NUMERISCHE MATHEMATIK 1

Vorlesungsskript, Wintersemester 2022/23

Christian Clason

Stand vom 12. Januar 2024

Institut für Mathematik und wissenschaftliches Rechnen
Universität Graz

Inhaltsverzeichnis

III NUMERISCHE ANALYSIS

Teil I

ALGORITHMEN UND IHRE GRENZEN

1 GRUNDLAGEN

In diesem Kapitel betrachten wir die grundlegenden Fragestellungen der numerischen Mathematik und sammeln die wichtigsten allgemeinen Werkzeuge zu ihrer Untersuchung. Begriffe, die nur für spezifische Probleme relevant sind, sollen grundsätzlich erst dort erläutert werden, wo sie auftreten.

1.1 GRUNDBEGRIFFE: PROBLEM UND ALGORITHMUS

Die numerische Mathematik beschäftigt sich mit der Entwicklung und Analyse konstruktiver Verfahren zur Lösung von Problemen, die mathematisch formulierbar sind. Üblicherweise sind dabei ein *Modell* und *Eingangsdaten* gegeben, aus denen eine (numerische) Antwort (die *Ausgabedaten*) zu ermitteln ist. Mathematisch wird das Modell als eine Funktion $f : X \to Y$ beschrieben, wobei X die Menge der möglichen Eingabe-, Y die Menge der möglichen Ausgabedaten ist. Ein konkretes *Problem* ist dann, für gegebenes $x \in X$ die Ausgabe $f(x) \in Y$ zu berechnen. Das Problem, zwei Zahlen zu addieren, wäre dann beschrieben durch

$$X = \mathbb{R}^2,\ Y = \mathbb{R}, \quad f : (x_1, x_2) \mapsto x_1 + x_2.$$

Ein komplizierteres Problem wäre gegeben durch die genaue Beschreibung eines Motorblocks x, das System partieller Differentialgleichungen zur Beschreibung der Wärmeentwicklung im laufenden Betrieb f und die Frage nach der Temperaturverteilung nach einer gewissen Laufzeit $f(x)$.

Unter einem *Algorithmus* verstehen wir eine eindeutig formulierte Vorschrift zur Lösung eines Problems.[1] Ein Algorithmus wird spezifiziert durch

- die *Eingabedaten*, die zur Durchführung benötigt werden,
- die *Ausgabedaten*, die das Ergebnis des Algorithmus darstellen,
- sowie die *Beschreibung der durchzuführenden Schritte* inklusive der benötigten *Hilfsgrößen*.

Von einem Algorithmus erwartet man in der Regel neben der Korrektheit (d. h. er liefert tatsächlich eine Lösung der Aufgabe), dass er

[1] Das bekannteste nicht-mathematische Beispiel ist sicher das Kochrezept.

- *durchführbar* ist (zum Beispiel, indem nur elementare Schritte[2] durchgeführt werden),
- *terminiert* (d. h. garantiert nach einer endlichen Anzahl von Schritten abgeschlossen ist) und
- *deterministisch* ist (d. h. für gleiche Eingabedaten stets das gleiche Resultat liefert).[3]

Ein guter Algorithmus sollte darüber hinaus die folgenden Eigenschaften erfüllen:

1. *Stabilität*: Kleine Fehler in den Eingabedaten sollen nur kleine Fehler in den Ausgabedaten verursachen.
2. *Genauigkeit*: Die Ausgabedaten sollen eine möglichst gute Lösung der gestellten Aufgabe sein.
3. *Flexibilität*: Der Algorithmus soll für eine große Klasse von Eingabedaten anwendbar sein.
4. *Effizienz*: Der Algorithmus soll auch für umfangreiche Probleme praktisch durchführbar sein.

Die Stabilität ist ein so fundamentales Problem, dass ihr ein eigenes Kapitel 3 gewidmet ist. Bei der Effizienz betrachtet man die Komplexität des Algorithmus, auf die wir weiter unten eingehen werden. Bei der Beurteilung der Genauigkeit wird man in der Regel eine Abschätzung des zu erwartenden Fehlers in den Ausgabedaten erwarten. Dafür benötigen wir einige wichtige Begriffe aus der Analysis.

1.2 MESSEN UND SCHÄTZEN: NORMEN

Der zentrale Begriff für jede Art von Messung ist die *Norm*. Diese ist die fundamentale Abbildung, mit deren Hilfe Fragestellungen in abstrakten Räumen (Vektorräume, Funktionenräume) in die reellen Zahlen übertragen werden können, wo man sich der bekannten Hilfsmittel bedienen kann (etwa der Ordnung).

Definition 1.1. Sei V ein Vektorraum über $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. Die Abbildung $\|\cdot\| : V \to \mathbb{R}$ heißt *Norm auf* V, falls gilt

(N1) $\|v\| \geq 0$ für alle $v \in V$, und $\|v\| = 0$ dann und nur dann, wenn $v = 0$;

(N2) Für alle $a \in \mathbb{K}$ und $v \in V$ gilt $\|av\| = |a|\|v\|$;

[2] Was "elementar" bedeutet, hängt natürlich von dem jeweiligen Kontext ab. Eine Beispiel ist die Anwendung eines existierenden Algorithmus, dessen Korrektheit bekannt ist.

[3] In letzter Zeit sind allerdings *randomisierte* Algorithmen zunehmend populär geworden, die zwar nicht in jedem Fall, aber doch mit hoher Wahrscheinlichkeit, erheblich schneller als deterministische sein können.

(N3) Für alle $v, w \in V$ gilt die *Dreiecksungleichung*

$$\|v + w\| \leq \|v\| + \|w\|.$$

Das Paar $(V, \|\cdot\|)$ heißt dann *normierter Raum.*

Beispiel 1.2 (p-Normen auf $\mathbb{R}^n$). Wir betrachten $V = \mathbb{R}^n$ und $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ und definieren für $1 \leq p < \infty$

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

Häufige Spezialfälle sind:

(i) $\|x\|_1 = \sum_{i=1}^{n} |x_i|$,

(ii) $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$ (*Euklidische Norm*),

(iii) $\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|$ (*Maximumsnorm*).

Spezielle Normen für Matrizen werden wir in Kapitel 4 kennen lernen. Wichtige Beispiele für Normen auf unendlichdimensionale Vektorräumen sind die folgenden.

Beispiel 1.3 (Normen auf Funktionenräumen). Auf dem Vektorraum $C([a, b])$ aller stetigen Funktionen vom Intervall $[a, b] \subset \mathbb{R}$ nach $\mathbb{R}$ stellen die folgenden Abbildungen eine Norm dar:

(i) $\|f\|_2 = \left(\int_a^b f(x)^2 \, dx \right)^{\frac{1}{2}}$,

(ii) $\|f\|_\infty = \max_{x \in [a,b]} |f(x)|$.

In einem normierten Raum definiert $\|v - w\|$ auf kanonische Weise einen Abstand zwischen zwei Vektoren v und w.[4] Damit haben wir schon ein Mittel zur Hand, um Fehler (den Abstand zwischen berechneter und gesuchter Größe) zu berechnen. Allerdings wird üblicherweise die gesuchte Größe nicht bekannt sein; eine wichtige Aufgabe wird also sein, obere Schranken für diesen Fehler anzugeben. Neben der Dreiecksungleichung ist dafür die folgende Ungleichung unerlässlich:

Satz 1.4 (Höldersche Ungleichung). *Für $x, y \in \mathbb{R}^n$ und $1 \leq p, q \leq \infty$, $1/p + 1/q = 1$ (bzw. $p = 1, q = \infty$) gilt*

$$|x^T y| \leq \|x\|_p \|y\|_q.$$

Ein wichtiger Spezialfall ist die Cauchy-Schwarzsche Ungleichung *für $p = q = 2$,*

$$|x^T y| \leq \|x\|_2 \|y\|_2.$$

[4] Insbesondere haben wir dadurch einen Konvergenzbegriff: Wir sagen, $x_n \to x^*$, falls $\|x_n - x^*\| \to 0$.

Dabei ist $\langle x, y\rangle := x^T y = \sum_{i=1}^{n} x_i y_i$ das *Euklidische Skalarprodukt*[5] im $\mathbb{R}^n$.

Mit Hilfe der Hölderschen Ungleichung sieht man zum Beispiel, dass man eine p-Norm durch eine andere abschätzen kann. Dies lässt sich präzisieren.

Definition 1.5. Zwei Normen $\|\cdot\|$, $\|\cdot\|'$ auf einem Vektorraum V heißen *äquivalent*, falls zwei Konstanten $c_1, c_2 > 0$ existieren, so dass für alle $x \in V$ gilt

$$c_1\|x\| \leq \|x\|' \leq c_2\|x\|.$$

Satz 1.6 (Äquivalenz der Normen). *Auf einem endlichdimensionalen Vektorraum sind alle Normen äquivalent.*

Beispiel 1.7. Für $x \in \mathbb{R}^n$ gilt

(i) $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$,

(ii) $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$,

(iii) $\|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty$.

Wozu benötigt man dann noch andere Normen, wenn man immer die Euklidische Norm für Abschätzungen nehmen könnte? Zum einen ist es eine der wesentlichen Aufgaben in der numerischen Mathematik, kontinuierliche (und damit unendlichdimensionale) Probleme durch endlichdimensionale Näherungen zu lösen. Im Unendlichdimensionalen gilt die Äquivalenz der Normen aber in der Regel nicht, und dieser Unterschied sollte berücksichtigt werden. Dies zeigt sich dadurch, dass die Normen mit wachsender Dimension "auseinander laufen", d. h. das Verhältnis c_2/c_1 immer größer wird, wie an den Beispielen ersichtlich wird. Für große Probleme (d. h. großes n) kann dies durchaus bemerkbare Auswirkungen haben.

Zum anderen haben die unterschiedlichen Normen verschiedene "Sichtweisen". So stellt die Maximumsnorm eine harte Schranke dar; alle Komponenten eines Vektors x müssen vom Betrag kleiner als $\|x\|_\infty$ sein. Die Euklidische Norm ist dagegen eine Schranke im Mittel (was man leicht sieht, indem man durch die Anzahl der Komponenten dividiert); einzelne Komponenten dürfen durchaus größer sein, solange sie nicht zu groß oder es zu viele sind. Dies gilt auch für die anderen p-Normen, wobei $p = 1$ eine Sonderstellung einnimmt: Die zugehörige Norm bestraft auch bei sehr kleinen Schranken Ausreißer stark. Dies wird in dem aktuellen Forschungsgebiet des *compressed sensing*[6] ausgenutzt.

[5] Die Tatsache, dass $\langle x, x\rangle = \|x\|^2$ gilt, zeichnet die Euklidische Norm unter allen p-Normen aus. Dies gilt auch für die $p = 2$-Norm in Beispiel 1.3, die durch das Skalarprodukt $\langle f, g\rangle = \int_a^b f(x)g(x)dx$ induziert wird. Physikalisch kann diese Norm als eine Gesamtenergie interpretiert werden; durch die Energieerhaltungssätze der Physik bekommt sie deshalb zusätzliche Bedeutung. Dies überträgt sich auch auf die Euklidische Vektornorm, da numerische Verfahren häufig auf physikalische Probleme angewendet werden.

[6] Siehe http://statweb.stanford.edu/~candes/l1magic/examples.html. Dieser Ansatz erlaubt es theoretisch

1.3 DIE TAYLORENTWICKLUNG

Ein wesentlicher Schritt in der Konstruktion und Untersuchung von Algorithmen zur Lösung kontinuierlicher Probleme ist deren Approximation durch endliche Darstellungen.

Ein einfaches Beispiel liefert die Berechnung der Exponentialfunktion, definiert durch die Reihe

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots .$$

Bricht man die Summation nach einem festen n ab, so erhält man einen endlichen Algorithmus, der eine Näherung der Exponentialfunktion liefert. Der Fehler, der dabei entsteht, ist ein *Approximationsfehler*. Diesen abzuschätzen ist eine wesentliche Aufgabe der numerischen Mathematik.

Ein Werkzeug dafür stellt die Taylorentwicklung dar, die auch die Approximation anderer Funktionen durch Reihen ermöglicht.

Satz 1.8 (Taylor). *Sei $I \subset \mathbb{R}$ ein offenes Intervall, f eine $n+1$ mal stetig differenzierbare Funktion (d. h. die $(n+1)$-te Ableitung $f^{(n+1)}$ ist stetig), und $x_0 \in I$. Dann gilt für alle $x \in I$*

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + R_{n+1}(x)$$

mit Restglied

$$R_{n+1}(x) = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\xi)$$

für ein $\xi \in (x_0, x)$ (bzw. $\xi \in (x, x_0)$).

Mit Hilfe des Restglieds haben wir auch sofort eine Abschätzung des Approximationsfehlers zur Hand.

Bemerkung (Taylorentwicklung im $\mathbb{R}^n$). Analog lässt sich auch für eine Funktion $f : \mathbb{R}^n \to \mathbb{R}$ die Taylorentwicklung definieren, wenn man Ableitung durch den Vektor (die Matrix etc.) der partiellen Ableitungen ersetzt. Für den Fall der Taylorentwicklung zweiter Ordnung sei dies exemplarisch dargestellt. Sei $\Omega \subset \mathbb{R}^n$ offen, $f \in C^3(\Omega)$, $x_0 \in \Omega$. Dann existiert ein $\xi \in \Omega$, so dass gilt

$$f(x) = f(x_0) + \sum_{i=1}^{n} \frac{\partial f(x_0)}{\partial x_i}(x_i - (x_0)_i) + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2 f(x_0)}{\partial x_j \partial x_i}(x_i - (x_0)_i)(x_i - (x_0)_j)$$
$$+ \frac{1}{6}\sum_{i,j,k=1}^{n} \frac{\partial^3 f(\xi)}{\partial x_k \partial x_j \partial x_i}(x_i - (x_0)_i)(x_j - (x_0)_j)(x_k - (x_0)_k).$$

sogar, eine Digitalkamera zu bauen, die mit einem Bildsensor, der nur aus einem einzigen Pixel besteht, Bilder aufnehmen kann: http://terrytao.wordpress.com/2007/04/13/compressed-sensing-and-single-pixel-cameras/

Indem man den *Gradienten*

$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\right)^T$$

und die *Hessematrix*

$$\nabla^2 f(x) := \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \ldots & \frac{\partial^2 f(x)}{\partial x_n^2}(x) \end{pmatrix}$$

einführt, lässt sich die Taylorentwicklung einfacher schreiben als

$$f(x) = f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \frac{1}{2}\langle x - x_0, \nabla^2 f(x_0)(x - x_0) \rangle + R_3(x). \tag{1.1}$$

Dabei sind $(x - x_0)$, $\nabla f(x_0)$ und $\nabla^2 f(x_0)(x - x_0)$ Vektoren im $\mathbb{R}^n$.

Bemerkung. Die Taylorentwicklung ist natürlich kein Zaubermittel. Zwei wichtige Fallstricke:

(i) Die Differenzierbarkeit von f ist eine zwingende Voraussetzung; insbesondere muss die $n+1$-te Ableitung stetig sein.

(ii) Die Entwicklung wird in der Regel nur für x sehr nahe bei x_0 vernünftige Ergebnisse liefern. (Dies kann man sich anhand der Restglieddarstellung leicht verdeutlichen.)

1.4 LANDAU-SYMBOLE

Oft ist man nicht an dem Fehler für eine feste Größe n interessiert, sondern an dem asymptotischen Verhalten für große n. Eine kompakte Notation erlauben die Landau-Symbole,[7] die den Begriff der Größenordnung präzisieren:

Definition 1.9 (Landau-Symbole für Folgen). Seien $\{f_n\}_{n\in\mathbb{N}}, \{g_n\}_{n\in\mathbb{N}} \subset \mathbb{R}$ Folgen. Dann schreiben wir

1. $f_n = O(g_n)$ für $n \to \infty$, falls ein $C > 0$ und ein n_0 existieren, so dass gilt

$$|f_n| \leq C|g_n| \qquad \text{für alle } n > n_0,$$

2. $f_n = o(g_n)$ für $n \to \infty$, falls für alle $\varepsilon > 0$ ein n_0 existiert, so dass gilt

$$|f_n| \leq \varepsilon|g_n| \qquad \text{für alle } n > n_0.$$

[7] Die "Groß-O"-Notation geht im Wesentlichen auf Paul Bachmann im Jahre 1894 zurück, Lev Landau präzisierte den Begriff und fügte 1909 "klein-O" hinzu.

Die zweite Definition ist äquivalent dazu, dass $|f_n| \leq \varepsilon_n |g_n|$ für eine (positive) Nullfolge $\{\varepsilon_n\}_{n\in\mathbb{N}}$ gilt.

Definition 1.10 (Landau-Symbole für Funktionen). Sei $I \subset \mathbb{R}$ ein offenes Intervall, $f, g : I \to \mathbb{R}$ und $x_0 \in \mathbb{R}$. Dann schreiben wir

1. $f(x) = O(g(x))$ für $x \to x_0$, falls ein $C > 0$ und ein $\delta > 0$ existieren, so dass gilt
$$|f(x)| \leq C|g(x)| \qquad \text{für alle } |x - x_0| < \delta,$$
2. $f(x) = o(g(x))$ für $x \to x_0$, falls für alle $\varepsilon > 0$ ein $\delta > 0$ existieren, so dass gilt
$$|f(x)| \leq \varepsilon|g(x)| \qquad \text{für alle } |x - x_0| < \delta.$$

Eine Funktion $O(f(x))$ verhält sich also für $x \to x_0$ im Wesentlichen wie $f(x)$, und eine Funktion $o(f(x))$ ist für x nahe bei x_0 gegenüber $f(x)$ vernachlässigbar. Analog definiert man die Landau-Symbole für $x \to \pm\infty$.

Bemerkung. Auch hier gibt es zwei wichtige Dinge zu beachten:

(i) Die Landau-Symbole sind nur sinnvoll in Verbindung mit der Angabe, für welche Umgebung sie gelten. Für eine andere Richtung als die angegebenen kann die Aussage falsch sein, wie das folgende Beispiel 1.12 (iii) zeigt. Wenn dies aus dem Kontext ersichtlich ist, wird diese Angabe jedoch häufig weggelassen. So ist es Konvention, dass $n \to \infty$ und $\varepsilon, h \to 0$ verstanden wird.

(ii) Die Landau-Symbole stehen immer auf der rechten Seite einer Gleichung; sobald Terme "unter den Teppich gekehrt" wurden, wird das Gleichheitszeichen zur Einbahnstraße.[8]

Für das Rechnen mit den Landau-Symbolen gibt es einige nützliche Regeln:

Lemma 1.11. *Es gilt jeweils für $x \to x_0$:*

(i) $f(x) = O(f(x))$,

(ii) $f(x) = o(g(x)) \Rightarrow f(x) = O(g(x))$,

(iii) $f(x) = O(g(x)) \Rightarrow Kf(x) = O(g(x))$ *für alle* $K \in \mathbb{R}$,

(iv) $f(x) = O(g_1(x))$ *und* $g_1(x) = O(g_2(x)) \Rightarrow f(x) = O(g_2(x))$,

(v) $f_1(x) = O(g_1(x))$ *und* $f_2(x) = O(g_2(x)) \Rightarrow f_1(x)f_2(x) = O(g_1(x)g_2(x))$,

[8] Formal steht $O(f(x))$ für die Menge aller Funktionen, für die $C|f(x)|$ eine obere Schranke ist. Das Gleichheitszeichen steht also in Wirklichkeit – aus rein historischen Gründen – für eine Mengeninklusion, die bekanntlich nicht symmetrisch ist.

(vi) $f(x) = O(g(x) + h(x))$ *und* $h(x) = O(g(x)) \Rightarrow f(x) = O(g(x))$.

Beispiel 1.12. Es sind

(i) $3n + 5n^2 = 5n^2 + O(n) = O(n^2)$ für $n \to \infty$,

(ii) $\sin(x) = x - \frac{1}{6}x^3 + O(x^5)$ für $x \to 0$,

(iii) $x^n = o(e^x)$ für $x \to \infty$.

Für das Restglied der Taylorentwicklung (und damit den Approximationsfehler) haben wir daher sofort, dass gilt

$$\|R_{n+1}(x)\| = O(\|x - x_0\|^{n+1}) \qquad \text{für } x \to x_0.$$

Allgemein können wir mit Hilfe der Landau-Symbole die Geschwindigkeit, mit der eine Folge konvergiert, mathematisch sinnvoll definieren.

Definition 1.13. Sei $(V, \|\cdot\|)$ ein normierter Raum, $x^* \in V$, und $\{x_n\}_{n\in\mathbb{N}} \subset V$ eine Folge mit $x_n \to x^*$. Wir sagen,

(i) x_n konvergiert (mindestens) *linear*, wenn es ein n_0 und ein $C < 1$ gibt, so dass gilt

$$\|x_{n+1} - x^*\| \leq C\|x_n - x^*\| \qquad \text{für alle } n > n_0.$$

(Beachte, dass dies eine stärkere Forderung als $\|x_{n+1} - x^*\| = O(\|x_n - x^*\|)$ ist.)

(ii) x_n konvergiert (mindestens) *superlinear*, wenn es ein n_0 gibt, so dass gilt

$$\|x_{n+1} - x^*\| = o(\|x_n - x^*\|) \qquad \text{für alle } n > n_0,$$

(iii) x_n besitzt (mindestens) *Konvergenzordnung* k, wenn es ein n_0 gibt, so dass gilt

$$\|x_{n+1} - x^*\| = O(\|x_n - x^*\|^k) \qquad \text{für alle } n > n_0.$$

Für den Fall $k = 2$ spricht man von *quadratischer Konvergenz.*

Bemerkung. (i) Bei linearer Konvergenz hängt die tatsächliche Konvergenzgeschwindigkeit stark von der Konstanten C ab: Für $C = 0.1$ gewinnt man pro Schritt eine Stelle Genauigkeit, für $C = 0.9$ sind dafür schon 10 bis 11 Schritte nötig. Ist C sehr nahe bei 1, kann die lineare Konvergenz sehr langsam sein.

(ii) Quadratische Konvergenz ist sehr schnell, wenn man schon hinreichend nahe der Lösung ist. Ist man jedoch weit von der Lösung entfernt, sagt die Abschätzung sehr wenig aus; tatsächlich ist sehr langsame Konvergenz möglich.

(iii) Man muss auch den Aufwand für einen einzelnen Iterationsschritt berücksichtigen. Dieser ist für ein quadratisch konvergierendes Verfahren in der Regel höher als für ein linear konvergierendes Verfahren, so dass, am Gesamtaufwand gemessen, ein linear konvergierendes Verfahren günstiger sein kann.

1.5 KOMPLEXITÄT VON ALGORITHMEN

Die Landau-Symbole erlauben es auch, bequem den Aufwand zu beschreiben, den man für die Ausführung eines Algorithmus in Abhängigkeit der Problemgröße n aufwenden muss. Dabei unterscheidet man üblicherweise

1. *Laufzeitkomplexität* (gemessen z. B. durch die Anzahl der elementaren Schritte) und
2. *Speicherkomplexität* (gemessen z. B. durch die Anzahl und Umfang der benötigten Hilfsgrößen).

Oft wird man eine niedrigere Laufzeitkomplexität gegen eine höhere Speicherkomplexität oder umgekehrt abwägen müssen.

Beispiel 1.14 (Komplexität der Polynomauswertung). Wir betrachten ein Polynom $P_n(x) = \sum_{i=0}^{n} a_i x^i$ mit $a_i \in \mathbb{R}$. Für gegebenes $x_0 \in \mathbb{R}$ bestimmen wir jetzt den Wert $P_n(x_0)$ des Polynoms an der Stelle x_0 auf zwei verschiedene Arten, wobei wir als elementare Schritte nur Addition und Multiplikation zulassen. Die naive Variante rechnet die das Polynom Summand für Summand aus, wobei die Potenzen als entsprechende Anzahl von Multiplikationen berechnet werden.

Algorithmus 1.1 : Polynomauswertung (naiv)

Input : $x_0, a_0, \ldots, a_n$
Output : p_0

```
1 p_0 ← a_0
2 for k = 1, ..., n do                          // Berechne a_k x_0^k
3     M ← a_k
4     for m = 1, ..., k do                      // Berechne x_0^k = x_0 ··· x_0
5         M ← M · x_0
6     p_0 ← p_0 + M
```

Zählen wir die Operationen pro Schritt, werden n mal je eine Addition und k Multiplikationen ausgeführt. Setzen wir für jede Addition und Multiplikation eine konstante Zeit C_1 bzw. C_2 an (und vernachlässigen die Zeit für die Zuweisungen), so erhalten wir als Laufzeitkomplexität

$$\sum_{k=1}^{n} (C_1 + C_2 k) = C_1 n + C_2 \frac{1}{2} n(n+1) = O(n^2).$$

Die Speicherkomplexität ist $2 = O(1)$, da – unabhängig von n – nur zwei Speicherplätze, p_0 und M, benötigt werden.

Dagegen basiert das *Horner-Schema* auf der Umformung durch Ausklammern

$$P_n(x) = a_0 + x\,(a_1 + x\,(\cdots + x(a_{n-1} + x a_n))) .$$

Rechnen wir die Klammern von innen nach außen durch, erhalten wir den folgenden Algorithmus.

Algorithmus 1.2 : Polynomauswertung (Horner-Schema)

Input : $x_0, a_0, \ldots, a_n$
Output : p_0

```
1 p_0 ← a_n
2 for k = n − 1, ..., 0 do
3     p_0 ← p_0 · x_0 + a_k
```

Die Speicherkomplexität ist zwar immer noch $1 = O(1)$, aber die Berechnung erfordert nur noch pro Schritt je eine Addition und Multiplikation: Die Laufzeitkomplexität ist hier also $2n = O(n)$. Der Unterschied zwischen $O(n)$ und $O(n^2)$ kann für große n beträchtlich sein. Dies wird uns im Kapitel über lineare Gleichungssysteme wieder begegnen.

WEITERFÜHRENDE LITERATUR

R. L. Graham, D. E. Knuth & O. Patashnik (1994), *Concrete mathematics*, 2. Aufl., A foundation for computer science, Addison-Wesley Publishing Company, Reading, MA, URL: http://www-cs-faculty.stanford.edu/~uno/gkp.html.

2 NUMERISCHE ARITHMETIK

Aus der Algebra wissen wir, dass wir für die Lösung bestimmter Aufgaben (zum Beispiel die Nullstellensuche von Polynomen) nicht mit einem diskreten Zahlbegriff auskommen. Für eine numerische Behandlung müssen wir jedoch eine endliche Näherung verwenden. Es ist klar, dass dies nicht ohne Fehler geschehen kann, die aber bei sorgfältiger Behandlung beherrschbar sind. Diese können aber nie ganz vermieden werden: Bereits bei der Eingabe entsteht ein Fehler.

Da sich komplexe Zahlen als Paar reeller Zahlen auffassen lassen, betrachten wir nur das Problem, reelle Zahlen auf endlichen Maschinen, insbesondere dem Computer, darzustellen.

2.1 GLEITKOMMADARSTELLUNG REELLER ZAHLEN

Grundlage der Maschinendarstellung von Zahlen ist die Dezimalbruchentwicklung, allgemeiner die Basisdarstellung $x = \pm \sum_{i=-\infty}^{\infty} a_i b^i$ mit $b \in \mathbb{N} \setminus \{1\}$ und $a_i \in \{0, \ldots, b-1\}$. Die Basisdarstellung einfach nach einer festen Anzahl von Stellen abzuschneiden (*Festkommadarstellung*) stellt keine befriedigende Lösung dar, da in einer Rechnung oft Zahlen sehr verschiedener Größenordnungen auftauchen.[1] Deshalb geht man aus von der *Gleitkommadarstellung*[2]

$$x = \sigma b^e \sum_{i=1}^{m} a_i b^{-i}.$$

Diese Darstellung enthält

- das *Vorzeichen* $\sigma \in \{-1, +1\}$,
- die *Basis* $b \in \mathbb{N} \setminus \{1\}$,

[1] Dies spricht auch gegen eine Approximation durch rationale Zahlen, da diese für sehr kleine oder sehr große Zahlen unhandlich werden. Auch verhindert die für eine eindeutige Darstellung notwendige Kürzung eine effiziente Nutzung.

[2] Die Gleitkommadarstellung wurde bekanntlich schon im Babylon des 19. Jahrhunderts v. Chr. mit der Basis 60 verwendet. Leider kannte man keine Notation für den Exponenten, so dass man sich für korrekte Rechnung auf das Verständnis des Rechners verlassen musste. Es sind auch (allerdings sehr wenige) Fälle bekannt, in denen falscher Stellenabgleich zu fehlerhafter Addition führte. Die moderne Nutzung des Stellensystems setzt mit der Erfindung des Logarithmus um 1600 ein.

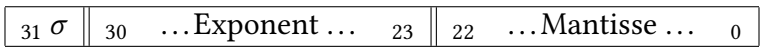
- die *Mantisse* $a_1 \dots a_m$ mit $a_i \in \{0, 1, \dots, b-1\}$ und der *Mantissenlänge* m,
- den *Exponenten* $e \in \mathbb{Z}$ mit $e_{\min} \leq e \leq e_{\max}$.

Eindeutigkeit der Darstellung lässt sich über zusätzliche Forderungen an die a_i erreichen. Die üblichste, sogenannte *normalisierte* Darstellung erhält man, indem für $x \neq 0$ gefordert wird, dass $a_1 \neq 0$ gilt. Ein bekanntes Beispiel ist die wissenschaftliche Notation, etwa $0.14 \cdot 10^{-3}$ mit $b = 10$ und $e = -3$. Für die Basis $b = 2$ führt das zu der gebräuchlichsten Gleitkommadarstellung nach dem Standard IEEE 754, die (seit 1985) Grundlage aller Rechnerarchitekturen ist.

Der Standard[3] definiert drei Zahltypen: *single*, *double*, und *extended precision*, die sich in der Bitlänge (32, 64, bzw. 80 Bit; ein Bit entspricht dem Speicherplatz für eine Binärziffer, 0 oder 1) unterscheiden. Ein Bit ist jeweils für das Vorzeichen reserviert, die restlichen Bits werden wie folgt auf Exponent und Mantisse verteilt:

	single	double	extended
Exponent	8 Bit	11 Bit	15 Bit
Mantisse	23 Bit	52 Bit	64 Bit

Die Kodierung einer Zahl in IEEE 754 ist wie folgt definiert (hier am Beispiel single precision):



- Das Vorzeichenbit $\sigma = 0$ bezeichnet eine positive Zahl, $\sigma = 1$ eine negative Zahl.
- Der Exponent ist als verschobene Binärzahl gespeichert; die Verschiebung ist so gewählt, dass nur positive Exponenten gespeichert werden müssen. Das Bitmuster $01 \dots 1$ steht für den Exponenten 0, größere Binärzahlen für positive und kleinere für negative Exponenten. Bei single precision etwa muss also von der gespeicherten Zahl 127 abgezogen werden, so dass der Wertebereich des Exponenten $[-126, 127]$ ist.
- Die Mantisse speichert die Nachkommastellen der Binärzahl, wobei die Normalisierung so erfolgt, dass die Vorkommastelle immer 1 ist (und deshalb nicht gespeichert werden muss).
- Zusätzlich existieren spezielle Bitmuster, um Sonderfälle anzuzeigen. Sind im Exponenten nur Einsen gesetzt, und ist die Mantisse gleich Null, so wird dies als $\pm\infty$ interpretiert. Ist mindestens ein Mantissenbit gesetzt, so steht dies für `NaN` – "Not a Number". Dies kann etwa durch eine nicht definierte arithmetische Operation wie

[3] Der im wesentlichen Konrad Zuses Patent von 1936 für die "selbständige Durchführung von Rechnungen mit Hilfe von Rechenmaschinen" folgt; allerdings ohne es zu zitieren.

$0 \cdot \infty$ ausgelöst werden. Die (positive oder negative) 0 wird durch Nullen in Exponent und Mantisse ausgedrückt.

