

# Numerik für Ingenieure

**Steffen Börm**

Stand 17. April 2020

Alle Rechte beim Autor.



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
1.1	Beispiel: Sortieren . . . . .	6
1.2	Beispiel: Schnelle Fourier-Transformation . . . . .	11
<b>2</b>	<b>Lineare Gleichungssysteme</b>	<b>15</b>
2.1	Beispiel: Differentialgleichung . . . . .	15
2.2	Allgemeine Aufgabenstellung . . . . .	17
2.3	Dreiecksmatrizen . . . . .	17
2.4	LR-Zerlegung . . . . .	20
2.5	Fehlerverstärkung . . . . .	27
2.6	QR-Zerlegung . . . . .	33
2.7	Ausgleichsprobleme . . . . .	42
<b>3</b>	<b>Nichtlineare Gleichungssysteme</b>	<b>47</b>
3.1	Beispiel: Lagrange-Punkte im Gravitationsfeld . . . . .	47
3.2	Bisektionsverfahren . . . . .	49
3.3	Allgemeine Fixpunktiterationen . . . . .	51
3.4	Newton-Verfahren . . . . .	56
<b>4</b>	<b>Eigenwertprobleme</b>	<b>63</b>
4.1	Beispiel: Schwingende Saite . . . . .	63
4.2	Vektoriteration . . . . .	64
4.3	Inverse Iteration . . . . .	69
4.4	Orthogonale Iteration . . . . .	73
4.5	QR-Iteration . . . . .	79
4.6	Praktische QR-Iteration* . . . . .	82
<b>5</b>	<b>Approximation von Funktionen</b>	<b>85</b>
5.1	Beispiel: Computergrafik . . . . .	85
5.2	Polynominterpolation . . . . .	86
5.3	Auswertung per Neville-Aitken-Verfahren . . . . .	87
5.4	Auswertung mit Newtons dividierten Differenzen . . . . .	89
5.5	Approximation von Funktionen . . . . .	93
<b>6</b>	<b>Numerische Integration</b>	<b>97</b>
6.1	Beispiel: Wahrscheinlichkeiten . . . . .	97
6.2	Quadraturformeln . . . . .	98
6.3	Fehleranalyse . . . . .	103

*Inhaltsverzeichnis*

<b>7</b>	<b>Gewöhnliche Differentialgleichungen</b>	<b>111</b>
7.1	Beispiel: Numerische Simulationen . . . . .	111
7.2	Integralgleichung . . . . .	112
7.3	Zeitschrittverfahren . . . . .	114
7.4	Verfeinerungen und Variationen* . . . . .	119
	<b>Literaturverzeichnis</b>	<b>125</b>

# 1 Einleitung

Viele Gesetzmäßigkeiten in den Natur-, Ingenieur- und auch Wirtschaftswissenschaften werden mit Hilfe mathematischer Gleichungen beschrieben: Die auf Newton zurückgehenden Axiome der klassischen Mechanik werden in der Regel mit Hilfe von Differentialgleichungen ausgedrückt, das Verhalten einfacher Schaltungen mit Hilfe der Kirchhoff'schen Gesetze, während bei der Modellierung des zu erwartenden Gewinns eines Aktienpakets Integrale zum Einsatz kommen.

In einfachen Fällen lassen sich diese Gleichungen per Hand lösen, in der Praxis ist es jedoch wesentlich häufiger nicht möglich: Entweder lässt sich die Lösung nicht in der üblichen Weise in geschlossener Form darstellen, oder es sind schlicht zu viele Variablen im Spiel.

Die *numerische Mathematik* (oder kurz *Numerik*) beschäftigt sich mit der Frage, wie derartige Probleme möglichst schnell gelöst werden können. Dabei ist die Wahl des richtigen Verfahrens von entscheidender Bedeutung: Ein lineares Gleichungssystem mit der Cramer'schen Regel aufzulösen ist zwar theoretisch machbar, führt in der Praxis aber schon bei relativ kleinen Problemen zu einem inakzeptablen Zeitaufwand. Das Gauß'sche Eliminationsverfahren dagegen ist wesentlich effizienter und kann auf modernen Computern auch noch Systeme mit 10 000 Unbekannten in vertretbarer Zeit behandeln.

Ein weiterer wichtiger Gesichtspunkt ist die Genauigkeit der Berechnung: Bei Berechnungen auf Grundlage von Messdaten treten immer auch Messfehler auf, die das Ergebnis der Berechnung verfälschen. Bei der Modellierung eines physikalischen Prozesses wird man in der Regel von vereinfachenden Annahmen ausgehen und so einen Modellfehler einführen, der ebenfalls das Ergebnis verändert. Schließlich kommen bei der Durchführung der Berechnung auf einem Computer auch noch Rundungsfehler hinzu. Um die Qualität des Ergebnisses beurteilen zu können, müssen wir also auch untersuchen, welche Genauigkeit wir überhaupt erwarten dürfen.

Viele numerische Verfahren bieten die Möglichkeit, einen Kompromiss zwischen Geschwindigkeit und Genauigkeit einzugehen: Falls wir wissen, dass unsere Messfehler ohnehin nur eine Genauigkeit von 3% des Ergebnisses zulassen, ist es unkritisch, auch die Berechnung so weit zu vereinfachen, dass sie einen zusätzlichen Fehler von höchstens 1% hinzufügt, dafür aber wesentlich schneller ausgeführt werden kann. Deshalb werden viele der im Rahmen dieser Vorlesung vorgestellten Verfahren Näherungsverfahren sein, bei denen wir sowohl Zeitbedarf als auch Genauigkeit analysieren.

In diesem Kapitel werden wir zunächst einige einfache Algorithmen kennen lernen, mit deren Hilfe sich Aufgaben, die auf den ersten Blick sehr rechenintensiv sind, sehr elegant und schnell lösen lassen.

## 1.1 Beispiel: Sortieren

Als Beispiel dafür, welchen Einfluss die Wahl des Algorithmus auf den Aufwand für das Lösen eines Problems hat, untersuchen wir die einfache Aufgabe,  $n$  Zahlen  $a_1, a_2, \dots, a_n \in \mathbb{R}$  zu sortieren. Es soll also ein sortierter Vektor

$$\hat{\mathbf{a}} = \begin{pmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_n \end{pmatrix}, \quad \hat{a}_1 \leq \hat{a}_2 \leq \dots \leq \hat{a}_n$$

konstruiert werden, der dieselben Einträge wie der ursprüngliche Vektor enthält, für den also

$$\{a_1, \dots, a_n\} = \{\hat{a}_1, \dots, \hat{a}_n\}$$

gilt. Ein einfacher Algorithmus für die Behandlung dieser Aufgabe ist das *Sortieren durch Einfügen*: Wir gehen davon aus, dass die ersten  $k$  Einträge bereits sortiert sind, dass also  $a_1 \leq a_2 \leq \dots \leq a_k$  gilt. Nun möchten wir  $a_{k+1}$  in diesem Teilvektor an die richtige Stelle bringen. Dazu vergleichen wir  $a_{k+1}$  zunächst mit dessen größtem Element  $a_k$ . Falls  $a_k \leq a_{k+1}$  gilt, sind wir fertig. Ansonsten muss  $a_{k+1}$  vor  $a_k$  eingefügt werden. Also prüfen wir, ob  $a_{k-1} \leq a_{k+1}$  gilt. Falls ja, gehört  $a_{k+1}$  zwischen  $a_{k-1}$  und  $a_k$ . Falls nein, muss  $a_{k+1}$  auch vor  $a_{k-1}$  eingefügt werden. Im ungünstigsten Fall, nämlich wenn  $a_{k+1} < a_1$  gilt, benötigen wir  $k$  Vergleiche, um die richtige Position für  $a_{k+1}$  zu finden.

Auf dieser Grundidee können wir nun aufbauen: Wir beginnen mit dem Teilvektor  $a_1$ , der trivial „sortiert“ ist. Dann fügen wir nach und nach  $a_2, a_3, \dots, a_n$  mit der beschriebenen Technik ein. Um den Arbeitsaufwand möglichst gering zu halten, speichern wir das gerade einzufügende Element in einer Hilfsvariablen  $y$  und schaffen Platz im bereits sortierten Vektor, indem wir dessen Einträge nach rechts verschieben, solange die richtige Position nicht gefunden ist. Der resultierende Algorithmus nimmt die folgende Form an:

```

procedure insertionsort( $n$ , var  $\mathbf{a}$ );
for  $k = 1$  to  $n - 1$  do begin
     $y \leftarrow a_{k+1}$ ;    $i \leftarrow k$ ;
    while  $i \geq 1$  and  $a_i > y$  do begin
         $a_{i+1} \leftarrow a_i$ ;    $i \leftarrow i - 1$ 
    end;
     $a_{i+1} \leftarrow y$ 
end

```

Die Funktion wird mit dem Vektor  $\mathbf{a} = (a_1, \dots, a_n)$  aufgerufen und überschreibt ihn mit dem aufsteigend sortierten Vektor  $\hat{\mathbf{a}} = (\hat{a}_1, \dots, \hat{a}_n)$ .

Es stellt sich die Frage nach dem Arbeitsaufwand, den dieser Sortieralgorithmus mit sich bringt. Ein sinnvolles Maß für den Aufwand ist die Anzahl der Vergleiche, die der Algorithmus im ungünstigsten Fall benötigt. Im schlimmsten Fall wird die innere Schleife

$k$  Vergleiche benötigen, so dass insgesamt

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

Vergleiche anfallen. Für große Vektoren erwarten wir also einen Aufwand von ungefähr  $n^2/2$  Vergleichsoperationen.

Ein weiterer einfacher Sortieralgorithmus ist das *bubblesort*-Verfahren: Für jedes  $i \in \{1, \dots, n-1\}$  wird  $a_i$  mit  $a_{i+1}$  verglichen, und falls  $a_i > a_{i+1}$  gelten sollte, werden die beiden Einträge getauscht. Dieses Vorgehen wird wiederholt, bis alle Einträge in der richtigen Reihenfolge vorliegen.

```

procedure bubblesort( $n$ , var  $\mathbf{a}$ );
repeat
   $\sigma \leftarrow 0$ ;
  for  $i = 1$  to  $n - 1$  do
    if  $a_i > a_{i+1}$  then begin
       $h \leftarrow a_{i+1}$ ;  $a_{i+1} \leftarrow a_i$ ;  $a_i \leftarrow h$ ;
       $\sigma \leftarrow \sigma + 1$ 
    end
  until  $\sigma = 0$ 

```

Nach dem ersten Durchgang der inneren Schleife ist sicher gestellt, dass das größte Element im letzten Eintrag des Vektors angekommen ist, nach dem zweiten ist auch das zweitgrößte an seinem Ort angekommen, nach dem  $k$ -ten Durchgang stehen die  $k$  größten Elemente sortiert am Ende des Vektors, so dass nach  $n-1$  Durchgängen der Vektor vollständig sortiert ist. Die Anzahl der Vergleiche beläuft sich also auf höchstens  $(n-1)^2$ . Wir können den Algorithmus so modifizieren, dass die innere Schleife im  $k$ -ten Durchgang nur noch bis  $n-k$  läuft, da die letzten  $k-1$  Einträge ja bereits sortiert sind. Dann genügen wie beim Sortieren durch Einfügen

$$\sum_{k=1}^{n-1} (n-k) = \frac{n(n-1)}{2}$$

Vergleichsoperationen, um den Vektor vollständig zu sortieren.

