Numerische Mathematik A

zusammengestellt von Lothar Nannen

Wintersemester 2019/2020

Version: 23. August 2019

Inhaltsverzeichnis

1 Einleitung

Die numerische Mathematik beschäftigt sich mit der Konstruktion und Analyse von Algorithmen für mathematische Probleme. Das Interesse an numerischen Methoden zu bestimmten Problemen besteht meist aus einem der folgenden Gründe:

1. Es gibt keine explizite Lösungsdarstellung.
2. Es gibt zwar eine Lösungsdarstellung, diese ist jedoch nicht geeignet, um die Lösung schnell auszurechnen.
3. Es gibt zwar eine Lösungsdarstellung, diese liegt aber in einer Form vor, in welcher sich Rechenfehler stark bemerkbar machen.

Anhand zweier Beispiele wollen wir zunächst verdeutlichen, dass es gute und weniger gute numerische Verfahren zum Lösen eines Problems geben kann.

Beispiel 1.1. *In diesem Beispiel wollen wir $\sqrt{2}$ numerisch bestimmen und betrachten dazu die Gleichung $x^2 = 2$ für $x > 0$.*

Ein Verfahren zur Lösung solcher nichtlinearer Gleichungen, welches wir später noch genauer untersuchen werden, ist das Verfahren der sukzessiven Approximation. Dazu bringen wir die Gleichung $x^2 = 2$ in eine Fixpunktform $f(x) = x$. Für $x \neq 0$ wäre für f theoretisch $f_1(x) := \frac{1}{x} + \frac{x}{2}$ oder $f_2(x) := \frac{2}{x}$ geeignet. Beginnen wir mit dem Startwert $x_0 = 5$ und nutzen die Iterationsvorschrift $x_{k+1} = f(x_k)$, so erhalten wir für $f_1(x)$ die Folge

$$5.0 \longrightarrow 2.7 \longrightarrow 1.72 \longrightarrow 1.441 \longrightarrow 1.415 \longrightarrow \dots$$

und für $f_2(x)$ die Folge

$$5.0 \longrightarrow 0.4 \longrightarrow 5.0 \longrightarrow 0.4 \longrightarrow 5.0 \longrightarrow \dots .$$

Wir sehen also, dass die zweite äquivalente Darstellung keineswegs geeignet ist $\sqrt{2}$ zu approximieren, die erste Darstellung jedoch schon.

Beispiel 1.2. *Wir betrachten den Einheitskreis in der Ebene und die eingeschriebenen regelmäßigen n-Ecke mit dem Umfang u_n. Der Anschauung entnehmen wir sofort*

$$u_n < u_{2n} < 2\pi.$$

Durch einfache geometrische Überlegungen lassen sich zwei Rekursionsformeln zur Bestimmung von u_n herleiten, nämlich

$$u_{2n} = \sqrt{4n \cdot \left(2n - \sqrt{4n^2 - u_n^2}\right)} = \sqrt{\frac{4n \cdot u_n^2}{\left(2n + \sqrt{4n^2 - u_n^2}\right)}}$$

Iteration	erste Vorschrift	zweite Vorschrift
1	6.00000	6.00000
2	6.21166	6.21166
3	6.26526	6.26526
4	6.27870	6.27870
⋮	⋮	⋮
15	6.28319	6.28319
16	6.28318	6.28319
⋮	⋮	⋮
26	6.00000	6.28319
27	6.92820	6.28319
28	0.00000	6.28319
⋮	⋮	⋮
40	0.00000	6.28319

Tabelle 1.1: Numerische Berechnung von 2π.

mit dem Startwert $u_3 = 3\sqrt{3}$*. Im folgenden haben wir beide Rekursionsformeln in Mathematica 5.0 eingegeben und einige Iterationsschritte berechnen lassen. Wie Tabelle 1.1 zeigt, liefert die erste Vorschrift bis zur 16. Iteration eine gegen* 2π *monoton wachsende Folge. Danach jedoch weichen die Folgeglieder stark von* 2π *ab und sind der 28. Iteration konstant* 0*. Die zweite Vorschrift konvergiert offenbar gegen* 2π *und zeigt kein derart unschönes Verhalten.*

Speziell kommt es in Beispiel 1.2 zur sogenannten Auslöschung. Bei der ersten Iterationsvorschrift werden für große n zwei fast gleich große Zahlen voneinander subtrahiert. Da jede reelle Zahl im Rechner aber nur mit endlich vielen Nachkommastellen genau angezeigt werden kann, werden während der Rekursion sehr kleine Zahlen auf 0 gerundet, obwohl ihr eigentlicher Wert durchaus bedeutsam für das exakte Ergebnis wäre. Dieses Problem tritt bei der zweiten Vorschrift nicht auf.

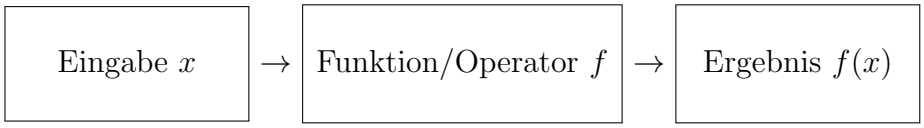
Diese Beispiele verdeutlichen uns, dass mit numerischen Ergebnissen durchaus kritisch umgegangen werden muss.

Literatur-Hinweis: Das Vorlesungsskript wurde in Teilen wörtlich von Thorsten Hohage, *Vorlesungsskript Numerische Mathematik 1* (2010) übernommen. Es ist als Begleitmaterial zur Vorlesung gedacht. Daher gibt es nur wenige Beispiele und Erläuterungen. Diese finden sich u.a. in folgenden Lehrbüchern:

- P. Deuflhard, A. Hohmann, *Numerische Mathematik 1*, W. de Gruyter
- J. Stoer, R. Bulirsch, *Numerische Mathematik I/II*, Springer
- H. Werner, R. Schaback, *Praktische Mathematik I/II*, Springer

2 Fehleranalyse

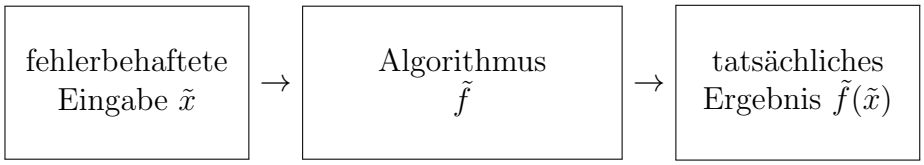
Wir betrachten das abstrakte Problem, aus gegebenen Daten x ein gesuchtes Resultat $y = f(x)$ zu berechnen, das durch eine Funktion oder einen Operator $f : x \mapsto y$ definiert ist. Dies wird durch folgendes Schema veranschaulicht:



Häufig sind die Eingabedaten x nicht exakt gegeben, sondern uns stehen nur fehlerbehaftete Daten $\tilde{x}$ zur Verfügung. Weiterhin können wir im allgemeinen aufgrund der folgenden fundamentalen Restriktionen auch die Funktion f nicht exakt auswerten:

- endliche Genauigkeit $\rightsquigarrow$ Rundungsfehler
- endlicher Speicherplatz $\rightsquigarrow$ Approximationsfehler
- endliche Zeit $\rightsquigarrow$ Verfahrensfehler

Daher müssen wir uns mit einem Algorithmus zur näherungsweisen Auswertung von f zufrieden geben. Dieser werde durch eine Funktion $\tilde{f}$ beschrieben. Die tatsächliche Situation wird also durch folgendes Schema dargestellt:



In diesem Kapitel werden wir Begriffe und Werkzeuge kennenlernen, mit denen entschieden werden kann, ob die berechnete Lösung $\tilde{f}(\tilde{x})$ eine akzeptable Näherung an die exakte Lösung $f(x)$ darstellt.

Definition 2.1. *Seien V und W normierte, lineare Räume, $f, \tilde{f} : V \to W$ und $x, \tilde{x} \in V$. Dann bezeichnet $f(x) - \tilde{f}(\tilde{x})$ den* absoluten *Fehler und $\frac{f(x)-\tilde{f}(\tilde{x})}{\|f(x)\|_W}$ für $f(x) \neq 0$ den* relativen *Fehler.*

Wir haben bereits gesehen, dass wir grundsätzlich zwei Ursachen für Fehler zu erwarten haben. Gegenüber Eingabefehlern sind wir prinzipiell machtlos, sie können

allenfalls durch eine Änderung der Problemstellung beeinflusst werden. Hingegen haben wir bei der Wahl des Algorithmus $\tilde{f}$ die Möglichkeit, das Gesamtergebnis $\tilde{f}(\tilde{x})$ zu verbessern. Die Unterscheidung dieser beiden Fehlerarten führt auf die Begriffe *Kondition eines Problems* und *Stabilität eines Algorithmus*.

2.1 Gleitkommaarithmetik, Rundungsfehler

Wir wollen uns nun ansehen, wie reelle Zahlen üblicherweise in einem Computer gespeichert werden. Da es unendlich viele reelle Zahlen gibt, kann dies nur näherungsweise erfolgen. In der heute fast überall verwendeten *Gleitkommadarstellung* (engl. *floating point representation*) wird eine reelle Zahl x bezüglich einer Basis b (typischerweise $b = 2$ oder $b = 16$) dargestellt als

$$x = ab^e. \tag{2.1}$$

Dabei ist der *Exponent* e eine ganze Zahl aus einem gegebenen Bereich

$$e \in \{e_{\min}, \ldots, e_{\max}\} \subset \mathbb{Z},$$

und die *Mantisse* a ist 0 oder eine Zahl der Form

$$a = v \sum_{i=1}^{l} a_i b^{-i}, \tag{2.2}$$

wobei $v \in \{\pm 1\}$ das *Vorzeichen*, $a_i \in \{0, \ldots, b-1\}$ die Ziffern und $l \in \mathbb{N}$ die *Mantissenlänge* ist. Der Exponent sei so gewählt, dass $a_1 \neq 0$. Durch diese Festlegung erhalten wir Eindeutigkeit der Darstellung von x, man spricht auch von *normalisierter Gleitkommadarstellung*. Die so darstellbaren Zahlen bilden eine endliche Teilmenge

$$\mathbb{F} := \{x \in \mathbb{R} : \text{es gibt } a, e \text{ wie oben, so dass } x = ab^e\}.$$

Der Vorteil der Gleitkommadarstellung (2.1) gegenüber einer Darstellung $x = v \sum_{i=1}^{l} a_i b^{-i}$ liegt darin, dass die *relative Genauigkeit* der nächstliegenden Maschinenzahl $\tilde{x}$ einer reellen Zahl x unabhängig von der Größe von x ist. Die relative Genauigkeit wird nur durch die Mantissenlänge bestimmt. Genauer: Jede reelle Zahl x aus dem Bereich $b^{e_{\min}-1} \leq |x| \leq b^{e_{\max}}(1 - b^{-l})$ kann durch eine nächstliegende Gleitkommazahl $\tilde{x} \in \mathbb{F}$ so approximiert werden, dass der relative Fehler der Abschätzung

$$\frac{|x - \tilde{x}|}{|x|} \leq \frac{\text{eps}}{2}, \qquad \text{eps} := b^{1-l}$$

genügt. Die Zahl eps heisst *relative Maschinengenauigkeit*. Falls $|x|$ größer als die betragsmäßig größte darstellbare Zahl

$$\max\{|x| : x \in \mathbb{F}\} = b^{e_{\max}}(1 - b^{-l})$$

ist, spricht man von einem *Exponentenüberlauf* (engl. *overflow*), und umgekehrt, falls $|x|$ kleiner als die betragsmäßig kleinste darstellbare Zahl

$$\min\{|x| : x \in \mathbb{F} \setminus \{0\}\} = b^{e_{\min}-1}$$

ist, von einem *Exponentenunterlauf* (engl. *underflow*).

Der vom Institute of Electrical and Electronics Engineers (IEEE) im Jahre 1985 herausgegebene Standard IEC559, der z.B. auch in Matlab realisiert ist, sieht zwei Gleitkommatypen zur Basis $b = 2$ vor: einen mit einfacher Genauigkeit (*single precision*, *float* in C/C++) und einen mit doppelter Genauigkeit (*double precision*, *double* in C/C++). Die charakteristischen Werte dieser Gleitkommatypen sind in der folgenden Tabelle wiedergegeben:

	single precision	double precision
Mantissenlänge l	23	52
kleinster Exponent $e_{\min}$	-125	-1021
größter Exponent $e_{\max}$	128	1024
rel. Maschinengenauigkeit eps	$1.1921e \cdot 10^{-7}$	$2.2204 \cdot 10^{-16}$
$\min\{\|x\| : x \in \mathbb{F} \setminus \{0\}\}$	$1.1755 \cdot 10^{-38}$	$2.2251 \cdot 10^{-308}$
$\max\{\|x\| : x \in \mathbb{F}\}$	$3.4028 \cdot 10^{38}$	$1.7977 \cdot 10^{308}$
Speicherplatz	4 Byte	8 Byte

Im IEEE Standard ist weiterhin eine Arithmetik auf der Zahlenmenge $\mathbb{F}$ definiert, die die Elementaroperationen

$$* \in \{+, -, \cdot, /\}$$

so realisiert, dass

$$a \circledast b = (a * b)(1 + \epsilon) \qquad \text{mit } |\epsilon| < \text{eps}.$$

Außerdem sind im IEEE Standard die Werte $+\infty$, $-\infty$, NaN (Not a Number) vorgesehen, damit Ausnahmeoperation wie etwa eine Division durch 0 nicht zu einem Programmabbruch führen müssen. Dafür wurde etwa bei *single precision numbers* der mit einem Exponenten aus 8 Bits[1] und $e_{\max} = 128$ kleinste realisierbare Werte $e_{\min} = -127$ auf $e_{\min} = -125$ heraufgesetzt. Die folgende Tabelle gibt einige Ausnahmesituationen wieder:

Ausnahme	Beispiel	Ergebnis
ungültige Operation	$0/0$, $0 \cdot \infty$	NaN
Exponentenüberlauf		$\pm\infty$
Division durch 0	$1/0$	∞

2.2 Kondition eines Problems

Im Folgenden werden wir die *Landau-Symbole* $\mathcal{O}$ und o verwenden.

Definition 2.2. *Seien (a_n) und (b_n) zwei reelle Folgen. Dann definieren wir $a_n = \mathcal{O}(b_n)$, wenn es ein $C \geq 0$ und ein $N \in \mathbb{N}$ gibt mit*

$$|a_n| \;\leq\; C\,|b_n|, \qquad n \geq N \qquad (a_n \text{ wächst nicht schneller als } b_n).$$

[1] Wir erinnern daran, dass ein Bit die kleinstmögliche Speichereinheit ist und die Werte 0 oder 1 speichern kann. Ein Byte besteht aus 8 Bits.

Wir definieren $a_n = o(b_n)$*, wenn es ein* $C(n) \geq 0$ *mit* $\lim_{n\to\infty} C(n) = 0$ *und ein* $N \in \mathbb{N}$ *gibt mit*

$$|a_n| \;\leq\; C(n)\,|b_n|, \qquad n \geq N \qquad (a_n \text{ wächst langsamer als } b_n).$$

Das Landau-Symbol lässt sich analog für andere Grenzübergänge wie z.B. $h \to 0$ definieren. Sofern der Kontext nicht eindeutig ist, sollte deshalb bei Verwendung von Landau-Symbolen immer angegeben werden, welcher Grenzübergang gemeint ist.

Beispiel 2.3. *Sei* $x > 0$*,* $f(x) := \frac{\exp(ix^2)}{x}$ *und* $g(x) := \frac{1}{x}$*. Dann ist* $f(x) = \mathcal{O}(g(x))$ *für* $x \to \infty$ *aber* $f'(x) \neq \mathcal{O}(g'(x))$ *für* $x \to \infty$*.*

Wir betrachten die Auswertung einer Funktionen $f : V' \subset V \to W$ an einem Punkt $x \in V'$. Dabei sei $V = \mathbb{K}^n$ und $W = \mathbb{K}^m$ mit Normen $\|\cdot\|_V$ und $\|\cdot\|_W$. Ist f zweimal stetig differenzierbar, so gilt für den Fehler $\Delta y := f(x+\Delta x) - f(x)$ bei einer kleinen Störung Δx aufgrund der Fehlerabschätzung für die Taylor-Approximation erster Ordnung die Ungleichung

$$\Delta y = Df(x)\;\Delta x + \mathcal{O}(\|\Delta x\|_V^2), \qquad \|\Delta x\|_V \to 0.$$

Dabei bezeichnen wir mit $Df(x) := \left(\frac{\partial f_i(x)}{\partial x_j}\right)_{i=1..m,j=1..n}$ die Jacobi-Matrix von f bei x. Daraus folgt für $i = 1, ..., m$

$$\Delta y_i = \sum_{j=1}^{n} \frac{\partial f_i(x)}{\partial x_j}\Delta x_j + \mathcal{O}(\|\Delta x\|_V^2), \qquad \|\Delta x\|_V \to 0, \tag{2.3}$$

bzw. für $y_i := f_i(x) \neq 0$ und $x_j \neq 0$

$$\frac{\Delta y_i}{y_i} = \sum_{j=1}^{n} \left(\frac{\partial f_i(x)}{\partial x_j}\frac{x_j}{f_i(x)}\right)\frac{\Delta x_j}{x_j} + \mathcal{O}(\|\Delta x\|_V^2), \qquad \|\Delta x\|_V \to 0. \tag{2.4}$$

In der letzten Formel ist zu beachten, dass im Landau-Symbol eine Abhängigkeit y_i enthalten ist, d.h. eigentlich sollten wir besser $\mathcal{O}\left(\|\Delta x\|_V^2/|y_i|\right)$ verwenden. Für hinreichend kleine $\|\Delta x\|_V/|y_i|$ können wir das Restglied vernachlässigen und erhalten so die folgende Definition.

Definition 2.4. *Für* $x \in V'$ *heißen die Zahlen*

$$\kappa_{ij}^{\text{abs}}(x) := \left|\frac{\partial f_i(x)}{\partial x_j}\right|, \qquad i = 1, ..., m, j = 1, ..., n, \tag{2.5}$$

die absoluten Konditionszahlen *des Problems* (f, x) *(Auswertung der Funktion* f *am Punkt* x*). Ist* $x_j \neq 0$ *und* $f_i(x) \neq 0$*, so nennen wir*

$$\kappa_{ij}^{\text{rel}}(x) := \left|\frac{\partial f_i(x)}{\partial x_j}\frac{x_j}{f_i(x)}\right|, \qquad i = 1, ..., m, j = 1, ..., n, \tag{2.6}$$

die relativen Konditionszahlen *des Problems* (f, x). *Die* relative Konditionszahl *des Problems* (f, x) *kann durch*

$$\kappa_{\text{rel}}(x) := \max\{k_{ij}^{\text{rel}}(x), i = 1, ..., m, j = 1, ..., n\} \tag{2.7}$$

definiert werden.

Ist die Funktion f nicht differenzierbar, so können die absoluten Konditionszahlen des Problems (f, x) auch als kleinste Zahlen κ_{ij}^{abs} definiert werden, sodass

$$|f_i(x + he_j) - f_i(x)| \leq \kappa_{ij}^{\text{abs}}|h|$$

für alle hinreichend kleinen $|h|$ gilt, wobei $h \in \mathbb{K}$ und e_j der j-te Einheitsvektor ist.

Die relativen Konditionszahlen sind so definiert, dass

$$\begin{aligned} \frac{|\Delta y_i|}{|y_i|} &\leq \sum_{j=1}^{n} \kappa_{ij}^{\text{rel}}(x) \frac{|\Delta x_j|}{|x_j|}(1 + o(1)) \\ &\leq \kappa_{\text{rel}}(x) \left(\sum_{j=1}^{n} \frac{|\Delta x_j|}{|x_j|} \right) (1 + o(1)) \qquad \|\Delta x\| \to 0 \end{aligned} \tag{2.8}$$

für eine stetig differenzierbare Funktion f. (Ist f sogar zweimal stetig differenzierbar, kann man $o(1)$ durch $O(\|\Delta x\|)$ ersetzen.)

Ein Problem heißt *gut konditioniert*, falls seine Kondition "klein" und *schlecht konditioniert*, falls sie "groß" ist. Im allgemeinen ist die relative Konditionszahl gebräuchlicher als die absolute Konditionszahl. Sie hat den Vorteil, dass sie invariant gegebenüber Umskalierungen der Normen auf V und W ist. Allerdings bricht das Konzept der relativen Kondition zusammen, wenn $x = 0$ oder $f(x) = 0$ ist.

Beispiel 2.5 (Arithmetische Grundoperationen)**.** *Wir untersuchen drei Grundoperationen.*

1. Multiplikation*: $y = x_1 x_2$ ist für $x_1 x_2 \neq 0$ gut konditioniert mit $\kappa_{\text{rel}}(x_1, x_2) \equiv 1$.*
2. Addition*: $y = x_1 + x_2$ hat für $x_1 + x_2 \neq 0$ die relative Konditionszahl*

   $$\kappa_{\text{rel}}(x_1, x_2) = \frac{\max\{|x_1|, |x_2|\}}{|x_1 + x_2|}$$

   und ist für $x_1 + x_2 \approx 0$ schlecht konditioniert.
3. Quadratwurzel*: $y = \sqrt{x}$ ist für $x > 0$ gut konditioniert mit $\kappa_{\text{rel}}(x) \equiv \frac{1}{2}$.*

Bei der Addition ähnlich großer Zahlen mit negativem Vorzeichen können somit kleine Störungen in den Eingangsdaten große Störungen in der Summe ergeben. Dies nennt man *Auslöschung*. Die Multiplikation ist immer gut konditioniert, sofern die relative Konditionszahl wohldefiniert ist.

Bemerkung 2.6. *Wer werden später Matrixnormen definieren. Damit lässt sich die absolute Konditionszahl des Problems (f, x) alternativ durch $\kappa_{\text{abs}}(x) := \|Df(x)\|$ und für $\|x\| \neq 0$ und $\|f(x)\| \neq 0$ die relative Konditionszahl durch*

$$\kappa_{\text{rel}}(x) := \frac{\|Df(x)\| \, \|x\|}{\|f(x)\|} \tag{2.9}$$

definieren. Wir erhalten dann

$$\frac{\|\Delta y\|}{\|y\|} \leq \kappa_{\text{rel}} \frac{\|\Delta x\|}{\|x\|}(1 + o(1)), \qquad \|\Delta x\| \to 0. \tag{2.10}$$

Die beiden Definitionen für $\kappa_{\text{rel}}(x)$ sind jedoch nicht äquivalent!

Die Kondition eines Problems hängt vom Maß ab, mit dem Fehler gemessen werden. Ein Vergleich von (2.8) und (2.10) anhand der Multiplikation von $x_1 = 10^{-16}$ mit $x_2 = 10^{16}$ zeigt die Unterschiede: Wählen wir einen Fehler $\Delta x = (10, 10)^\top \in \mathbb{R}^2$, so ist der Fehler in der rechten Seite von (2.10) $\|\Delta x\|_\infty / \|x\|_\infty = 10^{-15}$ klein, der Fehler in der linken Seite mit $|\Delta y|/|y| = 10^{17} + 100 + 10^{-15}$ sehr groß. Entsprechend ist die relative Konditionszahl der Multiplikation definiert nach der vorigen Bemerkung in diesem Fall sehr groß. Die Multiplikation ist jedoch wie vorher bewiesen gut konditioniert; $\|\Delta x\|/\|x\|$ ist ein zu grobes Maß für die Fehler in den Eingabedaten. Messen wir diesen Fehler wie in der rechten Seite von (2.8) durch $|\Delta x_1|/|x_1| + |\Delta x_2|/|x_2|$, so müssten wir für einen vergleichbaren Fehler in den Eingabedaten $\Delta x = (5 \cdot 10^{-32}, 5)^\top$ wählen und erhalten einen Ausgabefehler $|\Delta y|/|y| \approx 10^{-15}$ in der Größenordnung des Eingabefehlers.

2.3 Stabilität eines Algorithmus/ Vorwärtsanalyse

Während wir im vorigen Abschnitt den Einfluss von Eingabefehlern untersucht haben, widmen wir uns nun der Analyse von Fehlern, die durch den Algorithmus verursacht werden. Hierzu betrachten wir die Vorwärtsanalyse.

Wir erinnern daran, dass bei der Darstellung reeller Zahlen im Computer zwangsläufig Rundungsfehler auftreten. Ein Eingabedatum $\tilde{x} \in \mathbb{F} \subset \mathbb{R}$ repräsentiert also ein ganzes Intervall $E = E_{\tilde{x}}$ von möglichen exakten Eingabewerten x. Die Länge $|E|$ dieses Intervalls lässt sich für $\tilde{x} \neq 0$ abschätzen durch

$$\frac{|E_{\tilde{x}}|}{|\tilde{x}|} \leq \text{eps}.$$

Wir fragen uns nun, welchen Einfluss Rundungsfehler bei Zwischenwerten haben, die im Laufe eines Algorithmus $\tilde{f}$ auftreten. Dabei gehen wir von der Annahme aus, dass die Gleitkommaoperationen $* \in \{+, -, \cdot, /\}$ so realisiert sind, dass der Fehler der Abschätzung

$$a \circledast b = (a * b)(1 + \epsilon(a, b)) \qquad \text{mit } |\epsilon(a, b)| \leq \text{eps} \tag{2.11}$$

genügt. Wir legen uns dabei nicht auf eine bestimmte Gleitkommaarithmetik fest (etwa den IEEE-Standard) fest, sondern betrachten beliebige Gleitkommaarithmetiken im Limes eps $\to 0$.

Bei der Vorwärtsanalyse wird der Unterschied zwischen dem berechneten Ergebnis $\tilde{f}(x)$ bei einer Eingabe x mit dem exakten Ergebnis $f(x)$ verglichen. Ist der durch den Algorithmus verursachte Fehler im Ergebnis von derselben oder geringerer Größenordnung als der durch Eingabefehler verursachte Fehler, so heißt der Algorithmus stabil im Sinne der Vorwärtsanalyse.

Definition 2.7. *Sei (f, x) ein Problem mit wohldefinierter relativer Konditionszahl κ_{rel} definiert in (2.7). Dann ist der* Stabilitätsindikator σ_V der Vorwärtsanalyse *eines Algorithmus definiert als kleinste Zahl, so dass für alle Gleitkommarealisierungen $\tilde{f}_{\text{eps}}$ dieses Algorithmus, die der Bedingung (2.11) genügen, die Ungleichung*

$$\frac{\left\|\tilde{f}_{\text{eps}}(x) - f(x)\right\|_W}{\|f(x)\|_W} \leq \sigma_V \kappa_{\text{rel}} \text{eps} \cdot (1 + o(1)), \qquad \text{eps} \searrow 0$$

erfüllt ist.

σ_V misst also, um wieviel der durch den Algorithmus verursachte Fehler größer ist als der aufgrund der Eingabefehler verursachte Fehler.

Für die Stabilitätsindikatoren der elementaren Gleitkommaoperationen gilt

$$\sigma_V \kappa_{\text{rel}} \leq 1, \tag{2.12}$$

da wegen der Annahme (2.11)

$$\frac{|a \circledast_{\text{eps}} b - a * b|}{|a * b|} = \frac{|(a * b)(1 + \epsilon) - a * b|}{|a * b|} \leq \text{eps}$$

für alle $a, b \in \mathbb{F}$ mit $a * b \neq 0$ und alle eps > 0 gilt.

Beispiel 2.8 (Lösung einer quadratischen Gleichung)**.** *Die Lösungen einer quadratischen Gleichung der Form $y^2 - 2py - q = 0$ sind für $p^2 + q > 0$ gegeben durch $y_{1,2} = f_{1,2}(p, q)$ mit $f_{1,2}(p, q) := p \pm \sqrt{p^2 + q}$. Für $y_{1,2} \neq 0$ und $p^2 + q > 0$ gilt*

$$\kappa_{\text{rel}}(p, q) = \frac{\max\left\{2|p|, \left|p + \sqrt{p^2 + q}\right|, \left|p - \sqrt{p^2 + q}\right|\right\}}{2\sqrt{p^2 + q}}.$$

Das Problem ist somit schlecht konditioniert, wenn $\sqrt{p^2 + q} \approx 0$, d.h. wenn die beiden Nullstellen nahe beinander liegen. Ansonsten ist das Problem gut konditioniert.

Unter Verwendung von (2.11) und relativen Fehlern ϵ_j mit $|\epsilon_j| <$ eps für $j = 1, .., 4$ erhalten wir bei korrekten Eingabedaten die fehlerbehafteten Nullstellen

$$\tilde{y}_{1,2} = \left(p \pm \sqrt{(p^2(1 + \epsilon_1) + q)\,(1 + \epsilon_2)}(1 + \epsilon_3)\right)(1 + \epsilon_4).$$

Durch Linearisierung und Vernachlässigung von Termen höherer Ordnung (z.B. $\epsilon_1\epsilon_2$) erhalten wir

$$\frac{|y_{1,2} - \tilde{y}_{1,2}|}{|y_{1,2}|} \dot{\leq} \epsilon \quad \textit{mit} \quad 0 \leq \epsilon \leq \left(1 + \frac{|2p^2 + 3q|}{\sqrt{p^2+q}} \frac{1}{|p \pm \sqrt{p^2+q}|}\right) \text{eps}.$$

ϵ kann groß werden, wenn $\sqrt{p^2+q} \approx 0$ (dort ist das Problem schlecht konditioniert), aber auch für $|p \pm \sqrt{p^2+q}| \approx 0$ (Auslöschung, da dann $|p| \approx \sqrt{p^2+q}$). Letzteres kann vermieden werden, indem zunächst $y_1 := p + \text{sgn}(p)\sqrt{p^2+q}$ und dann $y_2 := \frac{-q}{y_1}$ berechnet wird. Da die Multiplikation und damit auch die Division gut konditioniert ist und für y_1 keine Auslöschung auftritt, ist dieser Algorithmus stabil.

Der Beweis der Stabilität eines Algorithmus ist verhältnismäßig aufwändig. Wir werden daher in dieser Vorlesung nur selten auf die Stabilität eingehen.

3 Interpolation

Eine Interpolationsaufgabe besteht darin, aus einer Klasse M von Funktionen eine zu bestimmen, die an vorgegebenen Punkten x_j gewissen Bedingungen genügt. Im einfachsten Fall einer *Lagrangeschen Interpolation* fordern wir, dass die gesuchte Funktion $f \in M$ an den Punkten x_j vorgegebene Werte $y_j \in \mathbb{K}$ annimmt,

$$f(x_j) = y_j, \qquad j = 0, \ldots, n. \tag{3.1}$$

M ist dabei häufig eine Klasse einfach strukturierter Funktionen wie

- *Polynome* $p(x) = a_0 + a_1 x + ... + a_n x^n$,
- *rationale Funktionen* $r(x) = \frac{a_0 + a_1 x + ... + a_n x^n}{b_0 + b_1 x + ... + b_m x^m}$,
- *trigonometrische Polynome* $t(x) = \frac{1}{2} a_0 + \sum_{k=1}^{n} (a_k \cos(kx) + b_k \sin(kx))$ oder
- *Exponentialsummen* $e(x) = \sum_{k=1}^{n} a_k \exp(b_k x)$.

Häufig sind die Funktionen aus M stückweise (auf Teilintervallen) wie oben definiert mit gewissen Glattheitseigenschaften an den Zwischenpunkten. Ist M ein Menge stückweise polynomieller Funktionen, so sprechen wir von *Spline-Interpolation*.

3.1 Polynom-Interpolation

3.1.1 Existenz, Eindeutigkeit und Kondition

Definition 3.1. *Unter einem Polynom p versteht man eine Funktion der Form*

$$p(x) = a_n x^n + \cdots + a_1 x + a_0, \qquad x \in \mathbb{K}$$

mit Koeffizienten $a_j \in \mathbb{K}$. Ist $a_n \neq 0$, so heißt n Grad des Polynoms *p. Der Grad des Polynoms $p \equiv 0$ wird durch -1 definiert. Mit Π_n bezeichnen wir den Vektorraum aller Polynome vom Grad $\leq n$.*

Problem 3.2 (Lagrange-Polynominterpolation)**.** *Gegebenen seien paarweise verschiedene Stützstellen $x_0, \ldots, x_n \in \mathbb{R}$ sowie Stützwerte $y_0, \ldots, y_n \in \mathbb{K}$. Bestimme ein Polynom $p \in \Pi_n$, das die Interpolationsbedingungen*

$$p(x_j) = y_j, \qquad j = 0, \ldots, n \tag{3.2}$$

erfüllt.

Lemma 3.3. *Gegebenen seien paarweise verschiedene Stützstellen* $x_0, \dots, x_n \in \mathbb{R}$. *Dann bilden folgende Polynome eine Basis von* Π_n. *Insbesondere gilt* $\dim \Pi_n = n+1$.

1. *Die* Monome $p_j : x \mapsto x^j$, $j = 0, ..., n$,
2. *die* Lagrange Basispolynome L_j *mit*

$$L_j(x) := \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}, \qquad j = 0, ..., n, \tag{3.3}$$

3. *Die* Newton Basispolynome $N_j \in \Pi_j$ *mit*

$$N_j(x) := \prod_{k=0}^{j-1} (x - x_k), \qquad j = 0, ..., n. \tag{3.4}$$

Weiterhin gilt $L_j(x_k) = \delta_{jk}$, *wobei das* Kronecker-Symbol δ_{jk} *definiert ist durch*

$$\delta_{jk} := \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}.$$

Beweis. Nachrechnen. Die lineare Unabhängigkeit von $\{L_0, ..., L_n\}$ und $\{N_0, ..., N_n\}$ folgt durch Testen mit $x = x_k$, $k = 0, ..., n$. □

Satz 3.4. *Die eindeutige Lösung* p *der Lagrangesche Interpolationsaufgabe 3.2 ist mit den Lagrange Basispolynomen* L_j *gegeben durch* $p = \sum_{j=0}^{n} y_j L_j$.

Beweis. Mit Hilfe des vorigen Lemmas ist klar, dass $\sum_{j=0}^{n} c_j L_j$ genau dann 3.2 löst, wenn $c_j = y_j$ für $j = 0, ..., n$. □

Satz 3.5. *Gegebenen seien paarweise verschiedene Stützstellen* $x_0, \dots, x_n \in [a, b]$, $\{L_0, ..., L_n\}$ *die Lagrange Polynome,* $y := (y_0, ..., y_n)^\top \in \mathbb{K}^{n+1}$ *der Vektor der Stützwerte und* p_y *das interpolierende Polynom abhängig von* y. *Weiter sei die* Lebesgue Konstante Λ *definiert durch* $\Lambda := \max\limits_{x \in [a,b]} \sum\limits_{j=0}^{n} |L_j(x)|$. *Dann gilt*

$$\max_{x \in [a,b]} |p_y(x)| \leq \Lambda \max\{|y_0|, ..., |y_n|\}, \tag{3.5}$$

und es existiert ein Vektor $\tilde{y} \in \mathbb{K}^{n+1}$, *sodass in* (3.5) *Gleichheit gilt.*

Beweis. Die Ungleichung (3.5) folgt mit der Dreiecksungleichung aus der (eindeutigen) Darstellung $p = \sum_{j=0}^{n} y_j L_j$ des Interpolationspolynoms. Zur Existenz des Vektors $\tilde{y}$ sei $\tilde{x} \in [a, b]$ mit $\Lambda = \sum_{j=0}^{n} |L_j(\tilde{x})|$ und $\tilde{y}_j = \operatorname{sgn} L_j(\tilde{x})$ für $j = 0, ..., n$. Dann gilt $\|\tilde{y}\|_\infty = 1$ und

$$\max_{x \in [a,b]} |p_{\tilde{y}}(x)| \geq |p_{\tilde{y}}(\tilde{x})| = \left| \sum_{j=0}^{n} \tilde{y}_j L_j(\tilde{x}) \right| = \sum_{j=0}^{n} |L_j(\tilde{x})| = \Lambda \|\tilde{y}\|_\infty.$$

Zusammen mit (3.5) folgt Gleichheit für dieses $\tilde{y}$. □

Korollar 3.6 (Kondition Polynominterpolation). *Unter den Voraussetzungen des vorigen Satzes sei p_y das Polynom zu den Stützwerten $\{y_0, ..., y_n\}$, $p_{y+\Delta y}$ das Polynom zu den gestörten Stützwerten $y + \Delta y \in \mathbb{K}^{n+1}$ und $\Delta p := p_y - p_{y+\Delta y}$. Dann gilt*

$$\|\Delta p\|_\infty \leq \Lambda \|\Delta y\|_\infty,$$

d.h. die Lebesgue Konstante Λ ist die absolute Konditionszahl des Problems (3.2).

Beweis. Folgt direkt aus dem vorigen Satz und der Linearität der Abbildung $y \mapsto p_y$. □

Die Lebesgue Konstanten können für $n \to \infty$ sehr groß werden (Übungsaufgabe).

3.1.2 Numerische Verfahren zur Lagrange Interpolation

In diesem Unterabschnitt gelten die Voraussetzungen der Lagrange Interpolation 3.2. Zur numerische Berechnung des Polynoms $p \in \Pi_n$ könnte man grundsätzlich eine beliebige Basis $\{\Phi_0, ..., \Phi_n\}$ des Π_n wählen und die Koeffizienten $\alpha_0, ..., \alpha_n \in \mathbb{K}$ in der Basisdarstellung $p = \sum_{j=0}^n \alpha_j \Phi_j$ durch Lösen des linearen Gleichungssystems

$$\begin{pmatrix} \Phi_0(x_0) & \dots & \Phi_n(x_0) \\ \vdots & \ddots & \vdots \\ \Phi_0(x_n) & \dots & \Phi_n(x_n) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix} \tag{3.6}$$

bestimmen. Die Koeffizientenmatrix dieses Gleichungssystems ist regulär und heißt *Vandermonde-Matrix*. Häufig ist man jedoch nicht am Polynom selber, sondern nur an der Auswertung des Polynoms an wenigen Stellen interessiert. Dazu eignet sich besonders das Verfahren von Aitken-Neville.

Lemma 3.7 (Aitken). *Für $j, m \geq 0$ mit $j + m \leq n$ sei $p_{j,m} \in \Pi_m$ das eindeutige Polynom mit*

$$p_{j,m}(x_k) = y_k, \qquad k = j, ..., j + m.$$

Dann gilt für $x \in \mathbb{K}$ die Rekursionsformel

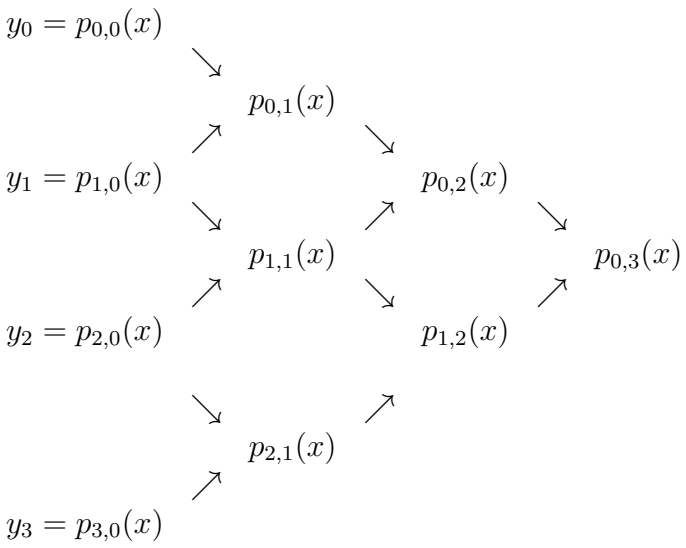
$$p_{j,0}(x) = y_j, \tag{3.7a}$$

$$p_{j,m}(x) = p_{j,m-1}(x) + (x - x_j)\frac{p_{j+1,m-1}(x) - p_{j,m-1}(x)}{x_{j+m} - x_j}, \tag{3.7b}$$

und schließlich $p_{0,n} = p$.

Beweis. Der Beweis erfolgt induktiv über $m = 0, ..., n$. Der Induktionsanfang ist klar. Seien nun $p_{j,m-1} \in \Pi_{m-1}$ für $j = 0, ..., n - m$ die interpolierenden Polynome zu den Stützstellen $x_j, ..., x_{j+m-1}$ und $q \in \Pi_m$ die rechte Seite von (3.7b). Dann folgt $q(x_k) = y_k$ für $k = j, ..., j + m$ und wegen der Eindeutigkeit des Interpolationspolynoms die Behauptung. □

Für festes $x \in \mathbb{K}$ beträgt der Aufwand zur Berechnung von $p(x)$ $\mathcal{O}(n^2)$ Rechenoperationen. Die ersten Schritte des Verfahrens von Aitken-Neville lassen sich in folgendem Schema veranschaulichen:



(3.8)

Für jede neu hinzukommende Stützstelle wird das Schema nach unten erweitert. Im obigen Fall würden nur die Auswertungen $p_{3,0}(x)$, $p_{2,1}(x)$, $p_{1,2}(x)$ und $p_{0,3}(x)$ benötigt, um die Auswertungen $p_{4,0}(x)$, $p_{3,1}(x)$,...,$p_{0,4}(x)$ zu berechnen. Es ist somit problemlos möglich, sukzessive neue Stützstelle hinzuzufügen.

Das Schema von Aitken-Neville eignet sich besonders, wenn die Werte des Interpolationspolynoms an wenigen Stellen benötigt werden. Will man nicht nur einzelne Werte berechnen, ist folgende Darstellung über die Newton Basispolynome (3.4) sinnvoller.

Satz 3.8 (Newton Darstellung)**.** *Das Interpolationspolynom lässt sich mit den Newton Basispolynomen* $\{N_0, ..., N_n\}$ *aus* (3.4) *darstellen durch*

$$p = \sum_{j=0}^{n} y[x_0, ..., x_j] N_j. \tag{3.9}$$

Dabei bezeichnen $y[x_j, ..., x_{j+m}]$ *die zu* (x_k, y_k), $k = j, ..., j+m$ *gehörenden* Dividierten Differenzen*, welche rekursiv definiert werden durch*

$$y[x_j] := y_j, \qquad j = 0, ..., n, \tag{3.10a}$$

$$y[x_j, ..., x_{j+m}] := \frac{y[x_{j+1}, ..., x_{j+m}] - y[x_j, ..., x_{j+m-1}]}{x_{j+m} - x_j}, \quad j, m \geq 0, j + m \leq n. \tag{3.10b}$$

Beweis. Die Darstellung ergibt sich induktiv aus dem vorigen Lemma über Koeffizientenvergleich des jeweils führenden Koeffizienten $y[x_j, ..., x_{j+m}]x^m$ von $p_{j,m}$. □

Analog zu (3.8) lassen sich die dividierten Differenzen in einem Dreiecksschema mit $\mathcal{O}(n^2)$ berechnen. Zur Auswertung des Interpolationspolynoms (3.9) verwenden wir das Horner Schema.

Algorithmus 3.9 (Horner Schema). *Sei $a_j := y[x_0, ..., x_j]$ für $j = 0, ..., n$. Mit der Darstellung*

$$\begin{aligned} p(x) &= a_0 + a_1(x - x_0) + ... + a_n(x - x_0) \cdot ... \cdot (x - x_{n-1}) \\ &= a_0 + (x - x_0)\,(a_1 + (x - x_1)\,(a_2 + ... \,(a_{n-1} + (x - x_{n-1})a_n))) \end{aligned}$$

definieren wir für $x \in \mathbb{K}$

$$b_n := a_n, \qquad b_k := a_k + (x - x_k)b_{k+1}, \qquad k = n - 1, ..., 0,$$

und erhalten $p(x) = b_0$.

Zum Schluß dieses Abschnitts noch einige Aussagen über die dividierten Differenzen, welche im Folgenden nützlich sein werden.

Korollar 3.10. *Die Dividierten Differenzen sind unabhängig von der Reihenfolge der Stützstellen. Insbesondere gilt für eine beliebige Indexmenge $\mathbb{I} \subset \{0, \ldots, n\}$ und beliebige $l, j \in \mathbb{I}$ mit $l \neq j$*

$$y[x_k, k \in \mathbb{I}] = \frac{y[x_k, k \in \mathbb{I} \setminus \{l\}] - y[x_k, k \in \mathbb{I} \setminus \{j\}]}{x_j - x_l}.$$

Lemma 3.11. *Seien $x_0, \ldots, x_n$ paarweise verschieden und $f[x_0, \ldots, x_n]$ die Dividierte Differenz zu den Stützwerten $f(x_0), \ldots, f(x_n)$ einer hinreichend oft stetig differenzierbaren Funktion f. Dann gilt*

$$f[x_0, \ldots, x_n] = \int_0^1 \int_0^{t_1} \cdots \int_0^{t_{n-1}} f^{(n)} \left(x_0 + \sum_{j=1}^{n} t_j (x_j - x_{j-1}) \right) dt_n \ldots dt_1. \tag{3.11}$$

Beweis. Betrachten wir zunächst die einfachsten Fälle $n = 0$ und $n = 1$. Für $n = 0$ wird die rechte Seite von (3.11) zu $f(x_0)$, was der Definition (3.10a) für $f[x_0]$ entspricht. Für $n = 1$ ergibt sich

$$\int_0^1 f'(x_0 + t_1(x_1 - x_0))dt_1 = \frac{f(x_0 + t_1(x_1 - x_0))}{x_1 - x_0}\Big|_{t_1=0}^{t_1=1} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

und damit aus der Definition (3.10b) die Behauptung. Für $n \geq 2$ gilt

$$\int_0^{t_{n-1}} f^{(n)} \left(x_0 + \sum_{j=1}^{n} t_j (x_j - x_{j-1}) \right) dt_n = \frac{f^{(n-1)} \left(x_0 + \sum_{j=1}^{n} t_j (x_j - x_{j-1}) \right)}{x_n - x_{n-1}} \Bigg|_{t_n=0}^{t_n=t_{n-1}}$$

$$= \frac{f^{(n-1)} \left(x_0 + \sum_{j=1}^{n-2} t_j (x_j - x_{j-1}) + t_{n-1}(x_n - x_{n-2}) \right) - f^{(n-1)} \left(x_0 + \sum_{j=1}^{n-1} t_j (x_j - x_{j-1}) \right)}{x_n - x_{n-1}}.$$

Mit

$$f[x_0, \ldots, x_n] = \frac{f[x_0, \ldots, x_{n-2}, x_n] - f[x_0, \ldots, x_{n-2}, x_{n-1}]}{x_n - x_{n-1}}$$

folgt die Behauptung per Induktion. □

Korollar 3.12. *Die Darstellung (3.11) der Dividierten Differenzen ermöglicht deren stetige Fortsetzung für den Fall, daß Stützstellen zusammenfallen. Insbesondere gilt für $x_0 = \cdots = x_n$*

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

3.1.3 Hermite-Interpolation

Im folgenden lassen wir Wiederholungen der Stützstellen x_j zu. Sollen an einem Punkt x_j neben dem Funktionswert auch die Werte der ersten d Ableitungen vorgeschrieben werden, so wählen wir $x_j = x_{j+1} = \cdots = x_{j+d}$.

Problem 3.13 (Hermitesches Interpolationsproblem)**.** *Gegeben seien $x_0 \leq x_1 \leq \cdots \leq x_n$ und $f \in C^n([x_0, x_n])$. Wir definieren die Zahl $i_0 := 0$ und für $j = 1, ..., n$ rekursiv $i_j := i_{j-1}$ falls $x_{j-1} = x_j$ und $i_j := i_{j-1} + 1$ falls $x_{j-1} < x_j$. Für alle $j = 0, \ldots, n$ definieren wir die Zahl $d_j := \max\{l : x_j = x_{j-l}\}$ sowie eine lineare Abbildung $\mu_j : C^{\max\{d_0,\ldots,d_n\}}([x_0, x_n]) \to \mathbb{K}$ durch*

$$\mu_j(f) := f^{(d_j)}(x_j).$$

Gesucht ist ein Polynom $p \in \Pi_n$, das den Interpolationsbedingungen

$$\mu_j(p) = y_{i_j}^{(d_j)}, \qquad j = 0, \ldots, n$$

genügt.

Satz 3.14. *Das Hermitesche Interpolationsproblem 3.13 besitzt genau eine Lösung.*

Beweis. Sind $p_1, p_2 \in \Pi_n$ Lösungen, so ist $p := p_1 - p_2$ ein Polynom vom Grad $\leq n$ mit den Nullstellen $x_0, \ldots, x_n$ (mit Vielfachheit). Mit Hilfe des Fundamentalsatzes der Algebra folgt $p \equiv 0$, also $p_1 = p_2$. Damit haben wir die Eindeutigkeit nachgewiesen. Zum Nachweis der Existenz betrachten die Abbildung

$$\begin{aligned} \mu \quad : \quad & \Pi_n \to \mathbb{K}^{n+1} \\ & p \mapsto (\mu_0(p), \ldots, \mu_n(p))^T. \end{aligned}$$

Aus der soeben bewiesenen Eindeutigkeit der Lösung von Problem 3.13 folgt, dass μ injektiv ist. Da aber $\dim(\Pi_n) = n + 1$, muss μ auch surjektiv sein. Also existiert immer eine Lösung von Problem 3.13. □

Zur Hermite-Interpolation verallgemeinern wir das Lemma von Aitken 3.7.

