 2.4: Resultate für Runges Testproblem mit $c > e/2$ und $n = 10, 20, 30$.

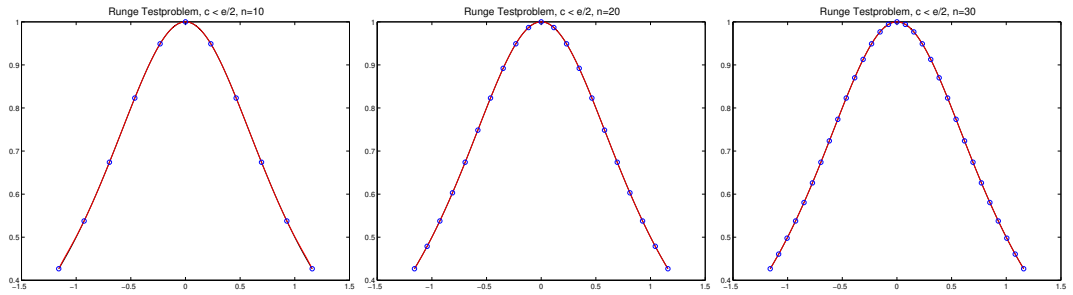


Abbildung 2.5: Resultate für Runges Testproblem mit $c < e/2$ und $n = 10, 20, 30$.

Bemerkung. Der Weierstraßsche Approximationssatz besagt, dass jede Funktion $f \in C([a, b])$ beliebig gut gleichmäßig auf $[a, b]$ durch Polynome approximiert werden kann. Die Vermutung, dass dies mit Interpolationspolynomen geschehen kann, ist jedoch im Allgemeinen falsch.

2.1.4 Tschebyscheff Knoten

Wir wollen uns nun mit der Frage beschäftigen, ob wir durch gute Wahl der Knoten den Interpolationsfehler verringern können. Die obere Abschätzung des Interpolationsfehlers (in Satz 2.12) wird minimiert, falls wir die Knoten so wählen, dass die Maximum-

norm des Knotenpolynoms

$$w_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$$

im betrachteten Intervall minimal ist. Wir betrachten zunächst die Interpolation auf dem Interval $[-1, 1]$.

Wir werden zeigen: Wählt man als Knoten für die Interpolationsaufgabe auf $[-1, 1]$ die Nullstellen der Tschebyscheff-Polynome, so wird $\|w_{n+1}\|_{\infty,[-1,1]}$ unter allen (normierten) Polynomen mit reellen Nullstellen minimal.

Die *Tschebyscheff-Polynome* sind eine spezielle Folge von Polynomen T_k, die bezüglich eines speziellen gewichteten Skalarprodukts auf $[-1, 1]$ orthogonal sind, das heißt

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} \cdot T_n(x) \cdot T_m(x)\, \mathrm{d}x = \begin{cases} 0, & m \neq n, \\ \pi, & n = m = 0, \\ \frac{\pi}{2} & n = m \neq 0. \end{cases}$$

Explizit lautet das k-te Polynom

$$T_k(x) = \cos(k \cdot \arccos(x)), \quad x \in [-1, 1].$$

Die Nullstellen von $T_k(x)$ sind gegeben durch

$$x_j = \cos\left(\frac{2j+1}{2k}\pi\right), \quad j = 0, \ldots, k-1.$$

Die Tschebyscheff-Polynome lassen sich auch durch eine Drei-Term-Rekursion berechnen und es gilt

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k \geq 2, \quad T_0(x) = 1, \quad T_1(x) = x.$$

In Abbildung 2.6 sind einige Tschebyscheff-Polynome dargestellt. Man erkennt, dass die Nullstellen am Rand des Intervalls dichter liegen als in der Mitte des Intervalls.

Die Nullstellen der Tschebyscheff-Polynome definieren Knoten für die Polynominterpolation, die die obere Abschätzung des Interpolationsfehlers gemäß Satz 2.12 minimieren.

Satz 2.14. *Es seien $t_0, \ldots t_n \in [-1, 1]$ die Nullstellen des Tschebyscheff-Polynoms T_{n+1}. Dann gilt*

$$\min_{x_0,\ldots,x_n \in [-1,1]} \max_{x \in [-1,1]} \prod_{j=0}^{n} |x - x_j| = \max_{x \in [-1,1]} \prod_{j=0}^{n} |x - t_j| = 2^{-n}.$$

Dabei sind $x_0, \ldots, x_n$ beliebige, paarweise verschiedene Knoten im Intervall $[-1, 1]$.

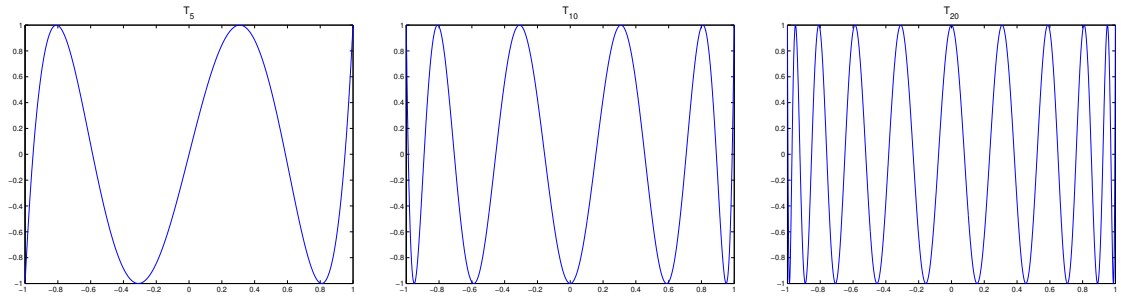


Abbildung 2.6: Die Tschebyscheff-Polynome T_5, T_{10} und T_{20}.

Einen Beweis dieses Satzes findet man im Lehrbuch von S. Bartels. Man nutzt dabei Eigenschaften der Tschebyscheff-Polynome, die wir in der Übung zeigen wollen.

In Abbildung 2.7 zeigen wir $|w_{n+1}(x)|$ für $x \in [-1, 1]$, $n = 10$ unter Verwendung von (links) äquidistante Knoten und (rechts) Tschebyscheff Knoten. Bei Verwendung der

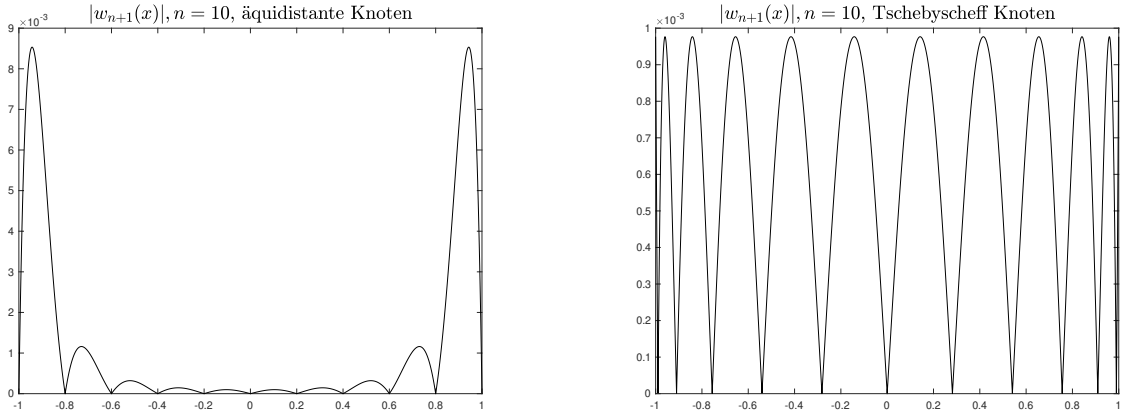


Abbildung 2.7: Das Stützstellenpolynom $|w_n(x)|$ für $x \in [-1, 1]$ und $n = 10$. (Links) unter Verwendung äquidistanter Knoten (rechts) für Tschebyscheff Knoten. Beachten Sie die unterschiedliche Skalierung der y-Achse.

Tschebyscheff Knoten nimmt das Stützstellenpolynom im betrachteten Intervall gleichmäßig kleine Werte an. Bei Verwendung äquidistanter Knoten nimmt das Stützstellenpolynom am Rand des Intervalls betragsmäßig größere Werte an. Dieses Verhalten ist für das Runge Phänomen verantwortlich. (Beachten Sie die unterschiedliche Skalierung der y-Achse.)

Aus den Sätzen 2.12 und 2.14 ergibt sich die folgende Fehlerabschätzung.

Korollar 2.15. *Sei $f \in C^{n+1}([-1, 1], \mathbb{R})$. Für den Interpolationsfehler der Polynominterpolation mit $n + 1$ Tschebyscheff Knoten $t_0, \ldots, t_n$ im Intervall $[-1, 1]$ gilt*

$$\|f - p(f|t_0, \ldots, t_n)\|_{\infty,[-1,1]} \leq 2^{-n} \frac{\|f^{(n+1)}\|_{\infty,[-1,1]}}{(n+1)!}.$$

Für allgemeine Intervalle $[a, b]$ erhält man die optimalen Stützstellen $x_0, \ldots, x_n$ mittels der affin-linearen Transformation

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} t_i, \quad i = 0, \ldots, n$$

aus den Tschebyscheff Knoten $t_0, \ldots, t_n$.

In Abbildung 2.8 zeigen wir Interpolationspolynome für das Runge-Problem bei Verwendung der Tschebyscheff-Knoten. Auch für (genügend kleine) $c > e/2$ liefert Polynominterpolation nun gute Approximationen.

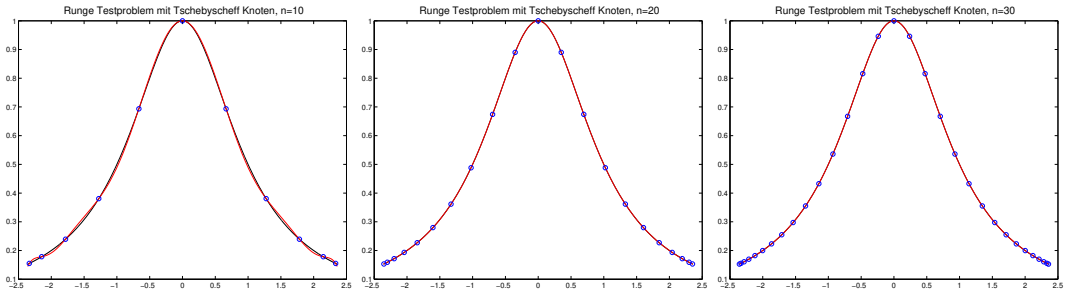


Abbildung 2.8: Resultate für Runges Testproblem mit Tschebyscheff Knoten, $c > e/2$ und $n = 10, 20, 30$.

2.1.5 Die baryzentrische Darstellung des Interpolationspolynoms

Zur Berechnung von Interpolationspolynomen mit Hilfe von Computerprogrammen benötigt man stabile Algorithmen. Es stellt sich heraus, dass typische Implementationen der Lagrangeschen oder Newtonschen Darstellung des Interpolationspolynoms für große Werte von n instabil sind. Dieser Abschnitt der Vorlesung basiert auf einem Fachartikel von J.-P. Berrut und L.N. Trefethen, den Sie über die HHU Bibliothek erhalten können[1]. Wir werden wesentliche Resultate kennenlernen aber nicht alle Aussagen beweisen.

Wir möchten zunächst veranschaulichen, dass die Newtonsche Darstellung des Interpolationspolynoms, so wie sie in den Algorithmen 2.9 und 2.10 beschrieben wurde, zu instabilen Resultaten führen kann. Dazu betrachten wir noch einmal Runges Testproblem, d.h. die Interpolation von

$$f(x) = \frac{1}{1+x^2}.$$

Analytisch kann man (wie bereits erwähnt) zeigen, dass das Interpolationspolynom auf dem Interval $[-c, c]$, $c < e/2$ für $n \to \infty$ gegen die Funktion f konvergiert. In Abbildung 2.5 zeigen wir Resultate für $n = 10$, $n = 20$ und $n = 30$, die eine sehr gute

[1] Berrut and Trefethen, Barycentric Lagrange Interpolation, SIAM Review, 46, p. 501-517, 2004

Übereinstimmung zwischen dem Interpolationspolynom und der Funktion belegen. In Abbildung 2.9 zeigen wir Resultate für größere Werte von n, insbesondere für $n = 66$, $n = 67$, $n = 70$. Wir sehen, dass der Algorithmus für größere n zu instabilen Resultaten führt. In Abbildung 2.10 zeigen wir Resultate für das Runge-Testproblem mit

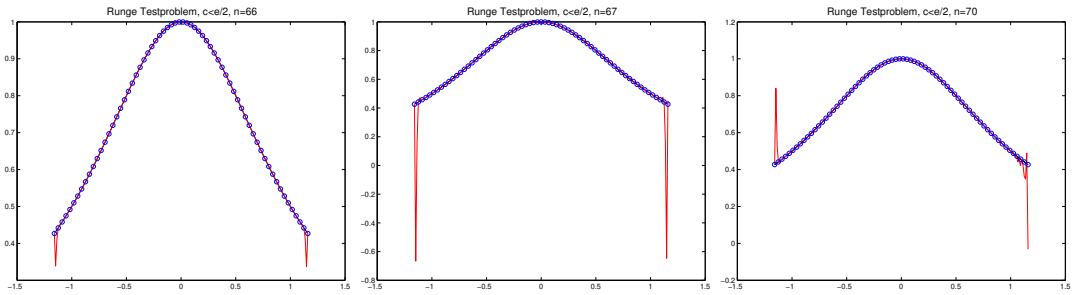


Abbildung 2.9: Resultate für Runges Testproblem mit $c < e/2$ und $n = 66, 67, 70$.

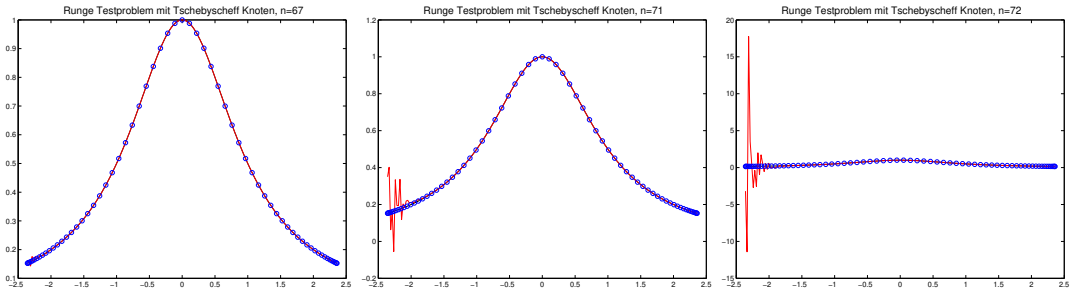


Abbildung 2.10: Resultate für Runges Testproblem mit Tschebyscheff Knoten, $c > e/2$ und $n = 67, 71, 72$.

Tschebyscheff-Knoten. Auch diese Rechnung wird für größere Werte von n instabil.

Diese Rechnungen wurden mit einem Programm durchgeführt, das direkt die Algorithmen 2.9 und 2.10 umsetzt. (In der 2. Programmieraufgabe wurden diese Algorithmen implementiert. Sie können also auch einmal Ihre Implementation der Algorithmen für größere Werte von n testen.)

Wir werden nun einen Algorithmus kennenlernen, der bei Verwendung von Tschebyscheff-Knoten auch für große n stabil ist.

Zunächst fassen wir noch einmal Vor- und Nachteile der Lagrangeschen und Newtonschen Darstellung des Interpolationspolynoms zusammen.

Lagrangesche Darstellung:

$$p(x) = \sum_{j=0}^{n} l_{jn}(x) f_j \quad \text{mit} \quad l_{jn}(x) = \prod_{k=0, k\neq j}^{n} \frac{(x - x_k)}{(x_j - x_k)}, \quad l_{jn}(x_k) = \delta_{jk}$$

Vor- und Nachteile:

+ Ideal, um etwas zu beweisen.
- Auswertung von $p(x)$ benötigt $\mathcal{O}(n^2)$ Rechenoperationen.
- Hinzufügen von (x_{n+1}, f_{n+1}) ändert alles.
- Numerisch instabil für größere Werte von n.

Newtonsche Darstellung:

$$p(x) = \sum_{j=0}^{n} [x_0, \ldots, x_j] f \prod_{i=0}^{j-1} (x - x_i)$$

mit

$$[x_j, \ldots, x_k] f = \frac{[x_{j+1}, \ldots, x_k] f - [x_j, \ldots, x_{k-1}] f}{x_k - x_j} \quad \text{und} \quad [x_j] f = f_j, \quad j = 0, \ldots, n$$

Vor- und Nachteile:

+ Auswertung von $p(x)$ benötigt $\mathcal{O}(n)$ Rechenoperationen.
+ Hinzufügen von (x_{n+1}, f_{n+1}) ist leicht umsetzbar.
- Rekursive Formel für $[x_0, \ldots, x_j] f$ benötigt $\mathcal{O}(n^2)$ Operationen.
- $[x_0, \ldots, x_j] f$ ist abhängig von der Funktion f.
- Numerisch instabil für größere Werte von n.

Die baryzentrische Darstellung des Interpolationspolynoms erlaubt (ähnlich wie die Newton-Darstellung) ein Hinzufügen zusätzlicher Knoten mit geringem zusätzlichen Rechenaufwand. Außerdem ist die baryzentrische Darstellung sehr robust gegenüber Rundungsfehlern und liefert auch für große n einen stabilen Algorithmus.

Modifizierte Lagrange-Darstellung

Es sei nun

$$l(x) := (x - x_0) \cdot \cdots \cdot (x - x_n) \quad \text{und} \quad \lambda_j := \prod_{k=0, k \neq j}^{n} \frac{1}{(x_j - x_k)} = \frac{1}{l'(x_j)}.$$

Jetzt können wir die Lagrange-Basispolynome darstellen durch

$$l_{jn}(x) = \prod_{k=0,k\neq j}^{n} \frac{(x-x_k)}{(x_j-x_k)} = l(x)\frac{\lambda_j}{x-x_j}$$

und erhalten für das Interpolatinspolynom

$$p(x) = l(x)\sum_{j=0}^{n} \frac{\lambda_j}{x-x_j} f_j.$$

Diese Darstellung hat die folgenden Eigenschaften:

- Einmalig $\mathcal{O}(n^2)$ Rechenoperationen zur Berechnung von λ_j (unabhängig von f_j).
- $\mathcal{O}(n)$ Rechenoperationen zur Berechnung von $p(x)$.
- Hinzufügen von (x_{n+1}, f_{n+1}):
  - Dividiere λ_j durch $x_j - x_{n+1} \to n+1$ Operationen.
  - Berechne $\lambda_{n+1} \to n+1$ Operationen.

Mit Hilfe der λ_j und $f(x) \equiv 1$ lässt sich dann $l(x)$ berechnen durch

$$1 = \sum_{j=0}^{n} l_{jn}(x) = l(x)\sum_{j=0}^{n} \frac{\lambda_j}{x-x_j},$$

also

$$l(x) = \frac{1}{\sum_{j=0}^{n} \frac{\lambda_j}{x-x_j}}.$$

Wir haben also gezeigt:

Satz 2.16 (Baryzentrische Interpolationsformel)**.** *Das Interpolationspolynom $p \in \mathbb{P}_n$ zu den $(n+1)$ Knoten $x_0, \ldots, x_n$ und den $(n+1)$ Daten $f_0, \ldots, f_n$ hat die Form*

$$p(x) = \begin{cases} \sum\limits_{j=0}^{n} \frac{\lambda_j f_j}{x-x_j} \Big/ \sum\limits_{j=0}^{n} \frac{\lambda_j}{x-x_j}, & x \neq x_j, \\ f_j, & x = x_j. \end{cases}$$

Die Stützkoeffizienten haben die Form

$$\lambda_j = \prod_{k=0,k\neq j}^{n} \frac{1}{x_j - x_k}.$$

Für verschiedene Knoten können wir die Stützkoeffizienten explizit angeben.

Baryzentrische Interpolation mit äquidistanten Knoten

Wir betrachten $n+1$ äquidistante Knoten auf $[-1,1]$ mit Abstand $h=2/n$ und erhalten zunächst

$$\lambda_j = (-1)^{n-j}\binom{n}{j}/(h^n n!).$$

Da sich in der Darstellung des Interpolationspolynoms (vergleiche mit Satz 2.16) Terme die unabhängig von j sind kürzen lassen, gibt man die Stützkoeffizienten direkt in der Form

$$\lambda_j = (-1)^j\binom{n}{j}$$

an.

Baryzentrische Interpolation in Tschebyscheff-Knoten

Die Tschebyscheff-Knoten 1. Art sind definiert durch die Nullstellen der Tschebyscheff-Polynome, d.h.

$$x_j = \cos\left(\frac{(2j+1)\pi}{2n+2}\right), \quad 0 \le j \le n.$$

Dann erhält man für die Gewichte, unter Vernachlässigung aller von j unabhängigen Bestandteile,

$$\lambda_j = (-1)^j \sin\left(\frac{(2j+1)\pi}{2n+2}\right).$$

Diese Beziehung wollen wir hier nicht herleiten.

In praktischen Rechnungen benutzt man meist die Tschebyscheff-Knoten 2. Art (Extremstellen der Tschebyscheff Polynome), die gegeben sind durch

$$x_j = \cos(j\pi/n), \quad 0 \le j \le n.$$

In diesem Fall erhält man die Gewichte (siehe Übung)

$$\lambda_j = (-1)^j\delta_j, \quad \text{mit} \quad \delta_j = \begin{cases} \frac{1}{2}, & j = 0, n, \\ 1, & \text{sonst.} \end{cases}$$

Das Interpolationspolynom hat dann die sehr einfache Form

$$p(x) = \sum_{j=0}^{n}{}' \frac{(-1)^j f_j}{x-x_j} \Big/ \sum_{j=0}^{n}{}' \frac{(-1)^j}{x-x_j}, \quad \text{und} \quad p(x_j) = f_j, \quad j = 0,\ldots,n.$$

Dabei bedeutet $\sum'$, dass der erste und letzte Summand jeweils mit $1/2$ multipliziert wird. Im Gegensatz zu den vorherigen Darstellungen des Interpolationspolynoms, liefert die baryzentrische Darstellung einen stabilen Algorithmus zur Berechnung des Interpolationspolynoms. In Abbildung 2.11 zeigen wir numerische Resultate für diesen robusten Algorithmus.

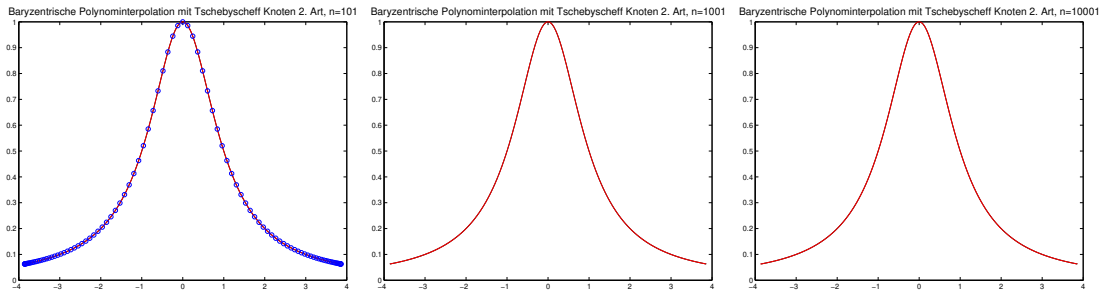


Abbildung 2.11: Resultate für Runges Testproblem mit Tschebyscheff Knoten 2. Art, $c > e/2$ und $n = 100, 1000, 10000$. In der 2. und 3. Abbildung zeigen wir nur das Interpolationspolynom sowie die zu interpolierende Funktion.

Die Berechnung von Interpolationspolynomen gehört zu den Problemstellungen in der Numerik, bei denen Rundungsfehler einen großen Einfluss auf die Genauigkeit der Lösung haben können. Wir werden in dieser Vorlesung keine Stabilitätsuntersuchungen für die Polynominterpolation durchführen. In einem Artikel von Higham[2] kann man solche Resultate finden. Für äquidistante Knoten ist die Berechnung des Interpolationspolynoms ein schlecht konditioniertes mathematisches Problem. Die Konditionszahl wächst exponentiell mit n. Für größere n sind Algorithmen zur Berechnung des Interpolationspolynoms mit äquidistanten Knoten daher instabil unabhängig von der Darstellungsform des Interpolationspolynoms. Dies gilt selbst in Fällen, für die man analytisch Konvergenz des Interpolationspolynoms gegen die zu interpolierende Funktion nachweisen kann. Bei Verwendung von Tschebyscheff Knoten konnte Higham zeigen, dass die baryzentrische Darstellung des Interpolationspolynoms einen stabilen Algorithmus liefert.

Wir wollen uns dieAussagen zur Kondition der Polynominterpolation hier nur anhand eines Beispiels veranschaulichen. Dazu betrachten wir die Berechnung des Interpolationspolynoms zu der Funktion $f(x) = 0$ auf dem Intervall $[-10, 10]$. Das exakte Interpolationspolynom $p_n \in \mathbb{P}_n$ ist durch $p_n(x) = 0$ gegeben. Wir nehmen nun an, dass n gerade ist und wir stören den Funktionswert an der Stelle $x = 0$ um den Wert $\varepsilon = 10^{-3}$. In Abbildung 2.12 zeigen wir die Interpolationspolynome für $n = 10$ und $n = 20$ bei Verwendung äquidistanter Knoten sowie bei Verwendung von Tschebyscheff Knoten erster und zweiter Art. Für äquidistante Stützstellen bewirkt die kleine Störung des

[2] N.J. Higham, The numerical stability of barycentric Lagrange interpolation, IMA Journal of Numerical Analysis, 24, pp. 547-556, 2004.

Funktionswertes eine große Änderung des Interpolationspolynoms. Das Problem ist also schlecht Konditioniert. Außerdem sieht man, dass sich die Kondition verschlechtert, wenn n vergrößert wird. Im Gegensatz dazu ist die Polynominterpolation mit Tschebyscheff Knoten erster und zweiterArt gut konditioniert.

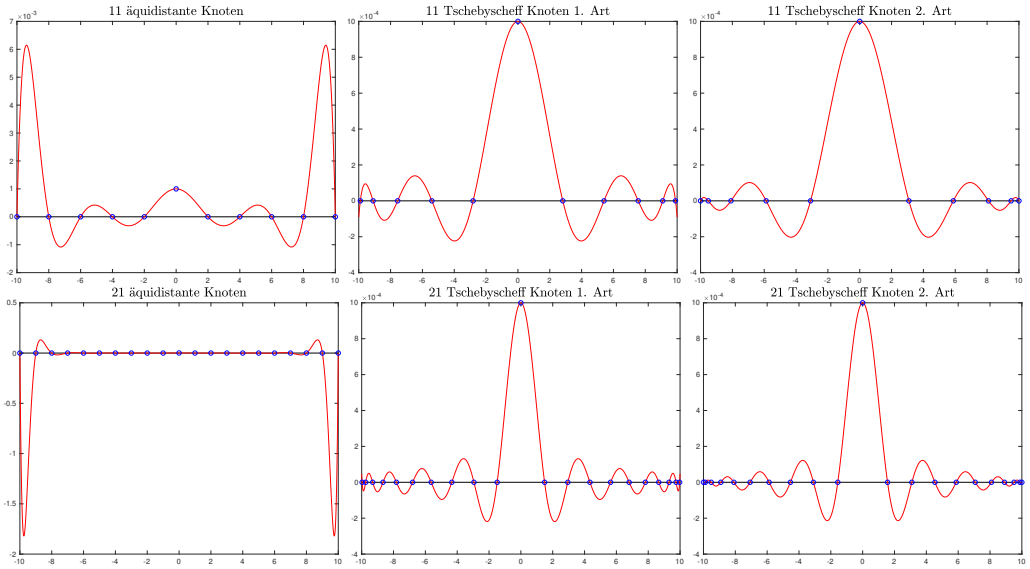


Abbildung 2.12: Testrechnung zur Illustration der Kondition der Polynominterpolationsaufgabe mit äquidistanten Knoten sowie mit Tschebyscheff Knoten erster und zweiter Art. Beachte die unterschiedliche Skalierung der y-Achse.

2.2 Hermite-Interpolation

Wir wollen nun Interpolationspolynome konstruieren, die an vorgegebenen Knoten mit gegebenen Funktionswerten und sowie Ableitungen übereinstimmen.

Definition 2.17. Die Hermite-Interpolationsaufgabe lautet:

gegeben:	Knoten	$x_0 < x_1 < \cdots < x_m,$	$m \geq 0,$
	Werte	$f_i^{(k)} \in \mathbb{R},$	$i = 0, \ldots, m, k = 0, \ldots, \mu_i,$
			$\mu_i \geq 0$ für $0 \leq i \leq m,$
gesucht:	$p \in \mathbb{P}_n, n = m + \sum_{i=0}^{m} \mu_i,$	mit $p^{(k)}(x_i) = f_i^{(k)},$	$i = 0, \ldots, m, k = 0, \ldots, \mu_i.$

Satz 2.18. *Die Hermite-Interpolationsaufgabe besitzt eine eindeutige Lösung.*

Beweis. Der Beweis wird analog zu dem von Satz 2.3 geführt, siehe z.B. Stoer, Bulirsch, Numerische Mathematik 1. □

Die Hermite-Interpolation ist eine Verallgemeinerung der klassischen Polynom-Interpolation, bei der man die Interpolation von Ableitungen formal als zusammenfallende Stützstellen interpretiert, das heißt wir betrachten Knoten $a \leq \tilde{x}_0 \leq \tilde{x}_1 \leq \cdots \leq \tilde{x}_n \leq b$. Sind in einem Knoten x_i die Funktionswerte $f(x_i)$ und die Ableitungen $f'(x_i), \ldots, f^{(\mu_i)}(x_i)$ bis zum Grad μ_i gegeben, so soll x_i genau $(\mu_i + 1)$-mal (in der Liste der $\tilde{x}$) auftreten. Sind alle Knoten paarweise verschieden, so erhalten wir die klassische Polynom-Interpolation. Stimmen alle Knoten überein, das heißt $\tilde{x}_0 = \tilde{x}_1 = \cdots = \tilde{x}_n = x_0$, so ist das Interpolationspolynom die abgebrochene Taylor-Reihe

$$p(f, \tilde{x}_0, \ldots, \tilde{x}_n)(x) = \sum_{j=0}^{n} \frac{(x - x_0)^j}{j!} f^{(j)}(x_0).$$

um x_0. Wir können das Interpolationspolynom wieder in der Newton-Darstellung

$$p = \sum_{i=0}^{n} [\tilde{x}_0, \ldots, \tilde{x}_i] f \cdot w_i, \quad w_i = \prod_{j=0}^{i-1} (x - \tilde{x}_j)$$

schreiben.

Die bisherige Definition der dividierten Differenzen können wir nicht ohne Weiteres verwenden, da Stützstellen nun doppelt auftreten können. Die Kombination von klassischer Interpolation und Taylor-Reihe ergibt

$$\begin{aligned}
[\tilde{x}_i, \ldots, \tilde{x}_{i+k}] f &= \frac{f^{(k)}(\tilde{x}_i)}{k!}, && \text{falls} \quad \tilde{x}_i = \tilde{x}_{i+k}, \\
[\tilde{x}_i, \ldots, \tilde{x}_{i+k}] f &= \frac{[\tilde{x}_{i+1}, \ldots, \tilde{x}_{i+k}] f - [\tilde{x}_i, \ldots, \tilde{x}_{i+k-1}] f}{\tilde{x}_{i+k} - \tilde{x}_i} && \text{sonst.}
\end{aligned}$$

Beispiel. Finde das Hermite-Interpolationspolynom $p \in \mathbb{P}_4$ mit

$$\begin{aligned}
&p(0) = f(0) = -1, && p'(0) = f'(0) = -2, && \\
&p(1) = f(1) = 0, && p'(1) = f'(1) = 10, && p''(1) = f''(1) = 40.
\end{aligned}$$

Wir verdoppeln den ersten und verdreifachen den zweiten Knoten und erhalten die dividierten Differenzen

$$\begin{array}{llllll}
\tilde{x}_0 = 0 & [\tilde{x}_0] f = -1 & & & & \\
 & & [\tilde{x}_0, \tilde{x}_1] f = -2 & & & \\
\tilde{x}_1 = 0 & [\tilde{x}_1] f = -1 & & [\tilde{x}_0, \tilde{x}_1, \tilde{x}_2] f = 3 & & \\
 & & [\tilde{x}_1, \tilde{x}_2] f = 1 & & [\tilde{x}_0, \ldots, \tilde{x}_3] f = 6 & \\
\tilde{x}_2 = 1 & [\tilde{x}_2] f = 0 & & [\tilde{x}_1, \tilde{x}_2, \tilde{x}_3] f = 9 & & [\tilde{x}_0, \ldots, \tilde{x}_4] f = 5 \\
 & & [\tilde{x}_2, \tilde{x}_3] f = 10 & & [\tilde{x}_1, \ldots, \tilde{x}_4] f = 11 & \\
\tilde{x}_3 = 1 & [\tilde{x}_3] f = 0 & & [\tilde{x}_2, \tilde{x}_3, \tilde{x}_4] f = 20 & & \\
 & & [\tilde{x}_3, \tilde{x}_4] f = 10 & & & \\
\tilde{x}_4 = 1 & [\tilde{x}_4] f = 0 & & & &
\end{array}$$

Daraus ergibt sich das Hermite-Interpolationspolynom als

$$\begin{aligned} p(f|\tilde{x}_0,\dots,\tilde{x}_4) &= -1 - 2(x-\tilde{x}_0) + 3(x-\tilde{x}_0)(x-\tilde{x}_1) \\ &\quad + 6(x-\tilde{x}_0)(x-\tilde{x}_1)(x-\tilde{x}_2) + 5(x-\tilde{x}_0)(x-\tilde{x}_1)(x-\tilde{x}_2)(x-\tilde{x}_3) \\ &= -1 - 2x + 3x^2 + 6x^2(x-1) + 5x^2(x-1)^2. \end{aligned}$$

Satz 2.19 (Fehler der Hermite-Interpolation)**.** *Ist $f \in C^{n+1}([x_0, x_m])$, so gibt es zu jedem $x \in [x_0, x_m]$ ein $\xi(x) \in [x_0, x_m]$, so dass für die Lösung $p \in \mathbb{P}_n$ der Hermite-Interpolationsaufgabe*

$$\begin{aligned} f(x) - p(x) &= [x_0,\dots,x_0,\dots,x_m,\dots x_m, x] f \prod_{i=0}^{m} (x-x_i)^{\mu_i+1} \\ &= \frac{1}{(n+1)!} f^{(n+1)}(\xi(x)) \prod_{i=0}^{m} (x-x_i)^{\mu_i+1} \end{aligned}$$

gilt.

Beweis. Der Beweis funktioniert wie in Satz 2.12. □

2.3 Spline-Interpolation

Wir haben gesehen, dass Polynominterpolation mit äquidistanten Stützstellen bei höherem Polynomgrad zu starken Oszillationen führen kann (vergl. mit Abb. 2.4). Möchte man, dass die interpolierende Kurve möglichst glatt durch die gegebenen Punkte (x_i, f_i) verläuft, dann kann man auf Teilintervallen Polynome niedrigeren Grades verwenden, und diese miteinander verheften. Dieser Ansatz wird bei der Spline-Interpolation verfolgt. Abbildung 2.1 zeigt zu gegebenen Daten sowohl ein interpolierendes Polynom als auch einen interpolierenden Spline.

Im Folgenden sei $I = [a, b]$ ein Intervall mit $a < b$. Weiter sei X eine Zerlegung von I, das heißt $X : a = x_0 < x_1 < \cdots < x_n = b$, $n \in \mathbb{N}$. Es sei $f_j \in \mathbb{R}$ für $j = 0, \dots, n$.

Definition 2.20. Es sei $k \in \mathbb{N}$. Dann heißt

$$S_k(X) = \left\{ s \in C^{k-1}(I), s|_{[x_{j-1},x_j]} \in \mathbb{P}_k, 1 \le j \le n \right\}$$

Raum der polynomialen Spline-Funktionen oder *Splines* vom Grad k auf der Zerlegung X.