Beispiel 2.1. (i) Wir betrachten die Darstellung nach IEEE 754 der Dezimalzahl 12.375. Zunächst müssen wir sie in Binärdarstellung umrechnen:

$$\begin{aligned}(12.375)_{10} &= (12)_{10} + (0.375)_{10} = (8+4)_{10} + (\tfrac{1}{4} + \tfrac{1}{8})_{10} \\ &= (1100)_2 + (0.011)_2 = (1100.011)_2.\end{aligned}$$

Die normalisierte Darstellung dieser Binärzahl ist (Kommaverschiebung entspricht Multiplikation mit Zweierpotenz)

$$(1100.011)_2 = (1.100011)_2 \cdot 2^3.$$

Die Nachkommastellen (von rechts mit Null aufgefüllt) ergeben genau die Mantisse. Durch die Verschiebung muss der Exponent nun wegen $3 = 130 - 127$ als 130 ebenfalls in Binärdarstellung gespeichert werden, d. h.

$$(130)_{10} = (128 + 2)_{10} = 10000010_2.$$

Schließlich ist die Zahl positiv ($\sigma = 0$); das gesamte zu speichernde Bitmuster ist also

0	10000010	1000110 . . . 0

(ii) Sei umgekehrt das Bitmuster

1	01111100	010 . . . 0

gegeben. Wegen $\sigma = 1$ ist die entsprechende Zahl negativ. Der Exponent ist gespeichert als

$$(01111100)_2 = \left(\sum_{k=2}^{6} 2^k\right)_{10} = (124)_{10} = (-3 + 127)_{10},$$

d. h. $e = -3$. Da in der Mantisse nur ein Bit, nämlich a_{21}, gesetzt ist, ist die Nachkommastelle

$$(0.01)_2 = (2^{-2})_{10} = (\tfrac{1}{4})_{10}.$$

Nach IEEE 754 entspricht dies also der Zahl

$$x = (-1)^\sigma \left(1 + \sum_{i=1}^{23} a_{23-i} 2^{-i}\right) \cdot 2^e = (-1)\left(1 + \frac{1}{4}\right) \cdot 2^{-3} = -0.15625.$$

Die größte und kleinste darstellbare Zahl wird im Wesentlichen durch $e_{\min}$ und $e_{\max}$ bestimmt:

	kleinste positive Zahl	größte positive Zahl
single prec.	$2^{-126} \approx 1.4 \cdot 10^{-38}$	$2^{128} - 1 \approx 3.4 \cdot 10^{38}$
double prec.	$2^{-1022} \approx 2.2 \cdot 10^{-308}$	$2^{1024} - 1 \approx 1.8 \cdot 10^{308}$
extended prec.	$2^{-16382} \approx 3.3 \cdot 10^{-4962}$	$2^{16384} - 1 \approx 1.2 \cdot 10^{4962}$

Während die Exponentenlänge den maximalen Wertebereich angibt, bestimmt die Mantissenlänge die *relative* Genauigkeit. Ein Nachteil der halb-logarithmischen Darstellung ist nämlich, dass die so darstellbaren Zahlen nicht gleichmäßig im überdeckten Zahlenbereich verteilt sind: So haben die für $b = 10$, $m = 1$ verfügbaren zehn Zahlen $0.1, \ldots, 0.9$ bei $e = -1$ den Abstand 0.01, für $e = 2$ jedoch bereits den Abstand 10.

Dem extended precision Format kommt eine besondere Bedeutung zu, da es Grundlage für die interne Berechnung (in den sogenannten *Registern*) von Computern ist. Auch dies geht auf den IEEE Standard zurück, der fordert, dass genauer gerechnet wird als dargestellt. Deshalb muss nach jeder Elementaroperation das Ergebnis mit geringerer Genauigkeit abgespeichert werden: man spricht von *Rundung*.

2.2 GLEITKOMMAARITHMETIK UND RUNDUNGSFEHLER

Durch die Exponenten- und die Mantissenlänge wird eine endliche Menge darstellbarer Zahlen definiert. Diese stellen in der Regel keinen abgeschlossenen Körper dar, da das Produkt zweier solcher Zahlen den darstellbaren Wertebereich verlassen kann. Um dem zu begegnen, fordert der IEEE Standard, dass jede arithmetische Operation

- intern immer mit höherer Genauigkeit durchgeführt[4] und
- danach korrekt gerundet werden soll.

Damit und mit den zusätzlichen "Sonderzahlen" ∞ und `NaN` wird sichergestellt, dass das Resultat jeder beliebigen arithmetischen Operation wieder als Gleitkommazahl darstellbar ist.

In den allermeisten Fällen wird die Rundung zur nächsten darstellbaren Zahl durchgeführt. Durch Einsetzen der Darstellung und Fallunterscheidung zeigt man die folgende Fehlerabschätzung.

[4] Dies kann zu einem arithmetischen Heisenberg-Effekt führen: Lässt man sich Zwischenergebnisse einer Rechnung, die eigentlich komplett in den Registern ablaufen würde, ausgeben, werden diese dafür mit geringerer Genauigkeit gespeichert, was zu Rundungsfehlern führen wird. Das Beobachten kann also die Ergebnisse der Rechnung verändern!

Satz 2.2. *Sei* $x \in [-b^{e_{\min}}, b^{e_{\min}}] \setminus \{0\} \subset \mathbb{R}$, *und sei* $fl(x)$ *die durch Rundung erhaltene Gleitkommazahl mit Mantissenlänge* $m \in \mathbb{N}$. *Dann gilt für den* relativen Fehler

$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2} b^{1-m} =: u.$$

Die rechte Seite gibt eine fundamentale Konstante für die Gleitkommazahldarstellung an, den *Rundungsfehler*. Sie entspricht auch der kleinsten Zahl, für die im Rechner $1 + u$ nicht gleich 1 ergibt. Eine verwandte häufig verwendete Konstante ist die *Maschinengenauigkeit* eps, die den Abstand zweier benachbarter Gleitkommazahlen zwischen 1 und 2 angibt; es gilt $\text{eps} = 2u$. Für single precision gilt etwa $\text{eps} = 2^{-23} \approx 1.192 \cdot 10^{-7}$, für double precision $\text{eps} = 2^{-52} \approx 4.220 \cdot 10^{-16}$.

Bemerkung. Man kann sich merken: single precision erlaubt etwa 6, double precision etwa 15, und extended precision etwa 18 verlässliche Nachkommastellen.

Wenn man dem IEEE Standard folgt, ergibt sich daraus eine Fehlerabschätzung für Gleitkommaoperationen.

Satz 2.3. *Sei* $\circ \in \{+, -, \times, /\}$ *eine arithmetische Operation, und* $\tilde{\circ}$ *ihre Realisierung in Gleitkommaarithmetik. Dann gilt*

$$x \,\tilde{\circ}\, y = fl(x \circ y) \leq (x \circ y)(1 + \text{eps}).$$

Dadurch, dass der entstandene Fehler von den Eingangsdaten abhängt, ist die Gleitkommaarithmetik in der Regel *weder assoziativ noch distributiv*!

Beispiel 2.4. Wir wählen der Einfachheit halber $b = 10, m = 3$. Eine arithmetische Gleitkomma-Addition wird durchgeführt, indem zuerst die Exponenten (auf den größeren) angeglichen werden, die Mantissen addiert werden, das Ergebnis normalisiert und zuletzt gerundet wird (und ähnlich für die Multiplikation).

(i) Wir wählen $x = 6590$, $y = 1$, $z = 4$, d. h. $x + y + z = 6595$. In Gleitkommadarstellung haben wir $fl(x) = 0.659 \cdot 10^4$, $fl(y) = 0.100 \cdot 10^1$, $fl(z) = 0.4 \cdot 10^1$, und die Gleitkomma-Summe ist

$$(x \,\tilde{+}\, y) \,\tilde{+}\, z = 0.659 \cdot 10^4 \,\tilde{+}\, z = 0.659 \cdot 10^4, \tag{2.1}$$

$$x \,\tilde{+}\, (y \,\tilde{+}\, z) = x \,\tilde{+}\, 0.500 \cdot 10^1 = 0.660 \cdot 10^4, \tag{2.2}$$

wobei das zweite Ergebnis der Rundung nach exakter Rechnung entspricht.

(ii) Wir wählen jetzt $x = 0.156 \cdot 10^2$, $y = 0.157 \cdot 10^2$ und berechnen $(x - y) \times (x - y) = 0.01$. In Gleitkommaarithmetik ergibt sich jedoch:

$$(x \tilde{-} y) \tilde{\times} (x \tilde{-} y) = 0.100 \cdot 10^{-1}, \tag{2.3}$$

$$(x \tilde{\times} x) \tilde{-} (x \tilde{\times} y) \tilde{-} (y \tilde{\times} x) \tilde{+} (y \tilde{\times} y) = -0.100 \cdot 10^{-1}. \tag{2.4}$$

(iii) Wir wählen $x = 0.73563$, $y = 0.73441$. Dann ist $x - y = 0.00122$, aber

$$fl(x) \tilde{-} fl(y) = 0.200 \cdot 10^{-2}.$$

Das letzte Phänomen wird *Auslöschung signifikanter Stellen* genannt. Hier wurde der relative Fehler in den Eingangsdaten von etwa 10^{-3} um nahezu vier Größenordnungen verstärkt. Die Subtraktion beinahe gleich großer Zahlen sollte also nach Möglichkeit vermieden werden, insbesondere, wenn danach durch eine sehr kleine Zahl geteilt wird.[5]

Offensichtlich spielt die Reihenfolge der Operationen also eine wesentliche Rolle; eine ungünstige Wahl kann die unvermeidlichen Fehler verstärken, bis das Ergebnis wertlos ist, eine günstige Wahl die Fehler hingegen dämpfen. Damit werden wir uns im nächsten Kapitel genauer beschäftigen.

WEITERFÜHRENDE LITERATUR

D. Goldberg (1991), What every computer scientist should know about floating-point arithmetic, *ACM Comput. Surv.* 23(1), 5–48, DOI: 10.1145/103162.103163.

J. H. Wilkinson (1994), *Rounding errors in algebraic processes*, Reprint of the 1963 original, Dover Publications, Inc., New York.

[5] Was in der Praxis auch unbedingt vermieden werden sollte, ist der Vergleich von zwei Gleitkommazahlen x und y auf Gleichheit, da eine unterschiedliche Reihenfolge in der Berechnung zweier an sich gleicher Zahlen in Gleitkommaarithmetik verschiedene Ergebnisse produzieren kann. Besser ist der Vergleich $|x - y| < \text{eps} \max(|x|, |y|)$.

3 FEHLERANALYSE

Wir haben gesehen, dass bei der konkreten Durchführung von mathematischen Operationen unweigerlich Fehler entstehen. Diese lassen sich unterteilen in

- *Datenfehler*: Die Eingaben vieler Berechnungen sind physikalische Messwerte oder empirische Untersuchungen, die durch Messungenauigkeiten immer fehlerbehaftet sind. Sie treten aber auch bei rein mathematischen Berechnungen wie $r^2\pi$ durch die Darstellung reeller Zahlen als Gleitkommazahlen auf.
- *Verfahrensfehler*: Häufig kann eine Aufgabe numerisch nur näherungsweise gelöst werden (etwa die Berechnung von e^x durch endlich Summation). Dabei entstehen Approximations- bzw. Diskretisierungsfehler (wozu auch Abbruchfehler bei iterativen Verfahren zählen).
- *Rundungsfehler*: Sind wie beschrieben die Folge der arithmetischen Grundoperationen mit Gleitkommazahlen, die die elementarsten Schritte jedes numerischen Algorithmus darstellen.

Hinzu kommen in der Praxis noch *Modellfehler*, die darauf beruhen, dass bei der Aufstellung des mathematischen Modells idealisierte Annahmen getroffen werden, die in der Realität nicht zutreffen (siehe Sleipner A), und *Darstellungsfehler*, die auf einer unzureichenden Interpretation des Ergebnisses beruhen (etwa durch Visualisierung komplexer dreidimensionaler Strömungen, in der wichtige Details zu Gunsten einer ansprechenden Darstellung vernachlässigt werden[1]). Nicht zu vernachlässigen sind auch *semantische Fehler*, die bei der Implementierung der Algorithmen entstehen (etwa Programmierfehler).

Da der Datenfehler außerhalb unserer Kontrolle liegt, müssen wir uns zuerst fragen, ob die gestellte Aufgabe prinzipiell unter dieser Voraussetzung sinnvoll lösbar ist. Erst dann können wir fragen, ob sich ein spezieller Algorithmus für deren Lösung eignet.

[1] Dieses oft in der numerischen Strömungsmechanik (engl. “computational fluid dynamics”, CFD) anzutreffende Phänomen hat den Spitznamen “colorful fluid dynamics” für diese Disziplin geprägt.

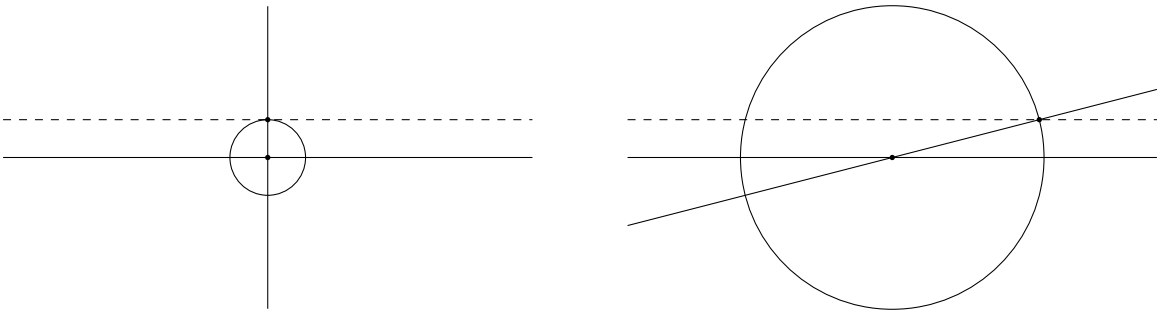


Abbildung 3.1: Schnittpunktbestimmung zweier Geraden. Links: gut konditioniert; rechts: schlecht konditioniert. (Kreisradius entspricht Fehler).

3.1 KONDITION EINES PROBLEMS

Die erste Frage beinhaltet, wie sensibel die Lösung eines Problem (unabhängig von dem verwendeten Verfahren) auf Störungen der Eingangsdaten reagiert. Führen kleine Änderungen der Eingabe auch nur zu kleinen Änderungen der Ausgabe, so nennt man ein Problem *gut konditioniert*. Können sich hingegen die Ausgaben nur leicht unterschiedlicher Daten stark unterscheiden, so spricht man von einem *schlecht konditionierten* Problem.

Ein einfaches Beispiel soll dies verdeutlichen:

Beispiel 3.1. Möchte man den Schnittpunkt zweier Geraden bestimmen (siehe Abb. 3.1), die annähernd senkrecht aufeinander stehen, so wird eine kleine Verschiebung einer Geraden (gestrichelte Linie) den Schnittpunkt nicht viel bewegen. Sind die Geraden hingegen fast parallel, ändert sich der Schnittpunkt stark, wenn eine Gerade verschoben wird. Das Problem ist also im Allgemeinen schlecht konditioniert. Wir sehen auch, dass die Kondition von den Eingabedaten abhängen kann.

Hinter der Kondition erkennt man sofort eine Stetigkeitsaussage; es liegt also nahe, den Begriff mathematisch zu präzisieren. Wir wollen dazu wieder ein Problem auffassen als eine Abbildung $f : X \to Y$, die für eine Eingabe x aus der Menge der zulässigen Eingabedaten X ein Resultat $y = f(x)$ aus der Menge der möglichen Resultate Y liefert, zusammen mit der gewünschten Eingabe x und einem maximalen Datenfehler[2] δ. Die fehlerbehaftete Eingabe bezeichnen wir mit $\tilde{x}$. Die Kondition drückt dann aus, wie stark sich die Eingabefehler auf das Ergebnis auswirken:

Definition 3.2. Für ein Problem (f, x, δ) ist die *absolute Kondition* die kleinste Zahl κ_{abs}, für die gilt

$$\|f(\tilde{x}) - f(x)\| \leq \kappa_{\text{abs}} \|\tilde{x} - x\| \qquad \text{für alle } \|\tilde{x} - x\| \leq \delta.$$

[2] Dieser wird für uns in der Regel der Maschinengenauigkeit eps entsprechen.

Die *relative Kondition* ist die kleinste Zahl κ_{rel}, für die gilt

$$\frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} \leq \kappa_{\text{rel}} \frac{\|\tilde{x} - x\|}{\|x\|} \qquad \text{für alle } \frac{\|\tilde{x} - x\|}{\|x\|} \leq \delta.$$

Ein gut konditioniertes Problem liegt dann vor, wenn $\kappa_{\text{rel}} \approx 1$ ist, der relative Fehler also in der Größenordnung des unvermeidlichen relativen Eingangsfehlers liegt. Da diese Definition unabhängig von Problem und Eingabedaten ist, wird man sich in der Regel nur für die relative Kondition interessieren.

Über die Taylorentwicklung von $f(x)$ erhält man sofort eine Abschätzung der Konditionen über die Ableitung:

Satz 3.3. *Ist f zweimal stetig differenzierbar, so gilt für das Problem (f, x, δ)*

$$\kappa_{\text{abs}} = \|f'(x)\| + O(\delta^2), \qquad \kappa_{\text{rel}} = \|f'(x)\| \frac{\|x\|}{\|f(x)\|} + O(\delta^2).$$

Analog lässt sich die Kondition eines Problems mit mehreren Eingabedaten $x_1, \ldots, x_n$ über die partiellen Ableitungen darstellen, z. B. durch

$$\kappa_{\text{abs}} = \max_{1 \leq j \leq n} \left\| \frac{\partial f(x)}{\partial x_j} \right\| + O(\delta^2), \quad \kappa_{\text{rel}} = \max_{1 \leq j \leq n} \left\| \frac{\partial f(x)}{\partial x_j} \cdot \frac{x_j}{f(x)} \right\| + O(\delta^2).$$

Für Matrizen werden wir noch eine spezielle Definition kennen lernen.

Beispiel 3.4. Für die Kondition der Addition (bzw. Subtraktion) betrachten wir für festes $y \neq 0$ die Funktion $f(x) := x + y$. Dann ist $f'(x) = 1$, es gilt also:

$$\kappa_{\text{rel}} = \frac{|x|}{|x + y|}.$$

Wir erkennen das Phänomen der Auslöschung wieder: ist $x \approx -y$, so wird der Nenner beliebig klein, und das Problem ist schlecht gestellt. Die Addition positiver Zahlen ist hingegen gut konditioniert, da dann immer $\frac{|x|}{|x+y|} \leq 1$ gilt.

Wir halten fest: Die Kondition ist eine Eigenschaft des Problems, nicht des Lösungsverfahrens. Es kann aber durchaus möglich sein, eine konkrete Fragestellung durch ein anderes (mathematisches) Problem auszudrücken, das besser konditioniert ist!

3.2 STABILITÄT EINES ALGORITHMUS

Für ein gut konditioniertes Problem[3] müssen wir nun entscheiden, ob das eingesetzte Verfahren *numerisch stabil* ist, d. h. neben den Eingabefehlern auch die Verfahrens- und Rundungsfehler nur geringen Einfluss auf das Resultat haben. Wir betrachten also ein numerisches Verfahren als Näherung $\tilde{f}$ eines Problems f, und fragen uns, wie sich $\tilde{f}(x)$ von $f(x)$ unterscheidet. Da Eingabefehler unvermeidlich sind, sind wir zufrieden, wenn $\|\tilde{f}(\tilde{x}) - f(x)\|$ in der Größenordnung von $\|f(\tilde{x}) - f(x)\|$ liegt. Dies lässt sich auf zwei Arten untersuchen:

3.2.1 VORWÄRTSANALYSE

Hierbei wird betrachtet, wie das Verfahren den durch die Kondition bestimmten, unausweichlichen, Fehler verstärkt oder dämpft.

Definition 3.5 (Vorwärtsstabilität). Sei $\tilde{f}$ ein numerisches Verfahren zur Lösung des Problems (f, x, δ). Der *Stabilitätsindikator* (der Vorwärtsanalyse) ist die kleinste Zahl $\sigma_v \geq 0$, für die gilt

$$\frac{\|\tilde{f}(\tilde{x}) - f(\tilde{x})\|}{\|f(\tilde{x})\|} \leq \sigma_v \kappa_{\text{rel}} \delta \qquad \text{für alle } \tilde{x} \text{ mit } \frac{\|\tilde{x} - x\|}{\|x\|} \leq \delta.$$

Ein Verfahren heißt *(vorwärts)stabil*, falls $\sigma_v \kappa_{\text{rel}}$ kleiner als die Anzahl der durchgeführten Elementarschritte ist.

Beispiel 3.6. Für die arithmetischen Operationen $+, -, \times, /$ und ihre Gleitkommarealisierungen $\tilde{+}, \tilde{-}, \tilde{\times}, \tilde{/}$ ist $\sigma_v \kappa_{\text{rel}} \leq 1$, denn nach Satz 2.3 gilt für die Gleitkommaarithmetik $x \circledcirc y \leq (x \circ y)(1 + \text{eps})$ und damit

$$\frac{|x \circledcirc y - x \circ y|}{|x \circ y|} \leq \frac{|(x \circ y)(1 + \text{eps}) - x \circ y|}{|x \circ y|} = \text{eps}.$$

Da für Gleitkommazahlen $\delta \geq \text{eps}$ angenommen werden kann (wir müssen mindestens mit Rundungsfehlern rechnen), folgt die Aussage.

In der Praxis ist es jedoch umständlich, die Konditionszahl eines Problems und die tatsächliche Anzahl der Schritte zu bestimmen. Für komplexere Verfahren bedient man sich deshalb oft einer anderen Methode.

[3] und nur für ein solches!

3.2.2 RÜCKWÄRTSANALYSE

Die Idee der Rückwärtsanalyse[4] ist es, den Verfahrensfehler auch wie einen Eingangsfehler zu behandeln: Das Ergebnis $\tilde{y} = \tilde{f}(\tilde{x})$ wird als das exakte Resultat $\tilde{y} = f(\hat{x})$ für gestörte Daten $\hat{x}$ angesehen. Es ist klar, dass das nur möglich ist, falls $\tilde{y}$ überhaupt eine mögliche Lösung des Problems f ist. Ist dies nicht der Fall, kann das Verfahren also sinnlose Ergebnisse liefern, betrachten wir den Algorithmus von vornherein als instabil.

Definition 3.7 (Rückwärtsstabilität). Sei $\tilde{f}$ ein numerisches Verfahren zur Lösung des Problems (f, x, δ). Der *Stabilitätsindikator* (der Rückwärtsanalyse) ist die kleinste Zahl $\sigma_r \geq 0$, für die für alle $\tilde{x}$ mit $\|\tilde{x} - x\|/\|x\| \leq \delta$ ein $\hat{x}$ existiert mit $\tilde{f}(\tilde{x}) = f(\hat{x})$ und

$$\frac{\|\hat{x} - \tilde{x}\|}{\|\tilde{x}\|} \leq \sigma_r \delta.$$

Ein Verfahren heißt *(rückwärts)stabil*, falls $\sigma_r \approx 1$ ist.

Während die Vorwärtsanalyse untersucht, wie gut das Ergebnis des Verfahrens mit der exakten Lösung übereinstimmt, überprüft die Rückwärtsanalyse, wie gut das Ergebnis das exakte Problem löst. Der Vorteil der Rückwärtsanalyse ist dabei, dass der Indikator direkt die Stabilität des Verfahrens angibt. Darüber hinaus gilt:

Lemma 3.8. *Für ein gut konditioniertes Problem (f, x, δ) ist jedes rückwärtsstabile Verfahren $\tilde{f}$ auch vorwärtsstabil.*

Beweis. Sei $\tilde{x}$ mit $\|\tilde{x} - x\| \leq \delta\|x\|$ gegeben. Nach Definition existiert für ein rückwärtsstabiles Verfahren dann ein $\hat{x}$ mit $\tilde{f}(\tilde{x}) = f(\hat{x})$ und

$$\frac{\|\hat{x} - \tilde{x}\|}{\|\tilde{x}\|} \leq \sigma_r \delta.$$

Hat f die Kondition κ_{rel}, dann erfüllt der relative Fehler im Ergebnis daher

$$\frac{\|\tilde{f}(\tilde{x}) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = \frac{\|f(\hat{x}) - f(\tilde{x})\|}{\|f(\tilde{x})\|} \leq \kappa_{\text{rel}} \frac{\|\hat{x} - \tilde{x}\|}{\|\tilde{x}\|} \leq \kappa_{\text{rel}} \sigma_r \delta.$$

Nach Annahme ist nun das Problem gut konditioniert und daher $\kappa_{\text{rel}} \approx 1$. Wegen der Rückwärtsstabilität ist weiter $\sigma_v := \sigma_r \approx 1$ und damit auch $\sigma_v \kappa_{\text{rel}} \approx 1$. Das Verfahren ist also auch vorwärtsstabil. □

[4] Für die Fehleranalyse wurde sie von James Hardy Wilkinson eingeführt; die Idee geht jedoch zurück auf Cornelius Lanczos.

Beispiel 3.9. Wir betrachten die Addition von drei Gleitkommazahlen x_1, x_2, x_3. Da für die Gleitkommaaddition die Reihenfolge eine Rolle spielt, legen wir diese als $(x_1 + x_2) + x_3$ fest. Wir haben also

$$\begin{aligned}(x_1 \tilde{+} x_2) \tilde{+} x_3 &= ((x_1 + x_2)(1 + \delta_2) + x_3)(1 + \delta_3) \\ &= x_1(1 + \delta_2 + \delta_3 + \delta_2\delta_3) + x_2(1 + \delta_2 + \delta_3 + \delta_2\delta_3) + x_3(1 + \delta_3) \\ &=: \hat{x}_1 + \hat{x}_2 + \hat{x}_3\end{aligned}$$

für je ein $|\delta_2|, |\delta_3| < \mathsf{eps}$. Nun ist für $\mathsf{eps} < 1$:

$$\begin{aligned}\frac{|\hat{x}_1 - x_1|}{|x_1|} &= |\delta_2 + \delta_3 + \delta_2\delta_3| &&< 3\mathsf{eps}, \\ \frac{|\hat{x}_2 - x_2|}{|x_2|} &= |\delta_2 + \delta_3 + \delta_2\delta_3| &&< 3\mathsf{eps}, \\ \frac{|\hat{x}_3 - x_3|}{|x_3|} &= |\delta_3| &&< \mathsf{eps}.\end{aligned}$$

Setzen wir $x := (x_1, x_2, x_3) \in \mathbb{R}^3$ und analog für $\hat{x}$, so folgt z. B.

$$\begin{aligned}\frac{\|\hat{x} - x\|_1}{\|x\|_1} &= \frac{|\hat{x}_1 - x_1|}{\|x\|_1} + \frac{|\hat{x}_2 - x_2|}{\|x\|_1} + \frac{|\hat{x}_3 - x_3|}{\|x\|_1} \\ &\leq \frac{|\hat{x}_1 - x_1|}{|x_1|} + \frac{|\hat{x}_2 - x_2|}{|x_2|} + \frac{|\hat{x}_3 - x_3|}{|x_3|} \\ &< 7\mathsf{eps}.\end{aligned}$$

Wegen $\delta \geq \mathsf{eps}$ ist die Gleitkommaaddition also auch rückwärtsstabil. Man sieht außerdem, dass die zuerst addierten Terme eine größere Fehlerverstärkung erfahren. Es ist also numerisch vorteilhaft, bei der Addition mehrerer Terme mit dem betragskleinsten zu beginnen.

WEITERFÜHRENDE LITERATUR

N. J. Higham (2002), *Accuracy and stability of numerical algorithms*, 2. Aufl., Society for Industrial & Applied Mathematics (SIAM), Philadelphia, PA.

Teil II

NUMERISCHE LINEARE ALGEBRA

4 GRUNDLAGEN

In diesem Kapitel werden wichtige Begriffe und Werkzeuge behandelt, die speziell für die numerische Lösung von Problemen aus der Linearen Algebra eine Rolle spielen. Dabei werden im Wesentlichen die allgemeinen Betrachtungen des ersten Teils konkretisiert.

4.1 MATRIXNORMEN

Jede Matrix $A \in \mathbb{R}^{n\times n}$ lässt sich als Vektor im Raum $\mathbb{R}^{(n^2)}$ auffassen, womit sich jede der bereits bekannten Vektornormen zur Untersuchung einer Matrix verwenden ließe. Jedoch haben Matrizen ein zweites Gesicht: Sie sind auch lineare Operatoren, die auf Vektoren wirken – und diese Sichtweise wird für uns weitaus relevanter sein. Wir verlangen von einer Matrixnorm daher über die Normaxiome in Definition 1.1 hinaus zusätzliche Eigenschaften.

Definition 4.1. Eine Matrixnorm $\|\cdot\|$ auf $\mathbb{R}^{n\times n}$ heißt *submultiplikativ*, wenn für alle $A, B \in \mathbb{R}^{n\times n}$ gilt

$$\|AB\| \leq \|A\|\|B\|.$$

Definition 4.2. Eine Matrixnorm $\|\cdot\|_M$ auf $\mathbb{R}^{n\times n}$ heißt *verträglich* mit einer Vektornorm $\|\cdot\|_V$, wenn für alle $A \in \mathbb{R}^{n\times n}$ und $x \in \mathbb{R}^n$ gilt

$$\|Ax\|_V \leq \|A\|_M\|x\|_V.$$

Die wichtigsten Beispiele für Matrixnormen sind die *Operatornormen*.

Definition 4.3. Für eine Vektornorm $\|\cdot\|_V$ auf $\mathbb{R}^n$ heißt die durch

$$\|A\|_M := \sup_{\|x\|\neq 0} \frac{\|Ax\|_V}{\|x\|_V}$$

definierte Norm die *natürliche* oder die (durch $\|\cdot\|_V$) *induzierte* Norm.

Satz 4.4. *Jede natürliche Norm ist submultiplikativ und mit der sie induzierenden Vektornorm verträglich.*

Die am häufigsten gebrauchten Normen sind auch hier die p-Normen, wobei die Euklidische Norm wieder hervorgehoben ist.

Definition 4.5. Der *Spektralradius* einer Matrix $A \in \mathbb{R}^{n\times n}$ ist definiert als

$$\rho(A) = \max_{1\le i\le n} |\lambda_i|,$$

wobei λ_i die Eigenwerte von A sind.

Beispiel 4.6 (Induzierte p-Normen auf $\mathbb{R}^{n\times n}$).

(i) $\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}|$ (*Spaltensummennorm*)

(ii) $\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|$ (*Zeilensummennorm*)

(iii) $\|A\|_2 = \sqrt{\rho(A^T A)}$ (*Spektralnorm*).

Nicht alle Matrixnormen sind induzierte Normen. Auch die folgende Norm, genannt *Frobeniusnorm*, ist submultiplikativ und verträglich mit der Euklidischen Norm:[1]

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2}.$$

Der Wert der Frobeniusnorm besteht darin, dass sie eine scharfe obere Schranke für die – schwer zu berechnende – Spektralnorm darstellt. Wie Vektornormen sind auch alle Matrixnormen äquivalent; insbesondere gelten die folgenden (optimalen) Abschätzungen:

Lemma 4.7. *Für $A \in \mathbb{R}^{n\times n}$ gilt*

(i) $\|A\|_2 \le \|A\|_F \le \sqrt{n}\|A\|_2$,

(ii) $\frac{1}{\sqrt{n}}\|A\|_\infty \le \|A\|_2 \le \sqrt{n}\|A\|_\infty$,

(iii) $\frac{1}{\sqrt{n}}\|A\|_1 \le \|A\|_2 \le \sqrt{n}\|A\|_1$.

Lemma 4.8. *Sei $A \in \mathbb{R}^{n\times n}$ und $\|\cdot\|$ eine induzierte Norm. Dann gilt:*

$$\rho(A) \le \|A\|$$

Beweis. Sei λ der betragsgrößte Eigenwert von A und v der zugehörige Eigenvektor. Dann gilt mit der induzierenden Vektornorm $\|\cdot\|_V$

$$|\lambda|\|v\|_V = \|\lambda v\|_V = \|Av\|_V \le \|A\|\|v\|_V.$$

Da Eigenvektoren nach Definition ungleich Null sind, können wir durch $\|x\|_V$ dividieren und erhalten die Aussage. □

[1] Sie entspricht der Euklidischen Norm von $A \in \mathbb{R}^{n\times n}$, aufgefasst als Vektor in $\mathbb{R}^{n^2}$.

4.2 SPEZIELLE MATRIZEN

Wir werden im Laufe der nächsten Kapitel erkennen, dass die Stabilität und Effizienz von Verfahren zur Lösung von Problemen der Linearen Algebra wesentlich von der Struktur der gegebenen Matrix abhängt. Hier betrachten wir zwei der wichtigsten Beispiele von Matrizen mit günstiger Struktur.

Definition 4.9. Eine Matrix symmetrische $A \in \mathbb{R}^{n\times n}$ heißt *positiv definit*, falls gilt[2]

$$x^T Ax > 0 \qquad \text{für alle } x \in \mathbb{R}^n \setminus \{0\}.$$

Durch Einsetzen geeigneter Vektoren x zeigt man folgende Eigenschaften positiv definiter Matrizen.

Satz 4.10. *Sei $A = (a_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann gilt:*

(i) $a_{ii} > 0$ *für alle* $1 \le i \le n$;

(ii) $a_{ij}^2 < a_{ii}a_{jj}$, *für alle* $i \neq j$, $1 \le i, j \le n$;

(iii) es existiert ein $k \in \{1, \dots, n\}$, *so dass* $\max_{1\le i,j\le n}|a_{ij}| = a_{kk}$.

Die positive Definitheit lässt sich einfach über die Eigenwerte charakterisieren.