Lemma 3.15 (verallgemeinertes Lemma von Aitken)**.** *Für eine Indexmenge $\mathbb{I} \subset \{0, ..., n\}$ mit $|\mathbb{I}| = m + 1$ bezeichnen wir analog zum Lemma 3.7 mit $p_{\{x_j, j\in\mathbb{I}\}} \in \Pi_m$ die eindeutige Lösung des Hermite-Interpolationsproblems 3.13 mit*

$$p^{(\tilde{d}_j)}(x_j) = y_{i_j}^{(\tilde{d}_j)}, \qquad \tilde{d}_j := |\{x_l = x_j, l \in \mathbb{I}, l < j\}|, \qquad j \in \mathbb{I}.$$

Für $l, j \in \mathbb{I}$ mit $x_l \neq x_j$ gilt

$$p_{\{x_k, k\in\mathbb{I}\}}(x) = \frac{x_l - x}{x_l - x_j} p_{\{x_k, k\in\mathbb{I}\setminus\{l\}\}}(x) + \frac{x - x_j}{x_l - x_j} p_{\{x_k, k\in\mathbb{I}\setminus\{j\}\}}(x). \tag{3.12}$$

Beweis. Wir bezeichnen die rechte Seite von (3.12) mit q und rechnen die Interpolationsbedingungen nach. Zur Vereinfachung sei $p_1 = p_{\{x_k, k\in\mathbb{I}\setminus\{l\}\}}$ und $p_2 = p_{\{x_k, k\in\mathbb{I}\setminus\{j\}\}}$.

Sei zunächst $k \in \mathbb{I}$ mit $\tilde{d}_k = 0$. Falls $x_k = x_j$, so gilt

$$\mu_k(q) = q(x_j) = p_1(x_j) = y_{i_j}^{(0)}.$$

Für $x_k = x_l$ folgt die Interpolationsbedingung analog und für $x_k \notin \{x_j, x_l\}$ gilt

$$\mu_k(q) = q(x_k) = \frac{x_l - x_k}{x_l - x_j} p_1(x_k) + \frac{x_k - x_j}{x_l - x_j} p_2(x_k) = \left(\frac{x_l - x_k}{x_l - x_j} + \frac{x_k - x_j}{x_l - x_j}\right) y_{i_k}^{(0)} = y_{i_k}^{(0)}.$$

Sei nun $k \in \mathbb{I}$ mit $\tilde{d}_k > 0$. Nach der Leibniz-Regel gilt für eine beliebige, hinreichend oft stetig differenzierbare Funktion f

$$\frac{d^s}{dx^s}\left(\frac{x_l - x}{x_l - x_j} f(x)\right) = \frac{x_l - x}{x_l - x_j} f^{(s)}(x) - \frac{s}{x_l - x_j} f^{(s-1)}(x).$$

Wegen $p_1^{(\tilde{d}_k - 1)}(x_k) = p_2^{(\tilde{d}_k - 1)}(x_k) = y_{i_k}^{(\tilde{d}_k - 1)}$ gilt

$$\begin{aligned}\mu_k(q) =& \frac{x_l - x_k}{x_l - x_j} p_1^{(\tilde{d}_k)}(x_k) - \frac{\tilde{d}_k}{x_l - x_j} p_1^{(\tilde{d}_k - 1)}(x_k) + \frac{x_k - x_j}{x_l - x_j} p_2^{(\tilde{d}_k)}(x_k) + \frac{\tilde{d}_k}{x_l - x_j} p_2^{(\tilde{d}_k - 1)}(x_k) \\ =& \frac{x_l - x_k}{x_l - x_j} p_1^{(\tilde{d}_k)}(x_k) + \frac{x_k - x_j}{x_l - x_j} p_2^{(\tilde{d}_k)}(x_k) = y_{i_k}^{(\tilde{d}_k)}.\end{aligned}$$

□

Die Newton-Darstellung aus Satz 3.8

$$p = \sum_{j=0}^{n} y[x_0, ..., x_j] N_j.$$

lässt sich auf die gleiche Art auch für das Hermite-Interpolationspolynom beweisen. Einzig die Dividierten Differenzen sind für zusammenfallende Knoten neu zu definieren. Kor. 3.12 gibt dazu bereits einen Hinweis.

Lemma 3.16. *Seien $x_0 \le x_1 \le \cdots \le x_n$ und sei $y[x_j, ..., x_{j+m}]$ mit $j, m \ge 0$, $j+m \le n$ der führende Koeffizient des Hermite-Interpolationspolynoms $p_{x_j,...,x_{j+m}} \in \Pi_m$ wie im vorigen Lemma. Dann gilt:*

$$y[x_j, \ldots, x_{j+m}] = \begin{cases} \frac{1}{m!} y_{i_j}^{(m)}, & \text{falls } x_j = \cdots = x_{j+m} \\ \frac{y[x_{j+1},...,x_{j+m}] - y[x_j,...,x_{j+m-1}]}{x_{j+m} - x_j}, & \text{falls } x_j < x_{j+m} \end{cases}. \tag{3.13}$$

Beweis. Die Aussage folgt durch Koeffizientenvergleich des führenden Koeffizienten zum einen aus dem letzten Lemma und zum anderen daraus, dass die Lösung des Hermite-Interpolationsproblems für $x_j = \cdots = x_{j+m}$ gegeben ist durch das Taylor-Polynom

$$p_{x_j,...,x_{j+m}}(x) = \sum_{k=0}^{m} \frac{(x - x_j)^k}{k!} y_{i_j}^{(k)}.$$

□

3.1.4 Interpolationsfehler

Häufig ist $y_j = f(x_j)$, $j = 0, ..., n$ das Bild der Stützstellen x_j unter einer hinreichend glatten Funktion f. Im Folgenden werden wir den Interpolationsfehler $p - f$ der Lagrange-Interpolationsaufgabe 3.2 untersuchen. Die Aussagen lassen sich jedoch komplett analog für die Hermite-Interpolationsaufgabe 3.13 nachweisen.

Lemma 3.17. *Besitzt eine Funktion $g \in C^n([a, b])$ mindestens $n+1$ Nullstellen (der Vielfachheit nach gezählt), so besitzt die Ableitung g' mindestens n Nullstellen.*

Beweis. Seien $x_0 < x_1 < \cdots < x_m$ die Nullstellen von g mit den Vielfachheiten $\mu_j \geq 1$ und $\sum_{j=0}^{m} \mu_j = n + 1$. Dann besitzt g' nach dem Satz von Rolle in jedem offenen Intervall (x_j, x_{j+1}) $(j = 0, \ldots, m - 1)$ mindestens eine Nullstelle. Weiterhin bleiben bei g' die Nullstellen t_j von g mit $\mu_j \geq 2$ mit um eins reduzierter Vielfachheit erhalten. Somit erhalten wir $m + \sum_{j=0}^{m}(\mu_j - 1) = n$ Nullstellen (der Vielfachheit nach gezählt) von g'. □

Satz 3.18. *Sei $f : [a, b] \to \mathbb{R}$ $(n + 1)$-mal stetig differenzierbar und p die Lösung des Lagrange-Interpolationsproblems 3.2 mit $y_j = f(x_j)$, $j = 0, ..., n$. Dann gibt es zu jedem $x \in [a, b]$ einen Punkt $\xi_x \in [a, b]$, so dass*

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{j=0}^{n}(x - x_j). \tag{3.14}$$

Weiter gilt für $x \in [a, b] \setminus \{x_0, ..., x_n\}$

$$f(x) - p(x) = f[x_0, \ldots, x_n, x] \prod_{j=0}^{n}(x - x_j), \tag{3.15}$$

wobei $f[x_0, \ldots, x_n, x] = y[x_0, ..., x_n, x]$ die in (3.10) definierten Dividierten Differenzen zu den Stützwerten $f(x_0), \ldots, f(x_n), f(x)$ sind.

Beweis. Da (3.14) für $x \in \{x_0, \ldots, x_n\}$ trivialerweise erfüllt ist, brauchen wir uns nur um den Fall zu kümmern, dass $x \notin \{x_0, \ldots, x_n\}$. Für den ersten Teil definieren wir

$$l(t) := \prod_{j=0}^{n}(t - x_j), \qquad c(x) := \frac{f(x) - p(x)}{l(x)}.$$

Die Funktion $F_x(t) := f(t) - p(t) - c(x)l(t)$ besitzt dann mindestens $n+2$ Nullstellen $x_0, ..., x_n, x \in [a, b]$ und ist $n + 1$ mal stetig differenzierbar. Mit Hilfe des vorigen Satzes besitzt $F_x^{(n+1)}$ eine Nullstelle $\xi_x \in [a, b]$ und die erste Behauptung folgt aus

$$F_x^{(n+1)}(t) = f^{(n+1)}(t) - c(x)(n + 1)!.$$

Den zweiten Teil zeigen wir induktiv über $n \geq 0$. Aus der Definition der Dividierten Differenzen folgt sofort der Induktionsanfang

$$f(x) - p_0(x) = f(x) - f(x_0) = f[x_0, x](x - x_0).$$

Sei nun die Behauptung richtig für $n-1 \geq 0$. Dann ist

$$\begin{aligned} f(x) - p_n(x) =& f(x) - \sum_{l=0}^{n} f[x_0, \ldots, x_l] \prod_{j=0}^{l-1}(x - x_j) \\ =& f(x) - p_{n-1}(x) - f[x_0, \ldots, x_n] \prod_{j=0}^{n-1}(x - x_j) \\ =& f[x_0, \ldots, x_{n-1}, x] \prod_{j=0}^{n-1}(x - x_j) - f[x_0, \ldots, x_n] \prod_{j=0}^{n-1}(x - x_j) \\ =& \frac{f[x_0, \ldots, x_{n-1}, x] - f[x_0, \ldots, x_n]}{x - x_n} \prod_{j=0}^{n}(x - x_j). \end{aligned}$$

Damit folgt die Behauptung, da die Dividierten Differenzen invariant gegenüber Permutation der Reihenfolge der Stützstellen sind. □

Korollar 3.19. *Sei $L_n : C^n([a,b]) \to \Pi_n$ die Abbildung, welche einer Funktion f das Lagrange-Interpolationspolynom nach 3.2 zuordnet und N_{n+1} das $(n+1)$-te Newton-Basispolynom (3.4). Dann gilt*

$$\|f - L_n f\|_\infty \leq \frac{\|N_{n+1}\|_\infty}{(n+1)!} \|f^{(n+1)}\|_\infty. \tag{3.16}$$

Beispiel 3.20. *Wir betrachten den* linearen Interpolationsoperator

$$(L_1 f)(x) := \frac{1}{h}[f(x_0)(x_1 - x) + f(x_1)(x - x_0)]$$

mit Schrittweite $h = x_1 - x_0 > 0$. Das Polynom $N_2(x) := (x - x_0)(x - x_1)$ erfüllt die Abschätzung $\|N_2\|_\infty \leq h^2/4$. Deshalb gilt für zweimal stetig differenzierbare Funktionen $f : [x_0, x_1] \to \mathbb{R}$ die Fehlerabschätzung

$$\|f - L_1 f\|_\infty \leq \frac{h^2}{8} \|f''\|_\infty.$$

3.1.5 Chebyshev-Polynome

Die Fehlerabschätzung (3.16) beinhaltet den von der Funktion f unabhängigen Term

$$\frac{\|N_{n+1}\|_\infty}{(n+1)!} = \frac{\sup_{x \in [a,b]} \prod_{j=0}^{n} |x - x_j|}{(n+1)!}.$$

Es liegt nahe, die (paarweise disjunkten) Stützstellen $x_0, \ldots, x_n \in [a,b]$ so zu wählen, dass der Zähler minimal wird.

Definition 3.21 (Chebyshev-Polynome)**.** *Für $n \in \mathbb{N}_0$ sind die* Chebyshev-Polynome *T_n definiert durch*

$$T_n(x) := \cos(n \arccos x), \qquad x \in [-1, 1]. \tag{3.17}$$

Lemma 3.22. *Für die Chebyshev-Polynome T_N gelten folgende Aussagen.*

1. $T_n(\cos\theta) = \cos(n\theta)$, $\theta \in [0,\pi]$, $n \in \mathbb{N}_0$.
2. *Für $x \in [-1,1]$ gilt die Rekursion*

$$T_0(x) \equiv 1, \qquad T_1(x) = x, \qquad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1. \tag{3.18}$$

3. $T_n \in \Pi_n$ *und der führende Koeffizient von T_n ist für $n \geq 1$ gleich 2^{n-1}.*
4. $\|T_n\|_\infty = 1$.
5. T_n *besitzt im Intervall $[-1,1]$ genau $n+1$ lokale Extrema $s_j^{(n)}$*

$$T_n(s_j^{(n)}) = (-1)^j \qquad \textit{mit} \qquad s_j^{(n)} := \cos\left(\frac{j\pi}{n}\right), \qquad j = 0,\dots,n.$$

6. T_n *besitzt im Intervall $[-1,1]$ genau n einfache Nullstellen $t_j^{(n)}$*

$$T_n(t_j^{(n)}) = 0 \qquad \textit{mit} \qquad t_j^{(n)} := \cos\left(\frac{(2j-1)\pi}{2n}\right), \qquad j = 1,\dots n.$$

Beweis. 1 und 4 folgen direkt aus der Definition, 5 und 6 aus 1 und 3 induktiv aus 2. Es bleibt also nur noch, die 2-Term-Rekursion 2 zu zeigen. Die Startwerte folgen dabei direkt aus der Definition, die Rekursion mit dem Additionstheorem

$$2\cos\left(\frac{\alpha-\beta}{2}\right)\cos\left(\frac{\alpha+\beta}{2}\right) = \cos\alpha + \cos\beta$$

mit $\alpha = (n+1)\theta$ und $\beta = (n-1)\theta$ unter Verwendung von 1 und $x = \cos\theta$. □

Satz 3.23. *Die $n+1$ Nullstellen $t_1^{(n+1)},\dots,t_{n+1}^{(n+1)}$ heißen* Chebyshev-Knoten *der Ordnung n bezüglich $[-1,1]$ und minimieren $\|N_{n+1}\|_\infty$, d.h.*

$$\min_{\xi_0,\dots,\xi_n\in[-1,1]} \max_{x\in[-1,1]} \prod_{j=0}^{n} |x-\xi_j| = \max_{x\in[-1,1]} \prod_{j=0}^{n} |x - t_{j+1}^{(n+1)}| = \frac{1}{2^n}. \tag{3.19}$$

Beweis. Wir zeigen zunächst die letzte Gleichung in (3.19). Diese folgt aus dem letzten Lemma mit (4), da wegen (3) und (6) gilt $T_{n+1}(x) = 2^n \prod_{j=0}^{n}(x - t_{j+1}^{(n+1)})$. Für die erste Gleichung beweisen wir mit einem Widerspruchsbeweis

$$\inf_{\xi_0,\dots,\xi_n\in[-1,1]} \max_{x\in[-1,1]} \prod_{j=0}^{n} |x-\xi_j| \geq \frac{1}{2^n}.$$

Daraus folgt dann die Behauptung.

Sei also im Widerspruch zur Behauptung $\xi_0, \dots, \xi_n \in [-1,1]$ so, dass $\|N_{n+1}\|_\infty < 1/2^n$. Wir definieren das Polynom $p \in \Pi_{n+1}$ mit $p := \frac{1}{2^n} T_{n+1} - N_{n+1}$. Es gilt $p \in \Pi_n$, denn $\frac{1}{2^n} T_{n+1}$ und N_{n+1} haben führenden Koeffizienten 1. Ferner hat p mindestens $n+1$ Vorzeichenwechsel, also $n+1$ Nullstellen, denn wegen (5) und der Annahme $\|N_{n+1}\|_\infty < 1/2^n$ gilt

$$\frac{1}{2^n} T_{n+1}\big(s_j^{(n+1)}\big) = \frac{(-1)^j}{2^n}, \quad \big|N_{n+1}\big(s_j^{(n+1)}\big)\big| < \frac{1}{2^n}, \qquad j = 0, \dots, n+1.$$

Da $p \in \Pi_n$ also $n+1$ paarweise disjunkte Nullstellen hat, folgt $p \equiv 0$ im Widerspruch zu $p\big(s_j^{(n+1)}\big) \neq 0$. □

Der vorige Satz lässt sich auf Interpolationen auf beliebigen Intervallen $[a,b]$ verallgemeinern. Dazu definieren wir die affin lineare Abbildung $\Psi : [-1,1] \to [a,b]$ durch

$$\Psi(\xi) := \frac{1}{2}\left(a + b + \xi(b-a)\right) \tag{3.20}$$

und die Chebyshev-Knoten der Ordnung n bezüglich $[a,b]$ durch $\Psi\left(t_1^{(n+1)}\right), \dots, \Psi\left(t_{n+1}^{(n+1)}\right)$. Wir erhalten somit

$$\min_{\xi_0,\dots,\xi_n \in [a,b]} \max_{x \in [a,b]} \prod_{j=0}^{n} |x - \xi_j| = \max_{x \in [a,b]} \prod_{j=0}^{n} \left|x - \Psi\left(t_{j+1}^{(n+1)}\right)\right| = \left(\frac{b-a}{2}\right)^{n+1} \frac{1}{2^n}. \tag{3.21}$$

Daraus ergibt sich für hinreichend glatte Funktionen f mit (3.16) bei der Benutzung von Chebyshev-Knoten die Fehlerabschätzung

$$\|f - L_n f\|_\infty \leq \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} \|f^{(n+1)}\|_\infty.$$

Falls $f \in C^\infty([a,b])$, d.h. falls f unendlich oft stetig differenzierbar und alle Ableitungen beschränkt sind, folgt somit bei Benutzung der Chebyshev-Knoten für $n \to \infty$ exponentielle Konvergenz der Interpolationspolynome gegen die Funktion f. Man kann sogar zeigen, dass für Lipschitz-stetige Funktionen f die Folge der Interpolationspolynome bei Benutzung der Chebyshev-Knoten gleichmäßig gegen f konvergiert.

3.2 Spline-Interpolation

Im letzten Abschnitt haben wir Funktionen durch globale Polynome interpoliert und gesehen, dass die Interpolationspolynome für Funktionen $f \in C^\infty([a,b])$ und $n \to \infty$ gleichmäßig gegen f konvergieren. Falls $\|f^{(n)}\|_\infty \to \infty$ für $n \to \infty$, so wird das im Allgemeinen nicht mehr der Fall sein. Man kann sogar zeigen, dass für jede Folge von Stützstellen eine Funktion f existiert, sodass $\liminf_{n\to\infty} \|f - L_n f\|_\infty > 0$. Zudem ist die Lagrange-Interpolationsaufgabe nicht-lokal, d.h. eine Änderung eines Stützwertes verändert das gesamte Polynom und hat somit global auf dem Intervall $[a,b]$ Auswirkungen. Abhilfe schafft die stückweise polynomielle Interpolation.

Definition 3.24. *Sei $\Delta = (x_0, \dots, x_n)$ eine Zerlegung von $[a, b]$, d.h. $a = x_0 < x_1 < \cdots < x_n = b$. Zu gegebenen $k, r \in \mathbb{N}_0$ heißt eine Abbildung $s : [a, b] \to \mathbb{K}$* Spline vom Grad k und Glattheit r *bezüglich Δ, falls $s|_{[x_{j-1},x_j]} \in \Pi_k$ für $j = 1, \dots, n$ und $s \in C^r([a, b])$ gilt. Wir verwenden im Folgenden die Schreibweise $s \in \mathbb{S}_r^k(\Delta)$. Im Spezialfall $r = k - 1$ schreiben wir $\mathbb{S}^k(\Delta) := \mathbb{S}_{k-1}^k(\Delta)$.*

Beispiel 3.25 (Stückweise lineare Interpolation)**.** *Im einfachsten Fall $k = 1$ und $r = 0$ wird eine gegebene Funktion $f : [a, b] \to \mathbb{R}$ durch einen linearen Polygonzug in den Stützstellen $x_0, \dots, x_n$ interpoliert, d.h. wir suchen $s \in \mathbb{S}^1(\Delta)$ mit*

$$s(x_j) = f(x_j), \qquad j = 0, ..., n.$$

Die (eindeutige) Lösung dieses Problems ist gegeben durch

$$s(x) = \frac{f(x_{j-1})(x - x_j) - f(x_j)(x - x_{j-1})}{x_{j-1} - x_j}, \qquad x \in [x_{j-1}, x_j]. \tag{3.22}$$

Der Fehler kann nach Beispiel 3.20 für hinreichend glattes f abgeschätzt werden durch

$$\|f - s\|_\infty \leq \frac{h^2}{8} \|f''\|_\infty,$$

wobei $h := \max_{j=1}^n |x_j - x_{j-1}|$ die maximale Schrittweite ist.

Die stückweise lineare Interpolation ist lokal, d.h. Änderungen an einem Stützwert $f(x_j)$ haben nur Auswirkungen auf die beiden benachbarten Intervalle $[x_{j-1}, x_{j+1}]$. Analog zu den Lagrangeschen Basispolynomen können die sogenannten *Knotenbasisfunktionen* $\varphi_j \in \mathbb{S}^1(\Delta)$ (manchmal auch *Hutfunktionen* oder *Dachfunktionen*) durch

$$\varphi_j(x_l) = \delta_{j,l}, \qquad j = 0, \dots, n, \quad l = 0, \dots, n,$$

definiert werden. Damit ergibt sich die stückweise lineare Interpolierende zu

$$s = \sum_{j=0}^{n} f(x_j)\varphi_j.$$

Die Funktionen aus $\mathbb{S}^1(\Delta)$ sind global stetig. Höhere Glattheit ist mit stückweise linearen Polynomen nicht zu erreichen.

3.2.1 Stückweise kubische Interpolation

In diesem Abschnitt betrachten wir stückweise kubische Polynome s auf einer Zerlegung Δ mit $a = x_0 < \cdots < x_n = b$, d.h.

$$s|_{[x_{j-1},x_j]}(x) = a_0^{(j)} + a_1^{(j)} x + a_2^{(j)} x^2 + a_3^{(j)} x^3, \qquad x \in [x_{j-1}, x_j], \quad j = 1, \dots, n. \tag{3.23}$$

Wir erhalten so $4n$ unbekannte Koeffizienten. Eine Forderung für die stückweise kubische Interpolierende s in diesem Abschnitt ist, dass sie an den Stützstellen die hinreichend glatte Funktion f interpoliert, d.h.

$$s(x_j) = f(x_j), \qquad j = 0, \dots, n. \tag{3.24}$$

Somit haben wir $2n$ Gleichungen für die $4n$ unbekannten Koeffizienten $a_l^{(j)}$, $l = 0, \dots, 3$, $j = 1, \dots, n$. Die restlichen Bedingungen erhalten wir üblicherweise aus Glattheitsanforderungen an s:

1. Suche $s \in \mathbb{S}_0^3(\Delta)$ mit (3.24) und

   $$s(x_j^{(l)}) = f(x_j^{(l)}), \qquad j = 1, \dots n, \quad l = 1, 2,$$

   mit zwei weiteren Stützstellen $x_j^{(l)} \in (x_{j-1}, x_j)$, $l = 1, 2$, in jedem Intervall.

2. Suche $s \in \mathbb{S}_1^3(\Delta)$ mit (3.24) und

   $$s'(x_j) = f'(x_j), \qquad j = 0, \dots, n.$$

   Dies entspricht einer stückweisen Hermite-Interpolation.

3. Suche $s \in \mathbb{S}_2^3(\Delta)$ mit (3.24) und 2 weiteren Bedingungen an s. Dies ist die *Spline-Interpolation*, welche im Folgenden näher ausgeführt wird.

Für die beiden ersten Fälle ergibt sich die eindeutige Lösbarkeit bereits aus dem letzten Abschnitt und wir erhalten direkt aus (3.16) die Fehlerabschätzung

$$\|f - s\|_\infty \leq \frac{h^4}{4!} \|f^{(4)}\|_\infty. \tag{3.25}$$

Die Lösung ist wie bei der stückweisen linearen Interpolation lokal.

Bei der Spline-Interpolation geben wir die Forderung nach Lokalität zu Gunsten einer höheren Glattheit von s auf. Wir erhalten wegen $s \in C^2([a, b])$ zusätzlich zu (3.24) die Gleichungen

$$\begin{aligned} s'|_{[x_{j-1}, x_j]}(x_j) &= s'|_{[x_j, x_{j+1}]}(x_j) \\ s''|_{[x_{j-1}, x_j]}(x_j) &= s''|_{[x_j, x_{j+1}]}(x_j) \end{aligned}, \qquad j = 1, \dots n - 1. \tag{3.26}$$

Die fehlenden zwei Bedingungen sind üblicherweise

1. *Hermite*-Randbedingungen ($s'(a)$ und $s'(b)$ werden vorgegeben),

2. *natürliche* Randbedingungen ($s''(a) = s''(b) = 0$) oder

3. *periodische* Randbedingungen($s'(a) = s'(b)$ und $s''(a) = s''(b)$).

Satz 3.26. *Sei Δ mit $a = x_0 < \dots < x_n = b$ eine Zerlegung und $f : [a, b] \to \mathbb{R}$ hinreichend glatt. Dann existiert genau ein interpolierender kubischer Spline $s \in \mathbb{S}_2^3(\Delta)$, welcher (3.24) und die Randbedingungen (Hermite, natürlich oder periodisch) erfüllt.*

Beweis. (3.24), (3.26) und die 2 Randbedingungen ergeben ein lineares Gleichungssystem mit $4n$ Gleichungen für die $4n$ unbekannten Koeffizienten $a_l^{(j)}$, $l = 0, \dots, 3$, $j = 0, \dots, n$. Eindeutigkeit der Lösung ist somit äquivalent zur Existenz einer Lösung.

Seien also s_1, s_2 zwei interpolierende kubische Splines, welche den obigen Bedingungen genügen und $s := s_1 - s_2$. Weiter sei die Menge N definiert durch

$$\begin{aligned}
N :=& \{w \in C^2([a,b]) \mid w(x_j) = 0,\ j = 0, \dots, n\}, && \text{(natürl. RB)}\\
N :=& \{w \in C^2([a,b]) \mid w(x_j) = 0,\ j = 0, \dots, n,\ w'(a) = w'(b) = 0\}, && \text{(Hermite-RB)}\\
N :=& \{w \in C^2([a,b]) \mid w(x_j) = 0,\ j = 0, \dots, n,\ w'(b) = w'(a)\}. && \text{(period. RB)}
\end{aligned}$$

Nach Voraussetzung gilt $s \in N$. Für beliebiges $w \in N$ erhalten wir

$$\begin{aligned}
\int_a^b s''(x)w''(x)dx =& \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} s''(x)w''(x)dx\\
=& \sum_{j=1}^{n} \left((s''(x)w'(x) - s'''(x)w(x)) \Big|_{x=x_{j-1}}^{x=x_j} + \int_{x_{j-1}}^{x_j} s^{(4)}(x)w(x)dx \right)\\
=& s''(b)w'(b) - s''(a)w'(a) = 0.
\end{aligned}$$

Für $w = s \in N$ ist somit $\int_a^b |s''(x)|^2 dx = 0$, d.h. s ist linear. Wegen $s(a) = s(b) = 0$ folgt $s \equiv 0$. □

Satz 3.27. *Für den natürlichen, interpolierenden kubischen Spline $s \in \mathbb{S}_2^3(\Delta)$ gilt*

$$\int_a^b |s''(x)|^2 dx \le \int_a^b |g''(x)|^2 dx \tag{3.27}$$

bezüglich aller anderen Funktionen $g \in C^2([a,b])$ mit $g(x_j) = f(x_j)$ für $j = 0, \dots, n$.

Beweis. Sei N wie im letzten Beweis. Jede interpolierende Funktion g kann in der Form $g = s + w$ mit $w \in N$ geschrieben werden. Damit folgt die Behauptung aus

$$\int_a^b |g''(x)|^2 dx = \int_a^b |s''(x)|^2 dx + 2 \underbrace{\int_a^b s''(x)w''(x)dx}_{=0,\ \text{s.o.}} + \underbrace{\int_a^b |w''(x)|^2 dx}_{\ge 0}.$$

□

Der letzte Satz ist der Grund für den Namen Spline (zu deutsch Biegestab). Das Integral $\int_a^b |g''(x)|^2 dx$ steht für die Energie, die in einem gebogenen Stab enthalten ist. Der interpolierende Spline ist die Funktion aus $C^2([a,b])$, welche minimale Energie aufweist und damit die Form, welche ein eingespannter Stab annimmt.

Satz 3.28. *Sei $s \in \mathbb{S}_2^3(\Delta)$ der interpolierende, natürliche Spline in der Darstellung*

$$s|_{[x_{j-1},x_j]}(x) = a_0^{(j)} + a_1^{(j)}(x - x_j) + a_2^{(j)}(x - x_j)^2 + a_3^{(j)}(x - x_j)^3, \qquad j = 1, \dots, n,$$

$y_j := f(x_j)$, $j = 0, \dots, n$ *und* $h_j := x_j - x_{j-1}$, $j = 1, \dots, n$.

Dann ist $a_2^{(n)} = 0$, *die Koeffizienten* $a_2^{(1)}, \dots, a_2^{(n-1)}$ *sind Lösungen des linearen Gleichungssystems*

$$\begin{pmatrix} 2(h_1+h_2) & h_2 & & \\ h_2 & 2(h_2+h_3) & \ddots & \\ & \ddots & \ddots & h_{n-1} \\ & & h_{n-1} & 2(h_{n-1}+h_n) \end{pmatrix} \begin{pmatrix} a_2^{(1)} \\ \vdots \\ a_2^{(n-1)} \end{pmatrix} = 3 \begin{pmatrix} \frac{y_2-y_1}{h_2} - \frac{y_1-y_0}{h_1} \\ \vdots \\ \frac{y_n-y_{n-1}}{h_n} - \frac{y_{n-1}-y_{n-2}}{h_{n-1}} \end{pmatrix} \tag{3.28}$$

und es gilt (zur Vereinfachung setzen wir $a_2^{(0)} := 0$*)*

$$\begin{aligned} a_0^{(j)} &= y_j, && j = 1, \dots, n, \\ a_1^{(j)} &= \frac{y_j - y_{j-1}}{h_j} + \frac{h_j}{3}\left(2a_2^{(j)} + a_2^{(j-1)}\right), && j = 1, \dots, n, \\ a_3^{(j)} &= \frac{a_2^{(j)} - a_2^{(j-1)}}{3h_j}, && j = 1, \dots, n. \end{aligned}$$

Beweis. Nachrechnen. □

Die Matrix des linearen Gleichungssystems (3.28) ist symmetrisch und strikt diagonaldominant. Wir werden später geeignete Lösungsverfahren für solche (einfachen) Gleichungssysteme kennenlernen. Abschließend noch ein Satz ohne Beweis.

Satz 3.29. *Sei* $f \in C^4([a,b])$ *und* s *der interpolierende, kubische Spline mit* $s''(a) = f''(a)$ *und* $s''(b) = f''(b)$. *Dann gilt*

$$\|f - s\|_\infty \leq \frac{1}{2} h^4 \|f^{(4)}\|_\infty.$$

Beweis. Siehe z.B. Werner & Schaback, *Praktische Mathematik II*, Springer. □

3.3 Trigonometrische Interpolation

Eine Funktion $f : \mathbb{R} \to \mathbb{K}$ heißt *periodisch mit Periodenlänge* $T > 0$, falls

$$f(t+T) = f(t) \qquad \text{für alle } t \in \mathbb{R}.$$

Zur Interpolation von periodischen Funktionen ist es naheliegend, ebenfalls periodische Funktionen zu verwenden. Die einfachste und wichtigste Klasse solcher Funktionen sind trigonometrische Polynome. Wir werden im folgenden ohne Beschränkung der Allgemeinheit annehmen, dass $T = 2\pi$.

3.3.1 Trigonometrische Polynome

Definition 3.30. *Für $n \in \mathbb{N}_0$ bezeichnen wir mit $\mathcal{T}_n$ den Raum der* trigonometrischen Polynome

$$q(t) = \frac{a_0}{2} + \sum_{k=1}^{n} (a_k \cos kt + b_k \sin kt)$$

mit reellen (oder komplexen) Koeffizienten $a_0, \ldots, a_n$ und $b_1, \ldots, b_n$. Wir sagen, ein trigonometrisches Polynom $q \in \mathcal{T}_n$ hat Grad n, falls $|a_n| + |b_n| > 0$.

Trigonometrische Polynome lassen sich auch in der Form

$$q(t) = \sum_{k=-n}^{n} c_k e^{ikt}$$

schreiben, wobei die Koeffizienten c_k aufgrund der Eulerschen Identität $e^{it} = \cos t + i \sin t$ gegeben sind durch

$$c_k := \frac{1}{2}(a_k - ib_k), \qquad c_{-k} := \frac{1}{2}(a_k + ib_k), \quad k = 0, \ldots, n.$$

Dabei haben wir $b_0 := 0$ gesetzt.

Wir betrachten im folgenden das Skalarprodukt

$$(f, g)_{L^2} := \int_0^{2\pi} f(t)\overline{g(t)}\, dt$$

für $f, g \in L^2([0, 2\pi])$ mit zugehöriger Norm $\|f\|_{L^2} := \sqrt{(f, f)_{L^2}}$. ($L^2([0, 2\pi])$ bezeichnet den Raum der im Lebesgueschen Sinne quadrat-integrierbaren Funktionen auf $[0, 2\pi]$. Der mit der Lebesgueschen Integrationstheorie nicht vertraute Leser möge sich im folgenden $L^2([0, 2\pi])$ durch $C([0, 2\pi])$ ersetzt denken. Es gilt $C([0, 2\pi]) \subset L^2([0, 2\pi])$.)

Man überprüft leicht, dass die Funktionen $\cos kt$, $k = 0, 1, \ldots$ und $\sin kt$, $k = 1, 2, \ldots$ paarweise orthogonal bezüglich des Skalarprodukts $(\cdot, \cdot)_{L^2}$ sind. Da Orthogonalsysteme linear unabhängig sind, besitzt der Vektorraum $\mathcal{T}_n$ die Dimension $2n + 1$. Ebenso sind auch die Funktionen e^{ikt}, $k \in \mathbb{Z}$ orthogonal bezüglich $(\cdot, \cdot)_{L^2}$.

Unter den Fourierkoeffizienten einer Funktion $f \in L^2([0, 2\pi])$ versteht man die Zahlen

$$\hat{f}(k) := \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-ikt}\, dt \qquad k \in \mathbb{Z}.$$

Die große Bedeutung von Fourierreihen in Mathematik, Technik und Naturwissenschaften ist u.a. durch den folgenden Satz begründet. Er besagt, dass man eine Funktion f aus ihren Fourierkoeffizienten $\hat{f}(k)$ über eine Fourierreihe wieder zurückgewinnen kann. Da wir diesen Satz in dieser Vorlesung nicht benötigen werden, zitieren wir ihn ohne Beweis (vgl. etwa Forster, *Analysis 1*, 1983).

Satz 3.31. *Sei $f \in L^2([0,2\pi])$ und sei $S_n(t) := \sum_{k=-n}^{n} \hat{f}(k)e^{ikt}$ die n-te Partialsumme der Fourierreihe von f für $n \in \mathbb{N}$. Dann gilt*

$$\lim_{n\to\infty} \|f - S_n\|_{L^2} = 0.$$

Ist f stetig differenzierbar, so konvergieren die Partialsummen sogar gleichmäßig, und es gilt für alle $t \in [0,2\pi]$

$$f(t) = \sum_{k=-\infty}^{\infty} \hat{f}(k)e^{ikt}.$$

Weiterhin gilt für alle $f \in L^2([0,2\pi])$ die Parsevalsche Gleichung

$$\frac{1}{2\pi}\|f\|_{L^2}^2 = \sum_{k=-\infty}^{\infty} |\hat{f}(k)|^2. \tag{3.29}$$

3.3.2 Existenz und Eindeutigkeit

Satz 3.32. *Sei $N \in \mathbb{N}$, $m \in \mathbb{Z}$ und seien $t_0, \dots, t_{N-1} \in [0,2\pi)$ paarweise verschiedene Stützstellen und $f_0, \dots, f_{N-1} \in \mathbb{K}$ gegebene Stützwerte. Dann gibt es genau ein trigonometrisches Polynom $q \in \operatorname{span}\{e^{imt}, e^{i(m+1)t}, \dots, e^{i(m+N-1)t}\}$, das die Interpolationsbedingungen*

$$q(t_j) = f_j, \qquad j = 0, \dots, N-1 \tag{3.30}$$

erfüllt.

Beweis. Eine Funktion $q(t) = \sum_{k=m}^{m+N-1} c_k e^{ikt}$ erfüllt genau dann die Interpolationsbedingungen (3.30), wenn die Koeffizienten c_k die Gleichungen

$$\sum_{k=m}^{m+N-1} e^{ikt_j} c_k = f_j, \qquad j = 0, \dots, N-1 \tag{3.31}$$

erfüllen. Wir müssen zeigen, dass die Matrix dieses Gleichungssystems regulär ist, also trivialen Nullraum hat. Sei $\sum_{k=m}^{m+N-1} e^{ikt_j} c_k = 0$ für $j = 0, \dots, N-1$. Wir führen die Variablensubstitution $z = e^{it}$ durch und betrachten die Funktion

$$\tilde{p}(z) := \sum_{k=m}^{m+N-1} c_k z^k.$$

Aufgrund unserer Annahme gilt $\tilde{p}(e^{it_j}) = 0$ für $j = 0, \dots, N-1$. Deshalb besitzt das Polynom $p(z) = z^{-m}\tilde{p}(z) = \sum_{k=0}^{N-1} c_{m+k} z^k$ vom Grad $\leq N-1$ die N Nullstellen e^{it_j}, $j = 0, \dots, N-1$ und somit folgt $c_m = \cdots = c_{m+N-1} = 0$. Damit haben wir gezeigt, dass die Matrix des Gleichungssystems (3.31) regulär ist, und der Beweis ist beendet. □

3.3.3 Trigonometrische Interpolation mit äquidistanten Stützstellen

Wir betrachten nun den wichtigen Spezialfall äquidistanter Stützstellen

$$t_j = \frac{2\pi j}{N}, \qquad j = 0, \dots, N-1.$$

Bemerkung 3.33. *(Periodizität der diskreten Fourierkoeffizienten) Sei $m \in \mathbb{Z}$ und sei $c_m, \dots, c_{m+N-1}$ die eindeutige Lösung der Interpolationsgleichungen*

$$\sum_{k=m}^{m+N-1} c_k e^{ikt_j} = f_j, \qquad j = 0, \dots, N-1 \tag{3.32}$$

zu gegebenen Funktionswerten $f_j \in \mathbb{C}$. Falls wir den Vektor $(c_m, c_{m+1}, \dots, c_{m+N-1})$ periodisch fortsetzen, so dass

$$c_{k+N} = c_k \qquad \text{für alle } k \in \mathbb{Z}, \tag{3.33}$$

dann sind die Interpolationsbedingungen (3.32) für jedes $m \in \mathbb{Z}$ erfüllt.

Beweis. Es gilt $c_{k+N} e^{i(k+N)t_j} = c_k e^{ikj\,2\pi/N} e^{2\pi i j N/N} = c_k e^{ikj\,2\pi/N}$. □

Wir können daher zur Berechnung der Koeffizienten c_k den Index m frei wählen und werden im folgenden der Einfachheit halber immer $m = 0$ verwenden.

Satz 3.34. *Für $N = 1, 2, \dots$ sei $\mathrm{DFT}_N \in \mathbb{C}^{N\times N}$ definiert durch*

$$\mathrm{DFT}_N := \left(e^{-ijk2\pi/N}\right)_{j,k=0,\dots,N-1}.$$

Dann gilt

$$\mathrm{DFT}_N^* \,\mathrm{DFT}_N = N \cdot I_N. \tag{3.34}$$

Insbesondere gilt $(\mathrm{DFT}_N^*)^{-1} = \frac{1}{N}\,\mathrm{DFT}_N$, und die Lösung des Gleichungssystems

$$\sum_{k=0}^{N-1} c_k e^{i\,jk\,2\pi/N} = f_j, \qquad j = 0, \dots, N-1 \tag{3.35}$$

ist gegeben durch

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} e^{-i\,jk2\pi/N} f_j, \qquad k = 0, \dots, N-1. \tag{3.36}$$

Beweis. Mit der N-ten Einheitswurzel $\omega_N := e^{2\pi i/N}$ können wir die Matrizen DFT_N und DFT_N^* schreiben als

$$\mathrm{DFT}_N = \left(\omega_N^{-jk}\right)_{j,k=0,\dots,N-1} \quad \text{und} \quad \mathrm{DFT}_N^* = \left(\omega_N^{jk}\right)_{j,k=0,\dots,N-1}.$$

Wir prüfen die Matrixgleichung (3.34) nun komponentenweise nach:

$$\sum_{l=0}^{N-1} \omega_N^{jl}\omega_N^{-lk} = \sum_{l=0}^{N-1} (\omega_N^{j-k})^l = \begin{cases} N & \text{für } j = k \\ \frac{1-\omega_N^{N(j-k)}}{1-\omega_N^{j-k}} = 0 & \text{für } j \neq k \end{cases}$$

für $j, k = 0, \dots, N-1$. Dabei haben wir die Formel für die geometrische Reihe und die Identität $\omega_N^N = 1$ ausgenutzt. Da Gl. (3.32) in Matrixform $\mathrm{DFT}_N^* c = f$ lautet mit $c = (c_k)_{k=0,\dots,N-1}$ und $f = (f_j)_{j=0,\dots,N-1}$, erhalten wir $c = (\mathrm{DFT}_N^*)^{-1} f = \frac{1}{N}\mathrm{DFT}_N f$ oder äquivalent (3.36). □

Bemerkung 3.35. *Die Bezeichnung* DFT *steht für* diskrete Fourier–Transformation, *da die durch die Matrix* DFT_N *beschriebene Abbildung* $\mathbb{C}^n \to \mathbb{C}^n$ *ein diskretes Pendant zur Fourierreihen-Transformation* $L^2([0, 2\pi]) \to l^2(\mathbb{Z})$, $f \mapsto (\hat{f}(n))_{n\in\mathbb{Z}}$ *ist. In der Tat gilt mit* $f_j = f(\frac{2\pi j}{N})$

$$\hat{f}(k) = \int_0^{2\pi} f(t)e^{-ikt}\,dt \approx \frac{1}{N}\sum_{j=0}^{N-1} e^{-i\,jk2\pi/N} f_j = c_k, \qquad k = 0, \dots, N-1,$$

wobei das Integral durch die später zu besprechende Trapezregel approximiert wurde. Aufgrund von (3.34) ist die normierte Matrix $N^{-1/2}\,\mathrm{DFT}_N$ *unitär, also norm-erhaltend. Deshalb gilt in Analogie zur Parsevalschen Gleichung*

$$\frac{1}{N}\sum_{j=0}^{N-1} |f_j|^2 = \sum_{k=0}^{N-1} |c_k|^2.$$

Es sei darauf hingewiesen, dass das Interpolationspolynom $p_N(t) = \sum_{k=m}^{m+N-1} c_k e^{ikt}$ für $t \notin \{t_0, \dots, t_{N-1}\}$ sehr wohl von m abhängt und dass für $m = 0$ i.a. keine Konvergenz des Interpolationspolynoms p_N gegen f für $N \to \infty$ vorliegt. So gilt etwa mit $f(t) = e^{-it}$ für alle $p_N \in \mathrm{span}\{e^{i0t}, e^{it}, \dots, e^{i(N-1)t}\}$ die Ungleichung

$$\|f - p_N\|_{L^2}^2 = \|f\|_{L^2}^2 + \|p_N\|_{L^2}^2 + 2\mathrm{Re}\,(f, p_N)_{L^2} = \|f\|_{L^2}^2 + \|p_N\|_{L^2}^2 \geq \|f\|_{L^2}^2.$$

Die besten Approximationseigenschaften erhält man, wenn man die Indizes der Koeffizienten c_k symmetrisch um 0 verteilt. Hat man also den Koeffizientenvektor $(c_0, \dots, c_{N-1})$ nach (3.36) berechnet, sollte man anschließend unter Benutzung der Periodizität (3.33) die höchsten Koeffizienten $c_{N/2}, \dots, c_{N-1}$ (für N gerade) mit $c_{-N/2}, \dots, c_{-1}$ identifizieren. In MATLAB steht dafür die Funktion fftshift zur Verfügung.

Bei einer symmetrischen Verteilung kann man statt der Exponentialfunktionen auch die Sinus- und Cosinusfunktionen als Basis nehmen. Dies hat den Vorteil, dass die Koeffizienten für reelle Funktionen wieder reell sind. Wir notieren die entsprechenden Sätze für gerade und ungerade Stützstellenzahl.

Korollar 3.36. *Sei* $n \in \mathbb{N}$ *und* $N := 2n+1$ *ungerade. Dann ist die eindeutige Lösung des Gleichungssystems*

$$\frac{a_0}{2} + \sum_{k=1}^{n} (a_k \cos kt_j + b_k \sin kt_j) = f_j, \qquad j = 0, \dots N-1 \tag{3.37}$$

mit den Stützstellen $t_j := \frac{2\pi j}{N}$ *gegeben durch*

$$a_k := \frac{2}{N}\sum_{j=0}^{N-1} f_j \cos kt_j, \qquad k = 0, \ldots, n, \tag{3.38}$$

$$b_k := \frac{2}{N}\sum_{j=0}^{N-1} f_j \sin kt_j, \qquad k = 1, \ldots, n \tag{3.39}$$

Beweis. Aufgrund des letzten Satzes und der Periodizität der Koeffizienten c_k ist die Lösung der Interpolationsgleichungen

$$\sum_{k=-n}^{n} c_k e^{ikt_j} = f_j, \qquad j = 0, \ldots, N-1,$$

gegeben durch $c_k = \frac{1}{N}\sum_{j=0}^{N-1} e^{-i\,jk2\pi/N} f_j$, $k = -n, \ldots, n$. Dabei haben wir benutzt, dass Gl. (3.36) für beliebiges $k \in \mathbb{Z}$ gerade die periodische Fortsetzung (3.33) der Koeffizienten c_k definiert. Deshalb gilt (3.37) mit $a_k = c_k + c_{-k}$ und $b_k = i(c_k - c_{-k})$, und wir erhalten

$$\begin{aligned} a_k &= \frac{1}{N}\sum_{j=0}^{N-1} f_j \left(e^{-ikt_j} + e^{ikt_j}\right) = \frac{2}{N}\sum_{j=0}^{N-1} f_j \cos kt_j, \\ b_k &= \frac{1}{N}\sum_{j=0}^{N-1} f_j \left(ie^{-ikt_j} - ie^{-ikt_j}\right) = \frac{2}{N}\sum_{j=1}^{N-1} f_j \sin kt_j. \end{aligned}$$

□

Bei einer geraden Anzahl von Stützstellen ist es offenbar nicht möglich, die Fourierkoeffizienten c_k, $k = m, \ldots, m+N-1$ symmetrisch um 0 zu verteilen. Wir benutzen hier neben den Basisfunktionen von $T_{N/2-1}$ nur die Basisfunktion $\cos(tN/2)$, da $\sin(tN/2)$ an den Stützstellen verschwindet. Für diesen Ansatzraum haben wir die Eindeutigkeit des Interpolationsproblems noch nicht geklärt.