Beide Algorithmen sind zwar einfach, allerdings aufgrund des quadratisch mit dem Datenvolumen wachsenden Arbeitsaufwands für große Vektoren zu langsam. Deshalb suchen wir nach einer schnelleren Variante.

Erfolgreich in diesem Bereich sind “divide and conquer”-Verfahren (lateinisch “divide et impera”, deutsch “teile und herrsche”), die auf der Idee beruhen, ein Problem in kleine Teilprobleme zu zerlegen, die sich einfacher lösen lassen, um dann aus den Lösungen der Teilprobleme eine Lösung des Gesamtproblems zu rekonstruieren.

Als Beispiel soll hier das *mergesort*-Verfahren vorgestellt werden. Zur Vereinfachung der Darstellung beschränken wir uns auf den Fall, dass die Länge der Vektoren eine

## 1 Einleitung

Zweierpotenz ist, dass also  $n = 2^p$  für ein  $p \in \mathbb{N}_0$  gilt. Den zu sortierenden Vektor  $\mathbf{a}$  können wir in zwei Hälften

$$\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n/2} \end{pmatrix} := \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n/2} \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n/2} \end{pmatrix} := \begin{pmatrix} a_{n/2+1} \\ a_{n/2+2} \\ \vdots \\ a_n \end{pmatrix}$$

zerlegen. Wir gehen davon aus, dass wir die kleineren Teilvektoren  $\mathbf{b}$  und  $\mathbf{c}$  sortieren, dass wir also sortierte Vektoren

$$\hat{\mathbf{b}} = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_{n/2} \end{pmatrix}, \quad \hat{b}_1 \leq \hat{b}_2 \leq \dots \leq \hat{b}_{n/2}, \quad \hat{\mathbf{c}} = \begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \vdots \\ \hat{c}_{n/2} \end{pmatrix}, \quad \hat{c}_1 \leq \hat{c}_2 \leq \dots \leq \hat{c}_{n/2}$$

praktisch konstruieren können. Sobald uns diese Vektoren zur Verfügung stehen, lässt sich einfach der kleinste Eintrag des Vektors  $\mathbf{a}$  berechnen, denn es gilt

$$\begin{aligned} \min\{a_1, \dots, a_n\} &= \min\{b_1, \dots, b_{n/2}, c_1, \dots, c_{n/2}\} \\ &= \min\{\hat{b}_1, \dots, \hat{b}_{n/2}, \hat{c}_1, \dots, \hat{c}_{n/2}\} = \min\{\hat{b}_1, \hat{c}_1\}. \end{aligned}$$

Wir müssen lediglich prüfen, ob  $\hat{b}_1$  kleiner oder größer als  $\hat{c}_1$  ist. Im ersten Fall ist  $\hat{b}_1$  der erste Eintrag  $\hat{a}_1$  des gesuchten sortierten Vektors  $\hat{\mathbf{a}}$ , im zweiten Fall ist es  $\hat{c}_1$ . Für die Berechnung des zweitkleinsten Eintrags bietet es sich an, den bereits berechneten kleinsten Eintrag aus unserer Menge zu entfernen und erneut nach dem Minimum zu suchen. Entsprechend können wir auch die folgenden Einträge des Ergebnisvektors  $\hat{\mathbf{a}}$  auf den Vergleich von jeweils nur zwei Zahlen zurückführen.

```

procedure mergesort( $n$ , var  $\mathbf{a}$ );
if  $n \geq 2$  then begin
   $m_1 \leftarrow n/2$ ;  $m_2 \leftarrow n - m_1$ ;
  for  $i \in \{1, \dots, m_1\}$  do  $b_i \leftarrow a_i$ ;
  for  $i \in \{1, \dots, m_2\}$  do  $c_i \leftarrow a_{i+m_1}$ ;
  mergesort( $m_1$ ,  $\mathbf{b}$ );
  mergesort( $m_2$ ,  $\mathbf{c}$ );
   $k \leftarrow 1$ ;  $\ell \leftarrow 1$ ;
  for  $i = 1$  to  $n$  do
    if  $\ell > m_2$  do begin  $a_i \leftarrow b_k$ ;  $k \leftarrow k + 1$  end
    else if  $k > m_1$  do begin  $a_i \leftarrow c_\ell$ ;  $\ell \leftarrow \ell + 1$  end
    else if  $b_k \leq c_\ell$  do begin  $a_i \leftarrow b_k$ ;  $k \leftarrow k + 1$  end
    else begin  $a_i \leftarrow c_\ell$ ;  $\ell \leftarrow \ell + 1$  end
end

```



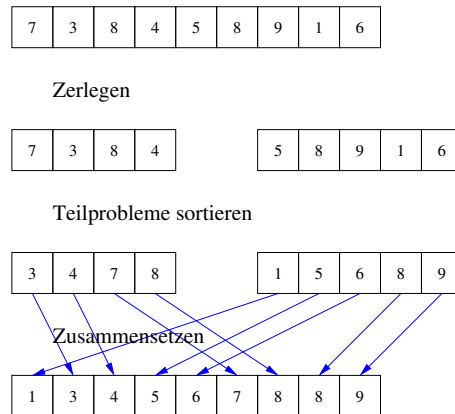


Abbildung 1.1: Prinzip des Mergesort-Algorithmus

Anschaulich gehen wir so vor, dass wir jeweils die ersten Einträge der beiden sortierten Teilvektoren  $\hat{\mathbf{b}}$  und  $\hat{\mathbf{c}}$  vergleichen, den kleineren der beiden in das Ergebnis übernehmen und aus dem ursprünglichen Vektor streichen, bis  $\hat{\mathbf{a}}$  vollständig gefüllt ist. Dabei gibt  $k$  jeweils den ersten noch gültigen Eintrag in dem Vektor  $\hat{\mathbf{b}}$  an, während  $\ell$  dieselbe Rolle für den Vektor  $\hat{\mathbf{c}}$  spielt und  $i$  den nächsten zu bestimmenden Eintrag des Ergebnisvektors  $\hat{\mathbf{a}}$  festlegt. Wir prüfen zunächst, ob einer der beiden Vektoren bereits leer ist. In diesem Fall übernehmen wir einfach den Rest des anderen Vektors. Ansonsten vergleichen wir die kleinsten Einträge der beiden Vektoren und kopieren den kleineren.

Ist dieser Algorithmus nun schneller als der vorige? Zur Beantwortung dieser Frage bezeichnen wir die Anzahl der Vergleichsoperationen, die für einen Vektor der Länge  $n$  erforderlich sind, mit  $R_n$ . Da unser Algorithmus für  $n = 1$  nichts tut, gilt

$$R_1 = 0.$$

Für  $n > 1$  werden zunächst die Teilvektoren sortiert, dafür sind  $R_{m_1}$  und  $R_{m_2}$  Vergleichsoperationen erforderlich. Um die beiden Teilvektoren zusammenzusetzen sind  $n$  Vergleiche erforderlich (Vergleiche der Laufvariablen zählen wir nicht). Insgesamt erhalten wir die Formel

$$R_n = R_m + R_{n-m} + n.$$

Unsere Aufgabe besteht nun darin, diese Rekurrenzgleichung zu lösen. Dazu beschränken wir uns zunächst auf den Fall, dass  $n$  eine Zweierpotenz ist, dass also  $n = 2^p$  mit  $p \in \mathbb{N}_0$  gilt, und erhalten

$$\begin{aligned} R_2 &= 2R_1 + 2 = 2 = 2 \cdot 1, \\ R_4 &= 2R_2 + 4 = 8 = 4 \cdot 2, \\ R_8 &= 2R_4 + 8 = 24 = 8 \cdot 3, \\ R_{16} &= 2R_8 + 16 = 64 = 16 \cdot 4, \\ R_{32} &= 2R_{16} + 32 = 160 = 32 \cdot 5. \end{aligned}$$

## 1 Einleitung

Diese Folge legt nahe, dass  $R_n = n \log_2(n)$  gilt, und diese Gleichung lässt sich in der Tat für Zweierpotenzen auch direkt beweisen.

Für allgemeine  $n \in \mathbb{N}$  können wir immerhin noch eine Abschätzung erhalten, indem wir geeignet auf- und abrunden. Dazu verwenden wir die *Gauß-Klammern*, die durch

$$\lfloor x \rfloor := \max\{n \in \mathbb{Z} : n \leq x\}, \quad \lceil x \rceil := \min\{n \in \mathbb{Z} : x \leq n\} \quad \text{für alle } x \in \mathbb{R}$$

definiert sind.

**Lemma 1.1 (Rekurrenzabschätzung)** *Die durch*

$$\begin{aligned} R_1 &:= 0, \\ R_n &:= R_m + R_{n-m} + n \text{ mit } m := \lfloor n/2 \rfloor \end{aligned} \quad \text{für alle } n \in \mathbb{N}$$

*definierten Größen erfüllen die Abschätzung*

$$R_n \leq n \lceil \log_2(n) \rceil \tag{1.1}$$

*für alle  $n \in \mathbb{N}$ .*

*Beweis.* Wir führen den Beweis per *abschnittsweiser Induktion*.

*Induktionsanfang.* Für  $n = 1$  gilt  $R_1 = 0 = 1 \cdot 0 = 1 \lceil \log_2(1) \rceil$ .

*Induktionsvoraussetzung.* Sei  $N \in \mathbb{N}$  so gegeben, dass (1.1) für alle  $n \in [1 : N]$  gilt.

*Induktionsschritt.* Sei  $n := N + 1$ . Falls  $n$  eine gerade Zahl ist, also  $n = 2m$  gilt, dürfen wir die Induktionsvoraussetzung wegen  $m \leq N$  anwenden, um

$$\begin{aligned} R_n &= R_m + R_{n-m} + n = 2R_m + n \leq 2m \lceil \log_2(n/2) \rceil + n \\ &= n \lceil \log_2(n) - 1 \rceil + n = n \lceil \log_2(n) \rceil \end{aligned}$$

zu erhalten. Falls  $n$  hingegen ungerade ist, müssen wir einen etwas genaueren Blick auf den Logarithmus werfen: Sei  $p := \lceil \log_2(n) \rceil$ , es gelte also  $p - 1 < \log_2(n) \leq p$  und damit auch  $2^{p-1} < n \leq 2^p$ . Wegen  $n = N + 1 \geq 2$  muss  $p \geq 1$  gelten, also ist  $2^p$  gerade. Da  $n$  ungerade ist, folgt  $n < 2^p$ , also  $2^{p-1} < n + 1 \leq 2^p$  und somit  $\lceil \log_2(n + 1) \rceil = \lceil \log_2(n) \rceil$ , so dass wir

$$\begin{aligned} \lceil \log_2(m) \rceil &= \lceil \log_2((n-1)/2) \rceil = \lceil \log_2(n-1) \rceil - 1 \leq \lceil \log_2(n) \rceil - 1, \\ \lceil \log_2(n-m) \rceil &= \lceil \log_2((n+1)/2) \rceil = \lceil \log_2(n+1) \rceil - 1 = \lceil \log_2(n) \rceil - 1 \end{aligned}$$

erhalten. Wir haben  $m = (n-1)/2 = N/2 \leq N$ , dürfen also die Induktionsvoraussetzung auf  $R_m$  anwenden. Da  $n = N + 1 \geq 2$  gilt, also  $m \geq 1$ , haben wir auch  $n-m \leq n-1 = N$ , so dass wir die Induktionsvoraussetzung auch auf  $R_{n-m}$  anwenden können und

$$\begin{aligned} R_n &= R_m + R_{n-m} + n \leq m \lceil \log_2(m) \rceil + (n-m) \lceil \log_2(n-m) \rceil + n \\ &\leq n(\lceil \log_2(n) \rceil - 1) + n = n \lceil \log_2(n) \rceil \end{aligned}$$

erhalten. Damit ist der Beweis vollständig. ■

Unser neuer Algorithmus benötigt also lediglich eine zu  $n \log_2 n$  proportionale Anzahl von Operationen anstelle einer zur  $n^2$  proportionalen, so dass beispielsweise das Sortieren von  $n = 2^{20} \approx 1\,000\,000$  Zahlen mit  $2^{20} \cdot 20 \approx 20\,000\,000$  Vergleichsoperationen erfolgen kann statt mit  $\frac{n(n-1)}{2} \approx 500\,000\,000\,000$ . In diesem Fall ist *mergesort* um einen Faktor von ungefähr 25 000 schneller.