Satz 4.11. *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch. Dann ist A positiv definit genau dann, wenn $\lambda_i > 0$ gilt für alle Eigenwerte λ_i, $1 \le i \le n$, von A.*

Für eine wichtige Unterklasse lässt sich die Definitheit direkt an den Matrixelementen erkennen.

Definition 4.12. $A \in \mathbb{R}^{n\times n}$ heißt *streng diagonal dominant*, falls gilt

$$|a_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \qquad \text{für alle } 1 \le i \le n. \tag{4.1}$$

Satz 4.13. *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und streng diagonal dominant mit strikt positiven Diagonaleinträgen. Dann ist A positiv definit.*

[2] Ist auch Gleichheit möglich, so spricht man von semidefiniten Matrizen; die umgekehrte Ungleichung charakterisiert negativ definite Matrizen. Ist keine dieser Aussage möglich, nennt man die Matrix indefinit.

4.3 DIE KONDITIONSZAHL EINER MATRIX

Der zentrale Begriff für die Untersuchung der Kondition von Problemen der linearen Algebra ist die Konditionszahl einer Matrix. Wir werden diese für die Lösung linearer Gleichungssysteme motivieren; sie ist aber auch für viele andere Problemstellungen, die wir im Weiteren betrachten werden, relevant.

Sei also $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$ gegeben, und derjenige Vektor $x \in \mathbb{R}^n$ gesucht, der $Ax = b$ erfüllt. Existiert überhaupt solch ein x, so lässt es sich über die inverse Matrix A^{-1} berechnen; es ist $x = A^{-1}b$. Wie in Abschnitt 3.1 schreiben wir dies als $f : b \mapsto A^{-1}b$. Dies ist eine lineare Funktion, es gilt also $f'(b) = A^{-1}$. Nach Satz 3.3 ist deshalb $\kappa_{\text{abs}} = \|A^{-1}\|$ und

$$\kappa_{\text{rel}} = \frac{\|b\|}{\|A^{-1}b\|}\|A^{-1}\| = \frac{\|Ax\|}{\|x\|}\|A^{-1}\| \le \|A\|\|A^{-1}\|.$$

Dies motiviert die folgende Definition:

Definition 4.14. Für eine reguläre (d. h. invertierbare) Matrix $A \in \mathbb{R}^{n\times n}$ und eine Matrixnorm $\|\cdot\|$ heißt

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

(relative) *Konditionszahl* von A bezüglich $\|\cdot\|$.

Dabei werden wir (falls nicht anders angegeben) im Weiteren annehmen, dass wir für die Kondition eine induzierte Matrixnorm zugrunde legen. Speziell für die Spektralnorm erhält man eine nützliche Darstellung:

Lemma 4.15. *Die Konditionszahl einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ bezüglich einer induzierten Matrixnorm $\|\cdot\|_p$ ist*

$$\kappa_p(A) = \frac{\max\|Ax\|_p}{\min\|Ax\|_p},$$

wobei das Maximum und Minimum über alle $x \in \mathbb{R}^n$ mit $\|x\|_p = 1$ genommen wird.

Ist A symmetrisch und positiv definit, so ist

$$\kappa_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}},$$

d. h. gleich dem Verhältnis zwischen betragsgrößtem und betragskleinstem Eigenwert von A.

Beweis. Wir zeigen zunächst, dass gilt $\|A\|_p = \max_{\|x\|_p=1}\|Ax\|_p$. Da die rechte Seite offensichtlich kleiner als die linke ist (da das Maximum über eine kleinere Menge gebildet wird), reicht es zu zeigen, dass das Maximum auf der linken Seite (auch) für einen Vektor der Norm 1 angenommen wird. Sei dafür $\bar{x} \in \mathbb{R}^n \setminus \{0\}$ mit $\|A\|_p = \|Ax\|_p/\|x\|_p$. Wir setzen

nun $\alpha := \|\bar{x}\|_p > 0$ und $\tilde{x} := \bar{x}/\|\bar{x}\|_p$ so, dass $\bar{x} = \alpha\tilde{x}$ und $\|\tilde{x}\|_p = 1$ gilt. Daraus folgt wegen der Linearität von A und der Homogenität der Norm

$$\|A\|_p = \frac{\|A\bar{x}\|_p}{\|\bar{x}\|_p} = \frac{\alpha\|A\tilde{x}\|_p}{\alpha\|\tilde{x}\|_p} = \|A\tilde{x}\|_p$$

und damit die Behauptung. Dies liefert den Zähler für κ_p.

Für den Nenner gehen wir analog vor. Da A regulär und damit bijektiv ist, können wir alle $y \in \mathbb{R}^n$ eindeutig darstellen als $y = Ax$ für ein $x \in \mathbb{R}^n$. Daraus folgt

$$\|A^{-1}\|_p = \max_{y\in\mathbb{R}^n\setminus\{0\}} \frac{\|A^{-1}y\|_p}{\|y\|_p} = \max_{x\in\mathbb{R}^n\setminus\{0\}} \frac{\|x\|_p}{\|Ax\|_p} = \max_{\|x\|_p=1} \frac{1}{\|Ax\|_p} = \frac{1}{\min_{\|x\|_p=1}\|Ax\|_p},$$

wobei wir im vorletzten Schritt genauso wie für $\|A\|_p$ vorgegangen sind.

Den Spezialfall $p = 2$ beweist man nun dadurch, dass das Maximum bzw. Minimum für den jeweiligen (normierten) Eigenvektor angenommen wird. □

Wir haben bislang noch nicht berücksichtigt, dass in der Regel auch die Matrixeinträge von A, etwa durch Darstellung als Maschinenzahlen, mit Fehlern behaftet sind. Wir müssen also untersuchen, wie sich die Lösung x verhält, wenn wir statt $Ax = b$ das System $\tilde{A}x := (A + \delta A)x = b$ lösen. Der folgende Satz[3] besagt insbesondere, dass eine kleine Störung die Invertierbarkeit einer Matrix nicht ändert.

Satz 4.16. *Seien $A, \tilde{A} \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$. Sei weiterhin A invertierbar und $x \in \mathbb{R}^n$ Lösung von $Ax = b$.*

Gilt $\|A^{-1}\|\|\tilde{A} - A\| < 1$, so ist auch $\tilde{A}$ invertierbar, und für die Lösung $\tilde{x} \in \mathbb{R}^n$ von $\tilde{A}\tilde{x} = b$ gilt für eine beliebige Vektornorm und der durch sie induzierten Matrixnorm

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{\|\tilde{A}-A\|}{\|A\|}} \frac{\|\tilde{A} - A\|}{\|A\|}.$$

Die Konditionszahl bestimmt also die Kondition der Lösung von linearen Gleichungssystemen, sowohl gegenüber Störungen der rechten Seite als auch der Matrix selber. Eine Möglichkeit, die Konditionszahl einer Matrix zu verbessern, liegt in der Skalierung der einzelnen Zeilen. Zwar ist keine allgemeine Methode bekannt, um dies für beliebige Normen zu erreichen, aber für die Maximumsnorm können wir eine solche angeben:

[3] siehe z. B. G. Hämmerlin & K.-H. Hoffmann (1994), *Numerische Mathematik*, 4. Aufl., Springer-Verlag, Berlin, Seite 79–80

Sei $D \in \mathbb{R}^{n\times n}$ eine Diagonalmatrix mit Diagonaleinträgen

$$d_i = \left(\sum_{j=1}^{n} |a_{ij}| \right)^{-1}.$$

Die *zeilenäquilibrierte* Matrix DA hat dann die Maximumsnorm

$$\|DA\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} \left(\sum_{k=1}^{n} |a_{ik}| \right)^{-1} |a_{ij}| = \max_{1\le i\le n} \left(\sum_{j=1}^{n} |a_{ij}| \right)^{-1} \sum_{j=1}^{n} |a_{ij}| = 1,$$

und man kann zeigen, dass $\kappa_\infty(DA) < \kappa_\infty(\tilde{D}A)$ für jede andere Diagonalmatrix $\tilde{D}$ (inklusive der Identitätsmatrix) gilt. Analog lässt sich (durch Multiplikation von rechts mit einer ähnlichen Diagonalmatrix) eine *Spaltenäquilibrierung* durchführen, die die Konditionszahl bezüglich der Spaltensummennorm minimiert.

5 LINEARE GLEICHUNGSSYSTEME

Das Lösen linearer Gleichungssysteme gehört zu den wichtigsten Aufgaben der numerischen Mathematik. Zum einen sind viele in der Anwendung erfolgreiche mathematische Modelle linearer Natur (etwa das Leontief-Modell in den Wirtschaftswissenschaften, die Kirchhoffschen Gesetze in der Elektrotechnik oder Bilanzgleichungen in der Chemie). Zum anderen basieren fast alle numerischen Verfahren der höheren Mathematik (etwa die Lösung partieller Differentialgleichungen) letztlich auf der Reduktion auf lineare Gleichungssysteme. Es ist also von zentralem Interesse, stabile und effiziente Verfahren für dieses Problem zu haben.

Wir beschäftigen uns daher in diesem Kapitel mit folgendem Problem:

- Gegeben:
$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \dots & \dots & a_{nn} \end{pmatrix} \in \mathbb{R}^{n\times n}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \in \mathbb{R}^n,$$
- Gesucht: $x \in \mathbb{R}^n$, so dass $Ax = b$.

Aus der linearen Algebra wissen wir, wann solch ein Gleichungssystem lösbar ist.

Satz 5.1. *Sei $A \in \mathbb{R}^{n\times n}$ und $b \in \mathbb{R}^n$. Ist $\det(A) \neq 0$, so existiert genau ein $x \in \mathbb{R}^n$ mit $Ax = b$.*

Über die Determinante lässt sich theoretisch auch die Lösung des Gleichungssystems berechnen; die Cramersche Regel besagt, dass $x_j = \frac{\det(A_j)}{\det(A)}$ ist, wobei A_j die Matrix ist, die durch Ersetzen der j-ten Spalte von A durch b entsteht. Dies ist allerdings ein denkbar unpraktikables Verfahren: Die notwendige Berechnung der $n+1$ Determinanten erfordert $O((n+1)!)$ Operationen; selbst moderne Graphikprozessoren, die 1 Teraflop – also eine Billion Gleitkommaoperationen pro Sekunde – leisten können, würden für die Lösung eines Gleichungssystems mit $n = 20$ Zeilen 1.6 Jahre benötigen; für $n = 22$ wären bereits 820 Jahre nötig; und $n = 30$ wäre in 266 Billionen Jahren noch nicht fertig berechnet.

Die in der Praxis auftretenden Matrizen haben aber üblicherweise Größen von mehreren Hundert bis zu mehreren Millionen Zeilen bzw. Spalten. Die in Frage kommenden Verfahren

müssen also auch solche Probleme behandeln können. Man unterscheidet dabei zwei Klassen von Verfahren:

1. *Direkte Verfahren* liefern nach endlich vielen Schritten die Lösung (wobei evtl. Rundungsfehler zu berücksichtigen sind).
2. *Iterative Verfahren* liefern ausgehend von einer Anfangsnäherung nach jedem Schritt eine verbesserte Näherung. Zu den Rundungsfehlern treten hier also noch Verfahrensfehler; dafür weisen diese Methoden eine günstigere Komplexität auf.

In dieser Vorlesung beschränken wir uns auf die klassischen Verfahren der ersten Klasse. Zunächst betrachten wir eine Klasse von Matrizen, für die das zugehörige Gleichungssystem sehr leicht lösbar ist. Im nächsten Schritt führen wir dann den allgemeinen Fall auf dieses Problem zurück.

5.1 AUFLÖSUNG VON DREIECKSMATRIZEN

Idee: Haben wir eine Matrix der Form $A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}$ gegeben, so können wir die Lösung von $Ax = b$ einfach durch sukzessives Einsetzen bestimmen: Zuerst finden wir $x_2 = b_2/a_{22}$, und setzen $x_1 = (b_1 - a_{12}x_2)/a_{11}$. Für den allgemeinen Fall $A \in \mathbb{R}^{n\times n}$ führt das auf die folgende Definition.

Definition 5.2. Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt

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

Ist A eine obere Dreiecksmatrix, so kann man das Gleichungssystem $Ax = b$ durch *Rückwärtssubstitution* bestimmen.

Algorithmus 5.1 : Rückwärtssubstitution

Input : a_{ij}, b_j

1 **for** $j = n, \dots, 1$ **do**

2 $\quad x_j \leftarrow \left(b_j - \sum_{k=j+1}^{n} a_{jk}x_k\right)/a_{jj}$

Output : x_j

Man verifiziert leicht durch Einsetzen, dass dies die gewünschte Lösung liefert. Dabei müssen $\frac{1}{2}n(n-1)$ Additionen und $\frac{1}{2}n(n+1)$ Multiplikationen durchgeführt werden, die Komplexität ist also $O(n^2)$. Sind alle Diagonalelemente $a_{jj} \neq 0$, so ist der Algorithmus durchführbar; dies ist (wegen der Dreiecksform) gleichbedeutend mit $\det(A) \neq 0$.

Analog führt man für eine untere Dreiecksmatrix eine *Vorwärtssubstitution* durch.

Algorithmus 5.2 : Vorwärtssubstitution

Input : a_{ij}, b_j

for $j = 1, \ldots, n$ **do**

  $x_j \leftarrow \left(b_j - \sum_{k=1}^{j-1} a_{jk} x_k\right) / a_{jj}$

Output : x_j

5.2 GAUSS-ELIMINATION

Idee: Eine allgemeine Matrix wird durch Zeilenumformungen auf obere Dreiecksform gebracht, so dass sich Rückwärtssubstitution zur Lösung anwenden lässt.[1]

Zur Herleitung des Verfahrens nehmen wir zunächst an, dass $a_{11} \neq 0$ gilt. Dann können wir geeignete Vielfache der ersten Zeile der Matrix A und der rechten Seite b von allen weiteren subtrahieren:

for $i = 2, \ldots, n$ **do**

  $l_i^{(1)} = \frac{a_{i1}}{a_{11}}$

  for $j = 1, \ldots, n$ **do**

    $a_{ij}^{(2)} = a_{ij} - l_i^{(1)} a_{1j}$

  $b_i^{(2)} = b_i - l_i^{(1)} b_1$

und wir erhalten eine neue Matrix $A^{(2)}$ und rechte Seite $b^{(2)}$:

$$A^{(2)} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix}, \qquad b^{(2)} = \begin{pmatrix} b_1 \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

Ist nun $a_{22}^{(2)} \neq 0$, so wiederholen wir diesen Schritt mit der Teilmatrix $\bar{A}^{(2)} = (a_{ij}^{(2)})_{i,j=2}^{n}$ und dem Vektor $\bar{b}^{(2)} = (b_i^{(2)})_{i=2}^{n}$. Da wir in jedem Schritt k nur die Matrixelemente mit $i, j \geq k$ betrachten, erhalten wir also garantiert nach $n-1$ Schritten eine obere Dreiecksmatrix $A^{(n)}$, die wir per Rückwärtssubstitution auflösen können. Dabei sind im Schritt k insgesamt $(n-k)((n-k+1)+1)$ Multiplikationen und Subtraktionen sowie $(n-k)$ Divisionen nötig; der komplette Aufwand für die Gauß-Elimination beträgt also $O(n^3)$. Der Algorithmus

[1] Carl Friedrich Gauß hat dieses Verfahren 1810 in einer Abhandlung zur Untersuchung des Orbits des Planetoiden Pallas verwendet (zur Lösung eines Gleichungssystems für sechs Unbekannte, das er aus einer Reihe von Beobachtungen ableitete). Das allgemeine Verfahren ist allerdings bereits 200 v.Chr. in dem chinesischen Text *Jiŭzhāng Suànshù* ("Neun Kapitel über die mathematische Kunst", dem chinesischen Äquivalent von Euklids *Elementen*) unter dem Begriff *fāngchéng shù* ("Vorschriften über quadratische Felder") beschrieben.

scheitert, wenn wir in einem Schritt durch Null dividieren müssen. Für streng diagonal dominante Matrizen können wir das im ersten Schritt ausschliessen; per Induktion zeigt man, dass diese Eigenschaft auch für alle modifizierten Matrizen $A^{(k)}$ erhalten bleibt. Wir halten fest:

Satz 5.3. *Für eine streng diagonal dominante Matrix $A \in \mathbb{R}^{n\times n}$ ist die Gauß-Elimination durchführbar.*

Ist A nicht diagonal dominant, können selbst für invertierbare Matrizen[2] Nullelemente auf der Diagonalen auftreten. Dies kann durch einen Zeilentausch (entspricht Umsortieren der Gleichungen) umgangen werden: Taucht im Schritt k ein Diagonalelement $a_{kk}^{(k)} = 0$ auf, so sucht man ein $a_{mk}^{(k)} \neq 0$ mit $m > k$ und vertauscht die Zeilen m und k. (Existiert kein so ein Element, so hat $A^{(k)}$ nicht vollen Rang, und wir werden sehen, dass dies nicht möglich ist, wenn A selber vollen Rang hat.) Dieses Verfahren heißt *Pivotsuche*, $a_{mk}^{(k)}$ *Pivotelement*. Solches Vorgehen ist auch sinnvoll, wenn $a_{kk}^{(k)}$ sehr klein ist, um Fehlerverstärkung bei der Berechnung von $l^{(k)}$ zu vermeiden. Daher wählt man sinnvollerweise als Pivotelement das betragsgrößte $a_{mk}^{(k)}$.[3]

Müssen wir mehrere Gleichungssysteme lösen, bei denen sich nur die rechte Seite ändert, sollten die $\mathcal{O}(n^3)$ Operationen nur einmal durchgeführt werden. Dies lässt sich erreichen, indem man die nötigen Zeilenumformungen in der Form von Matrixmultiplikationen speichert.

5.3 LR-ZERLEGUNG

Wir betrachten wieder zuerst den Fall ohne Pivotisierung. Im Schritt k der Gauß-Elimination wird von allen Zeilen mit Index $i > k$ ein Vielfaches der k-ten Zeile abgezogen, was sich als Multiplikation mit einer geeigneten Matrix darstellen lässt. Durch explizites Ausschreiben der Multiplikationen vergewissert man sich, dass gilt

$$A^{(k+1)} = L_k A^{(k)}, \qquad b^{(k+1)} = L_k b^{(k)},$$

mit $A^{(1)} = A$ und

$$L_k = \begin{pmatrix} 1 & 0 & \dots & \dots & & 0 \\ 0 & \ddots & & & & 0 \\ \vdots & & 1 & & & \vdots \\ \vdots & & -l_{k+1}^{(k)} & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & 0 \\ 0 & \dots & -l_n^{(k)} & \dots & \dots & 1 \end{pmatrix} = I - \underbrace{(0, \dots, l_{k+1}^{(k)}, \dots, l_n^{(k)})^T}_{=:l^{(k)}} \underbrace{(0, \dots, 0, 1, 0, \dots, 0)}_{=:e_k^T},$$

[2] z. B. $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

[3] Merke: Die Pivotsuche verbessert nur die Stabilität des *Verfahrens*, eine Zeilenäquilibrierung dagegen nur die Kondition des *Problems*. Man wird also beide Techniken anwenden.

wobei $l_i^{(k)} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$ genau die in der Gauß-Elimination berechneten Faktoren sind.

Solche Matrizen, die sich nur in einer Spalte von der Einheitsmatrix unterscheiden, heißen *Frobeniusmatrizen*. Wir haben also nach $n-1$ Schritten eine obere Dreiecksmatrix

$$A^{(n)} = L_{n-1} \cdots L_1 A.$$

Durch Nachrechnen sieht man die folgenden Eigenschaften.

Lemma 5.4. *Es sei $L_k = I - l^{(k)} e_k^T$ eine Frobeniusmatrix. Dann gilt:*

(i) $L_k^{-1} = I + l^{(k)} e_k^T$,

(ii) $L := \prod_{k=1}^{n-1} L_k^{-1} = I + \sum_{k=1}^{n-1} l^{(k)} e_k^T$.

Damit ist L also eine untere Dreiecksmatrix mit nicht verschwindenden Diagonaleinträgen ($= 1$), und wir können A als Produkt von unteren und oberen Dreiecksmatrizen schreiben

$$A = LR \quad \text{mit} \quad R := A^{(n)}, \quad L := (l_{ij})_{i,j=1}^n, \quad l_{ij} := \begin{cases} \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}} & i > j, \\ 1 & i = j, \\ 0 & i < j. \end{cases}$$

Diese Faktorisierung nennt man *LR-Zerlegung*; sie hat offensichtlich die gleiche Komplexität $O(n^3)$ wie die Gauß-Elimination. Wollen wir jetzt die Gleichung $Ax = b$ lösen, so lässt sich das schrittweise durchführen:

1. Berechne L, R, so dass $A = LR$ gilt,
2. Löse $Ly = b$ durch Vorwärtssubstitution,
3. Löse $Rx = y$ durch Rückwärtssubstitution.

Dabei ist der erste (und aufwendigste) Schritt unabhängig von b, was eine Ersparnis ermöglicht, falls man ein Gleichungssystem für mehrere rechte Seiten lösen möchte.

Nun soll auch die Pivotisierung berücksichtigt werden. Zunächst folgt aus Lemma 5.4 (i), dass alle L_k invertierbar sind; damit hat auch $A^{(k)}$ vollen Rang, wenn A – wie angenommen – vollen Rang hat. Dies impliziert, dass wir stets ein Pivot-Element $a_{mk}^{(k)} \neq 0$ finden können. Um dies wieder in Form einer LR-Zerlegung bringen zu können, müssen wir die Vertauschung

von zwei Zeilen k und m auch als Matrixmultiplikation schreiben. Es gilt $\tilde{A} = P_{km}A$ für die *Permutationsmatrix*

$$P_{km} = \begin{pmatrix} 1 & 0 & \dots & & & & & & & 0 \\ 0 & \ddots & & & & & & & & \\ \vdots & & 1 & & & & & & & \\ & & & 0 & \dots & \dots & \dots & 1 & & \\ & & & \vdots & \ddots & & & \vdots & & \\ & & & \vdots & & 1 & & \vdots & & \\ & & & \vdots & & & \ddots & \vdots & & \\ & & & 1 & \dots & \dots & \dots & 0 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ 0 & & & & & & & & \dots & 0 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow k\text{-te Zeile} \\ \\ \\ \\ \leftarrow m\text{-te Zeile} \\ \vdots \\ 0 \\ 1 \end{matrix}.$$

Solche Matrizen haben folgende Eigenschaften:

(i) $\det(P_{km}) = -1$;

(ii) $P_{km}^{-1} = P_{km}$;

(iii) P_{km} vertauscht nur Zeilen mit Index $m \geq k$.

Wird also im Schritt k die notwendige Zeilenvertauschung durch $P_k := P_{km}$ für ein $m \geq k$ (d. h. $P_k = I$ für $k = m$) durchgeführt, so ist $A^{(k+1)} = L_k P_k A^{(k)}$. Wegen $\det(L_k) = 1$ (untere Dreiecksform mit Diagonaleinträgen 1) und Eigenschaft (i) hat $A^{(k+1)}$ den selben Rang wie $A^{(k)}$; es muss also stets ein $a_{mk}^{(k+1)} \neq 0$ für $m \geq k$ existieren, so dass die Pivotsuche erfolgreich ist. Nach $n-1$ Schritten haben wir daher

$$A^{(n)} = L_{n-1}P_{n-1} \cdots L_1 P_1 A.$$

Wir zeigen jetzt, dass wir dies wieder als LR-Zerlegung auffassen können, indem wir die P_j "an den L vorbeiziehen": Aus Eigenschaften (ii) und (iii) sowie der Definition der Frobeniusmatrizen gilt für alle $m < k$

$$P_k L_m P_k = P_k I P_k - P_k(l^{(m)} e_m^T)P_k = I - (P_k l^{(m)})(P_k^T e_m)^T = I - \tilde{l}^{(m)} e_m^T =: \tilde{L}_m, \tag{5.1}$$

d. h. $\tilde{L}_m$ ist ebenfalls eine Frobeniusmatrix. Daraus folgt $P_k L_m = \tilde{L}_m P_k$ für $m < k$ und somit

$$A^{(n)} = L_{n-1}P_{n-1} \cdots L_2 P_2 L_1 P_1 A = L_{n-1}P_{n-1} \cdots L_2 \tilde{L}_1 P_2 P_1 A, \quad \tilde{L}_1 := P_2 L_1. \tag{5.2}$$

Dies lässt sich wiederholen für $L_2, \dots, L_{n-1}$. Multipliziert man wieder beide Seiten mit den Inversen der $\tilde{L}_k$, $k = 1, \dots, n-1$, so erhält man

$$\tilde{L} A^{(n)} := \tilde{L}_1^{-1} \cdots \tilde{L}_{n-1}^{-1} A^{(n)} = P_{n-1} \cdots P_1 A =: PA.$$

Da $\tilde{L}$ nach (5.1) und Lemma 5.4 eine untere Dreiecksmatrix ist, haben wir den folgenden Satz bewiesen.

Satz 5.5. *Für jede reguläre Matrix $A \in \mathbb{R}^{n\times n}$ existiert eine Permutationsmatrix P, eine untere Dreiecksmatrix L und eine obere Dreiecksmatrix R, so dass gilt*

$$PA = LR.$$

Die Stabilität der LR-Zerlegung lässt sich mit Hilfe der Rückwärtsanalyse und Satz 4.16 beweisen.

Der folgende Algorithmus berechnet die LR-Zerlegung einer Matrix A, die dabei überschrieben wird. Dadurch wird gewährleistet, dass die Permutationen auch auf die Matrixelemente von L angewendet werden (siehe (5.2)). Die Permutation P speichert man am einfachsten dadurch, dass jede Zeilenvertauschung auch auf einen Permutationsvektor $p = (1, \dots, n)^T$ angewendet wird; nach Ende des Algorithmus geben die Einträge von p die Reihenfolge der Einträge von Pb an (z. B. ist für $p = (3, 2, 1)$ dann $Pb = (b_3, b_2, b_1)$).

Algorithmus 5.3 : LR-Zerlegung mit Pivotisierung

Input : a_{ij}

$p \leftarrow (1, \dots, n)$ `// Initialisiere Permutationsvektor`
for $j = 1, \dots, n-1$ **do**
  Finde $s \geq j$ so dass $|a_{sj}| = \max_{j\leq i\leq n}|a_{ij}|$ `// Pivotsuche`
  Vertausche a_{sk} und a_{jk} für alle $1 \leq k \leq n$ `// Pivotisierung`
  Vertausche p_s und p_j `// Aktualisiere Permutationsvektor`
  for $i = j+1, \dots, n$ **do**
    $a_{ij} \leftarrow \frac{a_{ij}}{a_{jj}}$ `// Matrix L ( Diagonalelemente sind immer 1)`
    for $k = j+1, \dots, n$ **do**
      $a_{ik} \leftarrow a_{ik} - a_{ij}a_{jk}$ `// Matrix R`

Output : $l_{ij} = a_{ij}$ $(i > j)$, $r_{ij} = a_{ij}$ $(i \leq j)$, p_j

Zur Lösung von $Ax = b$ geht man nun wie folgt vor:

1. Berechne LR-Zerlegung mit Pivotsuche: $PA = LR$
2. Löse $Ly = Pb$ mit Vorwärtssubstitution
3. Löse $Rx = y$ mit Rückwärtssubstitution

5.4 CHOLESKY-ZERLEGUNG

Für eine symmetrisch positiv definite Matrix A kann die LR-Zerlegung vereinfacht werden. Natürlich ließe sich L mit Hilfe der LR-Zerlegung berechnen, aber man kann durch Ausnutzen der Symmetrie ein günstigeres Verfahren finden. Ist $A = LR$ symmetrisch, so muss $R = L^T$ gelten. Die Beziehung $A = LL^T$, betrachtet für die einzelnen Matrixeinträge a_{ij}, liefert $\frac{1}{2}n(n+1)$ unabhängige Gleichungen für die $\frac{1}{2}n(n+1)$ Unbekannten l_{ij}, $i \geq j$. Durch eine geeignete Anordnung kann man diese Gleichungen durch einfaches Einsetzen auflösen. Dies sieht man am einfachsten für $n = 2$, d. h.

$$A = \begin{pmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{pmatrix}, \qquad L = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix}, \qquad L^T = \begin{pmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{pmatrix}.$$

Ausmultiplizieren von $A = LL^T$ liefert dann die Gleichungen

$$a_{11} = (l_{11})^2, \qquad a_{21} = l_{11}l_{21}, \qquad a_{22} = (l_{21})^2 + (l_{22})^2.$$

Da A symmetrisch positiv definit ist, sind die Diagonaleinträge strikt positiv. Wir können also durch spaltenweises Vorgehen sämtliche Unbekannten in stabiler Weise der Reihe nach berechnen; eine Pivotsuche ist nicht notwendig. Es bleibt zu zeigen, dass das Verfahren allgemein für symmetrisch positiv definite Matrizen durchführbar ist.

Satz 5.6. *Jede symmetrische positiv definite Matrix $A \in \mathbb{R}^{n\times n}$ hat eine eindeutige Zerlegung der Form $A = LL^T$, wobei $L \in \mathbb{R}^n$ eine reguläre untere Dreiecksmatrix mit positiven Diagonaleinträgen ist.*

Beweis. Wir konstruieren die gewünschte Zerlegung zeilen- und spaltenweise durch Induktion nach dem Zeilen- bzw. Spalten-Index k. Dafür schreiben wir $A_k := (a_{ij})_{i,j=1}^k$ und $L_k := (l_{ij})_{i,j=1}^k$ für den linken oberen $k \times k$-Block von A bzw. L.

Induktionsanfang $k = 1$: Da A positiv definit ist, gilt $a_{11} = e_1^T A e_1 > 0$ für den Einheitsvektor e_1, und wir erhalten die Aussage durch $l_{11} := \sqrt{a_{11}} > 0$.

Induktionsschritt $k - 1 \mapsto k$ für $k \geq 2$: Wir schreiben

$$A_k = \begin{pmatrix} A_{k-1} & a_k \\ a_k^T & a_{kk} \end{pmatrix}, \qquad L_k = \begin{pmatrix} L_{k-1} & 0 \\ l_k^T & l_{kk} \end{pmatrix}, \qquad L_k^T = \begin{pmatrix} L_{k-1}^T & l_k \\ 0 & l_{kk} \end{pmatrix}$$

für $a_k, l_k \in \mathbb{R}^n$ und $a_{kk}, l_{kk} \in \mathbb{R}$. Blockweise Betrachtung von $A_k = L_k L_k^T$ ergibt dann

$$A_{k-1} = L_{k-1}L_{k-1}^T, \qquad a_k = L_{k-1}l_k, \qquad a_k^T = l_k^T L_{k-1}^T, \qquad a_{kk} = l_k^T l_k + l_{kk}^2.$$

Die erste Gleichung gilt nach Induktionsvoraussetzung; die anderen müssen wir durch Wahl von l_k und l_{kk} erfüllen. Da ebenfalls nach Induktionsvoraussetzung L_{k-1} invertierbar ist, können wir die zweite und damit die dritte Gleichung eindeutig nach $l_k := L_{k-1}^{-1} a_k$ auflösen.

Es bleibt die letzte Gleichung, für die wir nur noch $a_{kk} - l_k^T l_k > 0$ garantieren müssen. Dafür verwenden wir wieder die positive Definitheit von A für einen geeigneten Vektor $x \in \mathbb{R}^n \setminus \{0\}$. Sei zunächst $x_k := -(L_{k-1}^T)^{-1} l_k \in \mathbb{R}^{k-1}$ (da mit L_{k-1} auch L_{k-1}^T invertierbar ist, und setze dann $x := (x_k, 1, 0, \ldots, 0)^T \in \mathbb{R}^n \setminus \{0\}$. Dann folgt aus der Konstruktion von l_k und x_k

$$\begin{aligned} 0 < x^T A x &= \begin{pmatrix} x_k \\ 1 \end{pmatrix}^T A_k \begin{pmatrix} x_k \\ 1 \end{pmatrix} = x_k^T A_{k-1} x_k + x_k^T a_k + a_k^T x_k + a_{kk} \\ &= ((L_{k-1}^T)^{-1} l_k)^T (L_{k-1} L_{k-1}^T)((L_{k-1}^T)^{-1} l_k) - ((L_{k-1}^T)^{-1} l_k)^T (L_{k-1} l_k) - (L_{k-1} l_k)^T (L_{k-1}^T l_k) + a_{kk} \\ &= l_k^T l_k - l_k^T l_k - l_k^T l_k + a_{kk} = a_{kk} - l_k^T l_k. \end{aligned}$$

Also können wir $l_{kk} := \sqrt{a_{kk} - l_k^T l_k} > 0$ setzen, womit L_k wieder vollen Rang hat und damit invertierbar ist. □

Umgekehrt folgt aus $A = LL^T$ mit L invertierbar, dass A symmetrisch ist und gilt

$$x^T A x = x^T L L^T x = (L^T x)^T (L^T x) = \|L^T x\|_2^2 > 0 \qquad \text{für alle } x \neq 0,$$

d. h. A ist positiv definit.