Beispiel. (i) Für $k = 1$ erhält man stetige Polygonzüge. Die Interpolationsaufgabe lautet: Finde $s \in S_1(X)$ mit $s(x_j) = f_j$ für $j = 0, \dots, n$.

(ii) Für $k = 2$ erhält man quadratische Splines (wenig genutzt), also stückweise Polynome 2. Grades, die global einmal stetig differenzierbar sind. Beachte: Die Interpolationsaufgabe $s(x_j) = f_j$ für $j = 0, \ldots, n$ enthält eine Interpolationsbedingung zu wenig. Man kann zum Beispiel $s'(x_0) = f'(x_0)$ oder $s'(x_n) = f'(x_n)$ zusätzlich angeben.

(iii) Für $k = 3$ erhält man kubische Splines (häufig genutzt), also stückweise Polynome 3. Grades, die global zweimal stetig differenzierbar sind. Beachte: Hier enthält die Interpolationsaufgabe zwei Bedingungen zu wenig. Wähle zum Beispiel $s'(x_0) = f'(x_0)$ und $s'(x_n) = f'(x_n)$ als zusätzliche Interpolationsbedingungen.

Dimensionsbetrachtungen

Gesucht ist $s \in S_3(X)$ mit $s(x_j) = f_j$ für $j = 0, \ldots n$. Da wir Polynome 3. Grades betrachten, haben wir in jedem Teilintervall $[x_i, x_{i+1}]$ von I genau 4 Parameter. Dies liefert uns insgesamt $4n$ Parameter für das gesamte Intervall I. Durch die Stetigkeitsbedingungen $s^{(i)}(x_j - 0) = s^{(i)}(x_j + 0)$ für $i = 0, 1, 2$ und $j = 1, \ldots, n-1$ haben wir bereits $3(n-1)$ Bedingungen gegeben. Durch die Interpolationsbedingungen $s(x_j) = f_j$ für $j = 0, \ldots, n$ erhalten wir weitere $n + 1$ Bedingungen. Wir können also noch

$$4n - 3(n-1) - (n+1) = 2$$

weitere Bedingungen stellen.

2.3.1 Berechnung kubischer Splines

Woher kommt das Interesse an diesen Funktionen? Ingenieure benutzten früher Splines, das heißt „dünne Holzlatten“ bei der Konstruktion (zum Beispiel im Schiffsbau). Man zwang Splines durch bestimmte Knotenpunkte. Dadurch stellte sich für Rumpflinien von Schiffen eine günstige Kurve ein. Die Holzlatte nimmt eine Lage mit minimaler potentieller Energie an, das heißt

$$\int_a^b \frac{(y''(x))^2}{(1 + (y'(x))^2)^{3/2}} \, \mathrm{d}x$$

wird minimal. Falls $|y'(x)|$ klein ist, erhalten wir näherungsweise

$$\int_a^b (y''(x))^2 \, \mathrm{d}x$$

minimal. Außerhalb von $[a, b]$ verläuft die Latte linear, das heißt wegen $y \in C^2(\mathbb{R})$ gilt $y''(x) = 0$ in $(-\infty, a]$ und $[b, \infty)$. Insbesondere gilt $y''(x_0) = y''(x_n) = 0$ (natürliche Randbedingung). Daraus ergibt sich die

Aufgabenstellung 2.21. *Es seien die Zerlegung X und Daten $f_0, \dots, f_n$ gegeben und es sei*

$$A = \{f \in C^2(I) \mid f(x_j) = f_j,\ j = 0, \dots, n\}.$$

Gesucht ist $\tilde{f} \in A$ mit $E(\tilde{f}) \leq E(f)$ für alle $f \in A$. Dabei sei

$$E(f) := \int_a^b (f''(x))^2 \, \mathrm{d}x.$$

Satz 2.22. *Es existiert eine Lösung $\tilde{f} \in A$ von Problem 2.21. Für dieses $\tilde{f}$ gilt $\tilde{f} \in S_3(X)$ und $f''(x_0) = f''(x_n) = 0$. Diese Lösung ist eindeutig.*

Beweis. Zur Minimaltiät: Es seien $f, s \in A$. Dabei sei $s \in S_3(X)$ ein natürlicher kubischer Spline (das heißt ein kubischer Spline mit $s''(x_0) = s''(x_n) = 0$). Wir müssen zeigen, dass $E(s) \leq E(f)$ gilt. Es gilt

$$\begin{aligned}
0 \leq E(f - s) &= \int_a^b (f''(x) - s''(x))^2 \, \mathrm{d}x \\
&= \int_a^b (f''(x))^2 \, \mathrm{d}x - 2 \int_a^b (f''(x) s''(x)) \, \mathrm{d}x + \int_a^b (s''(x))^2 \, \mathrm{d}x \\
&= \int_a^b (f''(x))^2 \, \mathrm{d}x - 2 \int_a^b (f''(x) - s''(x)) s''(x) \, \mathrm{d}x - \int_a^b (s''(x))^2 \, \mathrm{d}x.
\end{aligned}$$

Wir zeigen nun, dass der zweite Term verschwindet, womit $E(s) \leq E(f)$ gezeigt wäre. Definiere $I_j = (x_{j-1}, x_j)$. Dann gilt

$$\int_a^b (f''(x) - s''(x)) s''(x) \, \mathrm{d}x = \sum_{j=1}^n \int_{I_j} (f''(x) - s''(x)) s''(x) \, \mathrm{d}x.$$

Wegen $s \in C^\infty(I_j)$ erhalten wir mit partieller Integration und der Tatsache, dass s''' konstant ist

$$\begin{aligned}
\int_{I_j} (f'' - s'') s'' \, \mathrm{d}x &= \left[(f' - s') s'' \right]_{x_{j-1}}^{x_j} - \int_{I_j} (f' - s') s''' \, \mathrm{d}x \\
&= \left[(f' - s') s'' \right]_{x_{j-1}}^{x_j} - s''' \big((f - s)(x_j) - (f - s)(x_{j-1}) \big) \\
&= \left[(f' - s') s'' \right]_{x_{j-1}}^{x_j},
\end{aligned}$$

da der zweite Term aufgrund der Interpolationsbedingungen verschwindet. Folglich erhalten wir

$$\int_a^b (f'' - s'')s'' \,\mathrm{d}x = \sum_{j=1}^{n} \left[(f' - s')s''\right]_{x_{j-1}}^{x_j} = \left[(f' - s')s''\right]_{x_0}^{x_n} = 0,$$

da s stetig auf I ist und wegen $s''(x_0) = s''(x_n) = 0$. Also gilt

$$0 \leq \int_a^b (f'' - s'')^2 \,\mathrm{d}x = \int_a^b (f'')^2 \,\mathrm{d}x - \int_a^b (s'')^2 \,\mathrm{d}x$$

für alle $f \in A$. Das heißt aber gerade, dass s die Lösung von Problem 2.21 ist, denn $E(s) \leq E(f)$. Wir haben also bisher gezeigt, dass wenn es eine Lösung von Problem 2.21 gibt, diese ein natürlicher kubischer Spline ist.

Zur Eindeutigkeit: Es seien $s, \tilde{s} \in S_3(X) \cap A$ Lösungen von Problem 2.21. Dann gilt: sowohl $E(s) \leq E(\tilde{s})$ (nach obiger Rechnung), als auch $E(\tilde{s}) \leq E(s)$, also $E(\tilde{s}) = E(s)$. Außerdem liefert die obige Rechnung

$$E(\tilde{s} - s) = \int_a^b (\tilde{s}'' - s'')^2 \,\mathrm{d}x = \int_a^b (\tilde{s}'')^2 \,\mathrm{d}x - \int_a^b (s'')^2 \,\mathrm{d}x = E(\tilde{s}) - E(s) = 0.$$

Also gilt $\tilde{s}''(x) - s''(x) = 0$ für alle $x \in [a, b]$. Mit Integration folgt nun $\tilde{s}(x) = s(x) + cx + d$ mit $c, d \in \mathbb{R}$. Wegen der Interpolationsbedingung sind s und $\tilde{s}$ mindestens an x_0 und x_n gleich. Daraus folgt wegen

$$\begin{aligned} \tilde{s}(x_0) &= s(x_0) + cx_0 + d, \\ \tilde{s}(x_n) &= s(x_n) + cx_n + d, \end{aligned}$$

dass $cx_0 + d = 0 = cx_n + d$, woraus wiederum $cx_0 = cx_n$ folgt und wegen $x_0 \neq x_n$ schließlich $c = 0$ und damit auch $d = 0$, also $s = \tilde{s}$.

Zur Existenz (konstruktiv): Wir wollen schließlich noch zeigen, dass es einen natürlichen kubischen Spline gibt. Im folgenden werden wir aber die Konstruktion eines solchen Splines etwas allgemeiner darstellen und auch andere Interpolationsprobleme lösen.

Es sei $s \in S_3(X)$ und s'' ist in jedem Teilintervall ein Polynom 1. Grades und global ein stetiger Polygonzug, also

$$s''(x)\Big|_{I_j} = a_j x + b_j, \quad a_j, b_j \in \mathbb{R}, \quad j = 1, \ldots, n.$$

Setze $M_j := s''(x_j)$ für $j = 0, \ldots, n$ und $h_j = x_j - x_{j-1}$ für $j = 1, \ldots, n$. Dann folgt

$$s''(x)\Big|_{I_j} = \frac{1}{h_j}\big(M_{j-1}(x_j - x) + M_j(x - x_{j-1})\big),$$

wegen der Stetigkeit von s''. Es gilt also

$$\begin{aligned} s'(x) &= \frac{1}{h_j}\left(M_j\frac{(x-x_{j-1})^2}{2} - M_{j-1}\frac{(x_j-x)^2}{2}\right) + c_j && \text{in } I_j, \\ s(x) &= \frac{1}{h_j}\left(M_j\frac{(x-x_{j-1})^3}{6} + M_{j-1}\frac{(x_j-x)^3}{6}\right) + c_j\,(x-x_{j-1}) + d_j && \text{in } I_j. \end{aligned}$$

Die Stetigkeit der ersten Ableitung in x_j für $j = 1, \ldots, n-1$ liefert $s'(x_j - 0) = s'(x_j + 0)$, woraus

$$\frac{h_j}{2}M_j + c_j = -M_j\frac{h_{j+1}}{2} + c_{j+1} \tag{$*$}$$

folgt und die Interpolationsbedingungen $s(x_j) = f(x_j)$ für $j = 0, \ldots, n$ ergeben

$$\begin{aligned} f_{j-1} &= \frac{h_j^2}{6}M_{j-1} + d_j && \text{in } I_j, \\ f_j &= \frac{h_j^2}{6}M_j + c_j h_j + d_j && \text{in } I_j \end{aligned}$$

und damit

$$\frac{f_j - f_{j-1}}{h_j} = \frac{h_j}{6}(M_j - M_{j-1}) + c_j. \tag{$**$}$$

Folglich gilt

$$c_j = [x_{j-1}, x_j]\,f - \frac{h_j}{6}(M_j - M_{j-1}), \quad d_j = f_{j-1} - \frac{h_j^2}{6}M_{j-1}, \quad j = 1, \ldots, n.$$

Setze nun c_j in $(*)$ ein und erhalte

$$h_j M_{j-1} + 2(h_j + h_{j+1})M_j + h_{j+1}M_{j+1} = 6\left([x_j, x_{j+1}]\,f - [x_{j-1}, x_j]\,f\right), \quad j = 1, \ldots, n-1.$$

Dies sind $n-1$ lineare Gleichungen für $n+1$ Unbekannte $M_0, \ldots, M_n$. Dieses Gleichungssystem wird in praktischen Rechnungen benutzt. Für theoretische Betrachtungen dividieren wir noch durch $h_j + h_{j+1} = x_{j+1} - x_{j-1}$ und erhalten

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = 6\,[x_{j-1}, x_j, x_{j+1}]\,f \tag{GS}$$

mit

$$\mu_j = \frac{h_j}{h_j + h_{j+1}}, \quad \lambda_j = \frac{h_{j+1}}{h_j + h_{j+1}}, \quad j = 1, \ldots, n-1,$$

wobei $\mu_j, \lambda_j > 0$ und $\mu_j + \lambda_j = 1$ gilt. □

Möglichkeiten zur Bestimmung der M_i

1. Fall: $M_0 = M_n = 0$ **(natürliche Splines)**

Das $(n-1)\times(n-1)$-Gleichungssystem mit den Unbekannten $M_1, \ldots, M_{n-1}$ ist gegeben durch

$$\begin{pmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ \vdots \\ M_{n-1} \end{pmatrix} = 6 \begin{pmatrix} [x_0, x_1, x_2]\, f \\ \vdots \\ [x_{n-2}, x_{n-1}, x_n]\, f \end{pmatrix}$$

2. Fall: Zusätzliche Interpolationsbedingungen (vollständige Randbedingungen)

Wir fordern zusätzlich $s'(x_0) = f'(x_0)$ und $s'(x_n) = f'(x_n)$. Für $j = 1$ folgt aus der Interpolationsbedingung $(**)$

$$[x_0, x_1]\, f = \frac{f_1 - f_0}{h_1} = \frac{h_1}{6}(M_1 - M_0) + c_1$$

und aus

$$s'(x) = \frac{1}{h_j}\left[M_j \frac{(x - x_j)^2}{2} - M_{j-1}\frac{(x_j - x)^2}{2}\right] + c_j \quad \text{in} \quad I_j$$

folgt für $x = x_0$ und $j = 1$ die Gleichung

$$s'(x_0) = f'(x_0) = [x_0, x_0]\, f = \frac{1}{h_1}\left[-M_0 \frac{h_1^2}{2}\right] + c_1 = -M_0 \frac{h_1}{2} + c_1.$$

Damit erhalten wir

$$[x_0, x_0, x_1]\, f = \frac{[x_0, x_1]\, f - f'(x_0)}{x_1 - x_0} = \frac{1}{6}(M_1 - M_0) + \frac{c_1}{h_1} + \frac{M_0}{2} - \frac{c_1}{h_1} = \frac{1}{6}(M_1 + 2M_0).$$

Eine analoge Gleichung wird aus der Bedingung $s'(x_n) = f'(x_n)$ hergeleitet. Zusätzlich zu (GS) erhalten wir somit

$$2M_0 + M_1 = 6\,[x_0, x_0, x_1]\, f,$$
$$M_{n-1} + 2M_n = 6\,[x_{n-1}, x_n, x_n]\, f,$$

also das $(n+1)\times(n+1)$-Gleichungssystem

$$\begin{pmatrix} 2 & 1 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & 1 & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ \vdots \\ M_n \end{pmatrix} = 6 \begin{pmatrix} \delta_0 \\ \vdots \\ \delta_n \end{pmatrix}$$

mit

$$\delta_0 = [x_0, x_0, x_1]\, f, \quad \delta_n = [x_{n-1}, x_n, x_n]\, f, \quad \delta_j = [x_{j-1}, x_j, x_{j+1}]\, f, \quad j = 1, \ldots, n-1.$$

3. Fall: Periodische Daten $f_n = f_0$

In diesem Fall setzen wir das Gitter mit der Periode $(b-a)$ periodisch auf $\mathbb{R}$ fort. Wir erhalten dadurch aus der Periodizität von $s \in C^2$ zwei zusätzliche Gleichungen $s'(x_0) = s'(x_n)$ und $s''(x_0) = s''(x_n)$. Es folgt $M_0 = M_n$ und damit verbleiben n Unbekannte $M_1, \ldots, M_{n-1}, M_n = M_0$. Folglich erhalten wir eine zusätzliche Bestimmungsgleichung

$$\mu_n M_{n-1} + 2M_n + \lambda_n M_{n+1} = 6\tilde{\delta}_n,$$

wobei $\tilde{\delta}_n = [x_{n-1}, x_n, x_{n+1}]\, f$ und $x_{n+1} = x_1 + (b-a)$, $f(x_{n+1}) = f(x_1)$. Das Gleichungssystem lautet dann:

$$\begin{pmatrix} 2 & \lambda_1 & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & \mu_n & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ \\ \vdots \\ \\ M_n \end{pmatrix} = 6 \begin{pmatrix} \delta_1 \\ \\ \vdots \\ \\ \tilde{\delta}_n \end{pmatrix}$$

mit δ_i wie in Fall 2.

4. Fall: not-a-knot-Spline

Als zusätzliche Bedingungen setzen wir $s'''(x_1 - 0) = s'''(x_1 + 0)$ und $s'''(x_{n-1} - 0) = s'''(x_{n-1} + 0)$, das heißt s ist auf $[x_0, x_2]$ und $[x_{n-2}, x_n]$ ein Polynom 3. Grades, x_1 und x_{n-1} sind also keine Knoten im eigentlichen Sinne. Es gilt

$$s'''(x) = \frac{1}{h_j} = \frac{1}{h_j}(M_j - M_{j-1})$$

und damit

$$s'''(x) = \frac{1}{h_1}(M_1 - M_0) \quad \text{in } I_1,$$
$$s'''(x) = \frac{1}{h_2}(M_2 - M_1) \quad \text{in } I_2.$$

Die erste zusätzliche Bedingung liefert uns

$$\frac{1}{h_1}(M_1 - M_0) = \frac{1}{h_2}(M_2 - M_1),$$

woraus wir

$$M_0\lambda_1 - M_1 + M_2\mu_1 = 0$$

erhalten. Analog ergibt sich für I_{n-1} und I_n mit der zweiten zusätzlichen Bedingung

$$M_{n-2}\lambda_{n-1} - M_{n-1} + M_n\mu_{n-1} = 0.$$

Insgesamt ist das Gleichungssystem dann durch

$$\begin{pmatrix} \lambda_1 & -1 & \mu_1 & & & & \\ \mu_1 & 2 & \lambda_1 & & & & \\ & \mu_2 & 2 & \lambda_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \mu_{n-2} & 2 & \lambda_{n-2} & \\ & & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & \lambda_{n-1} & -1 & \mu_{n-1} \end{pmatrix} \begin{pmatrix} M_0 \\ \\ \\ \vdots \\ \\ \\ M_n \end{pmatrix} = 6 \begin{pmatrix} 0 \\ \delta_1 \\ \\ \vdots \\ \\ \delta_{n-1} \\ 0 \end{pmatrix}$$

gegeben, wobei auch hier δ_i wie in den Fällen zuvor definiert ist.

Beispiel. Wir betrachten die Spline Interpolation der Funktion

$$f(x) = e^{-\sin 2\pi x}, \quad x \in [1, 2]$$

unter Verwendung von sechs äquidistanten Knoten und verschiedenen Randbedingungen. In Abbildung 2.13 zeigen wir die Splines mit natürlichen, vollständigen (d.h., $f'(x_0) = f'(x_n) = 0$), periodischen sowie not-a-knot Randbedingungen.

Nun ergibt sich die Frage, ob diese linearen Gleichungssysteme lösbar sind.

Definition 2.23. Eine Matrix $A \in \mathbb{R}^{n\times n}$ (oder $A \in \mathbb{C}^{n\times n}$) heißt *(strikt) diagonaldominant*, falls

$$|a_{ii}| > \sum_{k=1,k\neq i}^{n} |a_{ik}|, \quad i = 1, \ldots, n.$$

Sie heißt *schwach diagonaldominant*, wenn stattdessen

$$|a_{ii}| \geq \sum_{k=1,k\neq i}^{n} |a_{ik}|, \quad i = 1, \ldots, n$$

und für mindestens einen Index die strikte Ungleichung gilt.

Lemma 2.24. *Es sei $A \in \mathbb{R}^{n\times n}$ (oder $A \in \mathbb{C}^{n\times n}$) eine strikt diagonaldominante Matrix. Dann ist A invertierbar und es gilt:*

$$\|Az\|_\infty \geq c\|z\|_\infty \quad \textit{für alle} \quad z \in \mathbb{R}^n \quad (\textit{oder } z \in \mathbb{C}^n)$$

mit

$$c := \min_{1\leq i\leq n} \left(|a_{ii}| - \sum_{j\neq i}^{n} |a_{ij}| \right).$$

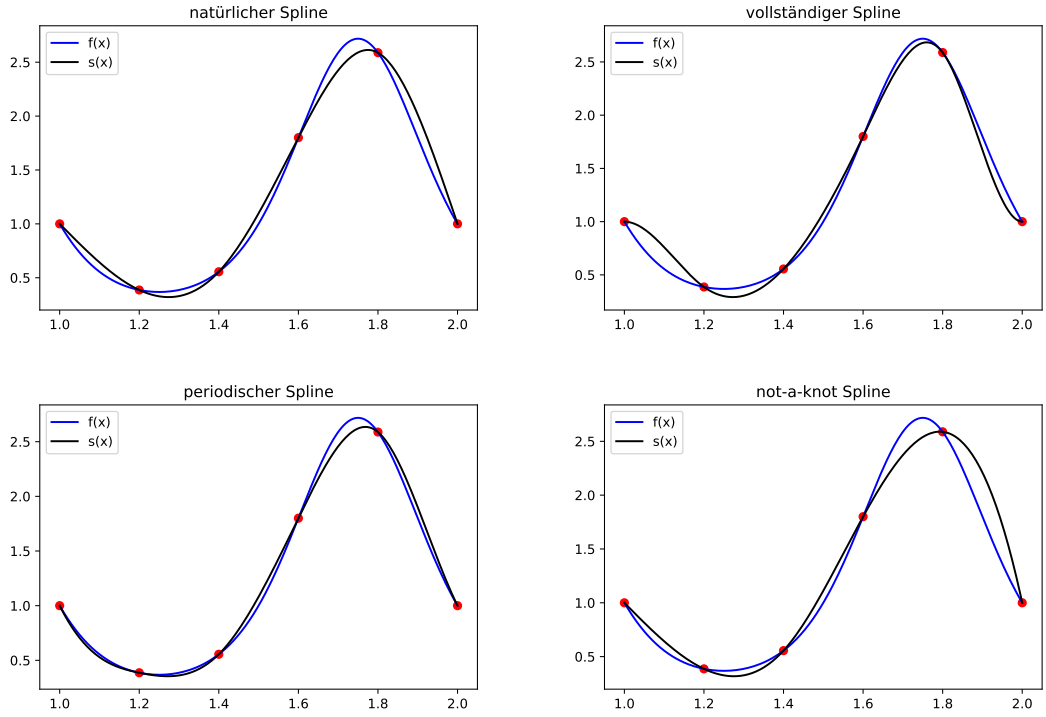


Abbildung 2.13: Spline-Interpolationen mit verschiedenen Randbedingungen
Beim vollst. Spline muss noch $s'(a) = 0$ zu $s'(a) = f'(a)$ korrigiert werden, dito bei b.

Beweis. Zum Beweis benutzen wir die aus der linearen Algebra bekannte Aussage A ist invertierbar genau dann, wenn $Ax = 0$ nur die triviale Lösung besitzt.

Es seien $x \in \mathbb{R}^n \backslash \{0\}$ und $b = Ax$. Weiter sei $k \in \{1, \dots, n\}$ so gewählt, dass $\|x\|_\infty = |x_k|$. Dann gilt

$$\begin{aligned}
\|b\|_\infty = \|Ax\|_\infty &= \max_{1 \le i \le n} \left| \sum_{j=1}^{n} a_{ij} x_j \right| \\
&\ge \left| \sum_{j=1}^{n} a_{kj} x_j \right| = \left| a_{kk} \frac{x_k}{|x_k|} + \sum_{j \ne k}^{n} a_{kj} \frac{x_j}{|x_k|} \right| \|x\|_\infty \\
&\ge \left(|a_{kk}| - \sum_{j \ne k}^{n} |a_{kj}| \right) \|x\|_\infty \\
&\ge \min_{1 \le i \le n} \left(|a_{ii}| - \sum_{j \ne i}^{n} |a_{ij}| \right) \|x\|_\infty = c \|x\|_\infty > 0.
\end{aligned}$$

Also folgt aus $x \neq 0$, dass $Ax = b \neq 0$ und damit ist A invertierbar. Außerdem gilt

$\|Ax\|_\infty \geq c\|x\|_\infty$ für alle $x \in \mathbb{R}^n$. Der Beweis funktioniert für $x \in \mathbb{C}^n$ analog. □

Bemerkung. Mit Lemma 2.24 folgt die Lösbarkeit für die Spline-Interpolation mit natürlichen, vollständigen und periodischen Randbedingungen.

Lemma 2.25. *Das im 4. Fall auftretende („not-a-knot") lineare Gleichungssystem ist eindeutig lösbar.*

Beweis. Wir wollen zeigen, dass $Ax = 0$ nur die triviale Lösung besitzt. Dazu formen wir die Matrix A aus dem 4. Fall zunächst so um, dass wir die Matrix

$$\tilde{A} = \begin{pmatrix} 1 & 1 & 1 & & & & \\ 0 & \mu_1 - 2 & \mu_1 - \lambda_1 & & & & \\ & \mu_2 & 2 & \lambda_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \mu_{n-2} & 2 & \lambda_{n-2} & \\ & & & & \lambda_{n-1} - \mu_{n-1} & \lambda_{n-1} - 2 & 0 \\ & & & & 1 & 1 & 1 \end{pmatrix}$$

erhalten. Bezeichne A_i die i-Zeile von A. Dann gilt nämlich $\tilde{A}_1 = A_1 + A_2$ und unter Verwendung von $\mu_1 + \lambda_1 = 1$ gilt $\tilde{A}_2 = \mu_1 A_1 - \lambda_1 A_2$. Analog erhält man die letzten beiden Zeilen von $\tilde{A}$. Wir betrachten nun $\tilde{A}x = 0$. Dann folgt aus

$$\left(\begin{array}{c|ccccc|c} 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ \hline 0 & & & & & & 0 \\ \vdots & & & A_{22} & & & \vdots \\ 0 & & & & & & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 & 1 & 1 \end{array}\right) \begin{pmatrix} x_1 \\ \vec{x_2} \\ x_3 \end{pmatrix} = 0$$

mit Lemma 2.24 zunächst $\vec{x}_2 = 0$, da A_{22} strikt diagonaldominant ist. Weiterhin sehen wir dann aber anhand der ersten und letzten Zeile, dass gerade $x_1 + 0 = 0$ und $x_3 + 0 = 0$ ist. Also hat $Ax = 0$ nur die triviale Lösung und A ist invertierbar. □

Im folgenden Satz fassen wir nochmal alle betrachteten Interpolationsprobleme zusammen.

Satz 2.26. *Die Interpolationsprobleme mit kubischen*

1) natürlichen Splines

$$\begin{aligned} &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &M_0 = M_n = 0 \quad (\textit{oder } M_0, M_n \textit{ beliebig}), \end{aligned}$$

2) vollständigen Splines

$$\begin{aligned} &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &s'(x_0) = f'(x_0), \\ &s'(x_n) = f'(x_n), \end{aligned}$$

3) periodischen Splines

$$\begin{aligned} &f(x_0) = f(x_n), \\ &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &s'(x_0) = s'(x_n), \\ &s''(x_0) = s''(x_n), \end{aligned}$$

4) not-a-knot-Splines

$$\begin{aligned} &s(x_j) = f(x_j), \quad j = 0, \ldots, n, \\ &s'''(x_1 - 0) = s'''(x_1 + 0), \\ &s'''(x_{n-1} - 0) = s'''(x_{n-1} + 0) \end{aligned}$$

sind stets eindeutig lösbar.

Beweis. Die entsprechenden lineare Gleichungssysteme sind nach Lemma 2.24 und Lemma 2.25 eindeutig lösbar. Folglich existiert s'' eindeutig. Also existiert ein s, dass die Interpolationsaufgabe löst. Ist $\tilde{s} \in S_3(X)$ eine weitere Lösung, so folgt $s - \tilde{s} = cx + d$, da $s'' = \tilde{s}''$. Aus der Interpolationsbedingung für $j = 0$ und $j = n$ folgt dann aber $s = \tilde{s}$. □

2.3.2 Konvergenz kubischer Splines

Es seien $X : a = x_0 < \cdots < x_n = b$ eine Zerlegung von $I = [a, b]$ sowie

$$\begin{aligned} h &:= \max_{j=1,\ldots,n} (x_j - x_{j-1}) && \text{der maximale Gitterabstand und} \\ h_{min} &:= \min_{j=1,\ldots,n} (x_j - x_{j-1}) && \text{der minimale Gitterabstand.} \end{aligned}$$

Wir sind nun am Verhalten des Fehlers $f - s$ für $n \to \infty$ interessiert.

Satz 2.27. *Es sei $f \in C^4([a, b])$ mit $f''(a) = f''(b) = 0$. Es sei weiter s der kubische natürliche Spline, der f auf der Zerlegung X interpoliert. Dann gilt die Fehlerabschätzung*

$$\|f - s\|_{\infty,[a,b]} \leq h^4 \|f^{(4)}\|_{\infty,[a,b]}.$$

Beweis. Sei $0 \leq k \leq n-1$ und $p(f-s|x_k, x_{k+1})(x) \in \mathbb{P}_1$ das lineare Polynom, das $f-s$ an den Knoten x_k und x_{k+1} interpoliert. Wegen

$$0 = p(f-s|x_k, x_{k+1})(x_k) = p(f-s|x_k, x_{k+1})(x_{k+1})$$

gilt dann aber $p(f-s|x_k, x_{k+1})(x) \equiv 0$. Somit liefert Satz 2.12 auf $[x_k, x_{k+1}]$ den Fehler

$$\begin{aligned}\|(f-s) - p(f-s|x_k, x_{k+1})\|_\infty &= \|f-s\|_\infty \\ &\leq \|(x-x_k)(x-x_{k+1})\|_\infty \frac{1}{2!}\|f''-s''\|_\infty \\ &\leq \frac{1}{2}h_{k+1}^2\|f''-s''\|_\infty. \end{aligned} \tag{1}$$

Wir wollen $\|f''-s''\|_{\infty,[x_k,x_{k+1}]}$ weiter abschätzen. Dazu sei $p_k = p(f''|x_k, x_{k+1})(x) \in \mathbb{P}_1$ das lineare Polynom, das f'' an den Knoten x_k und x_{k+1} interpoliert. Dann gilt unter erneuter Anwendung von Satz 2.12 und der Tatsache, dass p_k und s'' linear sind, die Abschätzung

$$\begin{aligned}\|f''-s''\|_{\infty,[x_k,x_{k+1}]} &\leq \|f''-p_k\|_{\infty,[x_k,x_{k+1}]} + \|p_k-s''\|_{\infty,[x_k,x_{k+1}]} \\ &\leq \frac{1}{2}h_{k+1}^2\|f^{(4)}\|_{\infty,[x_k,x_{k+1}]} + \max_{l=k,k+1}|f''(x_l)-s''(x_l)| \end{aligned} \tag{2}$$

Wir setzen nun $M_l = s''(x_l)$. Dann bleibt noch $\max\limits_{l=k,k+1}|f''(x_l)-M_l|$ abzuschätzen. Es sei

$$A = \begin{pmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{pmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}, \quad z \in \mathbb{R}^{n-1}.$$

Dann gilt nach Lemma 2.24 (mit $c := 2-\mu-\lambda = 1$) gerade

$$\|z\|_\infty \leq \|Az\|_\infty.$$

Dies wenden wir nun auf $f''(x_l) - M_l$ an. Mit $f''(a) = f''(b) = 0$ gilt

$$\begin{aligned}\max_{0\leq l\leq n}|f''(x_l)-M_l| &= \max_{1\leq l\leq n-1}|f''(x_l)-M_l| \\ &\leq \max_{1\leq l\leq n-1}|\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) - \mu_l M_{l-1} - 2M_l - \lambda_l M_{l+1}| \\ &= \max_{1\leq l\leq n-1}|\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) - 6[x_{l-1}, x_l, x_{l+1}]f|. \end{aligned}$$

Bei der letzten Gleichung wurde die Darstellung der rechten Seite des linearen Gleichungssystems zur Berechnung natürlicher Splines genutzt. Wir berechnen die vorkommenen dividierten Differenzen und betrachten im Folgenden den Ausdruck

$$\left|\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1}) - \frac{6}{h_l+h_{l+1}}\left(\frac{f(x_{l+1})-f(x_l)}{h_{l+1}} - \frac{f(x_l)-f(x_{l-1})}{h_l}\right)\right|. \tag{$*$}$$

Wir führen nun Taylor-Entwicklung um x_l durch und erhalten die Gleichungen

$$
\begin{aligned}
f''(x_{l-1}) &= f''(x_l) + f'''(x_l)(-h_l) + \frac{1}{2}f^{(4)}(x_l - \Theta_1 h_l)h_l^2,\\
f''(x_{l+1}) &= f''(x_l) + f'''(x_l)h_{l+1} + \frac{1}{2}f^{(4)}(x_l + \Theta_2 h_{l+1})h_{l+1}^2,\\
f(x_{l-1}) &= f(x_l) + f'(x_l)(-h_l) + \frac{1}{2}f''(x_l)h_l^2 - \frac{1}{6}f'''(x_l)h_l^3 + \frac{1}{24}f^{(4)}(x_l - \Theta_3 h_l)h_l^4,\\
f(x_{l+1}) &= f(x_l) + f'(x_l)h_{l+1} + \frac{1}{2}f''(x_l)h_{l+1}^2 + \frac{1}{6}f'''(x_l)h_{l+1}^3 + \frac{1}{24}f^{(4)}(x_l + \Theta_4 h_{l+1})h_{l+1}^4.
\end{aligned}
$$

Weiterhin erhalten wir dann mit $\mu_l = \frac{h_l}{h_l+h_{l+1}}$ und $\lambda_l = \frac{h_{l+1}}{h_l+h_{l+1}}$ die beiden Gleichungen

$$
\begin{aligned}
&\mu_l f''(x_{l-1}) + 2f''(x_l) + \lambda_l f''(x_{l+1})\\
&= 3f''(x_l) + (-\mu_l h_l + \lambda_l h_{l+1})f'''(x_l) + \frac{\mu_l h_l^2}{2}f^{(4)}(x_l - \Theta_1 h_l) + \frac{\lambda_l h_{l+1}^2}{2}f^{(4)}(x_l + \Theta_2 h_{l+1})
\end{aligned}
$$

sowie

$$
\begin{aligned}
&-\frac{6}{h_l + h_{l+1}}\left(\frac{f(x_{l+1}) - f(x_l)}{h_{l+1}} - \frac{f(x_l) - f(x_{l-1})}{h_l}\right)\\
&= -3f''(x_l) - (-\mu_l h_l + \lambda_l h_{l+1})f'''(x_l)\\
&\quad - \frac{1}{h_l + h_{l+1}}\left(\frac{h_{l+1}^3}{4}f^{(4)}(x_l + \Theta_4 h_{l+1}) + \frac{h_l^3}{4}f^{(4)}(x_l - \Theta_3 h_l)\right).
\end{aligned}
$$

Addieren wir die letzten beiden Gleichungen und setzen diese in $(*)$ ein, so erhalten wir die Abschätzung

$$
\begin{aligned}
(*) &= \left|\frac{1}{h_l + h_{l+1}}\left(\frac{h_l^3}{2}f^{(4)}(x_l - \Theta_1 h_l) + \frac{h_{l+1}^3}{2}f^{(4)}(x_l + \Theta_2 h_{l+1})\right.\right.\\
&\qquad \left.\left. -\frac{h_{l+1}^3}{4}f^{(4)}(x_l + \Theta_4 h_{l+1}) + \frac{h_l^3}{4}f^{(4)}(x_l - \Theta_3 h_l)\right)\right|\\
&\leq \frac{3}{2}h^2\|f^{(4)}\|_{\infty,[x_{l-1},x_{l+1}]}
\end{aligned}
$$

und damit insgesamt

$$
\max_{l=k,k+1}|f''(x_l) - s''(x_l)| \leq \max_{1\leq l\leq n-1}|f''(x_l) - s''(x_l)| \leq \frac{3}{2}h^2\|f^{(4)}\|_{\infty,[a,b]}. \tag{3}
$$

Aus den Abschätzungen (1), (2) und (3) folgt dann die Behauptung. □

3 Numerische Integration

Sei $f : I \to \mathbb{R}$ eine stetige Funktion auf $I = [a, b]$ und f habe eine Stammfunktion F. Dann gilt nach dem Hauptsatz der Differential- und Integralrechnung

$$I(f) = \int_a^b f(x)\,\mathrm{d}x = F(b) - F(a).$$

Diese Formel lässt sich aber nicht ohne Weiteres anwenden, denn

1) Die Stammfunktion F ist im Allgemeinen nicht bekannt.