Korollar 3.37. *Sei* $n \in \mathbb{N}$ *und* $N := 2n$ *gerade. Dann ist die das Gleichungssystem*

$$\frac{a_0}{2} + \sum_{k=1}^{n-1}(a_k \cos kt_j + b_k \sin kt_j) + \frac{a_n}{2}\cos nt_j = f_j, \qquad j = 0, \ldots N-1 \tag{3.40}$$

mit den Stützstellen $t_j := \frac{2\pi j}{N}$ *eindeutig lösbar, und die Lösung ist durch die Gleichungen (3.38) und (3.39) gegeben.*

Beweis. Wir wissen bereits, dass das Gleichungssystem

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

eindeutig lösbar ist und dass die Lösung durch $c_k = \frac{1}{N}\sum_{j=0}^{N-1} e^{-ijk2\pi/N} f_j$, $k = -n, \dots, n-1$ gegeben ist. Da aber $e^{-int_j} = e^{-ij\pi} = \cos(nt_j)$, ist auch das Gls. (3.40) eindeutig lösbar mit der Lösung $a_k = c_k + c_{-k}$, $b_k = i(c_k - c_{-k})$ für $k = 0, \dots, n-1$ und $a_n = 2c_{-n}$. Man überprüft leicht, dass dies mit (3.38) und (3.39) übereinstimmt. □

3.3.4 Die Schnelle Fourier-Transformation

Eine normale Matrix-Vektor-Multiplikation mit der Matrix DFT_N benötigt N^2 flops. Wir werden im folgenden einen Algorithmus kennenlernen, mit dem diese Aufgabe mit nur $O(N \log N)$ Rechenoperationen durchgeführt werden kann. Dieser Algorithmus heißt *Schnelle Fourier-Transformation* (engl. ***F**ast **F**ourier **T**ransform*, kurz FFT) und wurde 1965 von J.W.Cooley und J.W.Tukey angegeben. Er war jedoch bereits Gauß bekannt. Grundlage ist die folgende Faktorisierung der Matrix DFT:

Satz 3.38. *Sei $n \in \mathbb{N}$, $N := 2n$ und $\omega_N := e^{-2\pi i/N}$. Dann besitzt die diskrete Fourier-Transformation die Faktorisierung*

$$\mathrm{DFT}_N = P_N \begin{pmatrix} \mathrm{DFT}_n & 0_n \\ 0_n & \mathrm{DFT}_n \end{pmatrix} \begin{pmatrix} I_n & I_n \\ D_n & -D_n \end{pmatrix}$$

mit der Diagonalmatrix $D_n := \mathrm{diag}(\omega_N^0, \omega_N^1, \dots, \omega_N^{n-1})$ und der Permutationsmatrix $P_N := (e_1\, e_3 \cdots e_{N-1}\, e_2\, e_4 \cdots e_N)$, wobei e_j der j-te Einheitsvektor in $\mathbb{R}^N$ ist.

Beweis. Sei $f = (f_0, \dots, f_{N-1})^T \in \mathbb{C}^N$ und $c := \mathrm{DFT}_N f$. Dann gilt

$$P_N^T\, \mathrm{DFT}_N\, f = P_N^T c = (c_0, c_2, \dots, c_{N-2}, c_1, c_3, \dots, c_{N-1})^T.$$

Unter Verwendung der Identitäten

$$\begin{aligned} \omega_N^{2k(n+j)} &= \omega_N^{kN+2kj} = \omega_N^{2kj}, \\ \omega_N^{(2k+1)(n+j)} &= (\omega_N^n)^{2k+1}\omega_N^{(2k+1)j} = -(\omega_N^2)^{kj}\omega_N^j \end{aligned}$$

für $k, j \in \mathbb{Z}$ erhalten wir

$$\begin{aligned} c_{2k} &= \sum_{j=0}^{N-1} \omega_N^{2kj} f_j = \sum_{j=0}^{n-1} (\omega_N^2)^{kj}(f_j + f_{n+j}) = \sum_{j=0}^{n-1} \omega_n^{kj}(f_j + f_{n+j}) \\ c_{2k+1} &= \sum_{j=0}^{N-1} \omega_N^{(2k+1)j} f_j = \sum_{j=0}^{n-1} (\omega_N^2)^{kj}(f_j - f_{n+j})\omega_N^j = \sum_{j=0}^{n-1} \omega_n^{kj}(f_j - f_{n+j})\omega_N^j \end{aligned}$$

für $k = 0, \dots, n-1$. Damit haben wir gezeigt, dass

$$P_N^T\, \mathrm{DFT}_N\, f = \begin{pmatrix} \mathrm{DFT}_n & 0_n \\ 0_n & \mathrm{DFT}_n \end{pmatrix} \begin{pmatrix} I_n & I_n \\ D_n & -D_n \end{pmatrix} f,$$

und dies ist wegen $P_N P_N^T = I_N$ äquivalent zur behaupteten Faktorisierung. □

Wir nehmen im folgenden an, dass $N = 2^p$ eine Zweier-Potenz ist. Der vorangehende Satz erlaubt es, eine Matrix-Vektor-Multiplikation der Größe N im wesentlichen auf zwei Matrix-Vektor-Multiplikationen der halben Größe zurückzuführen. Auf diese kleineren Probleme können wir wiederum dasselbe Verfahren anwenden, usw. Es handelt sich also um eine *divide et impera*-Strategie (lat. teile und herrsche).

Komplexitätsanalysis: Sei $N = 2^p$. Wir nehmen an, dass alle Potenzen $\omega^0, \omega^1, \ldots, \omega^{N-1}$ im Vorfeld berechnet werden. Dann werden in jedem Rekursionsschritt $N/2$ Multiplikationen und N Additionen benötigt. Da es $p = \log_2(N)$ Rekursionsschritte gibt, erhalten wir einen Gesamtaufwand von

$$\frac{N}{2}\log_2(N)\,\text{kompl. Multiplikation}, \qquad N\log_2(N)\,\text{kompl. Additionen}.$$

Dies ist eine erhebliche Verbesserung gegenüber dem Aufwand von N^2 komplexen Multiplikationen und Additionen für die Matrix-Vektor-Multiplikation mit DFT_N.

Bemerkung 3.39.

1. *Falls $N = p_1^{n_1} \cdots p_l^{n_l}$ ein Produkt von beliebigen (kleinen) Primzahlen p_j ist, kann man eine analoge Faktorisierung von DFT_N und einen entsprechenden Algorithmus angeben, der die Anwendung von der Matrix DFT_N auf einen Vektor mit $O(N\log(N))$ Rechenoperationen implementiert.*

2. *Ist N nicht Produkt kleiner Primzahlen, kann man nach einigen Modifikationen die nächst größere ganze Zahl verwenden, die sich in kleine Primfaktoren zerlegen lässt. Der gegebene Vektor f ist dann durch Nullen zu ergänzen.*

3. *Der FFT-Algorithmus ist parallelisierbar, indem die rekursiven Aufrufe auf verschiedene Prozessoren verteilt werden.*

4. *Eine mit dem Wilkinson Software Preis ausgezeichnete, frei erhältliche Implementation des FFT-Algorithmus von M. Frigo und S.G. Johnson in der Programmiersprache C, die auch von MATLAB ab Version 6.0 verwendet wird, findet man unter* `http://www.fftw.org/`

3.3.5 Interpolationsfehler

Da die Abschätzung des trigonometrischen Interpolationsfehlers um einiges aufwendiger ist als bei der Polynom-Interpolation, zitieren wir hier ein Resultat ohne Beweis (vgl Hanke-Bourgeois, Grundlagen der Numerischen Mathematik und des Wissenschaflichen Rechnen, 2002):

Satz 3.40. *Sei $f \in C^s(\mathbb{R})$ 2π-periodisch ($s \in \mathbb{N}$), und sei q_N das trigonometrische Interpolationspolynom aus (3.37) für N ungerade, bzw. (3.40) für N gerade. Dann gibt es eine nur von s abhängige Konstante C_s, so dass*

$$\|f - q_N\|_{L^2} \le C_s N^{-s} \left(\int_0^{2\pi} |f(t)|^2 + |f^{(s)}(t)|^2 \, dt \right)^{1/2}$$

für alle $N \in \mathbb{N}$.

Beachte, dass für $f \in C^\infty(\mathbb{R})$ der Fehler $\|f - q_N\|_{L^2}$ schneller als jede Potenz N^s von N gegen 0 konvergiert.

3.4 Richardson-Extrapolation

Bei einem Interpolationsproblem versucht man, einen Funktionswert $f(x)$ zu approximieren, wobei der Auswertungspunkt x zwischen gegebenen Stützstellen liegt, z.B. $x \in [x_0, x_n]$ mit $x_0 < \cdots < x_n$, wobei die $f(x_k)$ bekannt sind. Die Fehlerabschätzung (3.14) zur Interpolation legt nahe, dass der Interpolationsfehler außerhalb des Interpolationsintervalls sehr groß werden kann, da $\prod_{j=0}^n (x - x_j)$ dann sehr schnell sehr groß werden kann. Von einer Extrapolation ist also eigentlich abzuraten. Im Folgenden werden wir dennoch x außerhalb aber in der Nähe des Intervalls $[x_0, x_n]$ wählen.

In diesem Abschnitt lautet die abstrakte Formulierung wie folgt.

Problem 3.41 (Richardson-Extrapolation)**.** *Gegeben sei eine stetige Funktion* $f : [0,1] \to \mathbb{K}$. *Für gewisse* $h_k \in (0,1]$ *sei* $f(h_k)$ *bekannt. Ziel ist es, eine Approximation für die nicht direkt zugängliche Größe* $f(0)$ *zu berechnen, indem eine Interpolierende* p *an Stützstellen* h_k *mit Stützwerten* $f(h_k)$ *im Punkt* 0 *ausgewertet wird.*

Beispiel 3.42 (Einseitiger Differenzenquotient)**.** *Ist* g *eine stetig differenzierbare Funktion, so betrachten wir für die numerische Differentiation den einseitigen Differenzenquotienten*

$$f(h) := \frac{g(x+h) - g(x)}{h}, \qquad h > 0, \tag{3.41}$$

um die Ableitung $\lim_{h\to 0} f(h) = g'(x)$ *an einer festen Stelle* x *zu approximieren. Ist die Funktion* g *hinreichend glatt, so zeigt Taylor-Entwicklung*

$$g(x+h) = \sum_{\ell=0}^{n+1} \frac{g^{(\ell)}(x)}{\ell!} h^\ell + \mathcal{O}(h^{n+2}),$$

und damit folgt

$$f(h) = g'(x) + \sum_{\ell=1}^{n} \frac{g^{(\ell+1)}(x)}{(\ell+1)!} h^\ell + \mathcal{O}(h^{n+1}).$$

Wir erhalten daraus eine sogenannte asymptotische Entwicklung *(bezüglich des Parameters* h*) von* $f(h)$ *für* $h \to 0$.

Beispiel 3.43 (Zentraler Differenzenquotient)**.** *Ist* g *mindestens zweimal stetig differenzierbar, so betrachten wir für die numerische Differentiation den zentralen Differenzenquotienten*

$$f(h) := \frac{g(x+h) - g(x-h)}{2h}, \qquad h > 0, \tag{3.42}$$

und erhalten analog für hinreichend glattes g

$$f(h) = g'(x) + \sum_{\ell=1}^{n} \frac{g^{(2\ell+1)}(x)}{(2\ell+1)!} h^{2\ell} + \mathcal{O}(h^{2(n+1)}).$$

Mit Hilfe der asymptotischen Entwicklungen erhalten wir mit $q := 1$ für den einseitigen Differenzenquotienten und $q := 2$ für den zentralen

$$f(0) - f(h) = \mathcal{O}(h^q), \qquad (h \to 0), \tag{3.43}$$

d.h. der Fehler an der Stelle 0 bei der Auswertung von f an einer Stelle $h > 0$ hat die gleiche Größenordnung wie h^q. Dies lässt sich wesentlich verbessern.

Satz 3.44 (Extrapolationsfehler)**.** *Für die Funktion $f : \mathbb{R}_+ \to \mathbb{K}$ sei bekannt, dass sie eine asymptotische Entwicklung der folgender Form hat:*

$$f(h) = a_0 + \sum_{\ell=1}^{n} a_\ell h^{\ell q} + a_{n+1}(h) h^{(n+1)q}, \qquad h > 0, \tag{3.44}$$

mit $q > 0$ und gewissen (unbekannten) Koeffizienten a_ℓ, $\ell = 0, \ldots, n+1$, und $a_{n+1}(h) = a_{n+1} + o(1)$ für $h \to 0$. Weiter sei $(h_k)_{k \in \mathbb{N}_0}$ eine Folge aus $\mathbb{R}_{>0}$ mit

$$0 < \frac{h_{k+1}}{h_k} \leq \rho < 1 \tag{3.45}$$

und für $n, k \in \mathbb{N}_0$ sei $p_n^{(k)} \in \Pi_n$ das interpolierende Polynom mit

$$p_n^{(k)}\left(h_{k+j}^q\right) = f\left(h_{k+j}\right), \qquad j = 0, \ldots, n. \tag{3.46}$$

Dann gilt

$$f(0) - p_n^{(k)}(0) = \mathcal{O}\left(h_k^{(n+1)q}\right), \qquad (k \to \infty). \tag{3.47}$$

Beweis. Wir setzen zur Abkürzung $z := h^q$ und $z_k := h_k^q$, $k \in \mathbb{N}_0$. Nach Satz 3.4 gilt

$$p_n^{(k)} = \sum_{j=0}^{n} f(h_{k+j}) L_{k+j}^{(n)}, \qquad L_{k+j}^{(n)}(z) := \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{z - z_{k+i}}{z_{k+j} - z_{k+i}}, \quad z \in \mathbb{R}.$$

Aus der Fehlerdarstellung (3.14) erhält man für $f(z) := z^r$

$$\sum_{j=0}^{n} z_{k+j}^r L_{k+j}^{(n)}(0) = \begin{cases} 1, & r = 0, \\ 0, & r = 1, \ldots, n \\ (-1)^n \prod_{j=0}^{n} z_{k+j}. & r = n+1 \end{cases}.$$

Damit erhalten wir

$$\begin{aligned} p_n^{(k)}(0) =& \sum_{j=0}^{n} \left(a_0 + \sum_{\ell=1}^{n} a_\ell z_{k+j}^{\ell} + a_{n+1}(h_{k+j}) z_{k+j}^{n+1} \right) L_{k+j}^{(n)}(0) \\ =& a_0 + a_{n+1}(-1)^n \prod_{j=0}^{n} z_{k+j} + \sum_{j=0}^{n} o(1) z_{k+j}^{n+1} L_{k+j}^{(n)}(0). \end{aligned}$$

Zu beachten ist, dass in der Schreibweise $o(1)$ nach Definition 2.2 eine Abhängigkeit von h_{k+j} steckt und somit $o(1)$ nicht vor die Summe gezogen werden kann. Wegen (3.45) gilt $\prod_{j=0}^{n} z_{k+j} = \mathcal{O}\left(z_k^{n+1}\right)$, $\sum_{j=0}^{n} o(1) z_{k+j}^{n+1} = o\left(z_k^{n+1}\right)$ und

$$\left| L_{k+j}^{(n)}(0) \right| = \prod_{\substack{i=0 \\ i \neq j}}^{n} \left| \frac{z_{k+i}}{z_{k+j} - z_{k+i}} \right| = \prod_{\substack{i=0 \\ i \neq j}}^{n} \left| \frac{1}{z_{k+j}/z_{k+i} - 1} \right| \leq \gamma(n, \rho) \tag{3.48}$$

mit einer Konstante $\gamma(n, \rho)$ unabhängig von k. Zusammen folgt die Behauptung. □

Es bietet sich an, das Schema von Aitken-Neville (3.8) zu verwenden. Wir definieren analog zur Notation in Lemma 3.7 $a_{k,m} := p_m^{(k)}(0)$ und erhalten für $n = 3$ und $k \geq 0$ das Schema

$$\begin{array}{lccccccc} f(h_0) = a_{0,0} & & & & & & & \\ & \searrow & & & & & & \\ f(h_1) = a_{1,0} & \to & a_{0,1} & & & & & \\ & \searrow & & \searrow & & & & \\ f(h_2) = a_{2,0} & \to & a_{1,1} & \to & a_{0,2} & & & \\ & \searrow & & \searrow & & \searrow & & \\ f(h_3) = a_{3,0} & \to & a_{2,1} & \to & a_{1,2} & \to & a_{0,3} & \\ & \searrow & & \searrow & & \searrow & & \\ f(h_4) = a_{4,0} & \to & a_{3,1} & \to & a_{2,2} & \to & a_{1,3} & \\ \vdots & & \vdots & & \vdots & & \vdots & \\ f(h_{k+3}) = a_{k+3,0} & \to & a_{k+2,1} & \to & a_{k+1,2} & \to & a_{k,3} & \end{array} \tag{3.49}$$

mit

$$\begin{aligned} a_{j,0} &= f(h_j), & & j = k, \ldots, k+n \\ a_{j,m} &= a_{j,m-1} + \frac{a_{j,m-1} - a_{j+1,m-1}}{\left(\frac{h_{j+m}}{h_j}\right)^q - 1}, & & m = 1, \ldots, n, \quad j = k, \ldots, k+n-m. \end{aligned}$$

Natürlich könnten wir im obigen Schema nicht nur $k \to \infty$ sondern auch $n \to \infty$ gehen lassen. Der vorige Satz gilt jedoch nur für festes n und wir haben bereits gesehen, dass die interpolierenden Polynome für $n \to \infty$ nicht unbedingt konvergieren müssen.

4 Numerische Integration

Wir untersuchen in diesem Abschnitt die näherungsweise Berechnung eines Integrals

$$Q(f) := \int_a^b f(x)\,dx \tag{4.1}$$

für eine stetige Funktion f auf einem Intervall $[a, b]$.

Satz 4.1 (Kondition Integration)**.** *Sei $f, \Delta f \in C([a, b])$. Dann gilt*

$$|Q(f) - Q(f + \Delta f)| \leq (b - a)\|\Delta f\|_\infty, \tag{4.2}$$

d.h. die absolute Konditionszahl der Integration ist $\kappa_{\text{abs}} := (b - a)$.

Beweis. Linearität des Integrals. □

4.1 Konstruktion von Interpolationsquadraturen

Es ist bekannt, dass sich viele Funktionen nicht exakt analytisch integrieren lassen. In diesem Fall ist man auf numerische Integration angewiesen. Eine Möglichkeit ist, auf die Definition des Riemann-Integrals zurückzugreifen:

$$Q(f) = \lim_{n\to\infty} \frac{b-a}{n} \sum_{j=0}^{n-1} f\left(x_j^{(n)}\right)$$

mit $x_j^{(n)} := a + j\frac{(b-a)}{n}$. Dabei wird die Funktion f in jedem Teilintervall $[x_j^{(n)}, x_{j+1}^{(n)}]$ durch eine konstante Funktion approximiert und dann exakt integriert. Die Genauigkeit dieser Approximation für eine gegebene Zahl n von Funktionsauswertung kann jedoch im allgemeinen erheblich verbessert werden.

Die Formeln zur numerischen Integration werden *Quadraturformeln* genannt und haben i.A. die Form

$$Q_n(f) = \sum_{j=0}^{n} \alpha_j f(x_j),$$

mit Gewichten α_j und Stützstellen x_j für $j = 0, \ldots, n$. Eine Möglichkeit zur Konstruktion solcher Quadraturformeln besteht darin, den Integranden f durch ein Interpolationspolynom $p_f \in \Pi_n$ zu den Stützstellen $x_0, \ldots, x_n$ zu ersetzen und dann exakt zu integrieren:

$$\int_a^b f(x)\,dx \approx Q_n(f) := \int_a^b p_f(x)dx.$$

Lemma 4.2. *Seien $x_0, \ldots, x_n \in [a, b]$ paarweise verschiedenen* Quadraturpunkte *und $p_f \in \Pi_n$ das interpolierende Polynom mit*

$$p_f(x_j) = f(x_j), \qquad j = 0, \ldots, n.$$

Dann ist die durch

$$Q_n(f) := \int_a^b p_f(x)dx$$

definierte Polynom-Interpolationsquadratur *von der Form*

$$Q_n(f) = \sum_{j=0}^{n} \alpha_j f(x_j) \tag{4.3}$$

mit Quadraturgewichten

$$\alpha_j = \int_a^b L_j(x)dx. \tag{4.4}$$

Dabei bezeichnet $L_j \in \Pi_n$ für $j = 0, \ldots, n$ die Lagrange-Funktion zur Stützstelle x_j (siehe (3.3)).

Beweis. Es gilt

$$\int_a^b p_f(x)\,dx = \int_a^b \sum_{j=0}^{n} f(x_j)L_j(x)\,dx = \sum_{j=0}^{n} \alpha_j f(x_j).$$

□

Satz 4.3. *Seien $x_0, \ldots, x_n$ paarweise verschiedene Punkte im Intervall $[a, b]$. Dann sind die Gewichte α_j einer in der Form (4.3) beschriebenen Polynom–Interpolationsquadraturformel Q_n eindeutig dadurch bestimmt, dass Polynome vom Grad $\leq n$ exakt integriert werden, d.h.*

$$Q_n(p) = Q(p) \qquad \text{für alle } p \in \Pi_n. \tag{4.5}$$

Beweis. Da für $p \in \Pi_n$ das Interpolierende Polynom p_p identisch mit p ist, folgt

$$Q_n(p) = \int_a^b p_p(x)dx = \int_a^b p(x)dx = Q(p),$$

d.h. die Quadraturformel ist exakt für $p \in \Pi_n$. Nehmen wir andersherum an, dass eine Quadraturformel der Form (4.3) die Bedingung (4.5) erfüllt, so folgt daraus

$$\alpha_j = Q_n(L_j) = Q(L_j) = \int_a^b L_j(x)\,dx.$$

Dies stimmt mit (4.4) überein, d.h. Q_n ist die Polynom-Interpolationsquadratur. □

Definition 4.4 (Ordnung Quadraturformel)**.** *Eine Quadraturformel Q_n hat die* Ordnung *m, wenn durch sie alle Polynome aus Π_{m-1} exakt integriert werden.*

Mit Hilfe des vorigen Satzes sind interpolatorische Quadraturformeln mit $n+1$ disjunkten Stützstellen mindestens von der Ordnung $n+1$.

Wir werden im Folgenden drei Sorten von Quadraturformeln untersuchen, die alle auf Interpolationen beruhen:

1. Quadraturformeln auf Basis von Lagrange-Interpolationen mit äquidistanten Stützstellen,
2. zusammengesetzte Quadraturformeln auf Basis von stückweise Lagrange-Interpolationen und
3. Gauß-Quadraturen, bei denen die Quadraturpunkte ähnlich wie bei den Chebyshev-Knoten in Abschnitt 3.1.5 optimal gewählt werden.

4.2 Newton-Cotes Formeln

Die Polynom-Interpolationsquadratur-Formel zu $n+1$ äquidistanten Quadraturpunkte $x_j := a + jh$, $j = 0, \dots, n$, $h := \frac{b-a}{n}$ heißt *(abgeschlossene) Newton-Cotes-Quadraturformel*. Im Gegensatz dazu verwenden *(offene) Newton-Cotes-Quadraturformeln* die Quadraturpunkte $x_j := a + (j+1)h$, $j = 0, \dots, n$, $h := \frac{b-a}{n+2}$.

Satz 4.5. *Für interpolatorische Quadraturformeln Q_n mit den paarweise verschiedenen Quadraturpunkten $x_0, \dots, x_n$ gilt*

$$Q(f) - Q_n(f) = \int_a^b f[x_0, \dots, x_n, x] \prod_{j=0}^{n} (x - x_j) dx. \tag{4.6}$$

Beweis. Folgt direkt aus Satz 3.18. □

Quadraturformeln werden häufig auf Standardintervallen wie $[0,1]$ oder $[-1,1]$ angegeben. Wie bereits (3.20) bei den Chebyshev-Knoten gesehen, können diese leicht mit Hilfe der Substitutionen

$$\Psi_1 : [0,1] \ni \xi \mapsto (a + \xi(b-a)) \in [a,b],$$
$$\Psi_2 : [-1,1] \ni \xi \mapsto \frac{1}{2}(a + b + \xi(b-a)) \in [a,b]$$

auf ein allgemeines Intervall $[a,b]$ umgerechnet werden.

Beispiel 4.6 (Newton-Cotes Formeln)**.** *Mit Hilfe von (4.4) errechnet man für das Intervall $[0,1]$ leicht die*

- *abgeschlossenen Newton-Cotes Formeln für $n = 0, 1, 2, 3, 4$*

  1. $Q_0(f) := f(0)$ *(*Rechtecksregel*),*

2. $Q_1(f) := \frac{1}{2}(f(0) + f(1))$ *(*Trapezregel*),*
3. $Q_2(f) := \frac{1}{6}(f(0) + 4f(1/2) + f(1))$ *(*Simpson-regel*),*
4. $Q_3(f) := \frac{1}{8}(f(0) + 3f(1/3) + 3f(2/3) + f(1))$ *(*3/8-regel*),*
5. $Q_4(f) := \frac{1}{90}(7f(0) + 32f(1/4) + 12f(1/2) + 32f(3/4) + 7f(1))$ *(*Mile-Regel*),*

- *und die offenen Newton-Cotes Formeln für* $n = 0, 1, 2$

1. $Q_0(f) := f(1/2)$ *(*Mittelpunktsregel*),*
2. $Q_1(f) := \frac{1}{2}(f(1/3) + f(2/3))$,
3. $Q_2(f) := \frac{1}{3}(2f(1/4) - f(1/2) + 2f(3/4))$.

Negative Gewichte wie bei der offenen Newton-Cotes Formel für $n = 2$ treten bei $n \geq 8$ regelmäßig auf und können zu Auslöschung führen. Von daher sind meist nur Newton-Cotes Formeln mit kleinem n gebräuchlich.

Satz 4.7. *Es gelten für hinreichend glatte Funktionen* $f : [a, b] \to \mathbb{R}$ *folgende Restglieddarstellungen mit Zwischenstellen* $\xi \in [a, b]$*:*

1. *Trapezregel:*

$$Q(f) - \frac{b-a}{2}(f(a) + f(b)) = -\frac{(b-a)^3}{12} f''(\xi), \qquad f \in C^2([a, b]),$$

2. *Simpson-Regel:*

$$Q(f) - \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) = -\frac{(b-a)^5}{2880} f^{(4)}(\xi), \qquad f \in C^4([a, b]),$$

3. *Mittelpunktsregel:*

$$Q(f) - (b-a)f\left(\frac{a+b}{2}\right) = \frac{(b-a)^3}{24} f''(\xi), \qquad f \in C^2([a, b]).$$

Beweis. Die Aussagen folgen mit Satz 4.5 und wegen der Stetigkeit der Dividierten Differenzen (siehe Kor. 3.12) aus dem Mittelwertsatz der Integralrechnung: Wenn $f \in C([a, b])$ und $g : [a, b] \to \mathbb{R}$ integrierbar ohne Vorzeichenwechsel in $[a, b]$, dann existiert ein $\xi \in [a, b]$ mit $\int_a^b f(x)g(x)dx = f(\xi)\int_a^b g(x)dx$.

Wir zeigen die Aussage des Satzes am Beispiel der Mittelpunktsregel: Es gilt nach Satz 4.5

$$Q(f) - (b-a)f\left(\frac{a+b}{2}\right) = \int_a^b f\left[\frac{a+b}{2}, x\right]\left(x - \frac{a+b}{2}\right) dx.$$

$x \mapsto x - \frac{a+b}{2}$ hat einen Vorzeichenwechsel auf $[a,b]$, sodass der Mittelwertsatz der Integralrechnung nicht direkt anwendbar ist. Mit Hilfe der verallgemeinerten Dividierten Differenzen (3.13) können wir jedoch umformulieren

$$\begin{aligned} Q(f) - (b-a) f\left(\frac{a+b}{2}\right) &= \int_a^b \frac{f\left[\frac{a+b}{2}, x\right] - f\left[\frac{a+b}{2}, \frac{a+b}{2}\right]}{x - \frac{a+b}{2}} \left(x - \frac{a+b}{2}\right)^2 dx \\ &\quad + f'\left(\frac{a+b}{2}\right) \underbrace{\int_a^b \left(x - \frac{a+b}{2}\right) dx}_{=0} \\ &= \int_a^b f\left[\frac{a+b}{2}, \frac{a+b}{2}, x\right] \left(x - \frac{a+b}{2}\right)^2 dx \\ &= f\left[\frac{a+b}{2}, \frac{a+b}{2}, \xi\right] \int_a^b \left(x - \frac{a+b}{2}\right)^2 dx. \end{aligned}$$

Mit Hilfe des 1.Teils von Satz 3.18 angewendet auf die Hermite-Interpolation 3.13 mit Stützstellen $\frac{a+b}{2}$ und $\frac{a+b}{2}$ erhalten wir die Behauptung. □

4.3 Summierte Quadraturformeln

Newton-Cotes Formeln mit vielen Quadraturpunkten können wegen den auftretenden negativen Gewichten instabil sein. Zudem haben wir bereits bei der Polynominterpolation gesehen, dass das Interpolierende Polynom $p_f^{(n)} \in \Pi_n$ nicht notwendig für $n \to \infty$ gegen f konvergieren muss. Somit wird auch $Q_n(f)$ nicht notwendig gegen $Q(f)$ konvergieren. Ein Ausweg bietet, wie bei der Polynominterpolation, die Benutzung von Quadratformeln niedrigerer Ordnung auf Teilintervallen von $[a,b]$.

Satz 4.8. *Sei $f : [a,b] \to \mathbb{R}$ hinreichend glatt. Weiter sei $h := \frac{b-a}{N}$, $x_j := a + jh$ für $j = 0, \dots, n$ und gelte für die Quadraturformeln $Q_n^{[x_{j-1}, x_j]}$ auf den Intervallen $[x_{j-1}, x_j]$ eine Fehlerdarstellung der Form*

$$\int_{x_{j-1}}^{x_j} f(x)dx - Q_n^{[x_{j-1},x_j]}(f) = w_n h^{m+2} f^{(m+1)}(\xi_j), \qquad \xi_j \in [x_{j-1}, x_j], \quad j = 1, \dots, n$$

mit $w_n \in \mathbb{R}$ und $m \geq n$. Dann gilt für die zusammengesetzte Quadraturformel

$$Q_h^{(n)}(f) := \sum_{j=1}^{N} Q_n^{[x_{j-1},x_j]}(f) \tag{4.7}$$

die Fehlerdarstellung

$$Q(f) - Q_h^{(n)}(f) = w_n (b-a) h^{m+1} f^{(m+1)}(\xi), \qquad \xi \in [a,b]. \tag{4.8}$$

Beweis. Zwischenwertsatz für $f^{(m+1)}$. □

Beispiel 4.9. *Wir erhalten mit Hilfe des vorigen Satzes aus Satz 4.7 die*

1. *summierte Trapezregel ($m = 1$):*

$$Q_h^{(1)}(f) := \frac{h}{2}\left(f(a) + 2\sum_{j=1}^{N-1} f(x_j) + f(b)\right) \text{ mit}$$
$$Q(f) - Q_h^{(1)}(f) = -\frac{b-a}{12}h^2 f''(\xi),$$

2. *summierte Simpson-regel ($m = 3$):*

$$Q_h^{(2)}(f) := \frac{h}{6}\left(f(a) + 2\sum_{j=1}^{N-1} f(x_j) + 4\sum_{j=1}^{N} f\left(\frac{x_{j-1}+x_j}{2}\right) + f(b)\right) \text{ mit}$$
$$Q(f) - Q_h^{(2)}(f) = -\frac{b-a}{2880}h^4 f^{(4)}(\xi),$$

3. *und die summierte Mittelpunktsregel ($m = 1$):*

$$Q_h^{(0)}(f) := h\sum_{j=1}^{N} f\left(\frac{x_{j-1}+x_j}{2}\right) \text{ mit}$$
$$Q(f) - Q_h^{(0)}(f) = \frac{b-a}{24}h^2 f''(\xi).$$

Summierte Quadraturformeln können teilweise mit der Richardson-Extrapolation aus Abschnitt 3.4 verknüpft werden. Für die summierte Trapezregel wird z.B. in Plato, *Numerische Mathematik kompakt*, Vieweg 2000, eine asymptotische Darstellung folgender Form bewiesen:

$$Q_h(f) = Q(f) + \sum_{\ell=1}^{m} a_\ell(f)h^{2\ell} + a_{m+1}(f,h)h^{2\ell+2} \tag{4.9}$$

mit $a_{m+1}(f,h) = a_{m+1}(f) + \mathcal{O}_f(1)$ für $h \to 0$. Dies ist genau die Voraussetzung aus Satz 3.44. Richardson Extrapolation angewandt auf die summierte Trapezregel wird *Romberg-Verfahren* genannt.

Unabhängig von der Extrapolation helfen solche asymptotische Formeln sogenannte *a-posteriori Fehlerschätzer* zu konstruieren. Diese sollen nach erfolgter Rechnung Aufschluss darüber geben, wie groß der gemachte Fehler in etwa ist. Für die zusammengesetzte Trapezregel erhalten wir aus (4.9)

$$a_1(f)h^2 = \frac{Q_h(f) - Q_{h/2}(f)}{1 - 2^{-2}} + \mathcal{O}(h^4)$$

und somit in 1.Näherung

$$a_1(f) \doteq \frac{Q_h(f) - Q_{h/2}(f)}{1 - 2^{-2}}h^{-2}$$

und daraus

$$|Q(f) - Q_h(f)| \dot{\leq} \left| \frac{Q_{h/2}(f) - Q_h(f)}{1 - 2^{-2}} \right|. \tag{4.10}$$

Der Fehler der Quadraturformel mit der Schrittweite h kann also in 1.Näherung abgeschätzt werden, wenn das Ergebnis der Quadratur zur Schrittweite $h/2$ bekannt ist. Auf diese Weise könnte man ausgehend von einer großen Schrittweite h die Schrittweiten sukzessiv halbieren bis eine gewünschte Fehlertoleranz erreicht ist. Dies ist sogar lokal möglich, d.h. nur die Teilintervalle $[x_{j-1}, x_j]$ werden halbiert, auf denen der Fehlerschätzer groß ist. Die ganze Prozedur benötigt außer der Forderung $f \in C^4([a, b])$ keine Vorkenntnisse an f.

4.4 Gauß-Quadratur

Die im letzten Abschnitt vorgestellte Simpson-Regel ist eine interpolatorische Quadraturformel mit 3 Quadraturpunkten und hat somit mindestens die Ordnung 3 nach Definition 4.4. Da sie jedoch auch Polynome vom Grad 3 exakt integriert (Nachrechnen!), ist sie sogar von der Ordnung 4. Generell stellt sich somit die Frage, welche Ordnung eine Quadraturformel maximal haben kann.

4.4.1 Definition und Eigenschaften

Lemma 4.10. *Die Ordnung einer Quadraturformel auf dem Intervall $[a, b]$ der Form (4.3) mit $n + 1$ Quadraturknoten ist maximal $2n + 2$.*

Beweis. Seien $x_0, \ldots, x_n$ beliebige Quadraturknoten und

$$q_{n+1}(x) := \prod_{j=0}^{n} (x - x_j). \tag{4.11}$$

Dann ist $q_{n+1}^2 : x \mapsto (q_{n+1}(x))^2 \in \Pi_{2n+2}$ und es gilt

$$0 = Q_n(q_{n+1}^2) < \int_a^b q_{n+1}^2(x) dx = Q(q_{n+1}^2),$$

d.h. Q_n ist nicht exakt für q_{n+1}^2. □

Es stellt sich die Frage, ob diese obere Schranke scharf ist. Offenbar ist eine Quadraturformel (4.3) genau dann exakt auf Π_{2n+1}, wenn

$$\sum_{j=0}^{n} \alpha_j x_j^k = \int_a^b x^k \, dx, \qquad k = 0, \ldots, 2n + 1.$$

Dies ist ein nichtlineares Gleichungssystem von $2n + 2$ Gleichungen für die $2n + 2$ Unbekannten $x_0, \dots, x_n$ und $\alpha_0, \dots, \alpha_n$. Da die Anzahl der Gleichungen mit der Anzahl der Unbekannten übereinstimmt, können wir hoffen, dass es eine eindeutige Lösung besitzt. Dies werden wir im folgenden tatsächlich beweisen können.

Dabei werden wir das allgemeinere Problem betrachten, ein Integral

$$Q(f) := \int_a^b w(x) f(x)\, dx$$

mit einer möglicherweise singulären Gewichtsfunktion w auszuwerten. Unter einer *zulässigen Gewichtsfunktion* verstehen wir eine stetige Funktion $w : (a, b) \to (0, \infty)$, für die das Integral $\int_a^b w(x)\, dx$ existiert. Typische Beispiele sind

$$w(x) = 1, \qquad w(x) = \sqrt{1 - x^2}, \qquad w(x) = \frac{1}{\sqrt{1 - x^2}},$$

wobei wir in den letzten beiden Fällen annehmen, dass $[a, b] = [-1, 1]$. Für eine allgemeine Gewichtsfunktion w können wir, wie bereits für $w = 1$ besprochen, Interpolationsquadratur-Formeln konstruieren, indem wir f durch das Interpolationspolynom p_f zu den Stützstellen $x_0, \dots, x_n$ ersetzen und dann exakt integrieren:

$$Q_n(f) := \int_a^b w(x) p_f(x)\, dx \tag{4.12}$$

Bemerkung 4.11. *Analog zum Spezialfall $w = 1$ kann man zeigen, dass die Gewichte der Quadraturformel (4.12) durch*

$$\alpha_j = \int_a^b w(x) L_j(x)\, dx, \qquad j = 0, \dots, n \tag{4.13}$$

mit den Lagrange-Basisfunktionen L_j gegeben sind und dass für gegebene Quadraturpunkte $x_0, \dots, x_n$ die Quadraturformel (4.12) eindeutig durch die Eigenschaft charakterisiert ist, dass Polynome vom Grad $\leq n$ exakt integriert werden.

Definition 4.12 (Gauß-Quadratur)**.** *Ein Quadraturformel*

$$Q_n(f) := \sum_{j=0}^{n} \alpha_j f(x_j) \approx \int_a^b w(x) f(x) dx = Q(f)$$

mit Gewichten $\alpha_0, \dots, \alpha_n$ und paarweise verschiedenen Quadraturknoten $x_0, \dots, x_n$ heißt Gauß-Quadratur zu der zulässigen Gewichtsfunktion w*, falls gilt*

$$\sum_{j=0}^{n} \alpha_j x_j^k = \int_a^b w(x) x^k\, dx, \qquad k = 0, \dots, 2n + 1.$$

Satz 4.13. *Sei $w : (a, b) \to (0, \infty)$ eine zulässige Gewichtsfunktion, $x_0, \dots, x_n \in [a, b]$ paarweise verschiedene Quadraturpunkte, $p_f \in \Pi_n$ das interpolierende Polynom zu einer Funktion $f \in C([a, b])$ und q_{n+1} sei durch (4.11) definiert. Dann sind folgende Aussagen äquivalent:*

1. *$Q_n(f) := \int_a^b w(x)p_f(x)\,dx$ ist eine Gaußsche Quadraturformel, d.h.*

$$Q_n(p) = Q(p) \qquad \text{für alle } p \in \Pi_{2n+1}.$$

2. *Es gilt*

$$\int_a^b w(x)q_{n+1}(x)q(x)\,dx = 0 \qquad \text{für alle } q \in \Pi_n. \tag{4.14}$$

Beweis. 1. $1 \Rightarrow 2$: Es gilt $Q_n(f) = \sum_{j=0}^n \alpha_j f(x_j)$ mit den durch (4.13) gegebenen Quadraturgewichten α_j. Da $q_{n+1}q \in \Pi_{2n+1}$ und da $q_{n+1}(x_j) = 0$, erhalten wir für alle $q \in \Pi_n$

$$\int_a^b w(x)q_{n+1}(x)q(x)\,dx = \sum_{j=0}^n \alpha_j q_{n+1}(x_j)q(x_j) = 0.$$

2. $2 \Rightarrow 1$: Nach dem Hauptsatz der Algebra lässt sich jedes Polynom $p \in \Pi_{2n+1}$ darstellen als

$$p = p_p + q_{n+1}q$$

für ein $q \in \Pi_n$, da $p - p_p \in \Pi_{2n+1}$ an den Punkten $x_0, \ldots, x_n$ verschwindet. Daher gilt nach (4.14)

$$\int_a^b w(x)p(x)\,dx = \int_a^b w(x)p_p(x)\,dx.$$

□

4.4.2 Orthogonale Polynome und Existenzsatz

Lemma 4.14. *Sei $w : (a,b) \to (0,\infty)$ eine zulässige Gewichtsfunktion. Dann ist durch*

$$(f,g)_w := \int_a^b w(x)f(x)g(x)\,dx$$

ein Skalarprodukt auf $C([a,b])$ definiert.

Beweis. Bilinearität und Symmetrie von $(\cdot,\cdot)_w$ sind offensichtlich. Weiterhin folgt wegen $w(x) > 0$ für $x \in (a,b)$ sofort $(f,f)_w \geq 0$ für $f \in C([a,b])$. Zu zeigen bleibt, dass aus $(f,f)_w = 0$ auch $f \equiv 0$ folgt (Übungsaufgabe). □

Mit dieser Notation lautet die Charakterisierung der Gaußschen Quadraturformeln aus Satz 4.13 einfach

$$(q_{n+1}, q)_w = 0 \qquad \text{für alle } q \in \Pi_n.$$

Mit anderen Worten, q_{n+1} muss orthogonal sein zu allen Polynomen, deren Grad echt kleiner als der von q_{n+1} ist. Eine solche Folge von Polynomen erhält man durch Anwendung des Gram-Schmidtschen Orthogonalisierungsverfahrens auf die Monome $M_n(x) := x^n$, $n = 0, 1, \ldots$:

Korollar 4.15. *Es existiert eine eindeutig bestimmte Folge (q_n) von Polynomen der Form $q_0(x) := 1$ und*

$$q_n(x) := x^n + r_{n-1}(x), \qquad x \in [a, b],$$

mit $r_{n-1} \in \Pi_{n-1}$, die die Orthogonalitätsbeziehungen

$$(q_n, q_m)_w = 0, \qquad n \neq m \tag{4.15}$$

erfüllen.

Da wir die Nullstellen von q_{n+1} als Quadraturpunkte verwenden möchten (siehe (4.11)), müssen diese im Intervall $[a, b]$ liegen. Dass dies tatsächlich der Fall ist, sichert das folgende Lemma.

Lemma 4.16. *Jedes der orthogonalen Polynome $q_n \in \Pi_n$ aus dem vorangehenden Korollar besitzt n einfache Nullstellen in dem offenen Intervall (a, b).*

Beweis. Seien $x_1, \ldots, x_m$ die Nullstellen von q_n im Intervall (a, b), bei denen q_n das Vorzeichen ändert. Wir nehmen an, es sei $m < n$ und definierten $r_m(x) := \prod_{j=1}^{m}(x - x_j)$, bzw. $r_m(x) := 1$ für $m = 0$. Da $r_m \in \Pi_{n-1}$, gilt $(r_m, q_n)_w = 0$. Dies ist ein Widerspruch, da der Integrand auf dem Intervall (a, b) das Vorzeichen nicht ändert und nicht identisch verschwindet. Deshalb war die Annahme $m < n$ falsch, und q_n besitzt n verschiedene Nullstellen in (a, b). □

Aus Satz 4.13, Korollar 4.15 und Lemma 4.16 folgt nun das Hauptresultat dieses Abschnitts:

Satz 4.17. *Zu jeder zulässigen Gewichtsfunktion w existieren eindeutig bestimmte Gaußsche Quadraturformeln Q_n, $n \in \mathbb{N}_0$, mit $n + 1$ Quadraturknoten und Ordnung $2n + 2$. Die Quadraturpunkte sind gegeben durch die Nullstellen des entsprechenden orthogonalen Polynoms q_{n+1} vom Grad $n + 1$ aus Korollar 4.15. Die Gewichte der Gauß-Quadratur sind alle positiv.*

Beweis. Die Existenz einer Gauß-Quadratur folgt aus der Existenz der Orthogonalpolynome und Lemma 4.16. Seien $L_0, \ldots, L_n \in \Pi_n$ die Lagrange-Basispolynome zu den Nullstellen $x_0, \ldots, x_n$ des orthogonalen Polynoms q_{n+1}. Dann ist $L_j^2 \in \Pi_{2n} \setminus \{0\}$ und es gilt

$$0 < \int_a^b w(x)\,(L_k(x))^2\,dx = \sum_{j=0}^{n} \alpha_j \underbrace{(L_k(x_j))^2}_{\delta_{jk}} = \alpha_k, \qquad k = 0, \ldots, n.$$

Damit sind die Gewichte positiv. Zur Eindeutigkeit der Gauß-Quadratur sei $\tilde{Q}^{(n)}(f) = \sum_{j=0}^{n} \tilde{\alpha}_j f(\tilde{x}_j)$ eine weitere Gauß-Quadratur. Auch deren Gewichte sind positiv und mit den zugehörigen Lagrange-Basispolynomen $\tilde{L}_0, \ldots, \tilde{L}_n$ folgt

$$0 = \int_a^b w(x)\tilde{L}_k(x)q_{n+1}(x)dx = \tilde{Q}^{(n)}(\tilde{L}_k q_{n+1}) = \sum_{j=0}^{n} \tilde{\alpha}_j \underbrace{\tilde{L}_k(\tilde{x}_j)}_{=\delta_{jk}} q_{n+1}(\tilde{x}_j) = \tilde{\alpha}_k q_{n+1}(\tilde{x}_k),$$

d.h. $\tilde{x}_0, \ldots, \tilde{x}_n$ sind Nullstellen von q_{n+1}. Damit sind die Quadraturknoten und folglich auch die Gewichte der beiden Quadraturformeln identisch. □

4.4.3 Fehleranalysis

Satz 4.18. *Sei $f \in C^{2n+2}([a,b])$. Dann erfüllt der Fehler der Gauß-Quadratur mit $n+1$ Quadraturknoten die Gleichung*

$$Q(f) - Q_n(f) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_a^b w(x) \prod_{j=0}^{n} (x - x_j)^2 dx \tag{4.16}$$

für ein $\xi \in (a,b)$.

Beweis. Sei $p_f \in \Pi_{2n+1}$ das durch

$$p_f(x_j) = f(x_j) \quad \text{und} \quad p_f'(x_j) = f'(x_j), \qquad j = 0, \ldots, n$$

eindeutig bestimmte Hermite-Interpolationspolynom. Dann gilt

$$Q_n(f) = \sum_{j=0}^{n} \alpha_j f(x_j) = \sum_{j=0}^{n} \alpha_j p_f(x_j) = Q_n(p_f) = Q(p_f),$$

und für den Fehler erhalten wir

$$\begin{aligned} Q(f) - Q_n(f) &= \int_a^b w(x)\,(f(x) - p_f(x))\,dx \\ &= \int_a^b w(x) q_{n+1}(x)^2 \frac{f(x) - p_f(x)}{q_{n+1}(x)^2}\,dx. \end{aligned}$$

Nach dem Mittelwertsatz der Integralrechnung gilt deshalb

$$Q(f) - Q_n(f) = \frac{f(\tilde{\xi}) - p_f(\tilde{\xi})}{q_{n+1}(\tilde{\xi})^2} \int_a^b w(x) q_{n+1}(x)^2\,dx$$

für eine Zwischenstelle $\tilde{\xi} \in (a,b)$. Nun folgt die Behauptung aus der Darstellung (3.14) des Hermiteschen Interpolationsfehlers. □

Aus dem vorigen Satz erhalten wir, wie bei der Polynominterpolation, die Konvergenz $Q_n(f) \to Q(f)$ für $n \to \infty$ für $f \in C^\infty([a,b])$. Die Voraussetzungen an f sind jedoch nicht notwendig, es reicht $f \in C([a,b])$. Dazu benötigen wir folgenden Satz.