Dieser Beweis lässt sich konstruktiv umsetzen, wobei man spaltenweise $l_k = L_{k-1}^{-1} a_k$ durch Vorwärtssubstitution löst und dafür jeweils zunächst das nötige Diagonalelement l_{kk} berechnet. Man erhält dadurch die *Cholesky-Zerlegung*.[4]

Algorithmus 5.4 : Cholesky-Zerlegung

Input : a_{ij}

1 **for** $k = 1, \ldots, n$ **do**
2 $l_{kk} \leftarrow \sqrt{a_{kk} - \sum_{j=1}^{k-1} (l_{kj})^2}$
3 **for** $i = k+1, \ldots, n$ **do**
4 $l_{ik} \leftarrow \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj}\right) / l_{kk}$

Output : l_{ij}, $i \geq j$

Der Aufwand für die Cholesky-Zerlegung beträgt zwar auch $O(n^3)$, bei genauerer Betrachtung sind aber nur halb so viele Operationen notwendig wie für eine LR-Zerlegung. Außerdem ist keine Pivotsuche notwendig.

[4] Von André-Louis Cholesky um 1905 für lineare Ausgleichsprobleme bei der Land-Neuvermessung Frankreichs entwickelt; die Methode wurde erst 1924, nach seinem Tod im ersten Weltkrieg, publiziert.

6 LINEARE AUSGLEICHSRECHNUNG

Die bislang betrachteten Verfahren setzen voraus, dass die Matrix A quadratisch ist. In zahlreichen Anwendungen ist dies jedoch nicht der Fall, so dass wir in diesem Kapitel Methoden zur Behandlung von linearen Gleichungssystemen mit nichtquadratischen Matrizen $A \in \mathbb{R}^{m\times n}$ untersuchen wollen.

Wir betrachten hier nur den Fall $m > n$. Das Gleichungssystem $Ax = b$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ ist dann überbestimmt, so dass wir in der Regel keine Lösung erwarten dürfen. Wir müssen also unseren Begriff von "Lösung" erweitern: Statt exakter Gleichheit erwarten wir nur noch, dass Ax "so nahe wie möglich" bei b liegt. Dies führt auf ein Minimierungsproblem: Wir nennen $x^* \in \mathbb{R}^n$ *Lösung des linearen Ausgleichsproblems*, falls gilt

$$\|Ax^* - b\|_2^2 = \min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2.$$

Dass hierbei die Euklidische Norm $\|\cdot\|_2$ verwendet wird, ist fundamental, genau wie die Minimierung des Quadrats der Norm. Aus diesem Grunde wird dieser Ansatz auch "Methode der kleinsten (Fehler-)Quadrate", engl. "least squares", genannt.

Aus der Fülle der Anwendungen in Mathematik und Praxis seien hier zwei genannt.

Beispiel 6.1. (i) In der Anwendung müssen häufig Größen bestimmt werden, die nicht direkt messbar sind. Gesucht sind etwa die Werte $x_1, \ldots, x_n$ (z. B. Temperatur in gleichmäßig verteilten Punkten eines Stabes, der zum Teil in einer Wand versenkt ist), verfügbar ist jedoch nur ein Messwert φ (z. B. die Temperatur am frei liegenden Stabende), der abhängig ist von den Unbekannten x_i sowie einem kontrollierbaren Parameter t (z. B. die Umgebungstemperatur). Letzterer erlaubt es, mehrere unabhängige Messungen φ_j, $1 \le j \le m$, durchzuführen. Ist nun die Abhängigkeit $\varphi_j = \varphi(t_j; x_1, \ldots, x_m)$ bekannt, so kann man hoffen, eine Schätzung der x_i zu bekommen, indem man diejenigen x_i^* bestimmt, für die

$$\sum_{j=1}^{m} |\varphi(t_j; x_1, \ldots, x_n) - \varphi_j|^2$$

minimal ist. Im einfachsten Fall einer linearen Abhängigkeit

$$\varphi(t_j; x_1, \ldots, x_2) = \sum_{i=1}^{n} a_i(t_j) x_i = \sum_{i=1}^{n} a_{ij} x_i$$

mit bekannten Koeffizienten a_{ij} führt dies auf das lineare Ausgleichsproblem $\|Ax - \varphi\|_2^2 \to \min$.

(ii) Eine weiteres Beispiel ist die *Funktionsapproximation*, deren einfachster Spezialfall die Suche nach der *Regressionsgerade* ist. Hierbei handelt es sich um das Problem, zu vorgegebenen Werten $x_1, \dots, x_m$ und $y_1, \dots, y_m$ eine Gerade $y = ax + b$ so zu finden, dass die Punkte (x_i, y_i) möglichst nahe bei ihr liegen. Mit der Schreibweise $\varphi(x_j; a, b) = ax_j + b$, $x = (x_1, \dots, x_m)^T$, $y = (y_1, \dots, y_m)^T$, $\mathbb{1} = (1, \dots, 1)^T \in \mathbb{R}^m$ führt dies auf das lineare Ausgleichsproblem

$$\left\| (x, \mathbb{1}) \begin{pmatrix} a \\ b \end{pmatrix} - y \right\|_2^2 = \sum_{j=1}^{n} |\varphi(x_j; a, b) - y_j|^2 \to \min .$$

6.1 DIE NORMALENGLEICHUNGEN

Die Lösung des Ausgleichsproblems besteht also im Minimieren der quadratischen Funktion $f(x) = \|Ax - b\|_2^2$. Die notwendige Optimalitätsbedingung dafür ist

$$\begin{aligned} 0 = \nabla f(x) &= \nabla \left[\langle Ax - b, Ax - b \rangle \right] = \nabla \left[\langle Ax, Ax \rangle - 2\langle Ax, b \rangle - \langle b, b \rangle \right] \\ &= 2A^T Ax - 2A^T b. \end{aligned}$$

Jeder Minimierer x^* von $f(x)$ erfüllt also die *Normalengleichungen*

$$A^T Ax = A^T b. \tag{6.1}$$

Daraus folgt für den Minimierer x^* und $x \in \mathbb{R}^n$ beliebig insbesondere

$$0 = x^T (A^T b - A^T Ax^*) = (Ax)^T (b - Ax^*);$$

geometrisch bedeutet dies also, dass das Residuum $b - Ax^*$ senkrecht steht auf dem Bild von A (siehe Abb. 6.1). Für die Untersuchung der Lösbarkeit der Normalengleichungen benötigen wir die folgenden allgemeinen Eigenschaften für das Bild $\mathcal{R}(A^T A)$ und den Kern $\mathcal{N}(A^T A)$.

Lemma 6.2. *Sei $A \in \mathbb{R}^{m \times n}$ mit $m, n \in \mathbb{N}$ beliebig. Dann gilt*

(i) $A^T A$ *ist positiv semidefinit, und positiv definit genau wenn A injektiv ist;*

(ii) $\mathcal{N}(A^T A) = \mathcal{N}(A)$;

(iii) $\mathcal{R}(A^T) = \mathcal{N}(A)^\perp$ *(das orthogonale Komplement);*

(iv) $\mathcal{R}(A^T A) = \mathcal{R}(A^T)$.

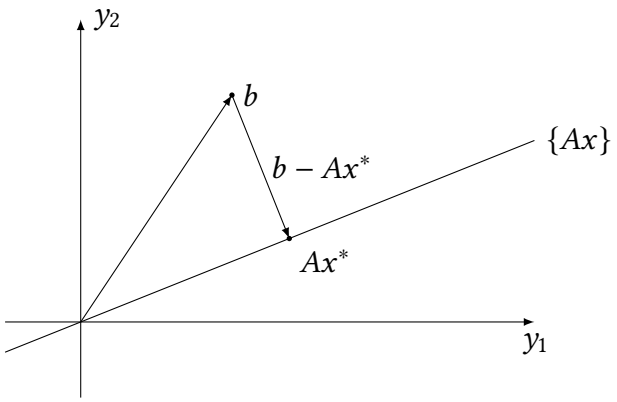


Abbildung 6.1: Die Minimierung von $\|Ax - b\|_2^2$ ist äquivalent dazu, dass $b - Ax^*$ senkrecht auf dem Bild von A (der Menge $\{Ax : x \in \mathbb{R}^n\}$) steht. (Hier: $m = 2$, $n = 1$.)

Beweis. Zu (i): Dies folgt direkt aus $x^T A^T Ax = (Ax)^T (Ax) = \|Ax\|_2^2 \geq 0$ für alle $x \in \mathbb{R}^n$ mit Gleichheit genau für $x \in \mathcal{N}(A)$.

Zu (ii): Aus $Ax = 0$ folgt sofort $A^T(Ax) = 0$ und damit $\mathcal{N}(A) \subset \mathcal{N}(A^T A)$. Umgekehrt folgt aus $A^T Ax = 0$ auch $\|Ax\|^2 = x^T A^T Ax = 0$ und damit $\mathcal{N}(A^T A) \subset \mathcal{N}(A)$.

Zu (iii): Sei $x = A^T y \in \mathcal{R}(A^T)$. Dann gilt für alle $z \in \mathcal{N}(A)$

$$z^T x = z^T (A^T y) = (Az)^T y = 0$$

und damit $x \in \mathcal{N}(A)^\perp$. Aus dem Dimensionssatz der linearen Algebra folgt aber auch

$$\dim(\mathcal{N}(A)^\perp) = n - \dim(\mathcal{N}(A)) = \dim(\mathcal{R}(A)) = \text{Rang}(A) = \text{Rang}(A^T) = \dim(\mathcal{R}(A^T))$$

und damit $\mathcal{R}(A^T) = \mathcal{N}(A)^\perp$.

Zu (iv): Aus (ii) und (iii) folgt wegen der Symmetrie von $A^T A$ sofort

$$\mathcal{R}(A^T) = \mathcal{N}(A)^\perp = \mathcal{N}(A^T A)^\perp = \mathcal{R}(A^T A). \qquad \square$$

Wir erhalten daraus das folgende Resultat über die Lösbarkeit des Ausgleichsproblems.

Satz 6.3. *Seien $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ mit $m, n \in \mathbb{N}$ beliebig. Dann gilt*

(i) Die Normalengleichungen $A^T Ax^* = A^T b$ haben eine Lösung $\bar{x} \in \mathbb{R}^n$, und die Menge aller Lösungen ist gegeben durch $\bar{x} + \mathcal{N}(A) := \{\bar{x} + x_0 \ : \ x_0 \in \mathcal{N}(A)\}$.

(ii) x^* ist Minimum von $\|Ax - b\|_2^2$ genau dann, wenn x^* die Normalengleichungen (6.1) löst.

Beweis. Zu (i): Aus Lemma 6.2 (iv) folgt nach Definition für $A^T b \in \mathcal{R}(A^T) = \mathcal{R}(A^T A)$ die Existenz eines $\bar{x} \in \mathbb{R}^n$ mit $A^T A\bar{x} = A^T b$, und alle Lösungen dieser Gleichung erhalten wir durch Addition eines Elements im Kern $\mathcal{N}(A^T A) = \mathcal{N}(A)$.

Zu (ii): Betrachte $\psi(x) := \|Ax - b\|_2^2$. Für $\bar{x}$ aus (i) und $x \in \mathbb{R}^n$ beliebig gilt dann

$$\begin{aligned}\psi(x) - \psi(\bar{x}) &= (Ax - b)^T(Ax - b) - (A\bar{x} - b)^T(A\bar{x} - b)\\ &= x^T A^T Ax - \bar{x}^T A^T A\bar{x} - 2x^T A^T b + 2\bar{x}^T A^T b\\ &= x^T A^T Ax - \bar{x}^T A^T A\bar{x} - 2x^T(A^T A\bar{x}) + 2\bar{x}^T(A^T A\bar{x})\\ &= (Ax)^T Ax + (A\bar{x})^T A\bar{x} - 2(Ax)^T(A\bar{x})\\ &= \|Ax - A\bar{x}\|_2^2 = \|A(x - \bar{x})\|_2^2 \geq 0\end{aligned}$$

mit Gleichheit genau für $x - \bar{x} \in \mathcal{N}(A)$. Also ist x Minimierer von ψ genau dann, wenn gilt $x \in \bar{x} + \mathcal{N}(A)$, d. h. x ist Lösung der Normalengleichung. □

Wir betrachten jetzt die Kondition des linearen Ausgleichsproblems. Dazu müssen wir uns zuerst darauf einigen, was mit der Konditionszahl einer nichtquadratischen Matrix gemeint sein soll: Diese war ja über die Norm der inversen Matrix definiert, welche nur für quadratische Matrizen existiert. Wir haben allerdings in Lemma 4.15 eine äquivalente Darstellung kennengelernt, die auch für nichtquadratische Matrizen einen Sinn ergibt. Für eine Matrix $A \in \mathbb{R}^{m\times n}$ definieren wir die Konditionszahl deshalb als

$$\kappa_2(A) = \frac{\max\|Ax\|_2}{\min\|Ax\|_2},$$

wobei das Maximum und Minimum über alle $x \in \mathbb{R}^n$ mit $\|x\|_2 = 1$ genommen wird.

Damit lässt sich das folgende Resultat über die Kondition des linearen Ausgleichproblems beweisen.

Satz 6.4. *Sei $A \in \mathbb{R}^{m\times n}$, $b, \tilde{b} \in \mathbb{R}^m$. Für das Minimum x^* von $\|Ax - b\|_2^2$ und das Minimum $\tilde{x}$ von $\|Ax - \tilde{b}\|_2^2$ gilt:*

$$\frac{\|x^* - \tilde{x}\|_2}{\|x^*\|_2} \leq \kappa_2(A)\frac{\|b\|_2}{\|Ax^*\|_2}\frac{\|\tilde{b} - b\|_2}{\|b\|_2}$$

Im Unterschied zur Lösung eines linearen Gleichungssystem wird für das lineare Ausgleichsproblem im Allgemeinen nicht $\|Ax^*\| = \|b\|$ gelten. Die Kondition $\kappa_2(A)\frac{\|b\|_2}{\|Ax^*\|_2}$ hängt also wesentlich von der Größe des Residuums $b - Ax^*$ ab: Ist dieses klein (gilt also $\|Ax^*\|_2 \approx \|b\|_2$), so verhält sich das Ausgleichsproblem wie ein lineares Gleichungssystem. Ist aber das Residuum groß, so ist $\|Ax^*\|_2$ klein obwohl $\|b\|_2$ dies nicht ist, und das Problem wird schlechter konditioniert.

Aus Lemma 6.2 folgt, dass für eine Matrix A mit vollem Spaltenrang die Matrix $A^T A$ invertierbar und sogar symmetrisch und positiv definit ist, so dass die Normalengleichungen

stabil mit dem Cholesky- oder CG-Verfahren gelöst werden können. Dieses Vorgehen hat für größere Matrizen allerdings zwei entscheidende Nachteile:

(i) Um die Cholesky-Zerlegung auf die Matrix A^TA anzuwenden, muss diese erst durch eine Matrix-Matrix-Multiplikation berechnet werden. Das ist jedoch eine aufwändige Operation (Komplexität $O(mn^2)$).

(ii) Es gilt $\kappa_2(A^TA) = \kappa_2(A)^2$; die Normalengleichungen werden also eine deutlich schlechtere Kondition haben, was sich negativ auf die Konvergenzgeschwindigkeit des CG-Verfahrens auswirkt.

6.2 QR-ZERLEGUNG

Eine für das Ausgleichsproblem vorteilhaftere Zerlegung erhalten wir, indem wir fordern, dass einer der beiden Matrixfaktoren orthogonal ist.

Definition 6.5. Eine Matrix $Q \in \mathbb{R}^{n\times n}$ heißt *orthogonal*, falls gilt $Q^T = Q^{-1}$.

Lemma 6.6. *Sei $Q \in \mathbb{R}^{n\times n}$ orthogonal. Dann gilt:*

(i) $\|Qx\|_2 = \|x\|_2$;

(ii) $\kappa_2(Q) = 1$;

(iii) ist $\tilde{Q} \in \mathbb{R}^{n\times n}$ orthogonal, so auch $\tilde{Q}Q$.

Wir suchen also für $A \in \mathbb{R}^{m\times n}$ mit $m > n$ eine Zerlegung $A = QR$, wobei $Q \in \mathbb{R}^{m\times m}$ eine orthogonale Matrix und $R \in \mathbb{R}^{m\times n}$ eine obere Dreiecksmatrix ist. Hat A vollen Spaltenrang (was wir in Folge annehmen wollen), so hat R die Form

$$R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \tag{6.2}$$

wobei $R_1 \in \mathbb{R}^{n\times n}$ eine reguläre obere Dreiecksmatrixist und 0 für einen $(m-n)\times n$-Block mit Null-Einträgen steht. Nehmen wir an, wir haben solch eine Zerlegung gefunden. Partitionieren wir auch die rechte Seite als

$$Q^Tb = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \tag{6.3}$$

mit $c_1 \in \mathbb{R}^n$ und $c_2 \in \mathbb{R}^{m-n}$, dann erhalten wir

$$\begin{aligned} \|Ax-b\|_2^2 = \|QRx-b\|_2^2 = \|Q(Rx-Q^Tb)\|_2^2 = \|Rx-Q^Tb\|_2^2 \\ = \|R_1x-c_1\|_2^2 + \|c_2\|_2^2 \end{aligned} \tag{6.4}$$

wegen der Orthogonalität von Q und Lemma 6.6 (i). Wenn A vollen Spaltenrang hat, dann ist auch der Rang von R_1 gleich n; die Gleichung $R_1 x = c_1$ hat also eine eindeutige Lösung. Da der zweite Term auf der rechten Seite von (6.4) nicht von x abhängt, ist diese Lösung auch das eindeutige Minimum des Ausgleichsproblems. Wir halten fest:

Satz 6.7. *Sei $A \in \mathbb{R}^{m\times n}$ mit $Rang(A) = n$, $Q \in \mathbb{R}^{m\times m}$ orthogonal und $R \in \mathbb{R}^{m\times n}$ eine obere Dreiecksmatrix, mit $A = QR$. Dann ist das Minimum x^* von $\|Ax - b\|_2^2$ die Lösung von $R_1 x = c_1$, und für das Residuum gilt $\|Ax^* - b\|_2^2 = \|c_2\|_2^2$ (mit R_1 aus (6.2) und c_1, c_2 aus (6.3)).*

Die gewünschte Zerlegung lässt sich analog zu der LR-Zerlegung konstruieren: Wir bringen die Matrix A stufenweise durch Elementarumformungen auf Dreiecksform, wobei wir allerdings nur solche Umformungen zulassen, die durch Multiplikation mit orthogonalen Matrizen repräsentiert werden können. Solche Transformationen sind längenerhaltend; geometrisch entsprechen sie zum Beispiel Rotationen um einen vorgegebenen Winkel oder Spiegelungen an einer vorgegebenen Hyperebene. Der erste Ansatz führt auf die *Givens-Rotationen*, der zweite auf die *Householder-Reflektionen*; wir beschränken uns hier auf Letztere.[1]

Wir sparen uns eine ausführliche geometrische Herleitung und verraten gleich, dass in $\mathbb{R}^n$ die Spiegelung an einer Hyperebene senkrecht zu einem gegebenen Vektor $u \in \mathbb{R}^n \setminus \{0\}$ realisiert wird durch die *Householder-Reflektion*[2]

$$H_u := I - 2\frac{uu^T}{u^T u}.$$

Man vergewissert sich leicht, dass H_u symmetrisch ist; weiter gilt

$$H_u H_u = I - 4\frac{uu^T}{u^T u} + 4\frac{1}{(u^T u)^2}(uu^T)(uu^T) = I$$

und damit ist H_u orthogonal.

Ziel ist nun, eine gegebene Matrix $A \in \mathbb{R}^{m\times n}$ analog zur Gauss-Elimination bzw. LR-Zerlegung spaltenweise auf obere Dreiecksform zu bringen, in dem wir jeweils eine geeignete Householder-Reflektion anwenden. Bezeichnen wir mit $a_1 = (a_{11}, \ldots, a_{m1})^T \in \mathbb{R}^m$ die erste Spalte von A, müssen wir also ein $u \in \mathbb{R}^m \setminus \{0\}$ finden so dass gilt

$$H_u a_1 = a_1 - 2\frac{u^T a_1}{u^T u}u = \rho e_1 \tag{6.5}$$

für ein $\rho \in \mathbb{R}$. Da H_u unitär ist, muss wegen $\|e_1\|_2 = 1$ gelten

$$|\rho| = \|\rho e_1\|_2 = \|H_u a_1\|_2 = \|a_1\|_2 > 0$$

[1] Givens-Rotationen werden bei der Berechnung von Eigenwerten in der Numerischen Mathematik 2 eine Rolle spielen.

[2] Für Fans: $\frac{u^T x}{u^T u}u$ ist genau die orthogonale Projektion von x senkrecht zu u, die wir zweimal von x abziehen müssen, um die Spiegelung an der Hyperebene zu erhalten.

falls $a_1 \neq 0$ (ansonsten wäre $\mathrm{Rang}(A) < n$, im Widerspruch zur Annahme). Weiter folgt aus (6.5) $a_1 - \rho e_1 = 2\frac{u^T a_1}{u^T u} u$, d. h. u ist ein skalares Vielfaches von $a_1 - \rho e_1$. Es bleibt nur noch, das Vorzeichen von ρ und die Skalierung von u zu wählen. Üblich ist dabei eine Wahl, die Subtraktionen und damit Auslöschung vermeidet:

$$\rho = -\mathrm{sign}(a_{11})\|a_1\|_2, \qquad u = (\mathrm{sign}(a_{11})(|a_{11}| - \|a_1\|_2), a_{21}, \ldots, a_{m1})^T,$$

mit $\mathrm{sign}(t) = 1$ für $t \geq 0$ und -1 sonst. Gilt bereits $a_1 = \rho e_1$, so ist $u = 0$ und damit $H_u = I$. Andernfalls ist

$$\begin{aligned} u^T u &= (a_1 - \rho e_1)^T (a_1 - \rho e_1) = \|a_1\|_2^2 - 2\rho a_1^T e_1 + |\rho|^2 \\ &= 2|\rho|^2 - 2\rho a_{11} = 2|\rho|(|\rho| + |a_{11}|) \end{aligned}$$

und daher

$$H_u = I - ww^T \qquad \text{für} \qquad w := \frac{a_1 - \rho e_1}{\sqrt{|\rho|(|\rho| + |a_{11}|)}}.$$

Da Matrixmultiplikationen spaltenweise wirken, haben wir also eine orthogonale Matrix $H_1 := H_u$ gefunden, so dass gilt

$$H_1 A = [H_1 a_1 | H_1 a_2 | \ldots | H_n a_n] = [\rho e_1 | \tilde{a}_2 | \ldots | \tilde{a}_n],$$

wobei $a_2, \ldots, a_n$ die restlichen Spalten von A bezeichnen, die sich durch Multiplikation mit H_1 ändern.

Wir gehen nun weiter vor durch Induktion über die Spalten $k = 1, \ldots, n$. Den Induktionsanfang $k = 1$ haben wir soeben gemacht; für den Induktionsschritt nehmen wir an, dass wir bereits orthogonale $H_1, \ldots, H_k$ gefunden haben mit

$$A_k := H_k \cdots H_1 A = \begin{pmatrix} R_k & B_k \\ 0 & C_k \end{pmatrix}$$

für $R_k \in \mathbb{R}^{k \times k}$ in oberer Dreiecksform und $B_k \in \mathbb{R}^{k \times (n-k)}, C_k \in \mathbb{R}^{(m-k) \times (n-k)}$ beliebig. Wir betrachten nun $a_k := (c_{11}, \ldots, c_{m-k,1})^T \in \mathbb{R}^{m-k}$. Ist $c_{11} = 0$, so sind die ersten $k+1$ Spalten von A_k linear abhängig und damit $\mathrm{Rang}(A) = \mathrm{Rang}(A_k) < n$ wegen der Orthogonalität von $H_k \cdots H_1$, im Widerspruch zur Annahme. Wir können also wie oben ein $\tilde{u} := a_k + \mathrm{sign}(c_{11})\|a_k\|_2 e_1 \in \mathbb{R}^{m-k}$ bestimmen mit $H_{\tilde{u}} a_k = -\mathrm{sign}(c_{11})\|a_k\|_2 e_1$. Wir setzen nun

$$H_{k+1} := \begin{pmatrix} I & 0 \\ 0 & H_{\tilde{u}} \end{pmatrix} = H_{u_k} \quad \text{für} \quad u_k = (0, \tilde{u}^T)^T \in \mathbb{R}^m$$

so dass gilt

$$A_{k+1} := H_{k+1} \cdots H_1 A = H_{k+1} A_k = \begin{pmatrix} R_k & B_k \\ 0 & H_{\tilde{u}} C_k \end{pmatrix} = \begin{pmatrix} R_{k+1} & B_{k+1} \\ 0 & C_{k+1} \end{pmatrix}$$

da die erste Spalte von C_k auf obere Dreiecksform gebracht wurde und H_{k+1} die ersten k Zeilen unverändert lässt.

Nach $k = n$ Schritten erhalten wir also

$$A_n = R_n = H_n \cdots H_1 A$$

in oberer Dreiecksform, und $Q := (H_n \cdots H_1)^{-1} = H_1 \cdots H_n$ ist als Produkt orthogonaler symmetrischer Matrizen ebenfalls orthogonal.

Dabei ist es in der Regel nicht notwendig, die Matrizen $H_1, \ldots, H_k$ aufzustellen und zu speichern (wenn man nicht ausnahmsweise an der vollen Matrix Q selber interessiert ist); es genügt, die entsprechenden Vektoren $u_1, \ldots, u_k$ zu speichern. Insbesondere ist

$$H_k A_k = \left(I - 2\frac{u_k u_k^T}{u_k^T u_k} \right) A_k = A_k - \frac{2}{u_k^T u_k} u_k u_k^T A_k = A_k - \frac{2}{u_k^T u_k} u_k (A^T u_k)^T,$$

so dass nur eine Matrix-Vektor- statt einer Matrix-Matrix-Multiplikation notwendig ist. Ist man nur an der Lösung eines konkreten Ausgleichsproblems interessiert, müssen auch die Vektoren nicht gespeichert werden, wenn man sofort $H_k b_k = b_k - 2\frac{u_k^T b_k}{u_k^T u_k} u_k$ berechnet. Sowohl Matrix als auch rechte Seite können dabei in jedem Schritt überschrieben werden.

Algorithmus 6.1 : Ausgleichsrechnung mit QR-Zerlegung

Input : A, b

1 **for** $k = 1$ *to* n **do**

2 $\quad a_k \leftarrow (a_{kk}, \cdots, a_{m,k})^T$

3 $\quad u_k \leftarrow (0, a_k + \operatorname{sign}(a_{kk}) \|a_k\|_2 e_1)^T$

4 $\quad z_k \leftarrow A^T u_k$

5 $\quad A \leftarrow A - \frac{2}{u_k^T u_k} u_k z_k^T$

6 $\quad b \leftarrow b - 2\frac{u_k^T b}{u_k^T u_k}$

7 Setze $A' = (a_{ij})_{i,j=1}^n$, $b' = (b_i)_{i=1}^n$

8 Löse $A'x = b'$ durch Rückwärtssubstitution

Output : $x, u_1, \ldots, u_n$

WEITERFÜHRENDE LITERATUR

G. H. Golub & C. F. Van Loan (2013), *Matrix Computations*, 4. Aufl., Johns Hopkins University Press, Baltimore, MD.

Teil III

NUMERISCHE ANALYSIS

7 POLYNOM-INTERPOLATION

Die Interpolation ist ein grundlegender Bestandteil einer konstruktiven Theorie der Funktionen: Zu einer diskreten Menge gegebener Werte soll eine Funktion konstruiert werden, die diese Werte annimmt. Präziser stellen wir folgende *Interpolationsaufgabe*: Gegeben seien die

Stützstellen $x_0, \ldots, x_n \in [a, b] \subset \mathbb{R}$ und

Funktionswerte $f(x_0), \ldots, f(x_n) \in \mathbb{R}$;

gesucht ist eine (einfache) Funktion $g : \mathbb{R} \to \mathbb{R}$, so dass gilt

$$g(x_i) = f(x_i) \quad \text{für alle } 0 \leq i \leq n.$$

Ursprünglich entstammt diese Fragestellung der *Tabelleninterpolation*; vor der Verbreitung von Rechenmaschinen waren spezielle Funktionen wie Kosinus, Logarithmus oder die Exponentialfunktion in lange Listen von Argumenten und Funktionswerten niedergeschrieben. War man etwa am Wert von $\log 0.456$ interessiert, so suchte man zum Beispiel den Eintrag für $\log 0.45$ und $\log 0.46$, und interpolierte linear:

$$\log 0.456 \approx \log 0.45 + \frac{\log 0.46 - \log 0.45}{0.46 - 0.45}(0.456 - 0.45) = -0.7925$$

(der exakte Wert ist etwa -0.785262). Eine höhere Genauigkeit ist natürlich zu erwarten, wenn statt einer linearen Funktion eine Funktion verwendet wird, die dem Verlauf der Logarithmusfunktion besser folgt. Heutzutage[1] wird die Interpolation hauptsächlich als Grundlage für Verfahren der numerischen Analysis verwendet: Um zum Beispiel eine Funktion näherungsweise zu integrieren oder zu differenzieren, bestimmt man für sie eine interpolierende Funktion, welche exakt integriert (oder differenziert) werden kann.

Als "einfache" Interpolationsfunktionen betrachten wir in diesem Kapitel Polynome, genauer gesagt, Funktionen aus dem Vektorraum

$$P_n := \left\{ \sum_{j=0}^{n} a_j x^j \ : \ a_j \in \mathbb{R} \right\}$$

[1] Tatsächlich wird dieses Prinzip unter dem Namen *lookup table* immer noch in integrierten Schaltungen verwendet, wenn eine sehr große Zahl von Berechnungen pro Sekunde erforderlich ist. Insbesondere moderne Grafikprozessoren können lineare Interpolation und Tabellenzugriffe mit wesentlich höherer Geschwindigkeit ausführen, als sie z. B. die ersten Glieder der Sinusreihe aufsummieren können.

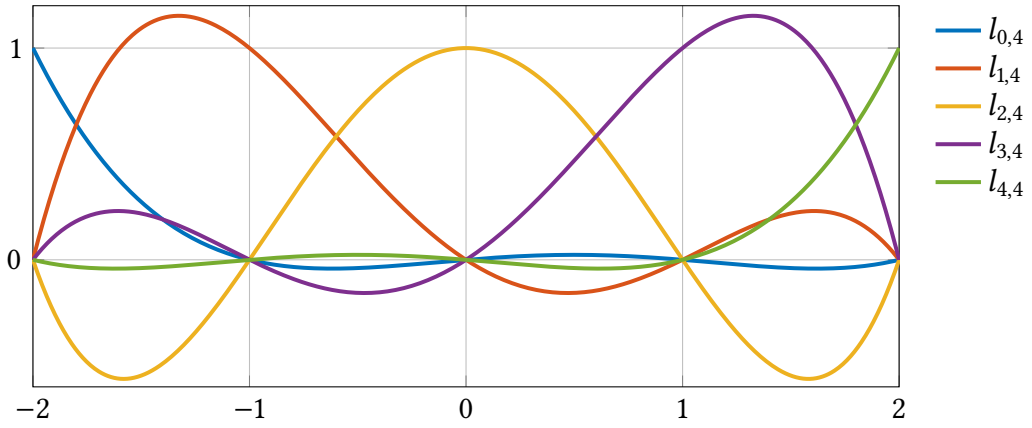


Abbildung 7.1: Die Lagrange-Polynome ℓ_{j4} zu den Stützstellen $\{-2, -1, 0, 1, 2\}$.

der *Polynome vom Höchstgrad n*. (Alternativen, die wir in den folgenden Kapiteln betrachten werden, sind trigonometrische Funktionen oder stückweise Polynome, genannt *Splines*.) Aus der Definition ist sofort ersichtlich, dass die *Monome* $1, x, x^2, \ldots, x^n$ ein Erzeugendensystem von P_n darstellen. Aus dem Hauptsatz der Algebra folgt auch, dass $\sum_{j=0}^{n} a_j x^j = 0$ für alle $x \in \mathbb{R}$ nur für das Nullpolynom gilt, d. h. $a_j = 0$ für alle $j = 0, \ldots n$ gelten muss. Also bilden die Monome tatsächlich eine Basis, und P_n hat die Dimension $n + 1$.

7.1 LAGRANGE-INTERPOLATIONSFORMEL

Wir suchen also zu den gegebenen Stützstellen $x_0, \ldots, x_n \in \mathbb{R}$ mit zugehörigen Funktionswerten $f(x_0), \ldots, f(x_n)$ ein Polynom $p_n \in P_n$, so dass $p_n(x_i) = f(x_i)$ für alle $0 \le i \le n$ gilt. Es ist einleuchtend, dass die $n + 1$ Interpolationsbedingungen an paarweise verschiedenen (was wir in Folge annehmen wollen) Stützstellen genügen, um die $n + 1$ Koeffizienten eines Polynoms vom Grad n eindeutig festzulegen. Durch eine geschickte Wahl der Basis von P_n kann man diese Koeffizienten explizit berechnen, ohne ein Gleichungssystem lösen zu müssen.