2) Häufig kennt man den Integranden f und eventuell einige seiner Ableitungen nur an einzelnen Punkten des Intervalls I.

Daraus folgt die Notwendigkeit, das Integral näherungsweise zu bestimmen.

Eine Approximation des Integrals ist zum Beispiel über Riemannsche Summen möglich, vergl. Abb. 3.1. Für Punkte $x_i = a + ih$, $i = 0, \ldots, n-1$ mit $h = \frac{b-a}{n}$ gilt

$$I(f) \approx S_n(f) = h \sum_{i=0}^{n-1} f(x_i).$$

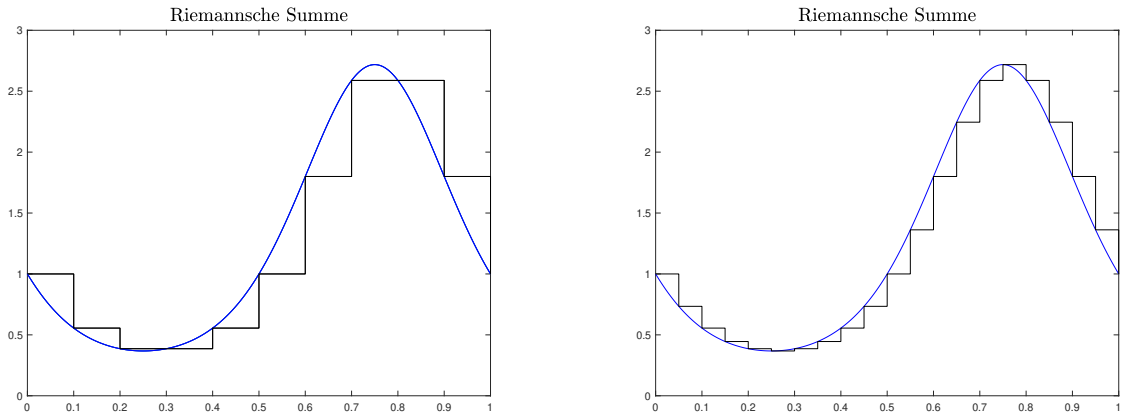


Abbildung 3.1: Riemannsche Summen definieren einfache Näherungsformeln zur Berechnung von bestimmten Integralen.

Es gilt $S_n(f) \to I(f)$ für $n \to \infty$, aber im Allgemeinen ist die Konvergenz langsam. Daher wollen wir in diesem Abschnitt besser geeignete Verfahren zur nährungsweisen Berechnung bestimmter Integrale kennenlernen.

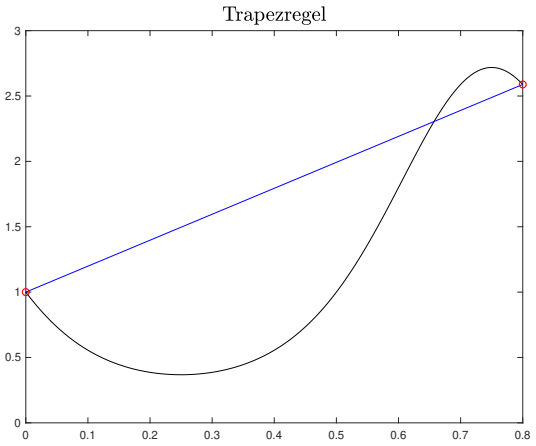
3.1 Einfache Integrationsformeln (Quadraturformeln)

Im folgenden Abschnitt werden wir zwei einfache Integrationsformeln zur näherungsweisen Berechnung von

$$I(f) = \int_a^b f(x)dx$$

kennenlernen.

1. Trapezregel



Die Trapezregel hat die Form

$$T(f) = \frac{b-a}{2}\left(f(a) + f(b)\right).$$

Man wählt als Näherungswert für den Wert des Integrals die Fläche des Trapezes mit den Ecken $(a, 0)$, $(b, 0)$, $(b, f(b)$, $(a, f(a))$.

In nebenstehendem Beispiel ist $a = 0$ und $b = 0.8$.

Alternativ können wir die Trapezregel wie folgt herleiten: Es sei $p_1 \in \mathbb{P}_1$ das lineare Polynom, das f in den Knoten a und b interpoliert, d.h.

$$\begin{aligned} p_1(x) &= p(f|a,b)(x) \\ &= \frac{(b-x)f(a) + (x-a)f(b)}{b-a}. \end{aligned}$$

Dann gilt

$$\begin{aligned} T(f) = \int_a^b p_1(x)\,\mathrm{d}x &= \left[\frac{f(b)}{b-a}\frac{1}{2}(x-a)^2 - \frac{f(a)}{b-a}\frac{1}{2}(b-x)^2\right]_a^b \\ &= \frac{1}{2}(b-a)f(b) + \frac{1}{2}(b-a)f(a) \\ &= \frac{b-a}{2}\left(f(a)+f(b)\right). \end{aligned}$$

Für das zu berechnende Integral gilt somit

$$I(f) = T(f) + R(f)$$

mit einem Fehler $R(f)$. Dieser Fehler berechnet sich für $f \in C^2([a,b])$ mit der Formel aus Satz 2.12 und dem Mittelwertsatz der Integralrechnung zu

$$\begin{aligned} R(f) = I(f) - T(f) &= \int_a^b \left(f(x) - p_1(x)\right)\,\mathrm{d}x \\ &= \int_a^b \frac{(x-a)(x-b)}{2!} f''(\xi(x))\,\mathrm{d}x \\ &= f''(\xi_0)\int_a^b \frac{(x-a)(x-b)}{2!}\,\mathrm{d}x \\ &= -\frac{1}{12}(b-a)^3 f''(\xi_0), \end{aligned}$$

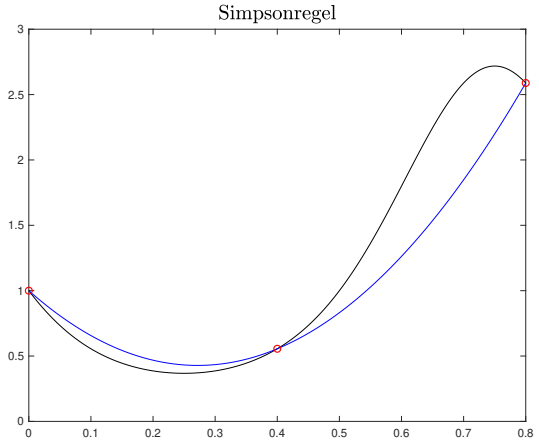
wobei $\xi_0 \in [a,b]$.

2. Simpsonregel

Es sei nun $p_2 \in \mathbb{P}_2$ das quadratische Polynom, das die Funktion f in den Knoten a, $\frac{a+b}{2}$ und b interpoliert, d.h. $p_2(x) = p(f|a,(a+b)/2,b)(x)$. Durch Integration von p_2 erhalten wir (ausführliche Rechnung siehe Übung) die Simpsonregel

$$S(f) = \int_a^b p_2(x)dx = \frac{h}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right),$$

mit $h = \frac{b-a}{2}$.



Die Fläche unter der blauen Kurve entspricht dem Nährungswert. Die schwarze Kurve beschreibt die zu integrierende Funktion.

Man kann zeigen (siehe Übung), dass die Simpsonregel nicht nur Polynome 2. Grades, sondern auch Polynome 3. Grades exakt integriert. Daraus folgt, ähnlich wie bei der Trapezregel, die Fehlerabschätzung

$$R(f) = I(f) - S(f) = -\frac{1}{90} h^5 f^{(4)}(\xi), \quad \xi \in [a, b],$$

falls $f \in C^4([a, b])$.

Das Vorgehen für die Trapez- und Simpsonregel lässt sich für beliebige Interpolationspolynome von f verallgemeinern, siehe auch Abschnitt 3.2.1. Dabei ersetzt man den Integranden durch ein Interpolationspolynom $p_n \in \mathbb{P}_n$ mit den Knoten $a \leq x_0 < x_1, \ldots, < x_n \leq b$ und setzt

$$Q(f) = \int_a^b p_n(x)\, \mathrm{d}x = \sum_{i=0}^{n} c_i f(x_i). \qquad \text{(Interpolatorische Integrationsformel)}$$

Diese Formeln erben allerdings die schlechten Eigenschaften der Interpolation mit Polynomen hohen Grades. Eine Ausnahme bilden hier die sogenannten Gauß-Quadraturformeln, die wir in Abschnitt 3.4 kennenlernen werden.

Eine weitere Möglichkeit besteht darin, das Intervall $I = [a, b]$ in Teilintervalle $I_1, \ldots, I_n$ zu zerlegen und in jedem Teilintervall I_i eine einfache Integrationsformel (zum Beispiel die Trapezregel) anzuwenden. Dieser Vorgang ähnelt dem Übergang von der Interpolation mit Polynomen zur Spline-Interpolation. Als wichtiges Beispiel werden wir in Abschnitt 3.3 die iterierte Trapezregel genauer untersuchen.

3.2 Grundlegende Definitionen

Definition 3.1. Es seien $a \leq x_0 < \cdots < x_n \leq b$. Ein auf $C[a,b]$ definiertes lineares Funktional der Form

$$Q_n(f) := \sum_{i=0}^{n} a_i f(x_i), \quad a_i \in \mathbb{R}$$

heißt *Quadraturformel (Integrationsformel)* n-ter Ordnung auf dem Grundintervall $[a,b]$. Die x_i heißen Stützstellen der Quadraturformel, die a_i heißen Gewichte der Quadraturformel.

Die Quadraturformel heißt *abgeschlossen*, wenn $a = x_0$ und $x_n = b$. Im Falle $a < x_0$ und $b > x_n$ heißt die Quadraturformel *offen*.

Definition 3.2. Ein *Quadraturverfahren* ist eine Folge von Quadraturformeln wachsender Ordnung zum gleichen Grundintervall.

Kommt es auf die Unterscheidung der Gewichte der verschiedenen Quadraturformeln eines Quadraturverfahrens an, dann schreibt man a_{in} an Stelle von a_i, analog bei den Stützstellen.

Definition 3.3. Der *Rest* einer Quadraturformel Q_n ist das lineare Funktional $R_n = I - Q_n$. Hierbei ist $I(f) := \int_a^b f(x)dx$.

Definition 3.4. Ein Quadraturverfahren Q_n, $n = 1, 2, \ldots$ heißt konvergent für eine Funktion f, wenn

$$\lim_{n\to\infty} Q_n(f) = I(f)$$

gilt

Konvergenz bedeutet also immer Konvergenz gegen den richtigen Wert $I(f)$.

Eine einfache Beziehung zwischen Quadraturformeln auf verschiedenen Grundintervallen wird durch die folgende Transformation beschrieben.

Transformationsformel: Es sei eine Quadraturformel zum Grundintervall $[a,b]$ gegeben, also

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=0}^{n} a_i f(x_i) + R[f],$$

aber die zu integrierende Funktion g sei aus $[\tilde{a},\tilde{b}]$. Dann gilt

$$\int_{\tilde{a}}^{\tilde{b}} g(x)\,\mathrm{d}x = \sum_{i=0}^{n} \tilde{a}_i g(\tilde{x}_i) + \frac{\tilde{b}-\tilde{a}}{b-a} R[g]$$

mit $\tilde{a}_i = \frac{\tilde{b}-\tilde{a}}{b-a} a_i$ und $\tilde{x}_i = \tilde{a} + (x_i - a)\left(\frac{\tilde{b}-\tilde{a}}{b-a}\right)$ (siehe Übung).

3.2.1 Interpolationsquadratur

Definition 3.5. Eine Quadraturformelformel heißen *Interpolationsquadraturformel*, wenn für ihren Rest

$$R_n(p) = 0 \quad \forall p \in \mathbb{P}_n$$

gilt.

Ein Interpolationsquadraturverfahren ist ein Quadraturverfahren, dessen Quadraturformeln sämtlich Interpolationsquadraturformeln sind.

Newton-Cotes-Formeln sind Interpolationsquadraturformeln mit äquidistanten Stützstellen.

Für Interpolationsquadraturformeln gilt

Satz 3.6. *Zu jedem System $a \leq x_0 < x_1 < \ldots < x_n \leq b$ paarweise verschiedener Stützstellen gibt es genau eine Interpolationsquadraturformel mit diesen Stützstellen.*

Beweis. Q_n ist offenbar dann und nur dann eine Interpolationsquadraturformel, wenn

$$Q_n(p_i) = \frac{b^{i+1} - a^{i+1}}{i+1}, \quad i = 0, 1, \ldots, n,\ p_i(x) = x^i$$

gilt. Wir erhalten also ein lineares Gleichungssystem für die Gewichte, nämlich

$$\sum_{j=0}^{n} a_j x_j^i = \frac{b^{i+1} - a^{i+1}}{i+1}, \quad i = 0, \ldots, n,$$

dessen Koeffizientenmatrix die Vandermondesche Matrix ist. Also ist das System eindeutig lösbar. □

Der Beweis liefert uns zugleich eine Methode zur expliziten Konstruktion von Interpolationsquadraturformeln. Alternativ können wir Interpolationsquadraturformeln herleiten, indem wir die zu integrierende Funktion f durch ein Interpolationspolynom ersetzen und dieses exakt integrieren.

Es sei $p_n \in \mathbb{P}_n$ mit $p_n(x_j) = f_j$ für $j = 0, \ldots, n$. Die Lagrange-Darstellung von p_n ist gegeben durch

$$p_n(x) = \sum_{j=0}^{n} f_j l_{jn}(x), \quad l_{jn}(x) = \prod_{i=0, i\neq j}^{n} \frac{x - x_i}{x_j - x_i}.$$

Dann gilt

$$Q_n(f) := \int_a^b p_n(x)\,\mathrm{d}x = \sum_{j=0}^{n} f(x_j) \int_a^b l_{jn}(x)\,\mathrm{d}x = \sum_{j=0}^{n} a_j f(x_j)$$

mit

$$a_j = \int_a^b l_{jn}(x)\,\mathrm{d}x.$$

Diese Herleitung kann man alternativ zur Definition von Interpolationsquadraturformeln nutzen.

Der einfachste Fall einer Interpolationsquadraturformel mit genau einer Stützstelle x_0 liefert die Quadraturformel

$$Q_0(f) = (b-a)f(x_0).$$

Nach Definition ist klar, dass diese Formel Polynome des Grades Null exakt integriert. Man kann jedoch leicht nachprüfen, dass die spezielle Wahl $x_0 = (a+b)/2$ eine Interpolationsquadraturformel liefert, die sogar Polynome ersten Grades exakt integriert. Die Verallgemeinerung dieses Sachverhaltes spielt bei der Betrachtung von Quadraturverfahren eine große Rolle.

Für $n = 2$, $[a, b] = [-1, 1]$ und bei Verwendung der Knoten

$$-c, 0, c \quad (0 < c \leq 1)$$

ergibt sich nach kurzer Rechnung

$$Q_2(f) = \frac{1}{3c^2} f(-c) + \left(2 - \frac{2}{3c^2}\right) f(0) + \frac{1}{3c^2} f(c).$$

Im Spezialfall $c = 1/\sqrt{3}$ verschwindet das mittlere Gewicht. Es entsteht also eine Quadraturformel mit zwei Knoten, die aber nach Konstruktion alle Polynome $p \in \mathbb{P}_2$ exakt integriert. Ähnlich sind alle Fälle von Graderhöhungen zu erklären, die uns in späteren Abschnitten begegnen werden.

Wir geben nun noch einige klassische Quadraturformeln an.

Klassische Quadraturformeln auf $[0, 1]$:

	Stützstellen	Gewichte a_j
Rechteckregel	0	1
Mittelpunktregel	$\frac{1}{2}$	1
Trapezregel	$0, 1$	$\frac{1}{2}, \frac{1}{2}$
Simpsonregel	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{2}{3}, \frac{1}{6}$
$3/8-$Regel	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$
Milne $-$ Regel	$0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$

Für $n > 6$ und äquidistante Stützstellen treten negative Gewichte auf, so dass die Formeln numerisch unbrauchbar werden. Stattdessen benutzt man zusammengesetzte Quadraturformeln oder nich äquidistante Knoten (Gauss Quadratur, siehe Abschnitt 3.4).

3.2.2 Zusammengesetzte Quadraturverfahren

Definition 3.7. Es sei Q_n eine Quadraturformel auf dem Grundintervall $[0,1]$ und es sei $H = \frac{b-a}{m}$. Ihre auf $[a+(r-1)H, a+rH]$ Transformierte werde mit $Q_n^{(r)}$ bezeichnet. Dann heißt:

$$Q_{(m)} := Q_n^{(1)} + Q_n^{(2)} + \dots Q_n^{(m)}$$

die durch m-fache Anwendung der Elementarformel Q_n entstandene *zusammengesetzte Quadraturformel*. Ein *zusammengesetztes Quadraturverfahren* erhalten wir durch $m = 1, 2, \dots$-fache Anwendnung einer festen Elementarformel, zum Beispiel die iterierte Trapezregel oder das iterierte Simpson-Verfahren.

Zusammengesetzte Quadraturverfahren haben also eine sehr einfache Struktur. Insbesondere haben die einzelnen Formeln nur wenige verschiedene Gewichte und sind daher leicht auszuwerten. Das einfachste Beispiel eines zusammengesetzten Quadraturverfahrens ist das Mittelpunktverfahren

$$Q_{(m)}^{Mi}(f) = H \sum_{i=1}^{m} f(x_i), \quad H = (b-a)/m, \quad x_i = a + (i - \frac{1}{2})H.$$

es entsteht aus der Elementarformel $Q_0(f) = f(\frac{1}{2})$.

Weitere bekannte zusammengesetzte Quadraturverfahren sind das Trapezverfahren

$$Q_{(m)}^{Tr}(f) = H \left(\frac{1}{2} f(a) + \sum_{i=1}^{m-1} f(x_i) + \frac{1}{2} f(b) \right), \quad H = (b-a)/m, \quad x_i = a + iH$$

und das Simpsonverfahren

$$Q_{(m)}^{Si}(f) = \frac{h}{3} \left(f(a) + 4 \sum_{i=1}^{m} f(x_{2i}) + 2 \sum_{i=1}^{m-1} f(x_{2i+1}) + f(b) \right)$$

mit

$$h = (b-a)/(2m), \quad x_i = a + (i-1)h.$$

Satz 3.8. *Ein zusammengesetztes Quadraturverfahren ist genau dann für alle stetigen Funktionen konvergent, wenn seine Elementarformeln Konstanten exakt integrieren.*

Beweis. Es sei

$$Q_n^{el}(f) = \sum_{i=0}^{n} a_i f(x_i)$$

eine Elementarformel auf $[0,1]$ und

$$Q_n^{(r)}(f) = \sum_{i=0}^{n} a_i H f(a + (r-1)H + x_i H)$$

sei die transformierte Elementarformel auf $[a + (r-1)H, a + rH]$. Dann gilt für das zusammengesetzte Quadraturverfahren

$$\begin{aligned} Q_{(m)}(f) &= \sum_{r=1}^{m} \sum_{i=0}^{n} a_i H f\big(a + (r-1+x_i)H\big), \quad m = 1, 2, \dots \\ &= \sum_{i=0}^{n} a_i \underbrace{\left[H \sum_{r=1}^{m} f\big(a + (r-1+x_i)H\big) \right]}_{\text{Riemannsche Summe für das Integral}}. \end{aligned}$$

In den eckigen Klammern stehen Riemannsche Summen, also gilt

$$\lim_{m\to\infty} Q_{(m)}(f) = \sum_{i=0}^{n} a_i \int_a^b f(x)\, \mathrm{d}x.$$

Hieraus folgt sofort die Behauptung, denn

$$\sum_{i=0}^{n} a_i = 1$$

ist äquivalent mit $R_n^{el}(p_0) = 0$ für alle $p_0 \in \mathbb{P}_0$. □

3.3 Die iterierte Trapezregel

Wir werden nun die iterierte Trapezregel genauer untersuchen. Wir betrachten Teilintervalle der Länge $h = \frac{b-a}{n}$, $n \in \mathbb{N}$. Dann gilt

$$\begin{aligned} Q(f) &= \sum_{i=0}^{n-1} h \frac{f(a+ih) + f(a+(i+1)h)}{2} \\ &= h\left(\frac{1}{2} f(a) + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{1}{2} f(b) \right) \\ &= T_h(f) \end{aligned}$$

und für das Integral somit

$$I(f) = T_h(f) + R_h(f)$$

für einen von h abhängigen Fehler $R_h(f)$. (Zur Vereinfachung der Notation bezeichnen wir in diesem Abschnitt die iterierte Trapezregel mit $T_h(f)$.)

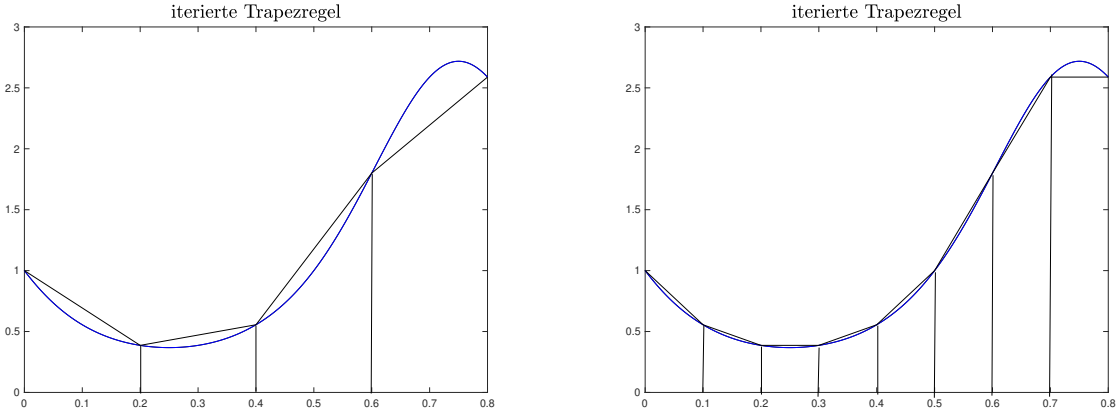


Abbildung 3.2: Illustration der iterierten Trapezregel mit $n = 4$ und $n = 8$.

Satz 3.9. *Es gilt*

1) *Ist* $f \in C([a,b])$, *so gilt* $R_h(f) \to 0$, *das heißt* $T_h(f) \to I(f)$ *für* $h \to 0$.

2) *Gilt sogar* $f \in C^2([a,b])$, *so folgt die Fehlerabschätzung* $R_h(f) = \mathcal{O}(h^2)$.

Beweis. Die erste Aussage folgt aus Satz 3.8, da die Trapezregel Konstanten exakt integriert.

Die zweite Aussage folgt direkt aus der Fehlerabschätzung der Trapezregel, denn

$$\begin{aligned} |R_h(f)| &= \left| \sum_{i=1}^{n} \left(-\frac{1}{12} h^3 f''(\xi_i) \right) \right|, \quad \xi_i \in [x_{i-1}, x_i] \\ &\leq \frac{1}{12} h^3 n \|f''\|_{\infty,[a,b]} \\ &= \frac{b-a}{12} h^2 \|f''\|_{\infty,[a,b]} \\ &= \mathcal{O}(h^2). \end{aligned}$$

□

Wir wollen uns nun den Fehler der iterierten Trapezregel genauer ansehen, d.h. wir möchten den Term $\mathcal{O}(h^2)$ genauer beschreiben. Diese genauere Darstellung des Fehlers der iterierten Trapezregel werden wir in Abschnitt 3.3.3 nutzen, um Quadraturverfahren höherer Genauigkeit herzuleiten. In Abschnitt 3.3.1 stellen wir zunächst Hilfsmittel bereit, die wir in Abschnitt 3.3.2 für den Beweis der genaueren Darstellung des Fehlers der iterierten Trapezregel nutzen.

3.3.1 Bernoulli-Polynome

Definition 3.10. Das durch die folgende Rekursionsvorschrift festgelegte System von Polynomen B_r heißt das System der *Bernoulli-Polynome*.

(i) $B_0(x) = 1$,

(ii) $B'_{r+1}(x) = (r+1)B_r(x)$, $r = 0, 1, 2, \ldots$,

(iii) $\int_0^1 B_r(x)\,\mathrm{d}x = 0$.

Die ersten Bernoulli-Polynome sind

- $B_0(x) = 1$,
- $B_1(x) = x - \frac{1}{2}$,
- $B_2(x) = x^2 - x + \frac{1}{6}$,
- $B_3(x) = x^3 - \frac{3}{2}x^2 + \frac{1}{2}x$,
- $B_4(x) = x^4 - 2x^3 + x^2 - \frac{1}{30}$.

Offensichtlich gilt $B_r \in \mathbb{P}_r$.

Satz 3.11. *(i) Es gilt* $B_r(x) = (-1)^r B_r(1-x)$.

(ii) Für $r \geq 2$ *gilt* $B_r(1) = B_r(0)$.

(iii) Für alle $r \geq 1$ *gilt* $B_{2r+1}(0) = B_{2r+1}(\frac{1}{2}) = B_{2r+1}(1) = 0$.

Beweis. Teil (i) wird in den Übungen bewiesen.

Aus der definierenden Bedingung Def. 3.10 (ii) folgt

$$\int B_r(x)dx = \frac{1}{r+1}B_{r+1}(x).$$

Wegen Def. 3.10 (iii) gilt dann

$$0 = \frac{1}{r+1}\left(B_{r+1}(1) - B_{r+1}(0)\right) \quad \text{für } r \geq 1$$

und somit folgt die 2. Aussage.

Die 3. Aussage folgt direkt aus (i). □

Definition 3.12. Die Konstanten $B_n(0) := B_n$ heißen Bernoulli-Zahlen.

Wir erhalten aus den Polynomen sofort die Bernoulli-Zahlen $B_0 = 1$, $B_1 = -\frac{1}{2}$, $B_2 = \frac{1}{6}$, $B_3 = 0$, $B_4 = -\frac{1}{30}$.

Bemerkung 3.13. Im folgenden Abschnitt verwenden wir skalierte Bernoulli-Polynome

$$b_r(x) = \frac{1}{r!} B_r(x).$$

Offensichtlich gilt Satz 3.11 analog für die Polynome b_r.

3.3.2 Die Eulersche Summenformel

In diesem Abschnitt wollen wir eine genauere Darstellung der Struktur des Fehlers der iterierten Trapezregel betrachten, das heißt eine Entwicklung nach Potenzen von h^2. Dazu sei $f : [a, b] \to \mathbb{R}$ genügend oft differenzierbar, $h := \frac{b-a}{n}$, $s = a + th$ für $t \in [0, n]$. Weiter sei $g(t) = f(a + th)$ für $t \in [0, n]$ und es gelte $\mathrm{d}s = h\mathrm{d}t$. Für genügend oft differenzierbares f gilt folglich

$$g^{(i)}(t) = f^{(i)}(a + th)h^i$$

und

$$\begin{aligned}
\int_a^b f(s)\,\mathrm{d}s &= h \int_0^n f(a + th)\,\mathrm{d}t \\
&= h \int_0^n g(t)\,\mathrm{d}t \\
&= h \sum_{r=1}^{n} \int_{r-1}^{r} g(t)\,\mathrm{d}t \\
&= h(T_h(g) + R_h(g))
\end{aligned}$$

Um $R_h(g)$ zu berechnen, betrachten wir für $r = 1, \ldots, n$ die Integrale

$$\int_{r-1}^{r} g(t)\,\mathrm{d}t = \int_{r-1}^{r} g(t) \cdot 1\,\mathrm{d}t = \left[g(t) \left(t - \left(r - \frac{1}{2} \right) \right) \right]_{r-1}^{r} - \int_{r-1}^{r} \left(t - \left(r - \frac{1}{2} \right) \right) g'(t)\,\mathrm{d}t,$$

wobei wir bei der partiellen Integration die Stammfunktion von $F' = 1$ so gewählt haben, dass

$$\int_{r-1}^{r} F(t)\,\mathrm{d}t = 0$$

gilt. Weiterhin können wir

$$F(t) = t - \left(r - \frac{1}{2}\right) = t - (r-1) - \frac{1}{2} = t - \lfloor t \rfloor - \frac{1}{2}, \quad t \in [r-1, r)$$

schreiben, um uns vom Index r zu lösen. Hier bezeichnet $\lfloor t \rfloor$ die größte ganze Zahl $m \in \mathbb{Z}$ mit $m \leq t$. Insgesamt erhalten wir dann

$$\begin{aligned}
\int_0^n g(t)\,\mathrm{d}t &= \sum_{r=1}^{n} \int_{r-1}^{r} g(t)\,\mathrm{d}t \\
&= \sum_{r=1}^{n} \left[g(t) \cdot F(t)\right]_{r-1}^{r} - \sum_{r=1}^{n} \int_{r-1}^{r} F(t) g'(t)\,\mathrm{d}t \\
&= \underbrace{\frac{1}{2}g(0) + g(1) + \cdots + g(n-1) + \frac{1}{2}g(n)}_{=T_h(g)} \underbrace{- \int_0^n F(t) g'(t) dt}_{=R_h(g)}
\end{aligned}$$

Wenden wir nun partielle Integration auf $R_h(g)$ an und benutzen die Darstellung der skalierten Bernoulli-Polynome, so erhalten wir

$$\begin{aligned}
R_h(g) &= -\int_0^n F(t) g'(t)\,\mathrm{d}t \\
&= -\sum_{r=1}^{n} \left(\int_{r-1}^{r} \left(t - \lfloor t \rfloor - \frac{1}{2}\right) g'(t) \mathrm{d}t \right) \\
&= -\sum_{r=1}^{n} \left(\int_{r-1}^{r} b_1\left(t - \lfloor t \rfloor\right) g'(t)\,\mathrm{d}t \right) \\
&= -\sum_{r=1}^{n} \left[b_2(t-(r-1)) g'(t) \right]_{r-1}^{r} - \int_{r-1}^{r} b_2\left(t - \lfloor t \rfloor\right) g''(t)\,\mathrm{d}t \\
&= \dots \\
&= -\sum_{r=1}^{n} \sum_{j=2}^{m} (-1)^j \left(b_j(1) g^{(j-1)}(r) - b_j(0) g^{(j-1)}(r-1) \right) + (-1)^m \int_0^n b_m\left(t - \lfloor t \rfloor\right) g^{(m)}(t)\,\mathrm{d}t \\
&= -\sum_{j=2}^{m} (-1)^j \left(b_j(1) g^{(j-1)}(n) - b_j(0) g^{(j-1)}(0) \right) + (-1)^m \int_0^n b_m\left(t - \lfloor t \rfloor\right) g^{(m)}(t)\,\mathrm{d}t
\end{aligned}$$

In der 4. Zeile wird über die Integrale summiert; das Minus-Zeichen vor der Summe geht für das gesamte Integral ab der 6. Zeile in den Vor-Faktor $(-1)^m$ ein. Ferner folgt die letzte Gleichung aufgrund der Tatsache, dass nach Satz 3.11 (ii) $b_j(1) = b_j(0)$ gilt und somit die inneren Terme der Summe über r verschwinden. Wenden wir jetzt Satz 3.11 (iii), also $b_{2r+1}(1) = b_{2r+1}(0)$, an, so bekommen wir

$$-\sum_{i=1}^{k}\Big(b_{2i}(1)g^{(2i-1)}(n) - b_{2i}(0)g^{(2i-1)}(0)\Big) + \begin{cases} +\int_0^n b_{2k}\left(t-\lfloor t\rfloor\right) g^{(2k)}(t)\,\mathrm{d}t, & m=2k, \\ -\int_0^n b_{2k+1}\left(t-\lfloor t\rfloor\right) g^{(2k+1)}(t)\,\mathrm{d}t, & m=2k+1. \end{cases}$$

Der obere Integralterm liefert dann (für $m = 2k+2$ statt $m = 2k$):

$$\begin{aligned}
\int_a^b f(s)\,\mathrm{d}s &= h\int_0^n f(a+th)\,\mathrm{d}t = h\int_0^n g(t)\mathrm{d}t \\
&= h\left[\frac{1}{2}g(0) + \sum_{r=1}^{n-1} g(r) + \frac{1}{2}g(n) - \sum_{i=1}^{k+1} b_{2i}(0)\big(g^{(2i-1)}(n) - g^{(2i-1)}(0)\big)\right. \\
&\quad \left. + \int_0^n b_{2k+2}(t-\lfloor t\rfloor)g^{(2k+2)}(t)\,\mathrm{d}t\right] \\
&= h\left[\frac{1}{2}f(a) + \sum_{r=1}^{n-1} f(a+rh) + \frac{1}{2}f(b) - \sum_{i=1}^{k+1} h^{2i-1}b_{2i}(0)\big(f^{(2i-1)}(b) - f^{(2i-1)}(a)\big)\right. \\
&\quad \left. + h^{2k+1}\int_a^b b_{2k+2}\left(\frac{s-a}{h} - \left\lfloor\frac{s-a}{h}\right\rfloor\right) f^{(2k+2)}(s)\,\mathrm{d}s\right]
\end{aligned}$$

Wir haben also den folgenden Satz bewiesen.

Satz 3.14. *Für $f \in C^{2k+2}([a,b])$ gilt*

$$T_h(f) = \int_a^b f(s)\,\mathrm{d}s + \sum_{i=1}^{k+1} c_i h^{2i} - h^{2k+2} I_{2k+2}(f)$$

mit $c_i = b_{2i}(0)\big(f^{(2i-1)}(b) - f^{(2i-1)}(a)\big)$ und

$$I_{2k+2}(f) = \int_a^b b_{2k+2}\left(\frac{s-a}{h} - \left\lfloor\frac{s-a}{h}\right\rfloor\right) f^{(2k+2)}(s)\,\mathrm{d}s \in \mathbb{R}.$$

Folglich besitzt der Fehler

$$R_h(f) = \int_a^b f(s)\,\mathrm{d}s - T_h(f)$$

eine Entwicklung nach Potenzen von h^2.

Bemerkung. 1) Insbesondere gilt für periodische Funktionen $f \in C^{2m}(\mathbb{R})$ mit Periode $b-a$ die Gleichung

$$R_h(f) = h^{2m} I_{2m}(f), \quad I_{2m}(f) \in \mathbb{R}.$$

Unter Verwendung der Bernoulli-Zahlen erhalten wir $c_i = \frac{B_{2i}}{(2i)!}\left(f^{(2i-1)}(b) - f^{(2i-1)}(a)\right)$.

3.3.3 Das Romberg-Verfahren

Wir werden nun eine andere Art von Integrationsverfahren kennenlernen, die auf der Trapezsumme basieren. Die bisher besprochenen Quadraturformeln bezogen sich alle auf ein festes Gitter $t_0, \ldots, t_n$ von Knoten, an denen die Funktion ausgewertet wurde. Im Gegensatz dazu benutzen wir bei der Romberg-Quadratur eine Folge von Gittern und versuchen aus den dazu gehörenden Trapezsummen eine bessere Approximation des Integrals zu konstruieren.

Es sei $h = \frac{b-a}{n}$ und $0 < q < 1$ mit $\frac{b-a}{qh} \in \mathbb{N}$. Es gilt $T_h(f) = I(f) - R_h(f)$. Seien weiter $T_h(f)$ und $T_{qh}(f)$ berechnet. Dann folgt

$$\begin{aligned} T_h(f) &= I(f) + c_1 h^2 + c_2 h^4 + \ldots \\ T_{qh}(f) &= I(f) + c_1 (qh)^2 + c_2 (qh)^4 + \ldots \end{aligned}$$

Wir multiplizieren die erste Gleichung mit q^2 und subtrahieren die zweite Gleichung von der ersten Gleichung. Somit erhalten wir

$$(q^2-1)I(f) = q^2 T_h(f) - T_{qh}(f) - c_2(q^2-q^4)h^4 - c_3(q^2-q^6)h^6 + \ldots$$

Folglich ist

$$I(f) = \frac{q^2 T_h(f) - T_{qh}(f)}{q^2-1} + c_2 q^2 h^4 + c_3 q^2(1+q^2)h^6 + \ldots$$

und wir sehen, dass die Fehlerentwicklung bei h^4 beginnt. Dieses Vorgehen kann man weiter wiederholen.