Satz 4.19 (Szegö)**.** *Sei Q_n eine Folge von Quadraturformeln auf $[a,b]$ mit $Q_n(f) = \sum_{j=0}^n \alpha_j^{(n)} f(x_j^{(n)})$. Dann sind folgende Aussagen äquivalent.*

1. *Es gilt $Q(f) = \lim_{n\to\infty} Q_n(f)$ für alle $f \in C([a,b])$.*
2. *Es gilt $Q(p) = \lim_{n\to\infty} Q_n(p)$ für alle Polynome p auf $[a,b]$ sowie*

$$\sup_{n\in\mathbb{N}} \sum_{j=0}^n \left|\alpha_j^{(n)}\right| < \infty. \tag{4.17}$$

Beweis. Der Satz benötigt Kenntnisse aus der Funktionalanalysis und wird deshalb hier nicht vollständig bewiesen. Wir beweisen nur die Richtung $2 \to 1$, welche wir im Folgenden benötigen:

Wie bei der Polynominterpolation der Satz 3.5 kann für $f \in C([a,b])$ nachgewiesen werden, dass

$$|Q_n(f)| \le \left(\sum_{j=0}^n \left|\alpha_j^{(n)}\right|\right) \|f\|_\infty$$

und dass diese Ungleichung scharf ist. Sei $M := \sup_{n\in\mathbb{N}} \sum_{j=0}^n \left|\alpha_j^{(n)}\right| > 0$, $C := \int_a^b w(x)dx > 0$ und $\epsilon > 0$ beliebig. Dann existiert zu jedem $f \in C([a,b])$ mit Hilfe des Approximationssatzes von Weierstrass ein Polynom p mit $\|f-p\|_\infty < \frac{\epsilon}{3\max(M,C)}$ und für dieses p wegen der Bedingung (2) ein $n_0 \in \mathbb{N}$ mit $|Q_n(p) - Q(p)| < \frac{\epsilon}{3}$ für alle $n \ge n_0$. Insgesamt folgt

$$\begin{aligned} |Q_n(f) - Q(f)| \le& |Q_n(f) - Q_n(p)| + |Q_n(p) - Q(p)| + |Q(p) - Q(f)| \\ \le& M\|f-p\|_\infty + \frac{\epsilon}{3} + C\|f-p\|_\infty < \epsilon, \qquad n \ge n_0. \end{aligned}$$

□

Satz 4.20 (Konvergenz Gauß-Quadratur)**.** *Die Gauß-Quadraturen $Q_n(f)$ konvergieren für $n \to \infty$ für jedes $f \in C([a,b])$ gegen $Q(f)$.*

Beweis. Folgt direkt aus dem vorigen Satz mit

$$\sum_{j=0}^n \left|\alpha_j^{(n)}\right| = \sum_{j=0}^n \alpha_j^{(n)} = \int_a^b w(x)dx.$$

□

4.4.4 Bestimmung von Gauß-Quadraturen

Beispiel 4.21 (Gauß-Chebyshev-Quadratur). *Die Orthogonalpolynome nach Korollar 4.15 für $w(x) = \frac{1}{\sqrt{1-x^2}}$, $x \in (-1,1)$ sind gegeben durch die Chebyshev-Polynome aus Abschnitt 3.1.5. Man rechnet leicht nach, dass die Quadraturformel gegeben ist durch*

$$Q_n(f) = \frac{\pi}{n+1} \sum_{j=0}^{n} f\left(\cos\left(\frac{(2j+1)\pi}{2(n+1)}\right)\right) \approx \int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}} dx.$$

Beispiel 4.22 (Gauß-Legendre-Quadratur). *Die Orthogonalpolynome nach Korollar 4.15 für $w \equiv 1$ auf $[-1,1]$ sind gegeben durch die Legendre-Polynome*

$$L_n(x) := \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2-1)^n, \qquad x \in [-1,1].$$

Die Bestimmung der Nullstellen und damit der Quadraturgewichte ist etwas aufwendiger als bei der Gauß-Chebyshev-Quadratur. Die ersten $n = 0, 1, 2$ Quadraturformeln lauten

$$\begin{aligned} Q_0(f) &= 2f(0), \\ Q_1(f) &= f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right), \\ Q_2(f) &= \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right). \end{aligned}$$

Numerisch können die Quadraturknoten und Gewichte mit Hilfe des folgenden Satzes berechnet werden, welchen wir hier ohne Beweis wiedergeben.

Satz 4.23. *Die Orthogonalpolynome $q_n \in \Pi_n$ aus Korollar 4.15 sind eindeutig durch die folgende 3-Term-Rekursion gegeben:*

$$\begin{aligned} q_0(x) := 1, \quad q_1(x) := (x-\beta_0)q_0(x) = x - \beta_0, \quad \textit{und} \\ q_{n+1}(x) := (x-\beta_n)q_n(x) - \gamma_n^2 q_{n-1}(x) \quad \textit{für} \quad n \geq 1, \end{aligned}$$

mit den reellen Koeffizienten

$$\beta_n := \frac{(xq_n, q_n)_w}{\|q_n\|_w^2} \quad \textit{und} \quad \gamma_n := \frac{\|q_n\|_w}{\|q_{n-1}\|_w}.$$

Weiter sind die Eigenwerte der Matrix

$$T := \begin{pmatrix} \beta_0 & \gamma_1 & & & \\ \gamma_1 & \beta_1 & \gamma_2 & & \\ & \gamma_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \gamma_n \\ & & & \gamma_n & \beta_n \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

die Nullstellen $x_0, \ldots, x_n$ *des* $(n+1)$*-ten Orthogonalpolynoms* q_{n+1} *und die zugehörigen Gewichte der Gauß-Quadratur sind gegeben durch*

$$\alpha_j := \left(\frac{(v_j)_1}{\|v_j\|_2} \right)^2 \int_a^b w(x)dx, \qquad j = 0, \ldots, n,$$

wobei $(v_j)_1$ *die 1.Komponente eines Eigenvektors* $v_j \in \mathbb{R}^{n+1} \setminus \{0\}$ *zum Eigenwert* x_j *ist.*

5 Matrixnormen und Kondition

Im Folgenden benötigen wir einige Grundlagen zu Matrizen und insbesondere zu Matrixnormen.

5.1 Matrixnormen

Eine Matrix $A \in \mathbb{K}^{m\times n}$ kann als lineare Abbildung von $\mathbb{K}^n$ nach $\mathbb{K}^m$ aufgefasst werden. Normen von linearen Abbildungen werden in der linearen Funktionalanalysis eingeführt und beruhen auf den Vektornormen im Bild- und Urbildraum.

5.1.1 Vektornormen

Definition 5.1. *Sei V ein Vektorraum über $\mathbb{K}$. Unter einer Norm auf V versteht man eine Abbildung $\|\cdot\| : V \to [0,\infty)$, die folgenden Axiomen genügt:*

1. $\|x\| > 0$ *für alle* $x \in V \setminus \{0\}$ *(Definitheit)*
2. $\|\alpha x\| = |\alpha|\|x\|$ *für alle* $\alpha \in \mathbb{K}$, $x \in V$ *(Homogenität)*
3. $\|x + y\| \leq \|x\| + \|y\|$ *für alle* $x, y \in V$ *(Dreiecksungleichung)*

Ein Paar $(V, \|\cdot\|)$ bestehend aus einem Vektorraum V und einer Norm $\|\cdot\|$ auf V heißt normierter Raum. *Ein vollständiger normierter Raum heißt* Banachraum.

Einige Beispiele von Normen auf $V = \mathbb{K}^n$ sind:

- euklidische Norm: $\|x\|_2 := (\sum_{j=1}^n |x_j|^2)^{1/2}$
- Maximumnorm: $\|x\|_\infty := \max_{j=1,\dots,n} |x_j|$
- Betragssummennorm: $\|x\|_1 := \sum_{j=1}^n |x_j|$
- l^p-Norm ($p \in [1,\infty)$): $\|x\|_p := (\sum_{j=1}^n |x_j|^p)^{1/p}$

Wir wissen aus der Analysis, dass alle Normen auf $\mathbb{K}^n$ äquivalent zur Maximumnorm sind:

Satz 5.2. *Ist $\|\cdot\|$ eine Norm auf $\mathbb{K}^n$, so gibt es Konstanten $c, C > 0$, so dass für alle $x \in \mathbb{K}^n$ gilt:*

$$c\|x\|_\infty \leq \|x\| \leq C\|x\|_\infty.$$

Dieser Satz impliziert, dass alle Normen auf $\mathbb{K}^n$ die gleiche Topologie induzieren und somit topologische Begriffe wie Konvergenz, Abgeschlossenheit, Beschränktheit und Kompaktheit nicht von der Wahl der Norm abhängen.

5.1.2 Zugeordnete Matrixnormen

Definition und Satz 5.3 (Natürliche Matrixnorm)**.** *Sei $V = \mathbb{K}^n$ versehen mit der Norm $\|\cdot\|_V$ und $W = \mathbb{K}^m$ versehen mit den Norm $\|\cdot\|_W$. Dann ist durch*

$$\|A\|_{V\to W} := \sup_{\|x\|_V=1} \|Ax\|_W$$

eine Norm auf dem Vektorraum $\mathbb{K}^{m\times n}$ definiert. Sie heißt die den Normen $\|\cdot\|_V$ und $\|\cdot\|_W$ zugeordnete Matrixnorm*. Falls $V = W$, schreiben wir $\|A\|_V := \|A\|_{V\to V}$. Für alle $x \in V$ und $A \in \mathbb{K}^{m\times n}$ gilt*

$$\|Ax\|_W \leq \|A\|_{V\to W}\|x\|_V. \tag{5.1}$$

Ist $U = \mathbb{K}^l$ ein weiterer Vektorraum mit Norm, so gilt

$$\|AB\|_{U\to W} \leq \|A\|_{V\to W}\|B\|_{U\to V}$$

für alle $A \in \mathbb{K}^{m\times n}$ und $B \in \mathbb{K}^{n\times l}$. Im Falle $U = V = W$ nennt man eine Norm auf $\mathbb{K}^{n\times n}$ mit dieser Eigenschaft submultiplikativ*.*

Beweis. Da die Einheitssphähre $\{x \in \mathbb{K}^n : \|x\| = 1\}$ kompakt ist, nimmt für jede Matrix $A \in \mathbb{K}^{m\times n}$ die stetige Abbildung $\mathbb{K}^n \to \mathbb{K}^m$, $x \mapsto Ax$ ihr Supremum auf der Einheitssphähre an. Deshalb gilt $\|A\|_{V\to W} < \infty$. Die Definitheit und Homogenität der Norm $\|\cdot\|_{V\to W}$ sind leicht nachzurechnen. Wegen

$$\begin{aligned}\|A+B\|_{V\to W} &= \sup_{\|x\|_V=1} \|Ax+Bx\|_W \leq \sup_{\|x\|_V=1} (\|Ax\|_W + \|Bx\|_W)\\ &\leq \sup_{\|x\|_V=1} \|Ax\|_W + \sup_{\|x\|_V=1} \|Bx\|_W = \|A\|_{V\to W} + \|B\|_{V\to W}\end{aligned}$$

gilt auch die Dreiecksungleichung. Für $x \in V$ und $A \in \mathbb{K}^{m\times n}$ gilt aufgrund der Definition von $\|A\|_{V\to W}$ und der Homogenität der Vektornorm $\|\cdot\|_W$ die Ungleichung

$$\|Ax\|_W = \|x\|_V \left\| A\left(\frac{x}{\|x\|_V}\right)\right\|_W \leq \|A\|_{V\to W}\|x\|_V.$$

Für $B \in \mathbb{K}^{n\times l}$ und $y \in \mathbb{K}^l$ folgt somit

$$\|ABy\|_W \leq \|A\|_{V\to W}\|By\|_V \leq \|A\|_{V\to W}\|B\|_{U\to V}\|y\|_U$$

und daraus die letzte Ungleichung durch Bilden des Supremums über alle y mit $\|y\|_U = 1$. □

Satz 5.4 (Zeilensummennorm). *Sind $\|\cdot\|_V$ und $\|\cdot\|_W$ die Maximumnorm, so ist die zugeordnete Matrixnorm gegeben durch die Zeilensummennorm*

$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|.$$

Beweis. Wir bezeichnen die rechte Seite dieser Gleichung mit γ. Dann gilt

$$\|Ax\|_\infty = \max_{i=1,\dots,m} \left| \sum_{j=1}^{n} a_{ij} x_j \right| \leq \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}| \cdot |x_j| \leq \gamma \|x\|_\infty$$

für alle $x \in \mathbb{K}^n$. Deshalb ist $\|A\|_\infty \leq \gamma$. Zum Nachweis der umgekehrten Ungleichung wählen wir $k \in \{1, \dots, m\}$ so, dass $\gamma = \sum_{j=1}^{n} |a_{kj}|$. Falls $A = 0$, ist die Behauptung trivialerweise richtig. Anderenfalls definieren wir $y \in \mathbb{K}^n$ durch

$$y_j = \begin{cases} \frac{\overline{a_{kj}}}{|a_{kj}|} & \text{falls } a_{kj} \neq 0 \\ 0 & \text{sonst} \end{cases}.$$

Dann gilt $\|y\|_\infty = 1$ und wir erhalten

$$\gamma = \sum_{j=1}^{n} |a_{kj}| = \sum_{j=1}^{n} a_{kj} y_j \leq \|Ay\|_\infty \leq \|A\|_\infty \|y\|_\infty = \|A\|_\infty.$$

□

Bemerkung 5.5. *Der Beweis erfolgt analog zu dem von Satz 3.5 zur Kondition der Polynominterpolation bzw. zu dem entsprechenden Teil im Beweis des Satzes von Szegö 4.19. In der Tat werden in diesen Beweisen Darstellungen für die Normen des linearen Interpolationsoperators bzw. des linearen Quadraturfunktionals bewiesen. Näheres dazu in der Funktionalanalysis.*

Analog zeigt man

Satz 5.6 (Spaltensummennorm). *Sind $\|\cdot\|_V$ und $\|\cdot\|_W$ die Betragssummennorm $\|\cdot\|_1$, so ist die zugeordnete Matrixnorm gegeben durch die Spaltensummennorm*

$$\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{m} |a_{ij}|.$$

Im weiteren Verlauf der Vorlesung werden folgende, aus der linearen Algebra bekannten Definitionen benötigt.

Definition 5.7. *Sei $A \in \mathbb{K}^{n \times n}$. Wir definieren*

1. Eigenwert: *$\lambda \in \mathbb{C}$ heißt genau dann Eigenwert von A wenn ein Eigenvektor $x \in \mathbb{K}^n \setminus \{0\}$ existiert mit $Ax = \lambda x$.*

2. hermitesch: *A heißt hermitesch, wenn $A = A^*$, d.h. $a_{ij} = \overline{a_{ji}}$ für $i, j = 1, \ldots, n$.*

3. symmetrisch: *A heißt symmetrisch, wenn $A \in \mathbb{R}^{n\times n}$ und $A = A^*$.*

4. positiv definit: *A heißt positiv definit, wenn $(Ax, x)_2 > 0$ für $x \in \mathbb{K}^n \setminus \{0\}$.*

5. positiv semidefinit: *A heißt positiv semidefinit, wenn $(Ax, x)_2 \geq 0$ für $x \in \mathbb{K}^n$.*

6. unitär: *A heißt unitär, wenn $AA^* = I$.*

7. orthogonal: *A heißt orthogonal, wenn $A \in \mathbb{R}^{n\times n}$ und $AA^* = I$.*

8. Dreiecksmatrix: *A heißt untere (bzw. obere) Dreiecksmatrix, falls $a_{ij} = 0$ für $j > i$ (bzw. $j < i$).*

9. normiert: *Eine Dreiecksmatrix A heißt normiert, falls $a_{ii} = 1$ für $i = 1, \ldots, n$.*

Lemma 5.8. *Die folgenden Mengen von Matrizen bilden eine Gruppe bezüglich der Matrixmultiplikation als Verknüpfung.*

1. *Die Menge aller regulären oberen Dreiecksmatrizen.*

2. *Die Menge aller regulären unteren Dreiecksmatrizen.*

3. *Die Menge aller normierten oberen Dreiecksmatrizen.*

4. *Die Menge aller normierten unteren Dreiecksmatrizen.*

Beweis. Nachrechnen. □

Folgende Aussagen werden als bekannt vorausgesetzt.

Satz 5.9. *Sei $A \in \mathbb{K}^{n\times n}$. Dann gilt:*

1. *Es existiert eine eine unitäre Matrix $Q \in \mathbb{C}^{n\times n}$ und eine obere Dreiecksmatrix $U \in \mathbb{C}^{n\times n}$, so dass $Q^*AQ = U$ (*Schur-Zerlegung*).*

2. *Ist A hermitesch, so sind alle Eigenwerte von A reell und es existiert eine Orthonormalbasis von $\mathbb{K}^n$ aus Eigenvektoren $x_1, \ldots, x_n$ von A. Mit anderen Worten: A ist diagonalisierbar mit $X^*AX = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ und $X := (x_1, \ldots, x_n) \in \mathbb{K}^{n\times n}$.*

3. *Ist A unitär, so ist A diagonalisierbar und alle Eigenwerte von A haben den Betrag 1. Insbesondere ist A invertierbar.*

4. *Ist A unitär, so ist das euklidische Skalarprodukt invariant unter Transformationen $x \mapsto Ax$, d.h. $(Ax, Ay)_2 = (x, y)$ für alle $x, y \in \mathbb{K}^n$.*

5. *Die Zeilen bzw. Spalten einer unitären Matrix bilden ein Orthonormalsystem bezüglich des euklidischen Skalarproduktes auf $\mathbb{K}^n$.*

6. *Ist A hermitesch, so ist A genau dann positiv definit, wenn alle Eigenwerte positiv sind. Insbesondere ist dann A invertierbar.*

7. *Ist A hermitesch, so ist A genau dann positiv semidefinit, wenn alle Eigenwerte reell und nicht negativ sind.*

Definition 5.10 (Spektralradius)**.** *Sei $A \in \mathbb{K}^{n\times n}$. Dann heißt*

$$\rho(A) := \max\{|\lambda| : \lambda \text{ ist Eigenwert von } A\}$$

der Spektralradius *von A.*

Satz 5.11 (Spektralnorm)**.** *Sind $\|\cdot\|_V$ und $\|\cdot\|_W$ die euklidische Norm $\|\cdot\|_2$, so ist die zugeordnete Matrixnorm gegeben durch*

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

Ist $A \in \mathbb{K}^{n\times n}$ hermitesch, so gilt $\|A\|_2 = \rho(A)$.

Beweis. A^*A ist offensichtlich hermitesch und positiv semidefinit. Sei nun $X^*\,(A^*A)\,X = D$ mit X wie in Satz 5.9 und $D := \operatorname{diag}(d_1, \ldots, d_n)$ und $d_j \geq 0$ für $j = 1, \ldots, n$, so folgt für $x \in \mathbb{K}^n \setminus \{0\}$ und $y := X^*x$

$$\begin{aligned}
\|Ax\|_2^2 &= x^*A^*Ax = y^*X^*A^*AXy = y^*Dy \\
&= \sum_{j=1}^{n} d_j|y_j|^2 \leq \left(\max_{j=1,\ldots,n} d_j\right) \sum_{j=1}^{n} |y_j|^2 \\
&= \rho(A^*A)\|y\|_2^2 = \rho(A^*A)\|x\|_2^2.
\end{aligned}$$

Daraus folgt $\|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} \leq \sqrt{\rho(A^*A)}$. Zum Beweis der umgekehrten Ungleichung sei $\lambda = \rho(A^*A)$ der größte Eigenwert von A^*A und $x \in \mathbb{K}^n \setminus \{0\}$ ein zugehöriger Eigenvektor. Dann gilt $\|Ax\|_2^2 = x^*A^*Ax = x^*\lambda x = \lambda\|x\|_2^2$. Daraus folgt $\|A\|_2 \geq \frac{\|Ax\|_2}{\|x\|_2} = \sqrt{\lambda} = \sqrt{\rho(A^*A)}$.

Falls $A \in \mathbb{K}^{n\times n}$ hermitesch, so sind die Eigenwerte von A^*A genau die Quadrate der Eigenwerte von A. □

5.2 Störungstheorie

Wir möchten zur Vorbereitung des nächsten Kapitels die Kondition der Operation $(A, b) \mapsto A^{-1}b$, d.h. "Lösung des linearen Gleichungssystems $Ax = b$", bestimmen.

5.2.1 Reguläre Matrizen

Definition 5.12. *Wir bezeichnen mit $GL(n, \mathbb{K}) \subset \mathbb{K}^{n\times n}$ die Menge aller invertierbaren $n \times n$-Matrizen (engl.* general linear group*).*

Lemma 5.13. *Sei $B \in \mathbb{K}^{n\times n}$ mit $\|B\| < 1$. Dann ist die Matrix $I + B$ regulär, und es gilt*

$$\|(I+B)^{-1}\| \leq \frac{1}{1-\|B\|}. \tag{5.2}$$

Beweis. $I + B$ ist regulär, da für alle $x \in \mathbb{K}^n$ gilt

$$\|(I+B)x\| \geq \|x\| - \|Bx\| \geq \underbrace{(1-\|B\|)}_{>0}\|x\|.$$

Aus

$$\begin{aligned} 1 =& \|I\| = \left\|(I+B)\,(I+B)^{-1}\right\| = \left\|(I+B)^{-1} + B\,(I+B)^{-1}\right\| \\ \geq& \left\|(I+B)^{-1}\right\| - \|B\|\left\|(I+B)^{-1}\right\| = \left\|(I+B)^{-1}\right\|(1-\|B\|) \end{aligned}$$

folgt dann die Behauptung. □

Satz 5.14 (Störungssatz)**.** *Sei $A \in GL(n,\mathbb{K})$ und $\delta A \in \mathbb{K}^{n\times n}$ mit $\|\delta A\| < \|A^{-1}\|^{-1}$.*

Dann ist $A + \delta A \in GL(n,\mathbb{K})$.

Weiter seien $b, \delta b \in \mathbb{K}^n$ mit $b \neq 0$ und $x \in \mathbb{K}^n$ bzw. $x + \delta x \in \mathbb{K}^n$ die (eindeutigen) Lösungen von $Ax = b$ bzw. $(A+\delta A)(x+\delta x) = b + \delta b$. Dann gilt

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\operatorname{cond}(A)}{1 - \operatorname{cond}(A)\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right) \tag{5.3}$$

mit der relativen Konditionszahl $\operatorname{cond}(A) := \|A\|\|A^{-1}\|$ von A.

Beweis. $A + \delta A = A\,(I + A^{-1}\delta A)$ ist regulär, da A regulär ist und wegen des letzten Lemmas und der Voraussetzung $\|A^{-1}\delta A\| \leq \|A^{-1}\|\|\delta A\| < 1$ auch $I + A^{-1}\delta A$ regulär ist. Wegen

$$(A+\delta A)\,(x+\delta x) = b + \delta b \quad \Leftrightarrow \quad (A+\delta A)\,\delta x = \delta b - \delta A x$$

folgt die Abschätzung des relativen Fehlers aus folgender Ungleichungskette nach Division durch $\|x\| \neq 0$

$$\begin{aligned} \|\delta x\| \leq& \left\|(A+\delta A)^{-1}\right\|(\|\delta b\| + \|\delta A\|\|x\|) \\ =& \left\|\left(I + A^{-1}\delta A\right)^{-1}A^{-1}\right\|(\|\delta b\| + \|\delta A\|\|x\|) \\ \leq& \left\|\left(I + A^{-1}\delta A\right)^{-1}\right\|\left\|A^{-1}\right\|(\|\delta b\| + \|\delta A\|\|x\|) \\ \leq& \frac{\|A^{-1}\|}{1 - \|A^{-1}\delta A\|}(\|\delta b\| + \|\delta A\|\|x\|) \\ \leq& \frac{\|A^{-1}\|\,\|A\|\|x\|}{1 - \|A^{-1}\delta A\|}\left(\frac{\|\delta b\|}{\|b\|}\underbrace{\frac{\|b\|}{\|A\|\|x\|}}_{\leq 1} + \frac{\|\delta A\|}{\|A\|}\right). \end{aligned}$$

□

Die Konditionszahl $\text{cond}(A)$ hängt von der zugrunde liegenden Norm ab. Wegen Satz 5.11 folgt für reguläre, hermitesche Matrizen A

$$\text{cond}_2(A) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}, \tag{5.4}$$

d.h. die *spektrale Konditionszahl* ist der Quotient aus betragsgrößtem durch betragskleinstem Eigenwert. Für unitäre Matrizen A folgt aus Satz 5.9 $\text{cond}_2(A) = 1$. Für jede natürliche Matrixnorm und jede reguläre Matrix A gilt

$$\text{cond}(A) = \|A\| \, \|A^{-1}\| \geq \|AA^{-1}\| = \|I\| = 1. \tag{5.5}$$

Lemma 5.15. *Die Abschätzung (5.3) ist für die Spektralnorm scharf.*

Beweis. Sei A positiv definit mit minimalem Eigenwert λ_1 und maximalem Eigenwert λ_n und zugehörigen normierten Eigenvektoren x_1 bzw. x_n. Weiter sei $\delta A = 0$, $b = x_n$ und $\delta b = \epsilon x_1$ mit $\epsilon > 0$. Dann ist

$$\frac{\|\delta x\|_2}{\|x\|_2} = \epsilon \frac{\lambda_n}{\lambda_1} \frac{\|x_1\|_2}{\|x_n\|_2} = \text{cond}_2(A) \frac{\|\delta b\|_2}{\|b\|_2}.$$

□

5.2.2 Ausgleichsprobleme

Wir betrachten Gleichungssysteme

$$Ax = b$$

mit $A \in \mathbb{K}^{m \times n}$, $b \in \mathbb{K}^m$, bei denen A entweder nicht quadratisch oder singulär ist. In diesem Fall existieren entweder gar keine oder unendlich viele Lösungen des Gleichungssystems.

Wir wiederholen zunächst einige Ergebnisse aus der linearen Algebra.

$$\begin{aligned} \mathcal{N}(A) &:= \{x \in \mathbb{K}^n : Ax = 0\} \qquad \text{(Nullraum von } A\text{)} \\ \mathcal{R}(A) &:= \{Ax : x \in \mathbb{K}^n\} \qquad \text{(Wertebereich von } A\text{)} \\ M^\perp &:= \{\forall y \in M : x \in \mathbb{K}^n : x^* y = 0\} \qquad \text{(Orthogonalraum von } M\text{)}, \end{aligned}$$

wobei $M \subset \mathbb{K}^n$ eine beliebige Teilmenge ist.

Lemma 5.16. *Für eine Matrix $A \in \mathbb{K}^{m \times n}$ gilt*

1. *A^*A ist genau dann positiv definit, wenn $\mathcal{N}(A) = \{0\}$.*
2. *$\mathcal{N}(A^*A) = \mathcal{N}(A)$.*
3. *$\mathcal{R}(A^*) = \mathcal{N}(A)^\perp$.*

4. $\mathcal{R}(A^*A) = \mathcal{R}(A^*)$.

Beweis. 1. Es gilt $x^*A^*Ax = \|Ax\|_2^2 \geq 0$.

2. Aus $Ax = 0$ folgt $A^*Ax = 0$ und somit $\mathcal{N}(A^*A) \supset \mathcal{N}(A)$. Sei $A^*Ax = 0$, so folgt $x^*A^*Ax = 0$ und daher $\|Ax\|_2^2 = 0$, also $Ax = 0$. Dies beweist $\mathcal{N}(A^*A) \subset \mathcal{N}(A)$.

3. Sei $y = A^*x \in \mathcal{R}(A^*)$. Dann gilt für alle $z \in \mathcal{N}(A)$ die Beziehung

$$z^*y = z^*A^*x = (Az)^*x = 0,$$

woraus $y \in \mathcal{N}(A)^\perp$ folgt. Also gilt $\mathcal{R}(A^*) \subset \mathcal{N}(A)^\perp$. Nun folgt Gleichheit aus dem folgenden Dimensionsargument:

$$\dim \mathcal{N}(A)^\perp \quad = \quad n - \dim \mathcal{N}(A) = \dim \mathcal{R}(A) = \dim \mathcal{R}(A^*).$$

4. Mit Hilfe der Teile 2 und 3 erhalten wir

$$\mathcal{R}(A^*) \overset{3}{=} \mathcal{N}(A)^\perp \overset{2}{=} \mathcal{N}(A^*A)^\perp \overset{3}{=} \mathcal{R}(A^*A).$$

□

Satz 5.17 (Normalengleichung)**.** *$x \in \mathbb{K}^n$ löst genau dann das lineare Ausgleichsproblem*

$$\|Ax - b\|_2^2 = \min!,$$

wenn x die Gaußsche Normalengleichung

$$A^*Ax = A^*b \tag{5.6}$$

löst. Es existiert immer mindestens eine Lösung $x_0 \in \mathbb{K}^n$ des linearen Ausgleichsproblems, und die Menge aller solcher Lösungen ist gegeben durch

$$x_0 + \mathcal{N}(A).$$

Beweis. Aus Lemma 5.16, Teil 4 folgt, dass es eine Lösung x_0 der Normalengleichung gibt, und aus Teil 2, dass die Menge aller Lösungen der Normalengleichung die angegebene Form hat. Sei $\psi(x) := \|Ax - b\|_2^2$. Dann gilt

$$\begin{aligned}
\psi(x) - \psi(x_0) &= x^*A^*Ax - x_0^*A^*Ax_0 - 2\mathrm{Re}\, x^* \underbrace{A^*b}_{A^*Ax_0} + 2\mathrm{Re}\, x_0^* \underbrace{A^*b}_{A^*Ax_0} \\
&= (x - x_0)^*A^*A(x - x_0) \\
&= \|A(x - x_0)\|_2^2 \\
&\geq 0.
\end{aligned}$$

Daher ist x ein Minimum von ψ genau dann wenn $x - x_0 \in \mathcal{N}(A)$. Letzteres ist, wie oben diskutiert, genau dann der Fall, wenn x die Normalengleichungen löst. □

Satz 5.18 (Minimallösung). *Sei $A \in \mathbb{K}^{m\times n}$ und $b \in \mathbb{K}^m$. Unter den Lösungen des linearen Ausgleichsproblems $\|Ax-b\|_2^2 = \min!$ gibt es ein eindeutig bestimmtes Element $x^\dagger$ mit minimaler euklidischer Norm und es gilt*

$$x^\dagger \in N(A)^\perp. \tag{5.7}$$

Beweis. Sei x_0 eine Lösung des linearen Ausgleichsproblems und $x_0 = x^\dagger + z$ eine Zerlegung in einen Anteil $x^\dagger \in N(A)^\perp$ und einen Anteil $z \in N(A)$. Die allgemeine Lösung des Ausgleichsproblems ist dann $x^\dagger + z$ mit $z \in N(A)$, und wegen

$$\|x^\dagger + z\|_2^2 = \|x^\dagger\|_2^2 + 2\mathrm{Re}\,\underbrace{(x^\dagger)^* z}_{=0} + \|z\|_2^2$$

hat $x^\dagger$ minimale Norm und ist eindeutig. □

Mit Hilfe der Singulärwertzerlegung von A lässt sich diese eindeutige Minimallösung exakt angeben.

Satz 5.19 (Singulärwertzerlegung). *Sei $A \in \mathbb{K}^{m\times n}$. Dann existieren unitäre Matrizen $U \in \mathbb{K}^{m\times m}$ und $V \in \mathbb{K}^{n\times n}$ und eine verallgemeinerte Diagonalmatrix $\Sigma \in \mathbb{R}^{m\times n}$, d.h. $\Sigma_{ij} = \sigma_j \delta_{ij}$ mit $\sigma_j \geq 0$ für $j = 1, \ldots, \min\{n, m\}$, sodass gilt*

$$U^* A V = \Sigma.$$

Beweis. Lineare Algebra. Eine Möglichkeit des Beweises benutzt Satz 5.9(2) angewendet auf die hermitesche, positiv semidifinite Matrix A^*A. Die Singulärwerte σ_j sind dann die Wurzeln der Eigenwerte von A^*A. □

Definition 5.20 (Pseudo-Inverse). *Sei $A = U\Sigma V^*$ eine Singulärwertzerlegung von $A \in \mathbb{K}^{m\times n}$ mit $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0 \ldots, 0)$ mit $r = \mathrm{rang}(A)$. Dann heißt*

$$A^\dagger := V\Sigma^\dagger U^*, \qquad \Sigma^\dagger := \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0 \ldots, 0) \in \mathbb{R}^{n\times m} \tag{5.8}$$

eine Pseudo-Inverse *von A.*

Satz 5.21. *Sei $A = U\Sigma V^*$ eine Singulärwertzerlegung von $A \in \mathbb{K}^{m\times n}$ mit $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0 \ldots, 0)$ mit $r = \mathrm{rang}(A)$ und $A^\dagger$ wie in (5.8) definiert. Dann ist die eindeutige Minimallösung $x^\dagger$ des linearen Ausgleichsproblems $\|Ax - b\|_2^2 = \min!$ gegeben durch*

$$x^\dagger = A^\dagger b.$$

Beweis. Seien $b \in \mathbb{K}^m$, $u_i \in \mathbb{K}^m$ für $i = 1, \ldots, m$ die Spaltenvektoren von U und $v_i \in \mathbb{K}^n$ für $i = 1, \ldots, n$ die Spaltenvektoren von V. Dann gilt für jedes $x \in \mathbb{K}^n$

$$\begin{aligned}\|Ax - b\|_2^2 =& \|AVV^*x - b\|_2^2 = \|U^*AVV^*x - U^*b\|_2^2 = \|\Sigma V^*x - U^*b\|_2^2 \\ =& \sum_{i=1}^{r} |\sigma_i (x, v_i)_2 - (b, u_i)_2|^2 + \sum_{i=r+1}^{m} |(b, u_i)_2|^2\end{aligned}$$

Aus der ersten Summe folgt direkt für ein Minimum x von $\|Ax-b\|_2^2$ dass $v_i^* x = (x, v_i)_2 = \sigma_i^{-1} u_i^* b$ für $i = 1, \dots, r$. Weiter ist $\|x\|_2 = \|V^* x\|_2$ genau dann minimal, wenn $v_i^* x = 0$ für $i = r+1, \dots, n$, was genau (5.7) entspricht. Insgesamt ergibt sich

$$x^\dagger = \sum_{i=1}^{r} \frac{u_i^* b}{\sigma_i} v_i$$

oder in Kurzform $x^\dagger = A^\dagger b$. □

Satz 5.22 (Störungssatz für Ausgleichsprobleme)**.** *Sei $A \in \mathbb{K}^{m\times n}$, $b, \tilde{b} \in \mathbb{K}^m$ und $x^\dagger = A^\dagger b$ sowie $\tilde{x}^\dagger = A^\dagger \tilde{b}$. Dann gibt es orthogonale Zerlegungen $b = b_R + b_N$ und $\tilde{b} = \tilde{b}_R + \tilde{b}_N$ mit $b_R, \tilde{b}_R \in \mathcal{R}(A)$ und $b_N, \tilde{b}_N \in \mathcal{N}(A^*) = \mathcal{R}(A)^\perp$ und es gilt*

$$\frac{\left\|x^\dagger - \tilde{x}^\dagger\right\|_2}{\|x^\dagger\|_2} \le \operatorname{cond}_2(A) \frac{\left\|b_R - \tilde{b}_R\right\|_2}{\|b_R\|_2}.$$

Dabei ist

$$\operatorname{cond}_2(A) := \|A\|_2 \|A^\dagger\|_2 = \frac{\sigma_1}{\sigma_r},$$

wobei σ_1 der größte Singulärwert und σ_r der kleinste, nicht verschwindende Singulärwert von A ist.

Beweis. Die Existenz der Zerlegungen von b und $\tilde{b}$ folgt aus Lemma 5.16(3) mit vertauschten Rollen von A und A^*. Für $y_N \in \mathcal{R}(A)^\perp$ und $y \in \mathbb{K}^n$ folgt aus dem Satz des Pythargoras

$$\|Ay - y_N\|_2 = \|Ay\|_2 + \|y_N\|_2$$

und daraus $A^\dagger y_N = 0$ für beliebiges $y_N \in \mathcal{R}(A)^\perp$. Damit ist

$$x^\dagger - \tilde{x}^\dagger = A^\dagger(b_R - \tilde{b}_R + b_N - \tilde{b}_N) = A^\dagger(b_R - \tilde{b}_R) + \underbrace{A^\dagger(b_N - \tilde{b}_N)}_{=0} = A^\dagger(b_R - \tilde{b}_R)$$

und

$$x^\dagger = A^\dagger(b_R + b_N) = A^\dagger b_R \quad \Rightarrow \quad Ax^\dagger = b_R.$$

Damit folgt die Behauptung aus (5.1). Die Darstellung der spektralen Konditionszahl folgt mit Satz 5.11. □

Weitere Details zur Störungstheorie bei Ausgleichsproblemen wie z.B. Störungen in der Matrix A findet man z.B. in Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithm*, Society for Industrial and Applied Mathematics (SIAM), 1996.

6 Lineare Gleichungssysteme (direkte Verfahren)

Wir besprechen in diesem Kapitel die numerische Lösung linearer Gleichungssysteme mit Hilfe von direkten Verfahren, welche in endlich vielen Schritten unter Vernachlässigung von Rundungsfehlern die exakte Lösung berechnen. Später werden wir iterative Verfahren kennenlernen, bei denen Folgen von Lösungen berechnet werden, die zwar gegen die exakte Lösung konvergieren, diese jedoch in der Regel nach endlich vielen Schritten nicht erreichen.

6.1 Gauß-Elimination und LU-Zerlegung

Wir betrachten zunächst lineare Gleichungssysteme

$$Ax = b$$

mit einer regulären Matrix $A \in \mathbb{K}^{n\times n}$ und $x, b \in \mathbb{K}^n$.

6.1.1 Elimination bei Dreiecksmatrizen

Betrachte ein Gleichungssystem in unterer Dreiecksgestalt,

$$\begin{array}{lllll} l_{11}x_1 & & & & = b_1 \\ l_{21}x_1 & +l_{22}x_2 & & & = b_2 \\ \vdots & & \ddots & & = \vdots \\ l_{n1}x_1 & +l_{n2}x_2 & +\ldots & +l_{nn}x_n & = b_n \end{array}$$

mit $l_{ii} \neq 0$ für $i = 1, \ldots, n$. Seine Lösung ist gegeben durch *Vorwärtselimination* (auch *Vorwärtssubstitution* genannt)

$$\begin{aligned} x_1 &= \tfrac{b_1}{l_{11}} \\ x_2 &= \tfrac{1}{l_{22}}\Big(b_2 - l_{21}x_1\Big) \\ &\vdots \\ x_j &= \tfrac{1}{l_{jj}}\Big(b_j - \textstyle\sum_{i=1}^{j-1} l_{ji}x_i\Big), \quad j = 1, \ldots, n. \end{aligned}$$

Algorithmus 6.1 Vorwärtssubstitution

Input: $L \in \mathbb{K}^{n\times n}$ sei eine invertierbare linke untere Dreiecksmatrix, $b \in \mathbb{K}^n$.

1: **for** $i = 1, \ldots, n$ **do**

2: $\quad x_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j\right) / l_{ii}$

3: **end for**

Output: $x = L^{-1} b$

Algorithmus 6.2 Rückwärtssubstitution

Input: $U \in \mathbb{K}^{n\times n}$ sei eine invertierbare rechte obere Dreiecksmatrix, $b \in \mathbb{K}^n$.

1: **for** $i = n, \ldots, 1$ **do**

2: $\quad x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j\right) / u_{ii}$

3: **end for**

Output: $x = U^{-1} b$

Analog können wir auch ein Gleichungssystem in oberer Dreiecksgestalt,

$$\begin{array}{lllll} u_{11}x_1 & +u_{12}x_2 & +\ldots & +u_{1n}x_n & = b_1 \\ & +u_{22}x_2 & +\ldots & +u_{2n}x_n & = b_2 \\ & & \ddots & & = \vdots \\ & & & u_{nn}x_n & = b_n \end{array}$$

betrachten, mit $u_{ii} \neq 0$ für $i = 1, \ldots, n$. Hier ist die Lösung gegeben durch *Rückwärtselimination* (auch *Rückwärtssubstitution*)

$$\begin{array}{lll} x_n & = & \frac{b_n}{u_{nn}} \\ x_{n-1} & = & \frac{1}{u_{n-1,n-1}}\left(b_{n-1} - u_{n-1,n}x_n\right) \\ \vdots & & \\ x_j & = & \frac{1}{u_{jj}}\left(b_j - \sum_{i=j+1}^{n} u_{ji}x_i\right), \quad j = n, n-1, \ldots, 1. \end{array}$$

Die Lösung von Gleichungssystemen mit einer unteren oder oberen Dreiecksmatrix durch Vorwärts- bzw. Rückwärtselimination ist sehr einfach (siehe Alg. 6.1 bzw. 6.2). Zur Beschreibung des Rechenaufwands benutzen wir den Begriff des flops. Unter einem *flop* (engl. floating point operation) versteht man eine Multiplikation mit anschließender Addition, also $a + b * c$. Eine Division zählen wir ebenfalls als flop. Die Anzahl der benötigten flops bei Vorwärts- und Rückwärtselimination beträgt

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + O(n),$$

sie wächst also mit der zweiten Potenz der Anzahl der Unbekannten.

6.1.2 LU-Zerlegung

Bei der Gauß-Elimination wird ein allgemeines, quadratisches lineares Gleichungssystem auf Dreiecksgestalt gebracht.

Beispiel 6.3. *Gesucht sei die Lösung des linearen Gleichungssystems*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned}$$

mit $A := \left(\begin{smallmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{smallmatrix}\right) \in \mathbb{K}^{2\times 2}$, *A regulär und* $a_{11} \neq 0$. *Multiplikation der 1.Gleichung mit* $-a_{21}/a_{11}$ *und Addition zur 2.Gleichung führt zu*

$$\begin{aligned} a_{11}x_1 + \qquad a_{12}\,x_2 \qquad &= b_1 \\ \left(a_{22} - \frac{a_{21}a_{12}}{a_{11}}\right) x_2 \quad &= b_2 - \frac{a_{21}}{a_{11}} b_1 \end{aligned}$$

und damit zur Lösung

$$x_2 = \frac{b_2 - \frac{a_{21}}{a_{11}} b_1}{a_{22} - \frac{a_{21}a_{12}}{a_{11}}}, \qquad x_1 = \frac{b_1 - a_{12}x_2}{a_{11}}.$$

Dieser Gauß-Algorithmus kann aufgefasst werden als Faktorisierung der Matrix $A = LU$ *mit normierter unterer Dreiecksmatrix* $L = \left(\begin{smallmatrix} 1 & 0 \\ \frac{a_{21}}{a_{11}} & 1 \end{smallmatrix}\right)$ *und oberer Dreiecksmatrix* $U = \left(\begin{smallmatrix} a_{11} & a_{12} \\ 0 & a_{22} - \frac{a_{21}a_{12}}{a_{11}} \end{smallmatrix}\right)$ *gefolgt von einer Vorwärtssubstitution für* $Ly = b$ *und Rückwärtssubstitution für* $Ux = y$.

Definition 6.4 (LU-Zerlegung)**.** *Eine Zerlegung einer quadratischen Matrix* $A \in \mathbb{K}^{n\times n}$ *in ein Produkt*

$$A = LU$$

einer unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix U heißt LU-Zerlegung *(engl. "lower" bzw. "upper") von A. Teilweise spricht man auch von einer* LR-Zerlegung.

6.1.3 Gaußsches Eliminationsverfahren ohne Pivotsuche

Wir wollen nun zunächst die Gaußschen Umformungsschritte in ihrer Matrixform kennenlernen.

Definition 6.5. *Sei für* $k \in \{1, \ldots, n\}$ *ein Vektor* $\tau^{(k)} = (0, \ldots, 0, t_{k+1}, \ldots, t_n)^\top \in \mathbb{K}^n$ *gegeben und sei* e_k *der k-te Einheitsvektor* $e_k := (0, \ldots, 0, 1, 0, \ldots, 0)^\top$. *Als* Gauß-Matrix *bezeichnet man eine Matrix der Form*

$$M_k := I - \tau^{(k)} e_k^\top = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -t_{k+1} & 1 & \\ & & \vdots & & \ddots \\ & & -t_n & & 1 \end{pmatrix}.$$

Gauß-Matrizen sind regulär, da $\det(M_k) = 1 \neq 0$. Die Multiplikation einer Gauß-Matrix M_k von links mit einer Matrix $B \in \mathbb{K}^{n\times n}$ bedeutet, für $j = k+1, \ldots, n$ t_j mal die k-te Zeile von B von der j-ten Zeile von B zu subtrahieren:

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -t_{k+1} & 1 & \\ & & \vdots & & \ddots \\ & & -t_n & & 1 \end{pmatrix} \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ b_{k1} & \cdots & b_{kn} \\ \vdots & & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ b_{k1} & \cdots & b_{kn} \\ b_{(k+1)1} - t_{k+1}b_{k1} & \cdots & b_{(k+1)n} - t_n b_{kn} \\ \vdots & & \vdots \\ b_{n1} - t_n b_{k1} & \cdots & b_{nn} - t_n b_{kn} \end{pmatrix}$$

Sei $x \in \mathbb{K}^n$ ein Vektor und $k \in \{1, \ldots, n\}$ mit $x_k \neq 0$. Für $j = k+1, \ldots, n$ setze $t_j := \frac{x_j}{x_k}$ und bezeichne M_k die zugehörige Gauß-Matrix. Dann gilt

$$M_k x = (x_1, \ldots, x_k, 0, \ldots, 0)^\top. \tag{6.1}$$

Wir haben die Gauß-Matrizen hauptsächlich eingeführt, um die Formulierung und die Analyse des Gauß–Algorithmus übersichtlicher zu gestalten. Bei einer praktischen Implementierung des Gauß–Algorithmus sollten diese Matrizen aus Effizienzgründen keinesfalls aufgestellt werden!

Algorithmus 6.6 Gauß–Algorithmus ohne Pivotsuche (Matrixversion)

Input: $A \in \mathbb{K}^{n\times n}$

1: $A^{(1)} = A$
2: **for** $k = 1. \ldots, n-1$ **do**
3: **if** $a_{kk}^{(k)} = 0$ **then**
4: STOP: Algorithmus nicht durchführbar
5: **else**
6: $\tau^{(k)} = \Big(\underbrace{0, \ldots, 0}_{k-mal}, \frac{a_{k+1,k}^{(k)}}{a_{kk}^{(k)}}, \ldots, \frac{a_{n,k}^{(k)}}{a_{kk}^{(k)}}\Big)^\top$
7: $M_k = I_n - \tau^{(k)} e_k^\top$
8: $A^{(k+1)} = M_k A^{(k)}$
9: **end if**
10: **end for**

Output: Wenn Algorithmus durchführbar, dann ist $U := A^{(n)}$ eine obere Dreiecksmatrix, $L := M_1^{-1} \ldots M_{n-1}^{-1}$ eine normierte untere Dreiecksmatrix, und es gilt $A = LU$.

Satz 6.7. *Wenn der Gauß–Algorithmus ohne Pivotsuche durchführbar ist, dann gilt:*

1. $a_{ij}^{(k+1)} = 0$ *für* $i > j$ *und* $k+1 > j$. *Insbesondere ist* U *eine obere Dreiecksmatrix.*
2. $A = LU$.
3. $L = I + \sum_{k=1}^{n-1} \tau^{(k)} e_k^\top$. *Insbesondere ist* L *eine normierte untere Dreiecksmatrix.*

Beweis. Der erste Punkt folgt direkt aus dem Algorithmus mit (6.1), der zweite aus

$$U = A^{(n)} = M_{n-1} A^{(n-1)} = \cdots = M_{n-1} M_{n-2} \ldots M_1 A = L^{-1} A.$$

Dabei ist zu beachten, dass die Gauß-Matrizen M_k regulär sind. Aus

$$\left(I + \tau^{(k)} e_k^\top\right)\left(I - \tau^{(k)} e_k^\top\right) = I - \tau^{(k)} \underbrace{e_k^\top \tau^{(k)}}_{=0} e_k^\top = I$$

folgt

$$M_k^{-1} = I + \tau^{(k)} e_k^\top.$$

Wir zeigen per Induktion über $j = 1, \ldots, n-1$

$$M_{n-j}^{-1} \cdots M_{n-1}^{-1} = I + \sum_{k=n-j}^{n-1} \tau^{(k)} e_k^\top.$$

Der Fall $j = 1$ ist schon gezeigt. Der Schluss von $j-1$ auf j ergibt sich wie folgt:

$$\begin{aligned} M_{n-j}^{-1}\left(M_{n-j+1}^{-1} \cdots M_{n-1}^{-1}\right) &= \left(I + \tau^{(n-j)} e_{n-j}^\top\right)\left(I + \sum_{k=n-j+1}^{n-1} \tau^{(k)} e_k^\top\right) \\ &= I + \sum_{k=n-j}^{n-1} \tau^{(n-j)} e_{n-j}^\top + \tau^{(n-j)} \sum_{k=n-j+1}^{n-1} \underbrace{\left(e_{n-j}^\top \tau^{(k)}\right)}_{=0} e_k^\top \end{aligned} \tag{6.2}$$

und der Beweis ist vollständig. □

Korollar 6.8. *Ist $A = LU$ eine LU-Zerlegung der Matrix A mit normierter untererer Dreiecksmatrix L und oberer Dreiecksmatrix U, so gilt*

$$\det(A) = \prod_{j=1}^{n} u_{jj}. \tag{6.3}$$

Beweis. Nach dem Determinanten-Multiplikationssatz haben wir

$$\det(A) = \det(L)\,\det(U).$$

Die Determinante einer Dreiecksmatrix ist das Produkt ihrer Diagonalelemente. Die Matrix L hat in der Diagonalen nur Einsen, somit folgt die Aussage (6.3). □

Lemma 6.9. *Besitzt eine reguläre Matrix $A \in \mathbb{K}^{n\times n}$ eine LU-Zerlegung $A = LU$ mit normierter unterer Dreiecksmatrix L, so ist die LU-Zerlegung eindeutig bestimmt.*

Beweis. Angenommen, es seien

$$A = L_1 U_1 = L_2 U_2$$

zwei LU-Zerlegungen. Dann gilt

$$L_2^{-1} L_1 = U_2 U_1^{-1}.$$

Wegen Lemma 5.8 steht links eine normierte untere Dreiecksmatrix und rechts eine obere Dreiecksmatrix. Die Gleichheit kann nur erfüllt sein, wenn

$$L_2^{-1} L_1 = U_2 U_1^{-1} = I$$

die Einheitsmatrix ist. Es folgt $L_2 = L_1$ und $U_2 = U_1$. □

6.1.4 Direkte LU-Zerlegung und Stabilität

Der Algorithmus 6.6 sollte keinesfalls so implementiert werden. Wenn eine Matrix $A \in \mathbb{K}^{n\times n}$ eine LU-Zerlegung besitzt, können die n^2 Gleichungen $A = LU$ zur Bestimmung der n^2 Unbekannten l_{ij} $(i > j,\ l_{jj} = 1)$ und u_{ij} $(i \le j)$ verwendet werden.