Definition 7.1. Für $0 \le j \le n$ sind die *Lagrange-Polynome*[2] definiert durch

$$\ell_{jn}(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}, \qquad x \in \mathbb{R}. \tag{7.1}$$

[2] Joseph-Louis Lagrange hat in der zweiten Hälfte des 18. Jahrhunderts zahlreiche Beiträge zur Mechanik, Wahrscheinlichkeitstheorie, Algebra und Zahlentheorie geleistet. Auf das Interpolationsproblem führten ihn seine Arbeiten zur Umformung unendlicher Reihen.

Man sieht sofort, dass die ℓ_{jn} Polynome vom Höchstgrad n sind. Durch Einsetzen erkennt man auch, dass gilt

$$\ell_{jn}(x_i) = \begin{cases} 1 & \text{für } j = i, \\ 0 & \text{für } j \neq i, \end{cases}$$

(siehe Abb. 7.1). Aus dieser Eigenschaft folgt sofort, dass die ℓ_{jn} linear unabhängig sind und daher eine Basis von P_n (genannt *nodale Basis*) bilden. Eine Lösung der Interpolationsaufgabe ist daher gegeben durch

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

Diese Lösung ist sogar eindeutig: Seien $p, q \in P_n$ zwei Polynome mit $p(x_i) = f(x_i) = q(x_i)$ für alle $0 \leq i \leq n$. Dann hat $p - q \in P_n$ die $n+1$ Nullstellen $x_0, \ldots, x_n$. Nach dem Hauptsatz der Algebra kann aber ein von Null verschiedenes Polynom vom Höchstgrad n höchstens n reelle Nullstellen haben; das Polynom $p - q$ muss also identisch Null sein, das heißt es gilt $p = q$. Wir halten fest:

Satz 7.2. *Für paarweise verschiedene $x_0, \ldots, x_n \in \mathbb{R}$ und $f(x_0), \ldots, f(x_n) \in \mathbb{R}$ existiert genau ein Polynom $p_n \in P_n$, so dass $p_n(x_i) = f(x_i)$ für alle $0 \leq i \leq n$ gilt.*

Die Lagrange-Darstellung hat jedoch wesentliche Nachteile: Einerseits besteht bei nahe beieinander liegenden Stützstellen die Gefahr der Auslöschung bei der Berechnung von (7.1). Andererseits müssen alle Lagrange-Polynome neu berechnet werden, falls sich eine Stützstelle ändert oder eine neue hinzukommt. Wir werden deshalb Algorithmen diskutieren, die – je nach Aufgabenstellung – besser geeignet sind.

7.2 POLYNOMAUSWERTUNG NACH AITKEN–NEVILLE

Ist man nur an (wenigen) Werten des Interpolationspolynoms interessiert (etwa bei der Tabelleninterpolation), so benötigt man die explizite Darstellung des Polynoms gar nicht. Der Algorithmus von Aitken und Neville wertet p_n rekursiv aus durch lineare Interpolation von Interpolationspolynomen niedrigeren Grades.

Um das Verfahren einfach darstellen zu können, definieren wir $p_{i,k}$ als Interpolationspolynom vom Höchstgrad k mit den Stützstellen $x_{i-k}, \ldots, x_i$. Das folgende Lemma stellt dann den Rekursionsschritt dar.

Lemma 7.3 (Aitken[3]). *Für alle $n \in \mathbb{N}$ und $x \in \mathbb{R}$ gilt*

$$p_n(x) = p_{n,n}(x) = \frac{x - x_0}{x_n - x_0} p_{n,n-1}(x) + \frac{x_n - x}{x_n - x_0} p_{n-1,n-1}(x). \tag{7.2}$$

[3] Alec Aitken (1895–1967) war einer der bedeutendsten Mathematiker Neuseelands. Er hatte ein phänomenales Gedächtnis und war ein Meister im Kopfrechnen.

Beweis. Wir überprüfen die Interpolationsbedingungen:

(i) Für $0 < i < n$ gilt

$$\frac{x_i - x_0}{x_n - x_0} p_{n,n-1}(x_i) + \frac{x_n - x_i}{x_n - x_0} p_{n-1,n-1}(x_i) = \frac{x_i - x_0}{x_n - x_0} f(x_i) + \frac{x_n - x_i}{x_n - x_0} f(x_i) = f(x_i),$$

da x_i Stützstelle sowohl für das Interpolationspolynom $p_{n,n-1}$ als auch für $p_{n-1,n-1}$ ist.

(ii) Für $i = 0$ gilt

$$\frac{x_0 - x_0}{x_n - x_0} p_{n,n-1}(x_0) + \frac{x_n - x_0}{x_n - x_0} p_{n-1,n-1}(x_0) = p_{n-1,n-1}(x_0) = f(x_0),$$

da x_0 Stützstelle für $p_{n-1,n-1}$ ist.

(iii) Für $i = n$ gilt analog

$$\frac{x_n - x_0}{x_n - x_0} p_{n,n-1}(x_n) + \frac{x_n - x_n}{x_n - x_0} p_{n-1,n-1}(x_n) = p_{n,n-1}(x_n) = f(x_n).$$

Nun sind $p_{n,n-1}$ und $p_{n-1,n-1}$ nach Definition Interpolationspolynome vom Höchstgrad $n-1$. Die rechte Seite von (7.2) ist also auch ein Interpolationspolynom vom Höchstgrad n. Aus der Eindeutigkeit der Interpolationsaufgabe folgt daher, dass dieses identisch sein muss mit dem Interpolationspolynom p_n. □

Da es sich bei $p_{n,n-1}$ und $p_{n-1,n-1}$ ebenfalls um Interpolationspolynome (nun vom Höchstgrad $n-1$) handelt, lassen sich diese ebenfalls mit Lemma 7.3 (mit entsprechend angepassten Stützstellen) berechnen; es gilt

$$\begin{cases} p_{i,k}(x) = \dfrac{x - x_{i-k}}{x_i - x_{i-k}} p_{i,k-1}(x) + \dfrac{x_i - x}{x_i - x_{i-k}} p_{i-1,k-1}(x), & k = 1, \dots, n, \quad i = k, \dots, n, \\ p_{i,0}(x) = f(x_i), & i = 0, \dots, n. \end{cases} \tag{7.3}$$

Dies führt auf das rekursive Schema von Neville[4] zur Auswertung von p_n an der (festen) Stelle x: Man berechnet jeweils für $k = 0, \dots, n$ und $i = k, \dots, n$ alle $p_{i,k}(x)$. Dies wird in folgendem Tableau verdeutlicht, in dem spaltenweise von oben nach unten jeder Wert durch Interpolation der beiden Werte direkt links und links oberhalb berechnet wird:

	p_{i0}	p_{i1}	p_{i2}	$\cdots$	p_{in}
x_0	$f(x_0)$				
x_1	$f(x_1)$	$p_{1,1}(x)$			
x_2	$f(x_2)$	$p_{2,1}(x)$	$p_{2,2}(x)$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
x_n	$f(x_n)$	$p_{n,1}(x)$	$p_{n,2}(x)$	$\cdots$	$p_{n,n}(x) = p_n(x)$

[4] Eric Harold Neville (1889–1961) war ein britischer Mathematiker, hauptsächlich im Bereich der Geometrie tätig. Als er 1914 als Gastprofessor in Indien war, konnte er den indischen Ausnahmemathematiker Srinivasa Ramanujan überreden, mit ihm zurück nach England zu kommen.

Dabei sind für $k = 1$ insgesamt $6n$ Operationen nötig, für $k = 2$ sind es $6(n-1)$ Operationen, und so weiter, bis zur Berechnung für $k = n$ mit 6 Operationen. Nach Hinzunahme einer weiteren Stützstelle x_{n+1} kann man dann den Wert des Interpolationspolynon p_{n+1} in x einfach durch Berechnung einer weiteren Zeile aus den bisher bestimmten Werten konstruieren. Dafür muss für jedes k nur $p_{n+1,k}(x)$ berechnet werden, was nun in $O(n)$ Operationen möglich ist.

7.3 POLYNOMDARSTELLUNG NACH NEWTON

Möchte man aber das Interpolationspolynom an vielen Stellen auswerten, ist es effizienter, das Polynom explizit zu berechnen. Auch für den Fall, dass man das Polynom differenzieren oder integrieren möchte, benötigt man eine Darstellung des Polynoms als Linearkombination von Basis-Polynomen. Wählt man die Basis geschickt, so lassen sich die entsprechenden Koeffizienten rekursiv mit Hilfe des Aitken–Neville-Schemas berechnen. Wir wählen dafür die so genannte *Newton-Form* des Interpolationspolynoms

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

Dabei handelt es sich offensichtlich um ein Polynom vom Grad n; außerdem gilt nach Konstruktion

$$p_n(x) = p_{n-1}(x) + a_n(x - x_0)\cdots(x - x_{n-1}). \tag{7.4}$$

Die Koeffizienten a_k können wir damit nacheinander aus den Interpolationsbedingungen berechnen:

$$\begin{aligned}
f(x_0) = p_0(x_0) = a_0 \qquad &\Rightarrow \qquad a_0 = f(x_0),\\
f(x_1) = p_1(x_1) = p_0(x_1) + a_1(x_1 - x_0) \qquad &\Rightarrow \qquad a_1 = \frac{f(x_1) - p_0(x_1)}{x_1 - x_0},\\
f(x_2) = p_2(x_2) = p_1(x_2) + a_2(x_2 - x_0)(x_2 - x_1) \qquad &\Rightarrow \qquad a_2 = \frac{f(x_2) - p_1(x_2)}{(x_2 - x_0)(x_2 - x_1)},
\end{aligned}$$

und so weiter. Da x_j in allen Linearfaktoren für a_k, $k > j$, vorkommt, ist dann

$$p_n(x_j) = p_j(x_j) = f(x_j) \qquad \text{für alle } j = 0, \ldots, n,$$

d. h. das durch diese Koeffizienten bestimmte Polynom in Newton-Form ist tatsächlich das gesuchte Interpolationspolynom.

Für die Koeffizienten der Newton-Darstellung gilt also

$$\begin{aligned}
a_k &= \frac{f(x_k) - p_{k-1}(x_k)}{(x_k - x_0)\cdots(x_k - x_{k-1})} =: [x_0, \ldots, x_k]f, \qquad k = 1, \ldots, n,\\
a_0 &= f(x_0) =: [x_0]f.
\end{aligned}$$

Diese Koeffizienten $[x_0, \ldots, x_k]f$ heißen *k-te dividierte Differenzen*; analog definiert man dividierte Differenzen niedrigerer Ordnung. Aus dem Aitken–Neville-Schema erhält man eine rekursive Formel für Ihre Berechnung: Einsetzen von (7.3) und (7.4) ergibt

$$\begin{aligned}
[x_0, \ldots, x_k]f\ (x - x_0) \cdots (x - x_{k-1}) &= p_k(x) - p_{k-1}(x) \\
&= \left(\frac{x - x_0}{x_k - x_0} p_{k,k-1}(x) + \frac{x_k - x}{x_k - x_0} p_{k-1,k-1}(x) \right) - p_{k-1,k-1}(x) \\
&= \frac{x - x_0}{x_k - x_0} \left(p_{k,k-1}(x) - p_{k-1,k-1}(x) \right) \\
&= \frac{x - x_0}{x_k - x_0} \big([x_1, \ldots, x_k]f\ (x - x_1) \cdots (x - x_{k-1}) \\
&\qquad - [x_0, \ldots, x_{k-1}]f\ (x - x_0) \cdots (x - x_{k-2}) + O\left(x^{k-2}\right) \big),
\end{aligned}$$

wobei wir alle Polynom-Terme niedrigerer Ordnung im Landau-Symbol zusammengefasst haben. Führt man nun in dieser Gleichung einen Koeffizientenvergleich für x^k durch, so erhält man daraus die Rekursionsformel

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

Das allgemeine Schema für $k = 0, \ldots, n$ lautet nun

$$\begin{cases} [x_i, \ldots, x_k]f = \dfrac{[x_{i+1}, \ldots, x_k]f - [x_i, \ldots, x_{k-1}]f}{x_k - x_i}, & i = 0, \ldots, k-1, \\ [x_k]f = f(x_k), \end{cases} \tag{7.5}$$

bzw. in Tableau-Form

	$[x_i]f$	$[x_{i-1}, x_i]f$	$[x_{i-2}, x_{i-1}, x_i]f$	$[x_{i-3}, \ldots, x_i]f$	$\cdots$
x_0	$f(x_0)$				
x_1	$f(x_1)$	$[x_0, x_1]f$			
x_2	$f(x_2)$	$[x_1, x_2]f$	$[x_0, x_1, x_2]f$		
x_3	$f(x_3)$	$[x_2, x_3]f$	$[x_1, x_2, x_3]f$	$[x_0, x_1, x_2, x_3]f$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Wieder geht man spaltenweise von oben nach unten vor und liest am Schluss die Koeffizienten a_k auf der Diagonalen ab. Hat man die Koeffizienten in der Newton-Darstellung des Interpolationspolynoms p_n einmal berechnet, kann man die Auswertung an der Stelle x effizient mit dem Horner-Schema durchführen:

$$p_n(x) = a_0 + (x - x_0)\left(a_1 + (x - x_1)\left(\cdots + (x - x_{n-2})\left(a_{n-1} + (x - x_{n-1})a_n\right)\right)\right).$$

7.4 INTERPOLATIONSFEHLER

Wir untersuchen nun den Interpolationsfehler, d. h. die Differenz zwischen der zu interpolierenden Funktion und dem Interpolationspolynom *außerhalb* der Stützstellen.

Satz 7.4. *Sei $f \in C^{n+1}([a,b])$, $a = x_0 < \cdots < x_n = b$, und $p_n \in P_n$ das Interpolationspolynom zu diesen Stützstellen. Dann existiert für alle $x \in [a,b]$ ein $\xi \in [a,b]$ mit*

$$f(x) - p_n(x) = (x - x_0) \cdots (x - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{7.6}$$

Beweis. Sei $x \in [a,b]$ gegeben. Für $x = x_i$ sind beide Seiten von (7.6) gleich Null; deshalb reicht es, nur $x \neq x_i$, $0 \leq i \leq n$, zu betrachten. Wir definieren dazu für $K \in \mathbb{R}$

$$g(t) := f(t) - p_n(t) - K(t - x_0) \cdots (t - x_n).$$

Diese Funktion ist $n+1$-mal stetig differenzierbar und hat die Nullstellen x_i, $0 \leq i \leq n$, da p_n die Interpolationsbedingungen erfüllt und der letzte Term in den Stützstellen verschwindet. Wählt man nun

$$K = \frac{f(x) - p_n(x)}{(x - x_0) \cdots (x - x_n)},$$

so ist auch x eine Nullstelle von g; insgesamt hat g also $n+2$ Nullstellen. Nach dem Satz von Rolle hat die Ableitung g' daher $n+1$ Nullstellen (je eine zwischen zwei Nullstellen von g). Weiter hat g'' also n Nullstellen, g''' hat $n-1$ Nullstellen, und so weiter. Schließlich hat $g^{(n+1)}$ genau eine Nullstelle $\xi \in [a,b]$. Für diese gilt daher

$$\begin{aligned} 0 &= g^{(n+1)}(\xi) \\ &= f^{(n+1)}(\xi) - p_n^{(n+1)}(\xi) - \frac{f(x) - p_n(x)}{(x - x_0) \cdots (x - x_n)} \left[(t - x_0) \cdots (t - x_n)\right]^{(n+1)}\big|_{t=\xi} \\ &= f^{(n+1)}(\xi) - \frac{f(x) - p_n(x)}{(x - x_0) \cdots (x - x_n)} (n+1)! \,. \end{aligned} \tag{7.7}$$

Dabei haben wir verwendet, dass p_n ein Polynom vom Grad n ist (dessen $(n+1)$-te Ableitung daher verschwindet), und die letzte Klammer ein Polynom vom Grad $n+1$ ist, dessen höchster Koeffizient gleich 1 ist – also die konstante $(n+1)$-te Ableitung $(n+1)!$ hat. Lösen wir die Gleichung (7.7) nach $f(x) - p_n(x)$ auf, erhalten wir die Behauptung. □

Bemerkung. Gilt $\|f^{(n+1)}\|_\infty \leq M$, so erhält man aus (7.6) die Abschätzung

$$\|f - p_n\|_\infty \leq \|\omega_n\|_\infty \frac{M}{(n+1)!}.$$

für das *Stützstellenpolynom* $\omega_n(x) := (x - x_0) \cdots (x - x_n)$. Ist also die $(n+1)$-te Ableitung von f beschränkt, so kann der Fehler in der Interpolation trotzdem sehr groß sein, da ω_n für große n sehr starke Oszillationen aufweisen und die rechte Seite daher beliebig groß sein kann (*Runge-Phänomen*).

Eine möglicher Ausweg ist, die Stützstellen so zu wählen, dass $\|\omega_n\|_\infty$ minimal ist. (Dieser Ansatz führt auf die Interpolation mit *Tschebyscheff-Polynomen*.) Eine andere Möglichkeit besteht darin, statt mit einem Polynom hohen Grades mit stückweisen Polynomen kleineren Grades zu arbeiten, was auf die *Spline-Interpolation* führt; siehe das folgenden Kapitel.

7.5 HERMITE-INTERPOLATION

Bei der Hermite-Interpolation ist ein Polynom gesucht, das in den Stützstellen zusätzlich vorgegebene Ableitungen hat. Genauer betrachten wir folgendes Problem:

Gegeben: $x_0, \dots, x_K \in [a,b] \subset \mathbb{R}$ (paarweise verschieden), $m_j \geq 1$, $0 \leq j \leq K$, $M = \max_j m_j$, $f \in C^{M-1}([a,b])$.

Gesucht: $p \in P_n$, so dass für $0 \leq j \leq K$ gilt

$$p^{(m-1)}(x_j) = f^{(m-1)}(x_j), \qquad m = 1, \dots, m_j.$$

Diese Aufgabe hat eine eindeutige Lösung, falls

$$\sum_{j=0}^{K} m_j - 1 = n$$

ist (d. h. $n+1$ Bedingungen für die $n+1$ Koeffizienten von $p \in P_n$ gegeben sind). Die Eindeutigkeit folgt wieder aus dem Fundamentalsatz, wenn man die Nullstellen $x_0, \dots, x_K$ nach ihrer Vielfachheit m_j zählt. Genauso vergewissert man sich auch, dass der Beweis von Satz 7.4 analog durchgeführt werden kann, indem man die Nullstellen von $g^{(j)}$, die jeweils aufgrund der kleineren Anzahl von Stützstellen im Satz von Rolle fehlen, durch die Interpolationsbedingungen der entsprechenden Ableitungen ersetzt.

Satz 7.5. *Sei $f \in C^{n+1}([a,b])$, $a = x_0 < \cdots < x_K = b$, und $p_n \in P_n$ das Hermite-Interpolationspolynom zu diesen Stützstellen. Dann existiert für alle $x \in [a,b]$ ein $\xi \in [a,b]$ mit*

$$f(x) - p_n(x) = (x - x_0)^{m_0} \cdots (x - x_K)^{m_K} \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Die Berechnung des Hermite-Interpolationspolynoms kann man mit Hilfe des Aitken–Neville-Schemas durchführen, wenn dieses geeignet erweitert wird. Als Vorbereitung überlegen wir uns, was mit den dividierten Differenzen passiert, wenn einzelne Stützstellen zusammenfallen, d. h. $x_i = x_j$ gilt für $i \neq j$. Dazu wenden wir einen Trick an: Wir betrachten wieder $a = x_0 < \cdots < x_n = b$ und führen ein beliebiges $x \in [a,b] \setminus \{x_0, \dots, x_n\}$ als neue Stützstelle ein. Betrachten wir nun das Interpolationspolynom p_{n+1} zu den Stützstellen $x_0, \dots, x_n, x$ in Newton-Form, dann gilt

$$f(x) = p_{n+1}(x) = p_n(x) + [x_0, \dots, x_n, x]f \; (x - x_0) \cdots (x - x_n)$$

und deshalb

$$f(x) - p_n(x) = [x_0, \dots, x_n, x]f \; (x - x_0) \cdots (x - x_n).$$

Die linke Seite ist identisch mit der in Gleichung mit (7.6), wir erhalten daher durch Koeffizientenvergleich

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

für ein $\xi \in [x_0, x_n]$. Da x und n beliebig waren, können wir auch x in x_{n+1} und dann n in $n-1$ umbenennen. Es existiert also für $f \in C^n([a,b])$, $a = x_0 < \cdots < x_n = b$ ein $\xi \in [a,b]$, so dass gilt

$$[x_0, \dots, x_n]f = \frac{f^{(n)}(\xi)}{n!}.$$

In dieser Darstellung können wir jetzt alle x_j gegen x_0 konvergieren lassen, und erhalten, da ξ zwischen x_0 und x_n liegt und $f^{(n)}$ stetig ist, dass gilt

$$[x_0, \dots, x_0]f = \frac{1}{n!} f^{(n)}(x_0).$$

Die dividierten Differenzen in $n+1$ zusammenfallenden Punkten konvergieren also gegen die (skalierte) n-te Ableitung.

Diese Tatsache können wir nun für die Hermite-Interpolation mit Newton-Darstellung wie folgt ausnutzen:

1. Schreibe die Stützstellen x_j, $j = 0, \dots, K$ jeweils m_j mal untereinander.
2. Tauchen im Tableau dividierte Differenzen zu identischen Stützstellen auf, verwende statt der Rekursionsrelation (7.5) die Vorschrift

$$[\underbrace{x_k, \dots, x_k}_{m\text{−mal}}]f = \frac{1}{(m-1)!} f^{(m-1)}(x_k).$$

(Sind nur einige der Stützstellen identisch, ist die Rekursion (7.5) wohldefiniert.)

Wir betrachten zwei Beispiele:

Beispiel 7.6.

(i) $K = 0$, $m_0 = 4$: In x_0 ist also der Funktionswert und die ersten drei Ableitungen vorgegeben, und wir können ein Interpolationspolynom $p \in P_3$ finden. Das Tableau ist nun

	$[x_0]f$	$[x_0, x_0]f$	$[x_0, x_0, x_0]f$	$[x_0, \dots, x_0]f$
x_0	$f(x_0)$			
x_0	$f(x_0)$	$f'(x_0)$		
x_0	$f(x_0)$	$f'(x_0)$	$\frac{1}{2}f''(x_0)$	
x_0	$f(x_0)$	$f'(x_0)$	$\frac{1}{2}f''(x_0)$	$\frac{1}{6}f'''(x_0)$

Auf der Diagonalen stehen wieder die Koeffizienten des Interpolationspolynoms

$$p(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \frac{1}{6}f'''(x_0)(x - x_0)^3$$

mit Interpolationsfehler

$$f(x) - p(x) = \frac{1}{24} f^{(4)}(\xi)(x - x_0)^4,$$

was der Taylorentwicklung von f um x_0 entspricht.[5]

(ii) $K = 2, m_0 = 1, m_1 = 2, m_2 = 1$: In x_1 ist hier zusätzlich die erste Ableitung geben. Es ist $m_0 + m_1 + m_2 = 4$, und wir suchen wieder ein Interpolationspolynom $p \in P_3$. Das Tableau ist hier

	$[x_i]f$	$[x_{i-1}, x_i]f$	$[x_{i-2}, \ldots, x_i]f$	$[x_{i-3}, \ldots, x_i]f$
x_0	$f(x_0)$			
x_1	$f(x_1)$	$[x_0, x_1]f$		
x_1	$f(x_1)$	$[x_1, x_1]f$	$[x_0, x_1, x_1]f$	
x_2	$f(x_2)$	$[x_1, x_2]f$	$[x_1, x_1, x_2]f$	$[x_0, x_1, x_1, x_2]f$

Alle Terme bis auf $[x_1, x_1]f$ – den wir durch $f'(x_1)$ ersetzen dürfen – können wir dabei mit Hilfe von (7.5) berechnen. Wir erhalten das Polynom

$$\begin{aligned} p(x) = f(x_0) + [x_0, x_1]f(x - x_0) + [x_0, x_1, x_1]f(x - x_0)(x - x_1) \\ + [x_0, x_1, x_1, x_2]f(x - x_0)(x - x_1)^2, \end{aligned}$$

und für den Interpolationsfehler wieder

$$f(x) - p(x) = \frac{1}{24} f^{(4)}(\xi)(x - x_0))(x - x_1)^2(x - x_2).$$

7.6 NUMERISCHE DIFFERENTIATION

Die Polynomdarstellung nach Newton lässt sich z. B. ausnützen, um die Ableitung einer Funktion numerisch zu berechnen: Wir wählen Stützstellen $x_0, \ldots, x_n$, interpolieren f durch ein Polynom $p_n \in P_n$, und differenzieren dann das Polynom. Da wir Polynome exakt differenzieren können, wird nur der Interpolationsfehler die Genauigkeit dieser Approximation bestimmen.

Angenommen, wir wollen eine Funktion $f : [a, b] \to \mathbb{R}$ in einem Punkt $x_0 \in [a, b]$ auswerten. Wir betrachten dafür zu den Stützstellen $x_0, \ldots, x_n$ das Interpolationspolynom in Newtonform:

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

und den Interpolationsfehler

$$r_n(x) = (x - x_0) \cdots (x - x_n) \frac{f^{(n+1)}(\xi(x))}{(n+1)!}.$$

[5] Tatsächlich hat Brook Taylor um 1715 die nach ihm benannte Entwicklung genau auf diesem Weg hergeleitet.

Ist $f \in C^{(n+2)}([a,b])$, so können wir beide Gleichungen differenzieren und $x = x_0$ einsetzen:

$$\begin{aligned} p_n'(x_0) &= [x_0, x_1]f \; + [x_0, x_1, x_2]f \; (x_0 - x_1) + \cdots \\ &\qquad + [x_0, \ldots, x_n]f \; (x_0 - x_1) \cdots (x_0 - x_{n-1}), \\ r_n'(x_0) &= (x_0 - x_1) \cdots (x_0 - x_n) \frac{f^{(n+1)}(\xi(x_0))}{(n+1)!}. \end{aligned}$$

Dabei haben wir ausgenützt, dass alle Terme, die aufgrund der Produktregel entstehen und $(x - x_0)$ enthalten, beim Einsetzen von $x = x_0$ wegfallen; übrig bleiben lediglich die Terme, in denen der Faktor $(x - x_0)$ wegdifferenziert wurde. Dadurch erhalten wir:

$$f'(x_0) = p_n'(x_0) + r_n'(x_0).$$

Je mehr Stützstellen hinzugenommen werden, desto genauer wird also die Approximation sein. Andererseits können wir Interpolationspolynom und -Fehler auch weiter differenzieren, um höhere Ableitungen näherungsweise zu berechnen. Insbesondere erhalten wir für die n-te Ableitung:

Satz 7.7. *Ist $f \in C^{n+2}([a,b])$ und sind $x_j \in [a,b]$, $j = 0, \ldots, n$, paarweise verschieden, so existiert ein $\xi \in [a,b]$, so dass gilt:*

$$f^{(n)}(x_0) = n![x_0, \ldots, x_n]f + (x_0 - x_1) \cdots (x_0 - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{7.8}$$

Sind die Stützstellen im Abstand $h > 0$ um x_0 verteilt, erhält man die klassischen *Differenzenquotienten*:

Beispiel 7.8.

(i) *Vorwärtsdifferenzenquotient* für die erste Ableitung: $x_1 = x_0 + h$,

$$\begin{aligned} f'(x_0) = p_1'(x_0) + r_1'(x_0) &= [x_0, x_1]f + (x_0 - x_1)\tfrac{1}{2}f''(\xi) \\ &= \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2}f''(\xi). \end{aligned}$$

Der Verfahrensfehler ist also $O(h)$ für $h \to 0$.

(ii) *Zentrale Differenzenquotienten*: $x_1 = x_0 + h$, $x_2 = x_0 - h$. Die drei Stützstellen legen ein Interpolationspolynom vom Grad $n = 2$ fest, das wir verwenden können, um die erste oder die zweite Ableitung von f in x_0 berechnen zu können.

Für die zweite Ableitung verwenden wir Satz 7.7 und die Rekursionsformel (7.5) und berechnen (Übung!)

$$\begin{aligned} f''(x_0) &= 2![x_0, x_1, x_2]f + (x_0 - x_1)(x_0 - x_2)\frac{1}{3!}f'''(\xi) \\ &= \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - h^2\frac{1}{6}f'''(\xi). \end{aligned}$$

Die erste Ableitung erhalten wir wieder aus der ersten Ableitung des Interpolationspolynoms und der Rekursionsformel:

$$\begin{aligned} f'(x_0) &= p_2'(x_0) + r_2'(x_0) \\ &= [x_0, x_1]f + [x_0, x_1, x_2]f\ (x_0 - x_1) + (x_0 - x_1)(x_0 - x_2)\frac{1}{6}f'''(\xi) \\ &= \frac{f(x_0 + h) - f(x_0 - h)}{2h} - h^2\frac{1}{6}f'''(\xi). \end{aligned}$$

In beiden Fällen ist der Fehler $O(h^2)$ für $h \to 0$.

WEITERFÜHRENDE LITERATUR

G. Hämmerlin & K.-H. Hoffmann (1994), *Numerische Mathematik*, 4. Aufl., Springer-Verlag, Berlin.

8 SPLINE-INTERPOLATION

Das Problem, dass der Polynomgrad bei der Polynominterpolation bei wachsender Stützstellenzahl zwangsläufig mit wächst – was zu immer größeren Oszillationen außerhalb der Stützstellen führt – kann umgangen werden, indem man die interpolierende Funktion stückweise aus Polynomen niedrigen Grades zusammensetzt. Da wir dabei die angenehme Eigenschaften von Polynomen, (beliebig) viele stetige Ableitungen zu haben, nicht völlig aufgeben wollen, fordern wir zusätzlich, dass dieses Stücke möglichst glatt zusammenfassen.

Definition 8.1 (Splines). Zu der vorgegebenen Menge $\Delta := \{x_0, \dots, x_n\}$ der *Knoten* $a = x_0 < \cdots < x_n = b$ heißt

$$s \in S_m(\Delta) := \left\{ s \in C^{m-1}([a,b]) \;:\; s|_{[x_i, x_{i+1}]} \in P_m,\ x_i \in \Delta \right\}$$

Spline vom Grad m.

Zwischen je zwei Knoten soll ein Spline s vom Grad m also ein Polynom vom Grad m sein, an den Knoten soll der Übergang so erfolgen, dass die Funktion über das gesamte Definitionsgebiet $(m-1)$-mal stetig differenzierbar ist. (Dabei wollen wir für $m = 0$ die Bedingung $s \in C^{m-1}([a,b])$ fallen lassen; $S_0(\Delta)$ besteht also aus stückweise konstanten Funktionen.)

Für die Interpolationsaufgabe ist wichtig, wie viele Freiheitsgrade ein Spline noch besitzt, der diesen Bedingungen gehorcht:

Lemma 8.2. *$S_m(\Delta)$ hat die Dimension $m + n$.*

Beweis. Sei $s \in S_m(\Delta)$. Wir betrachten zunächst das Intervall $[x_0, x_1]$. Dort ist $s|_{[x_0,x_1]}$ ein Polynom vom Grad m, hat also $m+1$ Freiheitsgrade. Das nächste Teilstück $s|_{[x_1,x_2]}$ ist auch ein Polynom vom Grad m, muss aber in x_1 die m Bedingungen erfüllen, dass gilt

$$\left(s|_{[x_0,x_1]}\right)^{(k)}(x_1) = \left(s|_{[x_1,x_2]}\right)^{(k)}(x_1) \qquad \text{für alle } 0 \le k \le m-1.$$

Übrig bleibt also nur noch ein Freiheitsgrad. Die gleiche Überlegung gilt für $s|_{[x_i,x_{i+1}]}$ im Punkt x_i für $2 \le i \le n-1$. Insgesamt ergibt dies $m+1+(n-1) = m+n$ Freiheitsgrade. □

8.1 LINEARE SPLINE-INTERPOLATION

Wir untersuchen den Themenbereich zuerst an einem einfachen Beispiel, bevor wir den allgemeinen Fall behandeln. Dafür betrachten wir die linearen Splines, das heißt stückweise lineare Funktionen, die in den Knoten stetig sind. Die Interpolationsaufgabe besteht also darin, zu einer Funktion f und gegebenen Knoten (die wir gleichzeitig als Stützstellen verwenden) $x_0 < x_1 < \cdots < x_n$ eine Funktion $s \in S_1(\Delta)$ zu finden, so dass $s(x_i) = f(x_i)$ für alle $0 \leq i \leq n$ gilt. Durch die Wahl der Knoten als Stützstellen sind hier die Stetigkeitsbedingungen automatisch erfüllt, so dass wir nur noch auf jedem Teilintervall $[x_i, x_{i+1}]$ ein lineares Interpolationspolynom finden müssen, ohne jeweils die anderen Teilintervalle betrachten zu müssen. Dieses Polynom können wir direkt angeben, zum Beispiel in der Newton-Form:

$$s|_{[x_i,x_{i+1}]}(x) = f(x_i) + (x - x_i)[x_i, x_{i+1}]f.$$

Insgesamt ergibt sich als linearer Interpolations-Spline einfach der Streckenzug, der die Funktionswerte verbindet. Die Eindeutigkeit zeigen wir wieder analog zu Lemma 8.2. Angenommen, wir haben zwei lineare Splines $s, \tilde{s} \in S_1(\Delta)$ mit $s(x_i) = f(x_i) = \tilde{s}(x_i)$. Dann ist auf jedem Teilintervall $(s - \tilde{s})|_{[x_i,x_{i+1}]}$ ein lineares Polynom, dass in den zwei Endpunkten x_i und x_{i+1} verschwindet; wegen der Eindeutigkeit der Polynominterpolation ist also $s - \tilde{s}$ konstant gleich 0 auf jedem Teilintervall $[x_i, x_{i+1}]$ und damit auf ganz $[a, b]$. Also gilt $s = \tilde{s}$.