Extrapolation

Wir werden nun noch eine alternative Herleitung des Romberg-Verfahrens betrachten.

Es seien $h_0 > h_1 > \cdots > h_n > 0$ Schrittweiten mit $\frac{b-a}{h_i} \in \mathbb{N}$ für $i = 0, \ldots, n$. Dann lautet das Vorgehen wie folgt:

1) Berechne $T_{h_i}(f)$ für alle $i = 0, \ldots, n$.

2) Interpoliere $(h_i^2, T_{h_i}(f))$ für $i = 0, \ldots, n$ mit einem Polynom $p \in \mathbb{P}_n$ in h^2, das heißt

$$p(h^2) = a_0 + a_1 h^2 + a_2 h^4 + \cdots + a_n h^{2n}$$

mit

$$p(h_i^2) = T_{h_i}(f), \quad i = 0, \ldots, n.$$

3) Betrachte $p(0)$ als neue Näherung für $I(f)$ (Extrapolation auf $h = 0$).

Berechnung von $p(0)$ nach dem Schema von Aitken-Neville

Sei $p_{j,k} \in \mathbb{P}_k$ für $k \leq j$ dasjenige Polynom in h^2, mit

$$p_{j,k}(h_i^2) = T_{h_i}(f) \text{ für } i = j-k, \ldots, j.$$

Für $k = 0$ sei

$$p_{j,0}(h^2) := T_{h_j}(f) \quad j = 0, \ldots, n.$$

Unter Verwendung von Lemma 2.7 erhalten wir für $k > 0$:

$$p_{j,k}(h^2) = \frac{p_{j-1,k-1}(h^2)(h^2 - h_j^2) + p_{j,k-1}(h^2)(h_{j-k}^2 - h^2)}{h_{j-k}^2 - h_j^2}$$

Daraus folgt:

$$\begin{aligned} p_{j,k}(0) &= \frac{p_{j,k-1}(0) h_{j-k}^2 - p_{j-1,k-1}(0) h_j^2}{h_{j-k}^2 - h_j^2} \\ &= \frac{p_{j,k-1}(0) \left(\frac{h_{j-k}}{h_j}\right)^2 - p_{j-1,k-1}(0)}{\left(\frac{h_{j-k}^2}{h_j^2} - 1\right)} \\ &= p_{j,k-1}(0) + \frac{p_{j,k-1}(0) - p_{j-1,k-1}(0)}{\left(\frac{h_{j-k}^2}{h_j^2} - 1\right)} \end{aligned}$$

Wir verwenden im Folgenden die Notation $T_{j,k} := p_{j,k}(0)$. Dann gilt:

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\left(\frac{h_{j-k}^2}{h_j^2} - 1\right)}$$

Schematische Darstellung des Romberg Verfahrens

$$
\begin{array}{llll}
T_{0,0} & & & \\
& T_{1,1} & & \\
T_{1,0} & & T_{2,2} & \\
& T_{2,1} & & T_{3,3} \\
T_{2,0} & & T_{3,2} & \\
& T_{3,1} & & \\
T_{3,0} & & & \\
\vdots & & &
\end{array}
$$

Es gibt zwei typische Möglichkeiten, die Folge der Schrittweiten zu wählen:

- Romberg-Folge: $h_0 = b - a$, $\frac{h_0}{2}, \frac{h_0}{4}, \ldots,$
- Bulirsch-Folge: $h_0 = b - a$, $\frac{h_0}{2}, \frac{h_0}{3}, \ldots$

Dabei wächst die Anzahl der Funktionsauswertungen bei der Bulirsch-Folge weniger schnell, aber man erhält eine langsamere Konvergenz.

Allgemein gilt: Die Berechnung der Größen $T_{0,0}, \ldots, T_{n,0}$ kann vergleichsweise viel Zeit in Anspruch nehmen, während die Berechnung der anderen Größen T_{ij} für $j \geq i$ nicht ins Gewicht fällt, aber im Allgemeinen einen großen Gewinn bringt.

Bemerkung 3.15. (1) Für $h_0 = (b - a)$, und $h_1 = \frac{h_0}{2}$ folgt

$$T_{1,1} = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right).$$

Dies ist gerade die Simpson-Formel, was sich leicht durch Nachrechnen überprüfen lässt (Übung).

(2) Abbruchkriterium: Setze $U_{j,k} := 2T_{j+1,k} - T_{j,k}$. Dann lässt sich zeigen (Bulirsch/-Stoer), dass asymptotisch

$$U_{j,k} - I(f) \approx I(f) - T_{j,k}$$

gilt. Daraus folgt

$$U_{j,k} - T_{j,k} \approx 2(I(f) - T_{j,k})$$

und

$$2(T_{j+1,k} - T_{j,k}) = U_{j,k} - T_{j,k} \approx 2(I(f) - T_{j,k}).$$

Damit lässt sich dann

$$|T_{j+1,k} - T_{j,k}| \leq \varepsilon$$

als geeignetes Abbruchkriterium festlegen, denn dann gilt

$$|I(f) - T_{j,k}| \approx \varepsilon,$$

das heißt der Fehler ist etwa von der Größenordnung ε. In der Praxis geht man typischerweise wie folgt vor: Falls $T_{j,k}$ dieses Kriterium erfüllt, berechne auch noch $T_{j+1,k+1}$ und verwende diesen Wert als endgültige Näherung.

(3) Für die Ordnung des Verfahrens gilt

$$|T_{j,k} - I(f)| = \mathcal{O}\left(\prod_{i=j-k}^{j} h_i^2\right),$$

das heißt die j-te Spalte des Romberg-Dreiecks (iterierte Trapezregel) konvergiert mit h^{2j+2} gegen $I(f)$.

Beispiel. Betrachte

$$\int_0^1 e^t \,\mathrm{d}t \approx 1{,}718\,281\,828\,459.$$

Weiter sei $h_0 = b - a$ und $h_i = \frac{h_{i-1}}{2}$ (Romberg-Folge). Dann gilt

$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\left(\frac{h_{j-k}}{h_j}\right)^2 - 1} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{4^k - 1} = \frac{4^k T_{j,k-1} - T_{j-1,k-1}}{4^k - 1}.$$

Das Romberg-Dreieck sieht dann wie folgt aus:

	$k=0$	1	2	3
$j=0$	1,8591			
1	1,7539	1,718 861		
2	1,7272	1,718 318	1,718 282 687	
3	1,7205	1,718 284	1,718 281 842	1,718 281 829
$\vdots$	$\vdots$			
12	1,718 281 837			

3.4 Gaußsche Quadraturformeln

Die interpolatorischen Quadraturformeln

$$Q_n(f) = \sum_{i=0}^{n} a_i f(x_i)$$

zu den Stützstellen $x_0, \ldots, x_n \in [a, b]$ sind nach Konstruktion exakt für alle Polynome n-ten Grades, das heißt für ihr Restglied gilt

$$R_n(p) = I(p) - Q_n(p) = 0, \quad \text{für alle} \quad p \in \mathbb{P}_n.$$

Unser Ziel in diesem Abschnitt ist es, die Stützstellen $x_0, \ldots, x_n$ und Gewichte $a_0, \ldots, a_n$ so zu wählen, dass der Exaktheitsgrad der Quadraturformel möglichst hoch wird.

Lemma 3.16. *Eine obere Grenze für den Exaktheitsgrad der Quadraturformel $Q_n(f)$ ist $2n+1$, das heißt es können höchstens alle Polynome $p \in \mathbb{P}_{2n+1}$ exakt integriert werden.*

Beweis. Es sei Q_n exakt für Polynome $p \in \mathbb{P}_{2n+2}$. Dann gilt

$$Q_n(p) = I(p) \quad \text{für} \quad p(x) = \prod_{i=0}^{n} (x - x_i)^2 \in \mathbb{P}_{2n+2}.$$

Daraus ergibt sich aber der Widerspruch

$$0 < \int_a^b p(x)\,\mathrm{d}x = Q_n(p) = 0. \qquad \square$$

Nun wollen wir zeigen, dass es tatsächlich Quadraturformeln zu $n+1$ Stützstellen gibt, die den maximalen Exaktheitsgrad $2n+1$ haben.

Dazu seien $p_n \in \mathbb{P}_n$ und $p_{2n+1} \in \mathbb{P}_{2n+1}$ die Interpolationspolynome einer Funktion $f \in C([a,b])$ zu den $n+1$ beziehungsweise $2n+2$ Stützstellen $x_0, \ldots, x_n, x_{n+1}, \ldots, x_{2n+1} \in [a,b]$. Für die zugehörigen Quadraturformeln Q_n und Q_{2n+1} gilt dann

$$\begin{aligned} I(f) - Q_{2n+1}(f) &= I(f) - \int_a^b \sum_{i=0}^{2n+1} [x_0 \ldots, x_i] f \prod_{j=0}^{i-1} (x - x_j)\,\mathrm{d}x \\ &= I(f) - \sum_{i=0}^{2n+1} [x_0 \ldots, x_i] f \int_a^b \prod_{j=0}^{i-1} (x - x_j)\,\mathrm{d}x \\ &= I(f) - Q_n(f) - \sum_{i=n+1}^{2n+1} [x_0 \ldots, x_i] f \int_a^b \prod_{j=0}^{i-1} (x - x_j)\,\mathrm{d}x. \end{aligned}$$

Für $i = n+1, \ldots, 2n+1$ schreiben wir nun

$$\int_a^b \prod_{j=0}^{i-1} (x - x_j)\,\mathrm{d}x = \int_a^b \prod_{j=0}^{n} (x - x_j) \prod_{j=n+1}^{i-1} (x - x_j)\,\mathrm{d}x.$$

Beachte, dass die $n+1$ Polynome

$$\left\{ 1, x - x_{n+1}, (x - x_{n+1})(x - x_{n+2}), \ldots, \prod_{j=n+1}^{2n} (x - x_j) \right\}$$

eine Basis von $\mathbb{P}_n$ bilden. Wähle nun die ersten $n+1$ Stützstellen $x_0, \ldots, x_n \in [a,b]$ so, dass

$$\int_a^b \prod_{j=0}^{n} (x - x_j) q(x) \, \mathrm{d}x = 0 \quad \text{für alle} \quad q \in \mathbb{P}_n. \tag{$*$}$$

Dann folgt

$$I(p) - Q_n(p) = I(p) - Q_{2n+1}(p) = 0 \quad \forall p \in \mathbb{P}_{2n+1}.$$

Das heißt, die interpolatorische Quadraturformel Q_n ist exakt für Polynome aus $\mathbb{P}_{2n+1}$.

3.4.1 Konstruktion von Quadraturformeln mit maximalem Exaktheitsgrad

Auf dem Funktionenraum $C([a,b])$ sei

$$(f,g) = \int_a^b f(x) g(x) \, \mathrm{d}x, \quad \|f\|_2 = \sqrt{(f,f)}$$

das L_2-Skalarprodukt und die zugehörige Norm. Dann besagt $(*)$, dass das Polynom

$$p(x) := \prod_{j=0}^{n} (x - x_j) = x^{n+1} + r(x), \quad r(x) \in \mathbb{P}_n$$

bezüglich des L_2-Skalarprodukts orthogonal zum Teilraum $\mathbb{P}_n[a,b] \subset C([a,b])$ sein muss.

Zur Konstruktion von $p(x)$ wenden wir das Gram-Schmidt-Orthogonalisierungsverfahren auf die Monombasis $\{1, x, \ldots, x^{n+1}\}$ an. Dann gilt

$$p_0(x) = 1, \quad p_k(x) = x^k - \sum_{j=0}^{k-1} \frac{(x^k, p_j)}{\|p_j\|_2^2} p_j(x), \quad k = 1, \ldots, n+1$$

und $\{p_0, \ldots, p_{n+1}\}$ bildet ein Orthogonalsystem in $\mathbb{P}_{n+1}[a,b]$. Offensichtlich gilt

$$p_{n+1}(x) = x^{n+1} + r(x), \quad r(x) \in \mathbb{P}_n.$$

Setze also $p(x) := p_{n+1}(x)$. Die $n+1$ Nullstellen $\lambda_0, \ldots, \lambda_n$ von $p_{n+1}(x)$ sind dann mögliche Kandidaten für „optimale“ Stützstellen der Integrationsformel.

In Abschnitt 3.4.2 diskutieren wir grundlegende Eigenschaften von Orthogonalpolynomen. Insbesondere wird gezeigt, dass das oben beschriebene Polynom p_{n+1} nur reelle, einfache Nullstellen besitzt, die alle im Inneren des Intervalls $[a,b]$ liegen (siehe Satz 3.19).

Desweiteren werden wir zeigen, dass für $[a,b] = [-1,1]$ die Polynome p_n Vielfache der *Legendre-Polynome* sind. Es gibt verschiedene Darstellungen dieser Polynome:

(i) Rodrigues-Formel:

$$p_n(x) = \frac{n!}{(2n)!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n,$$

wobei hier wie üblich $0! = 1$ gilt.

(ii) Rekursionsformel:

$$p_0(x) = 1, \quad p_1(x) = x, \quad p_{n+1}(x) = xp_n(x) - \frac{n^2}{4n^2-1}p_{n-1}(x), \quad n \geq 1.$$

Die Nullstellen werden für $n > 3$ numerisch bestimmt und können Tabellen entnommen werden. Zum Beispiel gilt

$$p_2(x) = x^2 - \frac{1}{3}, \quad \lambda_0 = -\sqrt{\frac{1}{3}}, \quad \lambda_1 = \sqrt{\frac{1}{3}},$$
$$p_3(x) = x^3 - \frac{3}{5}x, \quad \lambda_0 = -\sqrt{\frac{3}{5}}, \quad \lambda_1 = 0, \quad \lambda_2 = \sqrt{\frac{3}{5}}.$$

Die Nullstellen $\lambda_0, \ldots, \lambda_n$ des $(n+1)$-ten Legendre-Polynoms p_{n+1} können wir als Stützstellen einer interpolatorischen Quadraturformel auf dem Intervall $[-1, 1]$ verwenden und erhalten

$$Q_n(f) = \sum_{i=0}^{n} a_i f(\lambda_i), \quad a_i = \int_{-1}^{1} \prod_{i \neq j}^{n} \frac{x - \lambda_j}{\lambda_i - \lambda_j}\,\mathrm{d}x.$$

Beispiel. Für die Stützstellen $x_0 = -\sqrt{\frac{1}{3}}$, $x_1 = \sqrt{\frac{1}{3}}$ erhalten wir die Gewichte

$$a_0 = \int_{-1}^{1} \frac{x - \sqrt{\frac{1}{3}}}{-\sqrt{\frac{1}{3}} - \sqrt{\frac{1}{3}}}dx = -\frac{\sqrt{3}}{2}\int_{-1}^{1} x - \sqrt{\frac{1}{3}}dx = -\frac{\sqrt{3}}{2}\left(\frac{1}{2}x^2 - \sqrt{\frac{1}{3}}x\right)\Big|_{-1}^{1} = 1$$
$$a_1 = \int_{-1}^{1} \frac{x + \sqrt{\frac{1}{3}}}{\sqrt{\frac{1}{3}} + \sqrt{\frac{1}{3}}}dx = 1.$$

Die Quadraturformel

$$Q_1(f) = f(-\sqrt{\frac{1}{3}}) + f(\sqrt{\frac{1}{3}})$$

integriert auf dem Intervall $[-1, 1]$ Polynome vom Grad drei exakt.

Beispiel. Für die Stützstellen

$$x_0 = -\sqrt{\frac{3}{5}},\ x_1 = 0,\ x_2 = \sqrt{\frac{3}{5}}$$

erhalten wir die Gewichte

$$a_0 = \frac{5}{9},\ a_1 = \frac{8}{9},\ a_2 = \frac{5}{9}.$$

Die Quadraturformel

$$Q_2(f) = \frac{5}{9} f(-\sqrt{\frac{3}{5}}) + \frac{8}{9} f(0) + \frac{5}{9} f(\sqrt{\frac{3}{5}})$$

integriert auf dem Intervall $[-1, 1]$ Polynome vom Grad fünf exakt.

3.4.2 Theorie der Orthogonalpolynome

In diesem Abschnitt betrachten wir Aussagen über Orthogonalpolynome, die für das Verständnis von Quadraturverfahren hilfreich sind.

Satz 3.17. *Es sei $w : (a, b) \to \mathbb{R}$ eine nichtnegative Funktion mit $\int_a^b w(x)dx > 0$.*

Dann gibt es ein eindeutig bestimmtes System von Polynomen p_ν, $\nu = 0, 1, 2 \ldots$, das den folgenden Bedingungen genügt:

(i) $p_\nu \notin \mathbb{P}_{\nu-1}$, $p_\nu \in \mathbb{P}_\nu$

(ii) $\int_a^b p_\nu(x) p_\mu(x) w(x) dx = 0 \quad$ *für* $\nu \neq \mu$

(iii) $\int_a^b p_\nu^2(x) w(x) dx = 1$

(iv) p_ν *hat einen positiven Hauptkoeffizienten.*

Definition 3.18. Das System aus Satz 3.17 heißt Orthogonalpolynomsystem zur Gewichtsfunktion w.

Beweis. (Satz 3.17) Im Raum der Polynome führe man durch

$$(f, g)_w := \int_a^b f(x) g(x) w(x) dx \tag{3.1}$$

ein inneres Produkt ein. Orthogonalisiert man nun (mit Gram-Schmidt) die Folge 1, x, x^2, $x^3, \ldots$, so erhält man ein Funktionensystem des beschriebenen Typs.

Es bleibt zu zeigen, dass p_ν durch die Forderungen (i)-(iv) eindeutig bestimmt ist.

Hätte man nun zwei Systeme p_ν und p_ν^*, die beide (i)-(iv) genügen, so wäre bei passend gewähltem α_ν

$$p_\nu - \alpha_\nu p_\nu^* \in \mathbb{P}_{\nu-1}.$$

Somit wäre wegen (3.1)

$$\begin{aligned} \|p_\nu - \alpha_\nu p_\nu^*\|^2 &= (p_\nu - \alpha_\nu p_\nu^*, p_\nu - \alpha_\nu p_\nu^*)_w \\ &= (p_\nu - \alpha_\nu p_\nu^*, p_\nu)_w - (p_\nu - \alpha_\nu p_\nu^*, \alpha_\nu p_\nu^*)_w \\ &= 0, \end{aligned}$$

also $p_\nu = \alpha_\nu p_\nu^*$.

(iii) zeigt, dass für α_ν nur die Werte ± 1 in Frage kommen, und (iv) zeigt dann schließlich $p_\nu = p_\nu^*$. □

Der folgende Satz macht eine Aussage über Nullstellen von Orthogonalpolynomen.

Satz 3.19. *Die bezüglich des Skalarprodukts $(\cdot,\cdot)_w$ orthogonalen Polynome p_n besitzen lauter reelle, einfache Nullstellen, die alle im Inneren des Intervalls $[a,b]$ liegen.*

Beweis. Es sei $N_n := \{\lambda \in (a,b) \mid \lambda \text{ Nullstelle ungerader Vielfachheit von } p_n\}$. Setze $q(x) = 1$ für $N_n = \emptyset$ und

$$q(x) = \prod_{i=1}^{m}(x - \lambda_i) \quad \text{für} \quad N_n = \{\lambda_1, \ldots, \lambda_m\}.$$

Dann ist $p_n q \in \mathbb{P}_{n+m}$ und $p_n q$ hat in (a,b) keinen Vorzeichenwechsel. Es gilt

$$(p_n, q)_w = \int_a^b p_n(x)q(x)w(x)\,\mathrm{d}x > 0.$$

Für $m < n$ ist dies im Widerspruch zu $p_n \perp \mathbb{P}_{n-1}$. Also gilt $m = n$. □

Der Zusammenhang zur Quadraturverfahren wird durch den folgenden Satz hergestellt.

Satz 3.20. *Es seien $x_0, x_1, \ldots, x_n$ die Nullstellen von p_{n+1}. Es gibt ein System von positiven Zahlen $a_0, \ldots, a_n$ derart, dass für jedes $h \in \mathbb{P}_{2n+1}$ gilt*

$$\int_a^b h(x)w(x)dx = \sum_{i=0}^{n} a_i h(x_i). \tag{3.2}$$

Beweis. Es sei $p = p(h|x_0 \ldots, x_n)$ das Interpolationspolynom von h bezüglich der Knoten $x_0, \ldots, x_n$. Offenbar ist

$$\underbrace{p(h|x_0, \ldots, x_n)}_{\in \mathbb{P}_n} - \underbrace{h}_{\in \mathbb{P}_{2n+1}} = r p_{n+1} \quad r \in \mathbb{P}_n.$$

Wegen (3.1) folgt hieraus

$$\int_a^b p(h|x_0, \ldots, x_n)(x)w(x)dx - \int_a^b h(x)w(x)dx = \int_a^b r(x)p_{n+1}(x)w(x)dx = 0.$$

Schreibt man jetzt das Interpolytionspolynom in der Lagrangeschen Darstellung

$$p(h|x_0,\dots,x_n)(x) = \sum_{i=0}^{n} h(x_i) l_{in}(x),$$

so erkennt man sofort die Gültigkeit von (3.2).

Um noch zu zeigen, dass die $a_i > 0$ sind, setzt man in (3.2) $h = l_{in}^2 \in \mathbb{P}_{2n}$. Dann erhält man

$$a_i = \int_a^b l_{in}^2(x) w(x) dx > 0.$$

□

Der Spezialfall $w = 1$ dieses Satzes liefert ein Interpolationsquadraturverfahren von besonderem Interesse, da man gegenüber eines beliebigen Interpolationsquadraturverfahren eine optimale Graderhöhung erhält.

Der folgende Satz zeigt, dass man Orthogonalpolynome mit Hilfe einer Dreitermrekursionsformel berechnen kann.

Satz 3.21. *Sei $p_0, p_1, \dots$ ein Orthonormalpolynomsystem zur Gewichtsfunktion w. Dann gilt*

$$p_n(x) = (A_n x + B_n) p_{n-1}(x) - C_n p_{n-2}(x) \quad n = 2, 3, \dots$$

mit gewissen Konstanten A_n, B_n, C_n. Ist α_n der Hauptkoeffizient von p_n, so gilt

$$A_n = \frac{\alpha_n}{\alpha_{n-1}}, \quad C_n = \frac{\alpha_n \alpha_{n-2}}{\alpha_{n-1}^2}.$$

Beweis. Mit dem gegebenen A_n ist $p_n - A_n x p_{n-1} \in \mathbb{P}_{n-1}$ und es gilt

$$p_n(x) - A_n x p_{n-1}(x) = \sum_{\nu=0}^{n-1} \beta_\nu p_\nu(x), \quad \beta_i \in \mathbb{R}.$$

Multiplikation mit $p_i(x)w(x)$ und Integration ergibt

$$\int_a^b \left(p_n(x) - A_n x p_{n-1}(x)\right) p_i(x) w(x) dx = \sum_{\nu=0}^{n-1} \beta_\nu \int_a^b p_\nu(x) p_i(x) w(x) dx = \beta_i.$$

Es gilt

$$\beta_i = \underbrace{\int_a^b p_n(x) p_i(x) w(x) dx}_{0 \text{ sofern } i<n} - A_n \underbrace{\int_a^b p_{n-1}(x) \underbrace{x p_i(x)}_{\in \mathbb{P}_{i+1}} w(x) dx}_{0 \text{ sofern } i+1<n-1 (i<n-2)}$$

Also können nur β_{n-1} und β_{n-2} ungleich Null sein.

$$-C_n = \beta_{n-2} = -A_n \int_a^b p_{n-1}(x) \underbrace{x p_{n-2}(x)}_{\alpha_{n-2}x^{n-1}+\ldots=\frac{\alpha_{n-2}}{\alpha_{n-1}}\alpha_{n-1}x^{n-1}+\ldots} w(x)dx$$

$$= -A_n \frac{\alpha_{n-2}}{\alpha_{n-1}} \underbrace{\int_a^b p_{n-1}^2(x)w(x)dx}_{=1}$$

□

Definition 3.22. Das nicht normierte Orthogonalpolynomsystem zur Gewichtsfunktion $w(x) = (1-x)^\alpha(1+x)^\beta$ auf $[-1,1]$ mit $\alpha > -1$, $\beta > -1$, dessen freier Faktor durch $p_n(1) = \binom{n+\alpha}{n}$ festgelegt ist, heißt das System der Jacobi-Polynome. Diese Polynome bezeichnet man mit $P_n^{(\alpha,\beta)}$.

Definition 3.23. Für $\alpha = \beta = 0$ erhält man die Legendre-Polynome, die man mit P_n bezeichnet.

Definition 3.24. Das (nicht normierte) Orthogonalpolynomsystem zur Gewichtsfunktion $w(x) = (1-x^2)^{-\frac{1}{2}}$ auf $[-1,1]$, dessen freier Faktor durch $p_n(1) = 1$ festgelegt ist, heißt das System der Tschebyscheff-Polynome erster Art.

Die für die Numerik wichtigsten Orthogonalpolynomsysteme sind die Legendre und die Tschebyscheff Polynome.

Satz 3.25. *(Formel von Rodrigues)*

$$P_n^{(\alpha,\beta)}(x) = (1-x)^{-\alpha}(1+x)^{-\beta}\frac{(-1)^n}{2^n n!}\frac{d^n}{dx^n}\left((1-x)^{n+\alpha}(1+x)^{n+\beta}\right)$$

für $x \in (-1,1)$.

Wir benutzen die Leibnizsche Formel

$$D^n(f\cdot g) = \sum_{\nu=0}^{n}\binom{n}{\nu}D^\nu(f)D^{n-\nu}(g) \quad (D = \frac{d}{dx}).$$

Außerdem gilt:

$$D^\nu\left((1-x)^{n+\alpha}\right) = (-1)^\nu\binom{n+\alpha}{\nu}\nu!(1-x)^{n+\alpha-\nu}$$

Beweis. (Satz 3.25)

$$
\begin{aligned}
&(1-x)^{-\alpha}(1+x)^{-\beta}D^n\left((1-x)^{n+\alpha}(1+x)^{n+\beta}\right)\\
&=\sum_{\nu=0}^{n}\binom{n}{\nu}(-1)^{\nu}\binom{n+\alpha}{\nu}\nu!(1-x)^{n-\nu}\binom{n+\beta}{n-\nu}(n-\nu)!(1+x)^{\nu}\quad (Leibniz)\\
&=\sum_{\nu=0}^{n}\binom{n+\alpha}{\nu}\binom{n+\beta}{n-\nu}(1-x)^{n-\nu}(1+x)^{\nu}n!(-1)^{\nu}
\end{aligned}
$$

Einsetzen von $x=1$ liefert:

$$\binom{n+\alpha}{n}2^n n!(-1)^n$$

Es gilt also: $P_n^{(\alpha,\beta)}$ ist ein Polynom n-ten Grades, das bei $x=1$ den Wert $\binom{n+\alpha}{n}$ annimmt.

Es bleibt zu beweisen, dass dieses Polynom zu $p\in\mathbb{P}_{n-1}$ orthogonal ist, d.h.

$$\int_{-1}^{1}D^n\left((1-x)^{n+\alpha}(1+x)^{n+\beta}\right)p(x)dx=0.$$

Dies zeigt man durch n-fache partielle Integration. Dabei verschwinden alle ausintegrierten Terme, da $D^{\lambda}\left((1-x)^{n+\alpha}(1+x)^{n+\beta}\right)$ für $\lambda=0,\dots,n-1$ bei ±1 eine Nullstelle hat. □

Aus dem Beweis kann man auch ablesen, dass $P_n^{(\alpha,\beta)}$ den Hauptkoeffizienten

$$\frac{1}{2^n}\sum_{\nu=0}^{n}\binom{n+\alpha}{\nu}\binom{n+\beta}{n-\nu}=\frac{1}{2^n}\binom{2n+\alpha+\beta}{n}$$

besitzt. Die letzte Gleichung folgt aus der Vandermonde-Identität

$$\binom{x+y}{n}=\sum_{\nu=0}^{n}\binom{x}{\nu}\cdot\binom{y}{n-\nu}\quad\forall x,y\in\mathbb{R}$$

Für die Legendre-Polynome (d.h. $\alpha=\beta=0$) kann man den Normierungsfaktor elementar berechnen und erhält

$$\int_{-1}^{1}(P_n(x))^2dx=\frac{2}{2n+1}.$$

Die normierten Orthogonalpolynome zur Gewichtsfunktion 1 haben also die Form $\sqrt{\frac{2n+1}{2}}P_n$.

Für die Legendre-Polynome lautet die Formel von Rodrigues

$$P_n(x)=\frac{1}{2^n n!}\frac{d^n}{dx^n}\left((x^2-1)^n\right).$$

Beachten Sie den unterschiedlichen Faktor im Vergleich zur Formel auf Seite 75. Dieser kommt dadurch zustande, dass dort Polynome mit Haupkoeffizient 1 betrachtet wurden.

Mit Hilfe des Hauptkoeffizienten lässt sich nun auch die Rekursionsformel aus Satz 3.21 für die Legendre-Polynome explizit angeben. Sie lautet

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x).$$

3.4.3 Gaußsche Quadraturformeln

Nun können wir das Hauptresultat über Gaußsche Quadraturformeln angeben.

Satz 3.26 (Gaußsche Quadraturformel)**.** *Es gibt genau eine interpolatorische Quadraturformel zu $n+1$ paarweise verschiedenen Stützstellen über dem Intervall $[-1,1]$ mit Exaktheitsgrad $2n+1$ (das heißt exakt für alle Polynome $p \in \mathbb{P}_{2n+1}$). Ihre Stützstellen sind gerade die Nullstellen $\lambda_0, \ldots, \lambda_n \in [-1,1]$ des $(n+1)$-ten Legendre-Polynoms $P_{n+1} \in \mathbb{P}_{n+1}$ und ihre Gewichte genügen der Beziehung $a_i > 0$ für $i = 0, \ldots, n$. Für $f \in C^{2n+2}([-1,1])$ besitzt das Restglied die Darstellung*

$$R_n(f) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_{-1}^{1} \prod_{j=0}^{n} (x-\lambda_j)^2 \,\mathrm{d}x, \quad \xi \in (-1,1).$$

Beweis. (i) Zunächst ist p_{n+1} orthogonal zu $\mathbb{P}_n[-1,1]$ und hat die Nullstellen $\lambda_0, \ldots, \lambda_n \in (-1,1)$. Es gilt also

$$p_{n+1}(x) = \prod_{i=0}^{n} (x-\lambda_i) = x^{n+1} + \ldots$$

Aufgrund der obigen Betrachtungen hat die zugehörige interpolatorische Quadraturformel den Exaktheitsgrad $2n+1$. Die typische Berechnung der Gewichte einer interpolatorischen Quadraturformel liefert

$$a_j = \int_{-1}^{1} l_{jn}(x) \,\mathrm{d}x, \quad l_{jn}(x) = \prod_{i=0, i\neq j}^{n} \frac{x-\lambda_i}{\lambda_j - \lambda_i}, \quad j = 0, \ldots n.$$

Es gilt $l_{jn} \in \mathbb{P}_n$ und damit $l_{jn}^2 \in \mathbb{P}_{2n}$. Also wird auch l_{jn}^2 von der Quadraturformel exakt integriert und es folgt

$$0 < \int_{-1}^{1} (l_{jn}(x))^2 \,\mathrm{d}x = \sum_{i=0}^{n} a_i l_{jn}^2(\lambda_i) = a_j,$$

da $l_{jn}^2(\lambda_i) = \delta_{ij}$. Dies liefert $a_j > 0$ für alle $j = 0, \ldots, n$.

(ii) Um die Eindeutigkeit zu zeigen, nehmen wir an, dass es eine zweite Quadraturformel

$$\tilde{Q}_n(f) = \sum_{i=0}^{n} \tilde{a}_i f(\tilde{\lambda}_i)$$

gibt, die exakt für Polynome $p \in \mathbb{P}_{2n+1}$ ist. Mit den analog gebildeten Polynomen $\tilde{l}_{jn} \in \mathbb{P}_n$ folgt dann wie oben $\tilde{a}_i > 0, i = 0, \ldots, n$. Mit der Orthogonalität von p_{n+1} zu $\mathbb{P}_n[-1, 1]$ folgt damit

$$0 = \int_{-1}^{1} \frac{1}{\tilde{a}_i} \tilde{l}_{in}(x) p_{n+1}(x)\, \mathrm{d}x = \sum_{k=0}^{n} \frac{\tilde{a}_k}{\tilde{a}_i} \tilde{l}_{in}(\tilde{\lambda}_k) p_{n+1}(\tilde{\lambda}_k) = p_{n+1}(\tilde{\lambda}_i).$$

Also sind auch $\tilde{\lambda}_i$ für $i = 0, \ldots, n$ Nullstellen von p_{n+1} und wegen der eindeutigen Bestimmtheit dieser gilt füar $i = 0, \ldots, n$ folglich $\lambda_i = \tilde{\lambda}_i$ und somit auch $a_i = \tilde{a}_i$, womit die Eindeutigkeit gezeigt ist.

(iii) Schließlich bleibt die Fehlerabschätzung der Gauß-Quadraturformel zu zeigen. Dazu benutzen wir ein Hilfsresultat der Hermite-Interpolation.

gegeben:	Knoten	$x_0 < x_1 < \cdots < x_m,$	$m \geq 0,$
	Werte	$y_i^{(k)} \in \mathbb{R},$	$i = 0, \ldots, m, k = 0, \ldots, \mu_i,$
			$\mu_i \geq 0$ für $0 \leq i \leq m,$
gesucht:	$p \in \mathbb{P}_n, n = m + \sum_{i=0}^{m} \mu_i,$	mit $p^{(k)}(x_i) = y_i^{(k)},$	$i = 0, \ldots, m, k = 0, \ldots, \mu_i.$

Nach Satz 2.18 besitzt die Hermite-Interpolationsaufgabe eine eindeutige Lösung und nach Satz 2.19 gilt für den Fehler der Hermite-Interpolation für eine Funktion $f \in C^{n+1}([x_0, x_m]$ die Gleichung

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi(x)) \prod_{i=0}^{m} (x - x_i)^{\mu_i + 1}, \quad \xi(x) \in [x_0, x_m].$$

Also gibt es ein Polynom $h \in \mathbb{P}_{2n+1}$, das die Hermite-Interpolationsaufgabe

$$h(\lambda_i) = f(\lambda_i), \quad h'(\lambda_i) = f'(\lambda_i), \quad i = 0, \ldots, n$$

löst und für $f \in C^{2n+2}([-1, 1])$ die Restglieddarstellung

$$f(x) - h(x) = \frac{1}{(2n+2)!} f^{(2n+2)}(\xi(x)) \prod_{i=0}^{n} (x - \lambda_i)^2, \quad \xi(x) \in [-1, 1]$$

hat. Anwendung der Gaußschen Quadraturformel auf $h(x)$ ergibt wegen $Q_n(h) = I(h)$ dann

$$\begin{aligned} R_n(f) &= I(f) - Q_n(f) = I(f-h) - Q_n(f-h) \\ &= \int_{-1}^{1} \frac{1}{(2n+2)!} f^{(2n+2)}(\xi(x)) \prod_{i=0}^{n} (x-\lambda_i)^2 \,\mathrm{d}x - \sum_{i=0}^{n} a_i (f(\lambda_i) - h(\lambda_i)) \\ &= \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_{-1}^{1} \prod_{i=0}^{n} (x-\lambda_i)^2 \,\mathrm{d}x. \end{aligned}$$

Im letzten Schritt wurde der Mittelwertsatzes der Integralrechnung genutzt. □

Ohne Beweis geben wir nun noch zwei Sätze zu verwandten Interpolationsquadraturverfahren an. Die Beweise findet man in H. Braß, Quadraturverfahren, 1977.