Algorithmus 6.10 Gauß–Algorithmus ohne Pivotsuche (Direkte Zerlegung)

Input: $A \in \mathbb{K}^{n\times n}$

1: **for** $k = 1, \dots, n$ **do**
2: **for** $j = k, \dots, n$ **do**
3: $u_{kj} = a_{kj} - \sum_{i=1}^{k-1} l_{ki}u_{ij}$
4: **end for**
5: **if** $u_{kk} = 0$ **then**
6: STOP: Algorithmus nicht durchführbar
7: **else**
8: **for** $j = k+1, \dots, n$ **do**
9: $l_{jk} = u_{kk}^{-1}\left(a_{jk} - \sum_{i=1}^{k-1} l_{ji}u_{ik}\right)$
10: **end for**
11: **end if**
12: **end for**

Output: Wenn Algorithmus durchführbar, dann ist $A = LU$ mit der oberen Dreiecksmatrix $U = (u_{ij})_{i,j=1}^{n}$ der normierten unteren Dreiecksmatrix $L = (l_{ij})_{i,j=1}^{n}$.

Der Algorithmus 6.10 benötigt $\mathcal{O}(n^3)$ Rechenoperationen, welche in den Zeilen 3 und 9 stattfinden. Zur Untersuchung der Stabilität verwenden wir die Annahme (2.11), d.h. die Grundrechenarten werden mit einem relativen Fehler ausgeführt, welcher vom Betrage her kleiner als die Rechengenauigkeit eps ist.

Lemma 6.11. *Seien $\delta_i \in \mathbb{K}$ mit $|\delta_i| < \text{eps}$, $\rho_i \in \{-1, 1\}$ für $i = 1, \dots, n$ und $n\,\text{eps} < 1$.*

Dann ist

$$\left|\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} - 1\right| \le \gamma_n := \frac{n\,\text{eps}}{1 - n\,\text{eps}}.$$

Beweis. Wir setzen zur Abkürzung $\theta_n := \prod_{i=1}^{n}(1+\delta_i)^{\rho_i} - 1$ und beweisen die Aussage per Induktion über n. Für $n = 1$ gilt die Behauptung, da

1. $\rho_1 = 1$: $|\theta_1| = |(1+\delta_1) - 1| \le \text{eps} \le \frac{\text{eps}}{1-\text{eps}}$.
2. $\rho_1 = -1$: $|\theta_1| = \left|\frac{1}{1+\delta_1} - 1\right| = \left|\frac{-\delta_1}{1+\delta_1}\right| \le \frac{|\delta_1|}{1-|\delta_1|} \le \frac{\text{eps}}{1-\text{eps}}$.

Für $n - 1 \to n$ erhalten wir

1. $\rho_n = 1$:

$$\begin{aligned}|\theta_n| =& \left|\prod_{i=1}^{n-1}(1+\delta_i)^{\rho_i}(1+\delta_n) - 1\right| = \left|(1+\delta_n)\left(\prod_{i=1}^{n-1}(1+\delta_i)^{\rho_i} - 1\right) + \delta_n\right| \\ \leq& (1+|\delta_n|)\left|\prod_{i=1}^{n-1}(1+\delta_i)^{\rho_i} - 1\right| + |\delta_n| \leq (1+\text{eps})\frac{(n-1)\,\text{eps}}{1-(n-1)\,\text{eps}} + \text{eps} \\ =& \frac{(n-1)\,\text{eps} + (n-1)\,\text{eps}^2 + \text{eps} - (n-1)\,\text{eps}^2}{1-(n-1)\,\text{eps}} \leq \frac{n\,\text{eps}}{1-n\,\text{eps}}.\end{aligned}$$

2. $\rho_n = -1$:

$$\begin{aligned}|\theta_n| =& \left|\prod_{i=1}^{n-1}(1+\delta_i)^{\rho_i}(1+\delta_n)^{-1} - 1\right| = \left|\frac{1}{1+\delta_n}\left(\prod_{i=1}^{n-1}(1+\delta_i)^{\rho_i} - 1\right) - \frac{\delta_n}{1+\delta_n}\right| \\ \leq& \frac{1}{(1-\text{eps})}\frac{(n-1)\,\text{eps}}{(1-(n-1)\,\text{eps})} + \frac{\text{eps}}{1-\text{eps}} = \frac{n\,\text{eps} - (n-1)\,\text{eps}^2}{1-n\,\text{eps} + (n-1)\text{eps}^2} \\ \leq& \frac{n\,\text{eps}}{1-n\,\text{eps}}.\end{aligned}$$

□

Mit diesem Hilfsresultat erhalten wir folgendes Stabilitätsresultat zu Algorithmus 6.10.

Satz 6.12 (Rückwärtsstabilität der direkten LU Zerlegung)**.** *Falls Algorithmus 6.10 nicht abbricht und* $n\,\text{eps} < 1$*, dann sind die rundungsfehlerbehafteten Größen* $\hat{L}$ *und* $\hat{U}$ *die exakte LU-Zerlegung einer Matrix* $\hat{A} := A + \Delta A$*, wobei gilt*

$$|\Delta A| \leq \gamma_n |\hat{L}|\,|\hat{U}|. \tag{6.4}$$

Hier ist $|\cdot|$ *und auch* $\leq$ *komponentenweise zu verstehen, d.h. für* $B \in \mathbb{K}^{n\times n}$ *ist* $|B| := (|b_{ij}|)_{i,j=1}^n \in \mathbb{R}_{\geq 0}^{n\times n}$.

Beweis. Wir reformulieren zunächst die Zeile 3 von Alg. 6.10 mit fehlerbehafteten Rechenoperationen für fixe k, j mit $k \leq j$:

$$\begin{aligned} s_1 &= \Big(a_{kj} - (\hat{l}_{k1}\hat{u}_{1j})(1+\epsilon_1)\Big)(1+\delta_1), \\ s_i &= \Big(s_{i-1} - (\hat{l}_{ki}\hat{u}_{ij})(1+\epsilon_i)\Big)(1+\delta_i), \qquad i = 2,\dots,k-1 \\ \hat{u}_{kj} &= s_{k-1}, \end{aligned}$$

wobei $|\epsilon_i|, |\delta_i| \leq \text{eps}$ für $i = 1,\dots,k-1$. Insgesamt ergibt sich

$$\hat{u}_{kj} = a_{kj}\prod_{\ell=1}^{k-1}(1+\delta_\ell) - \sum_{i=1}^{k-1}\hat{l}_{ki}\hat{u}_{ij}\left((1+\epsilon_i)\prod_{\ell=i}^{k-1}(1+\delta_\ell)\right)$$

und daraus nach Division durch $\prod_{\ell=1}^{k-1}(1+\delta_\ell)$ mit der Notation des vorigen Lemmas

$$\hat{u}_{kj}(1+\theta_{k-1}) = a_{kj} - \sum_{i=1}^{k-1} \hat{l}_{ki}\hat{u}_{ij}(1+\theta_{i-1})(1+\epsilon_i).$$

Mit Hilfe des vorigen Lemmas und $\gamma_{i+1} \geq \gamma_i$ für $i = 1, \ldots, k-1$ erhalten wir wegen $\hat{l}_{kk} = 1$

$$\left| a_{kj} - \sum_{i=1}^{n} \hat{l}_{ki}\hat{u}_{ij} \right| \leq |\hat{u}_{kj}|\gamma_{k-1} + \sum_{i=1}^{k-1} |\hat{l}_{ki}|\,|\hat{u}_{ij}|\gamma_i \leq \gamma_{n-1} \sum_{i=1}^{n} |\hat{l}_{ki}|\,|\hat{u}_{ij}|. \tag{6.5a}$$

Analog aber mit einer Rechenoperation mehr folgert man aus Zeile 9 für $k > j$

$$\left| a_{kj} - \sum_{i=1}^{n} \hat{l}_{ki}\hat{u}_{ij} \right| \leq \gamma_n \sum_{i=1}^{n} |\hat{l}_{ki}|\,|\hat{u}_{ij}|. \tag{6.5b}$$

Aus $\Delta A = A - \hat{L}\hat{U}$ folgt dann die Behauptung. □

Wir sehen, dass bei der Vorwärts- bzw. Rückwärtssubstitution (Alg. 6.1 und Alg. 6.2) die gleichen Rechenoperationen wie in der Zeile 9 von Alg. 6.10 ausgeführt werden. Damit ist folgendes Resultat naheliegend.

Satz 6.13. *Wenn das Gleichungssystem $Tx = b$ mit regulärer Dreiecksmatrix $T \in \mathbb{K}^{n\times n}$ durch Vorwärts- bzw. Rückwärtssubstitution rundungsfehlerbehaftet gelöst wird, so ist die fehlerbehaftete Lösung $\hat{x} \in \mathbb{K}^n$ die exakte Lösung des Gleichungssystems $(T + \Delta T)\hat{x} = b$ mit $|\Delta T| \leq \gamma_n |T|$.*

Beweis. Siehe Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithm*, Society for Industrial and Applied Mathematics (SIAM), 1996. □

Satz 6.14. *Sei $A \in \mathbb{K}^{n\times n}$ regulär, $x \in \mathbb{K}^n$ Lösung von $Ax = b$ und der (rundungsfehlerbehaftete) Alg. 6.10 breche für A nicht ab.*

Dann bricht auch Alg. 6.2 nicht ab und es ist die rundungsfehlerbehaftete Lösung $\hat{x}$ aus Alg. 6.10 gefolgt von Alg. 6.1 und Alg. 6.2 die exakte Lösung von $(A+\Delta A)\hat{x} = b$ mit $|\Delta A| \leq (3\gamma_n + \gamma_n^2)|\hat{L}||\hat{U}|$.

Beweis. Wir wissen aus den letzten beiden Sätzen, dass $\hat{L}\hat{U} = A + \Delta A_1$ mit $|\Delta A_1| \leq \gamma_n |\hat{L}||\hat{U}|$ und dass die Ergebnisse $\hat{y}$ von Alg. 6.1 und $\hat{x}$ von Alg. 6.2 exakte Lösungen sind von

$$\begin{aligned} (\hat{L} + \Delta L)\hat{y} &= b, & |\Delta L| &\leq \gamma_n |\hat{L}|, \\ (\hat{U} + \Delta U)\hat{x} &= \hat{y}, & |\Delta U| &\leq \gamma_n |\hat{U}|. \end{aligned}$$

Insgesamt folgt

$$b = (\hat{L} + \Delta L)(\hat{U} + \Delta U)\hat{x} = \left(A + \underbrace{\Delta A_1 + \hat{L}\Delta U + \Delta L\hat{U} + \Delta L\Delta U}_{=\Delta A} \right) \hat{x}$$

und daraus die Behauptung. □

Wenn $(3\gamma_n + \gamma_n^2) \left\| |\hat{L}||\hat{U}| \right\| \leq \|A^{-1}\|^{-1}$ gilt, können wir nach der Rechnung mit Hilfe des Störungssatzes 5.3 den maximalen Rundungsfehler abschätzen.

Eine umfassendere Stabilitätstheorie findet sich in Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithm*, Society for Industrial and Applied Mathematics (SIAM), 1996.

6.1.5 Pivotisierung

Das Gauß-Verfahren ohne Pivot-Suche ist nicht immer durchführbar. Selbst wenn es durchführbar ist, kann es ungewünschte Effekte geben, wenn durch kleine Zahlen a_{kk} dividiert wird. Bei der Pivotisierung handelt es sich um ein Verfahren, das die Durchführbarkeit des Gaußschen Eliminationsverfahrens sicherstellt. Darüber hinaus verkleinert die Pivotisierung numerische Fehler bei der Lösung linearer Gleichungssysteme. Die Grundidee der Pivotisierung ist der Austausch von Zeilen und/oder Spalten während der Durchführung des Gaußschen Eliminationsverfahrens.

Definition 6.15. *Eine* Permutation *auf* $\{1, \ldots, n\}$ *ist eine bijektive Abbildung*

$$\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}.$$

Die zugehörige Permutationsmatrix $P \in \mathbb{R}^{n \times n}$ *ist gegeben durch*

$$P_\pi := (\delta_{\pi(i),j})_{i,j=1,\ldots,n}.$$

Für $i, j \in \{1, \ldots, n\}$ *bezeichnen wir mit* $\pi_{i \leftrightarrow j}$ *die Permutation*

$$\pi_{i \leftrightarrow j}(i) := j, \qquad \pi_{i \leftrightarrow j}(j) := i, \qquad \pi_{i \leftrightarrow j}(l) := l \quad \text{für } l \neq j, i.$$

Für die entsprechenden Permutationsmatrizen schreiben wir kurz $P_{i \leftrightarrow j} := P_{\pi_{i \leftrightarrow j}}$. *Falls* $i = j$, *setzen wir* $\pi_{i \leftrightarrow i} := id$.

Bemerkung 6.16. 1. *Es gilt*

$$P_\pi \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_{\pi^{-1}(1)} \\ \vdots \\ x_{\pi^{-1}(n)} \end{pmatrix}.$$

Insbesondere bedeutet Multiplikation von $P_{i \leftrightarrow j}$ *von links an eine Matrix* $A \in \mathbb{K}^{n \times m}$ *Vertauschung der* i*-ten und* j*-ten Zeile.*

2. *Es gilt*

$$(x_1, \ldots, x_n)\, P_\pi = \left(x_{\pi(1)}, \ldots, x_{\pi(n)}\right)$$

Insbesondere bedeutet Multiplikation von $P_{i \leftrightarrow j}$ *von rechts an eine Matrix* $A \in \mathbb{K}^{n \times m}$ *Vertauschung der* i*-ten und* j*-ten Spalte.*

3. *Die Permutationen bilden bezüglich der Hintereinanderausführung eine Gruppe. Es gilt*
$$P_{\pi_1} P_{\pi_2} = P_{\pi_1 \circ \pi_2}.$$

4. *Permutationsmatrizen sind* orthogonal*, d.h. es gilt*
$$P^T P = P P^T = I.$$
Insbesondere haben sie die Determinante ± 1.

Definition 6.17 (Pivot-Strategien)**.** *Sei* $A \in \mathbb{K}^{n\times n}$ *und* $A^{(k)}$ *für* $k = 1, \dots, n$ *die Matrix im* k*-ten Schritt von Alg. 6.6. Wir unterscheiden drei Pivot-Strategien:*

1. Spaltenpivotisierung: *Suche* $i \in \{k, \dots, n\}$ *mit* $|a_{ik}^{(k)}| \geq |a_{lk}^{(k)}|$ *für alle* $l = k, \dots, n$.
2. Zeilenpivotisierung: *Suche* $j \in \{k, \dots, n\}$ *mit* $|a_{kj}^{(k)}| \geq |a_{kl}^{(k)}|$ *für alle* $l = k, \dots, n$.
3. Totalpivotisierung: *Suche* $i, j \in \{k, \dots, n\}$ *mit* $|a_{ij}^{(k)}| \geq |a_{lm}^{(k)}|$ *für alle* $l, m = k, \dots, n$.

In allen Fällen heißt das jeweilige $a_{ij}^{(k)}$ *das* Pivotelement.

Wir können nun zu Alg. 6.6 eine beliebige Pivot-Strategie hinzufügen.

Algorithmus 6.18 Gauß–Algorithmus mit Spaltenpivotisierung (Matrixversion)

Input: $A \in \mathbb{K}^{n\times n}$ regulär

$A^{(0)} = A$
$M_0 = I$
for $k = 1. \dots, n-1$ **do**
  $\tilde{A}^{(k)} = M_{k-1} A^{(k-1)}$
  Suche $i \in \{k, \dots, n\}$ mit $|\tilde{a}_{ik}^{(k)}| \geq |\tilde{a}_{lk}^{(k)}|$ für alle $l = k, \dots, n$
  $P^{(k)} = P_{i \leftrightarrow k}$
  $A^{(k)} = P^{(k)} \tilde{A}^{(k)}$
  $\tau^{(k)} = \Big(\underbrace{0, \dots, 0}_{k-mal}, \frac{a_{k+1,k}^{(k)}}{a_{kk}^{(k)}}, \dots, \frac{a_{n,k}^{(k)}}{a_{kk}^{(k)}} \Big)^\top$
  $M_k = I_n - \tau^{(k)} e_k^\top$
end for
$A^{(n)} = M_{n-1} A^{(n-1)}$

Output: Es ist $PA = LU$ mit $P = P^{(n-1)} \cdots P^{(1)}$, $U := A^{(n)}$ eine obere Dreiecksmatrix, $L := I + \sum_{k=1}^{n-1} \theta^{(k)} e_k^\top$ eine normierte untere Dreiecksmatrix mit $\theta^{(k)} = P^{(n-1)} \dots P^{(k+1)} \tau^{(k)}$.

Satz 6.19. *Der Gauß–Algorithmus mit Spaltenpivotsuche ist durchführbar und liefert das versprochene Ergebnis.*

Beweis. Es gilt $a_{kk}^{(k)} \neq 0$ für $k = 1, \dots, n$, da A regulär ist und somit in der k-ten Spalte von $\tilde{A}^{(k)}$ ein nicht-trivialer Eintrag enthalten sein muss. Der Algorithmus ist damit durchführbar. Per Konstruktion von M_k, $k = 1, \dots, n-1$ ist

$$U = M_{n-1} P^{(n-1)} M_{n-2} P^{(n-2)} \cdots M_1 P^{(1)} A$$

eine obere Dreiecksmatrix, da die Zeilenvertauschung im k-ten Schritt die ersten $k-1$ Zeilen nicht verändert. Aus diesem Grund ist auch $P^{(k)} e_l = e_l$ für $l = 1, \dots, k-1$ und wir erhalten

$$P^{(n-1)} M_{n-2} = P^{(n-1)} \left(I - \tau^{(n-2)} e_{n-2}^\top\right) = \left(I - \theta^{(n-2)} e_{n-2}^\top\right) P^{(n-1)}.$$

Induktiv folgt

$$U = \left(I - \theta^{(n-1)} e_{n-1}^\top\right) \cdots \left(I - \theta^{(1)} e_1^\top\right) PA$$

und daraus wie im Beweis zu Satz 6.7 die Behauptung. □

Die folgende, Matrix-freie Version des Gauß-Algithmus speichert zusätzlich zur Matrix $A \in \mathbb{K}^{n \times n}$ lediglich Permutationsvektoren π_r bzw. π_c. Auch werden die Vertauschungen nicht direkt durchgeführt, sondern durch Zeiger realisiert.

Algorithmus 6.20 Gauß–Algorithmus mit Pivotisierung

Input: $A \in \mathbb{K}^{n \times n}$ regulär

1: $\pi_\mathrm{r} = (1, \dots, n)$
2: $\pi_\mathrm{c} = (1, \dots, n)$
3: **for** $k = 1. \dots, n-1$ **do**
4: Suche Pivotelement $a_{\pi_\mathrm{r}(i), \pi_\mathrm{c}(j)}$ mit $i, j \in \{k, \dots, n\}$
5: $temp = \pi_\mathrm{r}(k)$, $\pi_\mathrm{r}(k) = \pi_\mathrm{r}(i)$, $\pi_\mathrm{r}(i) = temp$
6: $temp = \pi_\mathrm{c}(k)$, $\pi_\mathrm{c}(k) = \pi_\mathrm{c}(j)$, $\pi_\mathrm{c}(j) = temp$
7: **for** $i = k+1, \dots n$ **do**
8: $a_{\pi_\mathrm{r}(i), \pi_\mathrm{c}(k)} = \frac{a_{\pi_\mathrm{r}(i), \pi_\mathrm{c}(k)}}{a_{\pi_\mathrm{r}(k), \pi_\mathrm{c}(k)}}$
9: **for** $j = k+1, \dots n$ **do**
10: $a_{\pi_\mathrm{r}(i), \pi_\mathrm{c}(j)} = a_{\pi_\mathrm{r}(i), \pi_\mathrm{c}(j)} - a_{\pi_\mathrm{r}(i), \pi_\mathrm{c}(k)} a_{\pi_\mathrm{r}(k), \pi_\mathrm{c}(j)}$
11: **end for**
12: **end for**
13: **end for**

Output: Es ist $PAQ = LU$ mit $P = P_{\pi_\mathrm{r}}$, $Q = P_{\pi_\mathrm{c}}^T$ und

$$L = \begin{pmatrix} 1 & & & \\ a_{\pi_\mathrm{r}(2), \pi_\mathrm{c}(1)} & 1 & & \\ \vdots & \ddots & \ddots & \\ a_{\pi_\mathrm{r}(n), \pi_\mathrm{c}(1)} & \cdots & a_{\pi_\mathrm{r}(n), \pi_\mathrm{c}(n-1)} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} a_{\pi_\mathrm{r}(1), \pi_\mathrm{c}(1)} & \cdots & a_{\pi_\mathrm{r}(1), \pi_\mathrm{c}(n)} \\ & \ddots & \vdots \\ & & a_{\pi_\mathrm{r}(n), \pi_\mathrm{c}(n)} \end{pmatrix}$$

Die Matrizen P, Q, L, U wird man in der Praxis nicht aufstellen, sondern direkt mit der überschriebenen Matrix A bzw. den Vektoren π_r und π_c arbeiten.

Die Komplexität des Gaußschen Eliminationsverfahrens **ohne** Pivotisierung berechnen wir durch eine Zählung der Operationen wie folgt. Die verschiedenen Schleifen führen zu der Operationssumme

$$\begin{aligned}\sum_{k=1}^{n-1}\sum_{i=k+1}^{n}\left(1+\sum_{l=k+1}^{n}1\right) &= \sum_{k=1}^{n-1}\big((n-k)+(n-k)^2\big)=\sum_{k=1}^{n-1}(k+k^2)\\ &= \frac{n(n-1)}{2}+\frac{1}{3}n^3+O(n^2)=\frac{1}{3}n^3+O(n^2).\end{aligned}$$

Im letzten Schritt benutzt man die Formel $\sum_{k=1}^{n-1}k^2=\frac{1}{6}n(n-1)(2n-1)=\frac{1}{3}n^3-\frac{1}{2}n^2+\frac{1}{6}n$. Die Komplexität des Gauß Algorithmus wächst also mit der 3. Potenz der Anzahl der Unbekannten. Zeilen- bzw. Spaltenpivotisierung erhöhen die benötigten Rechenoperationen mit $\mathcal{O}(n^2)$ und fallen dadurch nicht ins Gewicht. Totalpivotisierung dagegen benötigt $\mathcal{O}(n^3)$ Operationen und ist somit aufwendig.

6.1.6 Berechnung der Inversen

Mit Hilfe des Gaußschen Eliminationsverfahrens kann die Inverse einer Matrix A berechnet werden. Die Inverse einer Matrix kann geschrieben werden in der Form

$$A^{-1}=(a^{(1)},\ldots,a^{(n)})$$

mit Spaltenvektoren $a^{(j)}$, $j=1,\ldots,n$. Wegen $AA^{-1}=I$ erfüllen diese Vektoren das Gleichungssystem

$$Aa^{(j)}=e_j,\quad j=1,\ldots,n.$$

Mit Hilfe der LU-Zerlegung lässt sich diese Gleichung auflösen und somit A^{-1} berechnen.

In der Praxis ist es allerdings wesentlich effizienter (Faktor 4), zur Lösung einer Gleichung der Form $Ax=b$ unmittelbar eine LU-Zerlegung von A einzusetzen und nicht erst die Inverse A^{-1} zu berechnen und diese auf b anzuwenden.

6.2 Cholesky-Zerlegung

In vielen Anwendungen tauchen hermitesche und positiv definite Matrizen auf (siehe Def. 5.7 und Satz 5.9). In diesem Abschnitt besprechen wir eine spezielle LU-Zerlegung für diese wichtige Klasse von Matrizen.

Definition 6.21 (Cholesky Zerlegung)**.** *Unter einer* Cholesky-Zerlegung *einer hermiteschen Matrix A versteht man eine Faktorisierung*

$$A=LL^*,$$

wobei L eine untere Dreiecksmatrix ist.

*Unter einer LDL-*Zerlegung *versteht man eine Faktorisierung von A in der Form*

$$A = LDL^*,$$

wobei D eine Diagonalmatrix und L eine untere Dreiecksmatrix mit Einsen auf der Diagonalen ist.

Satz 6.22. *Sei $A \in \mathbb{K}^{n\times n}$ hermitesch und positiv definit. Dann ist das Gaußsche Elminitationsverfahren ohne Pivotisierung (z.B. Alg. 6.6 oder Alg. 6.20 ohne Permutationen) durchführbar, und die dabei auftretenden Diagonalelemente $a_{kk}^{(k)}$, die gleich den Diagonalelementen u_{kk} von U sind, sind positiv für $k = 1, \dots, n$.*

Beweis. Da A positiv definit ist, ist $a_{11} = (Ae_1, e_1)_2 > 0$ und somit der erste Gaußschritt durchführbar. Wegen

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} = \overline{a_{ji} - \frac{a_{j1}}{a_{11}} a_{1i}} = a_{ji}^{(1)}, \qquad i,j = k, \dots, n,$$

ist die im 1.Schritt erzeugte $(n-1)\times(n-1)$-Teilmatrix $\tilde{A}^{(1)} = \left(a_{ij}^{(1)}\right)_{i,j=2}^{n}$ ebenfalls hermitesch. Wenn wir zeigen können, dass sie positiv definit ist, folgt die Behauptung per Induktion.

Sei dazu $\tilde{x} = (x_2, \dots, x_n)^\top \in \mathbb{K}^{n-1} \setminus \{0\}$ beliebig und $x = (x_1, \tilde{x})^\top \in \mathbb{K}^n$ mit

$$x_1 = -\frac{1}{a_{11}} \sum_{\ell=2}^{n} a_{1\ell}\overline{x_\ell}.$$

Dann ist

$$\begin{aligned} 0 < \sum_{i,j=1}^{n} a_{ij} x_i \overline{x_j} &= \sum_{i,j=2}^{n} a_{ij} x_i \overline{x_j} + 2\mathrm{Re}\left\{ x_1 \sum_{\ell=2}^{n} a_{1\ell}\overline{x_\ell} \right\} + a_{11}|x_1|^2 \\ &\quad \underbrace{-\frac{1}{a_{11}} \sum_{i,j=2}^{n} a_{i1}a_{1j} x_i \overline{x_j} + \frac{1}{a_{11}} \left| \sum_{\ell=2}^{n} a_{1\ell}\overline{x_\ell} \right|^2}_{=0, \text{ da } \overline{a_{1\ell}} = a_{\ell 1}} \\ &= \sum_{i,j=2}^{n} \underbrace{\left(a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} \right)}_{=a_{ij}^{(1)}} x_i \overline{x_j} + a_{11} \underbrace{\left| x_1 + \frac{1}{a_{11}} \sum_{\ell=2}^{n} a_{1\ell}\overline{x_\ell} \right|^2}_{=0} \end{aligned}$$

und somit $(\tilde{A}^{(1)}\tilde{x}, \tilde{x})_2 > 0$ für beliebiges $\tilde{x} \in \mathbb{K}^{n-1} \setminus \{0\}$. □

Satz 6.23. *Sei $A \in \mathbb{K}^{n\times n}$ hermitesch und positiv definit. Dann existiert eine eindeutig bestimmte LDL-Zerlegung von A. Weiterhin existiert eine Cholesky-Zerlegung von A und die Diagonaleinträge von L können positiv gewählt werden. Mit dieser Nebenbedingung ist L eindeutig bestimmt.*

Beweis. Existenz einer LDL-Zerlegung: Nach dem vorigen Satz existiert eine normierte untere Dreiecksmatrix L und eine obere Dreiecksmatrix U mit $A = LU$. Setze $D = \operatorname{diag}(u_{11}, \ldots, u_{nn})$. Da $0 < \det A = \det L \cdot \det U = u_{11} \cdot \cdots \cdot u_{nn}$, ist D regulär. Mit $\tilde{U} = D^{-1}U$ gilt

$$A = LD\tilde{U}.$$

Da $\tilde{U}$ eine obere Dreiecksmatrix mit Einsen auf der Diagonalen ist, haben wir mit

$$A = A^* = \tilde{U}^*(D^*L^*)$$

eine weitere LU-Zerlegung von A, und aus der Eindeutigkeit (siehe Lemma 6.9) folgt $\tilde{U}^* = L$, also

$$A = LDL^*.$$

Eindeutigkeit der LDL-Zerlegung: Ist $A = L_1 D_1 L_1^*$ eine weitere LDL-Zerlegung von A, so folgt aus der Eindeutigkeit der LU-Zerlegung, dass $L_1 = L$ und $D_1 = D$ ist.

Existenz der Cholesky-Zerlegung: Sei $A = \tilde{L}D\tilde{L}^*$ die LDL-Zerlegung von A. Dann sind wegen des vorigen Satzes die Einträge von D positiv. Setze nun

$$L := \tilde{L} \begin{pmatrix} \sqrt{d_{11}} & & \\ & \ddots & \\ & & \sqrt{d_{nn}} \end{pmatrix}.$$

Dann ist $A = LL^*$, und die Diagonaleinträge von L sind positiv.

Eindeutigkeit der Cholesky-Zerlegung: Sei $A = L_1 L_1^*$ eine weitere Cholesky-Zerlegung mit positiven Diagonaleinträgen $\lambda_1, \ldots, \lambda_n > 0$. Setze

$$D_1 := \operatorname{diag}(\lambda_1^2, \ldots, \lambda_n^2)$$

und

$$\tilde{L}_1 = L_1 \operatorname{diag}\left(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}\right).$$

Dann ist

$$A = \tilde{L}_1 D_1 \tilde{L}_1^*,$$

und aus der Eindeutigkeit der LDL-Zerlegung folgt $\tilde{L}_1 = \tilde{L}$ und $\tilde{D}_1 = D$. Also gilt $L_1 = L$. □

Analog zu Alg. 6.10 kann die Gleichung $A = LL^*$ genutzt werden, um folgenden Algorithmus herzuleiten.

Komplexität: Es wird nur die untere Hälfte der Matrix A benötigt. Zur Berechnung einer Cholesky-Zerlegung werden neben n Quadratwurzel-Funktionen

$$\begin{aligned} \sum_{k=1}^{n} \sum_{i=k}^{n} (k-1) &= \sum_{k=1}^{n} (n-k+1)(k-1) = \sum_{k=1}^{n} (nk - k^2) + O(n^2) \\ &= \frac{1}{2}n^3 - \frac{1}{3}n^3 + O(n^2) = \frac{1}{6}n^3 + O(n^2) \end{aligned}$$

Algorithmus 6.24 Cholesky-Zerlegung

Input: $A \in \mathbb{K}^{n\times n}$ hermitesch und positiv definit

1: **for** $k = 1, \dots, n$ **do**
2: $\quad a_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} |a_{kj}|^2}$
3: $\quad$ **for** $i = k+1, \dots, n$ **do**
4: $\quad\quad a_{ik} = a_{kk}^{-1}\left(a_{ik} - \sum_{j=1}^{k-1} a_{ij}\overline{a_{kj}}\right)$
5: $\quad$ **end for**
6: **end for**

Output: $A = LL^*$ mit $L = \begin{pmatrix} a_{11} & & \\ \vdots & \ddots & \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$

flops benötigt. Der Aufwand für eine Cholesky-Zerlegung ist also in etwa halb so groß wie der für eine LU-Zerlegung nach dem Gaußschen Eliminationsverfahren.

6.3 Ausgleichsprobleme und QR-Zerlegung

6.3.1 QR-Zerlegung

Definition 6.25. *Sei $A \in \mathbb{K}^{m\times n}$. Unter einer* QR-Zerlegung von A *versteht man eine Faktorisierung*

$$A = QR,$$

wobei $Q \in \mathbb{K}^{m\times m}$ unitär ist und $R \in \mathbb{K}^{m\times n}$ von der Form

$$R = \left(\begin{array}{cccccc} r_{11} & r_{12} & \cdots & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & \cdots & \cdots & r_{2n} \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & 0 & r_{kk} & \cdots & r_{kn} \\ \hline 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{array}\right)$$

ist mit $r_{11}, \dots, r_{kk} \neq 0$ und $k \leq \min(m, n)$.

Bemerkung 6.26. *Obige Definition ist in der Literatur nicht eindeutig. Zum Teil wird für $A \in \mathbb{K}^{m\times n}$ mit $m \geq n$ auch eine Zerlegung der Form $A = QR$ mit $Q \in \mathbb{K}^{m\times n}$ und $R \in \mathbb{K}^{n\times n}$ als QR-Zerlegung bezeichnet. In diesem Fall ist R eine obere Dreiecksmatrix (z.T. mit der Zusatzforderung $r_{ii} > 0$ für $i = 1, \dots, n$) und für Q wird $Q^*Q = I$ gefordert.*

Die beiden Definitionen sind im gewissen Sinne äquivalent: Wenn $A = QR$ mit $Q = (\tilde{Q}, V) \in \mathbb{K}^{m\times m}$, $\tilde{Q} \in \mathbb{K}^{m\times n}$, $R = \left(\begin{smallmatrix} \tilde{R} \\ 0 \end{smallmatrix}\right) \in \mathbb{K}^{m\times n}$ und $\tilde{R} \in \mathbb{K}^{n\times n}$, dann ist $A = \tilde{Q}\tilde{R}$ eine QR-Zerlegung nach obiger Bemerkung.

Andererseits sei $A = \tilde{Q}\tilde{R}$ eine QR-Zerlegung mit $\tilde{Q} \in \mathbb{K}^{m\times n}$ und $\tilde{R} \in \mathbb{K}^{n\times n}$. Dann bilden die Spaltenvektoren q_i für $i = 1, \ldots, n$ von $\tilde{Q}$ ein Orthonormalsystem von $\mathbb{K}^m$ und können durch $v_1, \ldots, v_{m-n}$ zu einer Orthonormalbasis erweitert werden. Mit $V = (v_1, \ldots, v_{m-n})$ ist dann $A = QR$ mit $Q = (\tilde{Q}, V) \in \mathbb{K}^{m\times m}$ und $R = \binom{\tilde{R}}{0} \in \mathbb{K}^{m\times n}$ eine QR-Zerlegung von A.

6.3.2 Lösung linearer Ausgleichsprobleme mit Hilfe der QR-Zerlegung

Satz 6.27. *Sei $A \in \mathbb{K}^{m\times n}$ mit $m \geq n$, $\operatorname{rang}(A) = n$, $A = QR$ eine QR-Zerlegung von A und $b \in \mathbb{K}^m$. Weiter sei $R = \binom{\tilde{R}}{0}$ mit oberer Dreiecksmatrix $\tilde{R} \in \mathbb{K}^{n\times n}$ und $Q^*b = \binom{c}{d}$ mit $c \in \mathbb{K}^n$.*

Dann ist die eindeutige Lösung $x \in \mathbb{K}^n$ des linearen Ausgleichsproblems

$$\|Ax - b\|_2^2 = \min!$$

Lösung des linearen Gleichungssystems

$$\tilde{R}x = c. \tag{6.6}$$

Beweis. Wegen Satz 5.17 ist x genau dann Lösung des Ausgleichsproblems, wenn es die Normalengleichung $A^*Ax = A^*b$ löst. Wegen 5.16(1) und der Voraussetzung $\operatorname{rang}(A) = n$, d.h. $\mathcal{N}(A) = \{0\}$, ist die Normalengleichung und damit das Ausgleichsproblem eindeutig lösbar.

Da die euklidische Norm invariant unter unitären Transformationen ist (Satz 5.9), folgt die Behauptung aus

$$\|Ax - b\|_2^2 = \|Q^*Ax - Q^*b\|_2^2 = \left\| \begin{pmatrix} \tilde{R}x - c \\ -d \end{pmatrix} \right\|_2^2 = \|\tilde{R}x - c\|_2^2 + \|d\|_2^2.$$

□

(6.6) kann mit $\mathcal{O}(n^2)$ Rechenoperationen durch Rückwärtssubstitution gelöst werden, die Berechnung von c erfordert $\mathcal{O}(nm)$ Rechenoperationen.

6.3.3 Householder-Matrizen

Für die QR-Zerlegung nach Householder spielen sogenannte *Householder-Matrizen* eine ähnliche Rolle wie Gauß-Matrizen beim Gaußschen Eliminationsverfahren.

Definition 6.28. *Eine Matrix der Form*

$$H = I - \frac{2}{u^*u} uu^*$$

mit $u \in \mathbb{K}^n \setminus \{0\}$ heißt Householder-Matrix.

Lemma 6.29. *Sei $H = I - \frac{2}{u^*u}uu^*$ mit $u \in \mathbb{K}^n \setminus \{0\}$. Dann gilt*

1. *H ist hermitesch und unitär.*
2. *Falls $\mathbb{K} = \mathbb{R}$, ist die Abbildung $\mathbb{R}^n \to \mathbb{R}^n$, $x \mapsto Hx$ die Spiegelung an der Hyperebene $E := \{z \in \mathbb{R}^n : u^T z = 0\}$.*

Beweis. 1. Offensichtlich ist H hermitesch. Die folgende Rechnung zeigt, dass H auch unitär ist:

$$\left(I - \frac{2}{u^*u}uu^*\right)^2 = I - \frac{4}{u^*u}uu^* + \frac{4}{(u^*u)(u^*u)}u(u^*u)u^* = I.$$

2. Sei $x = \lambda u + z$ mit $\lambda \in \mathbb{R}$ und $z \in E$. Dann gilt $Hx = z + \lambda u - 2\lambda u = z - \lambda u$.

□

Wir möchten Householder-Matrizen ähnlich wie Gauß-Matrizen verwenden. Dazu müssen wir uns überlegen, ob es zu einem gegebenen $x \in \mathbb{K}^n \setminus \{0\}$ einen Vektor $u \in \mathbb{K}^n \setminus \{0\}$ gibt, so dass $Hx = \gamma e_1$ für eine Zahl $\gamma \in \mathbb{K}$, oder ausführlich

$$x - u\left(\frac{2u^*x}{u^*u}\right) = \gamma e_1.$$

Hinreichend hierfür ist

$$\frac{2u^*x}{u^*u} = 1 \qquad \text{und} \qquad u = x - \gamma e_1 \text{ für ein } \gamma \in \mathbb{K}.$$

Hieraus erhält man nach einfachen Umformungen

$$0 = u^*(2x - u) = (x - \gamma e_1)^*(x + \gamma e_1) = x^*x + 2i\mathrm{Im}\,(\overline{x_1}\gamma) - |\gamma|^2.$$

Für $x_1 \neq 0$ sind $\gamma = \pm\frac{x_1}{|x_1|}\|x\|_2$ mögliche Lösungen. Wir entscheiden uns für $\gamma = -\frac{x_1}{|x_1|}\|x\|_2$, denn dadurch wird garantiert, dass $u \neq 0$, falls x ein Vielfaches von e_1 ist. Außerdem sind wir mit dieser Wahl vor Rundnungsfehlern durch Subtraktion fast gleich großer Zahlen gefeit. Im Falle $x_1 = 0$ können wir $\gamma = -\|x\|_2$ verwenden.

Wir halten unser Ergebnis in folgendem Lemma fest:

Lemma 6.30. *Ist $x \in \mathbb{K}^n \setminus \{0\}$, $\gamma = x_1/|x_1|$ für $x_1 \neq 0$ und $\gamma = 1$ für $x_1 = 0$ und setzt man $u := x + \gamma\|x\|_2 e_1$ und $H = I - \frac{2}{u^*u}uu^*$, so gilt*

$$Hx = -\gamma\|x\|_2 e_1.$$

6.3.4 Berechnung der QR-Zerlegung nach Householder

Sei $A \in \mathbb{K}^{m\times n}$. Wir nehmen an, dass $\operatorname{rang}(A) = n$. Ähnlich wie bei der Gauß-Elimination bringen wir A Spalte für Spalte durch Multiplikation mit Householder-Matrizen auf obere Dreiecksgestalt. Angenommen, wir hätten nach $k \in \{0,\dots,n\}$ Schritten bereits unitäre Matrizen $H^{(1)},\dots,H^{(k)}$ so bestimmt, dass für

$$A^{(k)} := H^{(k)}\cdots H^{(1)}A$$

gilt $a_{ij}^{(k)} = 0$ für $j \le k$ und $i < j$. Mit anderen Worten sei $A^{(k)}$ eine Blockmatrix der Form

$$A^{(k)} = \begin{pmatrix} R^{(k)} & B^{(k)} \\ 0 & C^{(k)} \end{pmatrix} \tag{6.7}$$

mit einer oberen Dreiecksmatrix $R^k \in \mathbb{K}^{k\times k}$, $B^{(k)} \in \mathbb{K}^{k\times(n-k)}$ und $C^{(k)} \in \mathbb{K}^{(m-k)\times(n-k)}$. Falls $k = n$, sind wir fertig und haben mit

$$Q := H^{(1)}\cdots H^{(n)}, \qquad R := A^{(n)}$$

eine QR-Zerlegung von A berechnet, wobei wir die Eigenschaft $(H^{(j)})^{-1} = H^{(j)}$ der Householder-Matrizen benutzen. Anderenfalls verfahren wir wie folgt: Sei $\tilde{x}^{(k+1)} \in \mathbb{K}^{m-k}$ gegeben durch

$$\tilde{x}^{(k+1)} := (c_{1,1}^{(k)},\dots,c_{m-k,1}^{(k)})^T.$$

Wäre $\tilde{x}^{(k+1)} = 0$, wären die ersten $(k+1)$ Spalten von $A^{(k)}$ linear abhängig, also $\operatorname{rang}(A^{(k)}) < n$. Wegen der Unitarität der $H^{(j)}$ ist dies ein Widerspruch zur Voraussetzung $\operatorname{rang}(A) = n$. Gemäß Lemma 6.30 wählen wir

$$\tilde{u}^{(k+1)} := \tilde{x}^{(k+1)} + \gamma(\tilde{x}_1^{(k+1)})\|\tilde{x}^{(k+1)}\|e_1$$

und $\tilde{H}^{(k+1)} := I_{m-k} - \frac{2}{(\tilde{u}^{(k+1)})^*\tilde{u}^{(k+1)}}\tilde{u}^{(k+1)}(\tilde{u}^{(k+1)})^*$ so, dass die erste Spalte von $\tilde{H}^{(k+1)}C^{(k)}$ ein Vielfaches des ersten Einheitsvektors $e_1 \in \mathbb{R}^{m-k}$ ist. Wir setzen

$$H^{(k+1)} = \begin{pmatrix} I_k & 0 \\ 0 & \tilde{H}^{(k+1)} \end{pmatrix}.$$

Dies ist eine Householder-Matrix der Form

$$H^{(k+1)} := I_m - \frac{2}{(u^{(k+1)})^*u^{(k+1)}}u^{(k+1)}(u^{(k+1)})^*$$

mit $u^{(k+1)} := (\underbrace{0,\dots,0}_{k\,mal},(\tilde{u}^{(k+1)})^T)^T$. Insbesondere ist $H^{(k+1)}$ nach Lemma 6.29 unitär. Mit dieser Wahl von $H^{(k+1)}$ gilt

$$A^{(k+1)} := H^{(k+1)}A^{(k)} = \begin{pmatrix} R^{(k)} & B^{(k)} \\ 0 & \tilde{H}^{(k+1)}C^{(k)} \end{pmatrix} = \begin{pmatrix} R^{(k+1)} & B^{(k+1)} \\ 0 & C^{(k+1)} \end{pmatrix}$$

mit $R^{k+1} \in \mathbb{K}^{(k+1)\times(k+1)}$, $B^{(k+1)} \in \mathbb{K}^{(k+1)\times(n-k-1)}$ und $C^{(k+1)} \in \mathbb{K}^{(m-k-1)\times(n-k-1)}$. Per Induktion über k folgt damit die Existenz einer QR-Zerlegung von A.

Algorithmus 6.32 QR-Zerlegung nach Householder: Matrixversion

Input: $A \in \mathbb{K}^{m\times n}$ mit $\operatorname{rang}(A) = n$

1: $A^{(0)} = A$
2: **for** $k = 1, \dots, n$ **do**
3: $\quad d = a_{k,k}^{(k-1)} + \gamma(a_{k,k}^{(k-1)})\sqrt{\sum_{i=k}^{m} |a_{i,k}^{(k-1)}|^2}$
4: $\quad u^{(k)} = (\underbrace{0, \dots, 0}_{k-1\,mal}, d, a_{k+1,k}^{(k-1)}, \dots, a_{m,k}^{(k-1)})^T$
5: $\quad H^{(k)} = I_m - \frac{2}{(u^{(k)})^* u^{(k)}} u^{(k)} (u^{(k)})^*$
6: $\quad A^{(k)} = H^{(k)} A^{(k-1)}$
7: **end for**

Output: $A = QR$ mit $R := A^{(n)}$ und $Q := H^{(1)} \cdots H^{(n)}$

Satz 6.31. *Die Matrix $A \in \mathbb{K}^{m\times n}$ habe $\operatorname{rang}(A) = n$. Dann besitzt A eine QR-Zerlegung und diese kann mit Algorithmus 6.32 berechnet werden.*

Um diesen Algorithmus konkret zu implementieren müssen wir uns überlegen, wie wir die Anwendung einer Householder-Matrix auf einen Vektor implementieren. Sei $H = I - \frac{2}{u^*u} uu^*$ mit $u := x + \gamma(x_1)\|x\|_2 e_1$ wobei $u, x \in \mathbb{K}^m$. Dann gilt

$$H = I - \beta uu^* \qquad \text{mit } \beta = \frac{2}{u^*u} = \frac{1}{\|x\|_2(\|x\|_2 + |x_1|)}$$

und

$$Hv = v - su \qquad \text{mit } s = \beta u^* v$$

für $v \in \mathbb{K}^n$. Es ergibt sich der Matrix-freie Algorithmus 6.33.

Komplexität des Alg. 6.33: Für die Komplexität der QR-Zerlegung nach Householder im Fall $\operatorname{rang}(A) = n$ erhalten wir als Anzahl der benötigten flops

$$\begin{aligned}
\sum_{k=1}^{n} \sum_{j=k+1}^{n} 2(m-k) + \mathcal{O}(mn) &= \sum_{k=1}^{n} 2(n-k)(m-k) + \mathcal{O}(mn) \\
&= \sum_{k=1}^{n} \left(2mn - 2k(m+n) + 2k^2\right) + \mathcal{O}(mn) \\
&= 2mn^2 - n^2(m+n) + \frac{2}{3}n^3 + \mathcal{O}(mn) \\
&= n^2\left(m - \frac{1}{3}n\right) + \mathcal{O}(mn).
\end{aligned}$$

Im Spezialfall $m = n$ sind dies im wesentlichen $\frac{2}{3}n^3$ flops und somit etwa doppelt so viel wie bei der Gauß-Elimination. Die QR-Zerlegung kann zwar auch zum Lösen von Gleichungssystemen mit regulärer Matrix verwendet werden, ist aber gegenüber der Gauß-Elimination nicht konkurrenzfähig.

Algorithmus 6.33 QR-Zerlegung nach Householder

Input: $A \in \mathbb{K}^{m\times n}$ mit $\operatorname{rang}(A) = n$

$U = 0 \in \mathbb{K}^{m\times n}$
for $k = 1, \dots, n$ **do**
  for $i = k, \dots, m$ **do**
    $u_{ik} := a_{ik}$
  end for
  $\alpha = \sqrt{\sum_{i=k}^{m} |u_{ik}|^2}$
  $\beta = 1/(\alpha(\alpha + |u_{kk}|))$
  $u_{kk} = u_{kk} + \gamma(u_{kk})\alpha$
  $a_{kk} = -\gamma(a_{kk})\alpha$
  for $i = k+1, \dots, m$ **do**
    $a_{ik} = 0$
  end for
  for $j = k+1, \dots, n$ **do**
    $s = \beta \sum_{i=k}^{m} \overline{u_{ik}} a_{ij}$
    for $i = k+1, \dots, m$ **do**
      $a_{ij} = a_{ij} - s u_{ik}$
    end for
  end for
end for

Output: A ist mit der Matrix R überschrieben und $Q := H_1 \cdots H_n$ mit $H_k := I - \frac{2}{u_k^* u_k} u_k u_k^*$ und $u_k := (u_{ik})_{i=1,\dots,m}$ für $k = 1, \dots, n$, so dass $A = QR$

7 Nichtlineare Gleichungssysteme

In diesem Kapitel untersuchen wir Verfahren zur Lösung nichtlinearer Gleichungssysteme

$$f_j(x_1, \ldots, x_n) = 0, \qquad j = 1, \ldots, m$$

mit reellwertigen Funktionen $f_j : D \subset \mathbb{R}^n \to \mathbb{R}$, kurz

$$F(x) = 0$$

mit $x = (x_1, \ldots, x_n)^\top$ und $F(x) = (f_1(x), \ldots, f_m(x))^\top$. Offenbar können wir nichtlineare Gleichungssysteme durch Subtraktion immer so umformen, dass die rechte Seite 0 ist.