## 1.2 Beispiel: Schnelle Fourier-Transformation

Wir untersuchen die Berechnung der *Fourier-Transformation* eines Vektors  $\mathbf{x} \in \mathbb{C}^n$  mit geradem  $n \in \mathbb{N}$ . Sie ist gegeben durch

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}, \quad \hat{x}_\nu := \sum_{j=0}^{n-1} x_j e^{-2\pi i \nu j/n}, \quad \hat{\mathbf{x}} := \begin{pmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \vdots \\ \hat{x}_{n-1} \end{pmatrix}$$

Wenn wir die  $n$  Werte des Vektors  $\hat{\mathbf{x}}$  in dieser Weise berechnen, müssen für jeden mindestens  $n$  Multiplikationen durchgeführt werden, so dass mindestens  $n^2$  Rechenoperationen erforderlich werden. Die Berechnung lässt sich wesentlich schneller durchführen, wenn wir die Struktur der in der Formel auftretenden Koeffizienten ausnutzen: Wir setzen  $m := n/2$  und zerlegen die Summe in eine Summe über gerade Indizes  $j = 2k$  und eine Summe über ungerade Indizes  $j = 2k + 1$  und erhalten

$$\begin{aligned} \hat{x}_\nu &= \sum_{j=0}^{n-1} x_j e^{-2\pi i \nu j/n} = \sum_{k=0}^{m-1} x_{2k} e^{-2\pi i \nu 2k/n} + \sum_{k=0}^{m-1} x_{2k+1} e^{-2\pi i \nu (2k+1)/n} \\ &= \sum_{k=0}^{m-1} x_{2k} e^{-2\pi i \nu k/m} + e^{-2\pi i \nu/n} \sum_{k=0}^{m-1} x_{2k+1} e^{-2\pi i \nu k/m}. \end{aligned} \quad (1.2)$$

Falls  $\nu \in \{0, \dots, m-1\}$  gelten sollte, stellen wir fest, dass die erste Summe gerade die Fourier-Transformierte des Vektors der geraden Indizes ist,

$$\mathbf{y} := \begin{pmatrix} x_0 \\ x_2 \\ \vdots \\ x_{n-2} \end{pmatrix}, \quad \hat{y}_\mu := \sum_{k=0}^{m-1} y_k e^{-2\pi i \mu k/m}, \quad \hat{\mathbf{y}} := \begin{pmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \vdots \\ \hat{y}_{m-1} \end{pmatrix},$$

während sich die zweite Summe als Fourier-Transformierte des Vektors der ungeraden Indizes entpuppt, also als

$$\mathbf{z} := \begin{pmatrix} x_1 \\ x_3 \\ \vdots \\ x_{n-1} \end{pmatrix}, \quad \hat{z}_\mu := \sum_{k=0}^{m-1} z_k e^{-2\pi i \mu k/m}, \quad \hat{\mathbf{z}} := \begin{pmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \vdots \\ \hat{z}_{m-1} \end{pmatrix}.$$

In diesem Fall können wir also statt einer Fouriertransformation für einen Vektor der Länge  $n$  zwei Fouriertransformationen für Vektoren der Länge  $m = n/2$  durchführen und anschließend die Ergebnisse kombinieren, um das Ergebnis zu berechnen: Gemäß (1.2) gilt die Gleichung

$$\hat{x}_\nu = \hat{y}_\nu + e^{-2\pi i \nu/n} \hat{z}_\nu \quad \text{für alle } \nu \in \{0, \dots, m-1\}.$$

## 1 Einleitung

Falls dagegen  $\nu \in \{m, \dots, n-1\}$  gilt, können wir die Periodizität der komplexen Exponentialfunktion ausnutzen: Es gelten

$$e^{-2\pi i \nu k/m} = e^{-2\pi i (\nu-m)k/m}, \quad e^{-2\pi i \nu/n} = -e^{-2\pi i (\nu-m)/n},$$

und wir können (1.2) in der Form

$$\begin{aligned} \hat{x}_\nu &= \sum_{k=0}^{m-1} y_k e^{-2\pi i \nu k/m} + e^{-2\pi i \nu/n} \sum_{k=0}^{m-1} z_k e^{-2\pi i \nu k/m} \\ &= \sum_{k=0}^{m-1} y_k e^{-2\pi i (\nu-m)k/m} - e^{-2\pi i (\nu-m)/n} \sum_{k=0}^{m-1} z_k e^{-2\pi i (\nu-m)k/m} \\ &= \hat{y}_{\nu-m} - e^{-2\pi i (\nu-m)/n} \hat{z}_{\nu-m} \quad \text{für alle } \nu \in \{m, \dots, n-1\} \end{aligned}$$

schreiben und so ebenfalls auf Grundlage der Hilfsvektoren  $\hat{\mathbf{y}}$  und  $\hat{\mathbf{z}}$  berechnen.

Falls wir voraussetzen, dass auch  $n/2$ ,  $n/4$  und so weiter gerade sind, falls also  $n = 2^p$  für ein  $p \in \mathbb{N}_0$  gilt, können wir diese Rechenvorschrift rekursiv anwenden, um den besonders schnellen *Algorithmus von Cooley und Tukey* für die Berechnung der Fourier-Transformation zu erhalten:

```

procedure fft( $n$ ,  $\mathbf{x}$ , var  $\hat{\mathbf{x}}$ )
if  $n = 1$  then
     $\hat{x}_0 = x_0$ 
else begin
     $m \leftarrow n/2$ ;
    for  $k \in \{0, \dots, m-1\}$  do begin
         $y_k \leftarrow x_{2k}$ ;  $z_k \leftarrow x_{2k+1}$ 
    end;
    fft( $m$ ,  $\mathbf{y}$ ,  $\hat{\mathbf{y}}$ );
    fft( $m$ ,  $\mathbf{z}$ ,  $\hat{\mathbf{z}}$ );
     $\gamma \leftarrow 1$ ;  $\omega \leftarrow e^{-2\pi i/n}$ ;
    for  $\nu = 0$  to  $m-1$  do begin
         $\hat{x}_\nu \leftarrow \hat{y}_\nu + \gamma \hat{z}_\nu$ ;  $\hat{x}_{\nu+m} \leftarrow \hat{y}_\nu - \gamma \hat{z}_\nu$ ;
         $\gamma \leftarrow \gamma \omega$ 
    end
end

```

Hier werden die Werte  $e^{-2\pi i \nu/n}$  mit Hilfe der  $n$ -ten Einheitswurzel  $\omega = e^{-2\pi i/n}$  berechnet, um möglichst selten die aufwendige Berechnung der komplexen Exponentialfunktion zu benötigen. Es gilt  $\omega^\nu = (e^{-2\pi i/n})^\nu = e^{-2\pi i \nu/n}$ , und diese Koeffizienten werden der Reihe nach für  $\nu = 0, \dots, m$  in der Variablen  $\gamma$  bereit gestellt.

Wir haben bereits gezeigt, dass der Algorithmus das richtige Ergebnis berechnet, wir können aber auch zeigen, dass er das schneller als der ursprüngliche Ansatz tut: Dazu bezeichnen wir mit  $R_n$  die Anzahl der arithmetischen Operationen, die für eine Vektorlänge von  $n \in \mathbb{N}$  benötigt wird. Für  $n = 1$  benötigt unser Algorithmus keine arithmetischen

## 1.2 Beispiel: Schnelle Fourier-Transformation

Operationen (Kopiervorgänge zählen wir nicht mit), es gilt also  $R_1 = 0$ . Für ein gerades  $n \in \mathbb{N}$  fallen zwei rekursive Aufrufe für Vektoren der Länge  $n/2$  an, also ein Aufwand von  $2R_{n/2}$ . Die Berechnung der Einheitswurzel  $\omega$  zählen wir als eine Operation. In der Schleife für  $\nu$  müssen jeweils zwei Multiplikationen, eine Addition und eine Subtraktion ausgeführt werden, um  $\hat{x}_\nu$  und  $\hat{x}_{\nu+m}$  zu bestimmen. Insgesamt fallen also

$$R_n = 2R_{n/2} + 4n/2 + 1 = 2R_{n/2} + 2n + 1$$

arithmetische Operationen an. Ausgehend von  $R_1 = 0$  können wir diese Gleichung auflösen: Wir verwenden den Ansatz  $R_n = 2n \log_2 n + n - 1$ . Falls die Gleichung für  $n/2$  gilt, folgt

$$\begin{aligned} R_n &= 2R_{n/2} + 2n + 1 = 2(2(n/2) \log_2(n/2) + (n/2) - 1) + 2n + 1 \\ &= 2n(\log_2(n) - 1) + n - 2 + 2n + 1 = 2n \log_2(n) + n - 1. \end{aligned}$$

Der Rechenaufwand wächst also fast linear mit der Dimension des Vektors.



## 2 Lineare Gleichungssysteme

### 2.1 Beispiel: Differentialgleichung

Für eine gegebene Funktion  $f \in C[0, 1]$  möchten wir eine Funktion  $u \in C^2[0, 1]$  finden, die die Differentialgleichung

$$-u''(t) = f(t) \quad \text{für alle } t \in (0, 1) \quad (2.1)$$

mit den Randbedingungen

$$u(0) = 0, \quad u(1) = 0$$

erfüllt. Da die Lösung aus einem unendlich-dimensionalen Funktionenraum stammt und sich deshalb nicht in der endlichen Speichermenge, die einem Computer zur Verfügung steht, darstellen lässt, gehen wir zu einer Näherungslösung über: Für ein  $n \in \mathbb{N}$  ersetzen wir das kontinuierliche Intervall  $[0, 1]$  durch die durch

$$t_i := hi, \quad h := \frac{1}{n+1} \quad \text{für alle } i \in \{0, \dots, n+1\}$$

gegebene diskrete Punktmenge  $\{t_0, \dots, t_{n+1}\}$ . Wir sind nur noch daran interessiert, die Werte der Lösung in diesen Punkten zu berechnen, also

$$x_i := u(t_i) \quad \text{für alle } i \in \{0, \dots, n+1\}.$$

Die Werte  $x_0$  und  $x_{n+1}$  brauchen wir nicht zu berechnen, weil sie bereits durch die Randbedingungen

$$x_0 = u(t_0) = u(0) = 0, \quad x_{n+1} = u(t_{n+1}) = u(1) = 0$$

festgelegt sind

Um ein praktisch lösbares Problem zu erhalten, müssen wir die Ableitung in der Differentialgleichung durch etwas ersetzen, das sich mit Hilfe der Werte  $x_0, \dots, x_{n+1}$  berechnen lässt. Nach Definition der Ableitung gelten

$$u'(t_i) \approx \frac{u(t_{i+1}) - u(t_i)}{h}, \quad u'(t_i) \approx \frac{u(t_i) - u(t_{i-1}))}{h},$$

und indem wir die erste Approximation in die zweite einsetzen folgt

$$u''(t_i) \approx \frac{u(t_{i+1}) - 2u(t_i) + u(t_{i-1}))}{h^2}. \quad (2.2)$$

## 2 Lineare Gleichungssysteme

Durch Einsetzen in unsere Differentialgleichung erhalten wir

$$\frac{-x_{i+1} + 2x_i - x_{i-1}}{h^2} \approx -u''(t_i) = f(t_i) \quad \text{für alle } i \in \{1, \dots, n\}.$$

Wir können also Näherungen der Werte  $x_i$  berechnen, indem wir das lineare Gleichungssystem

$$\frac{-\tilde{x}_{i+1} + 2\tilde{x}_i - \tilde{x}_{i-1}}{h^2} = f(t_i) \quad \text{für alle } i \in \{1, \dots, n\}$$

mit  $n$  Gleichungen und den  $n$  Unbekannten  $\tilde{x}_1, \dots, \tilde{x}_n$  auflösen. Damit folgt  $u(t_i) = x_i \approx \tilde{x}_i$ , wir haben also eine Näherung der Lösung in den Punkten  $t_1, \dots, t_n$  gefunden, die sich ohne explizite Kenntnis der Funktion  $u$  berechnen lässt.