Satz 3.27. *Es gibt auf dem Grundintervall $[-1, 1]$ genau eine interpolatorische Quadraturformel zu $n+1$ paarweise verschiedenen Stützstellen und $x_n = 1$ mit Exaktheitsgrad $2n$ (d.h. exakt für alle Polynome $p \in \mathbb{P}_{2n}$). Es ist diejenige Interpolationsquadraturformel, bei der die Stützstellen die Nullstellen von $P_{n+1} - P_n$ sind.*

Definition 3.28. Das entsprechend konstruierte Interpolationsquadraturverfahren heißt Radau-Verfahren.

Satz 3.29. *Es gibt auf dem Grundintervall $[-1, 1]$ genau eine interpolatorische Quadraturformel zu $n+1$ paarweise verschiedenen Stützstellen und $x_0 = -1, x_n = 1$ mit Exaktheitsgrad $2n-1$ (d.h. exakt für alle Polynome $p \in \mathbb{P}_{2n-1}$). Es ist diejenige Interpolationsquadraturformel, bei der die Stützstellen die Nullstellen von $P_{n+1} - P_{n-1}$ sind.*

Definition 3.30. Das entsprechend konstruierte Interpolationsquadraturverfahren heißt Lobatto-Verfahren.

3.4.4 Berechnung der Knoten und Gewichte von Gauss Quadraturformeln

Wir betrachten nun noch eine Methode, mit der sich die Gewichte und Stützstellen von Gaußschen Quadraturformeln numerisch stabil berechnen lassen.

Der Ausgangspunkt zur Beschreibung dieser Methode ist der folgende Satz.

Satz 3.31. *Das Legendre-Polynom $P_m(x)$, $m \geq 1$ lässt sich als die folgende Determinante*

berechnen:

$$P_m(x) = \begin{vmatrix} \tilde{a}_1 x & b_1 & & & & \\ b_1 & \tilde{a}_2 x & b_2 & & & \\ & b_2 & \tilde{a}_3 x & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{m-2} & \tilde{a}_{m-1} x & b_{m-1} \\ & & & & b_{m-1} & \tilde{a}_m x \end{vmatrix}$$

mit $\tilde{a}_k := \frac{2k-1}{k}$, $k = 1, \ldots, m$ *und* $b_k := \sqrt{\frac{k}{k+1}}$, $k = 1, \ldots, m-1$.

Beweis. Die Entwicklung der Determinante nach der letzten Zeile liefert

$$P_m(x) = \tilde{a}_m x P_{m-1}(x) - b_{m-1}^2 P_{m-2}(x), \quad m \geq 3.$$

Nach Indexverschiebung erhalten wir

$$P_{m+1}(x) = \tilde{a}_{m+1} x P_m(x) - b_m^2 P_{m-1}(x), \quad m \geq 2$$

mit $\tilde{a}_{m+1} = \frac{2m+1}{m+1}$ und $b_m^2 = \frac{m}{m+1}$, also

$$(m+1)P_{m+1}(x) = (2m+1)xP_m(x) - mP_{m-1}(x).$$

Das ist gerade die Dreitermrekursionsformel der Legendre-Ploynome. Mit $P_0(x) = 1$ gilt der Satz auch für $m = 1$. □

Wir dividieren nun für $k = 1, \ldots, m$ die k-te Zeile und Spalte der Determinante durch $\sqrt{\tilde{a}_k}$ und kompensieren diese Umformungen durch entsprechende Faktoren. Wir erhalten

$$P_m(x) = \prod_{k=1}^{m} \tilde{a}_k \cdot \begin{vmatrix} x & c_1 & & & & \\ c_1 & x & c_2 & & & \\ & c_2 & x & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{m-2} & x & c_{m-1} \\ & & & & c_{m-1} & x \end{vmatrix}$$

mit $c_k = \frac{b_k}{\sqrt{\tilde{a}_k \tilde{a}_{k+1}}} = \frac{k}{\sqrt{4k^2-1}}$.

Die Nullstellen des m-ten Legendre-Polynoms sind die Eigenwerte der Tridiagonalmatrix

$$P_m := \begin{pmatrix} 0 & c_1 & & & & \\ c_1 & 0 & c_2 & & & \\ & c_2 & 0 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{m-2} & 0 & c_{m-1} \\ & & & & c_{m-1} & 0 \end{pmatrix},$$

Auch die Gewichte der Gauß Quadraturformel lassen sich mit Hilfe der Matrix P_m berechnen. Sei $P_m = \tilde{V}\Lambda\tilde{V}^T$, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_m)$, und $\tilde{V} = (\tilde{v}^{(1)}|\tilde{v}^{(2)}|\ldots|\tilde{v}^{(m)})$ die (orthogonale) Matrix der rechten Eigenvektoren. Der Eigenvektor von P_m zum Eigenwert $\lambda_{k+1} = x_k$ (für $k = 0, \ldots, m-1$) lässt sich schreiben als (siehe Übung)

$$v^{(k)} = \begin{pmatrix} d_0\sqrt{\tilde{a}_1}P_0(x_k) \\ d_1\sqrt{\tilde{a}_2}P_1(x_k) \\ \vdots \\ d_{m-1}\sqrt{\tilde{a}_m}P_{m-1}(x_k) \end{pmatrix}$$

mit $d_0 := 1$, $d_j := \frac{1}{\prod_{i=1}^{j} b_i}$, $j = 1, \ldots, m-1$ so dass $v_1^{(k)} = 1$. Setze $V = (v^{(1)}|v^{(2)}|\ldots|v^{(m)})$.

Die Legendre-Polynome $P_0, P_1, \ldots, P_{m-1}$ werden durch die Gaußsche Quadraturformel mit m Stützstellen exakt integriert. Unter Ausnutzung der Orthogonalitätseigenschaft erhalten wir

$$\int_{-1}^{1} P_0(x)P_i(x)dx = \int_{-1}^{1} P_i(x)dx = \sum_{k=1}^{m} a_k P_i(x_k) = \begin{cases} 2 & : \quad i = 0 \\ 0 & : \quad i = 1, \ldots, m-1. \end{cases}$$

Das ist ein lineares Gleichungssystem zur Berechnung der Gewichte $a = (a_0, \ldots, a_{m-1})^T$. Wir multiplizieren nun noch die j-te Zeile des Gleichungssystems mit $d_{j-1}\sqrt{\tilde{a}_j}$, $j = 1, \ldots, m$.

Dann hat das resultierende lineare Gleichungssystem die Form

$$Va = 2e_1,$$

d.h. die Koeffizientenmatrix stimmt mit der oben angegebenen orthogonalen Matrix der Eigenvektoren überein. Wir multiplizieren nun die Gleichung von links mit $(v^{(k)})^T$ und erhalten

$$(v^{(k)})^T v^{(k)} a_k = 2(v^{(k)})^T e_1 = 2v_1^{(k)} = 2.$$

Bezeichnen wir wie oben die normierten Eigenvektoren mit $\tilde{v}^{(k)}$, dann gilt

$$a_k = 2(\tilde{v}_1^{(k)})^2, \quad k = 1, \ldots, m.$$

Wir fassen nun noch einmal die Berechnung der Stützstellen und Gewischte in der für Quadraturformeln üblichen Indizierung zusammen.

Stützstellen und Gewichte der Gauß Quadraturformel: Die Knoten $x_0, \ldots, x_n$ der Gauß-Quadraturformel, also die Nullstellen der Legendre-Polynome, sowie die zugehörigen Gewichte, können durch die Lösung eines tridiagonalen Eigenwertproblems

berechnet werden. Die Nullstellen des $n+1$-ten Legendre-Polynoms sind Eigenwerte von

$$P_{n+1} = \begin{pmatrix} 0 & c_1 & & & & \\ c_1 & 0 & c_2 & & & \\ & c_2 & 0 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-1} & 0 & c_n \\ & & & & c_n & 0 \end{pmatrix}, \quad c_k = \frac{k}{\sqrt{4k^2-1}}, \quad k = 1, \ldots, n.$$

Es sei $P_{n+1} = VDV^T$ eine orthogonale Diagonalisierung von P_{n+1} mit $V = [v^{(1)} \cdots v^{(n+1)}]$ und $D = \mathrm{diag}(x_0, \ldots, x_n)$. Die Gewichte $a_0, \ldots, a_n$ der Gauß-Quadraturformel ergeben sich aus den Gleichungen

$$a_k = 2(\tilde{v}^{(k+1)})_1^2, \quad k = 0, \ldots, n.$$

Hier ist $(\tilde{v}^{(k+1)})_1$ die erste Komponente des normierten Eigenvektors von P_{n+1} zum Eigenwert x_k, $k = 0, \ldots, n$.

Testrechnungen: Wir wenden nun die Gaußschen Quadraturformeln auf drei Testprobleme an. In allen drei Testproblemen können wir den exakten Wert des Integrals leicht analytisch berechnen. In den Abbildungen 3.3 stellen wir den Fehler $|Q_n(f) - I(f)|$ als Funktion von n graphisch dar.

In Abbildung 3.3 (a) betrachten wir die Berechnung von

$$\int_{-1}^{1} x^{20} dx = \frac{2}{21}.$$

Für $n \geq 10$ liefert die Gauß QF, wie erwartet, die exakte Lösung.

In Abbildung 3.3 (b) betrachten wir die Berechnung von

$$\int_{-1}^{1} e^x dx = e^1 - e^{-1}.$$

Hier liefert die Gauß QF bereits für $n = 5$ sehr genaue Resultate.

In Abbildung 3.3 (c) betrachten wir die Berechnung von

$$\int_{-1}^{1} |x|^3 dx = \frac{1}{2}.$$

Der Fehler verringert sich mit zunehmendem n. Allerdings nicht so schnell wie in den ersten beiden Testproblemen.

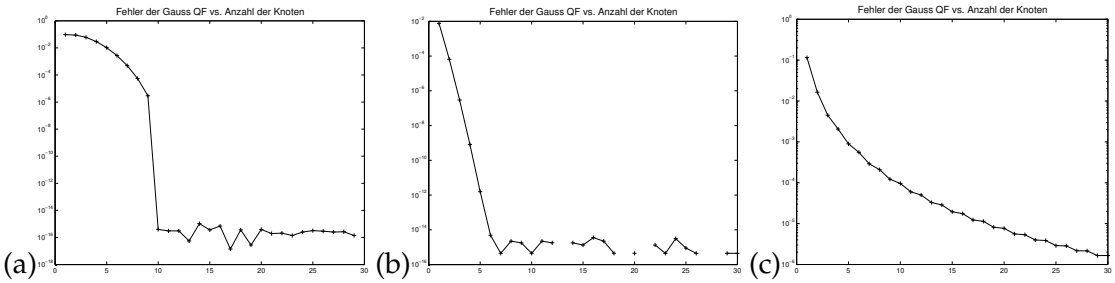


Abbildung 3.3: Fehler der Gauß-Quadratur vs. n für die Integration von a) $f(x) = x^{20}$, b) $f(x) = e^x$ und c) $f(x) = |x|^3$.

4 Direkte Lösung linearer Gleichungssysteme

Die numerischen Lösung linearer Gleichungssysteme spielt eine zentrale Rolle innerhalb der Numerik.

In diesem Kapitel beschäftigen wir uns mit direkten Lösungsverfahren für lineare Gleichungssysteme. Dazu zählen die LR-, Cholesky- und die QR-Zerlegung. Direkte Verfahren ermitteln, bei Vernachlässigung von Rundungsfehlern, die exakte Lösung des linearen Gleichungssystems in endlich vielen Schritten.

Bei den direkten Verfahren setzen wir zunächst voraus, dass die Koeffizientenmatrix regulär ist. Ist die Koeffizientenmatrix eines linearen Gleichungssystems nicht regulär oder liegt ein über- oder unterbestimmtes System vor, bei dem die Koeffizientenmatrix nicht quadratisch ist, so verwendet man in der Praxis die Methode der kleinsten Quadrate. Auch mit dieser Methode werden wir uns beschäftigen.

Ein weiterer Spezialfall sind sehr große Gleichungssysteme, deren Koeffizientenmatrix viele Nullen enthält. Solche Matrizen nennt man schwach besetzt. Für derartige Gleichungssysteme verwendet man in der Praxis typischerweise iterative Verfahren, wie sie in der Numerik 2 behandelt.

Zu Vereinfachung der Notation beschränken wir uns in dieser Vorlesung auf reellwertige Matrizen. Die Betrachtungen lassen sich auf komplexwertige Matrizen übertragen.

4.1 Das Gaußsche Eliminationsverfahren und die LR-Zerlegung

Gegeben seien eine Matrix $A \in \mathbb{R}^{n\times n}$ und ein Vektor $b \in \mathbb{R}^n$. Wir suchen einen Vektor $x \in \mathbb{R}^n$ mit $Ax = b$. Aus der linearen Algebra ist bekannt:

Satz 4.1. *Es sei $A \in \mathbb{R}^{n\times n}$ mit $\det A \neq 0$ und $b \in \mathbb{R}^n$. Dann existiert genau ein $x \in \mathbb{R}^n$ mit $Ax = b$.*

Es sei also $\det A \neq 0$. Dann gibt es folgende Situationen, in denen wir das lineare Gleichungssystem einfach lösen können.

1) A hat Diagonalform, das heißt $a_{ij} = 0$ für alle $i \neq j$. Wegen $\det A \neq 0$ ist $a_{ii} \neq 0$ für alle i. Aus $Ax = b$ folgt $a_{ii}x_i = b_i$ und damit

$$x_i = \frac{b_i}{a_{ii}} \quad \text{für alle} \quad i.$$

2) A ist obere Dreiecksmatrix, das heißt $a_{ij} = 0$ für alle $i > j$. Auch hier folgt aus $\det A \neq 0$, dass $a_{ii} \neq 0$ für alle i. Zu lösen ist das gestaffelte System

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{nn}x_n &= b_n. \end{aligned}$$

Analog für untere Dreiecksmatrizen.

Algorithmus 4.2.

- Berechne zunächst
$$x_n = \frac{b_n}{a_{nn}}.$$
- Für $i = n-1, \ldots, 1$ berechne
$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}.$$

Listing 5: Python Routinen

```
#Löse Rx = b
def SolveR(R,b):
    import numpy as np

    n = len(b)
    x = np.zeros(n)

    x[-1] = b[-1]/R[-1,-1]
    for k in range(n-2,-1,-1):
        x[k] = (b[k]-np.sum(R[k,k+1:]*x[k+1:]))/R[k,k]
    return x
```

```
# Löse Lx = b
def SolveL(L,b):
    import numpy as np

    n = len(b)
    x = np.zeros(n)

    x[0] = b[0]/L[0,0]
    for k in range(1,n):
        x[k] = (b[k]-np.sum(L[k,:k]*x[:k]))/L[k,k]
    return x
```

Lemma 4.3. *Es sei A eine nichtsinguläre obere bzw. untere Dreiecksmatrix. Dann ist A^{-1} ebenfalls eine obere bzw. untere Dreiecksmatrix.*

Beweis. Es sei A eine nichtsinguläre obere Dreiecksmatrix. Weiter sei $x^{(k)}$ die eindeutige Lösung von $Ax^{(x)} = e_k$, wobei e_k der k-te Einheitsvektor ist. Dann gilt $x^k_{k+1} = \cdots = x^{(k)}_n = 0$ nach obiger Formel. Es folgt $A^{-1} = \left(x^{(1)}|x^{(2)}|...|x^{(n)}\right)$ und damit ist A^{-1} eine obere Dreiecksmatrix.

Analog für untere Dreiecksmatrizen. □

3) Es sei $A = LR$ mit einer unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix R. Dann ist $Ax = b$ gleichbedeutend mit $LRx = b$.

(i) Löse $Ly = b$ wie in Fall 2.

(ii) Löse $Rx = y$ wie in Fall 2.

Bemerkung. Wegen $\det A \neq 0$ gilt nach dem Multiplikationstheorem für Determinanten auch $\det L \neq 0$ und $\det R \neq 0$. Ohne Einschränkung sei $L \in \mathbb{R}^{n\times n}$ so normiert, dass $l_{ii} = 1$ gilt für alle i.

Definition 4.4. Die Zerlegung einer Matrix $A \in \mathbb{R}^{n\times n}$ in ein Produkt $A = LR$ aus einer linken unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und einer rechten oberen Dreiecksmatrix $R \in \mathbb{R}^{n\times n}$, heißt LR-Zerlegung.

4.1.1 Bestimmung der LR-Zerlegung

Bemerkung. Nicht jede nichtsinguläre $(n \times n)$-Matrix besitzt eine LR-Zerlegung, denn es gilt beispielsweise

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \neq \begin{pmatrix} * & 0 \\ * & * \end{pmatrix} \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}.$$

Aus $l_{11}r_{11} = 0$ folgt $\det L = 0$ oder $\det R = 0$. Dies ist ein Widerspruch. Allerdings existiert eine Permutationsmatrix P, so dass PA eine LR-Zerlegung besitzt.

Im Folgenden nehmen wir an, dass alle auftretenden Divisionen durchgeführt werden können, das heißt, es tritt keine Division durch Null auf. Das lineare Gleichungssystem lautet

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + \cdots + a_{2n}x_n &= b_2, \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= b_n. \end{aligned}$$

Wir führen auch noch die folgende Definition ein:

Definition 4.5. Eine Matrix, deren Diagonale ausschließlich Einsen aufweist und die zusätzlich nur in einer Spalte unterhalb der Diagonale Werte ungleich null besitzt, wird als *Frobenius-Matrix* bezeichnet.

1. Schritt: Wir elimineren x_1 aus den Zeilen $2, ..., n$. Dazu substrahiere wir das $\frac{a_{i1}}{a_{11}}$-fache der 1. Zeile von der Zeile i und erhalten

$$0x_1 + \left(a_{i2} - \frac{a_{i1}}{a_{11}} * a_{12}\right) x_2 + ... + \left(a_{in} - \frac{a_{i1}}{a_{11}} a_{1n}\right) x_n = b_i - \frac{a_{i1}}{a_{11}} b_1.$$

Wir setzen dann

$$A^{(2)} = \left(\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline 0 & & & \\ \vdots & & \tilde{A} & \\ 0 & & & \end{array}\right)$$

mit $\tilde{A} \in \mathbb{R}^{(n-1)\times(n-1)}$, wobei $\tilde{A}$ die gerade neu berechneten Elemente enthält.

Dann ist $Ax = b$ äquivalent zu $A^{(2)}x = b^{(2)}$ mit $b^{(2)} = L_1 b$, wobei

$$L_1 = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & & & 1 \end{pmatrix}.$$

Hierbei ist $l_{i1} := -\frac{a_{i1}}{a_{11}}$ für $i = 2, \ldots, n$.

Insgesamt erhalten wir mit $A^{(1)} := A$ und $b^{(1)} := b$ die Beziehungen $A^{(2)} = L_1 A^{(1)}$ und $b^{(2)} = L_1 b^{(1)}$.

2. Schritt: Gehe genauso mit $\tilde{A}$ vor. Das heißt wir wählen $\tilde{L} \in \mathbb{R}^{(n-1)\times(n-1)}$ so, dass

$$\tilde{L}\tilde{A} = \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & * & \\ 0 & & & \end{array}\right)$$

gilt. Wir setzen dann

$$L_2 = \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \tilde{L} & \\ 0 & & & \end{array}\right) = \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & 1 & & \\ \vdots & l_{32} & & \\ & \vdots & & \\ 0 & l_{n2} & & 1 \end{array}\right)$$

mit $l_{i2} := -\frac{a_{i2}^{(2)}}{a_{22}^{(2)}}$ für $i = 3, \ldots, n$.

Jetzt ist $Ax = b$ gleichbedeutend mit $A^{(3)}x = b^{(3)}$ mit $A^{(3)} = L_2A^{(2)} = L_2L_1A^{(1)}$ und $b^{(3)} = L_2b^{(2)}$. Führen wir diese Idee nun weiter fort, so erhalten wir schließlich

$$A^{(n)} := L_{n-1}A^{(n-1)} = \begin{pmatrix} a_{11} & & \cdots & a_{1n} \\ & * & \cdots & * \\ & & \ddots & \vdots \\ & & & * \end{pmatrix} =: R$$

Es gilt

$$\begin{aligned} R &= L_{n-1}A^{(n-1)} \\ &= L_{n-1}L_{n-2}A^{(n-2)} \\ &\vdots \\ &= L_{n-1}L_{n-2}\cdots L_1A^{(1)}. \end{aligned}$$

Also ist $Ax = b$ gleichbedeutend mit $Rx = b^{(n)}$ und wir können dieses System nach Fall 2 lösen. Setze $L = (L_{n-1}L_{n-2}\cdots L_1)^{-1} = L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}$. Dann ist L eine untere Dreiecksmatrix nach Lemma 4.3.

Lemma 4.6. *Es gilt:*

1)

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

2)

$$L = (L_{n-1}L_{n-2}\dots L_1)^{-1} = \begin{pmatrix} 1 & & & \\ -l_{2,1} & \ddots & & \\ \vdots & \ddots & \ddots & \\ -l_{n,1} & \cdots & -l_{n,n-1} & 1 \end{pmatrix}.$$

Beweis. Nachrechnen (Übung, siehe auch VL13). □

4.1.2 Pivotisierung

Partielle Pivotisierung bei $\det A \neq 0$.

Bisher haben wir angenommen, dass alle Divisionen durchgeführt werden können, das heißt eine Division durch Null tritt nicht auf. Jetzt wollen wir geeignete Maßnahmen finden, falls solche Fälle doch auftreten.

Man kann zeigen, dass die Gauß-Eliminierung auf Rundungsfehler nicht so empfindlich reagiert, falls $|l_{ij}| \leq 1$ gilt (siehe Abschnitt 4.3). Daher empfiehlt sich das folgende Vorgehen:

1) Es sei $a_{11} = 0$. Wegen $\det A \neq 0$ existiert ein $a_{i1} \neq 0$ für $i \in \{2, \dots, n\}$. Allgemein (auch für $a_{11} \neq 0$) bestimmen wir $l \in \{1, \dots, n\}$ mit $|a_{l1}| \geq |a_{i1}|$ für alle i.

Wir vertauschen nun Zeile 1 mit Zeile l. Dies entspricht einer Linksmultiplikation mit der Permutationsmatrix $P_1 = (e_l, e_2, \dots, e_{l-1}, e_1, e_{l+1}, \dots, e_n)$. Mit $A^{(1)} := A$ erhalten wir dann $A^{(2)} = L_1 P_1 A^{(1)}$. Das neue a_{11} heißt *Pivotelement*.

2) Es sei

$$A^{(k)} = L_{k-1}P_{k-1}\cdots L_1 P_1 A^{(1)} = \left(\begin{array}{ccc|c} a_{11}^{(k)} & \cdots & * & * \\ & \ddots & \vdots & \vdots \\ & & a_{k-1,k-1}^{(k)} & * \\ \hline 0 & \cdots & 0 & \tilde{A} \end{array}\right)$$

mit

$$\tilde{A} = \begin{pmatrix} a_{kk}^{(k)} & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \end{pmatrix}.$$

Es gilt $0 \neq \det A = \det A^{(k)} = a_{11}^{(k)} \cdots a_{k-1,k-1}^{(k)} \det \tilde{A}$. Also gibt es ein $a_{ik}^{(k)} \neq 0$ für $i \in \{k, \dots n\}$. Es sei $l \in \{k, \dots, n\}$ mit $|a_{lk}^{(k)}| \geq |a_{ik}^{(k)}|$ für $i \in \{k, \dots, n\}$. Nun vertauschen wir die Zeilen k und l und es ergibt sich P_k.

3) Es gilt $R = A^{(n)} = L_{n-1}P_{n-1} \cdots L_2P_2L_1P_1A^{(1)}$. Unser Ziel ist es nun, alle Permutationsmatrizen P_j nach rechts zu verschieben. Dazu betrachten wir P_jL_i mit $j > i$. Dabei vertausche P_j die Zeilen l und m mit $i < j \leq l < m$. Es gilt

$$P_jL_i = \left(\begin{array}{ccc|c} 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1_{(i-1,i-1)} & 0 \\ \hline 0 & \cdots & 0 & \tilde{P} \end{array}\right) \left(\begin{array}{ccc|c} 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1_{(i-1,i-1)} & 0 \\ \hline 0 & \cdots & 0 & \tilde{L} \end{array}\right)$$

mit

$$\tilde{P} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & \cdots & 1 & & \\ & & \vdots & \ddots & \vdots & & \\ & & 1 & \cdots & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \quad \tilde{L} = \begin{pmatrix} 1 & & & \\ * & 1 & & \\ \vdots & & \ddots & \\ * & & & 1 \end{pmatrix}.$$

Es ist leicht einzusehen, dass eine Linksmultiplikation mit der Permutationsmatrix die Zeilen vertauscht und eine Rechtsmultiplikation mit dieser Permutationsmatrix vertauscht die Spalten. Insgesamt gilt

Lemma 4.7. *Es sei P_j eine $(n \times n)$-Permutationsmatrix, die zwei Zeilen $l, m \in \{j, \dots, n\}$ vertauscht und es sei L_i eine Frobenius-Matrix aus dem Gaußschen Eliminationsprozess mit $i < j$. Dann gibt es eine Frobenius-Matrix L_i^* mit*

$$P_jL_i = L_i^*P_j.$$

Die Matrizen L_i und L_i^* unterscheiden sich höchstens dadurch, dass die Elemente an den Positionen (l, i) und (m, i) vertauscht sind.

Das Lemma liefert uns nun

$$\begin{aligned} R = A^{(n)} &= L_{n-1}P_{n-1} \cdots L_2P_2L_1P_1A^{(1)} \\ &= L_{n-1}\hat{L}_{n-2} \cdots \hat{L}_1P_{n-1} \cdots P_1A^{(1)}, \end{aligned}$$

mit

$$\hat{L}_i := P_{n-1}P_{n-2}\dots P_{i+1}L_iP_{i+1}^{-1}\dots P_{n-2}^{-1}P_{n-1}^{-1}.$$

Die Matrix $\hat{L}_i$ erhält man (gemäß Lemma 4.7) durch mehrfachen Tausch innerhalb der i-ten Spalte.

Somit gibt es eine Permutationsmatrix $P = P_{n-1}\cdots P_1$, so dass PA eine LR-Zerlegung besitzt mit $L^{-1} = L_{n-1}\hat{L}_{n-2}\cdots\hat{L}_1$.

Zum genaueren Verständnis der Berechnung von $\hat{L}_i$ betrachten wir ein Beispiel. Angenommen wir haben für eine 4×4 Matrix A eine Zerlegung der Form

$$L_3P_3L_2P_2L_1P_1A = R$$

berechnet. Setze

$$\hat{L}_2 := P_3L_2\underbrace{P_3^{-1}}_{P_3}, \quad \hat{L}_1 := P_3P_2L_1\underbrace{P_2^{-1}}_{P_2}\underbrace{P_3^{-1}}_{P_3}.$$

Dann gilt

$$\begin{aligned}\underbrace{L_3\hat{L}_2\hat{L}_1}_{L^{-1}}\underbrace{P_3P_2P_1}_{P}A &= L_3(P_3L_2P_3^{-1})(P_3P_2L_1P_2^{-1}P_3^{-1})P_3P_2P_1A\\ &= L_3P_3L_2P_2L_1P_1A\\ &= R\end{aligned}$$

Die Permutationsmatrix wird während der Rechnung gefunden. Sie ist im Allgemeinen nicht eindeutig.

Listing 6: Python Programm

```
# LR Zerlegung mit Zeilentausch

def PLR(A):
    import numpy as np

    n = A.shape[0]
    L = np.eye(n)
    P = np.eye(n)
    R = A.astype('float')

    for k in range(n - 1):
        p = np.argmax(abs(R[k:n,k])) # finde Pivotelement
        # Tausche Zeilen von R, L, P
        R[[k,p+k]] = R[[p+k, k]]
        P[[k,p+k]] = P[[p+k,k]]
        L[[k,p+k]] = L[[p+k,k]]
        # Tausche Spalten von L
```

```
        L.T[[k,p+k]] = L.T[[p+k,k]]

        for j in range(k+1,n):
            L[j,k] = R[j,k]/R[k,k]
            R[j,k:] = R[j,k:]-L[j,k]*R[k,k:]

    return (P,L,R)
```

Wir fassen die obigen Resultate in den folgenden Sätzen nochmal zusammen.

Satz 4.8 (Existenz). *Es sei $A \in \mathbb{R}^{n\times n}$ mit $\det A \neq 0$. Dann gibt es eine Permutationsmatrix P, so dass PA eine LR-Zerlegung besitzt, das heißt $PA = LR$.*

Satz 4.9 (Eindeutigkeit). *Es sei $\det A \neq 0$ und seien $PA = LR = \tilde{L}\tilde{R}$ zwei LR-Zerlegungen von PA mit normierten Matrizen L und $\tilde{L}$. Dann gilt $L = \tilde{L}$ und $R = \tilde{R}$, das heißt die LR-Zerlegung ist eindeutig.*

Beweis. Aus $LR = \tilde{L}\tilde{R}$ folgt $R = L^{-1}\tilde{L}\tilde{R}$ bzw. $R\tilde{R}^{-1} = L^{-1}\tilde{L}$. Da nun $R\tilde{R}^{-1}$ eine obere Dreiecksmatrix ist und $L^{-1}\tilde{L}$ eine normierte untere Dreiecksmatrix ist, muss $R\tilde{R}^{-1} = L^{-1}\tilde{L} = I$ sein und damit $L = \tilde{L}$ bzw. $R = \tilde{R}$. □

Bemerkung. Ausführliche Beispielrechnungen zur Bestimmung der Zerlegungen $A = LR$ und $PA = LR$ wurden in der online Vorlesung durchgeführt, siehe VL15.pdf.

Wir wollen nun die Frage beantworten, für welche Matrizen A mit $\det A \neq 0$ eine LR-Zerlegung mit $A = LR$ existiert.

Definition 4.10. Die *Hauptabschnittsmatrix* (kurz: HA-Matrix) k-ten Grades einer Matrix $A \in \mathbb{R}^{n\times n}$ ist die Matrix

$$A[k] := \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix}, \quad k \in \{1, \ldots, n\}$$

und $\det A[k]$ ist die HA-Determinante k-ten Grades.

Satz 4.11. *Es sei $A \in \mathbb{R}^{n\times n}$ mit $\det A \neq 0$. Dann besitzt A genau dann eine LR-Zerlegung (ohne Permutationsmatrix), wenn alle HA-Determinanten von A ungleich Null sind.*

Beweis. Wir zeigen zunächst: *A besitzt LR-Zerlegung $\Rightarrow$ HA-Determinante $\neq 0$*

Sei $A = LR$. Aus $\det A \neq 0$ folgt $\det L \neq 0$ und $\det R \neq 0$.

Außerdem gilt $A[k] = LR[k] = L[k]R[k]$, da zur Bildung von $A[k]$ aufgrund der Struktur von L und R nur Elemente aus $L[k]$ und $R[k]$ benötigt werden.

Daraus folgt

$$\det A[k] = \underbrace{\det L[k]}_{=1} \cdot \underbrace{\det R[k]}_{=r_{11}\cdot\ldots\cdot r_{kk}\neq 0} \neq 0 \quad \text{für } k = 1, \ldots, n$$

Nun zeigen wir die Rückrichtung, *HA-Determinante* $\neq 0 \Rightarrow$ *A besitzt LR-Zerlegung*, uner Verwendung der vollständigen Induktion nach k.

Induktionsanfang: Für $k = 1$ gilt $a_{11} = \det A[1] \neq 0$, d.h. L_1 ist ohne Permutationsmatrix P_1 bildbar.

Induktionsschritt $k \to k+1$: $(1 \leq k < n)$ Es gelte

$$A^{(k+1)} = L_k \ldots L_1 A = \left(\begin{array}{ccc|c} * & \cdots & * & * \\ & \ddots & \vdots & \vdots \\ & & * & * \\ \hline 0 & \cdots & 0 & \tilde{A} \end{array}\right)$$

Außerdem sei (nach Vor.) $\det A[k+1] \neq 0$. Daraus folgt

$$\begin{aligned} 0 \neq \det A[k+1] &= \underbrace{\det L_k[k+1]}_{=1} \ldots \underbrace{\det L_1[k+1]}_{=1} \det A[k+1] \\ &= \det(L_k \ldots L_1 A)[k+1] \\ &= a_{11}^{(1)} a_{22}^{(2)} \ldots a_{kk}^{(k)} a_{k+1,k+1}^{(k+1)} \end{aligned}$$

Folglich ist $a_{k+1,k+1}^{(k+1)} \neq 0$, d.h. $a_{k+1,k+1}^{(k+1)}$ ist als Pivotelement verwendbar. □

4.2 Die Cholesky-Zerlegung

Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit, das heißt $A = A^T$ und $x^T Ax > 0$ für alle $x \in \mathbb{R}^n \setminus \{0\}$. Dann kann man auf die Pivotisierung verzichten und die Berechnung der LR-Zerlegung vereinfacht sich.

Satz 4.12. *Es sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann besitzt A eine LR-Zerlegung der Form $A = \tilde{L}\tilde{L}^T$ mit einer im Allgemeinen nicht normierten unteren Dreiecksmatrix $\tilde{L}$. Mit einer normierten unteren Dreiecksmatrix L kann die Zerlegung auch in der Form*

$$A = LDL^T = (L\sqrt{D})(L\sqrt{D})^T = \tilde{L}\tilde{L}^T$$

mit einer positiv definiten Diagonalmatrix D geschrieben werden.

Beweis. Da A positiv definit ist, sind alle HA-Determinanten positiv. Aus Satz 4.11 folgt, dass A eine LR-Zerlegung besitzt mit normiertem L. Aus $A = A^T$ folgt

$$LR = A = A^T = R^T L^T.$$

Diese Zerlegung kann man auch schreiben als

$$A = LR = (R^T D^{-1})(DL^T)$$

für jedes invertierbare $D \in \mathbb{R}^{n\times n}$, speziell auch für die Diagonalmatrix

$$D := \begin{pmatrix} r_{11} & & \\ & \ddots & \\ & & r_{nn} \end{pmatrix},$$

wobei die $r_{ii} \neq 0$ die Diagonalelemente von R sind. Also ist $R^T D^{-1}$ eine normierte untere Dreiecksmatrix. Aus der Eindeutigkeit der LR-Zerlegung folgt

$$L = R^T D^{-1}, \quad R = DL^T.$$

Dies beweist die Existenz einer Zerlegung $A = LDL^T$ mit einer normierten unteren Dreiecksmatrix L.

Da A positiv definit ist, sind alle $A[k]$ positiv definit und somit $\det A[k] > 0$ für alle k. Es gilt

$$\det A[k] = \det(LR[k]) = \det(L[k]R[k]) = \det L[k] \det R[k] = r_{11} \cdots r_{nn} > 0.$$

Daraus folgt $r_{ii} > 0$ für alle $i = 1, \ldots, n$. Es sei

$$\sqrt{D} = \begin{pmatrix} \sqrt{r_{11}} & & \\ & \ddots & \\ & & \sqrt{r_{nn}} \end{pmatrix}.$$

Dann folgt $\sqrt{D}\sqrt{D} = D$ und

$$A = (L\sqrt{D})(\sqrt{D}L^T) = \tilde{L}\tilde{L}^T$$

mit $\tilde{L} = L\sqrt{D}$. □

Die Konsequenz dieser Zerlegung ist, dass nur der halbe Speicherplatz und ein geringerer Rechenaufwand benötigt wird.

Bemerkung. Symmetrische und positiv definite Matrizen kommen häufig in Anwendungen vor, zum Beispiel bei der Diskretisierung von Differentialgleichungen.

Eine Beispielrechnung zur Bestimmung der Cholesky Zerlegung finden Sie im pdf-File der online Vorlesung VL16.pdf.

4.3 Stabilitätsuntersuchungen der Gauß-Elimination

Bei der Untersuchung der Genauigkeit von numerischen Verfahren betrachtet man typischerweise zunächst die Kondition der mathematischen Aufgabe und anschließend die Stabilität des gewählten numerischen Verfahrens. Falls die mathematische Aufgabe gut konditioniert ist, dann besteht die Hoffnung, dass sich ein stabiler Algorithmus finden lässt.