Im Gegensatz zu linearen Gleichungssystemen lassen sich nichtlineare Gleichungssysteme im allgemeinen nicht mehr durch eine Folge algebraischer Manipulationen exakt lösen. Wir werden in diesem Kapitel statt dessen sogenannte *iterative Verfahren* besprechen, bei denen eine gegebene Näherungslösung in jedem Iterationsschritt verbessert wird, bis eine geforderte Genauigkeit erreicht ist.

7.1 Skalare Probleme

Wir starten zur Vereinfachung mit skalaren Problemen, d.h. gesucht wird $x \in [a, b]$ mit $F(x) = 0$ und $F : [a, b] \to \mathbb{R}$.

7.1.1 Bisektionsverfahren

Das Bisektionsverfahren ist das einfachste Verfahren zur Nullstellenbestimmung einer reellwertigen stetigen Funktion und beruht auf dem Zwischenwertsatz: Ist $f : [a, b] \to \mathbb{R}$ stetig und $f(a)f(b) < 0$, so existiert mindestens eine Nullstelle von f in $[a, b]$. Damit kann eine Folge von Intervallen konstruiert werden, welche mindestens eine Nullstelle von f enthält (siehe Alg. 7.1).

Der Alg. 7.1 ist numerisch sehr stabil, aber auch recht langsam: In jedem Schritt wird das Intervall halbiert, d.h. nach t Schritten gilt $|b_t - a_t| = \left(\frac{1}{2}\right)^t |b - a|$. Einzige Konvergenzvoraussetzungen sind, dass f stetig ist und dass $f(a)f(b) < 0$ gilt, d.h. dass die Anzahl der Nullstellen im Ausgangsinvervall (gezählt nach ihrer Vielfachheit) ungerade ist. Das Verfahren ist aber auf skalare, reelle Funktionen beschränkt.

Algorithmus 7.1 Bisektionsverfahren

Input: $f : [a, b] \to \mathbb{R}$ stetig mit $f(a)f(b) < 0$, **TOL** vorgegebene Toleranz

```
while b − a > TOL do
    t = (a+b)/2
    if f(t) = 0 then
        a = t , b = t
    else
        if f(a)f(t) < 0 then
            b = t
        else
            a = t
        end if
    end if
end while
```

Output: Intervall $[a, b]$ mit $b - a <$ **TOL**, welches eine Nullstelle von f enthält

Bemerkung 7.2. *Beim Bisektionsverfahren ist es, wie bei den meisten Verfahren zur Nullstellenbestimmung, sehr schwierig zu kontrollieren, gegen welche Nullstelle das Verfahren konvergiert. Nullstellen mit gerader Vielfachheit werden, wenn nicht durch Zufall, nie gefunden.*

7.1.2 Newton-Verfahren

Ist $f : [a, b] \to \mathbb{R}$ stetig differenzierbar, so bietet sich das Newton-Verfahren zur Nullstellensuche an. In Abb. 7.1 ist das Newton-Verfahren geometrisch veranschaulicht: Ausgehend von einem Startwert x_0 wird die Tangente

$$T(x) = f'(x_0)(x - x_0) + f(x_0)$$

an den Graphen der Funktion f im Punkt $(x, f(x))$ berechnet und im Falle $f'(x_0) \neq 0$ die Nullstelle der Tangente als neue Näherung x_1 der gesuchten Nullstelle verwendet:

$$T(x_1) = 0 \qquad \Rightarrow \qquad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Wiederholt man dieses Vorgehen, so erhält man das *Newton-Verfahren.*

Satz 7.3 (Newton-Verfahren in $\mathbb{R}$)**.** *Die Funktion $f \in C^2([a, b])$ habe im Inneren des Invervalls $[a, b]$ eine Nullstelle z, und es gelte*

$$m := \min_{x \in [a,b]} |f'(x)| > 0.$$

Weiter sei $M := \max_{x \in [a,b]} |f''(x)|$, $K_\rho(z) := \{x \in \mathbb{R} \mid |x - z| \leq \rho\}$ und $\rho > 0$ so gewählt, dass

$$q := \frac{M}{2m}\rho < 1 \qquad \text{und} \quad K_\rho(z) \subset [a, b].$$

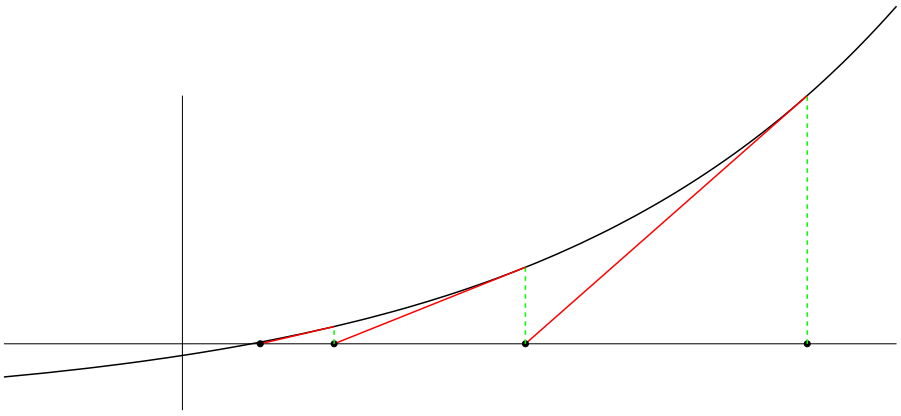


Abbildung 7.1: Geometrische Interpretation des Newton-Verfahrens

Dann sind für jeden Startpunkt $x_0 \in K_\rho(z)$ die Newton-Iterierten

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}, \qquad t = 0, 1, \ldots, \tag{7.1}$$

wohldefiniert und liegen in $K_\rho(z)$. Weiterhin konvergieren sie gegen die eindeutige Nullstelle z und es gelten die a priori Fehlerabschätzung

$$|x_t - z| \leq \frac{2m}{M} q^{2^t}, \qquad t \in \mathbb{N}, \tag{7.2}$$

und die a posteriori Fehlerabschätzung

$$|x_t - z| \leq \frac{M}{2m} |x_t - x_{t-1}|^2, \qquad t \in \mathbb{N}. \tag{7.3}$$

Beweis. Aus dem Mittelwertsatz der Differentialrechnung folgt für $x, y \in [a, b]$ mit $x \neq y$

$$|x - y| = \frac{|f(x) - f(y)|}{|f'(\xi)|} \leq \frac{1}{m} |f(x) - f(y)|$$

und daraus die Eindeutigkeit der nach Voraussetzung existierenden Nullstelle von f. Weiter erhalten wir mit der Formel von Taylor und der Restglieddarstellung nach Lagrange ein $\xi \in (\min\{x, y\}, \max\{x, y\})$ mit

$$f(y) = f(x) + f'(x)(y - x) + R(y; x), \quad R(y; x) := \frac{f''(\xi)}{2}(y - x)^2.$$

Sei nun

$$\phi(x) := x - \frac{f(x)}{f'(x)}, \qquad x \in K_\rho(z). \tag{7.4}$$

Wegen

$$\phi(x) - z = x - \frac{f(x)}{f'(x)} - z + \underbrace{f(z)}_{=0} = \frac{f(z) - f(x) - f'(x)(z-x)}{f'(x)} = \frac{R(z;x)}{f'(x)}$$

folgt

$$|\phi(x) - z| \le \frac{M}{2m}|x-z|^2 \le \frac{M}{2m}\rho^2 < \rho, \qquad x \in K_\rho(z)$$

und daraus $x_t = \phi(x_{t-1}) \in K_\rho(z)$ für $t \in \mathbb{N}$ wenn nur $x_0 \in K_\rho(z)$. Weiter folgt die a-priori Fehlerabschätzung und damit die Konvergenz $x_t \to z$ für $t \to \infty$ wegen

$$|x_t - z| \le \frac{M}{2m}|x_{t-1} - z|^2 \le \cdots \le \frac{2m}{M}\left(\frac{M}{2m}\underbrace{|x_0 - z|}_{\le \rho}\right)^{2^t} \le \frac{2m}{M}q^{2^t}, \quad t \in \mathbb{N}.$$

Für die a-posteriori Fehlerabschätzung verwenden wir obige Taylor-Formel für $y = x_t$ und $x = x_{t-1}$ und erhalten

$$f(x_t) = \underbrace{f(x_{t-1}) + (x_t - x_{t-1})f'(x_{t-1})}_{=0} + R(x_t; x_{t-1})$$

und daraus

$$|x_t - z| \le \frac{1}{m}|f(x_t) - \underbrace{f(z)}_{=0}| = \frac{|R(x_t; x_{t-1})|}{m} \le \frac{M}{2m}|x_t - x_{t-1}|^2.$$

□

Die Voraussetzungen des letzten Satzes schließen mehrfache Nullstellen aus. Betrachten wir dazu $f \in C^{p+1}([a,b])$ mit

$$f(z) = \cdots = f^{(p-1)}(z) = 0, \qquad f^{(p)}(z) \ne 0$$

für die p-fache Nullstelle $z \in [a,b]$. Wir erhalten mit der Formel von Taylor für x in einer Umgebung von z

$$f(x) = \sum_{\ell=0}^{p-1} \underbrace{f^{(\ell)}(z)}_{=0} \frac{(x-z)^\ell}{(\ell-1)!} + (x-z)^p R(z;x).$$

Wenn $f'(x) \ne 0$, so folgt

$$\frac{f(x)}{f'(x)} = \frac{1}{p}(x-z) - \frac{1}{p}(x-z)^2 \frac{\partial_x R(z;x)}{(x-z)\partial_x R(z;x) + pR(z;x)}.$$

Wenn an Stelle von (7.1) die modifizierte Iterationsvorschrift

$$x_{t+1} := x_t - p\frac{f(x_t)}{f'(x_t)}, \qquad t = 0, 1, \ldots, \tag{7.5}$$

verwendet wird, folgt wie im vorigen Satz quadratische Konvergenz aus

$$x_{t+1} - z = x_t - z - p\frac{f(x_t)}{f'(x_t)} = (x_t - z)^2 \frac{\partial_x R(z;x_t)}{(x_t - z)\partial_x R(z;x_t) + pR(z;x_t)}$$

sofern der Quotient $\frac{\partial_x R(z;x_t)}{(x_t-z)\partial_x R(z;x_t)+pR(z;x_t)}$ für x_t in einer Umgebung von z beschränkt bleibt. Dies wird durch $f^{(p)}(z) \ne 0$ garantiert.

7.1.3 Fixpunktiteration

Wir nehmen in diesem Abschnitt an, dass das Gleichungssystem in Form einer Fixpunktgleichung

$$x = \phi(x)$$

vorliegt. Eine Lösung dieser Gleichung heißt *Fixpunkt* von ϕ. Wir betrachten die Iterationsvorschrift

$$x^{(k+1)} = \phi(x^{(k)}), \qquad k = 0, 1, 2, \ldots$$

und hoffen, dass die so konstruierte Folge gegen eine Lösung der Fixpunktgleichung konvergiert.

Beispiel 7.4 (Vereinfachtes Newton-Verfahren)**.** *Das Newton-Verfahren* (7.1) *entspricht einer Fixpunktiteration mit*

$$\phi(x) := x - \frac{f(x)}{f'(x)}.$$

In jedem Iterationsschritt muss f' an der Stelle x_t ausgewertet werden, was bei komplizierten Funktionen f unter Umständen einen sehr großen Aufwand bedeuten kann. So könnte f z.B. nur implizit über die Lösung einer partiellen Differentialgleichung definiert sein, sodass eine Auswertung von f und f' das Lösen eines sehr großen Problems erfordert. Beim vereinfachten Newton-Verfahren wird deshalb zumindest für einige Schritte ein $c \in [a, b]$ festgehalten und eine Fixpunktiteration mit

$$\phi(x) := x - \frac{f(x)}{f'(c)}$$

verwendet.

Definition 7.5 (Ordnung Iterationsverfahren)**.** *Sei $(x_t)_{t\in\mathbb{N}}$ mit $x_t \in [a, b]$ für $t \in \mathbb{N}$ eine Folge, die gegen ein $z \in [a, b]$ konvergiert. Falls*

$$|x_{t+1} - z| \leq q|x_t - z|^p, \quad t \in \mathbb{N}, \qquad p > 1, q > 0, \tag{7.6}$$

so spricht man von Konvergenz der Ordnung p*. Im Falle $p = 1$, d.h.* linearer Konvergenz*, fordert man $q < 1$. Gilt die Abschätzung*

$$|x_{t+1} - z| \leq q_{t+1}|x_t - z|, \quad t \in \mathbb{N}, \quad \text{mit } \lim_{t\to\infty} q_t = 0,$$

so spricht man von superlinearer Konvergenz*.*

Satz 7.6. *Sei $\phi : [a, b] \to \mathbb{R}$ in einer Umgebung $K_\rho(z) := \{x \in \mathbb{R} \mid |x - z| \leq \rho\}$ ihres Fixpunktes z p-mal stetig differenzierbar mit $p \geq 2$. Genau dann hat die Fixpunktiteration $x_{t+1} = \phi(x_t)$ mit $x_0 \in K_\rho(z)$ und hinreichend kleinem ρ die maximale Ordnung p, wenn*

$$\phi'(z) = \cdots = \phi^{(p-1)}(z) = 0, \qquad \phi^{(p)}(z) \neq 0.$$

Beweis. Sei $\phi'(z) = \cdots = \phi^{(p-1)}(z) = 0$. Dann folgt aus der Taylor-Formel um z

$$|x_{t+1} - z| = |\phi(x_t) - \phi(z)| \leq \frac{1}{p!} \max_{x \in K_\rho(z)} |\phi^{(p)}(x)| \, |x_t - z|^p$$

und daraus die Konvergenz $x_t \to z$ für $t \to \infty$ mit (mindestens) der Ordnung p, falls $\frac{\rho^{p-1}}{p!} \max_{x \in K_\rho(z)} |\phi^{(p)}(x)| < 1$.

Sei umgekehrt die Iteration von p-ter Ordnung, d.h. $|x_{t+1} - z| \leq q|x_t - z|^p$ und $x_t \to z$ für $t \to \infty$. Wir nehmen an, es gäbe ein $m \leq p-1$ mit $\phi^{(m)}(z) \neq 0$ aber $\phi^{(\ell)}(z) = 0$ für $\ell = 0, \ldots, m-1$. Dann gilt wie oben

$$|x_{t+1} - z| = \frac{1}{m!} \left|\phi^{(m)}(\xi_t)\right| |x_t - z|^m, \qquad \xi_t \in (\min\{z, x_t\}, \max\{z, x_t\}).$$

Damit folgt im Widerspruch zur Annahme

$$\left|\phi^{(m)}(z)\right| = \lim_{t \to \infty} \left|\phi^{(m)}(\xi_t)\right| \leq qm! \lim_{t \to \infty} |x_t - z|^{p-m} = 0.$$

□

Wie bereits im Satz 7.3 gesehen, ist das Newton-Verfahren zur Bestimmung einer einfachen Nullstelle von der Ordnung 2, da

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

und somit $\phi'(z) = 0$ falls $f(z) = 0$.

Lineare Konvergenz ist die in der Praxis am häufigsten auftretende Konvergenzordnung. In diesem Fall hängt die Effizienz wesentlich von der Größe der Konstante q ab: Ist $q \approx 1$, ist die Konvergenz sehr langsam. Beim Bisektionsverfahren in Abschnitt 7.1.1 haben wir lineare Konvergenz mit $q = \frac{1}{2}$ gefunden. Bei Konvergenz der Ordnung $p \geq 2$ ver-p-facht sich in jedem Schritt die Anzahl der korrekten Stellen. Hier genügen in der Regel nur wenige Schritte.

7.2 Banachscher Fixpunktsatz

Wir wollen die Konvergenzaussagen aus dem vorigen Abschnitt nun in zweierlei Hinsicht entscheidend verbessern:

1. Wir wollen das Ergebnis auf mehrdimensionale Fixpunkt-Iterationen verallgemeinern. Wir werden sogar gleich beliebige, möglicherweise unendlich-dimensionale Banachräume betrachten. Dies führt zu keinerlei zusätzlichen Komplikationen.

2. Wir werden die Existenz eines Fixpunktes nicht voraussetzen, sondern beweisen.

7.2.1 Konvergenzbeweis

Da uns der Zwischenwertsatz in mehreren Raumdimensionen nicht mehr zur Verfügung steht, ist die Existenz eines Fixpunktes meist nicht mehr so einfach zu beweisen. Der folgende Satz liefert dazu ein wichtiges Hilfsmittel. Da er sogar in Banachräumen gilt, kann er auch verwendet werden, um die Existenz einer Lösung von Fixpunkt-Gleichungen zu zeigen, bei denen die Unbekannte eine Funktion ist, etwa Differential- und Integralgleichungen. In der Tat ist dies ein klassisches Anwendungsgebiet des Banachschen Fixpunktsatzes. In dieser Vorlesung werden wir allerdings nur endlichdimensionale Anwendungen des Banachschen Fixpunktsatzes diskutieren.

Zunächst benötigen wir folgende Definition:

Definition 7.7. *Sei $U \subset X$ eine nichtleere Teilmenge eines Banachraums X. Eine Abbildung $\phi : U \to X$ heißt* kontrahierend, *falls es einen* Kontraktionsfaktor $q < 1$ *gibt, so dass*

$$\|\phi(x) - \phi(y)\| \leq q\|x - y\| \qquad \text{für alle } x, y \in U.$$

Satz 7.8 (Banachscher Fixpunktsatz)**.** *Sei $U \subset X$ eine abgeschlossene, nicht-leere Teilmenge eines Banachraums X und $\phi : U \to U$ eine kontrahierende Abbildung mit Kontraktionsfaktor q. Dann besitzt ϕ einen eindeutig bestimmten Fixpunkt $x^* \in U$, und die Iterationsfolge $x^{(t+1)} := \phi(x^{(t)})$, $t = 0, 1, \ldots$, konvergiert für jeden Startwert $x^{(0)} \in U$ gegen x^*. Weiterhin gelten für $t = 1, 2, \ldots$ die* a-priori Fehlerschranke

$$\|x^{(t)} - x^*\| \leq \frac{q^t}{1-q}\|x^{(1)} - x^{(0)}\|$$

sowie die a-posteriori Fehlerschranke

$$\|x^{(t)} - x^*\| \leq \frac{q}{1-q}\|x^{(t)} - x^{(t-1)}\|.$$

Beweis. Der wesentliche Schritt des Beweises ist der Nachweis, dass $(x^{(t)})_{t\in\mathbb{N}}$ eine Cauchy-Folge ist. Da X als Banachraum vollständig ist, besitzt dann $(x^{(t)})$ als Cauchy-Folge ein Grenzelement $x^* \in X$, und da U abgeschlossen ist, gilt $x^* \in U$. x^* ist Fixpunkt von ϕ, da

$$\|\phi(x^*) - \phi(x^{(t)})\| \leq q\|x^* - x^{(t)}\| \to 0$$

für $t \to \infty$ und deshalb $\phi(x^*) = \lim_{t\to\infty} \phi(x^{(t)}) = \lim_{t\to\infty} x^{(t+1)} = x^*$.

Um zu zeigen, dass $(x^{(t)})$ Cauchy-Folge ist, beobachten wir zunächst, dass

$$\begin{aligned} \|x^{(t)} - x^{(t-1)}\| &= \|\phi(x^{(t-1)}) - \phi(x^{(t-2)})\| \leq q\|x^{(t-1)} - x^{(t-2)}\| \\ &\leq \ldots \leq q^j\|x^{(t-j)} - x^{(t-j-1)}\| \end{aligned}$$

für $j \leq t-1$. Deshalb gilt nach der Dreiecksungleichung für $l > t \geq 1$

$$\begin{aligned}\|x^{(l)} - x^{(t)}\| &\leq \|x^{(l)} - x^{(l-1)}\| + \|x^{(l-1)} - x^{(l-2)}\| + \cdots + \|x^{(t+1)} - x^{(t)}\| \\ &\leq \sum_{j=1}^{l-t} q^j \|x^{(t)} - x^{(t-1)}\| \leq q \sum_{j=0}^{\infty} q^j \|x^{(t)} - x^{(t-1)}\| \\ &= \frac{q}{1-q}\|x^{(t)} - x^{(t-1)}\| \\ &\leq \frac{q^t}{1-q}\|x^{(1)} - x^{(0)}\|.\end{aligned}$$

Die rechte Seite dieser Ungleichung strebt offenbar gegen 0 für $t \to \infty$. Deshalb ist $(x^{(t)})$ Cauchy-Folge.

Betrachten wir in der letzten Ungleichung den Grenzübergang $l \to \infty$, so erhalten wir die a-priori (bzw. die a-posteriori) Fehlerschranke aus der letzten (bzw. der vorletzten) Zeile.

Schließlich bleibt noch die Eindeutigkeit von x^* zu zeigen. Sei $x^\#$ ein weiterer Fixpunkt von ϕ. Dann gilt aufgrund der Kontraktionseigenschaft von ϕ

$$\|x^* - x^\#\| = \|\phi(x^*) - \phi(x^\#)\| \leq q\|x^* - x^\#\|.$$

Da $q < 1$, folgt $\|x^* - x^\#\| = 0$, also $x^* = x^\#$. □

Wir fassen die wichtigsten Voraussetzungen des Banachschen Fixpunktsatzes noch einmal zusammen:

1. ϕ ist kontrahierend.
2. U ist abgeschlossen. (Dies wird für den Schluss benötigt, dass die Cauchy-Folge $(x^{(t)})$ ein Grenzelement in U besitzt!)
3. ϕ bildet U in sich ab. (Dies wird benötigt, um die Wohldefiniertheit der Fixpunkt-Iteration sicherzustellen!)

Zum Nachweis der Kontraktionseigenschaft wird meist das folgende Lemma benutzt:

Lemma 7.9. *Sei $q < 1$, $U \subset \mathbb{R}^n$ konvex[1], sei $\|\cdot\|$ eine Norm auf $\mathbb{R}^n$ und $\phi : U \to \mathbb{R}^n$ eine stetig differenzierbare Abbildung mit $\|D\phi(x)\| \leq q$ für alle $x \in U$. Dabei verwenden wir die der Vektornorm $\|\cdot\|$ zugeordnete Matrixnorm. Dann ist ϕ kontrahierend mit Kontraktionsfaktor q.*

Beweis. Sei $x, y \in U$. Wir wenden den Hauptsatz der Differential- und Integralrech-

[1] Zur Erinnerung: Eine Teilmenge $D \subset \mathbb{R}^n$ heißt *konvex*, falls für alle $x, y \in D$ und alle $t \in [0,1]$ auch $(1-t)x + ty \in D$.

nung auf die Funktion $f(t) := \phi(x + t(y - x))$, $t \in [0, 1]$ an und erhalten

$$\begin{aligned} \|\phi(y) - \phi(x)\| &= \|f(1) - f(0)\| = \left\| \int_0^1 f'(t)dt \right\| \\ &= \left\| \int_0^1 D\phi(x + t(y - x))(y - x)dt \right\| \\ &\leq \int_0^1 \|D\phi(x + t(y - x))\|dt \; \|y - x\| \leq q\|y - x\|. \end{aligned}$$

□

7.2.2 Ein heuristisches Abbruchkriterium

Wir überlegen uns nun, wann wir eine Fixpunktiteration abbrechen müssen, um eine geforderte Genauigkeit $\epsilon > 0$ zu erreichen. Im Prinzip liefert die a-posteriori Fehlerschranke des Banachschen Fixpunktsatzes,

$$\|x^{(t)} - x^*\| \leq \frac{q}{1 - q}\|x^{(t)} - x^{(t-1)}\|.$$

ein geeignetes Abbruchkriterium: Breche die Iteration ab, sobald die rechte Seite dieser Ungleichung $\leq \epsilon$ ist. Die rechte Seite ist im Prinzip auch berechenbar. Allerdings ist es in der Praxis oft sehr schwierig und aufwendig, den Kontraktionsfaktor q analytisch zu berechnen oder abzuschätzen. Deshalb ersetzt man oft den exakten Kontraktionsfaktor q durch die Schätzung

$$\hat{q}_t := \frac{\|x^{(t)} - x^{(t-1)}\|}{\|x^{(t-1)} - x^{(t-2)}\|}.$$

$\hat{q}_t$ ist eine untere Schranke für q, denn

$$\hat{q}_t = \frac{\|\phi(x^{(t-1)}) - \phi(x^{(t-2)})\|}{\|x^{(t-1)} - x^{(t-2)}\|} \leq q.$$

Wir betonen, dass durch diesen Algorithmus die Fehlerschranke $\left\|x^{(k)} - x^*\right\| \leq \epsilon$ nicht garantiert werden kann, da $\frac{q_k}{1-q_k} \leq \frac{q}{1-q}$. Typischerweise konvergiert q_k gegen q, d.h. das Abbruchkriterium funktioniert gut, solange hinreichend oft iteriert wird. Um ein höheres Maß an Sicherheit zu bekommen, kann es sinnvoll sein, im Abbruchkriterium ϵ durch einen kleineren Wert, etwa $\epsilon/2$ zu ersetzen.

7.3 Newton-Verfahren in $\mathbb{R}^n$

Wir betrachten nun das Newton-Verfahren zur Lösung nichtlinearer Gleichungssysteme $F(x) = 0$ mit $F : U \subset \mathbb{R}^n \to \mathbb{R}^n$, welches gegeben ist durch

$$x^{(t+1)} = x^{(t)} - DF(x^{(t)})^{-1}F(x^{(t)}), \qquad t = 0, 1, \ldots, \tag{7.7}$$

mit Startwert $x^{(0)} \in U$.

Satz 7.10 (Newton-Mysovskikh)**.** *Gegeben seien eine offene, konvexe Teilmenge $U \subset \mathbb{R}^n$, eine Norm $\|\cdot\|$ auf $\mathbb{R}^n$, eine stetig differenzierbare Funktion $F : U \to \mathbb{R}^n$ und ein Startwert $x^{(0)} \in U$. Wir nehmen an:*

1. *Es existiert eine Nullstelle $x^* \in U$ der Funktion F.*
2. *$DF(x)$ ist regulär für alle $x \in U$.*
3. *Es gibt eine Konstante $\omega > 0$, so dass*
$$\|DF(x)^{-1}\,(DF(y) - DF(x))\,\| \le \omega\|x - y\| \tag{7.8}$$
für alle $x, y \in U$.
4. *Der Abstand $\rho := \|x^* - x^{(0)}\|$ genügt der Abschätzung*
$$\frac{\omega\rho}{2} < 1,$$
und die Kugel $B_\rho(x^*) := \{x \in \mathbb{R}^n : \|x - x^*\| < \rho\}$ mit Radius ρ um die Nullstelle x^* ist im Definitionsbereich U enthalten.

Dann bleibt die durch das Newton-Verfahren definierte Folge $\{x^{(t)} : t \in \mathbb{N}\}$ in $B_\rho(x^*)$ und konvergiert gegen x^*. Für $t = 0, 1, 2, \ldots$ gelten die Fehlerabschätzungen

$$\left\|x^{(t+1)} - x^*\right\| \le \frac{\omega}{2}\left\|x^{(t)} - x^*\right\|^2, \tag{7.9}$$

$$\left\|x^{(t)} - x^*\right\| \le \left(\frac{\omega\rho}{2}\right)^{2^t - 1}\rho. \tag{7.10}$$

Beweis. Der Beweis ist dem von Satz 7.3 sehr ähnlich. Wir verzichten deshalb auf den Beweis und verweisen z.B. auf Peter Deuflhard, *Newton Methods for Nonlinear Problems*, Springer, 2004 bzw. die Referenzen darin. Dort findet sich auch eine gute Übersicht über weitere Konvergenzaussagen zum Newton-Verfahren mit zum Teil unterschiedlichen Voraussetzungen. □

Korollar 7.11. *Sei $\mathcal{D}(F) \subset \mathbb{R}^n$ offen, $x^* \in \mathcal{D}(F)$, und sei $F : \mathcal{D}(F) \to \mathbb{R}^n$ eine zweimal stetig differenzierbare Funktion mit $F(x^*) = 0$ und $\det DF(x^*) \neq 0$. Dann existiert ein $\rho > 0$, so dass das Newton-Verfahren für alle Startwerte $x^{(0)} \in B_\rho(x^*)$ quadratisch konvergiert.*

Beweis. Wegen Lemma 5.13 und der Voraussetzung $\det DF(x^*) \neq 0$ existiert eine kompakte Umgebung $U := \{x \in \mathbb{R}^n : \|x - x^*\| \le r\} \subset \mathcal{D}(F)$ und ein $\beta > 0$, sodass $DF(x)$ regulär ist für $x \in U$ und die Inversen gleichmäßig durch β beschränkt sind, d.h.

$$\|DF(x)^{-1}\| \le \beta \qquad \text{für alle } x \in U.$$

Da DF sogar differenzierbar in x ist, gibt es eine Lipschitz-Konstante $L > 0$, so dass

$$\|DF(x) - DF(y)\| \le L\|x - y\| \qquad \text{für alle } x, y \in U.$$

Daraus erhalten wir die dritte Voraussetzung des Satzes mit $\omega := \beta L$. Schließlich ist auch die vierte Voraussetzung erfüllt, wenn wir $\rho < \min(r, 2/\omega)$ wählen. □

Selbstverständlich braucht und sollte man bei der Implementierung des Newton-Verfahrens die inversen Matrizen $DF(x^{(t)})^{-1}$ nicht ausrechnen, sondern man muss lediglich die Gleichungssysteme

$$DF(x^{(t)})\Delta x^{(t)} = -F(x^{(t)})$$

lösen, etwa mit Hilfe des Gaußschen Eliminationsverfahrens. Bei der Wahl des Abbruchkriteriums gehen wir ähnlich vor wie bei Fixpunktiterationen. Aufgrund der quadratischen Konvergenz des Newton-Verfahrens werden wir den Fehler dabei im allgemeinen zu pessimistisch schätzen.

Algorithmus 7.12 Newton-Verfahren

Input: $F : U \to \mathbb{R}^n$ differenzierbar mit Jacobi-Matrix $DF : U \to \mathbb{R}^{n\times n}$, Startvektor $x^{(0)} \in U$ und Toleranz **TOL**

Löse LGS $DF(x^{(0)})\Delta x = -F(x^{(0)})$
$x^{(1)} = x^{(0)} + \Delta x^{(0)}$
$t = 0$
repeat
    $t = t + 1$
    Löse LGS $DF(x^{(t)})\Delta x^{(t)} = -F(x^{(t)})$
    $x^{(t+1)} = x^{(t)} + \Delta x^{(t)}$
    $q_t = \|\Delta x^{(t)}\|/\|\Delta x^{(t-1)}\|$
    if $q_t \geq 1$ or $x^{(t+1)} \notin U$ **then**
        Stop: Newton-Verfahren konvergiert nicht.
    end if
until $\frac{q_t}{1-q_t}\|\Delta x^{(t)}\| \leq$ **TOL**

Output:Näherung $x^{(t+1)}$ an die Nullstelle $x^* \in U$ von F

Im allgemeinen besteht der Hauptaufwand bei der Durchführung des Newton-Verfahrens im Aufstellen und Invertieren der Jacobi-Matrizen $DF(x^{(t)})$. Im *vereinfachten Newton-Verfahren* (engl. *frozen Newton method*) wird nur einmal die Jacobi-Matrix $DF(x^{(0)})$ aufgestellt und eine LU-Zerlegung dieser Matrix berechnet. Die restlichen Newton-Schritte können dann mit Hilfe von Vorwärts- und Rückwärtssubstitution sehr effizient durchgeführt werden. Die Konvergenz dieses Verfahrens ist allerdings nur linear, wie man mit Hilfe des Banachschen Fixpunktsatzes zeigen kann.

Das Newton-Verfahren konvergiert nicht global, sondern nur lokal, d.h. für Startwerte, die hinreichend nahe bei einer Nullstelle der Funktion liegen. In vielen Fällen kann man den Konvergenzbereich des Newton-Verfahrens erheblich vergrößern, indem man am Anfang der Newton-Iteration eine Schrittweitensteuerung einführt:

$$x^{(t+1)} := x^{(t)} + \lambda_t \Delta x^{(t)}, \qquad \lambda_k \in (0, 1].$$

Dieses Verfahren heißt *gedämpftes Newton-Verfahren*. Interessant ist das gedämpfte Newton-Verfahren vor allem für mehrdimensionale Probleme. Zur Bestimmung von λ_t wird im t-ten Newton-Schritt ein eindimensionales Minimierungsproblem

$$\|F(x^{(t)} + \lambda\Delta x^{(t)})\| = \min! \qquad \lambda > 0$$

näherungsweise gelöst. Verfahren zur Lösung solcher eindimensionaler Minimierungsprobleme werden *Liniensuchmethoden* genannt. Ein mögliches solches Verfahren besteht etwa darin, den Wert $\lambda_t \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots, \lambda_{\min}\}$ zu wählen, für den das betrachtete Funktional minimal wird.

8 Lineare Gleichungssysteme (iterative Verfahren)

Wir betrachten in diesem Kapitel noch einmal das Problem, ein lineares Gleichungssystem

$$Ax = b$$

mit einer regulären Matrix $A \in \mathbb{K}^{n\times n}$ zu lösen. Der Gauß-Algorithmus liefert bis auf Rundungsfehler die exakte Lösung mit $n^3/3 + \mathcal{O}(n^2)$ Rechenopererationen (flops). Für sehr große Gleichungssysteme ist dieser Rechenaufwand oft nicht mehr vertretbar. Zudem ist A häufig *dünn besetzt*, d.h. pro Zeile sind z.B. nur $k \ll n$ Einträge von 0 verschieden. Der Speicherbedarf der Matrix A kann dann erheblich reduziert werden, indem nur die von 0 verschiedenen Einträge und ihre Position in der Matrix abgespeichert werden. Eine LU-Zerlegung einer dünn besetzten Matrix ist aber i.A. voll besetzt, sodass der Speicherbedarf der LU-Zerlegung sehr viel größer als der von A ist.

Deshalb werden sehr große Gleichungssysteme im allgemeinen mit Hilfe von iterativen Verfahren gelöst. In den meisten Fällen muss in jedem Iterationsschritt im wesentlichen ein Matrix-Vektor Produkt berechnet werden. Für eine voll besetzte Matrix A erfordert dies n^2 flops. Falls die Anzahl N der Iterationen kleiner als $n/3$ ist, haben wir also bereits einen Effizienzgewinn erreicht! Falls A nur k nicht verschwindende Matrixeinträge pro Zeile hat, ist die Multiplikation sogar mit Aufwand $\mathcal{O}(kn)$ durchführbar.

8.1 Fixpunktiterationen

Wir betrachten in diesem Abschnitt Fixpunktiterationen für Gleichungen

$$x = \phi(x)$$

mit einer affin-linearen Abbildung $\phi(x) := Bx + z$. Dabei ist $B \in \mathbb{K}^{n\times n}$ und $z \in \mathbb{K}^n$. Sei $\mathbb{K}^n$ mit einer Norm $\|\cdot\|_*$ ausgestattet. Dann gilt

$$\|\phi(x) - \phi(y)\|_* = \|B(x-y)\|_* \le \|B\|_*\|x-y\|_*,$$

wobei $\|\cdot\|_* : \mathbb{K}^{n\times n} \to \mathbb{R}$ die Matrixnorm bezeichnet, die der Vektornorm $\|\cdot\|_*$ zugeordnet ist. Nach dem Banachschen Fixpunktsatz 7.8 konvergiert die Iteration $x^{(t+1)} := Bx^{(t)} + z$ bezüglich der Vektornorm $\|\cdot\|_*$, falls $\|B\|_* < 1$.

8.1.1 Beispiele

Zur Konstruktion iterativer Verfahren zur Lösung von $Ax = b$ mit $A \in \mathbb{K}^{n\times n}$ und $b \in \mathbb{K}^n$ kann man eine reguläre $C \in \mathbb{K}^{n\times n}$ wählen und erhält obige Fixpunktform

$$Ax = b \Leftrightarrow Cx = Cx - Ax + b \Leftrightarrow x = \underbrace{(I - C^{-1}A)}_{=B} x + \underbrace{C^{-1}b}_{=z}. \tag{8.1}$$

Aus den Fehlerabschätzungen des Banachschen Fixpunktsatzes erkennen wir, dass $\|B\|_*$ möglichst klein sein sollte (mindestens kleiner 1). Andererseits sollte die Inverse C^{-1} möglichst leicht berechenbar sein. Zur Konstruktion einer geeigneten Matrix C zerlegen wir $A = L + D + R$ in der Form

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 0 & \cdots & & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n(n-1)} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{(n-1)n} \\ 0 & \cdots & & 0 \end{pmatrix}.$$

Das lineare Gleichungssystem $Ax = b$ lautet ausgeschrieben

$$\sum_{k=1}^{n} a_{jk} x_k = b_j, \qquad j = 1, \ldots, n,$$

bzw. falls $a_{jj} \neq 0$ für $j = 1, \ldots, n$

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

Beispiel 8.1 (Jacobi-Verfahren)**.** *Wir wählen einen Startwert $x^{(0)} \in \mathbb{R}^n$ und definieren die Iteration durch*

$$x_j^{(t)} := \frac{1}{a_{jj}} \left(b_j - \sum_{\substack{k=1 \\ k\neq j}}^{n} a_{jk} x_k^{(t-1)} \right), \qquad j = 1, \ldots, n, \quad t \in \mathbb{N}. \tag{8.2}$$

Mit der oben definierten Zerlegung $A = L + D + R$ entspricht dies einer Fixpunktiteration der Form (8.1) mit $C = D$.

Wenn in (8.2) $x_j^{(t)}$ berechnet wird, so sind die neuen Werte $x_1^{(t)}, \ldots, x_{j-1}^{(t)}$ bereits bekannt. Wenn diese an Stelle der alten Werte $x_1^{(t-1)}, \ldots, x_{j-1}^{(t-1)}$ verwendet werden, so erhält man das Gauß-Seidel-Verfahren.

Beispiel 8.2 (Gauß-Seidel-Verfahren)**.** *Wir wählen einen Startwert $x^{(0)} \in \mathbb{R}^n$ und definieren die Iteration durch*

$$x_j^{(t)} := \frac{1}{a_{jj}} \left(b_j - \sum_{k=1}^{j-1} a_{jk} x_k^{(t)} - \sum_{k=j+1}^{n} a_{jk} x_k^{(t-1)} \right), \qquad j = 1, \ldots, n, \quad t \in \mathbb{N}. \tag{8.3}$$

Dies entspricht einer Fixpunktiteration der Form (8.1) mit $C = L + D$.

Letzteres gilt, da (8.3) äquivalent ist zu

$$x^{(t)} = D^{-1}\left(b - Lx^{(t)} - Rx^{(t-1)}\right) \Leftrightarrow (D+L)x^{(t)} = b - Rx^{(t-1)}.$$

Daraus folgt die Behauptung wegen $I - (L+D)^{-1}(L+D+R) = -(L+D)^{-1}R$.

Beispiel 8.3 (SOR-Verfahren)**.** *Das SOR-Verfahren (Succesive Over Relaxation) verwendet nur einen Teil der Iterierten des Gauß-Seidel-Verfahrens. Für $x^{(0)} \in \mathbb{R}^n$ definiert man*

$$x^{(t)} = \omega D^{-1}\left(b - Lx^{(t)} - Rx^{(t-1)}\right) + (1-\omega)x^{(t-1)}, \qquad t \in \mathbb{N}, \tag{8.4}$$

mit einem festen Parameter $\omega \in (0,2)$. Dies entspricht einer Fixpunktiteration der Form (8.1) mit $C = L + \frac{1}{\omega}D$.

Für $\omega = 1$ entspricht das SOR-Verfahren dem Gauß-Seidel Verfahren. In allen drei Beispielen sind die bei der Iteration auftretenden Gleichungssysteme

$$Cx^{(t)} = (C-A)x^{(t-1)} + b$$

leicht zu lösen, da C eine Diagonalmatrix (Jacobi) bzw. eine untere Dreiecksmatrix (SOR) ist. Offen ist noch die Frage, für welche Matrizen solche Fixpunktiterationen konvergieren. Eine notwendige Bedingung dazu ist, dass D invertierbar ist, d.h. dass alle Hauptdiagonaleneinträge von A nicht 0 sind.

8.1.2 Konvergenz linearer Fixpunktiterationen

Wie bereits gesehen konvergiert wegen des Banachschen Fixpunktsatzes 7.8 die Iteration $x^{(t+1)} := Bx^{(t)} + z$ bezüglich der Vektornorm $\|\cdot\|_*$, falls $\|B\|_* < 1$. Wir könnten aber genauso gut eine andere Vektornorm auf $\mathbb{K}^n$ betrachten, da alle Normen auf $\mathbb{K}^n$ äquivalent sind. Es genügt also, dass in irgendeiner natürlichen Matrixnorm $\|B\| < 1$. Das Infimum aller natürlichen Matrixnormen von B ist durch den in Def. 5.10 definierten Spektralradius $\rho(B) = \max\{|\lambda| : \lambda \text{ Eigenwert von } B\}$ gegeben. Dieser ist selber i.A. keine Norm!

Lemma 8.4. *Für jede natürliche Matrixnorm $\|\cdot\|$ und jede Matrix $B \in \mathbb{K}^{n\times n}$ gilt*

$$\rho(B) \le \|B\|. \tag{8.5}$$

Umgekehrt gibt es zu jeder Matrix $B \in \mathbb{K}^{n\times n}$ und jedem $\epsilon > 0$ eine natürliche Matrixnorm $\|\cdot\|_*$ auf $\mathbb{K}^{n\times n}$, so dass

$$\|B\|_* \le \rho(B) + \epsilon. \tag{8.6}$$

Beweis. Wir müssen in diesem Beweis zwischen reellem und komplexem Körper unterscheiden und starten zunächst mit $\mathbb{K} = \mathbb{C}$.

Sei $\lambda \in \mathbb{C}$ ein Eigenwert von B mit $|\lambda| = \rho(B)$, und sei u ein zugehöriger Eigenvektor mit $\|u\| = 1$. Dann gilt (8.5) wegen

$$\|B\| = \sup_{x \in \mathbb{C}^n, \|x\|=1} \|Bx\| \geq \|Bu\| = \|\lambda u\| = |\lambda| = \rho(B).$$

Zum Beweis von (8.6) benutzen wir für $B \in \mathbb{C}^{n\times n} \setminus \{0\}$ (für die Nulllmatrix ist (8.6) trivial) eine Schur-Zerlegung $U = Q^* BQ$ mit einer unitären Matrix $Q \in \mathbb{C}^{n\times n}$ und einer oberen Dreiecksmatrix $U \in \mathbb{C}^{n\times n}$. Wegen $\det(\lambda I - B) = \det(\lambda I - U)$ sind die Eigenwerte von B gegeben durch $\lambda_j = u_{jj}$, $j = 1, \ldots, n$. Wir definieren

$$u := \max_{j,l=1,\ldots,n} |u_{jl}| > 0, \qquad \delta := \min\left(1, \frac{\epsilon}{(n-1)u}\right) > 0$$

sowie die Diagonalmatrix $D := \operatorname{diag}(1, \delta, \ldots, \delta^{n-1}) \in \mathbb{R}^{n\times n}$. Dann hat die obere Dreiecksmatrix $W := D^{-1}UD$ die Einträge $w_{jl} = \delta^{l-j} u_{jl}$ für $j \leq l$ und wegen $\delta \leq 1$ gilt

$$\|W\|_\infty = \max_{j=1,\ldots,n} \sum_{l=1}^{n} |w_{jl}| \leq \max_{j=1,\ldots,n} (|u_{jj}| + (n-1)\delta u) \leq \rho(B) + \epsilon.$$

Weiter gilt $W = R^{-1}BR$ mit der regulären Matrix $R := QD$. Diese verwenden wir zur Definition der Norm

$$\|x\|_* := \|R^{-1}x\|_\infty$$

auf $\mathbb{C}^n$ ein und erhalten

$$\|Bx\|_* = \|R^{-1}Bx\|_\infty = \|WR^{-1}x\|_\infty \leq \|W\|_\infty \|R^{-1}x\|_\infty = \|W\|_\infty \|x\|_*$$

für alle $x \in \mathbb{C}^n$. Daraus folgt $\|B\|_* \leq \|W\|_\infty \leq \rho(B) + \epsilon$.

Sei nun $\mathbb{K} = \mathbb{R}$. Zum Beweis von (8.6) können wir die bereits konstruierte komplexe Vektornorm $\|\cdot\|_*$ verwenden, weil diese auch eine Vektornorm auf $\mathbb{R}^n$ ist. Wegen

$$\|B\|_{*,\mathbb{R}} := \sup_{x \in \mathbb{R}^n, \|x\|_*=1} \|Bx\|_* \leq \sup_{x \in \mathbb{C}^n, \|x\|_*=1} \|Bx\|_* = \|B\|_*$$

folgt damit die Behauptung. Zum Beweis von (8.5) wählen wir wie oben $\lambda \in \mathbb{C}$ mit $|\lambda| = \rho(B)$ und einem zugehörigen Eigenvektor $\tilde{u} \in \mathbb{C}^n \setminus \{0\}$. Sei weiter $\tilde{\theta}$ der (existierende) Minimierer von $\|\operatorname{Im}(\theta\tilde{u})\|$ auf dem (kompakten) komplexen Einheitskreis $S^1 := \{\theta \in \mathbb{C}, |\theta| = 1\}$ und $u := \tilde{\theta}\tilde{u}$. Dann gilt mit $\lambda = |\lambda|\theta$ und $\theta \in S^1$

$$|\lambda| \|\operatorname{Im} u\| \leq |\lambda| \|\operatorname{Im}(\theta u)\| = \|\operatorname{Im}(\lambda u)\| = \|\operatorname{Im}(Bu)\| = \|B\operatorname{Im}(u)\| \leq \|B\| \|\operatorname{Im} u\|.$$

Falls $\|\operatorname{Im} u\| \neq 0$ folgt daraus die Behauptung. Andernfalls ist $u \in \mathbb{R}^n$ und damit auch $\lambda \in \mathbb{R}$ und die Behauptung folgt wie oben. □

Satz 8.5. *Sei $B \in \mathbb{K}^{n\times n}$. Dann konvergiert die lineare Fixpunktiteration*

$$x^{(t)} := Bx^{(t-1)} + z, \qquad t \in \mathbb{N},$$

genau dann für alle Startwerte $x^{(0)} \in \mathbb{K}^n$ und alle $z \in \mathbb{K}^n$, wenn der Spektralradius von B die Ungleichung

$$\rho(B) < 1$$

erfüllt.

Beweis. Falls $\rho(B) < 1$ existiert nach dem vorigen Lemma eine natürliche Matrixnorm $\|\cdot\|_*$, so dass $\|B\|_* < 1$. Deshalb ist die Abbildung $\phi(x) := Bx + z$ kontrahierend auf $\mathbb{K}^n$ bezüglich der Norm $\|\cdot\|_*$ mit Kontraktionsfaktor $\|B\|_*$, und die Folge $(x^{(t)})$ konvergiert nach dem Banachschen Fixpunktsatz bezüglich der Norm $\|\cdot\|_*$. Da nach Satz 5.2 alle Normen auf $\mathbb{K}^n$ äquivalent sind, konvergiert $(x^{(t)})$ auch bezüglich jeder anderen Vektornorm.

Ist $\rho(B) \geq 1$, so existiert ein Eigenwert $\lambda \in \mathbb{C}$ von B mit $|\lambda| \geq 1$. Sei $x \in \mathbb{C}^n \setminus \{0\}$ ein zugehöriger Eigenvektor. Dann ist die Iterationsfolge zum Startwert $x^{(0)} = x$ mit der rechten Seite $z = x$ gegeben durch $x^{(t)} = \left(\sum_{j=0}^{t} \lambda^j\right) x$, wie man leicht durch Induktion nach t zeigt. Da $|\lambda| \geq 1$, ist diese Folge nicht konvergent. Daraus folgt für $\mathbb{K} = \mathbb{C}$ die Behauptung.

Für $\mathbb{K} = \mathbb{R}$ folgt ebenfalls die Behauptung, falls λ und x reell sind. Andernfalls gilt $B\overline{x} = \overline{\lambda x}$. Sei nun entweder $x^{(0)} := \frac{1}{2}(x + \overline{x}) = \operatorname{Re}(x)$ oder $x^{(0)} := \frac{1}{2i}(x - \overline{x}) = \operatorname{Im}(x)$ und $z := x^{(0)}$. Dann gilt analog $x^{(t)} = \operatorname{Re}\left(\left(\sum_{j=0}^{t} \lambda^j\right) x\right)$ bzw. $x^{(t)} = \operatorname{Im}\left(\left(\sum_{j=0}^{t} \lambda^j\right) x\right)$. Da $\sum_{j=0}^{t} \lambda^j$ nicht konvergiert, kann mindestens eine der beiden Folgen $x^{(t)}$ nicht konvergieren und die Behauptung ist gezeigt. □

Wir können diesen Konvergenzsatz verwenden, um die Konvergenz der Fixpunktiterationen aus Abschnitt 8.1.1 zu untersuchen.