Zur Vereinfachung fassen wir die Unbekannten und die rechte Seite zu Vektoren zusammen und stellen das Gleichungssystem durch eine Matrix dar:

$$\mathbf{A} := \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & -1 & \\ & & & -1 & 2 \end{pmatrix}, \quad \mathbf{x} := \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{pmatrix}, \quad \mathbf{b} := \begin{pmatrix} f(t_1) \\ f(t_2) \\ \vdots \\ f(t_n) \end{pmatrix}. \quad (2.3)$$

Damit haben wir eine Matrix  $\mathbf{A} \in \mathbb{R}^n$  und einen Vektor  $\mathbf{b} \in \mathbb{R}^n$  und suchen einen Vektor  $\mathbf{x} \in \mathbb{R}^n$ , der die Gleichung

$$\mathbf{Ax} = \mathbf{b}$$

löst. Die Lösung der Gleichung können wir dann als Näherung der Lösung der Differentialgleichung interpretieren. Da sie die kontinuierliche Punktmenge  $[0, 1]$  durch die diskrete Punktmenge  $\{t_0, t_1, \dots, t_{n+1}\}$  ersetzt, bezeichnet man sie als *Diskretisierung* der Differentialgleichung (2.1).

**Bemerkung 2.1 (Genauigkeit)** Falls  $u$  viermal stetig differenzierbar ist, können wir die Taylor-Entwicklung um  $t_i$  in Kombination mit dem Zwischenwertsatz verwenden, um

$$\frac{u(t_{i+1}) - 2u(t_i) + u(t_{i-1}))}{h^2} = u''(t_i) + \frac{h^2}{12}u^{(4)}(\eta)$$

mit einem Punkt  $\eta \in [t_{i-1}, t_{i+1}]$  zu beweisen. Unsere Näherung der zweiten Ableitung sollte unter diesen Bedingungen also relativ genau sein, falls das Quadrat der Schrittweite  $h$  klein im Verhältnis zu der vierten Ableitung ist.

**Bemerkung 2.2 (Verallgemeinerung)** Differenzenquotienten der Form (2.2) lassen sich erheblich verallgemeinern, um partielle Differentialgleichungen aus vielen verschiedenen Anwendungsgebieten (Elektrostatik und -dynamik, Strömungsdynamik, Strukturmechanik) zu behandeln. Besonders erfolgreich ist die Methode der finiten Elemente, die sich besonders gut für die Behandlung komplizierter Geometrien eignet.



## 2.2 Allgemeine Aufgabenstellung

Wir interessieren uns für Verfahren, mit denen sich allgemeine lineare Gleichungssysteme der Form

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

lösen lassen. Wie zuvor verwenden wir die Matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  und die Vektoren  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ , gegeben durch

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

um die Gleichung in der kompakten Schreibweise

$$\mathbf{Ax} = \mathbf{b} \tag{2.4}$$

darstellen zu können. Im Folgenden werden wir Matrizen immer mit Großbuchstaben  $\mathbf{A}, \mathbf{L}, \dots$  bezeichnen, während wir für ihre Einträge die entsprechenden Kleinbuchstaben  $a_{ij}, \ell_{jk}, \dots$  verwenden.

## 2.3 Dreiecksmatrizen

Bevor wir uns allgemeinen linearen Gleichungssystemen zuwenden, untersuchen wir zunächst zwei wichtige Spezialfälle:

**Definition 2.3 (Dreiecksmatrizen)** Seien  $\mathbf{L}, \mathbf{R} \in \mathbb{R}^{n \times n}$ . Falls

$$\ell_{ij} = 0 \quad \text{für alle } i < j$$

gilt, nennen wir  $\mathbf{L}$  eine untere Dreiecksmatrix. Falls

$$r_{ij} = 0 \quad \text{für alle } i > j$$

gilt, nennen wir  $\mathbf{R}$  eine obere Dreiecksmatrix.

Ob eine Dreiecksmatrix regulär ist, lässt sich besonders einfach nachprüfen: Eine beliebige quadratische Matrix ist genau dann regulär, wenn ihre Determinante ungleich

## 2 Lineare Gleichungssysteme

null ist, und die Determinante einer Dreiecksmatrix ist gerade das Produkt ihrer Diagonalelemente: Für eine untere Dreiecksmatrix  $\mathbf{L} \in \mathbb{R}^{n \times n}$  und eine obere Dreiecksmatrix  $\mathbf{R} \in \mathbb{R}^{n \times n}$  gelten

$$\det(\mathbf{L}) = \prod_{i=1}^n \ell_{ii}, \quad \det(\mathbf{R}) = \prod_{i=1}^n r_{ii},$$

also sind beide Matrizen genau dann regulär, wenn alle Diagonalelemente ungleich null sind.

Lineare Gleichungssysteme mit regulären Dreiecksmatrizen lassen sich besonders einfach auflösen. Für die Herleitung der entsprechenden Algorithmen zerlegen wir die Aufgabe wieder in kleinere Teilaufgaben: Wir führen Teilmatrizen

$$\mathbf{L}_{**} = \begin{pmatrix} \ell_{22} & & \\ \vdots & \ddots & \\ \ell_{n2} & \dots & \ell_{nn} \end{pmatrix}, \quad \mathbf{L}_{*1} = \begin{pmatrix} \ell_{21} \\ \vdots \\ \ell_{n1} \end{pmatrix}$$

und Teilvektoren

$$\mathbf{x}_* = \begin{pmatrix} x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b}_* = \begin{pmatrix} b_2 \\ \vdots \\ b_n \end{pmatrix}$$

ein und schreiben das zu lösende Gleichungssystem  $\mathbf{Lx} = \mathbf{b}$  in *Blockform*:

$$\begin{pmatrix} \ell_{11} & \\ \mathbf{L}_{*1} & \mathbf{L}_{**} \end{pmatrix} \begin{pmatrix} x_1 \\ \mathbf{x}_* \end{pmatrix} = \mathbf{Lx} = \mathbf{b} = \begin{pmatrix} b_1 \\ \mathbf{b}_* \end{pmatrix}.$$

Durch Einsetzen der Teilvektoren und -matrizen sieht man leicht, dass beide Gleichungen äquivalent sind. Blockmatrizen und -vektoren lassen sich genau wie „normale“ Matrizen und Vektoren multiplizieren, so dass wir aus der Blockform der Gleichung die Beziehung

$$\begin{pmatrix} b_1 \\ \mathbf{b}_* \end{pmatrix} = \begin{pmatrix} \ell_{11} & \\ \mathbf{L}_{*1} & \mathbf{L}_{**} \end{pmatrix} \begin{pmatrix} x_1 \\ \mathbf{x}_* \end{pmatrix} = \begin{pmatrix} \ell_{11}x_1 \\ \mathbf{L}_{*1}x_1 + \mathbf{L}_{**}\mathbf{x}_* \end{pmatrix}$$

erhalten. Komponentenweise gesehen ergeben sich die Gleichungen

$$\ell_{11}x_1 = b_1, \quad \mathbf{L}_{*1}x_1 + \mathbf{L}_{**}\mathbf{x}_* = \mathbf{b}_*,$$

von denen sich die erste einfach auflösen lässt: Da  $\mathbf{L}$  als regulär vorausgesetzt ist, gilt  $\ell_{11} \neq 0$ , so dass wir die Gleichung

$$x_1 = \frac{b_1}{\ell_{11}}$$

erhalten. Da wir  $x_1$  nun kennen, können wir die zweite Gleichung in die Form

$$\mathbf{L}_{**}\mathbf{x}_* = \mathbf{b}_* - \mathbf{L}_{*1}x_1$$

bringen und stellen fest, dass wir wieder ein lineares Gleichungssystem mit einer unteren Dreiecksmatrix erhalten haben, allerdings diesmal mit lediglich  $n - 1$  Unbekannten. Falls  $n > 1$  gelten sollte, können wir die beiden Schritte (Division durch das Diagonalelement, Subtraktion einer Spalte von der rechten Seite  $\mathbf{b}$ ) auf immer kleiner werdende Matrizen anwenden, bis  $\mathbf{x}_*$  berechnet ist. Wir erhalten den folgenden Algorithmus:

```

procedure forward_subst( $n, \mathbf{L}, \mathbf{var} \mathbf{b}, \mathbf{x}$ );
for  $j = 1$  to  $n$  do begin
   $x_j \leftarrow b_j / \ell_{jj}$ ;
  for  $i \in \{j + 1, \dots, n\}$  do
     $b_i \leftarrow b_i - \ell_{ij} x_j$ 
end

```

Wie man sieht wird zunächst  $x_1$  berechnet, dann wird von  $\mathbf{b}_*$  der Vektor  $\mathbf{L}_{*1}x_1$  subtrahiert und anschließend zu dem durch  $\mathbf{L}_{**}$  definierten Teilproblem übergegangen. Die Variable  $j$  gibt dabei jeweils den Index des linken oberen Elements der gerade behandelten Teilmatrix an.

In praktischen Implementierungen wird man in der Regel auf den zusätzlichen Vektor  $\mathbf{x}$  verzichten und den Vektor  $\mathbf{b}$  im Zuge des Verfahrens mit den Komponenten des Lösungsvektors überschreiben.

Aus naheliegenden Gründen bezeichnet man dieses Verfahren mit dem Namen *Vorwärtseinsetzen*.