Bei der Lösung von linearen Gleichungssystemen gibt es zwei Fehlerquellen. Auf der einen Seite können Fehler in den Eingangsdaten auftreten, d.h. Fehler in den Komponenten von A und b. Andererseits treten Fehler aufgrund von Rundungsfehlern bei der Ausführung des Lösungsverfahrens auf. Um die Genauigkeit eines numerischen Verfahrens zu diskutieren, benötigt man zunächst erst einmal eine Methode um den Fehler zu messen. Es gibt verschiedene Möglichkeiten, den Fehler zu messen, die teilweise unterschiedliche Eindrücke über die Qualität der Lösung vermitteln.

4.3.1 Grundlagen der Fehleranalyse

Die folgenden Definitionen und Sätze sind sicherlich bereits aus den Grundvorlesungen bekannt und sollen hier nur noch einmal erwähnt werden. Wir haben uns bereits häufiger mit dem Fehler einer durch ein numerisches Verfahren berechneten Grö l sse beschäftigt. Bei der Lösung linearer Gleichungssysteme möchten wir den Fehler einer vektorwertigen Größe beschreiben.

Fehler in skalaren Größen

Beispiel. Berechne die Nullstellen einer Funktion $f : \mathbb{R} \to \mathbb{R}$. Gesucht ist also $\hat{z}$ mit $f(\hat{z}) = 0$. Es sei z die approximative Lösung. Dann gilt für den Fehler $e := z - \hat{z}$. Der absolute Fehler ist $|e| = |z - \hat{z}|$ und der relative Fehler ist $\left|\frac{z-\hat{z}}{\hat{z}}\right|$.

Fehler in vektorwertigen Größen

Beispiel. Es sei $\hat{z} \in \mathbb{R}^s$ Lösung eines linearen Gleichungssystems und $z \in \mathbb{R}^s$ eine approximative Lösung. In diesem Fall können wir den Fehler mit Hilfe einer Vektornorm messen. Es gibt verschiedene Möglichkeiten, eine Vektornorm zu definieren.

Definition 4.13. Eine Abbildung $\|\cdot\| : \mathbb{K}^n \to \mathbb{R}_+$ mit $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ heißt *Norm*, wenn sie die folgenden Eigenschaften besitzt:

(N1) $\|x\| > 0, \quad x \in \mathbb{K}^n \setminus \{0\}$.

(N2) $\|\alpha x\| = |\alpha|\|x\|, \quad x \in \mathbb{K}^n \quad \alpha \in \mathbb{K}$.

(N3) $\|x + y\| \leq \|x\| + \|y\|, \quad x, y \in \mathbb{K}^n$.

Beispiel.

- Maximum-Norm
$$\|e\|_\infty = \max_{1 \leq i \leq s} |e_i|,$$

- 1-Norm
$$\|e\|_1 = \sum_{i=1}^{s} |e_i|,$$

- 2-Norm
$$\|e\|_2 = \sqrt{\sum_{i=1}^{s} |e_i|^2},$$

- p-Norm
$$\|e\|_p = \left(\sum_{i=1}^{s} |e_i|^p \right)^{\frac{1}{p}}.$$

Es stellt sich die Frage: Hängen Konvergenzaussagen eines numerischen Verfahrens von der speziellen Wahl der verwendeten Norm ab?

Definition 4.14. Es sei X ein $\mathbb{K}$-Vektorraum und $\|\cdot\|, \|\cdot\|_*$ seien zwei Normen auf X. Diese Normen heißen *äquivalent*, wenn Konstanten $0 < c \leq C < \infty$ existieren mit

$$c\|x\| \leq \|x\|_* \leq C\|x\|, \quad \text{für alle} \quad x \in X.$$

Ein wichtiges Resultat aus der Analysis liefert der folgende Satz.

Satz 4.15. *Alle Normen auf $\mathbb{R}^n$ sind äquivalent.*

Dieser Satz besagt: Falls wir $\|e(h)\| = \mathcal{O}(h^q)$ in einer Norm zeigen können, so gilt die gleiche Fehlerabschätzung in jeder anderen Norm. Für $x \in \mathbb{R}^s$ gilt beispielsweise (Übung)

$$\|x\|_\infty \leq \|x\|_1 \leq s\|x\|_\infty, \tag{(i)}$$
$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{s}\|x\|_\infty, \tag{(ii)}$$
$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{s}\|x\|_2. \tag{(iii)}$$

Beachte: Diese Aussage ist im Allgemeinen nicht in unendlich-dimensionalen Räumen gültig.

Matrixnormen

Für jede Vektornorm $\|\cdot\|$ können wir eine korrespondierende Matrixnorm definieren. Es ist $C = \|A\|$ die kleinste Zahl, so dass $\|Ax\| \leq C\|x\|$ für alle $x \in \mathbb{R}^s$ und $A \in \mathbb{R}^{s\times s}$ gilt.

Definition 4.16. Die durch die Vektornorm $\|\cdot\|$ induzierte Matrixnorm ist definiert durch

$$\|A\| := \max_{x\in\mathbb{R}^s, x\neq 0} \frac{\|Ax\|}{\|x\|} = \max_{x\in\mathbb{R}^s, \|x\|=1} \|Ax\|.$$

Es gilt

$$\|A\|_1 = \max_{1\leq j\leq s} \left(\sum_{i=1}^{s} |a_{ij}| \right), \qquad \text{(max. Spaltensumme)}$$

$$\|A\|_\infty = \max_{1\leq j\leq s} \left(\sum_{i=1}^{s} |a_{ji}| \right), \qquad \text{(max. Zeilensumme)}$$

$$\|A\|_2 = \sqrt{\rho\left(A^T A\right)} = \max\{\sqrt{|\lambda|} \mid \lambda \in \mathbb{R},\ A^T Ax = \lambda x,\ x \in \mathbb{R}^n \setminus \{0\}\}.$$

Dabei bezeichnet $\rho(A^T A)$ den Spektralradius von $A^T A$.

Bemerkung. Es sei $A = A^T$. Dann gilt $\|A\|_2 = \rho(A)$.

Definition 4.17. Die *Konditionszahl* einer Matrix A in einer gegebenen Norm ist $\kappa(A) = \|A\|\|A^{-1}\|$, falls A invertierbar ist. Falls A nicht invertierbar ist, so sei $\kappa(A) = \infty$.

Bemerkung. Falls A symmetrisch und positiv definit ist, so gilt für die Konditionszahl in der 2-Norm

$$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)},$$

wobei $\lambda_{max,min}(A)$ der größte beziehungsweise kleinste Eigenwert von A ist.

4.3.2 Konditionsuntersuchung für die Lösung linearer Gleichungssysteme

Wir betrachten das lineare Gleichungssystem

$$Ax = b, \quad A \in \mathbb{K}^{n\times n} \quad \text{invertierbar.}$$

Die Daten A und b seien fehlerbehaftet. Wir erhalten ein gestörtes System

$$\tilde{A}\tilde{x} = \tilde{b}$$

mit $\tilde{A} = A + \delta A$, $\tilde{b} = b + \delta b$ und $\tilde{x} = x + \delta x$. Unser Ziel ist es nun, eine Abschätzung von $\tilde{x}$ in Abhängigkeit von $\tilde{A}$ und $\tilde{b}$ zu finden.

Lemma 4.18. *Die Matrix $B \in \mathbb{K}^{n\times n}$ habe die Norm $\|B\| < 1$. Dann ist $I + B$ regulär und es gilt*

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

Beweis. Es sei $x \in \mathbb{K}^n$. Es gilt

$$\|(I+B)x\| \geq \|x\| - \|Bx\| \geq (1-\|B\|)\|x\| > 0$$

für $x \neq 0$, da nach Voraussetzung $1-\|B\| > 0$ gilt. Also hat $(I+B)x = 0$ nur die triviale Lösung $x = 0$. Damit ist $I + B$ regulär. Außerdem gilt

$$\begin{aligned} 1 = \|I\| &= \|(I+B)(I+B)^{-1}\| \\ &= \|(I+B)^{-1} + B(I+B)^{-1}\| \\ &\geq \|(I+B)^{-1}\| - \|B\|\|(I+B)^{-1}\| \\ &= \|(I+B)^{-1}\|(1-\|B\|) \\ &> 0. \end{aligned}$$

Umformen liefert die Behauptung. □

Satz 4.19 (Störungssatz)**.** *Die Matrix $A \in \mathbb{K}^{n\times n}$ sei regulär und es sei $\|\delta A\| < \frac{1}{\|A^{-1}\|}$. Dann ist die gestörte Matrix $\tilde{A} = A + \delta A$ ebenfalls regulär und für den relativen Fehler der Lösung gilt*

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1-\kappa(A)\frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right).$$

Beweis. Es gilt nach Voraussetzung

$$\|A^{-1}\delta A\| \leq \|A^{-1}\|\|\delta A\| < 1, \qquad (*)$$

also ist $A + \delta A = A(Id + A^{-1}\delta A)$ nach Lemma 4.18 regulär. Betrachte

$$\begin{aligned} & & \tilde{A}\tilde{x} &= \tilde{b} \\ \Leftrightarrow & & (A+\delta A)(x+\delta x) &= b + \delta b \\ \Leftrightarrow & & (A+\delta A)\delta x &= b + \delta b - (A+\delta A)x = \delta b - \delta A x \\ \Leftrightarrow & & \delta x &= (A+\delta A)^{-1}(\delta b - \delta A x) \end{aligned}$$

Damit folgt

$$\begin{aligned}
\|\delta x\| &\leq \|(A+\delta A)^{-1}\|(\|\delta b\| + \|\delta A\|\|x\|) \\
&= \| \left(A\left(I + A^{-1}\delta A\right)\right)^{-1} \| \left(\|\delta b\| + \|\delta A\|\|x\|\right) \\
&= \|(I + A^{-1}\delta A)^{-1}A^{-1}\| \left(\|\delta b\| + \|\delta A\|\|x\|\right) \\
&\leq \|(I + A^{-1}\delta A)^{-1}\|\|A^{-1}\| \left(\|\delta b\| + \|\delta A\|\|x\|\right) \\
&\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\delta A\|} \left(\|\delta b\| + \|\delta A\|\|x\|\right) \\
&= \frac{\|A^{-1}\|\|A\|\|x\|}{1 - \|A^{-1}\delta A\|} \left(\frac{\|\delta b\|}{\|A\|\|x\|} + \frac{\|\delta A\|}{\|A\|}\right) \\
&= \frac{\kappa(A)\|x\|}{1 - \|A^{-1}\delta A\|\|A\|\|A\|^{-1}} \left(\frac{\|\delta b\|}{\|A\|\|x\|} + \frac{\|\delta A\|}{\|A\|}\right) \\
&\overset{(*)}{\leq} \frac{\kappa(A)}{1 - \kappa(A)\|\delta A\|\|A\|^{-1}} \left(\frac{\|\delta b\|}{\|A\|\|x\|} + \frac{\|\delta A\|}{\|A\|}\right) \|x\| \\
&\overset{\|b\| \leq \|A\|\|x\|}{\leq} \frac{\kappa(A)}{1 - \kappa(A)\|\delta A\|\|A\|^{-1}} \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right) \|x\|
\end{aligned}$$

□

Ist $\kappa(A)\|\delta A\|\|A\|^{-1} \ll 1$, so erhalten wir in erster Näherung

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A) \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right).$$

Die Konditionszahl der Matrix A gibt an wie stark sich Störungen in der Matrix A und / oder der rechten Seite b auf den relativen Fehler im Ergebnis auswirken.

Beispiel. Betrachten wir noch einmal die schlecht konditionierte Aufgabe aus Abschnitt 1.3 mit

$$A = \begin{pmatrix} 1{,}2969 & 0{,}8648 \\ 0{,}2161 & 0{,}1441 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 0{,}8642 \\ 0{,}1440 \end{pmatrix}.$$

Die Lösung von $Ax = b$ ist $x = (2, -2)^T$. Es sei

$$\tilde{b} = b + \varepsilon \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \text{mit} \quad \varepsilon = 0{,}0001$$

Dann ist die Lösung von $Ax = \tilde{b}$ gegeben durch $x = (0{,}9911, -0{,}4870)^T$. Es gilt

$$A^{-1} = 10^8 \begin{pmatrix} 0{,}1441 & -0{,}8648 \\ -0{,}2161 & 1{,}2969 \end{pmatrix}.$$

Daraus folgt

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = 2{,}1617 \cdot 10^8 \cdot 1{,}1513 \approx 3{,}27 \cdot 10^8$$

Das System ist also sehr schlecht konditioniert.

4.3.3 Instabilität der Gauß-Elimination

Wir betrachten für $0 < \epsilon \ll 1$ die Lösung des linearen Gleichungssystem

$$\underbrace{\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 1 \\ 2 \end{pmatrix}}_{b}. \tag{4.1}$$

Dieses Gleichungssystem hat die exakte Lösung $x_1 = 1/(1-\epsilon)$, $x_2 = (1-2\epsilon)/(1-\epsilon)$. Die Koeffizientenmatrix ist für $0 < \epsilon \ll 1$ gut konditioniert und somit ist, mit Satz 4.19 und Definition 1.2, die Lösung dieses Gleichungssystems ein gut konditioniertes mathematisches Problem.

Die Koeffizientenmatrix besitzt die LR-Zerlegung

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \epsilon & 1 \\ 0 & 1-\epsilon^{-1} \end{pmatrix}}_{R}.$$

Nun kann man die Systeme

$$Ly = b$$
$$Rx = y$$

durch Vorwärts- bzw. Rückwärtssubstitution lösen. Für $\epsilon < 10^{-17}$ erhält man (bei Verwendung eines Computers) die falsche Lösung $x_1 = 0$, $x_2 = 1$, d.h. der Algorithmus ist instabil.

Für kleine Werte von ϵ haben die Matrizen L und R Einträge mit betragsmäßig sehr großen Werten. Die Konditionszahl der Matrizen L und R ist für kleine ϵ sehr groß.

Die Teilprobleme $Ly = b$ und $Rx = y$ sind schlecht konditionierte mathematische Aufgaben und der beschriebene Algorithmus ist im Allgemeinen instabil.

Zur Vermeidung von Instabilitäten führt man einen Zeilentausch ein. Dabei möchte man betragsgroße Elemente in der Matrix L nach Möglichkeit vermeiden. In der k-ten Spalte sucht man $p \in \{k, \ldots, n\}$ mit

$$|a_{p,k}^{(k)}| = \max_{i=k,\ldots,n} |a_{i,k}^{(k)}|$$

und verstauscht anschließend die Zeilen p und k.

In unserem Testbeispiel (4.1) betrachten wir nach einem Zeilentausch das System

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

Die Koeffizientenmatrix hat nun die LR-Zerlegung

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

In den Matrizen L und R tauchen keine betragsmäßig großen Einträge mehr auf und der Algorithmus liefert eine sehr genaue Lösung.

Es gibt Matrizen, bei denen sich ein Anwachsen der Matrixelemente in R nicht vermeiden lässt. Betrachte dazu

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{pmatrix}}_{R}.$$

Für eine analog gebildete $m \times m$ Matrix A gilt $r_{m,m} = 2^{m-1}$. Dies führt zu instabilen Resultaten bei der Lösung linearer Gleichungssysteme unter Verwendung der LR-Zerlegung. Bei dieser speziellen Matrix hilft uns auch ein Zeilentausch nicht weiter.

Kann es sein, dass ein so weit verbreiteter Algorithmus zur Lösung linearer Gleichungssysteme instabil ist? Bei diesem Beispiel handelt es sich tatsächlich um eine Instabilität, die in praktischen Anwendungen bisher nicht beobachtet wurde. Trefethen und Bau[1] schreiben, daß innerhalb der letzten 50 Jahre in keinem für Anwender relevanten Gleichungssystem Matrizen aufgetreten sind, für die die Gaußelimination mit Pivotisierung instabile Resultate lieferte. Es scheint, dass derartige Matrizen sehr selten sind.

[1] Lloyd N. Trefethen, David Bau, Numerical Linear Algebra, Lecture 22, SIAM 1997

5 Die QR-Zerlegung und die Lösung linearer Ausgleichsprobleme

Wir betrachten nun Zerlegungen der Form $A = QR$ mit einer orthogonalen Matrix Q und einer oberen Dreiecksmatrix R. Wir beschränken unsere Betrachtungen in diesem Kapitel auf reellwertige Matrizen, obwohl eine Verallgemeinerung für komplexwertige Matrizen ohne größere Schwierigkeiten möglich ist.

Die QR Zerlegung ist ein wichtiger Teilschritt bei der Lösung fundamentaler Probleme der angewandten Mathematik. Mit Hilfe der QR Zerlegungen einer regulären quadratischen Matrix kann man einfach lineare Gleichungssysteme lösen. Außerdem können überbestimmte lineare Gleichungssysteme als sogenannte Ausgleichsprobleme gelöst werden.

In der Numerik 2 werden Algorithmen zur Berechnung von Eigenwerten und Eigenvektoren betrachtet, die ebenfalls auf QR-Zerlegungen basieren.

Betrachten wir nun allgemein rechteckige Matrizen $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und schreiben $A = (a_1|a_2|\dots|a_n)$, d.h. a_i bezeichne die $i$te Spalte von A. Es gilt

$$\langle a_1\rangle \subseteq \langle a_1, a_2\rangle \subseteq \langle a_1, a_2, a_3\rangle \subseteq \dots,$$

wobei $\langle \dots \rangle$ den durch die Vektoren innerhalb der Klammern aufgespannten Teilraum bezeichne.

Unser Ziel ist es, eine Folge von orthonormalen Vektoren $q_1, q_2, \dots$ zu konstruieren, die diese Unterräume aufspannen, also für $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und $\operatorname{rank}(A) = n$ soll

$$\langle q_1, \dots, q_j\rangle = \langle a_1, \dots, a_j\rangle, \quad j = 1, \dots, n$$

erfüllt sein. Weiterhin soll

$$(a_1|a_2|\dots|a_n) = (q_1|q_2|\dots|q_n)\begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ & & r_{nn} \end{pmatrix}$$

mit $r_{kk} \neq 0$ für alle $k = 1, \dots, n$ gelten. Damit erhalten wir die **reduzierte QR-Zerlegung** $A = \hat{Q}\hat{R}$ mit einer orthogonalen $(m\times n)$-Matrix $\hat{Q}$ und einer oberen $(n\times n)$-Dreiecksmatrix $\hat{R}$.

Bei der **vollständigen QR-Zerlegung** $A = QR$ ist $Q \in \mathbb{R}^{m\times m}$ orthogonal und die Spalten q_j mit $j > n$ sind orthogonal zu $\langle a_1, \ldots, a_n\rangle$. Es gilt $R \in \mathbb{R}^{m\times n}$, wobei $r_{ij} = 0$ für alle $i > n$ und alle j gilt.

Wir fassen die eingeführte Zerlegung noch einmal in der folgenden Definition zusammen.

Definition 5.1. Die Zerlegung einer Matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$ in ein Produkt

$$A = QR$$

aus einer orthogonalen Matrix $Q \in \mathbb{R}^{m\times m}$ und einer rechten oberen Dreiecksmatrix $R \in \mathbb{R}^{m\times n}$ heißt **vollständige QR-Zerlegung**.

Eine Zerlegung der gleichen Matrix A in ein Produkt

$$A = \hat{Q}\hat{R}$$

mit einer in den Spalten orthogonalen Matrix $\hat{Q} \in \mathbb{R}^{m\times n}$ und einer rechte obere Dreiecksmatrix $\hat{R} \in \mathbb{R}^{n\times n}$ heißt **reduzierte QR-Zerlegung**.

Aus der linearen Algebra wissen wir: Eine Matrix $Q \in \mathbb{R}^{m\times m}$ heißt *orthogonal*, wenn $Q^TQ = I$ gilt.

5.1 Die QR-Zerlegung mittels Gram-Schmidt

Der aus der linearen Algebra bekannte Gram-Schmidt Algorithmus berechnet zu linear unabhängigen Vektoren orthogonale Vektoren, die den gleichen Raum aufspannen. Im Pseudocode lässt sich dieser Algorithmus wie folgt ausdrücken:

for *j=1 to n* **do**
  $v_j = a_j$
  for *i=1 to j-1* **do**
    $r_{ij} = q_i^T a_j$
    $v_j = v_j - r_{ij} q_i$
  end
  $r_{jj} = \|v_j\|_2$
  $q_j = v_j / r_{jj}$
end

Algorithm: Der klassische Gram Schmidt Algorithmus im Pseudocode

Alle Matrizen $A \in \mathbb{R}^{m\times n}$, $m \geq n$ besitzen eine QR Zerlegung. Diese ist unter einer zusätzlichen Bedingung eindeutig.

Satz 5.2. *Jede Matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$ besitzt eine vollständige QR-Zerlegung und somit auch eine reduzierte QR-Zerlegung.*

Beweis. Wir nehmen zunächst an, dass A vollen Rang hat, d.h. $rank(A) = n$. Die Existenz einer reduzierten QR-Zerlegung folgt dann aus dem Gram-Schmidt Algorithmus. Der Algorithmus generiert die orthogonalen Spalten von $\hat{Q}$ und die Einträge der oberen Dreiecksmatrix $\hat{R}$. Probleme bei der Anwendung des Algorithmus können nur auftreten, falls v_j der Nullvektor ist und somit die Normalisierung zu einer Division durch Null führt. Dies würde bedeuten, dass

$$a_j \in \langle q_1, \ldots, q_{j-1}\rangle = \langle a_1, \ldots, a_{j-1}\rangle$$

gilt, ein Widerspruch zu $rank(A) = n$.

Nun nehmen wir an, dass $rank(A) < n$ gilt. In einem oder in mehreren Schritten liefert der Gram-Schmidt Algorithmus nun einen Nullvektor $v_j = 0$. Falls diese Situation eintritt, dann wählt man für q_j irgendeinen normalisierten Vektor , der orthogonal zu $\langle q_1, \ldots, q_{j-1}\rangle$ steht und setzt mit diesem Vektor den Algorithmus fort.

Die vollständige QR-Zerlegung einer $m \times n$ Matrix konstruiert man, indem analog zusätzliche orthonormale Vektoren eingeführt werden. □

Satz 5.3. *Die reduzierte QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m\times n}$, mit $m \geq n$, $rank(A) = n$ und $r_{jj} > 0$, $j = 1, \ldots, n$, ist eindeutig.*

Beweis. Seien $A = \hat{Q}_1\hat{R}_1 = \hat{Q}_2\hat{R}_2$ zwei reduzierte QR-Zerlegungen von A. Dann gilt:

$$\hat{Q}_2^T\hat{Q}_1 = \hat{R}_2\hat{R}_1^{-1}, \quad \hat{Q}_1^T\hat{Q}_2 = \hat{R}_1\hat{R}_2^{-1}.$$

Die Matrizen $\hat{R}_2\hat{R}_1^{-1}$ und $\hat{R}_1\hat{R}_2^{-1}$ sind rechte obere Dreiecksmatrizen. Außerdem ist

$$Q = \hat{Q}_2^T\hat{Q}_1 = \left(\hat{Q}_1^TQ_2\right)^T$$

als Produkt von orthogonalen Matrizen eine orthogonale Matrix, d.h. $Q^{-1} = Q^T$. Wegen $Q^{-1} = Q^T$ und $Q = \hat{R}_2\hat{R}_1^{-1}$ muss Q eine Diagonalmatrix sein. Aus

$$Q\hat{R}_1 = \hat{Q}_2^T\hat{Q}_1\hat{R}_1 = \hat{Q}_2^TA = \hat{R}_2$$

folgt für den j-ten Einheitsvektor e_j:

$$e_j^TQ\hat{R}_1e_j = q_{jj}r_{jj}^1 = r_{jj}^2 > 0,$$

also $q_{jj} > 0$ und wegen der Orthogonalität $Q = I$. D.h.:

$$\hat{R}_1 = \hat{R}_2 \quad \Rightarrow \quad \hat{Q}_1 = A\hat{R}_1^{-1} = A\hat{R}_2^{-1} = \hat{Q}_2.$$

□

Bemerkung. Ein Rechenbeispiel zur Bestimmung der vollständigen und reduzierten QR-Zerlegung unter Verwendung des Gram-Schmidt Algorithmus finden Sie im pdf-File der online Vorlesung VL17.pdf.

Aufgrund von Auslöschungseffekten ist das Gram-Schmidt-Verfahren zur Berechnung der QR-Zerlegung numerisch instabil, falls die Spaltenvektoren von A nahezu parallel sind. Dies wollen wir uns anhand eines Beispiels veranschaulichen. Wir verwenden dazu die Python Implementation des Gram-Schmidt Algorithmus aus Listing 7 und wenden diese auf die Matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 10^{-8} & 0 & 0 \\ 0 & 10^{-8} & 0 \\ 0 & 0 & 10^{-8} \end{pmatrix}$$

an. Anschließend benutzen wir die berechnete Matrix $\hat{Q}$ und bestimmen $\hat{Q}^T\hat{Q}$. Wir erhalten

$$\hat{Q}^T\hat{Q} = \begin{pmatrix} 1 & -7.071\ldots\cdot 10^{-9} & -7.071\ldots\cdot 10^{-9} \\ -7.071\ldots\cdot 10^{-9} & 1 & 0.5 \\ -7.071\ldots\cdot 10^{-9} & 0.5 & 1 \end{pmatrix},$$

und somit eine deutliche Abweichung von der Einheitsmatrix.

Listing 7: QR Zerlegung nach Gram-Schmidt

```
# Berechne reduzierte QR Zerlegung unter Verwendung von Gram-Schmidt
def GramSchmidt(A):
    import numpy as np
    import numpy.linalg as npl

    m, n = A.shape;
    Q = np.zeros([m,n])      # Null-Array der Größe mxn
    R = np.zeros([n,n])        # Null-Array der Größe nxn
    for j in range(n):
        Q[:,j] = A[:,j];
        # Orthogonalisiere A[:,j] gegen Q[:,1], ...,Q[:,j-1]
        for i in range(j):
            R[i,j]=Q[:,i].T@A[:,j]
            Q[:,j]=Q[:,j]-R[i,j]*Q[:,i]
        # Normiere Q[:,j]
        R[j,j]=npl.norm(Q[:,j])
        Q[:,j]=Q[:,j]/R[j,j]
    return Q, R
```

Daher werden wir andere Verfahren kennenlernen um die QR-Zerlegung numerisch stabil zu berechnen.

Für unsere Anwendungen benötigen wir die reduzierten QR-Zerlegung. Wir werden daher in den nächsten Abschnitten die Matrizen stets mit Q und R bezeichnen und auf den Zusatz reduziert bzw. vollständig verzichten.

5.2 Berechnung der QR-Zerlegung mittels Housholder Reflektion

Bei der Householder Reflektion erzeugt man die Matrix R, indem man A von links mit einer Folge orthogonaler Matrizen multipliziert. Die Basis des Algorithmus ist die folgende geometrische Betrachtung.

Bemerkung 5.4. 1) Es sei $w \in \mathbb{R}^n$ mit $w^T w = 1$. Dann ist $I - 2ww^T$ eine symmetrische orthognale Matrix.

2) Die Matrix $Q = I - 2ww^T$ mit $w^T w = 1$ beschreibt eine Spiegelung an der Hyperebene

$$H := \{x \in \mathbb{R}^n \mid w^T x = 0\}.$$

Der Beweis wurde in der online Vorlesung ausgeführt, siehe VL18.pdf.

Definition 5.5. Eine Matrix der Form $I - 2ww^T$ mit $w^T w = 1$ heißt *Housholdermatrix*.

Wir können nun die nach Satz 5.2 existierende QR-Zerlegung auch mittels Householder Spiegelungen konstruieren. Dabei geht man wie folgt vor:

1. Schritt: Finde eine Housholdermatrix $Q_1 = I - 2ww^T$ mit $w^T w = 1$ und

$$Q_1 A = \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & * & \\ 0 & & & \end{array}\right),$$

das heißt die erste Spalte ist ein Vielfaches von e_1. Spiegele den ersten Spaltenvektor y von A in ke_1 nach Bemerkung 5.4 2).

Durch die Winkelhalbierende ist w bestimmt. Alle weiteren Schritte werden dann analog auf die Teilmatrizen angewendet und wir erhalten

$$Q_m \cdots Q_1 A = R.$$

Aufgrund der Symmetrie der Q_i können wir somit $Q = (Q_m \cdots Q_1)^T = Q_1 \cdots Q_m$ setzen.

Berechnung von k: Es ist $ke_1 = (I - 2ww^T)y = Qy$. Daraus folgt

$$|k|^2 = \|ke_1\|_2^2 = \|Qy\|_2^2 = (Qy)^T(Qy) = y^TQ^TQy = y^Ty = \|y\|_2^2.$$

Also $|k| = \|y\|_2^2$.

Berechnung von w: Es sei $L := w^Ty$. Dann gilt

$$Qy = (I - 2ww^T)y = y - 2ww^Ty = y - 2Lw = ke_1.$$

Daraus folgt $2Lw = y - ke_1$. Nun gilt nach Voraussetzung $w^Tw = 1$ und wir erhalten

$$\begin{aligned} 4L^2 = 4L^2w^Tw &= (2Lw)^T(2Lw) \\ &= (y - ke_1^T)(y - ke_1) \\ &= y^Ty - 2ky_1 + k^2 \\ &= 2(k^2 - ky_1) \\ &= 2|k|(|k| - \operatorname{sign}(k)y_1). \end{aligned}$$

Zur Festlegung von $|L|$: Für $y_1 \neq 0$ gibt es zwei mögliche Spiegelungen.

1. Fall: $\operatorname{sign}(k) = -\operatorname{sign}(y_1)$.

Es folgt

$$2L^2 = |k|(|k| + \operatorname{sign}(y_1)y_1) = |k|(|k| + |y_1|) = \|y\|_2(\|y\|_2 + |y_1|).$$

2. Fall: $\operatorname{sign}(k) = \operatorname{sign}(y_1)$.

Es folgt

$$2L^2 = \|y\|_2(\|y\|_2 - |y_1|).$$

Aufgrund von Rundungsfehlern ist es besser, $\operatorname{sign}(k) = -\operatorname{sign}(y_1)$ zu wählen, da dann der Betrag von L am größten ist. Aus Gründen der numerischen Stabilität wählt man die Spiegelung so, dass der gespiegelte Punkt weiter von y entfernt ist.

Zusammenfassung:

$$\begin{aligned} |L| &:= \sqrt{\frac{\|y\|_2(\|y\|_2 + |y_1|)}{2}}, \\ k &:= \begin{cases} -\operatorname{sign}(y_1)\|y\|_2, & y_1 \neq 0, \\ -\|y\|_2, & y_1 = 0, \end{cases} \\ w &:= \frac{1}{2|L|}(y - ke_1). \end{aligned}$$

In der CompLA Vorlesung haben wir bereits den folgenden Pseudocode der QR-Zerlegung mittels Housholder-Reflektion als Python Funktion implementiert.

$A \in \mathbb{R}^{m \times n}$, $m \geq n$, rank(A) = n
$R = \text{copy}(A)$
$Q = I(m)$
for *k=1 to n* **do**
  $x = R[k:m, k]$
  $v[k] = \text{sign}(x[1])\|x\|_2\, e_1 + x$
  $v[k] = v[k]/\|v[k]\|_2$
  $R[k:m, k:n] = R[k:m, k:n] - 2v[k](v[k]^T R[k:m, k:n])$
  $Q[k:m, 1:m] = Q[k:m, 1:m] - 2v[k](v[k]^T Q[k:m, 1:m])$
end
return Q^T, R

Algorithm: Householder QR-Zerlegung im Pseudocode

Wir betrachten nun noch ein Beispiel zur Berechnung der QR-Zerlegung einer 3×3 Matrix mittels Householder-Reflektion. Im pdf-File der online Vorlesung finden Sie ein Beispiel zur Berechnung der QR-Zerlegung einer 4×3 Matrix.

Beispiel. Betrachte

$$A = \begin{pmatrix} 0 & -4 & 2 \\ 6 & -3 & -2 \\ 8 & 1 & -1 \end{pmatrix}.$$

1. Spalte: Es ist

$$y = \begin{pmatrix} 0 \\ 6 \\ 8 \end{pmatrix}, \quad \|y\|_2 = \sqrt{0^2 + 6^2 + 8^2} = \sqrt{100} = 10.$$

Weiter ist $k = -\|y\|_2 = -10$ und

$$L = \sqrt{\frac{\|y\|_2(\|y\|_2 + |y_1|)}{2}} = \frac{\|y\|_2}{\sqrt{2}} = \frac{10}{\sqrt{2}}.$$

Es folgt

$$w = \frac{1}{10\sqrt{2}}(y - ke_1) = \frac{1}{10\sqrt{2}} \begin{pmatrix} 10 \\ 6 \\ 8 \end{pmatrix}$$

und damit

$$Q_1 = I - 2ww^T = \begin{pmatrix} 0 & -0{,}6 & -0{,}8 \\ -0{,}6 & 0{,}64 & -0{,}48 \\ -0{,}8 & -0{,}48 & 0{,}36 \end{pmatrix}.$$

Es ist

$$Q_1 A = \begin{pmatrix} -10 & 1 & 2 \\ 0 & 0 & -2 \\ 0 & 5 & -1 \end{pmatrix}.$$

2. Spalte: Es ist

$$y = \begin{pmatrix} 0 \\ 5 \end{pmatrix}, \quad \|y\|_2 = 5.$$

Es folgt

$$k = -5, \quad L = \frac{5}{\sqrt{2}}, \quad w = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Wir erhalten

$$\tilde{Q}_2 = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

und damit

$$\tilde{Q}_2 \begin{pmatrix} 0 & -2 \\ 5 & 1 \end{pmatrix} = \begin{pmatrix} -5 & 1 \\ 0 & 2 \end{pmatrix}.$$

Dies liefert uns insgesamt

$$Q_2 Q_1 A = \begin{pmatrix} -10 & 1 & 2 \\ 0 & -5 & 1 \\ 0 & 0 & 2 \end{pmatrix} = R, \quad Q = Q_1^T Q_2^T = \begin{pmatrix} 0 & 0{,}8 & 0{,}6 \\ -0{,}6 & 0{,}48 & -0{,}64 \\ -0{,}8 & -0{,}36 & 0{,}48 \end{pmatrix},$$

wobei

$$Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

5.3 Die Berechnung der QR-Zerlegung mittels Givens Rotation

Der Householder Algorithmus benutzt Spiegelungen zur Berechnung der QR-Zerlegung. Alternativ kann man auch unter Verwendung von Rotationsmatrizen die QR-Zerlegung berechnen. Mit solch einem Ansatz wollen wir uns nun beschäftigen.

Definition 5.6. Sei $i < j$. Eine Givens-Rotation mit Winkel θ zwischen den Koordinaten i und j wird durch eine Matrix $G(i, j, \theta) \in \mathbb{R}^{n \times n}$ der Form

$$G(i, j, \theta) = I - (1 - \cos(\theta) e_i e_i^T - \sin(\theta) e_i e_j^T + \sin(\theta) e_j e_i^T - (1 - \cos(\theta)) e_j e_j^T$$

bzw. (ausgeschriebener)

$$G(i,j,\theta) = \begin{pmatrix} I_{i-1} & 0 & 0 & 0 & 0 \\ 0 & \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 0 & I_{j-i-1} & 0 & 0 \\ 0 & \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 0 & I_{n-j} \end{pmatrix}$$

beschrieben. Dabei ist e_i der i-te Einheitsvektor und I_k die $k \times k$ Einheitsmatrix.

Es gilt (einfaches Nachrechnen):

$$G(i,j,\theta) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{cases} x_k & : \quad k \neq i, k \neq j \\ x_i \cos(\theta) - x_j \sin(\theta) & : \quad k = i \\ x_i \sin(\theta) + x_j \cos(\theta) & : \quad k = j \end{cases}$$

D.h., die Matrix-Vektor-Multiplikation mit einer Givens-Rotationsmatrix ändert nur zwei Komponenten des Vektors. Multipliziert man eine Matrix von Links mit einer Givens-Rotationsmatrix, so ändern sich nur zwei Zeilen der Matrix.