Satz 8.6 (Starkes Zeilensummenkriterium)**.** *Sei $A = (a_{jk})_{j,k=1}^{n} \in \mathbb{K}^{n\times n}$ strikt diagonaldominant, d.h.*

$$\sum_{\substack{k=1\\k\neq j}}^{n} |a_{jk}| < |a_{jj}|, \qquad j = 1, \ldots, n. \tag{8.7}$$

Dann konvergiert das Jacobi- (Bsp. 8.1) sowie das Gauß-Seidel-Verfahren (Bsp. 8.2).

Beweis. Wegen $|a_{jj}| > 0$ für $j = 1, \ldots, n$ ist die Matrix $D = \operatorname{diag}(a_{11}, \ldots, a_{nn})$ regulär und das Jacobi-Verfahren mit der Iterationsmatrix $B = -D^{-1}(L + R)$ durchführbar. Sei λ Eigenwert von B mit Eigenvektor v mit $\|v\|_\infty = 1$. Dann gilt

$$|\lambda| \leq \|B\|_\infty = \|D^{-1}(L + R)\|_\infty = \max_{j=1,\ldots,n} \left(\frac{1}{|a_{jj}|} \sum_{\substack{k=1\\k\neq j}}^{n} |a_{jk}| \right) < 1$$

und die Behauptung folgt aus Satz 8.5.

Auch das Gauß-Seidel-Verfahren ist durchführbar mit der Iterationsmatrix $B = -(D + L)^{-1}R$. Sei wiederum λ Eigenwert von B mit Eigenvektor v mit $\|v\|_\infty = 1$. Dann folgt aus

$$-(D + L)^{-1}Rv = \lambda v \Leftrightarrow \lambda v = -D^{-1}(\lambda L + R)v$$

für $|\lambda| \geq 1$ der Widerspruch

$$|\lambda| \leq \|D^{-1}(\lambda L + R)\|_\infty \leq |\lambda| \|D^{-1}(L + R)\|_\infty < |\lambda|$$

und somit ebenfalls $|\lambda| < 1$. □

Häufig ist die Bedingung (8.7) zu einschränkend. Es gibt weitere hinreichende Bedingungen wie z.B. das *Sassenfeld*-Kriterium, welches hinreichend für die Konvergenz des Gauß-Seidel-Verfahrens ist. Auch kann man zeigen, dass für eine hermitesche, positiv definite Matrix das SOR-Verfahren für alle $\omega \in (0, 2)$, und damit auch für das Gauß-Seidel-Verfahren, konvergiert.

Leider liegt der Kontraktionsfaktor $\|B\|$ beim Jacobi- wie beim SOR-Verfahren für lineare Gleichungssystem, die durch Diskretisierung von Randwertproblemen für elliptische Differentialgleichungen entstehen, häufig sehr nahe bei 1. In diesen Fällen ist die Konvergenz extrem langsam und die Verwendung der Verfahren nicht zu empfehlen. Die Verfahren werden stattdessen häufig als Teil anderer Verfahren eingesetzt.

8.2 Das Verfahren der konjugierten Gradienten

Wir betrachten in dem gesamten Abschnitt lineare Gleichungssysteme $Ax = b$ mit einer hermiteschen, positiv definiten Matrix $A \in \mathbb{K}^{n \times n}$. Das *cg-Verfahren* (engl. conjugate gradients) ist das am häufigsten verwendete iterative Lösungverfahren für solche Gleichungssysteme. Der Einfachheit halber werden wir bei der Herleitung nur den Fall $\mathbb{K} = \mathbb{R}$ betrachten, das Verfahren funktioniert jedoch auch im Komplexen.

Definition und Satz 8.7. *Sei $A \in \mathbb{K}^n$ hermitesch und positiv definit. Dann wird durch*

$$(x, y)_A := x^* A y, \qquad x, y \in \mathbb{K}^n$$

ein Skalarprodukt auf $\mathbb{K}^n$ definiert. Die zugehörige Norm $\|x\|_A := \sqrt{(x, x)_A}$ heißt Energienorm.

Beweis. Offensichtlich ist $(\cdot, \cdot)_A$ antilinear im ersten und linear im zweiten Argument. Da $A^* = A$, gilt $\overline{(x, y)_A} = y^* A^* x = (y, x)_A$. Schließlich gilt $(x, x)_A = x^* A x > 0$ für $x \neq 0$, da A positiv definit. □

8.2.1 Abstiegsverfahren

Die Lösung $\hat{x} := A^{-1} b$ des Gleichungssystems ist das eindeutig bestimmte Minimum des quadratischen Funktionals

$$\Phi(x) := \frac{1}{2} x^* A x - x^* b, \qquad x \in \mathbb{R}^n, \tag{8.8}$$

denn

$$\Phi(x) - \Phi(\hat{x}) = \frac{1}{2}\|x - \hat{x}\|_A^2. \tag{8.9}$$

Geometrisch bedeutet dies, dass der Graph der Funktion Φ bezüglich der Energienorm ein kreisförmiges Paraboloid ist, dessen Mittelpunkt über $\hat{x}$ liegt. Diese Beobachtung ist die Grundlage unserer Herleitung des Verfahrens der konjugierten Gradienten.

Ausgehend von einer Näherungslösung $x^{(t)}$, bestimmen wir im t-ten Iterationsschritt zunächst eine *Suchrichtung* $d^{(t)} \in \mathbb{R}^n \setminus \{0\}$ und wählen die nächste Iterierte über den Ansatz

$$x^{(t+1)} = x^{(t)} + \alpha_t d^{(t)}. \tag{8.10}$$

Dabei wählen wir die *Schrittweite* $\alpha_t \in \mathbb{R}$ als Minimum der Funktion

$$f(\alpha) := \Phi(x^{(t)} + \alpha d^{(t)}) = \Phi(x^{(t)}) + \alpha\left(d^{(t)*}Ax^{(t)} - d^{(t)*}b\right) + \frac{\alpha^2}{2}d^{(t)*}Ad^{(t)}.$$

Da $d^{(t)*}Ad^{(t)} > 0$, erhalten wir das Minimum von f durch Auflösen der Gleichung $f'(\alpha_t) = 0$ nach α_t. Bezeichnen wir mit

$$r^{(t)} := b - Ax^{(t)} \tag{8.11}$$

das *Residuum* im t-ten Iterationsschritt, so gilt

$$\alpha_t := \frac{r^{(t)*}d^{(t)}}{d^{(t)*}Ad^{(t)}}. \tag{8.12}$$

Ein Möglichkeit, die Suchrichtungen $d^{(t)}$ zu bestimmen, wäre zyklisch die kartesischen Einheitsvektoren durchlaufen zu lassen, d.h. sukzessive $e^{(1)}, \ldots, e^{(n)}, e^{(1)}, \ldots$ zu wählen. Dies ist aber nicht optimal. Als Abbruchkriterium eignet sich z.B. die Vorgabe einer Toleranz an die Norm des Residuums, d.h. die Forderung $\|r^{(t)}\| \leq \mathbf{TOL}$.

8.2.2 Gradientenverfahren

Eine naheliegende Möglichkeit wäre $d^{(t)} = -\operatorname{grad}\Phi(x^{(t)})$, die Richtung des steilsten Abstiegs von Φ. Da

$$\Phi(x + \Delta x) = \Phi(x) + (\Delta x)^*(Ax - b) + \underbrace{\frac{1}{2}(\Delta x)^*A\Delta x}_{=\mathcal{O}(\|\Delta x\|_2^2)},$$

gilt

$$\operatorname{grad}\Phi(x) = Ax - b, \qquad \text{also} \qquad -\operatorname{grad}\Phi(x^{(t)}) = r^{(t)}.$$

Dies muss allerdings nicht die beste Wahl der Suchrichtung sein, wie Abb. 8.1 an einem zweidimensionalen Beispiel veranschaulicht. Die Iterierten $x^{(t)}$ nähern sich dort in einem Zickzackkurs der Lösung. Ohne Beweis geben wir folgen Konvergenzsatz an.

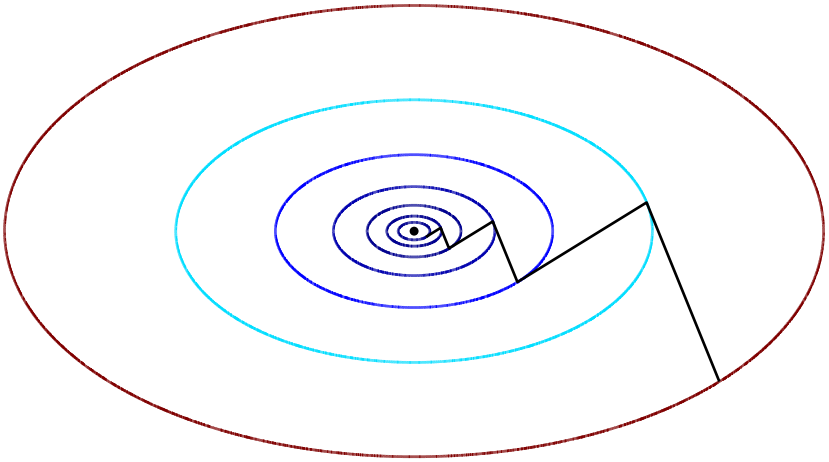


Abbildung 8.1: Gradientenverfahren für $A = \operatorname{diag}(1,5)$, $b = (1,5)^\top$ und $x^{(0)} = (3.5,0)^\top$: Die ersten Iterierten $x^{(t)}$ mit den zugehörigen Niveaulinien von Φ

Satz 8.8 (Gradientenverfahren)**.** *Für die Iterierten des Gradientenverfahrens (siehe Alg. 8.9 ohne Abbruchkriterium) gilt die Fehlerabschätzung*

$$\|x^{(t)} - A^{-1}b\|_A \leq \left(\frac{1-1/\kappa}{1+1/\kappa}\right)^t \|x^{(0)} - A^{-1}b\|_A, \qquad t \in \mathbb{N},$$

mit der spektralen Konditionszahl $\kappa = \operatorname{cond}_2(A)$.

8.2.3 cg-Verfahren

Bezüglich der $\|\cdot\|_A$-Geometrie sind die Niveaulinien

$$\{x \in \mathbb{R}^2 \mid \Phi(x) = \text{const}\}$$

im Fall $n = 2$ wegen (8.9) Kreise. Aufgrund der Minimalitätseigenschaft von α_t ist $x^{(1)}$ der Punkt, in dem die Gerade $\{x^{(0)} + \alpha d^{(0)} : \alpha \in \mathbb{R}\}$ eine Niveaulinie tangential berührt (siehe Abb. 8.1). Wählt man $d^{(1)}$ orthogonal zu $d^{(0)}$ bezüglich des $(\cdot,\cdot)_A$-Skalarprodukts, so liegt $x^{(2)}$ im Kreismittelpunkt, d.h. $x^{(2)}$ ist Lösung von $Ax = b$ und wir haben in 2 Schritten die exakte Lösung gefunden.

Dies führt uns auf den Ansatz

$$d^{(t+1)} = r^{(t+1)} + \beta_t d^{(t)}, \tag{8.13}$$

Algorithmus 8.9 Gradientenverfahren

Input: $A \in \mathbb{K}^{n\times n}$ hermitesch, positiv definit, $b \in \mathbb{K}^n$, Startvektor $x^{(0)} \in \mathbb{K}^n$, gewünschte Toleranz **TOL**

$t = 0$
$r^{(0)} = b - Ax^{(0)}$
$d^{(0)} = r^{(0)}$
while $\|r^{(t)}\| >$ **TOL do**
    $\alpha_t = \frac{r^{(t)*} d^{(t)}}{d^{(t)*} Ad^{(t)}}$
    $x^{(t+1)} = x^{(t)} + \alpha_t d^{(t)}$
    $t = t + 1$
    $r^{(t)} = b - Ax^{(t)}$
    $d^{(t)} = r^{(t)}$
end while

Output: Näherung $x^{(t)}$ an $x = A^{-1}b$ mit $\|Ax^{(t)} - b\| <$ **TOL**.

wobei wir β_t so bestimmen möchten, dass

$$(d^{(t+1)}, d^{(t)})_A = 0.$$

Dadurch erhalten wir die Gleichung

$$\beta_t = -\frac{r^{(t+1)*} Ad^{(t)}}{d^{(t)*} Ad^{(t)}}. \tag{8.14}$$

Vektoren, die bezüglich des inneren Produkts $(\cdot,\cdot)_A$ orthogonal sind, werden auch *A-konjugiert* genannt. Da die Suchrichtungen $d^{(t)}$ zueinander A-konjugiert gewählt werden, wird das Verfahren, das wir soeben hergeleitet haben, *Verfahren der konjugierten Gradienten* oder kurz *CG-Verfahren* (engl. *conjugate gradients*) genannt. Der Algorithmus, welcher in Alg. 8.10 formuliert ist, sollte so nicht implementiert werden. Durch einfache Umformungen können sowohl Rechenoperationen gespart werden als auch die Stabilität verbessert werden.

8.2.4 Analysis cg-Verfahren

Wir zeigen zunächst Orthogonalitätseigenschaften der Residuen und Suchrichtungen:

Lemma 8.11. *Ist $r^{(t)} \neq 0$ für $t = 0, \ldots, m$, so sind die ersten $m+1$ Schritte des CG-Verfahrens durchführbar (d.h. $d^{(t)} \neq 0$ für $t = 0, \ldots, m$), und es gilt*

(a) $r^{(m)*} d^{(j)} = 0$ *für alle* $0 \leq j < m$,
(b) $r^{(m)*} r^{(j)} = 0$ *für alle* $0 \leq j < m$,
(c) $(d^{(m)}, d^{(j)})_A = 0$ *für alle* $0 \leq j < m$.

Beweis. Wegen (8.10) gilt $Ax^{(t+1)} = Ax^{(t)} + \alpha_t Ad^{(t)}$ für $t \geq 0$, und somit

$$r^{(t+1)} = r^{(t)} - \alpha_t Ad^{(t)}. \tag{8.15}$$

Algorithmus 8.10 CG-Verfahren

Input: $A \in \mathbb{K}^{n\times n}$ hermitesch, positiv definit, $b \in \mathbb{K}^n$, Startvektor $x^{(0)} \in \mathbb{K}^n$, gewünschte Toleranz **TOL**

$t = 0$
$r^{(0)} = b - Ax^{(0)}$
$d^{(0)} = r^{(0)}$
while $\|r^{(t)}\| >$ **TOL do**
    $\alpha_t = \frac{r^{(t)*}d^{(t)}}{d^{(t)*}Ad^{(t)}}$
    $x^{(t+1)} = x^{(t)} + \alpha_t d^{(t)}$
    $t = t + 1$
    $r^{(t)} = b - Ax^{(t)}$
    $\beta_{t-1} = -\frac{r^{(t)*}Ad^{(t-1)}}{d^{(t-1)*}Ad^{(t-1)}}$
    $d^{(t)} = r^{(t)} + \beta_{t-1}d^{(t-1)}$
end while

Output: Näherung $x^{(t)}$ an $x = A^{-1}b$ mit $\|Ax^{(t)} - b\| <$ **TOL**.

Mit der Definition (8.12) von α_t erhalten wir

$$r^{(t+1)*}d^{(t)} = (r^{(t)} - \alpha_t Ad^{(t)})^* d^{(t)} = r^{(t)*}d^{(t)} - \alpha_t d^{(t)*}Ad^{(t)} = 0. \tag{8.16}$$

Wir nehmen nun an, es sei $d^{(t)} = 0$ für ein $t \leq m$, und t sei die kleinste Zahl mit dieser Eigenschaft. Dann gilt aufgrund von (8.13) $r^{(t)} = -\beta_{t-1}d^{(t-1)}$, und wegen (8.16) $0 = r^{(t)*}d^{(t-1)} = -\beta_{t-1}\|d^{(t-1)}\|_2^2$. Aufgrund der Minimalität von t ist $d^{(t-1)} \neq 0$ und deshalb $\beta_{t-1} = 0$. Hieraus folgt $r^{(t)} = -\beta_{t-1}d^{(t-1)} = 0$ im Widerspruch zu den Voraussetzungen. Also ist das CG-Verfahren durchführbar.

Wir zeigen nun durch Induktion nach ℓ, dass die Aussagen (a),(b) und (c) für alle $m = 1, \ldots, \ell$ gültig sind.

$\underline{\ell = 1}$:
Setzen wir $t = 0$ in (8.16), so erhalten wir (a) für $m = 1$ und $j = 0$. Da $r^{(0)} = d^{(0)}$, ist auch (b) erfüllt. Schließlich gilt auch (c) aufgrund von (8.14) und (8.13) für $t = 0$ (so hatten wir die Wahl (8.14) von β_t ja gerade hergeleitet!).

$\underline{\ell \rightsquigarrow \ell + 1}$:
Zunächst gilt $r^{(\ell+1)*}d^{\ell} = 0$ wegen (8.16). Mit (8.15) erhalten wir

$$r^{(\ell+1)*}d^{(j)} = \underbrace{r^{(\ell)*}d^{(j)}}_{=0,\ \text{IA (a)}} - \alpha_\ell \underbrace{d^{(\ell)*}Ad^{(j)}}_{=0,\ \text{IA (c)}} = 0, \qquad 0 \leq j < \ell.$$

Folglich gilt (a) mit $\ell + 1$ anstelle von ℓ.

$r^{(0)} = d^{(0)}$ und (8.13) ist äquivalent zu $r^{(j)} = d^{(j)} - \beta_{j-1}d^{(j-1)}$ für $j = 1, \ldots, \ell$. Deshalb folgt (b) für $\ell + 1$ aus der schon bewiesenen Behauptung (a).

Die Aussage (c) für $j = \ell$ und $m = \ell + 1$ folgt unmittelbar aus der Wahl von β_ℓ,

vgl. (8.14) und (8.13). Für $j < \ell$ ergibt sich aufgrund von (8.13)

$$(d^{(\ell+1)}, d^{(j)})_A = (r^{(\ell+1)}, d^{(j)})_A + \beta_\ell \underbrace{(d^{(\ell)}, d^{(j)})_A}_{=0,\ \text{IA (c)}} = r^{(\ell+1)*} A d^{(j)}.$$

Ersetzt man in dieser Gleichung $Ad^{(j)}$ mit Hilfe von (8.15), so erhält man mit Hilfe der schon bewiesenen Behauptung (b)

$$\alpha_j (d^{(\ell+1)}, d^{(j)})_A = r^{(\ell+1)*} r^{(j)} - r^{(\ell+1)*} r^{(j+1)} = 0, \qquad 0 \le j < \ell.$$

Wir müssen also zum Nachweis von (c) lediglich noch zeigen, dass $\alpha_j \neq 0$. Nehmen wir also an, es sei $\alpha_j = 0$. Aufgrund der Definition (8.12) gilt dann $r^{(j)*} d^{(j)} = 0$, und aus (8.13) folgt

$$0 = r^{(j)*} d^{(j)} = r^{(j)*} r^{(j)} + \beta_j \underbrace{r^{(j)*} d^{(j-1)}}_{=0,\ \text{(a)}} = \|r^{(j)}\|_2^2$$

für $1 \le j < \ell$, bzw. $0 = r^{(j)*} d^{(j)} = \|r^{(0)}\|_2^2$ für $j = 0$. In jedem Fall haben wir also einen Widerspruch zur Annahme $r^{(j)} \neq 0$ für $j = 0, \ldots, \ell$. Damit ist der Induktionsbeweis abgeschlossen. □

Teil (c) des Lemmas besagt, dass nicht nur aufeinanderfolgende, sondern *alle* Suchrichtungen paarweise zueinander A-konjugiert sind. Weiterhin sind auch alle Residuen paarweise orthogonal bezüglich des Euklidischen Skalarprodukts. Da Orthogonalsysteme linear unabhängig sind, muss spätestens nach n Schritten die Abbruchbedingung $r^{(t)} = 0$ erfüllt sein:

Korollar 8.12. *Das CG-Verfahren bricht nach spätestens n Schritten mit der exakten Lösung ab.*

Für die Praxis ist dieses Resultat allerdings nur von eingeschränkter Bedeutung, da das CG-Verfahren meist nur mit deutlich weniger als n Schritten effizient ist. Außerdem tritt durch Rundungsfehler ein Verlust der Orthogonalitätseigenschaften auf, durch den das Ergebnis dieses Lemmas für die Praxis kaum relevant ist. Für die Interpretation des CG-Verfahrens als iteratives Verfahren zitieren wir hier ohne Beweis (siehe z.B. Martin Hanke-Bourgeois, *Grundlagen der numerischen Mathematik und des wissenschaftlichen Rechnens*, B. G. Teubner, 2002)

Satz 8.13 (cg-Verfahren)**.** *Für die Iterierten des cg-Verfahrens (siehe Alg. 8.10 ohne Abbruchkriterium) gilt die Fehlerabschätzung*

$$\|x^{(t)} - A^{-1}b\|_A \le 2 \left(\frac{1 - 1/\sqrt{\kappa}}{1 + 1/\sqrt{\kappa}} \right)^t \|x^{(0)} - A^{-1}b\|_A, \qquad t \in \mathbb{N},$$

mit der spektralen Konditionszahl $\kappa = \mathrm{cond}_2(A)$.

Diese Fehlerabschätzung kann mit Hilfe folgender Optimalitätseigenschaft des CG-Verfahrens bezüglich der Energienorm bewiesen werden.

Definition 8.14. *Sei $A \in \mathbb{K}^{n\times n}$ und $y \in \mathbb{K}^n$. Dann heißt der Untervektorraum*

$$\mathcal{K}_t(A, y) := \operatorname{span}\left\{y, Ay, \ldots, A^{t-1}y\right\}$$

Krylov-Raum *der Dimension t von A bezüglich y.*

Lemma 8.15. *Ist $r^{(j)} \neq 0$ für $j = 0, \ldots, t$, so gilt*

$$\mathcal{K}_{t+1}(A, r^{(0)}) = \operatorname{span}\left\{d^{(0)}, \ldots, d^{(t)}\right\} = \operatorname{span}\left\{r^{(0)}, \ldots, r^{(t)}\right\}.$$

Beweis. Wegen Lemma 8.11 sind die Residuen $r^{(j)}$ für $j = 0, \ldots, t$ zueinander orthogonal (damit linear unabhängig) und die Richtungen $d^{(j)}$ für $j = 0, \ldots, t$ zueinander A-orthogonal und ebenfalls linear unabhängig. Wegen $d^{(0)} = r^{(0)}$ und der Konstruktion (8.13) der Richtungen folgt per Induktion

$$d^{(j)} \in \operatorname{span}\left\{r^{(0)}, \ldots, r^{(j)}\right\}, \qquad j = 0, \ldots, t. \tag{8.17}$$

Aus der linearen Unabhängigkeit von $d^{(0)}, \ldots, d^{(t)}$ folgt

$$\operatorname{span}\left\{d^{(0)}, \ldots, d^{(t)}\right\} = \operatorname{span}\left\{r^{(0)}, \ldots, r^{(t)}\right\}.$$

Die Behauptung folgt damit aus der linearen Unabhängigkeit von $r^{(0)}, \ldots, r^{(t)}$ falls gilt

$$r^{(j)} \in \mathcal{K}_{j+1}(A, r^{(0)}) = \operatorname{span}\left\{r^{(0)}, \ldots, A^j r^{(0)}\right\}, \qquad j = 0, \ldots, t.$$

Dies folgt wiederum per Induktion aus $r^{(0)} \in \mathcal{K}_1(A, r^{(0)})$, (8.15) und (8.17). □

Die entscheidende Eigenschaft des CG-Verfahrens lautet nun:

Satz 8.16. *Für die t-te Iterierte des CG-Verfahrens gilt, sofern das Verfahren nicht vorher abbricht,*

$$x^{(t)} \in x^{(0)} + \mathcal{K}_t(A, r^{(0)}), \tag{8.18}$$

und $x^{(t)}$ ist in diesem affinen Unterraum die eindeutig bestimmte Minimalstelle der Zielfunktion $\Phi(x) := \frac{1}{2}\|x - A^{-1}b\|_A^2$.

Beweis. Aus der Konstruktion (8.10) der Iterierten $x^{(j)}$ folgt

$$x^{(j)} = x^{(0)} + \sum_{\ell=0}^{j-1} \alpha_\ell d^{(\ell)}, \qquad j = 0, \ldots, m$$

mit einem Abbruchindex $m \leq n$ und das vorige Lemma liefert (8.18) mit $t \leq m$.

Sei weiter $\hat{x} := A^{-1}b = x^{(m)}$. Es gilt

$$\hat{x} - x^{(t)} = \sum_{\ell=1}^{m-1} \alpha_\ell d^{(\ell)} - \sum_{\ell=1}^{t-1} \alpha_\ell d^{(\ell)} = \sum_{\ell=t}^{m-1} \alpha_\ell d^{(\ell)}. \tag{8.19}$$

Für beliebiges $x \in x^{(0)} + \mathcal{K}_t(A, r^{(0)})$ gilt

$$\hat{x} - x = \hat{x} - x^{(t)} + x^{(t)} - x = \hat{x} - x^{(t)} + \sum_{\ell=0}^{t-1} \delta_\ell d^{(\ell)}$$

mit $\delta_\ell \in \mathbb{K}$. Wegen der A-Orthogonalität der Suchrichtungen folgt

$$\hat{x} - x^{(t)} \perp_A x^{(t)} - x$$

und daraus mit dem Satz des Pythargoras die Behauptung:

$$\Phi(x) - \Phi(\hat{x}) = \frac{1}{2}\|\hat{x} - x\|_A^2 = \frac{1}{2}\left\|\hat{x} - x^{(t)}\right\|_A^2 + \frac{1}{2}\left\|\sum_{\ell=0}^{t-1} \delta_\ell d^{(\ell)}\right\|_A^2 \geq \Phi(x^{(t)}) - \Phi(\hat{x}).$$

□

Bemerkung 8.17. *Alle vorgestellten iterativen Verfahren hängen von der rechten Seite b ab. Sollen mehrere lineare Gleichungssysteme mit der gleichen Matrix A jedoch unterschiedlichen rechten Seiten b gelöst werden, so muss jeweils das komplette Verfahren durchlaufen werden. Beim Algorithmus von Gauß muss nur einmal mit Aufwand $\mathcal{O}(n^3)$ die LU-Zerlegung berechnet werden. Danach können beliebig viele Gleichungssysteme mit der gleichen Matrix jeweils in $\mathcal{O}(n^2)$ Schritten gelöst werden.*

9 Eigenwertprobleme

In diesen Kapitel beschäftigen wir uns mit der numerischen Berechnung von Eigenwerten. Dabei stehen uns sehr unterschiedliche Mittel und Methoden zur Verfügung. Zum einen iterative Methoden, die eine Folge von Matrizen generieren, welche alle dieselben Eigenwerte besitzen und gegen eine Dreiecksmatrix konvergieren. Die gesuchten Eigenwerte sind dann genau die Hauptdiagonaleneinträge der Dreiecksmatrix.

Häufig sind wir jedoch nicht an allen Eigenwerten interessiert, sondern nur an einigen wenigen. So basiert z.B. der PageRank Algorithmus von google auf einem Eigenwertproblem, bei dem nur der Eigenvektor zum betragsgrößten Eigenwert gesucht wird. Die Einträge des Eigenvektors werden dann verwendet, um die Reihenfolge, in der Suchergebnisse ausgegeben werden, zu bestimmen. Speziell für solche Probleme gibt es einen denkbar simplen Algorithmus:

Algorithmus 9.1 Vektoriteration/ Potenzmethode

Input: $A \in \mathbb{K}^{n\times n}$ mit n Eigenwerten mit paarweise unterschiedlichem Betrag, Zufallsvektor v

1: **for** $i = 1, \dots$ **do**
2: $\quad v = Av$
3: $\quad v = \frac{1}{\|v\|}v$
4: **end for**

Output: Näherung v an einen Eigenvektor zum betragsgrößten Eigenwert von A.

Wir werden später sehen, warum dieser Algorithmus funktioniert. Zunächst werden wir uns jedoch, wie üblich, mit der Kondition eines Eigenwertproblems und einigen theoretischen Grundlagen beschäftigen.

9.1 Theoretische Grundlagen

Wir verweisen auf Def. 5.7 und Satz 5.9, welche aus der linearen Algebra bekannt sein sollten.

Problem 9.2 (Eigenwertproblem)**.** *Sei $A, B \in \mathbb{K}^{n\times n}$. Gesucht sind Eigenwerte $\lambda \in \mathbb{C}$ und Eigenvektoren $v \in \mathbb{C}^n \setminus \{0\}$, sodass*

$$Av = \lambda v. \tag{9.1}$$

Bei einem verallgemeinerten Eigenwertproblem suchen wir Eigenpaare $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n \setminus \{0\}$ *für die gilt*

$$Av = \lambda Bv. \tag{9.2}$$

Ist B invertierbar, so ist (9.2) äquivalent zum Eigenwertproblem $B^{-1}Av = \lambda v$.

Satz 9.3. *Sei* $A, Q \in \mathbb{K}^{n\times n}$ *und sei* Q *regulär. Dann heissen* A *und* $Q^{-1}AQ$ *ähnlich und besitzen dieselben Eigenwerte.*

Beweis. Die Eigenwerte von A sind Nullstellen des charakteristischen Polynoms $p(\lambda) := \det(A - \lambda \,\mathrm{id})$. Mit Hilfe des Determinanten-Multiplikationssatzes folgt die Aussage aus

$$\begin{aligned}\det(Q^{-1}AQ - \lambda \,\mathrm{id}) &= \det(Q^{-1}(A - \lambda \,\mathrm{id})Q) = \det(Q^{-1}) \det(A - \lambda \,\mathrm{id}) \det(Q) \\ &= \det(Q^{-1}Q) \det(A - \lambda \,\mathrm{id}) = \det(A - \lambda \,\mathrm{id}).\end{aligned}$$

□

Bemerkung 9.4. *Ist* A *eine Dreiecksmatrix, so sind die Hauptdiagonaleneinträge* a_{jj} *die Eigenwerte von* A*, da dann das charakteristische Polynom gegeben ist durch* $\prod_{j=1}^{n}(a_{jj} - \lambda)$*. Eine Möglichkeit zur Berechnung von Eigenwerten für beliebige Matrizen* A *besteht daher darin, sie durch Ähnlichkeitstransformationen auf Dreiecksgestalt zu bringen.*

Lemma 9.5. *Seien* $A, B \in \mathbb{C}^{n\times n}$ *und* $\|\cdot\|$ *eine natürliche Matrixnorm. Dann gilt für jeden Eigenwert* λ *von* B*, der nicht Eigenwert von* A *ist, die Ungleichung*

$$\|(\lambda \,\mathrm{id} - A)^{-1}(A - B)\| \geq 1.$$

Beweis. Sei $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n \setminus \{0\}$ Eigenpaar von B, $\|v\| = 1$ und sei λ kein Eigenwert von A. Dann gilt

$$(B - A)v = (\lambda \,\mathrm{id} - A)v \quad \Leftrightarrow \quad (\lambda \,\mathrm{id} - A)^{-1}(B - A)v = v.$$

Hieraus folgt die Behauptung wegen Satz und Definition 5.3 und

$$\|(\lambda \,\mathrm{id} - A)^{-1}(A - B)\| = \sup_{x\in\mathbb{C}^n, \|x\|=1} \|(\lambda \,\mathrm{id} - A)^{-1}(A - B)x\| \geq \|v\| = 1.$$

□

Satz 9.6 (Kondition Eigenwertproblem)**.** *Sei* $A, \delta A \in \mathbb{C}^{n\times n}$ *und sei* A *diagonalisierbar, d.h. es existieren* n *linear unabhängige Eigenvektoren* $v_1, \ldots, v_n$ *von* A*. Dann gibt es zu jedem Eigenwert* $\lambda(A + \delta A)$ *von* $A + \delta A$ *einen Eigenwert* $\lambda(A)$ *von* A*, sodass mit* $V := (v_1, \ldots, v_n) \in \mathbb{C}^{n\times n}$ *gilt*

$$|\lambda(A + \delta A) - \lambda(A)| \leq \mathrm{cond}_2(V)\|\delta A\|_2. \tag{9.3}$$

Beweis. Sei V die Matrix, welche die linear unabhängigen Eigenvektoren $v_1, \dots, v_n$ von A zu den Eigenwerten $\lambda_1(A), \dots, \lambda_n(A)$ als Spaltenvektoren besitzt. Dann gilt

$$A = V \operatorname{diag}(\lambda_1, \dots, \lambda_n) V^{-1}$$

und daraus für $\lambda \in \mathbb{C}$, der kein Eigenwert von A ist,

$$\|(\lambda \operatorname{id} - A)^{-1}\|_2 \leq \operatorname{cond}_2(V) \max_{j=1}^{n} |\lambda - \lambda_j(A)|^{-1}.$$

Ist λ Eigenwert von $B := A + \delta A$, so folgt die Behauptung aus dem vorigen Lemma. □

Falls A hermitesch, so folgt wegen Satz 5.9 die Existenz einer Orthonormalbasis aus Eigenvektoren, d.h. V ist unitär und $\operatorname{cond}_2(V) = 1$. In diesem Fall ist das Eigenwertproblem gut konditioniert.

Satz 9.7 (Gerschgorin)**.** *Alle Eigenwerte einer Matrix $A \in \mathbb{K}^{n\times n}$ liegen in der Vereinigung der* Gerschgorin-Kreise

$$K_j := \left\{ z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{k=1, k\neq j}^{n} |a_{jk}| \right\}, \qquad j = 1, \dots, n. \tag{9.4}$$

Beweis. Sei in Lem. 9.5 $A = D = \operatorname{diag}(a_{11}, \dots, a_{nn})$, $B = A$ und die Matrixnorm die Zeilensummennorm aus Satz 5.4. Dann folgt für $\lambda \neq a_{jj}$

$$\|(\lambda \operatorname{id} - D)^{-1}(A - D)\|_\infty = \max_{j=1}^{\infty} \frac{\sum_{k=1,k\neq j}^{n} |a_{jk}|}{|\lambda - a_{jj}|} \geq 1,$$

d.h. λ liegt in einem der Gerschgorin-Kreise. □

Per se ist nicht klar, ob in jedem Gerschgorin-Kreis auch tatsächlich ein Eigenwert liegt. Wenn die Vereinigung von m Gerschgorin-Kreisen $U = \bigcup_{i=1}^{m} K_{j_i}$ jedoch disjunkt zu den restlichen Kreisen $V = \overline{\bigcup_{j=1}^{n} K_j \setminus U}$ ist, so liegen in U genau m und in V genau $n-m$ Eigenwerte von A (der algebraischen Vielfachheit nach gezählt). Dies zeigt man wie folgt: Sei $A_t := D + t(A - D)$. Dann gilt die Aussage für $A_0 = D$. Weiter kann man zeigen, dass die Nullstellen eines Polynoms stetig von den Koeffizienten abhängen, d.h. die Eigenwerte $\lambda(A_t)$ der Matrizen A_t hängen stetig von den Koeffizienten des charakteristischen Polynoms und damit von den Einträgen der Matrix A_t ab. Da U und V disjunkt sind, können die Eigenwerte $\lambda(A_t)$ somit nicht von U nach V springen, d.h. die Aussage gilt auch für $A_1 = A$.

Satz 9.8 (Rayleigh)**.** *Sei $A \in \mathbb{C}^{n\times n}$ hermitesch und λ_n (bzw. λ_1) der kleinste (bzw. größte) Eigenwert von A. Dann gilt*

$$\lambda_n = \min_{\|x\|_2 = 1} x^* A x, \qquad \lambda_1 = \max_{\|x\|_2 = 1} x^* A x. \tag{9.5}$$

Für $x \neq 0$ nennt man x^*Ax/x^*x den Rayleigh-Quotienten von x. Zur Notation: $x^* = (\overline{x_1}, \dots, \overline{x_n})$ für $x = (x_1, \dots, x_n)^\top \in \mathbb{K}^n$.

Beweis. Sei $\{v_1, \dots, v_n\}$ eine Orthonormalbasis von Eigenvektoren von A mit zugehörigen Eigenwerten $\lambda_1, \dots, \lambda_n$. Für einen Vektor $x \in \mathbb{K}^n$ der Länge $\|x\|_2 = 1$ gilt daher

$$x = \sum_{j=1}^{n} (v_j^* x) v_j \qquad \text{und} \qquad \sum_{j=1}^{n} |v_j^* x|^2 = 1.$$

Es folgt

$$x^* A x = x^* \sum_{j=1}^{n} (v_j^* x) A v_j = \sum_{j=1}^{n} \lambda_j |v_j^* x|^2 \le \lambda_1 \sum_{j=1}^{n} |v_j^* x|^2 = \lambda_1,$$

d.h. $\sup_{\|x\|_2=1} x^* A x \le \lambda_1$. Da das Supremum für $x = v_n$ angenommen wird, folgt die erste Gleichung in (9.5). Die zweite Behauptung zeigt man analog. □

9.2 Vektoriteration

Die Vektoriteration Alg. 9.1 erzeugt ausgehend von einem Startvektor $x^{(0)} \in \mathbb{K}^n$ eine Folge von Iterierten $x^{(t)} := \frac{Ax^{(t-1)}}{\|Ax^{(t-1)}\|}$ für $t \in \mathbb{N}$.

Satz 9.9. *Sei $A \in \mathbb{K}^{n\times n}$ diagonalisierbar mit den Eigenwerten (ihrer Vielfachheit nach gezählt) $\lambda_1, \dots, \lambda_n$ mit $|\lambda_1| > |\lambda_2| \ge \lambda_j$ für alle $j = 3, \dots, n$. Ferner sei $\{v_1, \dots, v_n\} \subset \mathbb{K}^n$ eine Basis von $\mathbb{K}^n$ aus normierten Eigenvektoren zu A und $x^{(0)} = \sum_{j=1}^{n} \alpha_j v_j \in \mathbb{K}^n$ mit $\alpha_1 \neq 0$. Dann gilt für die Iterierten $x^{(t)} := \frac{Ax^{(t-1)}}{\|Ax^{(t-1)}\|}$*

$$\left\| x^{(t)} - \frac{\lambda_1^t \alpha_1}{|\lambda_1^t \alpha_1|} v_1 \right\| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^t \right), \qquad t \to \infty. \tag{9.6}$$

Sei $\lambda^{(t)} := \frac{(Ax^{(t)})_k}{x_k^{(t)}}$ für ein $k \in \{1, \dots, n\}$ mit $x_k^{(t)} \neq 0$. Dann gilt ebenfalls

$$|\lambda^{(t)} - \lambda_1| = \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^t \right), \qquad t \to \infty. \tag{9.7}$$

Beweis. Induktiv zeigt man $x^{(t)} = \frac{A^t x^{(0)}}{\|A^t x^{(0)}\|}$ für $t \in \mathbb{N}$. Weiter gilt

$$A^t x^{(0)} = \sum_{j=1}^{n} \alpha_j A^t v_j = \sum_{j=1}^{n} \alpha_j \lambda_j^t v_j = \lambda_1^t \alpha_1 \left(v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1} \right)^t v_j \right)$$

und daraus

$$x^{(t)} = \frac{\lambda_1^t \alpha_1}{|\lambda_1^t \alpha_1|} \frac{v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1} \right)^t v_j}{\left\| v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1} \right)^t v_j \right\|}.$$

Wegen $\|v_j\| = 1$ für $j = 1, \dots, n$ und $|\lambda_j| \leq |\lambda_2|$ für $j = 2, \dots, n$ gilt

$$\begin{aligned} 1 = \|v_1\| &\leq \left\| v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j \right\| + \left\| \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j \right\| \\ &\leq \left\| v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j \right\| + \underbrace{(n-1) \max_{j=2}^{n} \left|\frac{\alpha_j}{\alpha_1}\right|}_{=:\tilde{C}} \left|\frac{\lambda_2}{\lambda_1}\right|^t \end{aligned}$$

und damit für hinreichend großes t wegen $\left|\frac{\lambda_2}{\lambda_1}\right|^t \to 0$ für $t \to \infty$

$$\left\| v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j \right\| \geq \frac{1}{2}.$$

Es folgt

$$\begin{aligned} \left\| x^{(t)} - \frac{\lambda_1^t \alpha_1}{|\lambda_1^t \alpha_1|} v_1 \right\| &\leq 2 \sum_{j=2}^{n} \left|\frac{\alpha_j}{\alpha_1}\right| \left|\frac{\lambda_{j-1}}{\lambda_1}\right|^t + \left| \left\| v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j \right\|^{-1} - 1 \right| \\ &\leq 2\tilde{C} \left|\frac{\lambda_2}{\lambda_1}\right|^t + 2 \left(1 - \left\| v_1 + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j \right\| \right) \\ &\leq 4\tilde{C} \left|\frac{\lambda_2}{\lambda_1}\right|^t \end{aligned}$$

und damit die erste Behauptung. Die Konvergenz der Eigenwerte folgt analog wegen

$$\lambda^{(t)} = \frac{\left(A^{t+1} x^{(0)}\right)_k}{\left(A^t x^{(0)}\right)_k} = \frac{\lambda_1^{t+1} \alpha_1 \left((v_1)_k + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^{t+1} (v_j)_k \right)}{\lambda_1^t \alpha_1 \left((v_1)_k + \sum_{j=2}^{n} \frac{\alpha_j}{\alpha_1} \left(\frac{\lambda_j}{\lambda_1}\right)^{t} (v_j)_k \right)}.$$

□

Satz 9.10. *Sei zusätzlich zu den Voraussetzungen des vorigen Satzes $A \in \mathbb{K}^{n \times n}$ hermitesch. Definiert man die Approximation $\lambda^{(t)}$ an den Eigenwert λ_n durch den Rayleigh-Quotienten $\lambda^{(t)} := x^{(t)^*} A x^{(t)} / \|x^{(t)}\|_2^2$, so konvergiert $\lambda^{(t)}$ quadratisch gegen λ_1, d.h.*

$$|\lambda^{(t)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2t} \right), \qquad t \to \infty.$$

Beweis. Wir verwenden die Bezeichnungen wie im Beweis des vorigen Satzes. Da A hermitesch ist, sind wegen Satz 5.9 alle Eigenwerte reell und es existiert eine Ortho-

Algorithmus 9.11 Shift-and-Invert Iteration/ Wieland-Iteration

Input: $A, B \in \mathbb{K}^{n\times n}$, Zufallsvektor v, $\rho \in \mathbb{C}$

1: Berechne Zerlegung $LU = P(A - \rho B)Q$
2: **for** $i = 1, \ldots$ **do**
3: $\quad v = PBv$
4: $\quad$ Löse $Lz = v$ per Vorwärtssubstitution
5: $\quad$ Löse $Uv = z$ per Rückwärtssubstitution
6: $\quad v = Qx$
7: $\quad v = \frac{1}{\|v\|}x$
8: **end for**

Output: Näherung v an einen Eigenvektor zu dem Eigenwert von $Av = \lambda Bv$ mit kleinstem Abstand zu ρ

normalbasis $\{v_1, \ldots, v_n\} \subset \mathbb{K}^n$ aus Eigenvektoren von A. Wir erhalten

$$\begin{aligned}
\lambda^{(t)} =& \frac{x^{(t)^*} A x^{(t)}}{\|x^{(t)}\|_2} = \frac{\left(A^t x^{(0)}\right)^* A^{t+1} x^{(0)}}{\|A^t x^{(0)}\|_2} \\
=& \frac{\lambda_1^{2t+1}|\alpha_1|^2 \left(v_1^* + \sum_{j=2}^n \frac{\overline{\alpha_j}}{\overline{\alpha_1}} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j^*\right)\left(v_1 + \sum_{j=2}^n \frac{\alpha_j}{\alpha_1}\left(\frac{\lambda_j}{\lambda_1}\right)^{t+1} v_j\right)}{\lambda_1^{2t}|\alpha_1|^2 \left(v_1^* + \sum_{j=2}^n \frac{\overline{\alpha_j}}{\overline{\alpha_1}} \left(\frac{\lambda_j}{\lambda_1}\right)^t v_j^*\right)\left(v_1 + \sum_{j=2}^n \frac{\alpha_j}{\alpha_1}\left(\frac{\lambda_j}{\lambda_1}\right)^{t} v_j\right)} \\
=& \lambda_1 \frac{1 + \sum_{j=2}^n \left|\frac{\alpha_j}{\alpha_1}\right|^2 \left(\frac{\lambda_j}{\lambda_1}\right)^{2t+1}}{1 + \sum_{j=2}^n \left|\frac{\alpha_j}{\alpha_1}\right|^2 \left(\frac{\lambda_j}{\lambda_1}\right)^{2t}}
\end{aligned}$$

und daraus analog zum vorigen Beweis die Behauptung. □

Die Einschränkung $\alpha_1 \neq 0$ in den vorigen Sätzen ist nicht wesentlich, da sie im Laufe des Algorithmus fast immer durch Rundungsfehler erfüllt ist. Die Vektoriteration hat jedoch mehrere Nachteile: Zum einen erhalten wir nur eine Näherung des größen Eigenwerts von A, und zum anderen ist die Konvergenzgeschwindigkeit langsam, falls $|\lambda_2| \approx |\lambda_1|$.

Beides können wir zumindest abmilden, wenn eine Näherung ρ an einen gesuchten Eigenwert bekannt ist. Sei dazu (λ, v) Eigenpaar des verallgemeinerten Eigenwertproblems $Av = \lambda Bv$ und $\rho \in \mathbb{C}$ kein Eigenwert. Dann gilt für $\lambda = \frac{1}{\mu} + \rho$

$$Av = \lambda Bv = \left(\frac{1}{\mu} + \rho\right) Bv \quad \Leftrightarrow (A - \rho B)^{-1} Bv = \mu v,$$

d.h. (μ, v) ist Eigenpaar des Eigenwertproblems mit der Matrix $(A - \rho B)^{-1}B$. Gilt $|\rho - \lambda| \ll |\rho - \lambda_j|$ für die Eigenwerte $\lambda_j \neq \lambda$ so konvergiert Alg. 9.1 mit der Matrix $(A-\rho B)^{-1}B$ linear mit dem Faktor $\max_{j=1,\lambda_j\neq\lambda}^n \frac{|\rho-\lambda|}{|\rho-\lambda_j|} \ll 1$ gegen μ. Den gewünschten Eigenwert erhält man dann durch $\lambda = \frac{1}{\mu} + \rho$.

In Alg. 9.11 ist dieses Vorgehen zusammengefasst. Da die Berechnung der Inversen von $A - \rho B$ sehr aufwendig ist, wird man üblicherweise zunächst eine LU-Zerlegung durchführen (Aufwand $\mathcal{O}(n^3)$) und dann Vorwärts- und Rückwärtssubstitution (Aufwand $\mathcal{O}(n^2)$) verwenden.

9.3 QR-Verfahren

9.3.1 Basisalgorithmus und Konvergenzbeweis

Algorithmus 9.12 QR-Verfahren: Basisalgorithmus

Input: $A \in \mathbb{C}^{n\times n}$ mit n Eigenwerten mit paarweise unterschiedlichem Betrag

1: $A^{(0)} = A$
2: **for** $t = 1, \ldots$ **do**
3: Berechne QR-Zerlegung von $A^{(t-1)} = Q^{(t-1)}R^{(t-1)}$
4: $A^{(t)} = R^{(t-1)}Q^{(t-1)}$
5: **end for**

Output: Approximation an eine obere Dreiecksmatrix $A^{(t)}$ mit Approximationen an die Eigenwerte von A in der Hauptdiagonalen

Der Basisalgorithmus des QR-Verfahrens ist in Alg. 9.12 dargestellt. Zum Nachweis der Konvergenz benötigen wir zunächst zwei Hilfsresultate.