Für obere Dreiecksmatrizen können wir ähnlich vorgehen. Wir verwenden die Zerlegung

$$\begin{pmatrix} \mathbf{R}_{**} & \mathbf{R}_{*n} \\ & r_{nn} \end{pmatrix} \begin{pmatrix} \mathbf{x}_* \\ x_n \end{pmatrix} = \mathbf{R}\mathbf{x} = \mathbf{b} = \begin{pmatrix} \mathbf{b}_* \\ b_n \end{pmatrix}$$

mit passend definierten Teilvektoren und -matrizen und erhalten die Gleichungen

$$r_{nn}x_n = b_n, \quad \mathbf{R}_{**}\mathbf{x}_* + \mathbf{R}_{*n}x_n = \mathbf{b}_*,$$

die wir in die Form

$$x_n = b_n / r_{nn}, \quad \mathbf{R}_{**}\mathbf{x}_* = \mathbf{b}_* - \mathbf{R}_{*n}x_n$$

bringen können, die sich ähnlich wie zuvor direkt in einen Algorithmus übersetzen lässt:

```

procedure backward_subst( $n, \mathbf{R}, \mathbf{var} \mathbf{b}, \mathbf{x}$ );
for  $j = n$  downto  $1$  do begin
   $x_j \leftarrow b_j / r_{jj}$ ;
  for  $i \in \{1, \dots, j - 1\}$  do
     $b_i \leftarrow b_i - r_{ij} x_j$ 
end

```

In diesem Fall gibt  $j$  jeweils den Index des rechten unteren Elements der gerade behandelten Teilmatrix an. Auch hier lässt sich der zusätzliche Vektor  $\mathbf{x}$  einsparen, wenn man

## 2 Lineare Gleichungssysteme

die Einträge des Vektors  $\mathbf{b}$  mit dem Ergebnis überschreibt. Es überrascht nicht, dass dieser Algorithmus unter dem Namen *Rückwärtseinsetzen* bekannt ist.

Natürlich müssen wir uns auch in diesem Fall wieder die Frage stellen, wie groß der Rechenaufwand unserer Algorithmen ist. Traditionell werden dabei lediglich die arithmetischen Operationen mit reellen Zahlen gezählt und nicht Kopiervorgänge oder Manipulationen der Indizes. In diesem Sinne benötigt das Vorwärtseinsetzen

$$\sum_{j=1}^n (1 + 2(n-j)) = n + 2 \sum_{j=1}^n (n-j) = n + 2 \sum_{j=0}^{n-1} j = n + 2 \frac{n(n-1)}{2} = n^2$$

Operationen, und dasselbe Ergebnis erhalten wir auch für das strukturell sehr ähnliche Rückwärtseinsetzen. Da bei beiden Operationen jeweils

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

Matrixeinträge verarbeitet werden müssen, ist dieser Rechenaufwand kaum weiter zu reduzieren, der Algorithmus darf also als effizient angesehen werden.

### 2.4 LR-Zerlegung

Unser Ziel ist es nun, das Lösen des allgemeinen linearen Gleichungssystems (2.4) auf das Lösen unterer und oberer Dreiecksmatrizen zurückzuführen, denn diese Aufgaben lassen sich mit den bereits diskutierten Verfahren einfach lösen. Die Idee besteht darin, die Matrix als ein Produkt aus einer unteren und einer oberen Dreiecksmatrix darzustellen:

**Definition 2.4 (LR-Zerlegung)** Sei  $\mathbf{A} \in \mathbb{R}^{n \times n}$  eine Matrix. Sei  $\mathbf{L} \in \mathbb{R}^{n \times n}$  eine untere und  $\mathbf{R} \in \mathbb{R}^{n \times n}$  eine obere Dreiecksmatrix. Falls

$$\mathbf{A} = \mathbf{LR} \tag{2.5}$$

gilt, bezeichnen wir das Paar  $(\mathbf{L}, \mathbf{R})$  als eine LR-Zerlegung der Matrix  $\mathbf{A}$ .

Falls uns eine LR-Zerlegung zur Verfügung steht, können wir das Gleichungssystem in der Form

$$\mathbf{b} = \mathbf{Ax} = \mathbf{LRx}$$

schreiben und mit dem Hilfsvektor  $\mathbf{y} = \mathbf{Rx}$  auf die Gleichungen

$$\mathbf{b} = \mathbf{Ly}, \quad \mathbf{y} = \mathbf{Rx}$$

zurückführen, die sich durch Vorwärts- und Rückwärtseinsetzen einfach auflösen lassen.

Also wenden wir uns der Aufgabe zu, zu einer gegebenen Matrix  $\mathbf{A}$  eine LR-Zerlegung zu konstruieren. Dazu zerlegen wir die beteiligten Matrizen wieder in Teilmatrizen

$$\mathbf{A}_{**} = \begin{pmatrix} a_{22} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n2} & \dots & a_{nn} \end{pmatrix}, \quad \mathbf{A}_{*1} = \begin{pmatrix} a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}, \quad \mathbf{A}_{1*} = (a_{12} \quad \dots \quad a_{1n}),$$

$$\mathbf{L}_{**} = \begin{pmatrix} \ell_{22} & & \\ \vdots & \ddots & \\ \ell_{n2} & \dots & \ell_{nn} \end{pmatrix}, \quad \mathbf{L}_{*1} = \begin{pmatrix} \ell_{21} \\ \vdots \\ \ell_{n1} \end{pmatrix},$$

$$\mathbf{R}_{**} = \begin{pmatrix} r_{22} & \dots & r_{2n} \\ & \ddots & \vdots \\ & & r_{nn} \end{pmatrix}, \quad \mathbf{R}_{1*} = (r_{12} \quad \dots \quad r_{1n})$$

und stellen die definierende Gleichung (2.5) in der Form

$$\begin{pmatrix} a_{11} & \mathbf{A}_{1*} \\ \mathbf{A}_{*1} & \mathbf{A}_{**} \end{pmatrix} = \mathbf{A} = \mathbf{L}\mathbf{R} = \begin{pmatrix} \ell_{11} & \\ \mathbf{L}_{*1} & \mathbf{L}_{**} \end{pmatrix} \begin{pmatrix} r_{11} & \mathbf{R}_{1*} \\ & \mathbf{R}_{**} \end{pmatrix}$$

dar. Komponentenweise gelesen ergeben sich aus dieser Gleichung die Beziehungen

$$a_{11} = \ell_{11}r_{11}, \quad (2.6a)$$

$$\mathbf{A}_{*1} = \mathbf{L}_{*1}r_{11}, \quad (2.6b)$$

$$\mathbf{A}_{1*} = \ell_{11}\mathbf{R}_{1*}, \quad (2.6c)$$

$$\mathbf{A}_{**} = \mathbf{L}_{*1}\mathbf{R}_{1*} + \mathbf{L}_{**}\mathbf{R}_{**}, \quad (2.6d)$$

die wir der Reihe nach auflösen können. Wir gehen dazu für den Augenblick davon aus, dass  $a_{11} \neq 0$  gilt, und stellen fest, dass  $\ell_{11} = 1$  und  $r_{11} = a_{11}$  die erste Gleichung (2.6a) erfüllen. Da wir diese beiden Zahlen nun kennen, können wir die Gleichungen (2.6b) und (2.6c) auflösen:

$$\mathbf{L}_{*1} = \mathbf{A}_{*1}/r_{11}, \quad \mathbf{R}_{1*} = \mathbf{A}_{1*}.$$

Mit diesen Vektoren bringen wir die vierte Gleichung (2.6d) in die Form

$$\mathbf{A}_{**} - \mathbf{L}_{*1}\mathbf{R}_{1*} = \mathbf{L}_{**}\mathbf{R}_{**}$$

und stellen fest, dass es uns wieder gelungen ist, ein Problem der Dimension  $n$  auf ein gleichartiges Problem der Dimension  $n - 1$  zu reduzieren: Die Matrizen  $\mathbf{L}_{**}$  und  $\mathbf{R}_{**}$  ergeben sich aus der LR-Zerlegung der Matrix  $\mathbf{A}_{**} - \mathbf{L}_{*1}\mathbf{R}_{1*}$ .

Bei der praktischen Umsetzung dieser Konstruktion ist es wichtig, den zur Verfügung stehenden Speicher möglichst geschickt auszunutzen. Da wir in der Regel die Matrix  $\mathbf{A}$  nicht mehr benötigen, sobald wir ihre Zerlegung kennen, können wir den von ihr benutzten Speicher mit den Daten der Zerlegung überschreiben: Die erste Zeile können wir mit  $\mathbf{R}_{1*}$  überschreiben, die erste Spalte kann  $\mathbf{L}_{*1}$  aufnehmen, der Koeffizient  $\ell_{11}$  muss nicht gespeichert werden, weil er nach Definition immer gleich eins ist. Entsprechend können wir  $\mathbf{A}_{**}$  mit der Matrix  $\mathbf{A}_{**} - \mathbf{L}_{*1}\mathbf{R}_{1*}$  überschreiben und erhalten als Zwischenergebnis

$$\begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ \ell_{21} & a_{22} - \ell_{21}r_{12} & \dots & a_{2n} - \ell_{21}r_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & a_{n2} - \ell_{n1}r_{12} & \dots & a_{nn} - \ell_{n1}r_{1n} \end{pmatrix}.$$

## 2 Lineare Gleichungssysteme

Mit der rechten unteren Teilmatrix können wir nun in derselben Weise verfahren, um die nächsten Zeilen und Spalten der Faktoren  $\mathbf{L}$  und  $\mathbf{R}$  zu konstruieren, bis wir schließlich das Ergebnis

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ \ell_{21} & r_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ \ell_{n1} & \cdots & \ell_{n,n-1} & r_{nn} \end{pmatrix}, \quad (2.7)$$

konstruiert haben, das die vollständige LR-Zerlegung in sehr kompakter Form beschreibt.

Man könnte diesen Algorithmus rekursiv umsetzen, so wie wir es bei dem *mergesort*-Algorithmus und der schnellen Fourier-Transformation getan haben, aber in diesem Fall ist es einfacher und effizienter, eine Variable  $k$  einzuführen, die angibt, welche Zeile und Spalte der Matrizen  $\mathbf{L}$  und  $\mathbf{R}$  als nächstes zu berechnen ist. Dann erhalten wir die folgende Rechenvorschrift:

```

procedure lr_decomp( $n$ , var  $\mathbf{A}$ );
for  $k = 1$  to  $n$  do begin
  for  $i \in \{k + 1, \dots, n\}$  do
     $a_{ik} \leftarrow a_{ik}/a_{kk};$ 
  for  $i, j \in \{k + 1, \dots, n\}$  do
     $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$ 
end

```

Natürlich müssen wir auch in diesem Fall untersuchen, wie hoch der Rechenaufwand ist. Indem wir zählen, wieviele Rechenoperationen in den Schleifen auftreten und wie oft sie durchlaufen werden, erhalten wir

$$\begin{aligned} \sum_{k=1}^n ((n-k) + 2(n-k)^2) &= \sum_{k=0}^{n-1} k + 2 \sum_{k=0}^{n-1} k^2 = \frac{n(n-1)}{2} + 2 \frac{n(n-1)(2n-1)}{6} \\ &= \frac{3n(n-1) + 2n(n-1)(2n-1)}{6} = \frac{n(n-1)(4n+1)}{6} \\ &= \frac{n(4n^2 - 4n + n - 1)}{6} < \frac{2}{3}n^3 \end{aligned}$$

Operationen. Der kubische Rechenaufwand wird dadurch relativiert, dass wir ihn nur einmal für jede Matrix betreiben müssen: Sobald die LR-Zerlegung vorliegt, erfordert das Lösen für eine beliebige rechte Seite nur noch quadratischen Aufwand.