Wir werde auch die folgenden Eigenschaften der Givens-Rotationsmatrizen nutzen.

Bemerkung 5.7. Es gilt

(i) $G(i,j,\theta)G(i,j,\phi) = G(i,j,\theta+\phi)$

(ii) $G(i,j,\theta)^T = G(i,j,-\theta)$

(iii) $G(i,j,\theta)$ ist orthonormal

Beweis. Wir können uns darauf beschränken die Eigenschaften für 2×2 Matrizen nachzuweisen.

zu (i):

$$\begin{aligned} &\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta)\cos(\phi) - \sin(\theta)\sin(\phi) & -\cos(\theta)\sin(\phi) - \sin(\theta)\cos(\phi) \\ \sin(\theta)\cos(\phi) + \cos(\theta)\sin(\phi) & -\sin(\theta)\sin(\phi) + \cos(\theta)\cos(\phi) \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta+\phi) & -\sin(\theta+\phi) \\ \sin(\theta+\phi) & \cos(\theta+\phi) \end{pmatrix} \end{aligned}$$

zu (ii): folgt aus $\sin(\alpha) = -\sin(-\alpha)$ und $\cos(\alpha) = \cos(-\alpha)$

zu (iii): mit (ii) und (i) gilt

$$\begin{aligned} G(i,j,\theta)G(i,j,\theta)^T &= G(i,j,\theta)G(i,j,-\theta) \\ &= G(i,j,0) \\ &= I \end{aligned}$$

□

Bei der QR-Zerlegung mittels Givens-Rotationen transformiert man die Matrix A wieder schrittweise in eine obere rechte Dreiecksmatrix R. Durch Anwendung einer Givensrotation kann jedoch nur ein einzelnes Unterdiagonalelement eliminiert werden und nicht eine gesamte Spalte (wie bei der Householder-Spiegelung).

Schematisch lässt sich das Verfahren wie folgt darstellen:

$$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \to \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \end{pmatrix} \to \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} \to \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}$$

$$\to \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix} \to \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix} \to \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{pmatrix}$$

Wir wollen nun konkret die Givens-Rotation $G(i,j,\theta)$ zur Elimination von a_{ji} angeben. Es gilt

$$\left(G(i,j,\theta)A\right)_{j,i} = \sin(\theta)a_{ii} + \cos(\theta)a_{ji}.$$

Wir suchen also einen Winkel θ, so dass

$$\sin(\theta)a_{ii} + \cos(\theta)a_{ji} = 0 \tag{5.1}$$

gilt. Wir wählen

$$\cos(\theta) := \frac{a_{ii}}{\sqrt{a_{ii}^2 + a_{ji}^2}}, \quad \sin(\theta) := -\frac{a_{ji}}{\sqrt{a_{ii}^2 + a_{ji}^2}}. \tag{5.2}$$

Es ist einfacher direkt $\sin(\theta)$ und $\cos(\theta)$ (statt θ) anzugeben. Man kann leicht überprüfern, dass mit der Wahl (5.2) sowohl (5.1) als auch $\cos^2(\theta) + \sin^2(\theta) = 1$ erfüllt ist.

Für unseren schematisch dargestellten Fall einer 4×4 Matrix berechnet man

$$\underbrace{G(3,4,\theta_6)G(2,3,\theta_5)(2,4,\theta_4)G(1,2,\theta_3)G(1,3,\theta_2)G(1,4,\theta_1)}_{Q^T} A = R.$$

Einmal erzeugte Nulleinträge werden in späteren Schritten nicht durch andere Werte überschrieben.

Bemerkung 5.8. Die Berechnung der QR-Zerlegung mittels Givens-Rotation ist aufwendiger als die Berechnung mittels Householder-Transformation.

Die QR-Zerlegung nach Givens gewinnt aber an Bedeutung, wenn die Matrix A bereits dünn besetzt ist. Dies ist beispielsweise der Fall, wenn man die QR-Zerlegung einer Hessenberg-Matrix berechnen möchte.

Wir führen nun noch anhand eines konkreten Beispiels die QR-Zerlegung mittels Givens-Rotation durch.

Beispiel 5.9. Wir bestimmen die QR-Zerlegung der 3×3 Hessenberg-Matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0.01 & 0 & 0.01 \\ 0 & 0.01 & 0.01 \end{pmatrix}.$$

- Elimination von a_{21} durch Multiplikation mit

$$G(1,2,\theta_1) = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad c := \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}},\ s := -\frac{a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}}$$

$$\begin{aligned} A^{(2)} &= G(1,2,\theta_1)A \\ &= \begin{pmatrix} 0.99995000 & 0.00999950 & 0 \\ -0.00999950 & 0.99995000 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0.01 & 0 & 0.01 \\ 0 & 0.01 & 0.01 \end{pmatrix} \\ &= \begin{pmatrix} 1.00004999 & 0.99995000 & 1.00004999 \\ 0 & -0.00999950 & 0 \\ 0 & 0.01 & 0.01 \end{pmatrix} \end{aligned}$$

- Elimination von $a_{3,2}^{(2)}$ durch Multiplikation mit

$$G(2,3,\theta_2) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix}, \quad c := \frac{a_{22}^{(2)}}{\sqrt{(a_{22}^{(2)})^2 + (a_{32}^{(2)})^2}},\ s := -\frac{a_{32}^{(2)}}{\sqrt{(a_{22}^{(2)})^2 + (a_{32}^{(2)})^2}}$$

$$
\begin{aligned}
A^{(3)} &= G(2,3,\theta_2)A^{(2)} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.70708910 & 0.70712445 \\ 0 & -0.70712445 & -0.70708910 \end{pmatrix} \begin{pmatrix} 1.00004999 & 0.99995000 & 1.00004999 \\ 0 & -0.00999950 & 0 \\ 0 & 0.01 & 0.01 \end{pmatrix} \\
&= \begin{pmatrix} 1.00004999 & 0.99995000 & 1.00004999 \\ 0 & 0.01414178 & 0.00707124 \\ 0 & 0 & -0.00707089 \end{pmatrix} \\
&= R
\end{aligned}
$$

- Berechnung von Q:

$$
\begin{aligned}
Q &= (G(2,3,\theta_2)G(1,2,\theta_1))^T = Q(1,2,\theta_1)^T G(2,3,\theta_2)^T \\
&= \begin{pmatrix} 0.99995000 & 0.00707053 & 0.00707089 \\ 0.00999950 & -0.70705375 & -0.70708910 \\ 0 & 0.70712445 & -0.70708910 \end{pmatrix}
\end{aligned}
$$

5.4 Lösung linearer Gleichungssysteme unter Verwendung der QR-Zerlegung

Die QR-Zerlegung einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ kann genutzt werden um lineare Gleichungssysteme effizient zu lösen:

$$
\begin{aligned}
& Ax = b \\
\Leftrightarrow \quad & QRx = b \\
\Leftrightarrow \quad & Rx = Q^T b.
\end{aligned}
$$

Das Gleichungssystem $Rx = Q^T b$ kann einfach durch Rückwärtssubstitution gelöst werden.

Es gilt

$$\kappa_2(R) = \kappa_2(A),$$

d.h. die Matrix $R = Q^T A$ hat die gleiche Konditionszahl wie die Matrix A . Daher ist die Lösung linearer Gleichungssysteme mittels QR-Zerlegung stabiler als die Lösung mittels LR-Zerlegung. Andererseits erfordert die QR-Zerlegung etwa doppelt so viele Operationen und die Fälle in denen LR-Zerlegung mit Pivotisierung instabil wird sind extrem selten. In der Praxis löst man daher dichtbesetzte Gleichungssysteme typischerweise mittels Gaußelimination (LR-Zerlegung).

5.5 Lineare Ausgleichsrechnung

In mehreren naturwissenschaftlichen Fächern (z.B. Experimentalphysik und Biologie) stellt sich die Aufgabe, unbekannte Parameter einer Funktion, die entweder auf Grund von Naturgesetzen oder Modellannahmen gegeben sind, durch eine Reihe von Messungen oder Beobachtungen zu bestimmen. Die Anzahl der Messungen ist typischerweise sehr viel größer als die Anzahl der zu bestimmenden Parameter. Auf diese Weise verringert man den Einfluß von Beobachtungsfehlern und Messungenauigkeiten. Man bekommt ein überbestimmtes System von linearen oder nichtlinearen Gleichungen für die unbekannten Parameter, das im Allgemeinen nicht exakt lösbar ist.

Hier beschränken wir uns auf den Fall überbestimmter linearer Gleichungssysteme. Diese haben die Form $Ax = b$ mit $A \in \mathbb{R}^{m\times n}$, $m \geq n$ und $b \in \mathbb{R}^m$.

In diesem Abschnitt betrachten wir die von Gauß entwickelte Methode der kleinen Quadrate zur Bestimmung einer (in einem noch zu spezifizierenden Sinn optimalen) Lösung.

Wir werden zwei verschiedene Algorithmen zur Lösung des linearen Ausgleichsproblems kennenlernen. Wie wir es in dieser Vorlesung schon häufiger erlebt haben, ist eine dieser Methoden anfälliger gegenüber Rundungsfehlern und daher für größere Rechnungen (auf Computern) ungeeignet.

5.5.1 Die Normalengleichung

Wir betrachten das überbestimmte lineare Gleichungssystem

$$Ax = b, \quad A \in \mathbb{R}^{m\times n},\ m \geq n,\ b \in \mathbb{R}^m.$$

Es sei $r := Ax - b$ das Residuum. Die i-te Komponente von r hat die Form

$$r_i = \sum_{k=1}^{n} a_{ik}x_k - b_i, \quad i = 1, \dots m.$$

Die Unbekannten $x_1, \dots, x_n$ sollen so bestimmt werden, dass die Summe der Quadrate der Residuen $\sum_{i=1}^{m} r_i^2$ minimal ist.

Definition 5.10. Durch $A \in \mathbb{R}^{m\times n}$, $m \geq n$ und $b \in \mathbb{R}^m$ wird das *Ausgleichsproblem*

$$\|Ax - b\|_2 \stackrel{!}{=} \min$$

definiert.

Für $x \in \mathbb{R}^n$ heißt $r := Ax - b$ das *Residuum* von x.

Wir möchten nun den Vektor $x \in \mathbb{R}^n$ so bestimmen, dass das Quadrat der euklidischen Norm des Residuenvektors r minimal wird. Dabei nehmen wir zusätzlich an, dass die Matrix A maximalen Rang hat, d.h. rank$(A) = n$. Es gilt:

$$\begin{aligned}\|r\|_2^2 &= r^T r \\ &= (Ax-b)^T(Ax-b) \\ &= x^T A^T Ax - x^T A^T b - \underbrace{b^T A}_{(A^T b)^T} x + b^T b \\ &= x^T A^T Ax - 2(A^T b)^T x + b^T b.\end{aligned}$$

Da A maximalen Rang hat, ist die symmetrische Matrix $A^T A$ positiv definit, denn

$$x^T A^T Ax = (Ax)^T(Ax) \geq 0 \quad \text{für } x \in \mathbb{R}^n$$

und

$$x^T A^T Ax = 0 \quad \Leftrightarrow \quad Ax = 0 \quad \Leftrightarrow \quad x = 0.$$

Wir möchten nun also die quadratische Funktion

$$F(x) := r^T r = x^T A^T Ax - 2(A^T b)^T x + b^T b$$

minimieren. Zur Vereinfachung der Notation definieren wir $C := A^T A$ und $d := A^T b$ und erhalten $F(x) = x^T Cx - 2d^T x + b^T b$.

Die notwendige Bedingung dafür, dass $F(x)$ ein Minimum annimmt, lautet

$$\nabla F(x) = 0.$$

In Komponentenschreibweise erhalten wir

$$\frac{\partial F(x)}{\partial x_i} = 2\sum_{k=1}^{n} c_{ik}x_k - 2d_i, \quad i = 1, \ldots, n,$$

wobei c_{ij}, $i, j = 1, \ldots, n$ die Komponenten der Matrix C und d_i die Komponenten des Vektors d beschreibt.

Als notwendige Bedingung für ein Minimum von $F(x)$ erhalten wir das lineare Gleichungssystem

$$\sum_{k=1}^{n} c_{ik}x_k = d_i \quad i = 1, \ldots, n$$

(also $Cx = d$) bzw.

$$A^T Ax = A^T b. \tag{5.3}$$

Man nennt (5.3) die **Normalengleichungen** zu der Fehlergleichung $r = Ax - b$.

Da $A^T A$ positiv definit ist, besitzt die Normalengleichung eine eindeutige Lösung. Diese Lösung können wir mit Hilfe der Cholesky-Zerlegung (vergleiche Abschnitt 4.2) berechnen.

Die Funktion $F(x)$ wird durch die Lösung der Normalengleichung auch tatsächlich minimiert, denn die Hessematrix, gebildet aus den zweiten partiellen Ableitungen, ist die positiv definite Matrix $2A^T A$.

Wir können somit das lineare Ausgleichsproblem $Ax = b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $m \geq n$, $\text{rank}(A) = n$ wie folgt lösen:

1. Berechne $C = A^T A$, $d = A^T b$
2. Bestimme die Cholesky-Zerlegung $C = \tilde{L}\tilde{L}^T$
3. Löse die gestaffelten Probleme

$$\tilde{L}y = d$$
$$\tilde{L}^T x = y$$

   Der Vektor x ist Lösung des Ausgleichsproblems.

Das Ausgleichsproblem wird aufgrund der verwendeten $\|\cdot\|_2$-Norm auch als **Methode der kleinsten Quadrate** oder **Methode der kleinsten Fehlerquadrate** bezeichnet.

Etwas allgemeiner gilt der folgende Satz.

Satz 5.11. *Die Lösung des Ausgleichsproblems sind genau die Lösungen der Normalengleichung*

$$A^T Ax = A^T b.$$

Insbesondere existiert eine Lösung $x \in \mathbb{R}^n$. Ist $z \in \mathbb{R}^n$ eine weitere Lösung, so gilt $Ax = Az$.

Bemerkung. In der Definition des Ausgleichsproblems und in obigem Satz muss A nicht maximalen Rang haben.

Wir wollen Satz 5.11 nun mit Hilfe algebraischer Argumente beweisen. Daher fassen wir zunächst noch einmal ein paar Grundlagen der linearen Algebra zusammen.

Für eine Matrix $A \in \mathbb{R}^{m \times n}$ bzw. die damit identifizierte lineare Abbildung sind ihr Bild und Kern definiert durch

$$\text{Im}A := \{w \in \mathbb{R}^m : \exists v \in \mathbb{R}^n, w = Av\}$$
$$\text{Ker}A := \{v \in \mathbb{R}^n : Av = 0\}$$

Es gilt

$$\mathbb{R}^m = \text{Im}A + \text{Ker}A^T, \quad \mathbb{R}^n = \text{Im}A^T + \text{Ker}A.$$

Die Zerlegungen sind orthogonal, d.h. für $w \in Av \in \text{Im}A$ und $u \in \text{Ker}A^T$ gilt

$$w \cdot u = w^T u = (Av)^T u = v^T A^T u = v^T \mathbf{0} = 0.$$

Damit ist ImA das orthogonale Komplement von kerA^T (Bezeichnung Im$A = (\text{ker}A^T)^\perp$).

Analog gilt für $w = A^T u \in \text{Im}A^T$ und $v \in \text{Ker}A$

$$w \cdot v = w^T v = (A^T u)^T v = u^T A v = u^T \mathbf{0} = 0.$$

Außerdem gilt rank$(A) = \dim \text{Im}A$, d.h. der Rang entspricht der Anzahl der linear unabhängigen Spaltenvektoren.

Nach dieser kurzen Wiederholung können wir nun Satz 5.11 beweisen.

Beweis. Es gilt $\mathbb{R}^m = \text{Im}A + \text{ker}A^T$ und diese Zerlegung ist orthogonal.

Damit existieren zu $b \in \mathbb{R}^m$ eindeutig bestimmte Vektoren $y \in \text{Im}A$ und $r \in \text{Ker}A^T$ mit $y \cdot r = 0$ und $b = y + r$. Außerdem existiert ein $x \in \mathbb{R}^n$ mit $y = Ax$.

Damit folgt

$$A^T b = A^T(y + r) = A^T Ax + \underbrace{A^T r}_{=\mathbf{0}} = A^T Ax,$$

d.h. x löst die Normalengleichung.

Wir wollen nun noch zeigen, dass x Lösung des Ausgleichsproblems ist. Sei $z \in \mathbb{R}^n$ beliebig, $r = b - Ax$ und $A^T r = 0$, dann gilt

$$\begin{aligned}
\|b - Az\|_2^2 &= \|(b - Ax) + A(x - z)\|_2^2 \\
&= ((b - Ax) + A(x - z))^T ((b - Ax) + A(x - z)) \\
&= (b - Ax)^T (b - Ax) + (A(x - z))^T (A(x - z)) \\
&\quad + (b - Ax)^T (A(x - z)) + (A(x - z))^T (b - Ax) \\
&= \|b - Ax\|_2^2 + \|A(x - z)\|_2^2 + \underbrace{r^T A(x - z)}_{(x-z)^T A^T r} + (x - z)^T A^T r \\
&= \|b - Ax\|_2^2 + \|A(x - z)\|_2^2 + 2\underbrace{(A^T r)}_{=0} \cdot (x - z) \\
&= \|b - Ax\|_2^2 + \|A(x - z)\|_2^2 \\
&\geq \|b - Ax\|_2^2
\end{aligned}$$

Also löst x das Ausgleichsproblem. Gleichheit liegt genau dann vor, wenn $A(x - z) = 0$ gilt. In diesem Fall ist z weitere Lösung des Ausgleichsproblems. □

5.5.2 Lösung des Ausgleichsproblems unter Verwendung der QR-Zerlegung

Die Lösung des Ausgleichsproblems unter Verwendung der Normalengleichung ist anfällig gegenüber Rundungsfehler. In diesem Abschnitt lernen wir eine andere Möglichkeit kennen, mit der das Ausgleichsproblem gelöst werden kann. Diese Methode ist weniger anfällig gegenüber Rundungsfehlern und sollte daher verwendet werden um Ausgleichsprobleme unter Verwendung von Computern zu lösen.

Satz 5.12. *Es sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$. Weiter sei der Rang von A maximal, also* $\operatorname{rank}(A) = n$. *Außerdem besitze A die QR-Zerlegung $A = QR$ mit $Q \in \mathbb{R}^{m\times m}$ orthogonal und $R \in \mathbb{R}^{m\times n}$ obere Dreiecksmatrix. Es seien*

$$R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}, \quad \hat{R} \in \mathbb{R}^{n\times n}, \quad Q^T b = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad y_1 \in \mathbb{R}^n, \quad y_2 \in \mathbb{R}^{m-n}.$$

Dann löst x^* das lineare Ausgleichsproblem $\|Ax - b\|_2^2 \stackrel{!}{=} \min$ für alle $x \in \mathbb{R}^n$ genau dann, wenn $\hat{R}x^* = y_1$ erfüllt ist.

Beweis. Für $z \in \mathbb{R}^m$ gilt

$$\|Qz\|_2^2 = (Qz)^T(Qz) = z^T Q^T Q z = z^T z = \|z\|_2^2. \qquad (*)$$

Für $x \in \mathbb{R}^n$ gilt

$$\begin{aligned} \|Ax - b\|_2^2 &= \|QRx - QQ^T b\|_2^2 \\ &= \|Q(Rx - Q^T b)\|_2^2 \\ &\stackrel{(*)}{=} \|Rx - Q^T b\|_2^2 \\ &= \|\hat{R}x - y_1\|_2^2 + \|y_2\|_2^2. \end{aligned}$$

Also gilt für alle $x \in \mathbb{R}^n$ gerade

$$\|Ax - b\|_2 \geq \|y_2\|_2$$

und Gleichheit wird genau dann angenommen, wenn $\hat{R}x = y_1$ erfüllt ist. □

Aus Satz 5.12 ergibt sich der folgende Algorithmus zur Lösung linearer Ausgleichsprobleme.

Unter Verwendung der QR-Zerlegung können wir das Ausgleichsproblem $Ax = b$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}$, $m \geq n$, $\operatorname{rank}(A) = n$ wie folgt lösen:

1. Berechne die reduzierte QR-Zerlegung $A = \hat{Q}\hat{R}$.

2. Berechne den Vektor $\hat{Q}^T b$.

3. Löse das lineare Gleichungssystem

$$\hat{R}x = \hat{Q}^T b \qquad \text{nach } x.$$

Der Vektor x ist die Lösung des Ausgleichsproblems.

Beispiel 5.13. Wir betrachten das Modell $g(t) = \alpha \cos t + \beta \sin t$. Gegeben seien die folgenden Messwerte:

t	0	$\pi/2$	π	$3\pi/2$
y	$0{,}25$	$0{,}8$	$-0{,}1$	$-0{,}75$

Mit

$$A = \begin{pmatrix} \cos(0) & \sin(0) \\ \cos\left(\frac{\pi}{2}\right) & \sin\left(\frac{\pi}{2}\right) \\ \cos(\pi) & \sin(\pi) \\ \cos\left(\frac{3}{2}\pi\right) & \sin\left(\frac{3}{2}\pi\right) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 0{,}25 \\ 0{,}8 \\ -0{,}1 \\ -0{,}75 \end{pmatrix}$$

folgt als Lösung $\alpha = 0{,}175$ und $\beta = 0{,}775$.

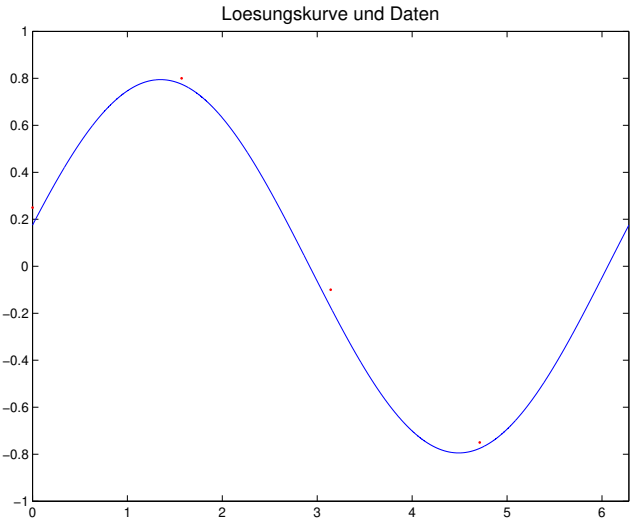


Abbildung 5.1: Lösungskurve und Daten für Beispiel 5.13

Beispiel 5.14. Bei einem physikalischen Gesetz gelte zwischen der Zeit t und dem gemessenen Ergebnis $E = E(t)$ ein linearer Zusammenhang $E(t) = c + d \cdot t$ mit unbekannten Konstanten $c, d \in \mathbb{R}$. Zu den Zeiten $t_1 < t_2 < \cdots < t_m$ liegen Messwerte $E_i = E(t_i)$ für $i = 1, \ldots, m$ vor. Im Allgemeinen erhält man nicht E_i, sondern $E_i + \varepsilon_i = e_i$ mit betragsmäßig kleinem Fehler ε_i.

Zur Bestimmung von c und d wählen wir $g(t) = c + d \cdot t$, so dass

$$\sum_{i=1}^{m} (g(t_i) - e_i)^2$$

minimal wird. Definieren wir nun

$$A := \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix}, \quad x := \begin{pmatrix} c \\ d \end{pmatrix}, \quad b := \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix},$$

so gilt

$$\sum_{i=1}^{m} (g(t_i) - e_i)^2 = \|Ax - b\|_2^2.$$

Gesucht ist also ein $x^* \in \mathbb{R}^2$ mit

$$\|Ax^* - b\|_2 \leq \|Ax - b\|_2 \quad \text{für alle} \quad x \in \mathbb{R}^2.$$

Im linken Bild von Abbildung 5.2 ist eine Ausgleichsgerade zu gegebenen Datenpunkten zu sehen.

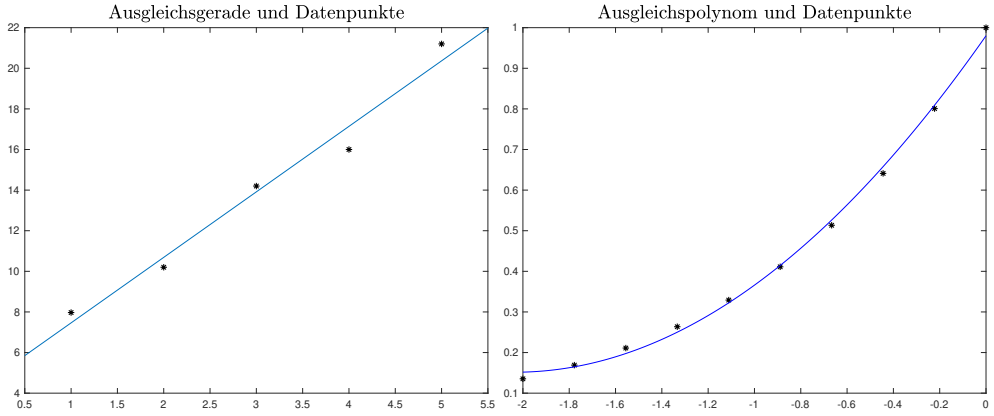


Abbildung 5.2: Ausgleichsgerade (links) und Ausgleichspolynom 2. Grades (rechts) zu gegebenen Datenpunkten.

Es sei allgemeiner $p(t) = c_0 + c_1 t + \cdots + c_{n.1} t^{n-1}$ ein Polynom vom Grad $n - 1$. Dann setzen wir

$$A := \begin{pmatrix} 1 & t_1 & \cdots & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & t_m & \cdots & t_m^{n-1} \end{pmatrix} \in \mathbb{R}^{m \times n}, \quad x := \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix} \in \mathbb{R}^n, \quad b := \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{pmatrix} \in \mathbb{R}^m.$$

Es gilt typischerweise $m \gg n$. Auch hier liegt die Aufgabe darin, ein x zu finden, so dass $\|Ax - b\|_2^2$ minimal wird.

Im rechten Bild von Abbildung 5.2 ist ein Polynom 2. Grades zu sehen, dass das Ausgleichsproblem zu den gegebenen Datenpunkten löst.

Weitere Beispiele für lineare Ausgleichsprobleme wurden in VL19 diskutiert, siehe VL19.pdf und das zugeghörige Python File.

6 Nichtlineare Gleichungen

Eine weitere grundlegende Problemstellung der Mathematik, für deren Lösung numerische Verfahren benötigt werden werden, ist die Bestimmung von Nullstellen einer gegebenen Funktion. Dies entspricht der Lösung einer nichtlinearen Gleichung bzw. der Lösung eines nichtlinearen Gleichungssystems.

In Anwendungen ist die Lösung einer nichtlinearen Gleichung oder eines Systems von nichtlinearen Gleichungen oft eine Teilaufgabe einer komplexeren Problemstellung. Eine exakte analytische Lösung nichtlinearer Gleihungen ist nur für relativ einfache Gleichungen möglich. Daher verwendet man im Allgemeinen iterative Verfahren, die eine Lösung als Grenzwert einer Folge von Näherungen liefern. Als theoretische Grundlage nutzen wir den Banachschen Fixpunktsatz.

6.1 Theoretische Grundlagen

Gegeben sei eine stetige Funktion $f : \Omega \to \mathbb{R}^n$ mit $\Omega \subseteq \mathbb{R}^n$ nichtleer und abgeschlossen. Gesucht ist eine Nullstelle von $f(x)$, d.h. ein Vektor $x^* \in \mathbb{R}^n$ mit

$$f(x^*) = 0.$$

Um dieses Problem der iterativen Lösung zugänglich zu machen betrachtet man ein äquivalentes Fixpunktproblem

$$x = F(x),$$

für das gilt

$$f(x^*) = 0 \quad \Leftrightarrow \quad x^* = F(x^*).$$

Wir werden Iterationsverfahren der Form

$$x_{k+1} = F(x_k), \quad k = 0, 1, 2, \ldots \tag{6.1}$$

betrachten. Diese Verfahren benötigen einen Startwert x_0 und liefern dann eine Folge von Iterierten. Die Iterationsvorschrift (6.1) nennt man Fixpunktiteration.

Als mathematisches Werkzeug bei der Untersuchung von Fixpunktiterationen dient uns der Banachsche Fixpunktsatz.

Definition 6.1. Eine Abbildung $F : \mathbb{R}^n \to \mathbb{R}^n$ heißt Kontraktion bezüglich einer Norm $\|\cdot\|$ auf $\mathbb{R}^n$, wenn es eine Zahl $\kappa < 1$ gibt, sodass für alle $x, y \in \mathbb{R}^n$ gilt

$$\|F(x) - F(y)\| \leq \kappa \|x - y\|.$$

Satz 6.2 (Banachscher Fixpunktsatz). *Es sei $(X, \|\cdot\|)$ ein normierter, linearer Raum und $Y \subseteq X$ vollständig. Weiter sei $F : Y \to Y$ eine kontrahierende Abbildung mit Kontraktionszahl $\kappa < 1$, das heißt*

$$\|F(x) - F(y)\| \leq \kappa \|x - y\|, \quad \textit{für alle} \quad x, y \in Y.$$

Dann gibt es genau einen Fixpunkt $y^* \in Y$, das heißt $y^* = F(y^*)$. Dieser Fixpunkt kann iterativ beliebig genau approximiert werden. Dazu sei $y_0 \in Y$ ein beliebiger Startwert und $(y_i)_{i \in \mathbb{N}_0}$ die durch $y_{i+1} = F(y_i)$ definierte Folge. Dann konvergiert (y_i) gegen den Fixpunkt y^* und es gelten die Abschätzungen

$$\|y^* - y_i\| \leq \frac{\kappa^i}{1 - \kappa} \|y_0 - y_1\|, \qquad \text{(a-priori Abschätzung)}$$

$$\|y^* - y_i\| \leq \frac{\kappa}{1 - \kappa} \|y_i - y_{i-1}\|. \qquad \text{(a-posteriori Abschätzung)}$$

Der Beweis ist Ihnen sicherlich aus der Analysis bekannt. Er wurde in der Vorlesung ausgeführt, siehe VL20.pdf.

Aus der a-priori Abschätzung lässt sich bestimmen, wie viele Iterationen höchstens benötigt werden, um eine vorgegebene Fehlertoleranz zu erreichen.

Je kleiner die Kontraktionszahl ist, desto besser ist die Konvergenz der Folge der Iterationen gegen den Fixpunkt.

Mit der a-posteriori Fehlerabschätzung kann nach i ausgeführten Iterationen die Abweichung von y_i gegenüber y^* abgeschätzt werden.

Die Aussage von Satz 6.2 hat in vielen praktischen Anwendungen nur lokalen Charakter, denn die abgeschlossene Teilmenge Y kann sehr klein sein. Außerdem ist es häufig schwierig die Menge Y quantitativ zu beschreiben und man muss sich mit der Existenz zufrieden geben.

Zur Beurteilung der Konvergenzgeschwindigkeit einer Fixpunktiteration definieren wir den Begriff der Konvergenzordnung einer Folge.

Definition 6.3. Die Folge $(x_k)_{k \in \mathbb{N}}$ mit dem Grenzwert x^* hat mindestens die Konvergenzordnung $p \geq 1$, wenn es eine von i unabhängige Konstante $C > 0$ gibt, so dass

$$\|x_{i+1} - x^*\| \leq C \|x_i - x^*\|^p \quad \forall i \geq 0$$

mit $C < 1$, falls $p = 1$ und $C < \infty$ sonst. C wird Fehlerkonstante genannt.

Im Fall $p = 1$ sprechen wir von *linearer Konvergenz*, für $p = 2$ von *quadratischer Konvergenz*. Die Folge heißt *superlinear* konvergent, falls es eine nichtnegative Nullfolge $c_i \geq 0$ mit $\lim_{k\to\infty} c_i = 0$ gibt, so dass

$$\|x_{i+1} - x^*\| \leq c_i \|x_i - x^*\|$$

für $i \to \infty$ gilt.

Bemerkung. Manchmal definiert man die Konvergenzordnung p aus Gründen einer einfacheren Analysis alternativ durch die analogen Ungleichungen für die Iterierten

$$\|x_{i+1} - x_i\| \leq C \|x_i - x_{i-1}\|^p$$

bzw.

$$\|x_{i+1} - x_i\| \leq c_i \|x_i - x_{i-1}\|^p.$$

Wir verwenden Definition 6.3.

6.2 Nichtlineare Gleichungen in einer Unbekannten

Wir möchten zu einer nichtlinearen Funktion $f(x)$ mit $f : \mathbb{R} \to \mathbb{R}$ Lösungen der Gleichung

$$f(x) = 0 \tag{6.2}$$

berechnen. Wir behandeln den Fall reeller Lösungen.

6.2.1 Das Bisektionsverfahren

Das einfachste Verfahren zur Berechnung der Nullstelle einer stetigen Funktion $f : \mathbb{R} \to \mathbb{R}$ ist die Bisektion. Dabei bestimmt man systematisch kleiner werdende Intervalle, in denen eine reelle Nullstelle von (6.2) liegt. Ausgangspunkt ist ein Intervall $I = [a, b]$, so dass $f(a) \cdot f(b) < 0$ gilt. Aus Stetigkeitsgründen existiert eine Lösung $x^* \in (a, b)$ mit $f(x^*) = 0$. Für den Mittelpunkt $m = (a + b)/2$ wird nun der Funktionswert $f(m)$ berechnet. Ist $f(m) \neq 0$, entscheidet das Vorzeichen in welchem der beiden Teilintervalle $[a, m]$ oder $[m, b]$ die gesuchte Lösung x^* liegt. Die Lage der Lösung x^* ist auf das Innere eines gegenüber I halb so langen Intervalls eingeschränkt. Das Verfahren wird so lange fortgeführt, bis x^* im Inneren eines hinreichend kleinen Intervalls liegt oder bis eine Nullstelle gefunden wurde. Der Mittelpunkt x_i des Intervalls nach i Intervallhalbierungen stellt für x^* eine Nährung dar. Es gilt die a-priori Fehlerabschätzung

$$|x_i - x^*| \leq \frac{b-a}{2^{i+1}}, \quad i = 0, 1, 2, \ldots.$$

Da diese Fehlerabschätzung wie eine geometrische Folge mit dem Quatienten $\frac{1}{2}$ abnimmt, ist die Konvergenzordnung $p = 1$.

Während das Bisektionsverfahren nur in einer Dimension sinnvoll ist, wollen wir uns nun mit der Herleitung von Verfahren beschäftigen, für die Verallgemeinerungen auf den mehrdimensionalen Fall möglich sind. Trotzdem beschränken wir uns zunächst weiterhin auf die Betrachtung nichtlinearer Funktionen $f : \mathbb{R} \to \mathbb{R}$.

6.2.2 Möglichkeiten zur Bestimmung von Iterationsfunktionen

Es sei $I \subset \mathbb{R}$ ein abgeschlossenes Intervall und $f : I \to \mathbb{R}$ eine stetige, eventuell mehrmals stetig differenzierbare Funktion. Wir setzen voraus, dass es ein x^* gibt mit $f(x^*) = 0$. Gesucht ist dann eine kontrahierende Funktion $F : J \to J$, $J \subseteq I$, mit Fixpunkt x^*. Die Iteration mit dieser Funktion konvergiert dann nach dem Banachschen Fixpunktsatz gegen x^*. Der Fixpunkt wir durch Iteration mittels F bestimmt, das heißt durch die Folge $x_{i+1} = F(x_i)$ mit $x^* = \lim\limits_{i\to\infty} x_i$.

Für stetig differenzierbare Funktionen $F : J \to \mathbb{R}$ können wir die Lipschitzkonstante / Kontraktionszahl leicht berechnen.

Satz 6.4. *Ist F eine stetig differenzierbare reellwertige Funktion auf einem abgeschlossenen Intervall $[a, b]$, so ist F genau dann kontrahierend in $[a, b]$, wenn die Ableitung von F in $[a, b]$ dem Betrag nach kleiner als 1 ist.*

Beweis. Folgt aus dem Mittelwertsatz. □

Wir betrachten nun zwei Möglichkeiten zur Bestimmung einer geeigneten Funktion F.