Lemma 9.13. *Die Matrizen $A^{(t)}$ des QR-Verfahrens sind paarweise ähnlich und besitzen dieselben Eigenwerte.*

Beweis. Per Induktion über $t \in \mathbb{N}_0$ wegen

$$A^{(t+1)} = \underbrace{{Q^{(t)}}^{-1}Q^{(t)}}_{=\mathrm{id}} R^{(t)}Q^{(t)} = {Q^{(t)}}^{-1}A^{(t)}Q^{(t)}.$$

□

Lemma 9.14. *Seien $A = Q_1R_1$ und $A = Q_2R_2$ zwei QR-Zerlegungen einer regulären Matrix $A \in \mathbb{C}^{n\times n}$. Dann existiert eine unitäre Diagonalmatrix $S \in \mathbb{C}^{n\times n}$ mit $Q_1 = Q_2S^*$ und $R_1 = SR_2$.*

Beweis. Da A und somit auch R_2 regulär und $Q_1^{-1} = Q_1^*$ ist, gilt

$$Q_1^*Q_2 = R_1R_2^{-1},$$

d.h. die obere Dreiecksmatrix $S := R_1R_2^{-1}$ ist unitär. Da die Spalten von S orthogonal zueinander sind, ist S eine Diagonalmatrix. □

Satz 9.15 (QR-Algorithmus). *Sei $A \in \mathbb{K}^{n\times n}$ diagonalisierbar mit Eigenwerten $\lambda_1, \dots, \lambda_n \in \mathbb{K}$ und $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0$. Weiter seien $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n) \in \mathbb{K}^{n\times n}$ und $V = (v_1, \dots, v_n) \in \mathbb{K}^{n\times n}$ die Matrix aus Eigenvektoren v_j zu den Eigenwerten λ_j für $j = 1, \dots, n$. Zudem existiere eine untere, normierte Dreiecksmatrix L und eine obere Dreiecksmatrix U sodass $V^{-1} = LU$.*

Dann konvergieren die Hauptdiagonaleneinträge der Matrizen $A^{(t)}$ aus Alg. 9.12 linear gegen die Eigenwerte von A.

Beweis. Wegen Lemma 9.13 und Bem. 9.4 genügt es zu zeigen, dass $A^{(t)}$ im gewissen Sinne gegen eine obere Dreiecksmatrix konvergiert. Der Beweis erfolgt in mehreren Schritten.

1. QR-Zerlegung von A^t: Wir unterscheiden die Matrizen $A^{(t)}$ aus dem QR-Verfahren und die t-te Potenz A^t. Für A^t gilt mit $V^{-1} = LU$

$$A^t = \left(V\Lambda V^{-1}\right)^t = V\Lambda^t V^{-1} = V\Lambda^t LU = \underbrace{V\Lambda^t L\Lambda^{-t}}_{=:V_t}\Lambda^t U = V_t\Lambda^t U.$$

Da die Eigenwerte nicht verschwinden, ist Λ regulär und somit V_t wohldefiniert und regulär. Sei $V_t = \tilde{Q}_t\tilde{R}_t$ eine QR-Zerlegung von V_t. Dann ist auch $\tilde{R}_t$ regulär, $\tilde{R}_t\Lambda^t U$ eine obere Dreiecksmatrix und

$$A^t = \tilde{Q}_t\left(\tilde{R}_t\Lambda^t U\right) \tag{9.8}$$

eine QR-Zerlegung von A^t. Insbesondere ist A^t regulär.

2. QR-Zerlegung von A^t: Sei $Q_t := Q^{(0)}\cdots Q^{(t-1)}$ und $R_t := R^{(t-1)}\cdots R^{(0)}$. Dann folgt aus Lemma 9.13 induktiv $A^{(t)} = Q_t^* A Q_t$ und

$$A^t = Q_t R_t \tag{9.9}$$

ist eine weitere QR-Zerlegung von A^t, da

$$Q_t R_t = \underbrace{Q^{(0)}\cdots Q^{(t-2)}}_{=Q_{t-1}}\ \underbrace{Q^{(t-1)}R^{(t-1)}}_{=A^{(t-1)}=Q_{t-1}^*AQ_{t-1}}\ \underbrace{R^{(t-2)}\cdots R^{(0)}}_{=R_{t-1}} = A\ \underbrace{Q_{t-1}R_{t-1}}_{=A^{t-1}\text{ per Induktion}} = A^t.$$

Darstellung von $A^{(t)}$: Mit Hilfe von Lemma 9.14 existieren unitäre Diagonalmatrizen S_t mit $Q_t = \tilde{Q}_t S_t^*$ und $R_t = S_t\tilde{R}_t\Lambda^t U$, da A^t regulär ist. Es folgt

$$\begin{aligned} Q^{(t)} &= Q_t^* Q_{t+1} = S_t\tilde{Q}_t^*\tilde{Q}_{t+1}S_{t+1}^*, \\ R^{(t)} &= R_{t+1}R_t^{-1} = \left(S_{t+1}\tilde{R}_{t+1}\Lambda^{t+1}U\right)\left(U^{-1}\Lambda^{-t}\tilde{R}_t^{-1}S_t^*\right) = S_{t+1}\tilde{R}_{t+1}\Lambda\tilde{R}_t^{-1}S_t^* \end{aligned}$$

und daraus

$$\begin{aligned} A^{(t)} &= Q^{(t)}R^{(t)} = S_t\tilde{Q}_t^*\tilde{Q}_{t+1}\underbrace{S_{t+1}^* S_{t+1}}_{=\mathrm{id}}\tilde{R}_{t+1}\Lambda\tilde{R}_t^{-1}S_t^* \\ &= S_t\left(\tilde{R}_t\tilde{R}_t^{-1}\right)\tilde{Q}_t^*\underbrace{\tilde{Q}_{t+1}\tilde{R}_{t+1}}_{=V_{t+1}}\Lambda\tilde{R}_t^{-1}S_t^* \\ &= S_t\tilde{R}_t\underbrace{(\tilde{Q}_t\tilde{R}_t)^{-1}}_{=V_t^{-1}}V_{t+1}\Lambda\tilde{R}_t^{-1}S_t^* = S_t\tilde{R}_t V_t^{-1}V_{t+1}\Lambda\tilde{R}_t^{-1}S_t^*. \end{aligned} \tag{9.10}$$

Asymptotik von V_t: Nach Definition gilt $V_t = V\Lambda^t L\Lambda^{-t}$ mit der normierten unteren Dreiecksmatrix L und der Diagonalmatrix Λ, bei der die Diagonaleinträge der Größe des Betrages nach fallend geordnet sind. Somit gilt

$$\Lambda^t L\Lambda^{-t} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21}\left(\frac{\lambda_2}{\lambda_1}\right)^t & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1}\left(\frac{\lambda_n}{\lambda_1}\right)^t & \cdots & l_{n(n-1)}\left(\frac{\lambda_n}{\lambda_{n-1}}\right)^t & 1 \end{pmatrix}, \tag{9.11}$$

d.h. $\Lambda^t L\Lambda^{-t} = \mathrm{id} + E_t$ mit $\|E_t\| = \mathcal{O}(q^t)$ für $t \to \infty$ und einem $q < 1$. Entsprechend folgt

$$V_t = V + VE_t \qquad \text{mit} \quad \|VE_t\| = \mathcal{O}(q^t), \qquad t \to \infty.$$

Asymptotik von $A^{(t)}$: Mit Hilfe der Asymptotik von V_t erhält man

$$V_t^{-1}V_{t+1} = (V + VE_t)^{-1}(V + VE_{t+1}) = \mathrm{id} + F_t \quad \text{mit} \quad \|F_t\| = \mathcal{O}(q^t), \quad t \to \infty.$$

Die Darstellung von $A^{(t)}$ aus (9.10) wird damit zu

$$A^{(t)} = S_t\tilde{R}_t\Lambda\tilde{R}_t^{-1}S_t^* + S_t\tilde{R}_tF_t\Lambda\tilde{R}_t^{-1}S_t^*.$$

Der zweite Term konvergiert gegen Null, da wegen $\|V_t\| = \|\tilde{Q}_t\tilde{R}_t\| = \|\tilde{R}_t\|$ und $\|V_t^{-1}\| = \|\tilde{R}_t^{-1}\|$ gilt

$$\left\|S_t\tilde{R}_tF_t\Lambda\tilde{R}_t^{-1}S_t^*\right\| \le |\lambda_1|\,\mathrm{cond}(V_t)\|F_t\| = \mathcal{O}(q^t), \qquad t \to \infty.$$

Da die Hauptdiagonaleneinträge der oberen Dreiecksmatrix $S_t\tilde{R}_t\Lambda\tilde{R}_t^{-1}S_t^*$ von t unabhängig und gleich den Eigenwerten λ_j von A sind, konvergieren die Hauptdiagonaleneinträge von $A^{(t)}$ linear gegen $\lambda_1, \ldots, \lambda_n$ und die Behauptung ist gezeigt. □

Die Voraussetzungen des vorigen Beweises sind zum Teil nicht notwendig, wie folgende Überlegungen zeigen:

1. Die Voraussetzung, dass V^{-1} eine LU-Zerlegung besitzt, ist lediglich für diese Version des Konvergenzbeweises des QR-Verfahrens notwendig. V ist invertierbar, sodass immer eine Permutationsmatrix P existiert mit $PV^{-1} = LU$.

2. Wenn $A \in \mathbb{R}^{n\times n}$ so wird gefordert, dass alle Eigenwerte ebenfalls reell sind. Dies ist im Allgemeinen nicht der Fall. Zudem haben konjugiert komplexe Eigenwerte den gleichen Betrag und verstoßen damit gegen die Bedingung $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Da das asymptotische Verhalten von $A^{(t)}$ auf der Gleichung (9.11) beruht, werden die Matrizen $A^{(t)}$ z.B. im Falle $n = 6$ und $|\lambda_1| > |\lambda_2| > |\lambda_3| = \cdots = |\lambda_5| > |\lambda_6|$ gegen eine eine Matrix folgender Form konvergieren:

$$\begin{pmatrix} \lambda_1 & \star & \cdots & & & \star \\ 0 & \lambda_2 & \star & \cdots & & \star \\ 0 & 0 & D_{11} & D_{12} & D_{13} & \star \\ 0 & 0 & D_{21} & D_{22} & D_{23} & \star \\ 0 & 0 & D_{31} & D_{32} & D_{33} & \star \\ 0 & 0 & 0 & 0 & 0 & \lambda_6 \end{pmatrix}.$$

Dies ist eine Block-Dreiecksmatrix mit Diagonalblöcken in der Größe der Anzahl der betragsgleichen Eigenwerte. Die Eigenwerte von A können dann aus den Eigenwerten dieser Diagonalblöcke berechnet werden.

3. Die Voraussetzung $|\lambda_n| > 0$ ist nicht einschränkend. Falls nötig, kann wie im vorigen Abschnitt an Stelle der Matrix A der QR-Algorithmus auf die Matrix $A - \rho\,\mathrm{id}$ mit einem geeigneten shift-Parameter ρ angewendet werden.

Die letzte Überlegung kann auch zur Beschleunigung der Konvergenz eingesetzt werden. Diese hängt maßgeblich von den Quotienten der Eigenwerte $|\lambda_j/\lambda_{j+1}|$ ab. Durch geschickte Wahl des shiftes, kann dieser erheblich verkleinert werden. Betrachen wir dazu wieder (9.11) und darin die letzte Zeile. Da $\left(A^{(t)}\right)_{nn}$ eine Näherung an den betragskleinsten Eigenwert von A darstellt, könnten wir in jedem Schritt $\rho^{(t)} = \left(A^{(t)}\right)_{nn}$ wählen und erhalten so eine sehr schnelle Konvergenz der letzten Zeile von $A^{(t)}$ gegen $(0, \ldots, 0, \lambda_n)$, da die Quotienten $|\frac{\lambda_n - \rho^{(t)}}{\lambda_j - \rho^{(t)}}|$ für $j = 1, \ldots, n-1$ sehr klein sind. Sobald der mutmaßlich größte Eintrag $\left(A^{(t)}\right)_{n(n-1)}$ in der letzten Zeile von $A^{(t)}$ unter einer vorgegebenen Toleranz ist, können wir eine Zeile nach oben rücken und $\rho^{(t)} = \left(A^{(t)}\right)_{(n-1)(n-1)}$ wählen.

In Alg. 9.16 ist dieses Vorgehen mit einer kleinen Änderung zur Wahl des shifts beschrieben. Für diese ist die Konvergenzbeschleunigung noch größer als bei der skizzierten Wahl des shifts.

Algorithmus 9.16 QR-Verfahren mit shifts

Input: $A \in \mathbb{C}^{n\times n}$ mit n Eigenwerten mit paarweise unterschiedlichem Betrag, **TOL** > 0 vorgegebene Toleranz

for $j = n, n-1, \ldots, 2$ **do**
  while $|A(j-1,j)| >$ **TOL** $(|A(j-1,j-1)| + |A(j,j)|)$ **do**
    Berechne Eigenwerte ρ_1, ρ_2 von $\begin{pmatrix} A(j-1,j-1) & A(j-1,j) \\ A(j,j-1) & A(j,j) \end{pmatrix}$
    if $|\rho_1 - A(j,j)| < |\rho_2 - A(j,j)|$ **then**
      $\rho = \rho_1$
    else
      $\rho = \rho_2$
    end if
    Berechne QR-Zerlegung von $A - \rho\,\mathrm{id} = QR$
    $A = RQ + \rho\,\mathrm{id}$
  end while
end for

Output: A wird mit einer unitär äquivalenten oberen Dreiecksmatrix überschrieben

Aufwand: Der Hauptaufwand des QR-Verfahrens ist die Berechnung der QR-Zerlegung in jedem Schritt. Für eine beliebige Matrix $A \in \mathbb{K}^{n\times n}$ benötigt diese $\mathcal{O}(n^3)$ Rechenoperationen und ist somit sehr aufwendig. Daher ist es wesentlich sinnvoller, zunächst die Matrix A so zu modifizieren, dass eine QR-Zerlegung günstiger gewonnen werden kann.

9.3.2 Reduktionsmethoden

Nach Satz 9.3 besitzen ähnliche Matrizen die gleichen Eigenwerte. Eine Möglichkeit zur Bestimmung der Eigenwerte einer Matrix $A \in \mathbb{K}^{n\times n}$ besteht somit darin, eine reguläre Matrix Q zu finden, sodass die Eigenwerte von $Q^{-1}AQ$ einfach(er) zu bestimmen sind. Ist $Q^{-1}AQ$ eine Dreiecksmatrix (z.B. die Schur-Zerlegung nach Satz 5.9(1)), so können wegen Bem. 9.4 die Eigenwerte direkt abgelesen werden. Leider benötigt die Konstruktion einer Schur-Zerlegung bereits die Eigenvektoren von A wie der folgende Beweis nahelegt.

Satz 9.17 (Schur-Zerlegung)**.** *Zu jeder Matrix $A \in \mathbb{C}^{n\times n}$ existiert eine unitäre Matrix $Q \in \mathbb{C}^{n\times n}$ und eine obere Dreiecksmatrix U, so dass*

$$Q^*AQ = U. \tag{9.12}$$

Beweis. Wir führen einen Induktionsbeweis nach n. Für $n = 1$ können wir $Q = 1$ wählen. Angenommen, die Behauptung sei wahr für Matrizen der Dimension $n-1$. Sei u ein Eigenvektor von A zum Eigenwert λ mit $\|u\|_2 = 1$. (Da Eigenwerte die Nullstellen des charakteristischen Polynoms sind, folgt die Existenz eines Eigenwertes folgt aus dem Fundamentalsatz der Algebra.) Wir ergänzen u zu einer Orthonormalbasis $\{u, v_2, \dots, v_n\}$ von $\mathbb{C}^n$. Dann ist die Matrix

$$Q := (u, v_2, \dots, v_n)$$

unitär. Da $u^*v_j = 0$ für $j = 2, \dots, n$, folgt

$$Q^*AQ = Q^*(\lambda u, Av_2, \dots, Av_n) = \begin{pmatrix} \lambda & x_{n-1}^* \\ 0 & A_{n-1} \end{pmatrix}$$

für eine Matrix $A_{n-1} \in \mathbb{C}^{(n-1)\times(n-1)}$ und einen Vektor $x_{n-1}^* \in \mathbb{C}^{n-1}$. Nach Induktionsannahme existiert eine unitäre Matrix $Q_{n-1} \in \mathbb{C}^{(n-1)\times(n-1)}$, so dass $U_{n-1} := Q_{n-1}^*A_{n-1}Q_{n-1}$ eine obere Dreiecksmatrix ist. Setzen wir

$$Q_n := Q \begin{pmatrix} 1 & 0 \\ 0 & Q_{n-1} \end{pmatrix},$$

so ist Q_n unitär und

$$Q_n^*A_nQ_n = \begin{pmatrix} 1 & 0 \\ 0 & Q_{n-1}^* \end{pmatrix} \begin{pmatrix} \lambda & x_{n-1}^* \\ 0 & A_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & Q_{n-1} \end{pmatrix} = \begin{pmatrix} \lambda & x_{n-1}^*Q_{n-1} \\ 0 & U_{n-1} \end{pmatrix}$$

ist eine obere Dreiecksmatrix. □

Die Schur-Zerlegung ist zur Bestimmung der Eigenwerte von A somit wenig hilfreich. Das Gleiche gilt für eine Transformation auf Jordan-Normalenform.

Definition 9.18 (Hessenberg-Matrix)**.** *Eine Matrix $A = (a_{ij})_{i,j=1}^n \in \mathbb{K}^{n\times n}$ heißt (obere)* Hessenberg-Matrix*, wenn gilt $a_{ij} = 0$ für alle $j = 1, \dots, n$ und $i = j+2, \dots, n$.*

Satz 9.19. *Zu jeder Matrix $A \in \mathbb{K}^{n\times n}$ existiert eine Folge von Householder-Matrizen Q_i mit $i = 1, \dots, n-2$ (siehe Abschnitt 6.3.3), sodass mit $Q := Q_{n-2} \cdot \dots \cdot Q_1$ die Matrix QAQ^* eine Hessenberg-Matrix ist. Für hermitesche Matrizen A ist QAQ^* eine Tridiagonalmatrix.*

Beweis. Ist A hermitesch, so auch QAQ^* und die letzte Aussage folgt direkt, wenn QAQ^* eine Hessenberg-Matrix ist. Dazu sei $A \in \mathbb{K}^{n\times n}$ beliebig. Analog zu Abschnitt 6.3.4 konstruieren wir eine Folge von Householder-Matrizen Q_k, sodass

$$Q_{k-1} \cdots Q_1 A Q_1 \cdots Q_{k-1} = \left(\begin{array}{c|c} H_k & B_k \\ \hline \mathbf{0}_{n-k,k-1} \;\; \tilde{a}_k^{(k)} & C_k \end{array}\right) \tag{9.13}$$

mit einer Hessenberg-Matrix $H_k \in \mathbb{K}^{k\times k}$, einem reduzierten Spaltenvektor $\tilde{a}_k^{(k)} \in \mathbb{K}^{n-k}$ und Matrizen $B_k \in \mathbb{K}^{k\times(n-k)}$ und $C_k \in \mathbb{K}^{(n-k)\times(n-k)}$. Hierbei ist zu beachten, dass wegen Lemma 6.29 $Q_k^{-1} = Q_k^* = Q_k$ gilt und das im Gegensatz zur QR-Zerlegung A von beiden Seiten mit den Householder-Matrizen multipliziert werden muss, um Ähnlichkeitstransformationen zu erhalten.

Seien also $Q_1, \dots, Q_{k-1}$ so konstruiert, dass (9.13) gilt und $k < n-1$ (andernfalls ist $Q_{k-1} \cdots Q_1 A Q_1 \cdots Q_{k-1}$ bereits eine Hessenberg-Matrix). Falls $\tilde{a}_k^{(k)} \neq 0$ wählen wir mit Hilfe von Lemma 6.30 eine Householder-Matrix $\tilde{Q}_k \in \mathbb{K}^{(n-k)\times(n-k)}$ mit $\tilde{Q}_k \tilde{a}_k^{(k)} \in \operatorname{span}(e_1)$. Im Falle $\tilde{a}_k^{(k)} = 0$ sei $\tilde{Q}_k = I_{(n-k)\times(n-k)}$. Mit $Q_k := \left(\begin{smallmatrix} I_{k\times k} & \mathbf{0} \\ \mathbf{0} & \tilde{Q}_k \end{smallmatrix}\right)$ folgt

$$Q_k Q_{k-1} \cdots Q_1 A Q_1 \cdots Q_{k-1} Q_k = \left(\begin{array}{c|c} H_k & B_k \tilde{Q}_k \\ \hline \mathbf{0}_{n-k,k-1} \;\; \tilde{Q}_k \tilde{a}_k^{(k)} & \tilde{Q}_k C_k \tilde{Q}_k \end{array}\right),$$

d.h. $Q_k Q_{k-1} \cdots Q_1 A Q_1 \cdots Q_{k-1} Q_k$ ist von der Form (9.13) mit $k+1$ an Stelle von k. Per Induktion folgt somit die Behauptung. □

Der Beweis ist konstruktiv. Analog zu Alg. 6.33 können wir eine Matrix-freie Version des Algorithmus formulieren (siehe Alg 9.20). Wir erhalten dabei in etwa

$$\begin{aligned} \sum_{k=1}^{n-2} \left(\sum_{j=1}^{n} \sum_{i=1}^{n-k} 2 + \sum_{j=k+1}^{n} \sum_{i=1}^{n-k} 2 \right) &= \sum_{k=1}^{n-2} \left(2n(n-k) + 2(n-k)^2\right) \\ &= \sum_{l=2}^{n-1} \left(2nl + 2l^2\right) \approx \frac{2}{3}n^3 + n^3 = \frac{5}{3}n^3 \end{aligned}$$

Rechenoperationen.

Das QR-Verfahren aus Alg. 9.16 erhält die Hessenberg-Struktur, d.h. die Reduktion auf Hessenberg-Form muss nur einmal durchgeführt werden.

Lemma 9.21. *Sei $A \in \mathbb{K}^{n\times n}$ eine reguläre Hessenberg-Matrix und $A = QR$ eine QR-Zerlegung. Dann sind Q und RQ Hessenberg-Matrizen.*

Algorithmus 9.20 Reduktion auf Hessenberg-Matrix

Input: $A = (a_{ij})_{i,j=1}^{n} \in \mathbb{K}^{n\times n}$

for $k = 1, \dots, n-2$ **do**
  $\alpha := \sqrt{\sum_{j=k+1}^{n} |a_{jk}|^2}$
  if $\alpha \neq 0$ **then**
    if $|a_{k+1,k}| \neq 0$ **then**
      $\gamma = a_{k+1,k}/|a_{k+1,k}|$
    else
      $\gamma = 1$
    end if
    $u = (a_{k+1,k} + \gamma\alpha, a_{k+2,k}, \dots, a_{n,k})^\top$
    $\beta = 1/(\alpha(\alpha + |a_{k+1,k}|))$
    $a_{k+1,k} = -\gamma\alpha$
    for $j = k+2, \dots, n$ **do**
      $a_{j,k} = 0$
    end for
    for $j = 1, \dots, n$ **do**
      $s = \sum_{i=1}^{n-k} a_{j,k+i} u_i$
      for $i = 1, \dots, n-k$ **do**
        $a_{j,k+i} = a_{j,k+i} - \beta s \overline{u_i}$
      end for
    end for
    for $j = k+1, \dots, n$ **do**
      $s = \sum_{i=1}^{n-k} a_{k+i,j} \overline{u_i}$
      for $i = 1, \dots, n-k$ **do**
        $a_{k+i,j} = a_{k+i,j} - \beta s u_i$
      end for
    end for
  end if
end for

Output: A ist mit einer ähnlichen Hessenberg-Matrix überschrieben

Beweis. Da Q unitär ist, ist die obere Dreiecksmatrix R invertierbar und $Q = AR^{-1} = (q_{jk})_{j,k=1}^{n}$ ist eine Hessenberg-Matrix: Sei $A = (a_{jk})_{j,k=1}^{n}$ mit $a_{jk} = 0$ für $j > k+1$ und $R^{-1} = (\tilde{r}_{jk})_{j,k=1}^{n}$ mit $\tilde{r}_{jk} = 0$ für $j > k$. Dann gilt

$$q_{jk} = \sum_{l=1}^{n} a_{jl}\tilde{r}_{lk} = \sum_{l=j-1}^{k} a_{jl}\tilde{r}_{lk}, \qquad j,k = 1, \dots, n,$$

d.h. $q_{jk} = 0$ für $j > k+1$. Analog folgt die Hessenberg-Form von RQ. □

9.3.3 Givens-Rotationen zur Berechnung der QR-Zerlegung einer Hessenberg Matrix

Die QR-Zerlegung einer Hessenberg-Matrix kann erheblich schneller durchgeführt werden, wenn statt der Householder-Matrizen andere unitäre Matrizen verwendet werden.

Definition 9.22. *Eine $n \times n$-Matrix der Form*

$$G(j,k,c,s) = \left(\begin{array}{c|c|c} \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} & & \\ \hline & \begin{matrix} \overline{c} & & \overline{s} \\ & 1 & \\ & & \ddots \end{matrix} & \\ & \begin{matrix} & & 1 \\ -s & & c \end{matrix} & \\ \hline & & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} \end{array}\right) \tag{9.14}$$

heißt Givens-Rotation*, falls $|c|^2 + |s|^2 = 1$. Dabei grenzen die durchgezogenen Linien den Bereich von der j-ten bis zur k-ten Zeile, bzw. Spalte ab.*

Givens-Rotationen sind unitär, da

$$\begin{pmatrix} \overline{c} & \overline{s} \\ -s & c \end{pmatrix} \begin{pmatrix} c & -\overline{s} \\ s & \overline{c} \end{pmatrix} = \begin{pmatrix} |c|^2 + |s|^2 & 0 \\ 0 & |c|^2 + |s|^2 \end{pmatrix}.$$

Der wichtigste Fall ist, dass c und s reell sind. In diesem Fall gibt es ein $\theta \in [0, 2\pi)$ mit $c = \cos\theta$ und $s = \sin\theta$, und die Matrix $\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$ beschreibt eine Drehung in der Ebene $\mathbb{R}^2$ um den Winkel θ.

Offenbar bewirkt eine Multiplikation GA einer Givens-Rotation $G = G(j,k,c,s)$ von links an eine Matrix $A \in \mathbb{K}^{n\times n}$ eine Ersetzung der j-ten und k-ten Zeile a_j^*, bzw. a_k^* von A durch $\overline{c}a_j^* + \overline{s}a_k^*$, bzw. $-sa_j^* + ca_k^*$.

Die Bedeutung der Givens-Rotationen liegt darin, dass man zu gebenen Indizes j, k, l mit $j \neq k$ und gegebener Matrix $A \in \mathbb{K}^{m\times n}$ eine Givens-Rotation $G(j,k,c,s)$ der Form (9.14) finden kann, so dass $(GA)_{k,l} = 0$. Dazu ist die Gleichung $-sa_{jl} + ca_{kl} = 0$ unter der Nebenbedingung $|c|^2 + |s|^2 = 1$ zu lösen. Dieses Problem besitzt zwei Lösungen, die sich durch das Vorzeichen unterscheiden. Eine der Lösungen lautet

$$\begin{pmatrix} c \\ s \end{pmatrix} = \mathbf{rot}(a_{jl}, a_{kl}) := \frac{1}{\sqrt{|a_{jl}|^2 + |a_{kl}|^2}} \begin{pmatrix} a_{jl} \\ a_{kl} \end{pmatrix}. \tag{9.15}$$

Um einen over- oder underflow zu vermeiden, benutzt man üblicherweise die äquivalenten Formeln

$$\begin{aligned} c &= \frac{a_{jl}/|a_{jl}|}{\sqrt{1+|t|^2}}, \quad s = \frac{t}{\sqrt{1+|t|^2}}, \quad t = \frac{a_{kl}}{|a_{jl}|}, && \text{für } |a_{jl}| \geq |a_{kl}|, \\ c &= \frac{t}{\sqrt{1+|t|^2}}, \quad s = \frac{a_{kl}/|a_{kl}|}{\sqrt{1+|t|^2}}, \quad t = \frac{a_{jl}}{|a_{kl}|}, && \text{für } |a_{jl}| < |a_{kl}|. \end{aligned}$$

Um eine QR-Zerlegung einer Hessenberg-Matrix $A \in \mathbb{K}^{n\times n}$ zu berechnen, definieren wir nun

$$A^{(k)} := G(k, k+1, c_k, s_k)A^{(k-1)} \qquad \text{mit} \qquad (c_k, s_k)^\top = \mathbf{rot}(A^{(k-1)}_{k,k}, A^{(k-1)}_{k+1,k})$$

für $k = 1, \ldots, n-1$ und $A^{(0)} := A$. Dieses Vorgehen veranschaulichen wir für $n = 4$ in folgendem Diagramm, indem wir die Matrixelemente, die in einem Schritt verändert werden, mit $*$ kennzeichnen und die übrigen von Null verschiedenen Elemente mit $+$:

$$A = \begin{pmatrix} ++++ \\ ++++ \\ \;+++ \\ \;\;++ \end{pmatrix} \xrightarrow{k=1} \begin{pmatrix} * & * & * & * \\ & * & * & * \\ & + & + & + \\ & & + & + \end{pmatrix} \xrightarrow{k=2} \begin{pmatrix} ++++ \\ \;\;*** \\ \;\;\;** \\ \;\;\;++ \end{pmatrix} \xrightarrow{k=3} \begin{pmatrix} ++++ \\ \;\;+++ \\ \;\;\;** \\ \;\;\;\;* \end{pmatrix} = R$$

In Alg. 9.23 ist dieses Vorgehen zusammengefasst. Die Anzahl der Multiplikationen zur Durchführung des Verfahrens beträgt etwa

$$\sum_{k=1}^{n-1} 4(n-k+1) = \sum_{k=2}^{n} 4k \approx 2n^2$$

und ist damit um einen Faktor $n/3$ geringer als bei einer QR-Zerlegung mit Householder-Matrizen.

Algorithmus 9.23 QR-Zerlegung einer Hessenberg-Matrix mit Givens-Rotationen

Input: $A = (a_{jk}) \in \mathbb{K}^{n\times n}$ Hessenberg-Matrix

1: **for** $k = 1, \ldots, n-1$ **do**
2: $\quad (c_k, s_k)^\top := \mathbf{rot}(a_{kk}, a_{k+1,k})$
3: $\quad$ **for** $l = l, \ldots, n$ **do**
4: $\quad\quad \begin{pmatrix} a_{k,l} \\ a_{k+1,l} \end{pmatrix} := \begin{pmatrix} \overline{c_k} & \overline{s_k} \\ -s_k & c_k \end{pmatrix} \begin{pmatrix} a_{k,l} \\ a_{k+1,l} \end{pmatrix}$
5: $\quad$ **end for**
6: **end for**

Output: A wird überschrieben mit der oberen Dreiecksmatrix $R = G(n-1, n, c_{n-1}, s_{n-1}) \cdots G(1, 2, c_1, s_1)A$

9.4 Lanczos-Verfahren

Das QR-Verfahren berechnet sämtliche Eigenwerte einer Matrix und benötigt dafür einen Aufwand, der kubisch mit der Größe der Matrix wächst. In vielen Anwendungen, insbesondere bei Eigenwertproblemen für gewöhnliche und partielle Differentialoperatoren, ist man jedoch häufig nur an einigen wenigen extremalen Eigenwerten

interessiert, und die auftretenden Matrizen sind sehr groß, aber nur dünn besetzt. In vielen Fällen ist die Dimension der Matrizen so groß, dass das QR-Verfahren nicht mehr mit den verfügbaren Rechenkapazitäten durchführbar ist.

Daher ist man an Verfahren interessiert, die nur Matrix-Vektor-Multiplikationen benötigen. Mit der Vektoriteration (siehe Abschnitt 9.2) haben wir bereits ein solches Verfahren kennengelernt, allerdings konnten wir damit nur eine Näherung an einen Eigenwert berechnen. Mit dem von dem ungarischen Mathematiker Lanczos (gesprochen *Lanzosch*) im Jahre 1950 vorgeschlagenen Verfahren, das wir im folgenden kennenlernen werden, können auch weitere Eigenwerte berechnet werden, und zudem wird die Näherung des größten Eigenwertes aus der Vektor–Iteration verbessert.

9.4.1 Idee

Im folgenden sei $A \in \mathbb{K}^{n\times n}$ eine hermitesche Matrix mit Eigenwerten $\lambda_j = \lambda_j(A)$, $j = 1, \dots, n$, die in absteigender Größe sortiert seien. Bei der Vektor–Iteration werden Vektoren $x^{(t)} := A^t x^{(0)}$, $t = 0, \dots$ (oder Vielfache hiervon) berechnet und als Näherung des größten Eigenwertes λ_1 von A wegen Satz 9.10 die Zahlen

$$\mu^{(t+1)} := \frac{x^{(t)^*} A x^{(t)}}{x^{(t)^*} x^{(t)}}, \qquad t = 0, 1, \dots$$

verwendet. Wegen des Satzes von Rayleigh 9.8 erhalten wir somit $\mu^{(t+1)} \leq \lambda_1$ für alle $t = 0, 1, \dots$.

Beim Lanczos-Verfahren wird als Näherung von λ_1 das Maximum des Rayleigh-Quotienten über dem Krylov-Raum (siehe Def. 8.14)

$$\mathcal{K}_t = \mathcal{K}_k(A, x_0) = \operatorname{span}\{x_0, x_1, \dots, x_{t-1}\}$$

verwendet:

$$\mu_1^{(t)} := \max_{x\in\mathcal{K}_t\setminus\{0\}} \frac{x^* A x}{x^* x}, \qquad t = 1, 2, \dots.$$

Da $\operatorname{span}\{x_{t-1}\} \subset \mathcal{K}_t \subset \mathbb{K}^n$, gilt

$$\mu^{(t)} \leq \mu_1^{(t)} \leq \lambda_1.$$

$\mu_1^{(t)}$ ist also eine mindestens ebenso gute Approximation von λ_1 wie $\mu^{(t)}$.

Wir können die Lanczos-Approximationen $\mu_1^{(t)}$ des größten Eigenwertes von A auch anders interpretieren, indem wir die Einschränkungen der durch A gegebenen linearen Abbildung auf den t-ten Krylov-Raum betrachten:

$$\begin{aligned} \mathcal{A}_t : \quad & \mathcal{K}_t \to \mathcal{K}_t \\ & x \mapsto P_t A x. \end{aligned}$$

Dabei sei $P_t \in \mathbb{K}^{n\times n}$ die orthogonale Projektion auf $\mathcal{K}_t$, also die eindeutig bestimmte hermitesche Matrix mit $P_t^2 = P$ und $\mathcal{R}(P_t) = \mathcal{K}_t$. $\mathcal{A}_t$ ist bezüglich des inneren

Produktes $(x, y) := x^* y$ selbstadjungiert, da $(\mathcal{A}_t x, y) = (\mathcal{A}_t P_t x, y) = x^* P_t A P_t y = (x, \mathcal{A}_t y)$. Also besitzt die Abbildung $\mathcal{A}_t$ nur reelle Eigenwerte, die wir wie bei Matrizen der Größe nach anordnen: $\lambda_1(\mathcal{A}_t) \geq \lambda_2(\mathcal{A}_t) \geq \cdots \geq \lambda_t(\mathcal{A}_t)$. Es gilt

$$\lambda_1(\mathcal{A}_t) = \sup_{x \in \mathcal{K}_t \setminus \{0\}} \frac{(x, \mathcal{A}_t x)}{(x, x)} = \sup_{x \in \mathcal{K}_t \setminus \{0\}} \frac{x^* A x}{x^* x} = \mu_1^{(t)},$$

da $(x, \mathcal{A}_t x) = x^* P_t A x = x^* P_t^* A x = (P_t x)^* A x = x^* A x$ für $x \in \mathcal{K}_t$. Es liegt nun nahe, den Wert

$$\mu_j^{(t)} := \lambda_j(\mathcal{A}_t)$$

für $j \leq t$ als Approximationen des Eigenwertes $\lambda_j(A)$ zu betrachten.

9.4.2 Herleitung des Verfahrens

Wir wollen uns nun überlegen, wie wir die Zahlen $\mu_j^{(t)}$ berechnen können. Offenbar benötigen wir hierzu eine Orthonormalbasis $\{v_0, \ldots, v_{t-1}\}$ des Krylov-Raums $\mathcal{K}_t$ und die Matrix T_t, die die Abbildung $\mathcal{A}_t$ bezüglich dieser Basis repräsentiert. Die Matrixeinträge von T_t sind dann gegeben durch $(T_t)_{ij} = (v_i, \mathcal{A}_t v_j) = v_i^* A v_j$. Also ist

$$T_t = V_t^* A V_t \qquad \text{mit} \qquad V_t = (v_0\, v_1\, \cdots\, v_{t-1}) \in \mathbb{K}^{n \times t}. \tag{9.16}$$

Es gilt $\mu_j^{(t)} = \lambda_j(\mathcal{A}_t) = \lambda_j(T_t)$, und für $t \ll n$ können wir die Eigenwerte von T_t mit wenig Aufwand mit dem QR-Verfahren berechnen.

Eine Möglichkeit, eine Orthonormalbasis der Krylov-Räume $\mathcal{K}_t = \operatorname{span}\{x_0, \ldots, x_{t-1}\}$ zu bestimmen, wäre eine QR-Zerlegung der Matrix $(x_0 \ldots, x_{t-1})$ zu berechnen, etwa mit Hilfe von Householder-Transformationen. Eine wesentlich elegantere und effizientere Möglichkeit besteht darin, das CG-Verfahren Alg. 8.10 mit Startvektor $x^{(0)} := 0$ auf die Gleichung $Ax = x_0$ anzuwenden.

In Lemma 8.15 hatten wir gezeigt, dass die Residuen $r^{(0)}, \ldots, r^{(t-1)}$, die in diesem Algorithmus berechnet werden, eine orthogonale Basis des Krylov-Raum $\mathcal{K}_t$ bilden. Also erhalten wir durch die Normierung $v_j := r^{(j)} / \|r^{(j)}\|_2$, $j = 0, \ldots, t-1$ die gewünschte Orthonomalbasis von $\mathcal{K}_t$.

Es stellt sich heraus, dass die Matrix T_t sogar eine Tridiagonalmatrix ist (damit Hessenberg-Matrix). Außerdem lassen sich die Matrixeinträge von T_t sehr einfach aus Größen berechnen, die im CG-Verfahren auftauchen:

Lemma 9.24. *Sei A hermitesch und positiv definit, und sei T_t durch (9.16) gegeben mit $v_j := r^{(j)} / \|r^{(j)}\|_2$ und den Residuen $r^{(j)}$ aus (8.11). Dann ist die Matrix T_t tridiagonal, und es gilt*

$$T_t = \Delta_t^{-1} B_t^* \operatorname{diag}(d_0^* A d_0, \ldots, d_{t-1}^* A d_{t-1}) B_t \Delta_t^{-1} \tag{9.17}$$

mit

$$\Delta_t := \operatorname{diag}(\|r^{(0)}\|_2, \dots, \|r^{(t-1)}\|_2) \qquad \textit{und} \qquad B_t := \begin{pmatrix} 1 & \beta_0 & 0 & \cdots & 0 \\ 0 & 1 & \beta_1 & & \vdots \\ & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \beta_{t-2} \\ 0 & \cdots & & 0 & 1 \end{pmatrix}.$$

Beweis. Ist $R_t := (r^{(0)}, \cdots, r^{(t-1)})$ die Matrix der Residuen, so gilt $V_t = R_t \Delta_t^{-1}$ und deshalb

$$T_t = \Delta_t^{-1} R_t^* A R_t \Delta_t^{-1}.$$

Wir führen weiterhin die Matrix $D_t := (d_0 \cdots d_{t-1})$ der Suchrichtungen ein und erhalten wegen $r^{(0)} = d^{(0)}$ und (8.13) die Beziehung $R_t = D_t B_t$. Da die Suchrichtungen A-konjugiert sind, ist die Matrix $D_t^* A D_t = \operatorname{diag}(d_0^* A d_0, \dots, d_{t-1}^* A d_{t-1})$ diagonal. Deshalb gilt

$$R_t^* A R_t = B_t^* D_t^* A D_t B_t = B_t^* \operatorname{diag}(d_0^* A d_0, \dots, d_{t-1}^* A d_{t-1}) B_t,$$

und daraus folgt (9.17). □

Einziger Nachteil des CG-Verfahrens ist, dass es nur für positiv-definite Matrizen anwendbar ist. Andererseits sind die im CG-Verfahren berechneten Vektoren x_t in unserem Zusammenhang nicht von Interesse. Motiviert durch Lemma 9.24 versuchen wir daher nun unmittelbar, mit dem Ansatz

$$T_t = \begin{pmatrix} \gamma_0 & \delta_0 & 0 & \cdots & 0 \\ \delta_0 & \gamma_1 & \delta_1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \delta_{t-2} \\ 0 & \cdots & 0 & \delta_{t-2} & \gamma_{t-1} \end{pmatrix} \tag{9.18}$$

Rekursionsformeln für die Vektoren v_j herzuleiten. Dazu schreiben wir die Gleichung $AV_t = V_t T_t$ mit $V_t := (v_0 \cdots v_{t-1})$ spaltenweise auf:

$$Av_0 = \gamma_0 v_0 + \delta_0 v_1, \tag{9.19a}$$
$$Av_j = \delta_{j-1} v_{j-1} + \gamma_j v_j + \delta_j v_{j+1}, \qquad j = 1, \dots, t-2. \tag{9.19b}$$

Da die Vektoren v_j orthogonal sein sollen, erhalten wir $\gamma_j = v_j^* A v_j$ für $j = 0, 1, 2, \dots$. Wir können die Vektoren v_j rekursiv berechnen, indem wir (9.19b) nach v_{j+1} (bzw. (9.19a) nach v_1) auflösen: Falls $w_j := (A - \gamma_j I) v_j - \delta_{j-1} v_{j-1}$ ($j \geq 1$), bzw. $w_0 := (A - \gamma_0 I) v_0$ von 0 verschieden ist, so können wir $v_{j+1} := w_j / \delta_j$ setzen mit $\delta_j := \|w_j\|_2$. Falls $w_j = 0$, so muss die Iteration abgebrochen werden. Wir werden sehen, dass wir in diesem Fall einen exakten invarianten Unterraum gefunden haben. Indem wir diese Gleichungen in geeigneter Reihenfolge aufschreiben, erhalten wir Algorithmus 9.25.

Algorithmus 9.25 Lanczos-Verfahren

Input: $A \in \mathbb{K}^{n\times n}$ hermitesche Matrix, zufälliger Startvektor $v_0 \in \mathbb{K}^n$ mit $\|v_0\|_2 = 1$

$\gamma_0 = v_0^* A v_0$
$w_0 = (A - \gamma_0 I)v_0$
$\delta_0 = \|w_0\|_2$
$j = 0$
while $\delta_j \neq 0$ **do**
    $v_{j+1} = w_j/\delta_j$
    $j = j + 1$
    $\gamma_j = v_j^* A v_j$
    $w_j = (A - \gamma_j I)v_j - \delta_{j-1} v_{j-1}$
    $\delta_j = \|w_j\|_2$
end while
Berechne Eigenwert-Zerlegung $T_j = S_j \Lambda_j S_j^*$ mit $T \in \mathbb{K}^{(j+1)\times(j+1)}$ aus (9.18)

Output: Die Diagonalelemente von Λ_j sind Approximation an Eigenwerte von A und $V_j S_j$ mit $V_j = (v_0, \ldots, v_j) \in \mathbb{K}^{n\times(j+1)}$ und $S_j \in \mathbb{K}^{(j+1)\times(j+1)}$ sind Approximationen der zugehörigen Eigenvektoren

9.4.3 Analysis

Satz 9.26. *Das Lanczos-Verfahren breche nach m Schritten ab. Dann bilden die Vektoren $v_0, \ldots, v_{t-1}$ eine Orthonormalbasis des Krylov-Raums $\mathcal{K}_t$ mit $t = 1, \ldots, m$, und es gilt*

$$AV_t = V_t T_t + w_{t-1} e_t^\top, \qquad t = 1, \ldots, m \tag{9.20}$$

mit $e_t = (0, \ldots, 0, 1)^\top \in \mathbb{R}^t$. Weiterhin ist $m \leq n$, und $\mathcal{K}_t$ ist ein invarianter Unterraum von A.

Beweis. Der Beweis erfolgt durch Induktion nach t. Für $t = 1$ lautet (9.20) einfach $Av_0 = v_0\gamma_0 + w_0$, und dies folgt aus den Definitionen. Angenommen wir hätten durch das Lanczos-Verfahren bereits Vektoren $v_0, \ldots, v_{t-1}$, $t \leq m$ bestimmt, die eine Orthonormalbasis von $\mathcal{K}_t$ bilden. Es ist leicht zu sehen, dass die durch den Algorithmus erzeugten Vektoren die Gleichungen (9.19) erfüllen, von denen ausgehend wir den Algorithmus hergeleitet haben. Daraus ergibt sich, spaltenweise gelesen, (9.20), wobei wir für die letzte Spalte die Definition von w_t benutzen. Multiplizieren wir (9.20) von links von V_t, so erhalten wir wegen $V_t^* V_t = I_t$ die Identität

$$V_t^* A V_t = T_t + V_t^* w_{t-1} e_t^\top.$$

Andererseits ist $V_t^* A V_t = T_t$, da $v_j^* A v_j = \gamma_j$ für $j = 0, \ldots, t-1$ und wegen (9.19b)

$$v_{j+1}^* A v_j = v_{j+1}^*(\delta_{j-1} v_{j-1} + \gamma_j v_j + \delta_j v_{j+1}) = v_{j+1}^*(\delta_j v_{j+1}) = \delta_j$$

für $j = 0, \ldots, t-2$. Ebenso zeigt man, dass $v_l A v_j = 0$ für $l \geq j + 2$. Daher gilt $V_t^* w_{t-1} = 0$. Ist $w_{t-1} \neq 0$, so folgt daraus wegen $v_t = w_{t-1}/\delta_{t-1}$, dass $\{v_0, \ldots, v_t\}$ ein Orthonomalsystem ist. Aus der Definition von v_t und der Induktionsvoraussetzung

folgt $\operatorname{span}\{v_0, \dots, v_t\} \subset \mathcal{K}_{t+1}$, und aus Dimensionsgründen gilt sogar Gleichheit. Ist $w_{t-1} = 0$, so gilt $AV_t = V_t T_t$. Da nach Induktionsvoraussetzung $\mathcal{R}(V_t) = \mathcal{K}_t$, folgt daraus $A\mathcal{K}_t \subset \mathcal{K}_t$, d.h. $\mathcal{K}_t$ ist ein invarianter Unterraum von A. □

Satz 9.27 (a-posteriori Fehlerschranken)**.** *Ist im t-ten Schritt des Lanczos-Verfahrens $S_t = (s_{jl})_{j,l=0,\dots,t-1}$, $\Lambda_t = \operatorname{diag}(\mu_0^{(t)}, \dots, \mu_{t-1}^{(t)})$ und $V_t S_t = Y_t = (y_0 \dots y_{t-1})$, so gilt*

$$\|Ay_j - \mu_j^{(t)} y_j\|_2 = |\delta_{t-1}||s_{t-1,j}|, \tag{9.21a}$$

$$\min_{l=0,\dots,n-1} |\mu_j^{(t)} - \lambda_l| \leq |\delta_{t-1}|\,|s_{t-1,j}| \tag{9.21b}$$

für $j = 0, \dots, t-1$.

Beweis. Multiplizieren wir (9.20) mit S_t von rechts, so erhalten wir

$$AY_t = Y_t \Lambda_t + w_{t-1} e_t^\top S_t,$$

also $Ay_j = \mu_j^{(t)} y_j + w_{t-1}(e_t^\top S_t e_{j+1})$ für $j = 0, \dots, t-1$. Da $\|w_{t-1}\| = |\delta_{t-1}|$, folgt hieraus (9.21a).

Ist $\mu_j^{(t)}$ ein Eigenwert von A, so ist die Ungleichung (9.21b) trivial. Anderenfalls ist die Matrix $A - \mu_j^{(t)} I$ invertierbar, und $(A - \mu_j^{(t)} I)^{-1}$ besitzt die Eigenwerte $1/(\lambda_l - \mu_j^{(t)})$, $l = 0, \dots, n-1$. Deshalb ist

$$\|(A - \mu_j^{(t)} I)^{-1}\|_2 = \max_{l=0,\dots,n-1} \frac{1}{|\lambda_l - \mu_j^{(t)}|}.$$

Folglich gilt für $z := Ay_j - \mu_j^{(t)} y_j$ die Ungleichung

$$\max_{l=0,\dots,n-1} \frac{1}{|\lambda_l - \mu_j^{(t)}|} \geq \frac{\|(A - \mu_j^{(t)} I)^{-1} z\|_2}{\|z\|_2} = \frac{\|y_j\|_2}{\|Ay_j - \mu_j^{(t)} y_j\|_2} = \frac{1}{|\delta_{t-1}||s_{t-1,j}|}.$$

Daraus folgt (9.21b). □

Aufgrund dieses Satzes ist es sinnvoll, die Lanczos-Iteration abzubrechen, sobald die Bedingung $|\delta_{t-1}||s_{t-1,j}| \leq \mathbf{TOL}$ erfüllt ist. Insbesondere folgt aus (9.21) für $\delta_t = 0$, dass die Vektoren $y_0, \dots, y_{t-1}$ exakte Eigenvektoren zu den Eigenwerten $\mu_j^{(t)}$ sind.