Mit Hilfe von Satz 7.4 können wir auch den Interpolationsfehler abschätzen. Setzen wir $h_i := x_{i+1} - x_i$, so gilt für alle $0 \leq i \leq n$

$$f(x) - s|_{[x_i,x_{i+1}]}(x) = (x - x_i)(x - x_{i+1})\frac{1}{2}f''(\xi) \leq h_i^2 \frac{1}{2}\|f''\|_\infty \tag{8.1}$$

mit einem $\xi \in [x_i, x_{i+1}]$. Haben alle Stützstellen den gleichen Abstand h (oder ist h der maximale Abstand zwischen zwei benachbarten Stützstellen), so erhalten wir

$$\|f - s\|_\infty \leq \frac{h^2}{2}\|f''\|_\infty.$$

Ist f'' also beschränkt, so können wir den Fehler beliebig klein machen, indem wir genug Stützstellen hinzunehmen.

Wir benötigen später auch eine Fehlerabschätzung bezüglich der L^2-*Norm*

$$\|f\|_{L^2} := \left(\int_a^b f(x)^2\, dx\right)^{1/2}.$$

Satz 8.3. *Sei $f \in C^2([a, b])$ und erfülle $s \in S_1(\Delta)$ die Interpolationsbedingungen. Dann gilt für $h := \max_{0\leq i\leq n-1}(x_{i+1} - x_i)$*

$$\|f - s\|_{L^2} \leq \frac{h^2}{2}\|f''\|_{L^2}. \tag{8.2}$$

Beweis. Betrachte $e := f - s$. Aus den Interpolationsbedingungen folgt dann $e(x_i) = 0$ für $i = 0, \dots, n$ und daher nach dem Hauptsatz der Differenzial- und Integralrechnung

$$e(x) = \int_{x_i}^{x} e'(t)\,dt \qquad \text{für alle } x \in [x_i, x_{i+1}],\ i = 0, \dots, n-1.$$

Aus der Hölder-Ungleichung $\int fg \le (\int f^2)^{1/2} (\int g^2)^{1/2}$ folgt dann

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} e(y)^2\,dy &= \int_{x_i}^{x_{i+1}} \left(\int_{x_i}^{y} 1 \cdot e'(t)\,dt \right)^2 dy \\
&\le \int_{x_i}^{x_{i+1}} (y - x_k) \int_{x_i}^{y} e'(t)^2\,dt\,dy \\
&\le \int_{x_i}^{x_{i+1}} (y - x_k)\,dy \int_{x_i}^{x_{i+1}} e'(t)^2\,dt \\
&= \frac{h_i^2}{2} \int_{x_i}^{x_{i+1}} e'(t)^2\,dt.
\end{aligned}$$

Wir integrieren jetzt partiell:

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} e'(t)^2\,dt &= \left[e'(t)e(t) \right]_{t=x_i}^{t=x_{i+1}} - \int_{x_i}^{x_{i+1}} e(t)e''(t)\,dt \\
&= -\int_{x_i}^{x_{i+1}} (f(t) - s(t))f''(t)\,dt
\end{aligned}$$

wegen der Interpolationsbedingungen und $s'' = 0$.

Abschätzen von $h_i \le h$, Summieren über $i = 0, \dots, n-1$ und ein weiteres Anwenden der Hölder-Ungleichung ergibt dann

$$\begin{aligned}
\|f - s\|_{L^2}^2 = \int_a^b (f(x) - s(x))^2\,dx &\le \frac{h^2}{2} \left| \int_a^b (f(x) - s(x))f''(x)\,dx \right| \\
&\le \frac{h^2}{2} \left(\int_a^b (f(x) - s(x))^2\,dx \right)^{1/2} \left(\int_a^b f''(x)^2\,dx \right)^{1/2} \\
&= \frac{h^2}{2} \|f - s\|_{L^2} \|f''\|_{L^2}.
\end{aligned}$$

Ist nun $f \neq s$ (sonst ist nichts zu zeigen) können wir auf beiden Seiten durch $\|f - s\|_{L^2} > 0$ dividieren und erhalten die Aussage. □

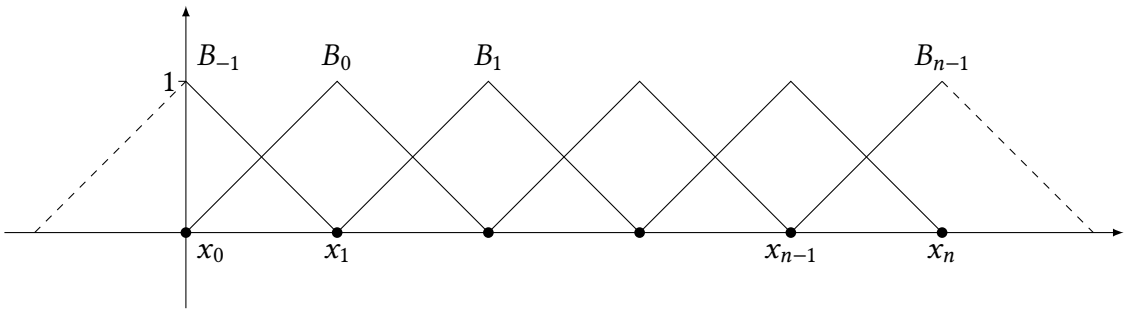


Abbildung 8.1: Die Hutfunktionen B_i für $i = -1, \ldots, n-1$ mit den "Geisterknoten" x_{-1} und x_{n+1}. Gestrichelt sind die Teile von B_i, die außerhalb des Definitionsbereiches $[x_0, x_n]$ liegen.

Genauso wie bei der Polynom-Interpolation sind wir auch bei der Spline-Interpolation an einer Basis-Darstellung der interpolierenden Funktion interessiert. Da nach Lemma 8.2 die Dimension von $S_1(\Delta)$ gleich $n+1$ ist, suchen wir also $n+1$ linear unabhängige lineare Splines, die $S_1(\Delta)$ aufspannen. Dafür definieren wir die *Hut-Funktionen*

$$B_i(x) := \begin{cases} \frac{x-x_i}{x_{i+1}-x_i} & \text{für } x_i < x \le x_{i+1}, \\ \frac{x_{i+2}-x}{x_{i+2}-x_{i+1}} & \text{für } x_{i+1} < x \le x_{i+2}, \\ 0 & \text{sonst,} \end{cases} \tag{8.3}$$

siehe Abb. 8.1. Durch Einsetzen erkennt man leicht, dass gilt:

$$B_i(x_j) = \begin{cases} 1 & \text{falls } i = j-1, \\ 0 & \text{falls } i \neq j-1. \end{cases} \tag{8.4}$$

Damit die Definition (8.3) auch für $i = -1$ und $i = n-1$ Sinn ergibt, müssen wir die "Geisterknoten" $x_{-1} < x_0$ und $x_{n+1} > x_n$ einführen (deren genaue Lage nicht wichtig ist). Aus (8.4) folgt sofort, dass für f ein linearer interpolierender Spline $s \in S_1(\Delta)$ gegeben ist durch

$$s(x) = \sum_{j=0}^{n} f(x_j) B_{j-1}(x), \qquad x \in [x_0, x_n].$$

Um die Eindeutigkeit zu zeigen, weisen wir nach, dass diese B_i eine Basis von $S_1(\Delta)$ bilden. Einerseits gilt für alle $s \in S_1(\Delta)$, dass $s = \sum_{j=-1}^{n-1} s(x_j) B_j(x)$ ist, da auf jedem Teilstück $[x_j, x_{j+1}]$ der Spline s (als lineares Polynom) auf Grund der Eindeutigkeit der Polynominterpolation sein eigenes Interpolationspolynom ist. Jeder Spline s lässt sich also als Linearkombination der B_j darstellen. Andererseits sei $s(x) = \sum_{j=-1}^{n-1} c_j B_j(x) = 0$. Aus (8.4) folgt dann sofort, dass $c_j = s(x_{j+1}) = 0$ für $-1 \le j \le n-1$ gilt; die B_j sind also linear unabhängig, und deshalb eine Basis von $S_1(\Delta)$.

8.2 KUBISCHE SPLINE-INTERPOLATION

Für Splines höherer Ordnung muss man zusätzliche Bedingungen an den Randpunkten x_0 und x_n fordern, damit die Interpolationsaufgabe in den Randintervallen $[x_0, x_1]$ und $[x_{n-1}, x_n]$ wohldefiniert ist. Dies kann man entweder über zusätzliche Geisterknoten oder durch Vorgabe der Ableitungen erreichen. Von besonderem Interesse sind dabei Splines ungerader Ordnung; sowohl wegen der Symmetrie der "Randbedingungen" als auch wegen ihrer Optimalitätseigenschaften. Wir betrachten hier beispielhaft den Fall $m = 3$ *kubischer Splines*.

Wir suchen also zu einer Funktion $f \in C^2([a,b])$ ein $s \in S_3(\Delta)$ so, dass gilt

$$s(x_i) = f(x_i), \qquad i = 0, \dots, n.$$

Dies sind $n+1$ Interpolationsbedingungen; nach Lemma 8.2 fehlen also noch zwei zusätzliche Randbedingungen. Möglich sind z. B.

- *natürliche Randbedingungen*: $s''(a) = 0 = s''(b)$;
- *Hermite-Randbedingungen*: $s'(a) = f'(a)$ und $s'(b) = f'(b)$;
- *periodische Randbedingungen*: $s'(a) = s'(b)$ und $s''(a) = s''(b)$.

(Letzteres macht nur Sinn, wenn f dieselben Bedingungen erfüllt, während Hermite-Randbedingungen zusätzliche Interpolationsbedingungen darstellen.) Wir zeigen nun, dass für jede dieser Randbedingungen das kubische Spline-Interpolationsproblem eine eindeutige Lösung hat, und dass diese Lösung unter *allen* so interpolierenden Funktionen die minimale zweite Ableitung bezüglich der L^2 Norm hat. Dafür benötigen wir das folgende Lemma.

Lemma 8.4. *Sei $s \in S_3(\Delta)$ ein kubischer Spline, der die Interpolationsbedingungen für $f \in C^2([a,b])$ sowie natürliche (Hermite-, periodische) Randbedingungen erfüllt. Dann gilt für jede Funktion $g \in C^2([a,b])$, die die selben Interpolationsbedingungen (sowie Randbedingungen im Hermite-Fall bzw. periodischen Fall) erfüllt,*

$$\int_a^b g''(x)^2\,dx = \int_a^b (g''(x) - s''(x))^2\,dx + \int_a^b s''(x)^2\,dx.$$

Beweis. Aus der Linearität des Integrals folgt

$$\begin{aligned}\int_a^b (g''(x) - s''(x))^2\,dx &= \int_a^b g''(x)\,dx - 2\int_a^b g''(x)s''(x)\,dx + \int_a^b s''(x)^2\,dx \\ &= \int_a^b g''(x)\,dx - 2\int_a^b (g''(x) - s''(x))s''(x)\,dx - \int_a^b s''(x)^2\,dx,\end{aligned}$$

es genügt also zu zeigen, dass der mittlere Term verschwindet. Dazu integrieren wir partiell (auf jedem Teilintervall von Δ separat, auf dem s''' wegen $s \in S_3(\Delta)$ konstant ist):

$$\int_a^b (g-s)''(x)s''(x)\,dx = [(g-s)'(x)s''(x)]_{x=a}^{x=b} - \int_a^b (g'(x)-s'(x))s'''(x)\,dx.$$

Für den ersten Term verwenden wir nun die Randbedingungen:

natürliche Randbedingungen: hier gilt $(g-s)'(a)s''(a) = 0 = (g-s)'(b)s''(b)$ wegen $s''(a) = 0 = s''(b)$ (beachte, dass g hier *keine* Randbedingungen erfüllen muss);

Hermite-Randbedingungen: hier gilt $(g-s)'(a)s''(a) = 0 = (g-s)'(b)s''(b)$ wegen $g'(a) = f'(a) = s'(a)$ und $g'(b) = f'(b) = s'(b)$;

periodische Randbedingungen: hier gilt

$$(g'-s')(b)s''(b) - (g'-s')(a)s''(a) = (g'-s')(b)(s''(b)-s''(a)) = 0.$$

Für den zweiten Term spalten wir das Integral auf und verwenden, dass $s'''|_{[x_j,x_{j+1}]} = s_j'''$ stückweise konstant ist:

$$\begin{aligned}\int_a^b (g'(x)-s'(x))s'''(x)\,dx &= \sum_{j=0}^{n-1} s_j''' \int_{x_j}^{x_{j+1}} (g-s)'(x)\,dx \\ &= \sum_{j=0}^{n-1} s_j''' \,[(g-s)(x)]_{x=x_j}^{x=x_{j+1}} = 0\end{aligned}$$

wegen $s(x_j) = f(x_j) = g(x_j)$ aufgrund der Interpolationsbedingungen. □

Damit können wir nun den Hauptsatz über kubische Interpolation zeigen.

Satz 8.5. *Sei $f \in C^2([a,b])$ und erfülle $s \in C^2([a,b])$ die Interpolationsbedingungen für f sowie natürliche (Hermite-, periodische) Randbedingungen. Dann sind äquivalent*

(i) $s \in S_3(\Delta)$;

(ii) für jede Funktion $g \in C^2([a,b])$, die die selben Bedingungen im Sinne von Lemma 8.4 erfüllt, gilt

$$\|g''\|_{L^2} \geq \|s''\|_{L^2}.$$

Weiterhin existiert genau eine Funktion, die diese Bedingungen erfüllt.

Beweis. Wir zerlegen den Beweis in Teilschritte, die aufeinander aufbauen.

(i) ⇒ (ii): Aus Lemma 8.4 folgt für $s \in S_3(\Delta)$ und $g \in C^2([a,b])$ wie angenommen sofort

$$\|g''\|_{L^2}^2 - \|s''\|_{L^2}^2 = \|g''-s''\|_{L^2}^2 \geq 0.$$

Eindeutigkeit für (i): Seien $s, \tilde{s} \in S_3(\Delta) \subset C^2([a,b])$ zwei interpolierende kubische Splines. Dann gilt nach dem eben bewiesenen $\|s''\|_{L^2} = \|\tilde{s}''\|_{L^2}$ und damit wegen Lemma 8.4

$$\|s'' - \tilde{s}''\|_{L^2}^2 = \|s''\|_{L^2}^2 - \|\tilde{s}''\|_{L^2}^2 = 0.$$

Also ist $(s - \tilde{s})'' = s'' - \tilde{s}'' = 0$, d. h. es muss gelten $s - \tilde{s} \in P_1$. Aus den Interpolationsbedingungen folgt aber insbesondere $s(x_0) = \tilde{s}(x_0)$ und $s(x_n) = \tilde{s}(x_n)$. Damit ist nach dem Hauptsatz der Algebra $s - \tilde{s} \in P_1$ das Nullpolynom und daher $s = \tilde{s}$.

Existenz für (i): Dies folgt nun aus der Eindeutigkeit zusammen mit $\dim S_3(\Delta) = n + 3$ und der Tatsache, dass die Interpolationsbedingungen zusammen mit den Randbedingungen genau $n + 3$ Freiheitsgrade festlegen.

(ii) $\Rightarrow$ (i): Erfülle $s \in C^2([a,b])$ die Aussage (ii) und sei $s^* \in S_3(\Delta)$ die gerade bewiesene eindeutige Lösung der Spline-Interpolationsaufgabe. Dann gilt wieder $\|s''\|_{L^2} = \|(s^*)''\|_{L^2}$, und wie im Beweis der Eindeutigkeit folgt $s = s^* \in S_3(\Delta)$. □

Für die Fehlerabschätzung verwenden wir das folgende Lemma.

Lemma 8.6. *Sei $f \in C^2([a,b])$ und erfülle $s \in S_3(\Delta)$ die Interpolationsbedingungen für f sowie natürliche (Hermite-, periodische falls f periodisch ist) Randbedingungen. Dann gilt für alle $r \in S_1(\Delta)$*

$$\|f'' - s''\|_{L^2} \leq \|f'' - r\|_{L^2}.$$

Beweis. Sei $r \in S_1(\Delta)$ und wähle $\tilde{s} \in S_3(\Delta)$ so, dass $\tilde{s}'' = r$ gilt und die gewählten Randbedingungen erfüllt sind (z. B. durch $\tilde{s}(x) = \int_a^b \max\{0, x - t\} r(t)\, dt$ für Hermite-Randbedingungen). Betrachte nun $g := f - \tilde{s} \in C^2([a,b])$. Dann verifiziert man leicht, dass $s - \tilde{s} \in S_3(\Delta)$ die Interpolationsbedingungen für g sowie die Randbedingungen erfüllt. Wir können also Lemma 8.4 anwenden und erhalten

$$\begin{aligned}\|f'' - r\|_{L^2}^2 = \|(f - \tilde{s})''\|_{L^2}^2 &= \|g'' - (s - \tilde{s})''\|_{L^2}^2 + \|(s - \tilde{s})''\|_{L^2}^2 \\ &\geq \|g'' - (s - \tilde{s})''\|_{L^2}^2 = \|f'' - s''\|_{L^2}^2.\end{aligned}$$ □

Satz 8.7. *Sei $f \in C^4([a,b])$ und erfülle $s \in S_3(\Delta)$ die Interpolationsbedingungen für f sowie natürliche (Hermite-, periodische falls f periodisch ist) Randbedingungen. Dann gilt*

$$\|f - s\|_{L^2} \leq \frac{h^4}{4} \|f^{(4)}\|_{L^2}.$$

Beweis. Sei $r \in S_1(\Delta)$ die lineare Spline-Interpolation von f''. Dann gilt nach Lemma 8.6 und (8.2)

$$\|f'' - s''\|_{L^2} \leq \|f'' - r\|_{L^2} \leq \frac{h^2}{2} \|(f'')''\|_{L^2}.$$

Nun ist wegen der Interpolationsbedingungen $(f - s)(x_i) = 0$ für alle $i = 0, \dots, n$. Die eindeutige lineare Spline-Interpolation $\varphi \in S_1(\Delta)$ für $f - s \in C^2([a, b])$ ist daher gegeben durch $\varphi = 0$. Also folgt wieder aus (8.2)

$$\|f - s\|_{L^2} = \|(f - s) - \varphi\|_{L^2} \leq \frac{h^2}{2}\|(f - s)''\|_{L^2},$$

und zusammen erhalten wir die Aussage. □

Die Basisdarstellung betrachten wir gleich für den allgemeinen Fall.

8.3 B-SPLINES

Wir verallgemeinern nun die Betrachtung am Ende vom Abschnitt 8.1 auf Splines vom Grad m. Wir betrachten wieder die Knotenmenge $\Delta = \{x_0, \dots, x_n\}$ und die Geisterknoten $x_{-m} < \cdots < x_{-1}$ und $x_{n+1} < \cdots < x_{n+m}$. Die Basis- oder *B-Splines* höheren Grades definieren wir rekursiv:

Definition 8.8. Für $-m \leq j \leq n - 1$ und $m \geq 0$ sind die B-Splines vom Grad m definiert als

$$B_{mj}(x) = \frac{x - x_j}{x_{j+m} - x_j} B_{m-1,j}(x) + \frac{x_{j+m+1} - x}{x_{j+m+1} - x_{j+1}} B_{m-1,j+1}(x),$$

$$B_{0j}(x) = \begin{cases} 1 & \text{für } x_j \leq x < x_{j+1}, \\ 0 & \text{sonst.} \end{cases}$$

Aus den Rekursionsformeln folgen direkt einige nützliche Eigenschaften.

Lemma 8.9. *Für die B-Splines B_{mj}, $-m \leq j \leq n - 1$ und $m \geq 0$, gilt*

(i) $B_{mj}(x) = 0$ *für* $x \notin [x_j, x_{j+m+1})$;

(ii) $B_{mj}(x) > 0$ *für* $x \in (x_j, x_{j+m+1})$;

(iii) $\sum_{j=-m}^{n-1} B_{mj}(x) = 1$ *für alle* $x \in [x_0, x_n)$.

Beweis. Wir verwenden Induktion nach m. Die Aussagen gelten offensichtlich für $m = 0$. Also sei angenommen, dass sie für ein $m - 1 \in \mathbb{N}$ gelten, und sei $j \in \{-m, \dots, n - 1\}$ beliebig.

Zu (i): Nach Induktionsannahme gilt $B_{m-1,j}(x) = 0$ für $x < x_j$ oder $x \geq x_{j+m}$. Weiter ist $B_{m-1,j+1}(x) = 0$ für $x < x_{j+1}$ oder $x \geq x_{j+m+1}$. Also verschwinden beide und damit $B_{mj}(x)$ für alle $x < x_j < x_{j+1}$ und für alle $x \geq x_{j+m+1} > x_{j+m}$.

Zu (ii): Sei $x \in (x_j, x_{j+m+1})$ Dann sind offensichtlich die Brüche positiv. Weiter gilt nach Induktionsannahme und (i):

- $B_{m-1,j}(x) > 0$ und $B_{m-1,j+1}(x) > 0$ für $x \in (x_{j+1}, x_{j+m})$;

- $B_{m-1,j}(x) > 0$ und $B_{m-1,j+1}(x) = 0$ für $x \in (x_j, x_{j+1}]$;
- $B_{m-1,j}(x) = 0$ und $B_{m-1,j+1}(x) > 0$ für $x \in [x_{j+m}, x_{j+m+1})$.

In jedem Fall ist also $B_{mj}(x) > 0$.

Zu (iii): Wir rechnen nach: Für $x \in [x_0, x_n)$ gilt

$$\begin{aligned}
\sum_{j=-m}^{n-1} B_{mj}(x) &= \sum_{j=-m}^{n-1} \frac{x - x_j}{x_{j+m} - x_j} B_{m-1,j}(x) + \sum_{j=-m}^{n-1} \frac{x_{j+m+1} - x}{x_{j+m+1} - x_{j+1}} B_{m-1,j+1}(x) \\
&= \frac{x - x_{-m}}{x_0 - x_{-m}} B_{m-1,-m}(x) + \sum_{j=-(m-1)}^{n-1} \frac{x - x_j}{x_{j+m} - x_j} B_{m-1,j}(x) \\
&\quad + \sum_{j=-m}^{n-2} \frac{x_{j+m+1} - x}{x_{j+m+1} - x_{j+1}} B_{m-1,j+1}(x) + \frac{x_{n+m} - x}{x_{n+m} - x_n} B_{m-1,n}(x) \\
&= \sum_{j=-(m+1)}^{n-1} \left[\frac{x - x_j}{x_{j+m} - x_j} B_{m-1,j}(x) + \frac{x_{j+m} - x}{x_{j+m} - x_j} B_{m-1,j}(x) \right] \\
&= \sum_{j=-(m+1)}^{n-1} \frac{x_{j+m} - x_j}{x_{j+m} - x_j} B_{m-1,j}(x) = 1,
\end{aligned}$$

wobei wir in der vorletzten Zeile eine Indexverschiebung $j + 1 \mapsto j$ durchgeführt haben und verwendet haben, dass nach (i) gilt $B_{m-1,-m}(x) = 0$ für $x \geq x_{-m+(m-1)+1} = x_0$ und $B_{m-1,n}(x) = 0$ für $x < x_n$. □

Analog zeigt man durch Induktion und mühsame Umformung folgende Rekursion für die Ableitungen.

Lemma 8.10. *Die Ableitungen der B-Splines B_{mj}, $-m \leq j \leq n-1$ sind für $m \geq 1$ gegeben durch*

$$B'_{mj}(x) = m \left(\frac{B_{m-1,j}(x)}{x_{j+m} - x_j} - \frac{B_{m-1,j+1}(x)}{x_{j+m+1} - x_{j+1}} \right),$$

mit der Konvention $B_{m-1,-m} = B_{m-1,n} = 0$.

Satz 8.11. *Für alle $m \geq 0$ ist $\{B_{mj} : -m \leq j \leq n-1\}$ eine Basis von $S_m(\Delta)$.*

Beweis. Zunächst ist aus der Rekursionsformel klar, dass alle B_{mj} stückweise Polynome vom Höchstgrad m sind. Weiter folgt durch $(m-1)$-fache Anwendung von Lemma 8.10, dass $B_{mj} \in C^{m-1}([x_0, x_n])$ und damit $B_{mj} \in S_m(\Delta)$ gilt. Es genügt also wegen $\dim S_m(\Delta) = m+n$ nach Lemma 8.2 zu zeigen, dass die B_{mj} linear unabhängig sind auf $[x_0, x_n]$.

Wir verwenden wieder Induktion nach m. Für $m = 0$ folgt die Aussage sofort aus $0 = \sum_{j=0}^{n-1} a_j B_{0j}(x_j) = a_j$ für alle $j = 0, \ldots, n-1$. Es sei daher $m \in \mathbb{N}$ beliebig und alle $B_{m-1,j}$,

$j = -m+1, \dots, n-1$, linear unabhängig. Sei nun $\sum_{j=-m}^{n-1} a_j B_{mj}(x) = 0$ für alle $x \in [x_0, x_n]$. Aus Lemma 8.10 folgt dann

$$\begin{aligned} 0 &= m \sum_{j=-m}^{n-1} a_j \left[\frac{B_{m-1,j}(x)}{x_{j+m} - x_j} - \frac{B_{m-1,j+1}(x)}{x_{j+1+m} - x_{j+1}} \right] \\ &= m \left[\sum_{j=-m}^{n-1} a_j \frac{B_{m-1,j}(x)}{x_{j+m} - x_j} - \sum_{j=-m+1}^{n} a_{j-1} \frac{B_{m-1,j}(x)}{x_{j+m} - x_j} \right] \\ &= m \sum_{j=-m+1}^{n-1} \frac{a_j - a_{j-1}}{x_{j+m} - x_j} B_{m-1,j}(x), \end{aligned}$$

wobei wir $B_{m-1,-m} = B_{m-1,n} = 0$ in der Rekursion für die Ableitungen verwendet haben. Nach Induktionsannahme sind die $B_{m-1,j}$ nun linear unabhängig, so dass wegen $x_{j+m} - x_j > 0$ gelten muss $a_j = a_{j-1}$ für alle $j = -m-+1, \dots, n-1$ und damit $a_j = a$ für alle $j = -m, \dots, n-1$ für ein $a \in \mathbb{R}$. Aus Lemma 8.9 (iii) folgt dann

$$0 = \sum_{j=-m}^{n-1} a_j B_{mj}(x) = a \sum_{j=-m}^{n-1} B_{mj}(x) = a$$

und damit $a_j = 0$ für alle $j = -m, \dots, n-1$. □

Für die Berechnung eines interpolierenden Splines wählt man nun $n + m$ Stützstellen

$$a \le \xi_{-m} < \cdots < \xi_{n-1} \le b$$

so, dass $B_{mj}(\xi_j) \neq 0$ ist für $j = -m, \dots, n-1$ (was nach Lemma 8.9 (ii) für $\xi_j \in (x_j, x_{j+m+1})$ garantiert ist) und bestimmt die Koeffizienten $a := (a_{-m}, \dots, a_{n-1}) \in \mathbb{R}^{n+m}$ als Lösung des Gleichungssystems

$$\sum_{j=-m}^{n} a_j B_{mj}(\xi_k) = f(\xi_k), \qquad k = -m, \dots, n-1,$$

welches unter dieser Annahme wegen der linearen Unabhängigkeit eine eindeutige Lösung hat. Dabei bestimmt man $B_{mj}(\xi_j)$ jeweils mit dem Rekursionsschema und nutzt aus, dass sich die Summe wegen Lemma 8.9 (i) vereinfachen lässt, da $B_{mj}(\xi_k) = 0$ für $\xi_k \notin (x_j, x_{j+m+1})$ ist.

WEITERFÜHRENDE LITERATUR

L. L. Schumaker (2007), *Spline functions: basic theory*, 3. Aufl., Cambridge Mathematical Library, Cambridge University Press, Cambridge, DOI: 10.1017/cbo9780511618994.

L. L. Schumaker (2015), *Spline functions*, Computational methods, Society for Industrial & Applied Mathematics, Philadelphia, PA, DOI: 10.1137/1.9781611973907.ch1.

9 TRIGONOMETRISCHE INTERPOLATION UND FFT

Für die Interpolation von Werten, die durch periodische Vorgänge wie zum Beispiel Schwingungen erzeugt werden, ist es sinnvoll, ebenfalls periodische Funktionen zu verwenden. Wir nehmen der Einfachheit an, dass die Periodenlänge 2π ist, also $f(x) = f(x+2\pi)$ für alle x gilt. Besonders einfach wird die Aufgabe dann, wenn wir unsere Interpolationspolynome auf dem *komplexen Einheitskreis* betrachten. Dazu verwenden wir die Variablentransformation $z : [0, 2\pi) \to \mathbb{C}$, $x \mapsto e^{ix}$ und definieren das *trigonometrische Polynom*

$$p_n(z(x)) = \sum_{k=0}^{n} c_k z(x)^k = \sum_{k=0}^{n} c_k \left(e^{ix}\right)^k = \sum_{k=0}^{n} c_k e^{ikx}.$$

Dabei ist es wegen der Periodizität – im Unterschied zu bisher – vorteilhaft, n (statt $n+1$) gleichverteilte Stützstellen zu verwenden. Wir betrachten also folgendes *trigonometrisches Interpolationsproblem*:

Gegeben: $x_j = 2\pi\frac{j}{n}$ und $f_j := f(x_j) \in \mathbb{C}$ für $j = 0, \ldots, n-1$

Gesucht: $c_k \in \mathbb{C}$ für $k = 0, \ldots, n-1$, so dass gilt

$$\sum_{k=0}^{n-1} c_k e^{2\pi i k \frac{j}{n}} = f_j, \qquad j = 0, \ldots, n-1.$$

Dies sind n Bedingungen von denen wir hoffen können, dass sie die n unbekannten Koeffizienten eindeutig festlegen. Tatsächlich können wir eine geschlossene Formel für diese finden.