Der Aufwand lässt sich weiter reduzieren, wenn wir besondere Eigenschaften der Matrix ausnutzen können. Ein Beispiel ist die Tridiagonalmatrix, die wir in unserem Beispiel (2.3) kennen gelernt haben: Da  $a_{31} = a_{41} = \dots = a_{n1} = 0$  und  $a_{13} = a_{14} = \dots = a_{1n} = 0$  gelten, müssen wir in unserem Algorithmus diese Werte nicht berücksichtigen und können den ersten Schritt mit nur drei Rechenoperationen ausführen. Da auch die verbleibende Teilmatrix tridiagonal ist, stellen wir fest, dass insgesamt nicht mehr als  $3n$  Operationen erforderlich sind. Entsprechend können wir auch das Vorwärts- und

Rückwärtseinsetzen mit jeweils nicht mehr als  $3n$  Operationen durchführen, erhalten also einen Lösungsalgorithmus, dessen Rechenaufwand *linear* mit der Matrixdimension wächst. Viel effizienter kann ein Algorithmus nicht sein, der  $n$  Koeffizienten des Ergebnisvektors  $\mathbf{x}$  berechnet.

Jetzt können wir uns einem Punkt widmen, den wir bisher vernachlässigt haben: Unsere Konstruktion ist nur durchführbar, falls  $a_{11} \neq 0$  gilt, denn bei der Berechnung von  $\mathbf{L}_{*1}$  wird durch  $r_{11} = a_{11}$  dividiert. Da der Algorithmus auch auf Teilmatrizen angewendet wird, muss sicher gestellt sein, dass auch bei allen auftretenden Teilmatrizen diese Bedingung erfüllt ist.

In diesem Zusammenhang wird häufig der Fehler begangen, nicht zwischen den Diagonalelementen der Matrix  $\mathbf{A}$  und den Diagonalelementen der Matrizen  $\mathbf{A}_{**}$  zu unterscheiden. Diese Unterscheidung ist allerdings wichtig: Wenn wir die Beispiele

$$\mathbf{B} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} 1 & 2 \\ 2 & 0 \end{pmatrix}$$

untersuchen, stellen wir fest, dass nach dem ersten Schritt unseres Algorithmus  $\mathbf{B}_{**} = 0$  und  $\mathbf{C}_{**} = -4$  gelten. Im ersten Fall besitzt also  $\mathbf{B}_{**}$  eine Null auf der Diagonalen, obwohl  $\mathbf{B}$  das nicht tut. Im zweiten Fall besitzt zwar  $\mathbf{C}$  eine Null auf der Diagonalen, aber  $\mathbf{C}_{**}$  nicht.

In der Praxis sind wir natürlich daran interessiert, ein Verfahren zu benutzen, das für alle regulären Matrizen  $\mathbf{A}$  funktioniert. Beispielsweise ist das System

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

problemlos lösbar und die Matrix auch regulär, aber da der linke obere Eintrag gleich null ist, können wir keine LR-Zerlegung finden. Es gibt allerdings einen einfachen Ausweg: Wir tauschen die erste und zweite Zeile des Systems, um

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

zu erhalten. Die in diesem System auftretende Matrix besitzt eine LR-Zerlegung. Wir dürfen also hoffen, dass wir durch das Umordnen der Zeilen der Matrix zu einem allgemein einsetzbaren Verfahren kommen.

**Definition 2.5 (Permutationsmatrix)** Sei  $\mathbf{P} \in \mathbb{R}^{n \times n}$ . Falls in jeder Zeile von  $\mathbf{P}$  genau ein Eintrag gleich eins und alle anderen gleich null sind und falls dasselbe auch für jede Spalte gilt, heißt  $\mathbf{P}$  Permutationsmatrix.

**Bemerkung 2.6 (Permutation)** Unter einer Permutation versteht man eine Abbildung  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , die bijektiv ist, also lediglich eine Umordnung der Zahlen von 1 bis  $n$  beschreibt.

Wenn  $\mathbf{P}$  eine Permutationsmatrix ist, können wir

$$\pi(j) = i \quad \text{für alle } i, j \in \{1, \dots, n\} \text{ mit } p_{ij} = 1$$

## 2 Lineare Gleichungssysteme

definieren. Da jede Spalte der Matrix  $\mathbf{P}$  nur genau eine Eins enthält, ist  $\pi$  wohldefiniert. Da jede Zeile nur genau eine Eins enthält, ist  $\pi$  auch injektiv und damit eine Permutation. Für einen beliebigen Vektor  $\mathbf{x} \in \mathbb{R}^n$  gilt

$$\mathbf{P}\mathbf{x} = \begin{pmatrix} x_{\pi(1)} \\ \vdots \\ x_{\pi(n)} \end{pmatrix},$$

also beschreibt die Matrix  $\mathbf{P}$  gerade die Umordnung der Zeilen des Vektors entsprechend der zugehörigen Permutation  $\pi$ .

**Lemma 2.7 (Permutationsmatrizen)** Seien  $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{n \times n}$  Permutationsmatrizen. Dann ist auch ihr Produkt  $\mathbf{PQ}$  eine Permutationsmatrix.

*Beweis.* Sei  $i \in \{1, \dots, n\}$ . Da  $\mathbf{P}$  eine Permutationsmatrix ist, existiert genau ein  $\hat{j} \in \{1, \dots, n\}$  mit  $p_{i\hat{j}} = 1$ . Da  $\mathbf{Q}$  ebenfalls eine Permutationsmatrix ist, existiert genau ein  $\hat{k} \in \{1, \dots, n\}$  mit  $q_{\hat{j}\hat{k}} = 1$ , und wir erhalten

$$(\mathbf{PQ})_{ik} = \sum_{j=1}^n p_{ij}q_{jk} = q_{\hat{j}k} = \begin{cases} 1 & \text{falls } k = \hat{k}, \\ 0 & \text{ansonsten} \end{cases} \quad \text{für alle } k \in \{1, \dots, n\},$$

also enthält die  $i$ -te Zeile des Produkts nur genau eine Eins und ist ansonsten gleich null. Entsprechend können wir mit den Spalten verfahren. ■

Unser Ziel ist es nun, mit Hilfe einer geeigneten Permutation LR-Zerlegungen für beliebige reguläre Matrizen zu konstruieren. Man spricht von einer *LR-Zerlegung mit Pivotsuche* oder einer *pivotisierten LR-Zerlegung*.

**Satz 2.8 (LR-Zerlegung mit Pivotsuche)** Sei  $\mathbf{A}$  eine reguläre Matrix. Dann existiert eine Permutationsmatrix  $\mathbf{P} \in \mathbb{R}^{n \times n}$  so, dass die Matrix  $\mathbf{PA}$  eine LR-Zerlegung  $(\mathbf{L}, \mathbf{R})$  besitzt, dass also

$$\mathbf{PA} = \mathbf{LR}$$

mit einer unteren Dreiecksmatrix  $\mathbf{L}$  und einer oberen Dreiecksmatrix  $\mathbf{R}$  gilt.

*Beweis.* Wir führen den Beweis per Induktion über  $n \in \mathbb{N}$ . Für  $n = 1$  setzen wir  $p_{11} = 1$ ,  $\ell_{11} = 1$  und  $r_{11} = a_{11}$  und sind fertig.

Sei nun  $n \in \mathbb{N}_{\geq 2}$  so gegeben, dass die Behauptung für alle Matrizen  $\mathbf{A} \in \mathbb{R}^{(n-1) \times (n-1)}$  gilt. Sei  $\mathbf{A} \in \mathbb{R}^{n \times n}$  regulär. Um den bereits bekannten Zugang anwenden zu können, müssen wir dafür sorgen, dass der Eintrag  $a_{11}$  nicht gleich null ist. Dieses Ziel erreichen wir, indem wir ein  $i_* \in \{1, \dots, n\}$  mit

$$|a_{i_*1}| \geq |a_{i1}| \quad \text{für alle } i \in \{1, \dots, n\}$$



wählen und mit Hilfe einer Permutationsmatrix die  $i_*$ -te Zeile mit der ersten Zeile vertauschen. Dazu verwenden wir  $\widehat{\mathbf{P}} \in \mathbb{R}^{n \times n}$  mit

$$\widehat{p}_{ij} = \begin{cases} 1 & \text{falls } i = j, i \notin \{1, i_*\}, \\ 1 & \text{falls } i = 1, j = i_* \text{ oder } i = i_*, j = 1, \\ 0 & \text{ansonsten} \end{cases} \quad \text{für alle } i, j \in \{1, \dots, n\},$$

denn diese Matrix leistet genau das Geforderte. Nun untersuchen wir die Matrix

$$\widehat{\mathbf{A}} := \widehat{\mathbf{P}}\mathbf{A}.$$

Da  $\widehat{a}_{11} = a_{i_*1}$  das betragsgrößte Element der ersten Spalte ist, kann es nur gleich null sein, wenn die gesamte Spalte gleich null ist. Das wäre aber ein Widerspruch zu der Regularität von  $\mathbf{A}$ , also muss  $\widehat{a}_{11} \neq 0$  gelten. Somit können wir wie zuvor verfahren, indem wir

$$\widehat{\mathbf{A}} = \begin{pmatrix} \widehat{a}_{11} & \mathbf{A}_{1*} \\ \mathbf{A}_{*1} & \mathbf{A}_{**} \end{pmatrix}, \quad \widehat{\mathbf{L}} = \begin{pmatrix} \ell_{11} & \\ \mathbf{L}_{*1} & \mathbf{L}_{**} \end{pmatrix}, \quad \widehat{\mathbf{R}} = \begin{pmatrix} r_{11} & \mathbf{R}_{1*} \\ & \mathbf{R}_{**} \end{pmatrix}$$

einführen, wieder

$$\ell_{11} := 1, \quad r_{11} := \widehat{a}_{11} = a_{k1}, \quad \mathbf{L}_{*1} := \mathbf{A}_{*1}/\widehat{a}_{11}, \quad \mathbf{R}_{1*} := \mathbf{A}_{1*}$$

setzen und nach einer LR-Zerlegung der Teilmatrix

$$\mathbf{B} := \mathbf{A}_{**} - \mathbf{L}_{*1}\mathbf{R}_{1*} \in \mathbb{R}^{(n-1) \times (n-1)}$$

suchen. Im Allgemeinen braucht eine derartige Zerlegung nicht zu existieren, aber falls wir zeigen können, dass  $\mathbf{B}$  regulär ist, könnten wir uns auf die Induktionsvoraussetzung berufen, um immerhin eine pivotisierte LR-Zerlegung zu finden.

Da  $\mathbf{A}$  und  $\widehat{\mathbf{P}}$  regulär sind, gilt dasselbe auch für

$$\widehat{\mathbf{P}}\mathbf{A} = \widehat{\mathbf{A}} = \begin{pmatrix} \widehat{a}_{11} & \mathbf{A}_{1*} \\ \mathbf{A}_{*1} & \mathbf{A}_{**} \end{pmatrix} = \begin{pmatrix} \ell_{11} & \\ \mathbf{L}_{*1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} r_{11} & \mathbf{R}_{1*} \\ & \mathbf{B} \end{pmatrix},$$

also muss der rechte Faktor

$$\begin{pmatrix} r_{11} & \mathbf{R}_{1*} \\ & \mathbf{B} \end{pmatrix}$$

regulär sein, also insbesondere surjektiv. Damit muss auch  $\mathbf{B}$  surjektiv sein, also regulär.