1) Es gilt $f(x^*) = 0$. Das ist gleichbedeutend mit $x^* = x^* - cf(x^*)$ für $c \neq 0$. Wir können also

$$F(x) = x - cf(x)$$

betrachten. Falls f stetig differenzierbar ist, so ist auch F stetig differenzierbar und F ist genau dann kontrahierend in einem abgeschlossenen Intervall $J \subset I$, wenn

(a) $F : J \to J$,

(b) $|F'(x)| \leq \kappa < 1$ für alle $x \in J$ für ein $0 \leq \kappa < 1$.

Die zweite Bedingung bedeutet $|1 - cf'(x)| \leq \kappa$ für alle $x \in J$ und das ist erfüllbar, wenn $0 < cf'(x) < 2$ für alle $x \in J$ gilt.

Zur Ermittlung der Nullstelle wird dann die Iterationsfunktion F verwendet. Wir setzen

$$x_{i+1} = F(x_i) = x_i - cf(x_i) \Leftrightarrow f(x_i) + \frac{1}{c}(x_{i+1} - x_i) = 0,$$

das heißt x_{i+1} ist Nullstelle der Geradengleichung

$$g(x) = f(x_i) + \frac{1}{c}(x - x_i).$$

2) Anstelle einer festen Konstante c zur Bestimmung von F wie im obigen Verfahren, wird eine stetig differenzierbare Funktion $c(x)$ verwendet. Wir nehmen auch an, dass f stetig differenzierbar in I ist mit $f'(x) \neq 0$ für alle $x \in I$. Dann ist analog zu oben $F(x) = x - c(x)f(x)$. Wir setzen erneut

$$x_{i+1} = F(x_i) = x_i - c(x_i)f(x_i).$$

Es gilt $F'(x) = 1 - c'(x)f(x) - c(x)f'(x)$. Wir wählen $c(x)$ nun so, dass $F'(x^*) = 0$ (kleine Kontraktionskonstante in der Nähe von x^*, das heißt die Iteration mit F konvergiert in der Nähe von x^* schnell). Es folgt also

$$0 = F'(x^*) = 1 - c'(x^*)f(x^*) - c(x^*)f'(x^*) = 1 - c(x^*)f'(x^*),$$

da $f(x^*) = 0$ nach Voraussetzung. Damit folgt

$$c(x^*) = \frac{1}{f'(x^*)}.$$

Insgesamt erhalten wir damit

$$F(x) = x - \frac{f(x)}{f'(x)},$$

also

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

Diese Iterationsvorschrift wird auch als Newton-Verfahren bezeichnet.

6.2.3 Das Newton-Verfahren in einer Dimension

Geometrische Interpretation, vergl. mit Abb. 6.1:

Die Tangente von f im Punkt x_i ist gegeben durch

$$T(x) = f'(x_i)(x - x_i) + f(x_i).$$

Nun sei x_{i+1} die Nullstelle der Tangente, das heißt es gilt

$$0 = f'(x_i)(x_{i+1} - x_i) + f(x_i),$$

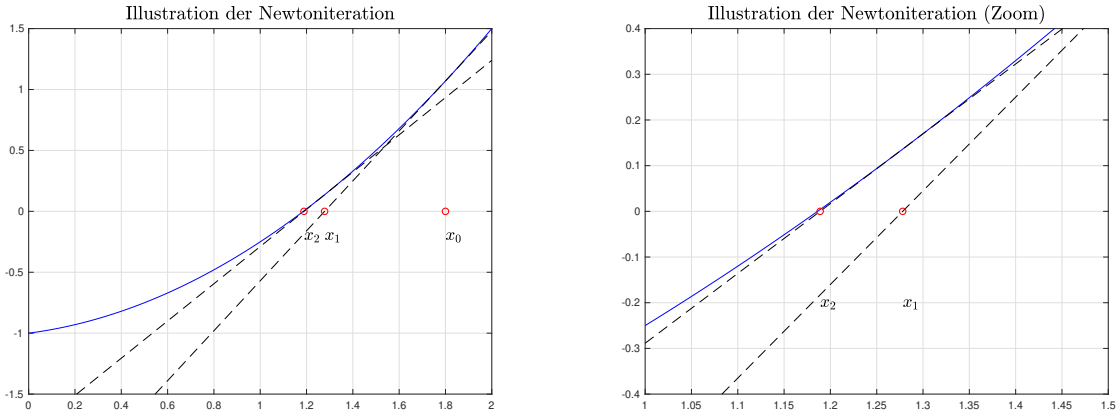


Abbildung 6.1: Darstellung der Newton Iteration.

also

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

Satz 6.5. *Es sei I ein offenes und nicht leeres Intervall. Weiter sei $f : I \to \mathbb{R}$ zweimal stetig differenzierbar mit $f'(x) \neq 0$ für alle $x \in I$. Außerdem erfülle $x^* \in I$ die Gleichung $f(x^*) = 0$. Dann gibt es ein abgeschlossenes Intervall $J \subset I$ mit $x^* \in J$, so dass für das Newton-Verfahren*

$$F(x) = x - \frac{f(x)}{f'(x)}, \quad x \in J$$

die Voraussetzungen des Banachschen Fixpunktsatzes erfüllt sind. Folglich konvergiert jede Folge $(x_i)_{i\in\mathbb{N}_0}$ mit $x_0 \in J$ gegen x^*.

Beweis. Es gilt

$$\begin{aligned} F'(x) &= 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} \\ &= \frac{f(x)f''(x)}{(f'(x))^2} \end{aligned}$$

Aus $f(x^*) = 0$ folgt $F'(x^*) = 0$.

Weiterhin folgt aus der Stetigkeitsannahme von f die Stetigkeit von F'. D.h., es existiert ein $\delta > 0$, so dass in $J := [x^* - \delta, x^* + \delta] \subset I$ gilt

$$|F'(x)| \leq \kappa < 1 \quad \forall x \in J$$

zu zeigen: $F : J \to J$ (selbstabbildend)

Sei $x \in J$, dann gilt

$$\begin{aligned} |F(x) - x^*| &= |F(x) - F(x^*)| \\ &\overset{MWS}{=} |F'(\xi)| \cdot |x - x^*| \quad \text{für ein } \xi \in J \\ &\leq \kappa |x - x^*| \\ &\leq \kappa\delta < \delta. \end{aligned}$$

Also folgt $F(x) \in J$ für alle $x \in J$.

zu zeigen: F ist kontrahierend in J

Für $x, y \in J$ gilt

$$|F(x) - F(y)| \overset{MWS}{=} |F'(\xi)| \cdot |x - y| \leq \kappa |x - y|$$

mit $\xi = \xi(x, y) \in J$.

Da J ein abgeschlossenes Intervall in $\mathbb{R}$ ist, ist J vollständig. Also sind alle Voraussetzungen des Banachschen Fixpunktsatzes erfüllt.

Jede Folge $(x_i)_{i \in \mathbb{N}_0}$ mit $x_0 \in J$ konvergiert also gegen x^*, den eindeutigen Fixpunkt von F in J. □

Bemerkung. Das Newton-Verfahren konvergiert gut, wenn der Startwert x_0 nahe an x^* liegt. Man spricht hier auch von *lokaler Konvergenz*. Andernfalls konvergiert das Verfahren im Allgemeinen schlecht oder gar nicht. Das Newton-Verfahren ist somit ein lokal konvergentes Verfahren.

In Abbildung 6.2 zeigen wir ein Beispiel bei dem das Newton Verfahren nicht Konvergiert. Das Verfahren springt zwischen zwei Werten hin und her. Welche Voraussetzung aus Satz 6.5 ist verletzt?

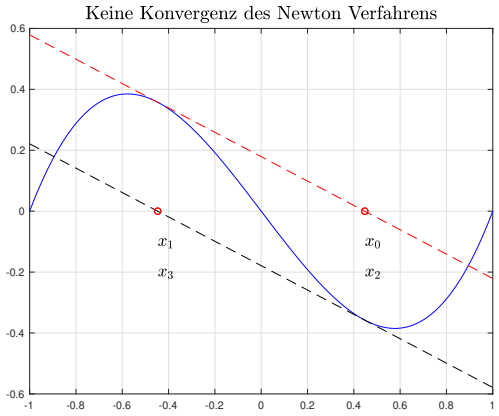


Abbildung 6.2: Keine Konvergenz beim Newton Verfahren

Beispiel 6.6. Wir wollen $x^* = \sqrt[n]{a} > 0$ mit $a > 0$ berechnen. Das heißt wir suchen eine Nullstelle von $f(x) = x^n - a$. Es gilt $f'(x) = nx^{n-1}$. Das Newton-Verfahren hat also die Iterationsvorschrift

$$x_{i+1} = x_i - \frac{x_i^n - a}{nx_i^{n-1}} = \frac{1}{n}\left((n-1)x_i + \frac{a}{x_i^{n-1}}\right).$$

Für $n = 2$ erhalten wir exemplarisch

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{a}{x_i}\right).$$

Für Startwerte $x_0 > \sqrt{2}$ folgt

$$x_{i+1} - \sqrt{a} = \frac{1}{2x_i}\left(x_i^2 + a - 2x_i\sqrt{a}\right) = \frac{1}{2x_i}(x_i - \sqrt{a})^2 \geq 0,$$

das heißt die Folge der x_i monoton fallend, denn

$$0 \leq \frac{x_i - \sqrt{a}}{2x_i} \leq \frac{1}{2} \quad \text{für} \quad x_0 > 0$$

und damit

$$x_{i+1} - \sqrt{a} < x_i - \sqrt{a}.$$

Außerdem gilt $x_i > \sqrt{a}$ für $i \geq 1$. Daraus ergibt sich insgesamt

$$x_{i+1} - \sqrt{a} \leq \frac{1}{2\sqrt{a}}(x_i - \sqrt{a})^2,$$

also quadratische Konvergenz (vergl. mit Definition 6.3).

Als kleines Rechenbeispiel sei $x^* = \sqrt{2}$. Dann gilt

$$\begin{aligned}
x_0 &= 2, \\
x_1 &= \underline{1},5, \\
x_2 &= \underline{1,416}, & |x_2 - \sqrt{2}| &\leq 5 \cdot 10^{-3}, \\
x_3 &= \underline{1,41421}568, & |x_3 - \sqrt{2}| &\leq 5 \cdot 10^{-6}, \\
x_4 &= \underline{1,41421356137}469, & |x_4 - \sqrt{2}| &\leq 5 \cdot 10^{-12}.
\end{aligned}$$

Wir werden nun noch ein Kriterium zur Bestimmung der Konvergenzordnung kennenlernen.

Satz 6.7. *Sei $I \subset \mathbb{R}$ ein Intervall und F eine reellwertige p-mal stetig differenzierbare Iterationsfunktion mit dem Fixpunkt x^*. Gilt*

$$|F'(x^*)| < 1 \quad \textit{für } p = 1$$

und

$$F'(x^*) = \ldots = F^{(p-1)}(x^*) = 0,\ F^{(p)}(x^*) \neq 0 \textit{ für } p > 1,$$

so ist das Iterationsverfahren $x_{i+1} = F(x_i)$ lokal konvergent gegen x^* und hat die Ordnung p.

Beweis. Taylorentwicklung von F um x^* mit Lagrange-Restglied liefert:

$$\begin{aligned} x_{i+1} &= F(x_i) \\ &= \underbrace{F(x^*)}_{x^*} + \underbrace{F'(x^*)}_{0}(x_i - x^*) + \ldots + \frac{1}{(p-1)!}\underbrace{F^{(p-1)}}_{0}(x^*)(x_i - x^*)^{p-1} \\ &\quad + \frac{1}{p!}F^{(p)}(\xi_i)(x_i - x^*)^p \end{aligned}$$

mit ξ_i zwischen x^* und x_i.

Daraus folgt

$$x_{i+1} - x^* = \frac{1}{p!}F^{(p)}(\xi_i)(x_i - x^*)^p$$

sowie

$$\frac{|x_{i+1} - x^*|}{|x_i - x^*|^p} = \frac{1}{p!}|F^{(p)}(\xi_i)|.$$

$F^{(p)}$ ist nach Voraussetzung in einer kompakten Umgebung von x^* beschränkt. □

Korollar 6.8. *Ist x^* eine einfache Nullstelle einer Funktion $f \in C^3(I, \mathbb{R})$, so ist das Newton-Verfahren lokal gegen x^* konvergent und hat mindestens die Ordnung 2.*

Beweis. Es gilt $F(x) = x - \frac{f(x)}{f'(x)}$, $F \in C^2(I, \mathbb{R})$,

$$\begin{aligned} F'(x) &= 1 - \frac{(f'(x))^2 - f''(x)f(x)}{(f'(x))^2} \quad f'(x^*) \neq 0 \text{ (einfache Nullstelle)} \\ &= \frac{f''(x)f(x)}{(f'(x))^2}. \end{aligned}$$

Aus $f(x^*) = 0$ folgt $F'(x^*) = 0$ und mit Satz 6.7 folgt die Behauptung. □

6.2.4 Konstruktion eines Fixpunktverfahrens 3. Ordnung

Wir konstruieren ein Fixpunktverfahren 3. Ordnung. Das heißt nach Satz 6.7 muss $F'(x^*) = F''(x^*) = 0$ sein.

Wir machen dazu den Ansatz

$$F(x) = x - r(x) + s(x)r(x)^2, \quad r(x) = \frac{f(x)}{f'(x)}.$$

Es gilt $r(x^*) = 0$ und

$$r'(x) = \frac{(f'(x))^2 - f''(x)f(x)}{(f'(x))^2}$$

und somit $F'(x^*) = 0$. Wir konstruieren $s(x)$ nun so, dass $F''(x^*) = 0$ gilt. Dazu können wir etwa

$$s(x) = \frac{r''(x)}{2(r'(x))^2}$$

wählen.

Beispiel 6.9. Wir wenden das Verfahren an um $x^* = \sqrt{2}$ als Nullstelle von $f(x) = x^2 - 2$ zu berechnen. Bei Verwendung des Startwertes $x_0 = 2$ erhalten wir die Werte

$$\begin{aligned}
x_0 &= 2, \\
x_1 &= \underline{1,4}44444444444444, && |x_1 - \sqrt{2}| \approx 3{,}1 \cdot 10^{-2} \\
x_2 &= \underline{1,414220287227105} && |x_2 - \sqrt{2}| \approx 6{,}7 \cdot 10^{-6}, \\
x_3 &= \underline{1,414213562373095} && |x_3 - \sqrt{2}| = 0.
\end{aligned}$$

6.2.5 Mehrfache Nullstellen

Sei $f \in C^2(I, \mathbb{R})$ und sei x^* zweifache Nullstelle der Funktion f, d.h.

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

Die Voraussetzung $f'(x) \neq 0 \; \forall x \in I$ aus Satz 6.5 ist also nicht erfüllt.

Dann gilt:

$$\begin{aligned}
x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\
&= x_i - \frac{f(x_i) - f(x^*)}{f'(x_i) - f'(x^*)} \\
&\overset{MWS}{=} x_i - \frac{f'(\xi_i)}{f''(\eta_i)}
\end{aligned}$$

mit Zwischenstellen ξ_i, η_i zwischen x_i und x^*. D.h., $\frac{f(x_i)}{f'(x_i)}$ bleibt für $x_i \to x^*$ wohl definiert.

Sei x^* p-fache Nullstelle der Funktion $f \in C^{p+1}(I, \mathbb{R})$, also

$$f(x^*) = f'(x^*) = \ldots = f^{(p-1)}(x^*) = 0, \quad f^{(p)}(x^*) \neq 0.$$

Dann gilt (Taylor)

$$f(x) = \underbrace{f(x^*)}_{0} + \ldots + \frac{1}{(p-1)!} \underbrace{f^{(p-1)}(x^*)}_{0} (x - x^*)^{p-1} + \underbrace{\frac{(x - x^*)^p}{p!} f^{(p)}(\xi)}_{(x-x^*)^p Q(x^*;x)}$$

und

$$f'(x) = p(x-x^*)^{p-1}Q(x^*;x) + (x-x^*)^p Q'(x^*;x).$$

Also erhalten wir für $f'(x) \neq 0$:

$$\begin{aligned}\frac{f(x)}{f'(x)} &= \frac{(x-x^*)Q(x^*;x)}{Q'(x^*;x)(x-x^*) + pQ(x^*;x)} \\ &= \frac{(x-x^*)}{p} - \frac{1}{p}(x-x^*)^2 \frac{Q'(x^*;x)}{Q'(x^*;x)(x-x^*) + pQ(x^*;x)}\end{aligned}$$

Für den Iterationsansatz

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

folgt

$$\begin{aligned}x_{i+1} - x^* &= x_i - x^* - \alpha \frac{f(x_i)}{f'(x_i)} \\ &= (x_i - x^*)\left(1 - \frac{\alpha}{p}\right) + (x_i - x^*)^2 \frac{\alpha Q'(x^*;x_i)}{pQ'(x^*;x_i)(x_i - x^*) + p^2 Q(x^*;x_i)}\end{aligned}$$

Für $\alpha = p$ erhält man das quadratisch konvergente Verfahren

$$x_{i+1} = x_i - p\frac{f(x_i)}{f'(x_i)}.$$

6.2.6 Die Sekantenmethode

Soll die Berechnung der Ableitung der Funktion f vermieden werden, so kann man statt der Nullstelle der Tangente, die das Newton-Verfahren verwendet, die Nullstelle einer Sekante berechnen. Dazu werden zwei Startwerte x_0 und x_1 benötigt, die aber nicht wie bei der Bisektion die gesuchte Nullstelle einschließen müssen. Der iterierte Wert x_{i+1} ist die Nullstelle der Sekante durch die Punkte $(x_i, f(x_i))$ und $(x_{i-1}, f(x_{i-1}))$. Die Sekante hat die Form

$$s(x) = f(x_i) + (x - x_i)\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}.$$

Aus der Forderung $s(x_{i+1}) = 0$ erhält man die Sekantenmethode

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}.$$

Die Sekantenmethode ist ein zweistufiges Iterationsverfahren. In jedem Iterationsschritt muss $f(x_i) - f(x_{i-1}) \neq 0$ erfüllt sein. Abbildung 6.3 zeigt eine graphische Darstellung dieser Methode.

Man kann zeigen, dass für $f(x^*) \neq 0$ und $f''(x^*) \neq 0$ die Sekantenmethode lokal konvergiert und die Konvergenzordnung $p = (1+\sqrt{5})/2 \approx 1{,}618$ besitzt.

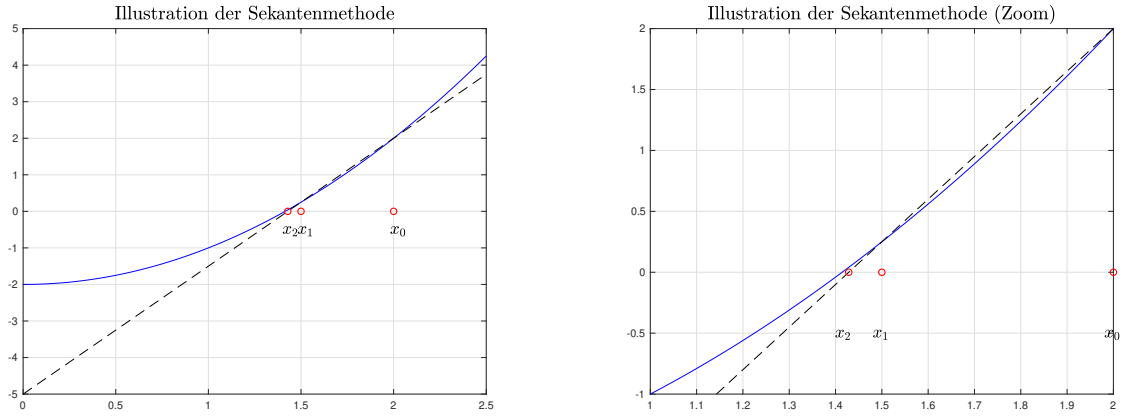


Abbildung 6.3: Darstellung der Sekantenmethode.

Beispiel 6.10. Wir wenden nun auch die Sekantenmethode an, um $x^* = \sqrt{2}$ als Nullstelle von $f(x) = x^2 - 2$ zu berechnen. Bei Verwendung des Startwertes $x_0 = 2$, $x_1 = 1.5$ erhalten wir die Werte

$$
\begin{aligned}
x_0 &= 2, \\
x_1 &= \underline{1,}5, \\
x_2 &= \underline{1,}428571428571429 && |x_2 - \sqrt{2}| \approx 1.43 \cdot 10^{-2}\,, \\
x_3 &= \underline{1,414}634146341463 && |x_3 - \sqrt{2}| \approx 4.2 \cdot 10^{-4} \\
x_4 &= \underline{1,414}215686274510 && |x_4 - \sqrt{2}| \approx 2.12 \cdot 10^{-6} \\
x_5 &= \underline{1,41421356}2688870 && |x_5 - \sqrt{2}| \approx 3.15 \cdot 10^{-10} \\
x_6 &= \underline{1,41421356237309}5 && |x_6 - \sqrt{2}| \approx 0
\end{aligned}
$$

Bemerkung. In der Praxis kombiniert man die Sekantenmethode mit der Bisektion. Man verwendet auch Kombinationen aus Sekantenmethode, Bisektion und der inversen quadratischen Interpolation, bei der eine Parabel $P(y)$ durch drei Punkte berechnet wird. $P(0)$ ist dann die neue Iterierte. All diese Methoden benötigen keine Kenntnis der Ableitung $f'(x)$.

Auf ein explizites Überprüfen der Kontraktionsbedingung wird in der Praxis verzichtet. Stattdessen überprüft man, ob die neue Iterierte innerhalb eines zuvor festgelegten Intervalls liegt. Falls dies nicht der Fall ist, so verwendet man Bisektion.

6.3 Das Newton Verfahren für nichtlineare Gleichungssysteme

Wir betrachten nun nichtlineare Gleichungssysteme der allgemeinen Form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

mit $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ stetig differenzierbar.

Notation:

$$\mathbf{f} = (f_1, f_2, \ldots, f_n)^T, \quad \mathbf{x} = (x_1, x_2, \ldots, x_n)^T$$

In Komponentenschreibweise betrachten wir also ein System der Form

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\ldots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

Bemerkung. Das Newton-Verfahren für reellwertige Funktionen einer Variablen $f : \mathbb{R} \to \mathbb{R}$ kann man auch wie folgt herleiten: Für genügend glatte Funktionen liefert Taylorentwicklung

$$0 = f(x^*) = f(x_i + (x^* - x_i)) = f(x_i) + (x^* - x_i)f'(x_i) + \frac{1}{2}(x^* - x_i)^2 f''(x_0) + \ldots.$$

Bei Vernachlässigung höherer Potenzen von $(x^* - x_i)$ betrachtet man als Approximation an x^* nur die ersten beiden Terme auf der rechten Seite, d.h.

$$0 = f(x_i) + (x_{i+1} - x_i)f'(x_i).$$

Diese so erhaltene Nährung an x^* bezeichnet man mit x_{i+1}. Umstellung obiger Gleichung liefert das bereits bekannte Newton-Verfahren. Diesen Ansatz werden wir nun zur Herleitung des Newton-Verfahrens für Systeme nutzen.

Taylorentwicklung der multivariaten Funktion $\mathbf{f}$ um den Nährungswert $\mathbf{x}_i \in \mathbb{R}^n$ liefert:

$$\mathbf{0} = \mathbf{f}(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}_i + (\mathbf{x}^* - \mathbf{x}_i)) = \underbrace{\mathbf{f}(\mathbf{x}_i) + D\mathbf{f}(\mathbf{x}_i)(\mathbf{x}^* - \mathbf{x}_i)}_{=:\bar{\mathbf{f}}(\mathbf{x})} + o(\|\mathbf{x}^* - \mathbf{x}_i\|) \quad \text{für } \mathbf{x}_i \to \mathbf{x}^*$$

Die in obiger Gleichung eingeführte Funktion $\bar{\mathbf{f}}(\mathbf{x})$ ist eine lineare Ersatzabbildung. $D\mathbf{f}(\mathbf{x}_i)$ ist die Jacobimatrix von $\mathbf{f}$ ausgewertet an der Stelle $\mathbf{x}_i$. Die Jacobimatrix hat die Form

$$D\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{pmatrix}.$$

Wir approximieren nun den Vektor $\mathbf{x}^*$ durch die Nullstelle der linearen Ersatzabbildung. Die Nullstelle der Ersatzabbildung ist die nächste Iterierte:

$$\begin{aligned} & \mathbf{0} = \bar{\mathbf{f}}(\mathbf{x}_{i+1}) \\ \Leftrightarrow \quad & \mathbf{0} = \mathbf{f}(\mathbf{x}_i) + D\mathbf{f}(\mathbf{x}_i)(\mathbf{x}_{i+1} - \mathbf{x}_i) \\ \Leftrightarrow \quad & \mathbf{x}_{i+1} = \mathbf{x}_i - (D\mathbf{f}(\mathbf{x}_i))^{-1}\mathbf{f}(\mathbf{x}_i) \end{aligned}$$

Auf die Berechnung der Inversen der Jacobimatrix (ausgewertet an der Iterierten $\mathbf{x}_i$) kann man verzichten. Stattdessen löst man in jedem Iterationsschritt ein lineares Gleichungssystem. Das Newton-Verfahren zur iterativen Lösung nichtlinearer Gleichungssysteme unter Verwendung der Anfangsnährung $\mathbf{x}_0 \in \mathbb{R}^n$ hat dann die Form:

$$\begin{aligned} D\mathbf{f}(\mathbf{x}_i)\Delta\mathbf{x}_i &= -\mathbf{f}(\mathbf{x}_i) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \Delta\mathbf{x}_i \quad i = 0, 1, \ldots . \end{aligned}$$

In jedem Iterationsschritt wird die Newton-Korrektur $\Delta\mathbf{x}_i := \mathbf{x}_{i+1} - \mathbf{x}_i$ berechnet.

Wir haben somit die Lösung eines nichtlinearen Gleichungssystems auf die Lösung einer Folge von linearen Gleichungssystemen zurückgeführt.

Beispiel 6.11. Wir werden nun das Newton Verfahren anwenden um eine Lösung des nichtlinearen Gleichungssystems

$$\begin{aligned} x^2 + y^2 - 6 &= 0 \\ x^3 - y^2 &= 0 \end{aligned} \tag{6.3}$$

zu berechnen. Wir nutzen als Startwert $\mathbf{x}_0 = (x_0, y_0)^T = (1, 1)^T$. Es gilt also

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \end{pmatrix} = \begin{pmatrix} x^2 + y^2 - 6 \\ x^3 - y^2 \end{pmatrix}.$$

Abbildung 6.4 zeigt die Funktionen $f_1(x, y)$ und $f_2(x, y)$ zusammen mit der Null-Ebene. In Abbildung 6.5 zeigen wir die Kurven $f_1(x, y) = 0$ und $f_2(x, y) = 0$. Man erkennt, dass es zwei Schnittpunkte gibt, also zwei Lösungen von (6.3). Im linken Bild ist auch der Startwert der Iteration, d.h. der Punkt $(1, 1)$ markiert.

Zur Durchführung des Newton-Verfahrens benötigen wir die Jacobimatrix

$$D\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x & 2y \\ 3x^2 & -2y \end{pmatrix}.$$

Nun betrachten wir den ersten Iterationsschritt. Dabei benötigen wir

$$D\mathbf{f}(\mathbf{x}_0) = \begin{pmatrix} 2 & 2 \\ 3 & -2 \end{pmatrix}, \quad \mathbf{f}(\mathbf{x}_0) = \begin{pmatrix} -4 \\ 0 \end{pmatrix}.$$

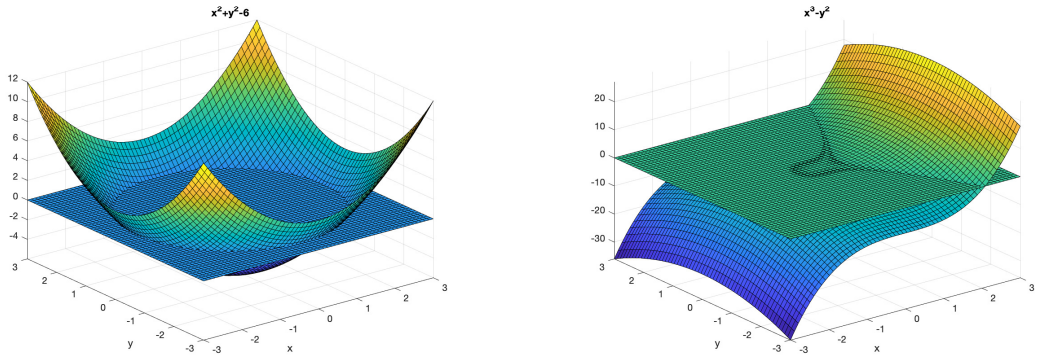


Abbildung 6.4: Die Funktionen $f_1(x, y)$ und $f_2(x, y)$ zusammen mit der Null-Ebene.

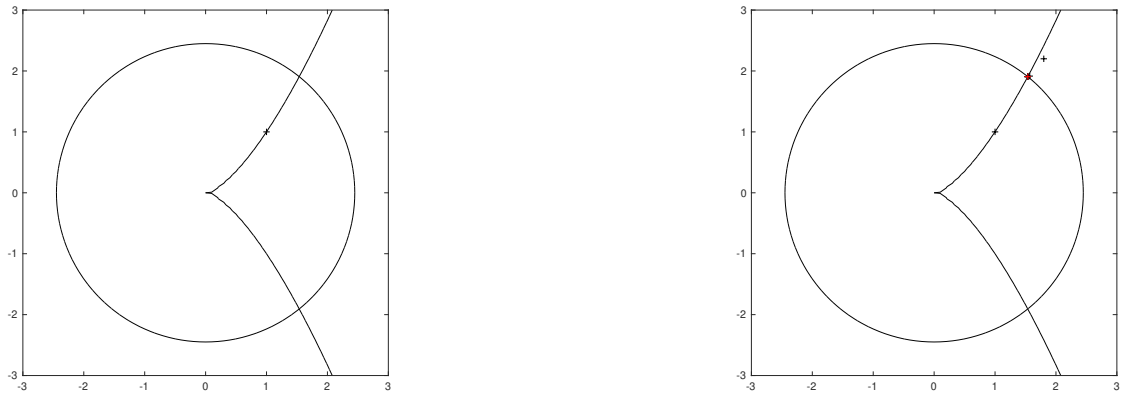


Abbildung 6.5: Die Kurven $f_1(x, y) = 0$ und $f_2(x, y) = 0$. Beachte, dass es zwei Schnittpunkte gibt. Das '+'-Symbol im linken Bild zeigt den Startwert $\mathbf{x}_0 = (1, 1)^T$, Die '+'-Symbole im rechten Bild zeigen die berechneten Nährungen an $\mathbf{x}^*$.

Nun lösen wir das Gleichungssystem

$$\begin{pmatrix} 2 & 2 \\ 3 & -2 \end{pmatrix} \begin{pmatrix} \Delta x_0 \\ \Delta y_0 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

und erhalten

$$\begin{pmatrix} \Delta x_0 \\ \Delta y_0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 1.2 \end{pmatrix}$$

sowie

$$\mathbf{x}_1 = \begin{pmatrix} x_0 + \Delta x_0 \\ y_0 + \Delta y_0 \end{pmatrix} = \begin{pmatrix} 1.8 \\ 2.2 \end{pmatrix}.$$

Die weiteren Iterierten kann man der folgenden Tabelle entnehmen:

i	x_i	y_i	$\max(\lvert f_1(x_i,y_i)\rvert, \lvert f_2(x_i,y_i)\rvert)$
1	1.8	2.2	2.08
2	1.569369369369369	1.915970515970516	$1.9428 \cdot 10^{-1}$
3	1.538198456516616	1.906569044603901	$4.4558 \cdot 10^{-3}$
4	1.537656333960022	1.906728433189744	$1.3306 \cdot 10^{-6}$
5	1.537656171698436	1.906728480313266	$1.2034 \cdot 10^{-13}$

Auch für nichtlineare Systeme ist das Newton-Verfahren lokal quadratisch konvergent.

Satz 6.12. *Sei $f \in C^2(U, \mathbb{R}^n)$ und $\boldsymbol{x}^* \in U$ eine Nullstelle von $\boldsymbol{f}$ in U, sodass $D\boldsymbol{f}(\boldsymbol{x}^*)$ regulär ist. Dann existiert ein $\varepsilon > 0$, sodass für jeden Startwert $\boldsymbol{x}_0 \in B_\varepsilon(\boldsymbol{x}^*) \cap U$ das Newton-Verfahren durchführbar und konvergent ist. Für die Iterierten $(x_k)_{k=0,1,\ldots}$ gilt*

$$\|\boldsymbol{x}^* - \boldsymbol{x}_{k+1}\| \leq c\|\boldsymbol{x}^* - \boldsymbol{x}_k\|^2$$

mit einer Konstante $c \geq 0$.

Beweis. Da nach Voraussetzung $\det D\mathbf{f}(\mathbf{x}^*) \neq 0$ gilt und die Abbildung $x \mapsto \det D\mathbf{f}(\mathbf{x})$ stetig ist, existiert ein $\tilde{\varepsilon} > 0$, sodass $\det D\mathbf{f}(\mathbf{x}) \neq 0$ und $\|D\mathbf{f}(\mathbf{x})^{-1}\| \leq c_1$ für alle $\mathbf{x} \in B_{\tilde{\varepsilon}}(\mathbf{x}^*) \subset U$ gilt.

Es sei nun $\mathbf{x}_k \in B_{\tilde{\varepsilon}}(\mathbf{x}^*)$ für ein $k \geq 0$. Taylorentwicklung liefert uns

$$\mathbf{0} = \mathbf{f}(\mathbf{x}^*) = \mathbf{f}(\mathbf{x}_k) + D\mathbf{f}(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k) + \mathcal{O}(\|\mathbf{x}^* - \mathbf{x}_k\|^2),$$

d.h., es gibt eine Konstante $c_2 \geq 0$ mit

$$\|\mathbf{f}(\mathbf{x}_k) + D\mathbf{f}(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k)\| \leq c_2\|\mathbf{x}^* - \mathbf{x}_k\|^2.$$

Mit der Iterationsvorschrift des Newton-Verfahrens erhalten wir

$$\mathbf{x}^* - \mathbf{x}_{k+1} = D\mathbf{f}(\mathbf{x}_k)^{-1}\left(\mathbf{f}(\mathbf{x}_k) + D\mathbf{f}(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k)\right).$$

Damit folgt

$$\begin{aligned}\|\mathbf{x}^* - \mathbf{x}_{k+1}\| &= \|D\mathbf{f}(\mathbf{x}_k)^{-1}\left(\mathbf{f}(\mathbf{x}_k) + D\mathbf{f}(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k)\right)\| \\ &\leq \|D\mathbf{f}(\mathbf{x}_k)^{-1}\| \cdot \|\mathbf{f}(\mathbf{x}_k) + D\mathbf{f}(\mathbf{x}_k)(\mathbf{x}^* - \mathbf{x}_k)\| \\ &\leq c_1c_2\|\mathbf{x}^* - \mathbf{x}_k\|^2.\end{aligned}$$

Mit $\varepsilon \leq \min\{\frac{1}{c_1c_2}, \tilde{\varepsilon}\}$ folgt für $\mathbf{x}_k \in B_\varepsilon(\mathbf{x}^*)$

$$\|\mathbf{x}^* - \mathbf{x}_{k+1}\| \leq c_1c_2\varepsilon\|\mathbf{x}^* - \mathbf{x}_k\| \leq \|\mathbf{x}^* - \mathbf{x}_k\| < \varepsilon \leq \tilde{\varepsilon}$$

und somit $\mathbf{x}_{k+1} \in B_{\tilde{\varepsilon}}(\mathbf{x}^*)$.

Die Iteration ist somit wohldefiniert und konvergiert, sofern $\mathbf{x}_0 \in B_\varepsilon(\mathbf{x}^*)$ gilt. □