Satz 9.1. *Das trigonometrische Interpolationsproblem besitzt die eindeutige Lösung*

$$c_k = \frac{1}{n} \sum_{j=0}^{n-1} f_j e^{-2\pi i \frac{jk}{n}}, \qquad k = 0, \ldots, n-1. \tag{9.1}$$

Beweis. Wir schreiben das Interpolationsproblem in Matrixform

$$Mc = f,$$

mit $c = (c_0, \dots, c_{n-1})^T \in \mathbb{C}^n$, $f = (f_0, \dots, f_{n-1})^T \in \mathbb{C}^n$, und

$$M = (m_{kj})_{k,j=0}^{n-1} \in \mathbb{C}^{n\times n}, \qquad m_{kj} = e^{2\pi i \frac{kj}{n}},$$

und zeigen, dass M invertierbar ist. Dazu betrachten wir die adjungierte Matrix $\overline{M^T}$, d. h. die transponierte Matrix mit den komplex konjugierten Einträgen von M. Wegen $\overline{e^{ikx}} = e^{-ikx}$ besitzt diese die Einträge

$$(\overline{M^T})_{kj} = \overline{m_{jk}} = e^{-2\pi i \frac{jk}{n}}.$$

Wir betrachten nun die Einträge des Produkts $\overline{M^T}M$,

$$\begin{aligned}(\overline{M^T}M)_{kj} &= \sum_{l=0}^{n-1} \overline{m_{lk}} m_{lj} = \sum_{l=0}^{n-1} e^{-2\pi i \frac{lk}{n}} e^{2\pi i \frac{lj}{n}} = \sum_{l=0}^{n-1} e^{-2\pi i l \frac{(k-j)}{n}} \\ &= \begin{cases} \sum_{l=0}^{n-1} 1 = n & \text{falls } k = j, \\ \sum_{l=0}^{n-1} \left(e^{-2\pi i \frac{(k-j)}{n}}\right)^l = \frac{1-e^{-2\pi i (k-j)}}{1-e^{-2\pi i \frac{(k-j)}{n}}} & \text{sonst,} \end{cases}\end{aligned}$$

wobei wir im zweiten Fall die geometrische Summenformel $\sum_{k=0}^{n-1} q^k = \frac{1-q^n}{1-q}$ verwendet haben. Der letzte Bruch verschwindet aber, da gilt

$$e^{-2\pi i (k-j)} = \left(e^{-\pi i}\right)^{2(k-j)} = \left((-1)^2\right)^{k-j} = 1.$$

Also gilt $\overline{M^T}M = nI$, d. h. die eindeutige Lösung der Interpolationsaufgabe ist gegeben durch

$$c = \frac{1}{n}\overline{M^T} f,$$

was komponentenweise genau (9.1) ergibt. □

Die Abbildung $f \mapsto c$ bezeichnet man als *diskrete Fouriertransformation* (DFT). Mit ihrer Hilfe kann man zum Beispiel periodisch abgetastete Signale in ihre Grundschwingungen zerlegen (c_k gibt dabei die Amplitude der Grundschwingung mit der Frequenz k an), was fundamental für die gesamte digitale Signalverarbeitung (und damit die mathematische Basis für CDs, MP3-Musikkompression, JPEG-Bildkompression, Kernspintomographie, etc.) ist. Die *inverse DFT* entspricht damit der Polynomauswertung

$$f_j = \sum_{k=0}^{n-1} c_k e^{2\pi i \frac{jk}{n}}, \qquad j = 0, \dots, n-1. \tag{9.2}$$

9.1 SCHNELLE FOURIERTRANSFORMATION

Ein Nachteil der diskreten Fouriertransformation ist ihr Aufwand von $O(n^2)$ (entsprechend einem Matrix-Vektor-Produkt). Bei einer Anzahl von 10^6 gegebenen Werten[1] benötigt

[1] was etwa 20 Sekunden Musik in CD-Qualität entspricht

ein Prozessor, der eine Milliarde Operationen pro Sekunde schafft, bereits 17 Minuten. Die Berechnung kann aber mit der *schnellen Fouriertransformation* (FFT, "fast Fourier transform")[2] wesentlich beschleunigt werden.

Der klassische Ansatz geht davon aus, dass n gerade ist. Dann lässt sich die Symmetrie der *Einheitswurzeln*

$$\omega_n := e^{-2\pi i \frac{1}{n}}$$

(d. h. $\omega_n^n = 1$) ausnutzen, um den Berechnungsaufwand zu halbieren. Es gilt nämlich offensichtlich

$$\omega_{2n}^2 = \left(e^{-2\pi i \frac{1}{2n}}\right)^2 = e^{-2\pi i \frac{2}{2n}} = \omega_n = \omega_{n/2}^{1/2}.$$

Wir zerlegen jetzt die Summe in (9.1) in je eine Summe für die geraden und für die ungeraden j, die wir mit $j = 2j'$ beziehungsweise $j = 2j' + 1$ durchzählen:

$$\begin{aligned} nc_k &= \sum_{j \text{ gerade}} \omega_n^{kj} f_j + \sum_{j \text{ ungerade}} \omega_n^{kj} f_j \\ &= \sum_{j'=0}^{n/2-1} \omega_n^{k2j'} f_{2j'} + \sum_{j'=0}^{n/2-1} \omega_n^{k(2j'+1)} f_{2j'+1} \\ &= \sum_{j'=0}^{n/2-1} \omega_{n/2}^{kj'} f_{2j'} + \omega_n^k \sum_{j'=0}^{n/2-1} \omega_{n/2}^{kj'} f_{2j'+1} \\ &=: c'_k + \omega_n^k c''_k. \end{aligned}$$

Für $k \le \frac{n}{2} - 1$ sind die c'_k gerade die DFT zu den $\frac{n}{2}$ Werten $f_0, f_2, \dots, f_{n-2}$, die c''_k diejenige zu den $\frac{n}{2}$ Werten $f_1, f_3, \dots, f_{n-1}$. Für diese k lassen sich die Koeffizienten c_k also durch zwei DFTs der Länge $\frac{n}{2}$ berechnen. Die Arbeitsersparnis entsteht dadurch, dass die restlichen c_k auch mit Hilfe der c'_k, c''_k ausgedrückt werden können. Ist nämlich $k \ge \frac{n}{2}$, also $k = \frac{n}{2} + k'$, $0 \le k' \le \frac{n}{2} - 1$, so gilt

$$\begin{aligned} \omega_n^{2jk} &= \omega_n^{2jk'+jn} = \omega_n^{2jk'} (\omega_n^n)^j = \omega_n^{2jk'}, \qquad j = 0, \dots, \frac{n}{2} - 1, \\ \omega_n^k &= \omega_n^{k'+n/2} = \omega_n^{k'} \omega_n^{n/2} = \omega_n^{k'} e^{-\pi i} = -\omega_n^{k'}. \end{aligned}$$

Für $k = k' + \frac{n}{2} \ge \frac{n}{2} - 1$ haben wir daher

$$\begin{aligned} nc_k &= \sum_{j'=0}^{n/2-1} \omega_n^{k2j'} f_{2j'} + \omega_n^k \sum_{j'=0}^{n/2-1} \omega_n^{k2j'} f_{2j'+1} \\ &= \sum_{j'=0}^{n/2-1} \omega_n^{k'2j'} f_{2j'} - \omega_n^{k'} \sum_{j'=0}^{n/2-1} \omega_n^{k'2j'} f_{2j'+1} \\ &= c'_{k'} - \omega_n^{k'} c''_{k'}. \end{aligned}$$

[2] Publiziert 1965 von James Cooley und John Tukey in einem Artikel, der immer noch zu den am öftesten zitierten mathematischen Artikeln zählt. Das Verfahren war aber – wie so oft – bereits Gauß bekannt.

Der Aufwand $A(n)$ für eine DFT der Länge n entspricht also (bei Vorberechnung der Einheitswurzeln) dem von zweier DFTs halber Länge sowie n Additionen, $\frac{n}{2}$ Multiplikationen und n Divisionen:

$$A(n) = 2A\left(\frac{n}{2}\right) + \frac{5}{2}n.$$

Ist nun $\frac{n}{2}$ wieder gerade, lassen sich die DFTs der Länge $\frac{n}{2}$ wieder durch zwei DFTs halber Länge ausdrücken. Dies lässt sich m mal wiederholen, wenn $n = 2^m$ eine Zweierpotenz ist, bis man bei der trivialen DFT der Länge 1, offensichtlich gegeben durch $c_0 = f_0$, ankommt (siehe Algorithmus 9.1). Für die rekursive Berechnung der DFT erhält man so per Induktion den Aufwand

$$A(n) = O(nm) = O(n \log_2(n)).$$

Auf diese Weise lässt sich eine DFT der Länge 10^6 in $A(10^6) \approx 10^7$ Operationen durchführen, was bei einer Milliarde Operationen pro Sekunde etwa 0.1 Sekunden benötigt.

Algorithmus 9.1 : FFT

Input : $f_0, \ldots, f_{n-1}$, $n = 2^m$

1 **if** $n = 1$ **then**
2 $\quad c_0 \leftarrow f_0$
3 $a_0, \ldots, a_{\frac{n}{2}} \leftarrow \mathrm{FFT}(f_0, f_2, \ldots, f_{n-2})$
4 $b_0, \ldots, b_{\frac{n}{2}} \leftarrow \mathrm{FFT}(f_1, f_3, \ldots, f_{n-1})$
5 **for** $k = 0, \ldots, \frac{n}{2} - 1$ **do**
6 $\quad c_k \leftarrow (a_k + e^{-2\pi ik/n} b_k)/n$
7 $\quad c_{k+\frac{n}{2}} \leftarrow (a_k - e^{-2\pi ik/n} b_k)/n$

Output : $c_0, \ldots, c_{n-1}$

Auch die inverse DFT kann durch Algorithmus 9.1 berechnet werden: Vergleicht man (9.2) mit (9.1) und verwendet $\overline{e^{ikx}} = e^{-ikx}$, so erhält man die Beziehung

$$\frac{1}{n}\overline{f_j} = \frac{1}{n}\sum_{k=0}^{n-1} \overline{c_k} e^{-2\pi i \frac{jk}{n}},$$

d. h.

$$f = n\,\overline{FFT(\overline{c})}.$$

9.2 REELLE TRIGONOMETRISCHE INTERPOLATION

Sollen reelle Funktionswerte f_j interpoliert werden, möchte man in der Regel auch ein reelles Interpolationspolynom finden. Als periodische Ansatzfunktionen kommen Sinus und Kosinus in Frage. Wegen der Beziehung $e^{ix} = \cos(x) + i\sin(x)$ können wir auch die Koeffizienten des reellen trigonometrischen Interpolationspolynoms mit Hilfe der FFT berechnen.

Die Idee lässt sich besonders einfach darstellen, wenn $n = 2p+1$ ungerade ist. Dann können wir die Koeffizienten des komplexen Interpolationspolynoms wie folgt umnummerieren:

$$\begin{array}{ccccccccc} c_0, & , \ldots , & c_p, & c_{p+1}, & , \ldots , & c_{n-1} \\ \downarrow & & \downarrow & \downarrow & & \downarrow \\ c_0 & , \ldots , & c_p, & c_{-p} & , \ldots , & c_{-1} \end{array}$$

Da analog $e^{2\pi i \frac{n-1}{n}} = e^{-2\pi i \frac{1}{n}}, \ldots, e^{2\pi i \frac{p+1}{n}} = e^{-2\pi i \frac{p}{n}}$ gilt,[3] können wir die Interpolationsbedingungen schreiben als

$$f(x_j) = \sum_{k=-p}^{p} c_k e^{ikx_j}.$$

Wenn jetzt $f(x_j) \in \mathbb{R}$ für alle $j = 0, \ldots, n-1$ ist, so müssen sich die Imaginärteile auf der rechten Seite zu Null summieren. Wegen $\overline{e^{ikx}} = e^{-ikx}$ bedeutet das, dass $\overline{c_k} = c_{-k}$ (und insbesondere $c_0 \in \mathbb{R}$) sein muss.

Schreiben wir $c_k = c_k^r + ic_k^i$, erhalten wir damit (durch die Symmetrie von Sinus und Kosinus und der Tatsache, dass die Imaginärteile verschwinden müssen)

$$\begin{aligned} f(x_j) &= c_0 + \sum_{k=1}^{p} \left(c_k e^{ikx_j} + \overline{c_k} e^{-ikx_j} \right) \\ &= c_0 + \sum_{k=1}^{p} \left((c_k^r + ic_k^i)(\cos(kx_j) + i\sin(kx_j) + (c_k^r - ic_k^i)(\cos(kx_j) - i\sin(kx_j)) \right) \\ &= c_0 + \sum_{k=1}^{p} \left(2c_k^r \cos(kx_j) - 2c_k^i \sin(kx_j) \right). \end{aligned}$$

Damit haben wir gezeigt, dass das eindeutige reelle Interpolationspolynom eine Linearkombination der (linear unabhängigen) $\left\{1, \cos(kx), \sin(kx) \ : \ k = 1, \ldots, \frac{n-1}{2}\right\}$ ist, und dass die Koeffizienten mit Hilfe der komplexen FFT berechnet werden können.

WEITERFÜHRENDE LITERATUR

C. Van Loan (1992), *Computational frameworks for the fast Fourier transform*, Bd. 10, Frontiers in Applied Mathematics, Society for Industrial & Applied Mathematics (SIAM), Philadelphia, PA.

[3] Es ist egal, ob wir $n-k$ Schritte gegen den Uhrzeigersinn oder k Schritte im Uhrzeigersinn gehen.

10 NUMERISCHE INTEGRATION

Die numerische Berechnung bestimmter Integrale ist eine der ältesten Aufgaben der Mathematik: Sie liegt der Bestimmung des Inhalts krummliniger Flächen zugrunde. Das bekannteste solche Problem ist die Quadratur des Kreises (d. h. die Bestimmung der Seitenlänge desjenigen Quadrats, das den selben Flächeninhalt wie ein Kreis gegebenen Durchmessers hat). Aus diesem Grund wird die numerische Integration auch heute noch *Quadratur* genannt. Sie kommt auch zum Einsatz, falls eine Funktion integriert werden soll, für die eine Stammfunktion nicht bekannt oder schwierig auszuwerten ist. Als dritte, heutzutage wohl wichtigste, Anwendung ist wieder die numerische Lösung von Differentialgleichungen zu nennen. So kann zum Beispiel die Gleichung $y'(x) = f(y, x)$ äquivalent ausgedrückt werden als $y(x) = \int f(y, x)\, dx + C$. Wir betrachten daher folgende Aufgabe: Gegeben sei eine Funktion $f \in C([a, b])$ für $a < b \in \mathbb{R}$; gesucht ist ein Näherungswert $\tilde{I}$ für $I(f) := \int_a^b f(x)\, dx$.

Dabei berechnen wir diese Näherung wieder aus Funktionswerten an vorgegebenen Stützstellen.

Definition 10.1. Für *Stützstellen* $x_0, \ldots, x_n \in [a, b]$ und *Gewichte* $w_0, \ldots, w_n \in \mathbb{R}$ heißt

$$I_n(f) := \sum_{j=0}^{n} w_j f(x_j)$$

Quadraturformel (der Ordnung n). Wir nennen sie *exakt vom Grad m*, falls gilt:

$$I_n(q) = \int_a^b q(x)\, dx \quad \text{für alle } q \in P_m.$$

Wir fordern also, dass Polynome vom Höchstgrad m durch die Quadraturformel exakt integriert werden. Dies ist wichtig bei der numerischen Integration von polynomialen Ansatzfunktionen, etwa in der Methode der Finiten Elemente, erlaubt aber in Kombination mit der Taylorentwicklung auch Aussagen über die Genauigkeit für andere Funktionen.

10.1 INTERPOLATIONSQUADRATUR: NEWTON–COTES-FORMELN

Eine nahe liegende Idee ist, die Stützstellen zur Bestimmung eines Interpolationspolynom zu verwenden, welches dann exakt integriert werden kann. Zu den Stützstellen $x_0, \dots, x_n$ betrachten wir also das Interpolationspolynom in Lagrangeform

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

und damit erhalten wir die Quadraturformel

$$I_n(f) := \int_a^b p_n(x)\,dx = \sum_{j=0}^{n} f(x_j) \int_a^b l_{jn}(x)\,dx.$$

Die Lagrangepolynome l_{jn} sind aus (7.1) bekannt; diese Integrale können also einmal berechnet und anschließend als Gewichte $w_j := \int_a^b l_{jn}(x)\,dx$ verwendet werden. Diese Quadraturformeln sind nach Konstruktion exakt vom Grad n, da wegen der Eindeutigkeit der Polynominterpolation jedes Polynom $q \in P_n$ sein eigenes Interpolationspolynom vom Höchstgrad n oder größer ist.

Für äquidistante Stützstellen mit $a = x_0 < \cdots < x_n = b$ erhält man die *Newton–Cotes-Formeln*.[1] Wir betrachten zuerst die zwei einfachsten Beispiele, und untersuchen deren Exaktheit und Genauigkeit.

Trapezregel ($n = 1$) Der Einfachheit halber betrachten wir $a = 0$, $b = h$ und setzen $x_0 = 0$ und $x_1 = h$. Dann ist

$$I_1(f) = f(0) \int_0^h \frac{x-h}{0-h}\,dx + f(h) \int_0^h \frac{x-0}{h-0}\,dx = \frac{h}{2}(f(0) + f(h)).$$

Die Trapezregel ist exakt vom Grad 1. Für den *Quadraturfehler* folgt aus der Darstellung (7.6) des Interpolationsfehlers

$$\begin{aligned} |I(f) - I_1(f)| = |I(f) - I(p_1)| &= \left| \int_0^h f(x) - p_1(x)\,dx \right| \\ &= \left| \int_0^h \frac{1}{2} f''(\xi(x))(x-0)(x-h)\,dx \right| \le \frac{1}{2}\|f''\|_\infty \left| \int_0^h x(x-h)\,dx \right| \\ &= \frac{1}{2}\|f''\|_\infty \frac{h^3}{6} \end{aligned}$$

falls $f \in C^2([0,h])$. Je kleiner also das Integrationsintervall, desto genauer wird die Quadraturformel.

[1] Roger Cotes (1682–1716) war mit der Vorbereitung der zweiten Ausgabe von Newtons *Principia* betraut. Dabei arbeitete er Newtons Idee zur numerischen Integration aus und publizierte die Gewichte für die Quadraturformeln bis $n = 10$.

Simpson-Regel ($n = 2$) Jetzt setzen wir $x_0 = 0$, $x_1 = \frac{h}{2}$ und $x_2 = h$. Wir erhalten dann analog die Simpson-Regel[2]

$$I_2(f) = \frac{h}{6}\left(f(0) + 4f\left(\frac{h}{2}\right) + f(h)\right).$$

Diese Formel ist sogar exakt vom Grad 3: Ist $q \in P_3$, und $p_2 \in P_2$ dessen Interpolationspolynom, so gilt nach Satz 7.4

$$q(x) - p_2(x) = \frac{q'''}{6}(x-0)\left(x - \frac{h}{2}\right)(x-h),$$

wobei q''' nicht von x abhängt, da q ein kubisches Polynom ist. Der Quadraturfehler erfüllt also

$$I(q) - I_2(q) = \int_a^b \frac{q'''}{6} x\left(x - \frac{h}{2}\right)(x-h) = \frac{q'''}{6}\left[\frac{1}{4}x^4 - \frac{1}{2}hx^3 + \frac{1}{4}h^2x^2\right]_{x=0}^{x=h} = 0,$$

da der letzte Term sowohl für $x = 0$ als auch für $x = h$ verschwindet.

Die höhere Exaktheit spiegelt sich auch im Quadraturfehler wider. Statt des Lagrange-Interpolationspolynoms $p_2 \in P_2$ betrachten wir dafür das Hermite-Interpolationspolynom $q_3 \in P_3$ zu den Stützstellen $t_0 = 0$, $t_1 = t_2 = \frac{h}{2}$, und $t_3 = h$. Wegen der Exaktheit der Simpson-Regel und der Interpolationseigenschaft von q_3 gilt dann ebenfalls

$$I(q_3) = I_2(q_3) = \frac{h}{6}\left(f(t_0) + 4f(t_1) + f(t_3)\right) = I_2(f).$$

Wir verwenden jetzt analog zu oben die Interpolationsfehlerabschätzung für Hermite-Interpolation aus Satz 7.5:

$$\begin{aligned}
|I(f) - I_2(f)| = |I(f) - I(q_3)| &= \left|\int_0^h x(x-\tfrac{h}{2})^2(x-h)\frac{f^{(4)}(\xi(x))}{4!}\,dx\right| \\
&\le \frac{1}{24}\|f^{(4)}\|_\infty \left|\int_0^h x(x-\tfrac{h}{2})^2(x-h)\,dx\right| \\
&= \frac{1}{2880}\|f^{(4)}\|_\infty h^5.
\end{aligned}$$

Diese erhöhte Exaktheit und Fehlerordnung trifft auf alle Newton–Cotes-Formeln mit geradem n zu.

[2] Thomas Simpson (1710–1761) war ein britischer Mathematiker und Autor zahlreicher, zu seiner Zeit beliebter, Lehrbücher. Die heute üblichen Bezeichnungen Sinus, Kosinus, Tangens und Kotangens wurden erstmals von ihm verwendet. Die nach ihm benannte, 1722 publizierte, Formel war allerdings schon Cavalieri (um 1639) bekannt.

Allgemein lauten die Newton–Cotes Formeln für $\int_a^b f(x)\,dx$

$$I_n(f) = (b-a)\sum_{j=0}^{n} w_j f(x_j) \qquad \text{für} \quad x_j = a + \frac{j}{n}(b-a), \quad w_j = \int_a^b l_{jn}(x)\,dx.$$

Die Gewichte und Genauigkeiten der ersten vier Quadraturformeln sind in der folgenden Tabelle gesammelt.

n	Name	w_j	exakt	Fehler
1	Trapez	$\frac{1}{2}, \frac{1}{2}$	1	$\|f''\|_\infty O((b-a)^3)$
2	Simpson	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	3	$\|f^{(4)}\|_\infty O((b-a)^5)$
3	Newtons $\frac{3}{8}$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	3	$\|f^{(4)}\|_\infty O((b-a)^5)$
4	Milne	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	5	$\|f^{(6)}\|_\infty O((b-a)^7)$

Ab $n = 8$ treten allerdings negative Gewichte w_j auf, so dass Auslöschung ein Problem werden kann.

10.2 SUMMIERTE QUADRATURFORMELN

Um eine höhere Genauigkeit zu erreichen ohne Quadraturformeln höherer Ordnung zu verwenden, kann man analog zur Spline-Interpolation das Integral stückweise berechnen.

Wir unterteilen dafür das komplette Integrations-Intervall $[a, b]$ in die N Teilintervalle $[x_0, x_1], [x_1, x_2], \ldots, [x_{N-1}, x_N]$, und wenden auf jedes Intervall eine Quadraturformel an. Definieren wir

$$I\left(f; [x_j, x_{j+1}]\right) := \int_{x_j}^{x_{j+1}} f(x)\,dx$$

und

$$I_n\left(f; [x_j, x_{j+1}]\right) := (x_{j+1} - x_j)\sum_{k=0}^{n} w_{j,k} f(t_{j,k})$$

(wobei $t_{j,k}$ die Stützstellen im Intervall $[x_j, x_{j+1}]$ und $w_{j,k}$ die entsprechenden Gewichte sind), so ist unsere *summierte Quadraturformel* gegeben durch

$$I_n^N(f) := \sum_{j=0}^{N-1} I_n\left(f; [x_j, x_{j+1}]\right).$$

Für jede Newton–Cotes-Formel kann man solch eine summierte Quadraturformel aufstellen. Wir betrachten zuerst als einfaches Beispiel die *summierte Trapezregel*. Wir nehmen wieder

an, dass alle Intervalle die gleiche Länge haben, d. h. $x_{j+1} - x_j = h$ für alle $0 \le j < N$. Dann ist

$$I_1\left(f; [x_j, x_{j+1}]\right) = \frac{h}{2}\left(f(x_j) + f(x_{j+1})\right),$$

und deshalb

$$I_1^N(f) = \sum_{j=0}^{N-1} \frac{h}{2}\left(f(x_j) + f(x_{j+1})\right) = h\left(\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{N-1}) + \frac{1}{2}f(x_N)\right).$$

Da lineare Funktionen auf jedem Intervall linear sind, hat die summierte Formel offensichtlich die gleiche Exaktheit wie die zugrunde liegende Newton–Cotes-Formel. Für den Gesamtfehler gilt wegen $(b-a) = Nh$

$$\begin{aligned}|I(f) - I_1^N(f)| &\le \sum_{j=0}^{N-1} |I_1\left(f; [x_j, x_{j+1}]\right) - I\left(f; [x_j, x_{j+1}]\right)| \le \sum_{j=0}^{N-1} \frac{h^3}{12}\|f''\|_\infty \\ &= \frac{h^3}{12}\|f''\|_\infty N = \frac{h^2}{12}\|f''\|_\infty (b-a).\end{aligned}$$

Durch die Summation verliert man also eine Fehlerordnung gegenüber der einfachen Quadraturformel; dafür kann man h (und damit den Fehler) durch feinere Unterteilung beliebig klein machen, ohne negative Gewichte zu bekommen.

Für Quadraturformeln höherer Ordnung benötigt man mehr Stützstellen, so dass ein Intervall nicht ausreicht. So muss man etwa für die summierte Simpson-Regel jeweils zwei Intervalle zusammenfassen. Nehmen wir der Einfachheit halber an, dass N gerade ist, so erhält man

$$\begin{aligned}I_2^N(f) &= \sum_{j=0}^{N/2-1} \frac{h}{6}\left(f(x_{2j}) + 4f(x_{2j+1} + f(x_{2j+2})\right) \\ &= \frac{h}{6}\left(f(a) + 4\sum_{j=0}^{N/2-1} f(x_{2j+1}) + 2\sum_{j=1}^{N/2-1} f(x_{2j}) + f(b)\right),\end{aligned}$$

die ebenfalls exakt vom Grad 3 ist. Für den Gesamtfehler folgt durch Summation der einzelnen Quadraturfehler auf jedem der $\frac{N}{2}$ Teilintervallen der Länge $\frac{b-a}{N/2} = 2h$

$$\begin{aligned}|I(f) - I_2^N(f)| &\le \sum_{j=0}^{N/2-1} |I_2\left(f; [x_{2j}, x_{2j+2}]\right) - I\left(f; [x_{2j}, x_{2j+1}]\right)| \\ &\le \sum_{j=0}^{N/2-1} \left(\frac{b-a}{N/2}\right)^5 \frac{1}{2880}\|f^{(4)}\|_\infty \\ &= \frac{h^4}{180}\|f^{(4)}\|_\infty (b-a).\end{aligned}$$

10.3 GAUSS-QUADRATUR

Die summierten Quadraturformeln erreichen zwar bei vergleichsweise wenig Aufwand eine beliebig hohe Genauigkeit, in manchen Anwendungen ist es jedoch wichtig, dass Polynome vorgegebenen Grades exakt integriert werden. Dies erfordert aber eine Quadraturformel entsprechend hohen Grades.[3] Um negative Gewichte zu vermeiden, sind äquidistante Stützstellen also nicht geeignet (man vergleiche das Runge-Phänomen bei der Polynominterpolation). Wir fragen uns also, welche Stützstellen (und Gewichte) wir verwenden sollten, um Polynome möglichst hohen Grades exakt zu integrieren.

Da wir nun für eine Quadraturformel I_n die $n+1$ Gewichte und $n+1$ Stützstellen (also $2n+2$ Freiheitsgrade) optimal wählen können, liegt die Vermutung nahe, dass wir Polynome vom Grad $2n+1$ exakt integrieren können. Das ist auch der höchstmögliche Grad: Angenommen, wir könnten mit den $n+1$ Stützstellen $x_0, \dots, x_n$ Polynome vom Grad $2n+2$ exakt integrieren. Dann gilt dies insbesondere für das Polynom

$$q(x) := (x - x_0)^2 \dots (x - x_n)^2 \in P_{2n+2}.$$

Da $q(x) > 0$ für $x \neq x_j$ gilt, ist wegen der Monotonie des Integrals auch

$$0 < \int_a^b q(x) = \sum_{j=0}^{n} w_j q(x_j) = 0,$$

was einen Widerspruch darstellt.

Wir suchen also Stützstellen $x_0, \dots, x_n \in [a, b]$ und Gewichte $w_0, \dots, w_n \in \mathbb{R}$ so, dass gilt

$$I_n(q) := \sum_{j=0}^{n} w_j q(x_j) = \int_a^b q(x)\,dx \quad \text{für alle } q \in P_{2n+1}.$$

Um die Bedingungen herzuleiten, die die Stützstellen erfüllen müssen, definieren wir wieder das Stützstellenpolynom $\omega_n(x) := (x - x_0) \dots (x - x_n) \in P_{n+1}$. Die Polynomdivision zeigt, dass jedes Polynom $q \in P_{2n+1}$ geschrieben werden kann als

$$q = \omega_n p + r \quad \text{mit } p, r \in P_n.$$

Setzen wir dies ein, erhalten wir

$$I(q) = \int_a^b \omega_n(x) p(x)\,dx + \int_a^b r(x)\,dx,$$
$$I_n(q) = \sum_{j=0}^{n} w_j(\omega_n(x_j) p(x_j)) + \sum_{j=0}^{n} w_j r(x_j).$$

[3] Freilich spricht nichts dagegen, die Ansätze dieses und des vorherigen Abschnitts zu kombinieren.

Nun ist $\omega_n(x_j) = 0$ für alle $0 \le j \le n$, der erste Term von $I_n(p)$ verschwindet also. Solange die Stützstellen $x_0, \dots, x_n$ paarweise verschieden sind, können wir wie in Abschnitt 10.1 die Gewichte $w_0, \dots, w_j$ so bestimmen, dass die Quadraturformel I_n (mindestens) exakt vom Grad n ist. Da $r \in P_n$ ist, gilt dann

$$I_n(r) = \sum_{j=0}^{n} w_j r(x_j) = \int_a^b r(x)\,dx.$$

Dies legt die Gewichte fest; für die Stützstellen erhalten wir damit die Bedingung

$$I(q) - I_n(q) = \int_a^b \omega_n(x)p(x)\,dx = 0 \qquad \text{für alle } q \in P_{2n+1}.$$

Anders gesagt müssen also die Stützstellen $x_0, \dots, x_n$ Nullstellen eines Polynoms ω_n sein, für das gilt

$$\int_a^b \omega_n(x)p(x)\,dx = 0 \qquad \text{für alle } p \in P_n. \tag{10.1}$$

Solche Polynome nennt man *Orthogonalpolynome*; aus der Linearen Algebra ist nämlich bekannt, dass $\langle p, q\rangle := \int_a^b p(x)q(x)\,dx$ ein Skalarprodukt definiert auf dem Raum P_n für $n \in \mathbb{N}$ beliebig (und sogar dem Raum aller stetigen Funktionen). Wir können ein Orthogonalpolynom also mit Hilfe des Gram–Schmidt-Verfahrens angewendet auf die Basis $\{1, x, \dots, x^{n+1}\}$ von P_{n+1} berechnen, d. h. wir setzen

$$p_0 := 1, \qquad p_{m+1} := x^{m+1} - \sum_{k=0}^{m} \frac{\langle x^{m+1}, p_k\rangle}{\langle p_k, p_k\rangle} p_k, \quad m = 0, \dots, n. \tag{10.2}$$

Dann ist $p_{n+1} \in P_{n+1}$ nach Konstruktion orthogonal zum $\operatorname{span}\{p_0, \dots, p_n\} = \operatorname{span}\{1, \dots, x^n\} = P_n$ und hat außerdem den führenden Koeffizienten 1, d. h. $\omega_n(x) = x^{n+1} + O(x^n)$. Tatsächlich muss man in jedem Schritt nur gegen die jeweils letzten beiden Polynome orthogonalisieren, was eine effiziente Berechnung auch für große n ermöglicht.

Lemma 10.2. *Die Polynome p_m aus (10.2) erfüllen die* Drei-Term-Rekursion

$$\begin{aligned} p_0 &= 1, \qquad p_1 = x - \beta_0, \\ p_{m+1} &= (x - \beta_m)p_m - \gamma_m p_{m-1}, \qquad m = 1, \dots, n, \end{aligned}$$

für

$$\beta_m := \frac{\langle xp_m, p_m\rangle}{\langle p_m, p_m\rangle}, \qquad \gamma_m := \frac{\langle p_m, p_m\rangle}{\langle p_{m-1}, p_{m-1}\rangle}.$$

Beweis. Wir verwenden Induktion nach m. Die Aussage ist klar für $m = 0, 1$. Sei also $m \ge 1$ und bezeichne p_k die Polynome aus (10.2) sowie q_k die Polynome aus der Drei-Term-Rekursion. Es sei $q_k = p_k$ für alle $k \le m$. Dann folgt, dass sowohl p_{m+1} als auch q_{m+1} ein

Polynom vom Grad $m+1$ mit führendem Koeffizienten 1 ist. Also ist $r := p_{m+1} - q_{m+1} \in P_m$. Es genügt also zu zeigen, dass r orthogonal auf P_m steht, denn der einzige Vektor aus P_m, der dies erfüllt, ist der Null-Vektor. Da p_{m+1} nach Konstruktion orthogonal zu P_m ist, müssen wir das nur noch für q_{m+1} zeigen. Dies tun wir in drei Schritten:

- Für $p_m \in P_m$ gilt

$$\begin{aligned}\langle q_{m+1}, p_m\rangle &= \langle xp_m, p_m\rangle - \beta_m\langle p_m, p_m\rangle - \gamma_m\langle p_{m-1}, p_m\rangle \\ &= \langle xp_m, p_m\rangle - \beta_m\langle p_m, p_m\rangle = 0\end{aligned}$$

  wegen der Definition von β_m und $\langle p_m, p_{m-1}\rangle = 0$ nach Konstruktion.

- Analog gilt für $p_{m-1} \in P_{m-1}$

$$\begin{aligned}\langle q_{m+1}, p_{m-1}\rangle &= \langle xp_m, p_{m-1}\rangle - \beta_m\langle p_m, p_{m-1}\rangle - \gamma_m\langle p_{m-1}, p_{m-1}\rangle \\ &= \langle xp_{m-1}, p_m\rangle - \langle p_m, p_m\rangle \\ &= \langle xp_{m-1} - p_m, p_m\rangle = 0,\end{aligned}$$

  wobei wir im ersten Schritt die Definition des Skalarprodukts als Integral, die Tatsache, dass p_m nach Konstruktion orthogonal zu p_{m-1} ist, sowie die Definition von γ_m, und im zweiten Schritt die Tatsache, dass xp_{m-1} ebenso wie p_m ein Polynom vom Grad m mit führendem Koeffizienten 1 ist und damit p_m nach Konstruktion orthogonal ist zu $xp_{m-1} - p_m \in P_{m-1}$, verwendet haben.