Nun dürfen wir die Induktionsvoraussetzung anwenden, die die Existenz einer Permutationsmatrix  $\mathbf{P}_{**} \in \mathbb{R}^{(n-1) \times (n-1)}$  und einer LR-Zerlegung  $(\mathbf{L}_{**}, \mathbf{R}_{**})$  mit

$$\mathbf{P}_{**}\mathbf{B} = \mathbf{L}_{**}\mathbf{R}_{**}$$

garantiert. Mit Hilfe dieser Matrizen können wir nun die gesuchte Zerlegung konstruieren: Wir setzen

$$\mathbf{P} := \begin{pmatrix} 1 & \\ & \mathbf{P}_{**} \end{pmatrix} \widehat{\mathbf{P}}, \quad \mathbf{L} := \begin{pmatrix} \ell_{11} & \\ \mathbf{P}_{**}\mathbf{L}_{*1} & \mathbf{L}_{**} \end{pmatrix}, \quad \mathbf{R} := \begin{pmatrix} r_{11} & \mathbf{R}_{1*} \\ & \mathbf{R}_{**} \end{pmatrix}$$

## 2 Lineare Gleichungssysteme

und erhalten

$$\begin{aligned}
 \mathbf{PA} &= \begin{pmatrix} 1 & \\ & \mathbf{P}_{**} \end{pmatrix} \widehat{\mathbf{P}}\mathbf{A} = \begin{pmatrix} 1 & \\ & \mathbf{Q} \end{pmatrix} \begin{pmatrix} \hat{a}_{11} & \mathbf{A}_{1*} \\ \mathbf{A}_{*1} & \mathbf{A}_{**} \end{pmatrix} = \begin{pmatrix} \hat{a}_{11} & \mathbf{A}_{1*} \\ \mathbf{P}_{**}\mathbf{A}_{*1} & \mathbf{QA}_{**} \end{pmatrix} \\
 &= \begin{pmatrix} \ell_{11}r_{11} & \ell_{11}\mathbf{R}_{1*} \\ \mathbf{P}_{**}\mathbf{L}_{*1}r_{11} & \mathbf{P}_{**}\mathbf{B} + \mathbf{P}_{**}\mathbf{L}_{*1}\mathbf{R}_{1*} \end{pmatrix} = \begin{pmatrix} \ell_{11}r_{11} & \ell_{11}\mathbf{R}_{1*} \\ \mathbf{P}_{**}\mathbf{L}_{*1}r_{11} & \mathbf{L}_{**}\mathbf{R}_{**} + \mathbf{P}_{**}\mathbf{L}_{*1}\mathbf{R}_{1*} \end{pmatrix} \\
 &= \begin{pmatrix} \ell_{11} & \\ \mathbf{P}_{**}\mathbf{L}_{*1} & \mathbf{L}_{**} \end{pmatrix} \begin{pmatrix} r_{11} & \mathbf{R}_{1*} \\ & \mathbf{R}_{**} \end{pmatrix} = \mathbf{LR}.
 \end{aligned}$$

Nach Lemma 2.7 ist  $\mathbf{P}$  als Produkt zweier Permutationsmatrizen auch eine Permutationsmatrix, also ist der Induktionsbeweis vollständig. ■

Für das Lösen eines linearen Gleichungssystems ist die pivotisierte LR-Zerlegung genauso nützlich wie die ursprüngliche: Wir erhalten

$$\mathbf{Ax} = \mathbf{b} \iff \mathbf{PAx} = \mathbf{Pb} \iff \mathbf{LRx} = \mathbf{Pb} \iff \mathbf{Ly} = \mathbf{Pb}, \mathbf{Rx} = \mathbf{y}$$

und können das System wie zuvor mit Vorwärts- und Rückwärtseinsetzen auflösen.

Der Algorithmus für die Berechnung der Zerlegung muss nur geringfügig modifiziert werden: Wir müssen für jede Teilmatrix den betragsgrößten Eintrag der  $k$ -ten Spalte berechnen und dann die Zeilen von  $\mathbf{A}$  und  $\mathbf{L}$  vertauschen.

```

procedure lr_pivot( $n$ , var  $\mathbf{A}$ ,  $\mathbf{p}$ );
for  $k = 1$  to  $n$  do begin
     $i_* \leftarrow k$ ;                                     { Finde maximales Element }
    for  $i \in \{k + 1, \dots, n\}$  do
        if  $|a_{ik}| > |a_{i_*k}|$  then  $i_* \leftarrow i$ ;
     $p_k \leftarrow i_*$ ;
    for  $j \in \{1, \dots, n\}$  do begin                 { Tausche Zeilen }
         $\gamma \leftarrow a_{kj}$ ;    $a_{kj} \leftarrow a_{i_*j}$ ;    $a_{i_*j} \leftarrow \gamma$ 
    end;
    for  $i \in \{k + 1, \dots, n\}$  do
         $a_{ik} \leftarrow a_{ik}/a_{kk}$ ;                   {  $\ell_{ik} \leftarrow a_{ik}/r_{kk}$  }
    for  $i, j \in \{k + 1, \dots, n\}$  do
         $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$              {  $a_{ij} \leftarrow a_{ij} - \ell_{ik}r_{kj}$  }
end

```

Der Algorithmus protokolliert in dem Vektor  $\mathbf{p}$ , welche Vertauschungen während der Berechnung durchgeführt wurden, damit sie später auch für die rechte Seite  $\mathbf{b}$  durchgeführt werden können.

Da wir nur arithmetische Operationen zählen, ist der Rechenaufwand der LR-Zerlegung mit Pivotsuche nicht höher als der der gewöhnlichen LR-Zerlegung, und auch das Auflösen des Gleichungssystems mit Vorwärts- und Rückwärtseinsetzen wird nicht aufwendiger.

## 2.5 Fehlerverstärkung

Die LR-Zerlegung mit Pivotsuche ermöglicht es uns, beliebige lineare Gleichungssysteme zu lösen, sofern die Matrix regulär ist. Allerdings haben wir bisher einen Punkt nicht berücksichtigt, der für die Praxis von erheblicher Bedeutung ist: Der Computer rechnet nicht mit „echten“ reellen Zahlen, sondern mit Approximationen, die nur endlich viele Stellen aufweisen können. Damit wird bei fast jeder Rechenoperation ein zusätzlicher Fehler, der sogenannte *Rundungsfehler*, eingeführt, und wir sollten unsere Algorithmen so entwerfen, dass diese Fehler sich nicht so weit verstärken, dass sie das Ergebnis unbrauchbar machen.

Da die detaillierte Analyse der bei den bisher diskutierten Algorithmen auftretenden Fehlerfortpflanzung relativ aufwendig ist, beschränken wir uns hier darauf, lediglich die Verstärkung eines Fehlers auf der rechten Seite des Gleichungssystems zu analysieren. Wir verwenden dazu die durch

$$\|\mathbf{y}\|_2 := \left( \sum_{i=1}^n y_i^2 \right)^{1/2} \quad \text{für alle } \mathbf{y} \in \mathbb{R}^n \quad (2.8)$$

definierte *euklidische Norm*, die anschaulich als ein Maß für die Länge eines Vektors interpretiert werden kann.

**Lemma 2.9 (Norm des Matrix-Vektor-Produkts)** Sei  $\mathbf{A} \in \mathbb{R}^{m \times n}$  gegeben. Dann sind

$$\alpha_2(\mathbf{A}) := \min\{\|\mathbf{A}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\}, \quad (2.9a)$$

$$\beta_2(\mathbf{A}) := \max\{\|\mathbf{A}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} \quad (2.9b)$$

wohldefinierte nicht-negative Zahlen und erfüllen

$$\alpha_2(\mathbf{A})\|\mathbf{z}\|_2 \leq \|\mathbf{A}\mathbf{z}\|_2 \leq \beta_2(\mathbf{A})\|\mathbf{z}\|_2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n. \quad (2.10)$$

Falls  $\mathbf{A}$  injektiv ist (falls also sein Kern nur den Nullvektor enthält), gilt  $\alpha_2(\mathbf{A}) > 0$ .

*Beweis.* Die Einheitssphäre

$$S := \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}\|_2 = 1\}$$

ist beschränkt und abgeschlossen, also nach dem Satz von Heine-Borel auch kompakt. Da die Abbildung

$$f : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}, \quad \mathbf{y} \mapsto \|\mathbf{A}\mathbf{y}\|_2,$$

stetig ist, muss auch

$$f(S) = \{f(\mathbf{y}) : \mathbf{y} \in S\} = \{\|\mathbf{A}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\}$$

## 2 Lineare Gleichungssysteme

kompakt sein. Demzufolge besitzt sie ein Minimum  $\alpha$  und ein Maximum  $\beta$ , und beide sind endlich und nicht-negativ.

Sei nun  $\mathbf{z} \in \mathbb{R}^n$ . Falls  $\mathbf{z} = \mathbf{0}$  gilt, ist (2.10) offenbar erfüllt. Anderenfalls ist  $\mathbf{y} := \mathbf{z}/\|\mathbf{z}\|_2$  ein Vektor mit  $\|\mathbf{y}\|_2 = \|\mathbf{z}\|_2/\|\mathbf{z}\|_2 = 1$ , so dass wir (2.9) verwenden können, um

$$\begin{aligned}\alpha_2(\mathbf{A})\|\mathbf{z}\|_2 &\leq \|\mathbf{A}\mathbf{y}\|_2\|\mathbf{z}\|_2 = \|\mathbf{A}\mathbf{z}\|_2 \frac{\|\mathbf{z}\|_2}{\|\mathbf{z}\|_2} = \|\mathbf{A}\mathbf{z}\|_2, \\ \beta_2(\mathbf{A})\|\mathbf{z}\|_2 &\geq \|\mathbf{A}\mathbf{y}\|_2\|\mathbf{z}\|_2 = \|\mathbf{A}\mathbf{z}\|_2 \frac{\|\mathbf{z}\|_2}{\|\mathbf{z}\|_2} = \|\mathbf{A}\mathbf{z}\|_2\end{aligned}$$

zu erhalten, also (2.10).

Sei nun  $\mathbf{A}$  injektiv. Sei  $\mathbf{y} \in \mathbb{R}^n$  ein Vektor mit  $\|\mathbf{y}\|_2 = 1$ , für den das Minimum in (2.9a) angenommen wird. Da  $\mathbf{A}$  injektiv und  $\mathbf{y} \neq \mathbf{0}$  ist, muss auch  $\mathbf{A}\mathbf{y} \neq \mathbf{0}$  gelten, also

$$\alpha_2(\mathbf{A}) = \|\mathbf{A}\mathbf{y}\|_2 > 0.$$

■

Die Zahlen  $\alpha_2(\mathbf{A})$  und  $\beta_2(\mathbf{A})$  beschreiben also, wie sich die Längen von Vektoren verändern, wenn sie mit der Matrix  $\mathbf{A}$  multipliziert werden. Mit Hilfe dieser Größen lassen sich Aussagen darüber treffen, wie sich Störungen der rechten Seite  $\mathbf{b}$  auf die Lösung  $\mathbf{x}$  des Gleichungssystems (2.4) auswirken.