- Schließlich gilt für beliebige $q \in P_{m-2}$

$$\langle q_{m+1}, q\rangle = \langle p_m, xq\rangle - \beta_m\langle p_m, q\rangle - \gamma_m\langle p_{m-1}, q\rangle = 0,$$

  da p_m orthogonal ist sowohl zu q als auch zu $xq \in P_{m-1}$. □

Für die Stützstellen müssen wir nun die Nullstellen von p_{n+1} bestimmen, was in der Regel nur numerisch möglich ist. Immerhin können wir garantieren, dass tatsächlich stets $n+1$ verschiedene reelle Nullstellen im Intervall $[a, b]$ existieren, was ja alles andere als offensichtlich ist.

Lemma 10.3. *Das Polynom $p_{n+1} \in P_{n+1}$ aus (10.2) hat $n+1$ einfache Nullstellen $x_0, \dots, x_n \in (a, b)$.*

Beweis. Angenommen, p_{n+1} hat $m+1 < n+1$ Nullstellen $a < x_0 < \cdots < x_m < b$ mit ungerader Vielfachheit. Dann ändert p_{n+1} genau $(m+1)$-mal das Vorzeichen. Genau den selben Vorzeichen*wechsel* hat aber auch $q := (x-x_0)\cdots(x-x_m) \in P_{m+1}$. Also hat $qp_{n+1} \neq 0$ auf ganz (a, b) ein einziges Vorzeichen, und damit gilt

$$0 \neq \int_a^b q(x)p_{n+1}(x)\,dx = \langle q, p_{n+1}\rangle = 0,$$

da p_{n+1} nach Konstruktion und Annahme $m+1 < n+1$ orthogonal ist zu $q \in P_{m+1}$. Da $p_{n+1} \in P_{n+1}$ maximal $n+1$ verschiedene Nullstellen haben kann, muss also p_{n+1} genau $n+1$ einfache Nullstellen haben. □

Weiter kann man zeigen, dass die Gewichte stets positiv sind.

Wir können also die gewünschte Quadraturformel durch folgendes Vorgehen konstruieren:

1. Bestimme p_{n+1} durch die Drei-Term-Rekursion aus Lemma 10.2.
2. Bestimme Nullstellen $x_0, \dots, x_n$ von p_{n+1}.
3. Bestimme Gewichte $w_j = \int_a^b \ell_{jn}(x)\,dx$ für die Lagrange-Polynome zu den Stützstellen $x_0, \dots, x_n$.

Wir halten also fest:

Satz 10.4. *Zu $n \in \mathbb{N}$ und $a, b \in \mathbb{R}$ existieren $x_0, \dots, x_n \in [a,b]$ und $w_0, \dots, w_n \in \mathbb{R}$, so dass die zugehörige Interpolationsquadraturformel exakt ist vom Grad $2n+1$.*

Aus der Exaktheit kann man wieder schliessen, dass der Quadraturfehler (bei genügender Differenzierbarkeit von f) von der Ordnung $O((b-a)^{2n+3})$ ist. Dafür verwenden wir wie bei der Simpson-Regel das Hermite-Interpolationspolynom q_{2n+1} zu den Interpolationsbedingungen

$$q_{2n+1}(x_j) = f(x_j), \qquad q'_{2n+1}(x_j) = f'(x_j), \qquad j = 0, \dots, n,$$

für das wegen der Exaktheit gilt

$$I(q_{2n+1}) = I_n(q_{2n+1}) = I_n(f).$$

Aus der Interpolationsfehlerdarstellung aus Satz 7.5 folgt dann

$$\begin{aligned} |I_n(f) - I(f)| &\le \frac{1}{(2n+2)!} \|f^{(2n+2)}\|_\infty \left| \int_a^b (x-x_0)^2 \dots (x-x_n)^2 \, dx \right| \\ &= O\left((b-a)^{2n+3}\right) \end{aligned}$$

für $f \in C^{2n+2}([a,b])$.

Diese Quadraturformeln nennt man *Gauß-Formeln*. Für das Intervall $[a,b] = [-1,1]$ erfüllen die so genannten *Legendre-Polynome* die Orthogonalitätsbedingung (10.1); ihre Nullstellen lassen sich in der Regel nur numerisch berechnen. Die ersten drei sind aber exakt bekannt und lauten

n	x_j	w_j	exakt
0	0	2	1
1	$-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}$	1, 1	3
2	$-\sqrt{\frac{3}{5}}, 0, \sqrt{\frac{3}{5}}$	$\frac{5}{9}, \frac{8}{9}, \frac{5}{9}$	5

Formeln für beliebige Intervalle $[a, b]$ erhält man daraus durch Transformation auf das Intervall $[-1, 1]$ mit Hilfe der Substitutionsformel.

WEITERFÜHRENDE LITERATUR

G. Hämmerlin & K.-H. Hoffmann (1994), *Numerische Mathematik*, 4. Aufl., Springer-Verlag, Berlin.

11 NICHTLINEARE GLEICHUNGEN

Viele mathematische Modelle aus Physik (Mehrkörperprobleme in der Mechanik), Biologie (Populationswachstum) und Chemie (Reaktionskinetik) sind nicht linear. Wir betrachten daher nun die Aufgabe, für eine gegebene Funktion $f : [a, b] \to \mathbb{R}$ ein $x^* \in [a, b]$ zu bestimmen mit $f(x^*) = 0$. Wie das Beispiel $f(x) = x^5 - x + 1$ zeigt, kann es im Allgemeinen kein direktes Verfahren für die Lösung dieses Problems geben; wir müssen wieder iterative Verfahren betrachten.

Eine einfache Methode für skalare Funktionen $f : \mathbb{R} \to \mathbb{R}$ ist das *Bisektionsverfahren*, das auf der Tatsache beruht, dass eine stetige Funktion zwischen zwei Punkten mit unterschiedlichem Vorzeichen eine Nullstelle haben muss. Dieses Intervall, in dem die Nullstelle liegen muss, wird nun wiederholt halbiert:

Algorithmus 11.1 : Bisektionsverfahren

```
1 Wähle a_0, b_0 mit x* ∈ [a_0, b_0] und f(a_0)f(b_0) < 0
2 for k = 0, 1, ... do
3     x^k ← (a_k + b_k)/2
4     if f(x^k)f(a_k) < 0 then
5         a_{k+1} ← a_k, b_{k+1} ← x^k
6     else
7         a_{k+1} ← x^k, b_{k+1} ← b_k
```

In jedem Schritt wird dabei die Länge des Intervalls, in dem x^{k+1} und x^* liegen müssen, um den Faktor 2 kleiner; es gilt also mit Induktion

$$(b_k - a_k) = \frac{1}{2}(b_{k-1} - a_{k-1}) = \cdots = \frac{1}{2^k}(b_0 - a_0) \to 0 \qquad \text{für } k \to \infty$$

und damit wegen $x^{k+1}, x^* \in (b_k - a_k)$ auch $|x^k - x^*| \leq \frac{1}{2k} \to 0$. Deshalb konvergiert die Iteration zwar lediglich linear, aber dafür unabhängig davon, wie weit a_0 und b_0 von x^* entfernt liegen. Die Erweiterung für Funktionen mehrerer unabhängiger Variablen ist allerdings nicht offensichtlich.[1]

[1] Allerdings enthalten viele kompliziertere Verfahren als Teilschritt eine *Schrittweitensuche*, die mit dem Bisektionsverfahren durchgeführt werden kann.

11.1 FIXPUNKTITERATION

Eine sehr allgemeine Klasse von Verfahren beruht auf der Formulierung der Nullstellensuche $F(x) = 0$ für $F : \mathbb{R}^n \to \mathbb{R}^n$ als *Fixpunktgleichung* $\Phi(x) = x$ für eine geeignete Funktion $\Phi(x)$. Wählen wir eine invertierbare Matrix $A \in \mathbb{R}^{n\times n}$, so können wir zum Beispiel schreiben

$$F(x^*) = 0 \quad \Leftrightarrow \quad x^* = x^* - AF(x^*) =: \Phi(x^*). \tag{11.1}$$

(Oft sind nichtlineare Modelle, die etwa Gleichgewichtszustände beschreiben, auch direkt als Fixpunktpunktgleichungen gegeben.) Die Lösung dieser Fixpunktgleichung kann nun iterativ durch Fixpunktiteration bestimmt werden, die für beliebige Selbstabbildungen $\Phi : D \to D$ mit $D \subset \mathbb{R}^n$ funktioniert.

Algorithmus 11.2 : Fixpunktiteration

```
1 Wähle x^0 ∈ D
2 for k = 0, 1, ... do
3     x^{k+1} ← Φ(x^k)
```

Wir müssen uns nun fragen, in welchen Fällen diese Fixpunktiteration konvergiert, d. h. wann $\lim_{k\to\infty} x^k = x^*$ gilt. Der Banachsche Fixpunktsatz sagt uns, dass dies der Fall ist, wenn Φ eine *Kontraktion* ist.

Satz 11.1 (Banachscher Fixpunktsatz). *Sei $D \in \mathbb{R}^n$ nichtleer und abgeschlossen. Gilt $\Phi(D) \subset D$ und*

$$\|\Phi(x) - \Phi(y)\| \le L\|x - y\| \quad \text{für alle } x, y \in D \tag{11.2}$$

mit $L < 1$, so konvergiert für jedes $x^0 \in D$ die Fixpunktiteration gegen den (eindeutigen) Fixpunkt x^* von Φ.

Beweis. Die Konvergenz der Fixpunktiteration folgt aus der Tatsache, dass die x^k eine Cauchyfolge bilden. Aus der Iterationsvorschrift und der Bedingung (11.2) folgt nämlich, dass gilt

$$\|x^{k+1} - x^k\| = \|\Phi(x^k) - \Phi(x^{k-1})\| \le L\|x^k - x^{k-1}\| \le \cdots \le L^k\|x^1 - x^0\|.$$

Deshalb ist

$$\begin{aligned}\|x^{m+k} - x^k\| &\le \|x^{m+k} - x^{(m+k-1)} + x^{m+k-1} - \cdots + x^{k+1} - x^k\| \\ &\le \|x^{m+k} - x^{(m+k-1)}\| + \cdots + \|x^{k+1} - x^k\| \\ &\le (L^{m+k} + \cdots + L^k)\|x^1 - x^0\| = L^k \frac{1 - L^m}{1 - L}\|x^1 - x^0\|.\end{aligned}$$

Da $L < 1$ gilt, konvergiert $L^k \to 0$ für $k \to \infty$. Also gilt $\|x^{m+k} - x^k\| \to 0$ für alle $m \in \mathbb{N}$ und $k \to \infty$. Die Cauchyfolge x^k konvergiert daher gegen einen Punkt $x^* \in D$ wegen der Abgeschlossenheit von D.

Dass der Grenzwert x^* ein Fixpunkt von Φ ist, folgt nun daraus, dass Φ wegen (11.2) Lipschitzstetig und damit insbesondere stetig ist:

$$x^* - \Phi(x^*) = x^* - \lim_{k\to\infty} \Phi(x^k) = x^* - \lim_{k\to\infty} x^{k+1} = x^* - x^* = 0.$$

Die Eindeutigkeit erhält man aus der Kontraktionsbedingung (11.2). Seien $x^* = \Phi(x^*)$ und $x^{**} = \Phi(x^{**})$, dann gilt

$$\|x^* - x^{**}\| = \|\Phi(x^*) - \Phi(x^{**})\| \le L\|x^* - x^{**}\|$$

mit $L < 1$. Dies kann aber nur gelten, wenn $\|x^* - x^{**}\| = 0$ ist. □

Für differenzierbare Funktionen kann man die Kontraktionsbedingung (11.2) recht einfach nachprüfen. Hier bezeichnet $\Phi'(x) \in \mathbb{R}^{n\times n}$ die *Jacobi-Matrix* der partiellen Ableitungen von Φ, und $\|\Phi'\|_\infty := \sup_{x\in\mathbb{R}^n} \|\Phi'(x)\|$ für eine beliebige Matrixnorm.

Folgerung 11.2. *Sei $D \in \mathbb{R}^n$ abgeschlossen und konvex. Ist $\Phi \in C^1(D)$ mit $\Phi(D) \subset D$, und gilt $\|\Phi'\|_\infty < 1$, so konvergiert die Fixpunktiteration für jeden Startwert x^0.*

Beweis. Seien $x, y \in D$ beliebig. Da D konvex ist, ist auch $x + t(y - x) \in D$ für alle $t \in [0, 1]$. Für

$$\psi : [0, 1] \to \mathbb{R}^n, \qquad \psi(t) := \Phi(x + t(y - x)),$$

folgt dann aus dem Hauptsatz der Differential- und Integralrechnung

$$\begin{aligned}
\|\Phi(y) - \Phi(x)\| = \|\psi(1) - \psi(0)\| &= \left\| \int_0^1 \psi'(t)\, dt \right\| \\
&= \left\| \int_0^1 \Phi'(x + t(y - x))(y - x)\, dt \right\| \\
&\le \int_0^1 \|\Phi'(x + t(y - x))\| \|y - x\|\, dt \\
&\le \|\Phi'\|_\infty \|x - y\|
\end{aligned}$$

und damit die Behauptung. □

Umgekehrt können wir damit auch entscheiden, wann eine Fixpunktiteration *nicht* konvergiert: Gilt im Fixpunkt $\|\Phi'(x^*)\| > 1$, so folgt aus der Stetigkeit von Φ', dass eine Umgebung

U um x^* existiert, so dass $\|\Phi'(x)\| > 1$ gilt für alle $x \in U$. Dann folgt analog zum Beweis von Folgerung 11.2 aus dem Mittelwertsatz für ψ, dass gilt

$$\|x^{k+1} - x^*\| = \|\Phi(x^k) - \Phi(x^*)\| = \left\| \int_0^1 \Phi'(x + t(x^* - x^k))(x^* - x^k)\, dt \right\| > \|x^k - x^*\|.$$

Der Fehler wird also *größer*, sobald man in die Nähe von x^* kommt. Ein Fixpunkt x^*, für den $\|\Phi'(x^*)\| > 1$ gilt, wird *abstoßend* genannt; die Fixpunktiteration wird niemals gegen solch einen Fixpunkt konvergieren. (Ist $\|\Phi'(x^*)\| = 1$, so kann man keine definitive Aussage treffen; eine Konvergenz ist jedoch in der Regel nicht zu erwarten.)

Falls die Fixpunktiteration konvergiert, ist auch ihre Konvergenzgeschwindigkeit interessant. Dazu benutzen wir den Begriff der Konvergenzordnung aus Definition 1.13. Um die Konvergenzordnung der Fixpunktiteration zu bestimmen, entwickeln wir $\Phi \in C^2(D)$ in eine Taylorreihe um x^*. Dann gilt

$$\Phi(x) = \Phi(x^*) + \Phi'(x^*)(x - x^*) + O(\|x - x^*\|^2) \qquad \text{für } x \to x^*.$$

Speziell für $x = x^k$ folgt daraus

$$\|x^{k+1} - x^*\| = \|\Phi(x^k) - \Phi(x^*)\| \leq \|\Phi'(x^*)\|\|x^k - x^*\| + O(\|x^k - x^*\|^2).$$

Wir halten fest:

(i) Die Fixpunktiteration wird in der Regel nur linear konvergieren.

(ii) Gilt $\Phi'(x^*) = 0$, so wird die Iteration superlinear konvergieren, sobald x^k nahe genug bei x^* liegt. Falls Φ zweimal stetig differenzierbar ist (man also obigen Fehlerterm in der Taylorentwicklung angeben kann), ist die Konvergenz sogar *quadratisch*.

(iii) Ist $\|\Phi'(x^*)\| > 1$, so ist die Funktion Φ für die Fixpunktiteration ungeeignet; es gibt keine Konvergenz.

Beispiel 11.3. Wir berechnen $\sqrt{2}$ als Nullstelle von $f(x) = x^2 - 2$ durch Fixpunktiteration. Jede der folgenden Funktionen hat $\sqrt{2}$ als Fixpunkt:

(i) Setze $\Phi : (0, \infty) \to (0, \infty)$, $\Phi(x) = \frac{x^3}{2}$. Dann ist aber $\Phi'(\sqrt{2}) = 3 > 1$; diese Fixpunktiteration konvergiert also nicht, da $\sqrt{2}$ ein abstoßender Fixpunkt ist.

(ii) Setze $\Phi : [1, 2) \to [1, 2)$, $\Phi(x) = 1 + x - \frac{1}{2}x^2$. Hier ist $\Phi'(x) = 1 - x \in (-1, 0]$ und damit $|\Phi'(x)| < 1$, weshalb die Fixpunktiteration für jedes $x^0 \in [1, 2)$ konvergiert. Da $\Phi'(\sqrt{2}) = 1 - \sqrt{2} \neq 0$ ist, ist die Konvergenz lediglich linear.

(iii) Setze $\Phi : [1, 2] \to [1, 2]$, $\Phi(x) = \frac{1}{2}x + \frac{1}{x}$. Also ist $\Phi'(x) = \frac{1}{2} - \frac{1}{x^2}$, und $\|\Phi'\|_\infty = \frac{1}{2} < 1$. Diese Fixpunktiteration konvergiert deshalb für jeden Startwert $x^0 \in [1, 2]$. Weiterhin ist $\Phi'(\sqrt{2}) = 0$, weshalb das Verfahren sogar quadratisch konvergiert.

Es liegt also auf der Hand, für die Nullstellenbestimmung die Iterationsfunktion Φ so zu wählen, dass $\Phi'(x^*) = 0$ erfüllt ist: Dies führt auf das Newton-Verfahren.

11.2 NEWTON-VERFAHREN

Zur Herleitung des Newton-Verfahrens zur Nullstellenbestimmung einer Funktion $F \in C^1(D)$ greifen wir wieder auf den Ansatz (11.1) zurück:

$$\Phi(x) = x - AF(x),$$

wobei $A \in \mathbb{R}^{n\times n}$ invertierbar sein soll. Wir bestimmen jetzt A so, dass $\Phi'(x^*) = 0$ ist und wir damit superlineare Konvergenz haben, d. h. wir fordern

$$0 = \Phi'(x^*) = I - AF'(x^*).$$

In anderen Worten, es muss gelten

$$A = F'(x^*)^{-1},$$

falls $F'(x^*)$ invertierbar ist. Da F' als stetig vorausgesetzt ist, existiert dann nach dem Störungslemma (Satz 4.16) eine Umgebung U um x^*, so dass $F'(x)$ invertierbar ist für alle $x \in U$. Innerhalb dieser Umgebung können wir also die Funktion

$$\Phi(x) = x - F'(x)^{-1}F(x)$$

betrachten, für die $F(x^*) = 0$ und $\Phi'(x^*) = 0$ für jeden Fixpunkt x^* von Φ gilt. Aus der Stetigkeit von F' folgt weiter, dass es eine Umgebung V von x^* gibt, so dass $\Phi'(x) < 1$ für alle $x \in V \cap U$ gilt. In dieser Umgebung $U \cap V$ konvergiert also die Fixpunktiteration nach Konstruktion superlinear; damit erhalten wir das *Newton-Verfahren*. Natürlich wird man in der Regel die Inverse $F(x)^{-1}$ nie direkt berechnen, sondern stattdessen das zur Fixpunktiteration $x^{k+1} = \Phi(x^k)$ äquivalente Gleichungssystem

$$F'(x^k)(x^{k+1} - x^k) = -F(x^k) \tag{11.3}$$

mit den Verfahren aus Kapitel 5 lösen.

Algorithmus 11.3 : Newton-Verfahren

1 Wähle $x^0 \in \mathbb{R}^n$ "hinreichend nahe" bei x^*
2 **for** $k = 0, 1, \ldots$ **do**
3 Berechne $F(x^k)$, $F'(x^k)$
4 Löse $F'(x^k)s^k = -F(x^k)$
5 $x^{k+1} \leftarrow x^k + s^k$

Beispiel 11.4. Wir betrachten wieder $f(x) = x^2 - 2$. Das Newton-Verfahren kann man hier direkt nach x^{k+1} auflösen und erhält dadurch die Iterationsvorschrift

$$x^{k+1} = x^k - \frac{(x^k)^2 - 2}{2x^k} = x^k - \frac{x^k}{2} + \frac{1}{x^k} = \frac{1}{2}x^k + \frac{1}{x^k},$$

womit wir Beispiel 11.3 (iii) hergeleitet haben.[2]

[2] Dieses Verfahren zur Wurzelberechnung war den Babyloniern schon lange vor Newton bekannt.

Wir zeigen nun die Konvergenz des Newton-Verfahrens.

Satz 11.5. *Sei $F : \mathbb{R}^n \to \mathbb{R}^n$ zweimal stetig differenzierbar in einer Umgebung $U \subset \mathbb{R}^n$ um eine Nullstelle $x^* \in U$ von F, in der $F'(x^*)$ invertierbar ist. Dann konvergiert das Newton-Verfahren lokal quadratisch, d. h. für $x^0 \in U$ hinreichend nahe an x^* gilt $x^k \to x^*$ und existiert eine Konstante $C > 0$ mit*

$$\|x^{k+1} - x^*\| \leq C\|x^k - x^*\|^2 \qquad \text{für alle } k \in \mathbb{N}.$$

Beweis. Zunächst folgt aus der Iterationsvorschrift für das Newton-Verfahren

$$\begin{aligned} x^{k+1} - x^* &= x^k - x^* - F'(x^k)^{-1}F(x^k) \\ &= x^k - x^* - F'(x^k)^{-1}(F(x^k) - F(x^*)) \\ &= F'(x^k)^{-1}\left[F'(x^k)(x^k - x^*) - F(x^k) + F(x^*)\right] \\ &= F'(x^k)^{-1}\left[F(x^*) - F(x^k) - F'(x^k)(x^* - x^k)\right]. \end{aligned}$$

Es genügt also, die beiden Faktoren separat abzuschätzen. Nach Voraussetzung ist zunächst F' stetig und $F'(x^*)$ invertierbar; nach dem Störungslemma (Satz 4.16) ist daher F' invertierbar in einer Umgebung von x^* und es existiert eine Konstante $c > 0$ und ein Radius $\varepsilon > 0$ mit $\|F'(x)^{-1}\| \leq c$ für alle $\|x - x^*\| \leq \varepsilon$.

Für den zweiten Term gehen wir analog zu Folgerung 11.2 vor: Für alle $x, y \in B_\varepsilon(x^*) := \{x \in \mathbb{R}^n : \|x - x^*\| \leq \varepsilon\}$ gilt

$$\|F'(x) - F'(y)\| \leq \sup_{z \in B_\varepsilon(x^*)} \|F''(z)\|\|x - y\| \leq C\|x - y\|,$$

da nach dem Satz von Weierstraß die stetige Funktion F'' auf der kompakten Menge $B_\varepsilon(x^*)$ ihr Maximum annimmt. Also ist insbesondere für $x^k \in B_\varepsilon(x^*)$

$$\begin{aligned} &\|(F(x^*) - F(x^k)) - F'(x^k)(x^* - x^k)\| \\ &\qquad = \left\| \int_0^1 F'(x^k + t(x^* - x^k))(x^* - x^k)\, dt - F'(x^k)(x^* - x^k) \right\| \\ &\qquad \leq \int_0^1 \|F'(x^k + t(x^* - x^k)) - F'(x^k)\|\|x^* - x^k\|\, dt \\ &\qquad \leq \int_0^1 C\|t(x^* - x^k)\|\|x^* - x^k\|\tilde{}, dt \\ &\qquad = \frac{C}{2}\|x^k - x^*\|^2, \end{aligned}$$

da für $x^k, x^* \in B_\varepsilon(x^*)$ auch $x^k + t(x^* - x^k) \in B_\varepsilon(x^*)$ gilt für alle $t \in [0, 1]$. Damit haben wir gezeigt, dass für $x^k \in B_\varepsilon(x^*)$ gilt

$$\|x^{k+1} - x^*\| \leq \frac{Cc}{2}\|x^k - x^*\|^2. \tag{11.4}$$

Wir zeigen nun per Induktion, dass aus $x^0 \in B_\rho(x^*)$ für $\rho := \min\{\varepsilon, \frac{1}{Cc}\}$ auch $x^k \in B_\rho(x^*)$ für alle $k \in \mathbb{N}$ folgt. Für $k = 0$ ist nichts zu zeigen; gilt die Aussage für ein $k \in \mathbb{N}$, so ist insbesondere $\|x^k - x^*\| \leq \rho \leq \varepsilon$ und damit folgt aus (11.4) wegen $\rho < (Cc)^{-1}$

$$\|x^{k+1} - x^*\| \leq \frac{Cc}{2}\|x^k - x^*\|^2 \leq \frac{1}{2}\|x^k - x^*\| < \rho.$$

Also gilt auch $x^{k+1} \in B_\rho(x^*)$.

Daraus folgt nun insbesondere, dass $F'(x^k)$ invertierbar ist für alle $k \in \mathbb{N}$; das Newton-Verfahren ist also durchführbar. Weiterhin folgt

$$\|x^{k+1} - x^*\| \leq \frac{1}{2}\|x^k - x^*\| \leq \frac{1}{2^{k+1}}\|x^0 - x^*\| \to 0 \quad \text{für } k \to \infty$$

und damit die (lokale) Konvergenz. Dass die Konvergenz quadratisch ist, folgt nun schließlich aus (11.4). □

Für einen hinreichend guten Startwert wird das Newton-Verfahren also quadratisch gegen eine einfache Nullstelle einer zweimal stetig differenzierbaren Funktion konvergieren. Liegt eine mehrfache Nullstelle vor (bzw. ist $F'(x^*)$ nicht invertierbar), geht die superlineare Konvergenz verloren. Liegt der Startwert nicht nahe genug bei der gesuchten Nullstelle, kann man im Allgemeinen gar keine Konvergenz beobachten. Für skalare Funktionen kann man eine gute Anfangsnäherung für das Newton-Verfahren zum Beispiel mit dem Bisektionsverfahren konstruieren, das global konvergent ist. (Dies ist z. B. ein Standard-Ansatz, um Nullstellen von Polynomen zu bestimmen, etwa für die Gauß-Quadratur.)

Das Newton-Verfahren hat auch eine einfache geometrische Interpretation. Betrachtet man die Taylorentwicklung von $f : \mathbb{R} \to \mathbb{R}$ im Punkt x^k und vernachlässigt den quadratischen Term, so erhält man eine lineare Näherung an f im Punkt x^k,

$$y = f(x^k) + f'(x^k)(x - x^k),$$

nämlich die Tangente an f im Punkt $(x^k, f(x^k))$. Ihr Schnittpunkt mit der x-Achse ist $y = 0$, d. h.

$$x^{k+1} := x^k - \frac{f(x^k)}{f'(x^k)},$$

welcher im Allgemeinen eine bessere Näherung an die Nullstelle x^* liefert.[3]

[3] Diese Idee der iterativen Linearisierung führte auch Newton auf das nach ihm benannte Verfahren. (Er entwickelte es 1669 zur Lösung einer kubischen Gleichung.)

11.3 NICHTLINEARE AUSGLEICHSRECHNUNG: GAUSS-NEWTON

Sind $m > n$ Gleichungen für n Unbekannte zu lösen, so ist $F'(x^k) \in \mathbb{R}^{m\times n}$ niemals invertierbar; das linearisierte Gleichungssystem (11.3) ist überbestimmt. Man löst also stattdessen in jedem Schritt das lineare Ausgleichsproblem

$$\min_{x\in\mathbb{R}^n} \|F'(x^k)(x - x^k) + F(x^k)\|_2^2. \tag{11.5}$$

Solange $F'(x^k)$ immer vollen Spaltenrang n hat, kann dieses Minimum mit den Methoden aus Kapitel 6 berechnet werden. Insbesondere erfüllt die Lösung x^{k+1} von (11.5) die Normalengleichungen

$$F'(x^k)^T F'(x^k)(x^{k+1} - x^k) = F'(x^k)^T F(x^k),$$

d. h.

$$x^{k+1} = x^k - [F'(x^k)^T F'(x^k)]^{-1} F'(x^k)^T F(x^k) =: x^k - F'(x^k)^\dagger F(x^k). \tag{11.6}$$

Analog zum Newton-Verfahren kann man zeigen, dass diese Iteration – genannt *Gauß–Newton-Verfahren* – lokal quadratisch gegen eine Nullstelle x^* von F konvergiert. Hat F keine Nullstelle, so konvergiert die Iteration unter einer Zusatzannahme immer noch (wenn auch nur linear) gegen eine Lösung des *nichtlinearen Ausgleichsproblems*

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|F(x)\|_2^2. \tag{11.7}$$

Satz 11.6. *Sei $F : \mathbb{R}^n \to \mathbb{R}^m$ stetig differenzierbar in einer Umgebung $U \subset \mathbb{R}^n$ um eine Lösung $x^* \in U$ des Ausgleichsproblems (11.7). Es gelte*

(i) $F'(x)$ hat vollen Spaltenrang für alle $x \in U$;

(ii) es existiert ein $L > 0$ mit

$$\|F'(x)^\dagger (F'(y) - F'(x))\| \le L\|x - y\| \quad \text{für alle } x, y \in U;$$

(iii) es existiert ein $\kappa \in [0, 1)$ mit

$$\|F'(x)^\dagger F(x^*)\| \le \kappa\|x - x^*\| \quad \text{für alle } x \in U.$$

Dann konvergiert das Gauß–Newton-Verfahren lokal linear gegen x^*. Gilt $F(x^*) = 0$, so ist die Konvergenz sogar quadratisch.

Beweis. Der Beweis ist analog zum Konvergenzbeweis für das Newton-Verfahren. Aus der Iterationsvorschrift folgt diesmal mit der produktiven Null sowie

$$F'(x)^\dagger F'(x) = (F'(x)^T F'(x))^{-1} F'(x)^T F'(x) = I$$

nach Annahme (i) die Fehlerrekursion

$$\begin{aligned} x^{k+1} - x^* &= x^k - x^* - F'(x^k)^\dagger F(x^k) \\ &= x^k - x^* - F'(x^k)^\dagger \Big(F(x^k) - F(x^*) \Big) - F'(x^k)^\dagger F(x^*) \\ &= F'(x^k)^\dagger \Big(F(x^*) - F(x^k) - F'(x^k)(x^* - x^k) \Big) - F'(x^k)^\dagger F(x^*) \\ &= \int_0^1 F'(x^k)^\dagger \left[F'(x^k + t(x^* - x^k)) - F'(x^k) \right] (x^* - x^k)\, dt - F'(x^k)^\dagger F(x^*). \end{aligned}$$

Mit Annahme (ii) und (iii) erhalten wir damit wie für das Newton-Verfahren die Abschätzung

$$\begin{aligned} \|x^{k+1} - x^*\| &\leq \int_0^1 \|F'(x^k)^\dagger [F'(x^k + t(x^* - x^k)) - F'(x^k)]\| \|x^k - x^*\|\, dt \\ &\quad + \|F'(x^k)^\dagger F(x^*)\| \\ &\leq \frac{L}{2} \|x^k - x^*\|^2 + \kappa \|x^k - x^*\|. \end{aligned}$$

Wähle nun $x^0 \in B_\rho(x^*) \subset U$ für $\rho < \frac{2(1-\kappa)}{L}$ und sei $x^k \in B_\rho(x^*)$ für $k \in \mathbb{N}$ beliebig. Dann folgt aus der gerade gezeigten Abschätzung

$$\|x^{k+1} - x^*\| \leq \left(\frac{L}{2}\rho + \kappa \right) \|x^k - x^*\| \leq \left(\frac{L}{2}\rho + \kappa \right) \rho < \rho$$

nach Annahme an ρ. Also ist auch $x^{k+1} \in B_\rho(x^*)$, und wir erhalten

$$\|x^{k+1} - x^*\| \leq \left(\frac{L}{2}\rho + \kappa \right)^{k+1} \|x^0 - x^*\| \to 0 \quad \text{für } k \to \infty$$

und damit die lineare Konvergenz.

Ist schließlich $F(x^*) = 0$, dann können wir $\kappa = 0$ wählen und erhalten wie für das Newton-Verfahren die quadratische Konvergenz. □

WEITERFÜHRENDE LITERATUR

C. T. Kelley (1995), *Iterative methods for linear and nonlinear equations*, Bd. 16, Frontiers in Applied Mathematics, Society for Industrial & Applied Mathematics (SIAM), Philadelphia, PA.

C. T. Kelley (1999), *Iterative methods for optimization*, Bd. 18, Frontiers in Applied Mathematics, Society for Industrial & Applied Mathematics (SIAM), Philadelphia, PA.