**Lemma 2.10 (Störung der rechten Seite)** *Sei  $\mathbf{A} \in \mathbb{R}^{n \times n}$  eine reguläre Matrix. Seien  $\mathbf{b}, \tilde{\mathbf{b}} \in \mathbb{R}^n$  gegeben, sei  $\mathbf{b} \neq \mathbf{0}$ , und seien  $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^n$  die Lösungen der Gleichungssysteme*

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}.$$

Dann gilt

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq \kappa_2(\mathbf{A}) \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2} \quad \text{mit} \quad \kappa_2(\mathbf{A}) := \frac{\beta_2(\mathbf{A})}{\alpha_2(\mathbf{A})}.$$

*Beweis.* Mit (2.10) erhalten wir

$$\begin{aligned}\alpha_2(\mathbf{A})\|\mathbf{x} - \tilde{\mathbf{x}}\|_2 &\leq \|\mathbf{A}(\mathbf{x} - \tilde{\mathbf{x}})\|_2 = \|\mathbf{A}\mathbf{x} - \mathbf{A}\tilde{\mathbf{x}}\|_2 = \|\mathbf{b} - \tilde{\mathbf{b}}\|_2, \\ \beta_2(\mathbf{A})\|\mathbf{x}\|_2 &\geq \|\mathbf{A}\mathbf{x}\|_2 = \|\mathbf{b}\|_2 > 0,\end{aligned}$$

so dass wir schließlich mit

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} = \kappa_2(\mathbf{A}) \frac{\alpha_2(\mathbf{A})\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\beta_2(\mathbf{A})\|\mathbf{x}\|_2} \leq \kappa_2(\mathbf{A}) \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2}$$

das angestrebte Ergebnis erhalten. ■

Diese Abschätzung bedeutet, dass der *relative Fehler* der Lösung sich durch das Produkt aus dem relativen Fehler der rechten Seite und der *Konditionszahl*  $\kappa_2(\mathbf{A})$  abschätzen

lässt. Wenn wir also daran interessiert sind, eine möglichst gute Näherung der Lösung zu berechnen, sollten wir versuchen, die Konditionszahl möglichst klein zu halten.

In der Praxis wird häufig nicht nur die rechte Seite  $\mathbf{b}$  Störungen unterliegen, sondern auch die Matrix  $\mathbf{A}$ . Wenn wir die Auswirkungen dieser letzteren Störungen auf die Lösung analysieren wollen, brauchen wir ein Maß dafür, wie sehr Matrizen voneinander abweichen, beispielsweise die folgende Norm:

**Definition 2.11 (Spektralnorm)** *Die Abbildung*

$$\|\cdot\|_2 : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}_{\geq 0}, \quad \mathbf{A} \mapsto \max \left\{ \frac{\|\mathbf{A}\mathbf{z}\|_2}{\|\mathbf{z}\|_2} : \mathbf{z} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \right\} = \beta_2(\mathbf{A}), \quad (2.11)$$

ist eine Norm auf dem Raum  $\mathbb{R}^{m \times n}$ , die als die Spektralnorm bezeichnet wird. Wegen (2.10) ist sie mit der euklidischen Norm  $\|\cdot\|_2$  verträglich, erfüllt also

$$\|\mathbf{A}\mathbf{y}\|_2 \leq \|\mathbf{A}\|_2 \|\mathbf{y}\|_2 \quad \text{für alle } \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^n. \quad (2.12)$$

Um das Verhalten der Inversen unter Störungen zu analysieren, brauchen wir Aussagen über die Norm des Produkts zweier Matrizen.

**Lemma 2.12 (Matrix-Produkt)** *Seien  $\mathbf{A} \in \mathbb{R}^{n \times m}$  und  $\mathbf{B} \in \mathbb{R}^{m \times k}$  gegeben. Dann gelten*

$$\alpha_2(\mathbf{AB}) \geq \alpha_2(\mathbf{A})\alpha_2(\mathbf{B}), \quad \beta_2(\mathbf{AB}) \leq \beta_2(\mathbf{A})\beta_2(\mathbf{B}), \quad \|\mathbf{AB}\|_2 \leq \|\mathbf{A}\|_2 \|\mathbf{B}\|_2. \quad (2.13)$$

*Beweis.* Indem wir (2.10) mit (2.9) kombinieren, erhalten wir

$$\begin{aligned} \alpha_2(\mathbf{AB}) &= \min\{\|\mathbf{AB}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} \\ &\geq \min\{\alpha_2(\mathbf{A})\|\mathbf{B}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} = \alpha_2(\mathbf{A})\alpha_2(\mathbf{B}), \\ \beta_2(\mathbf{AB}) &= \max\{\|\mathbf{AB}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} \\ &\leq \max\{\beta_2(\mathbf{A})\|\mathbf{B}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} = \beta_2(\mathbf{A})\beta_2(\mathbf{B}). \end{aligned}$$

Mit  $\|\mathbf{A}\|_2 = \beta_2(\mathbf{A})$ ,  $\|\mathbf{B}\|_2 = \beta_2(\mathbf{B})$  und  $\|\mathbf{AB}\|_2 = \beta_2(\mathbf{AB})$  folgt die letzte Aussage aus der bereits bewiesenen. ■

Mit Hilfe der Spektralnorm lassen sich auch *Störungen der Matrix* messen und ihre Auswirkung abschätzen. Ein nützliches Hilfsmittel ist die folgende Aussage:

**Lemma 2.13 (Neumannsche Reihe)** *Sei  $\mathbf{X} \in \mathbb{R}^{n \times n}$  eine Matrix mit  $\|\mathbf{X}\|_2 < 1$ . Dann ist  $\mathbf{I} - \mathbf{X}$  regulär und es gilt*

$$\|(\mathbf{I} - \mathbf{X})^{-1}\|_2 \leq \frac{1}{1 - \|\mathbf{X}\|_2}. \quad (2.14)$$

*Beweis.* Für alle  $m \in \mathbb{N}_0$  definieren wir die Partialsumme

$$\mathbf{Y}_m := \sum_{k=0}^m \mathbf{X}^k.$$

## 2 Lineare Gleichungssysteme

Aus (2.13) folgt mit einer einfachen Induktion

$$\|\mathbf{X}^k\|_2 \leq \|\mathbf{X}\|_2^k \quad \text{für alle } k \in \mathbb{N}_0,$$

also erhalten wir wegen  $\|\mathbf{X}\|_2 < 1$  auch

$$\lim_{m \rightarrow \infty} \|\mathbf{X}^m\|_2 \leq \lim_{m \rightarrow \infty} \|\mathbf{X}\|_2^m = 0, \quad (2.15a)$$

$$\|\mathbf{Y}_m\|_2 = \left\| \sum_{k=0}^m \mathbf{X}^k \right\|_2 \leq \sum_{k=0}^{\infty} \|\mathbf{X}\|_2^k = \frac{1}{1 - \|\mathbf{X}\|_2}, \quad (2.15b)$$

wobei wir für die zweite Gleichung die geometrische Summenformel verwendet haben.

Wir müssen zeigen, dass die Folge der Partialsummen  $\mathbf{Y}_m$  konvergiert. Das tun wir, indem wir nachweisen, dass es sich um eine Cauchy-Folge handelt. Seien  $m, \ell \in \mathbb{N}_0$  mit  $m < \ell$  gegeben. Dann gilt

$$\begin{aligned} \|\mathbf{Y}_\ell - \mathbf{Y}_m\|_2 &= \left\| \sum_{k=0}^{\ell} \mathbf{X}^k - \sum_{k=0}^m \mathbf{X}^k \right\|_2 = \left\| \sum_{k=m+1}^{\ell} \mathbf{X}^k \right\|_2 \\ &\leq \sum_{k=m+1}^{\ell} \|\mathbf{X}^k\|_2 \leq \sum_{k=m+1}^{\ell} \|\mathbf{X}\|_2^k = \|\mathbf{X}\|_2^{m+1} \sum_{k=m+1}^{\ell} \|\mathbf{X}\|_2^{k-m-1} \\ &= \|\mathbf{X}\|_2^{m+1} \sum_{k=0}^{\ell-m-1} \|\mathbf{X}\|_2^k \leq \|\mathbf{X}\|_2^{m+1} \sum_{k=0}^{\infty} \|\mathbf{X}\|_2^k = \frac{\|\mathbf{X}\|_2^{m+1}}{1 - \|\mathbf{X}\|_2}, \end{aligned}$$

wobei wir im letzten Schritt (2.15b) verwendet haben. Für ein beliebiges  $\epsilon \in \mathbb{R}_{>0}$  können wir wegen  $\|\mathbf{X}\|_2 < 1$  ein  $m_0 \in \mathbb{N}_0$  mit  $\|\mathbf{X}\|_2^{m_0+1} < \epsilon(1 - \|\mathbf{X}\|_2)$  finden, und für alle  $m, \ell \in \mathbb{N}_0$  mit  $m_0 \leq m \leq \ell$  folgt dann  $\|\mathbf{Y}_\ell - \mathbf{Y}_m\|_2 < \epsilon$ . Also ist  $(\mathbf{Y}_m)_{m=0}^{\infty}$  eine Cauchy-Folge. Da der Raum der Matrizen  $\mathbb{R}^{n \times n}$  ein Banach-Raum ist, besitzt diese Folge einen Grenzwert  $\mathbf{Y} := \lim_{m \rightarrow \infty} \mathbf{Y}_m$ . Es gelten

$$\begin{aligned} (\mathbf{I} - \mathbf{X})\mathbf{Y}_m &= (\mathbf{I} - \mathbf{X}) \sum_{k=0}^m \mathbf{X}^k = \sum_{k=0}^m \mathbf{X}^k - \sum_{k=0}^m \mathbf{X}^{k+1} = \mathbf{I} - \mathbf{X}^{m+1}, \\ (\mathbf{I} - \mathbf{X})\mathbf{Y} &= \lim_{m \rightarrow \infty} (\mathbf{I} - \mathbf{X})\mathbf{Y}_m = \lim_{m \rightarrow \infty} \mathbf{I} - \mathbf{X}^{m+1}, \end{aligned}$$

und wegen (2.15a) konvergiert die rechte Seite für  $m \rightarrow \infty$  gegen  $\mathbf{I}$ , so dass wir insgesamt  $(\mathbf{I} - \mathbf{X})\mathbf{Y} = \mathbf{I}$  erhalten. Also ist  $\mathbf{Y}$  die Inverse der Matrix  $\mathbf{I} - \mathbf{X}$ . Die Abschätzung von  $\|(\mathbf{I} - \mathbf{X})^{-1}\|_2 = \|\mathbf{Y}\|_2$  folgt unmittelbar aus (2.15b). ■

**Bemerkung 2.14 (Störung invertierbarer Matrizen)** Sei  $\mathbf{A} \in \mathbb{R}^{n \times n}$  eine invertierbare Matrix und  $\mathbf{b} \in \mathbb{R}^n$ . Sei  $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$  eine weitere Matrix, bei der wir uns vorstellen, dass sie aus  $\mathbf{A}$  durch eine kleine Störung hervorgegangen ist.

Damit auch das gestörte Gleichungssystem  $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}$  lösbar ist, sollte  $\tilde{\mathbf{A}}$  invertierbar sein. Da  $\mathbf{A}$  invertierbar ist, ist das genau dann der Fall, wenn das Produkt  $\mathbf{A}^{-1}\tilde{\mathbf{A}}$  invertierbar ist.

Um Lemma 2.13 anwenden zu können, brauchen wir eine Matrix  $\mathbf{X}$  mit

$$\mathbf{I} - \mathbf{X} = \mathbf{A}^{-1}\tilde{\mathbf{A}}, \quad \mathbf{X} = \mathbf{I} - \mathbf{A}^{-1}\tilde{\mathbf{A}} = \mathbf{A}^{-1}(\mathbf{A} - \tilde{\mathbf{A}}).$$

Nach Lemma 2.13 ist  $\tilde{\mathbf{A}}$  invertierbar, falls  $\|\mathbf{X}\|_2 < 1$  gilt.

Mit der Abschätzung (2.13) gilt

$$\|\mathbf{X}\|_2 = \|\mathbf{A}^{-1}(\mathbf{A} - \tilde{\mathbf{A}})\|_2 \leq \|\mathbf{A}^{-1}\|_2 \|\mathbf{A} - \tilde{\mathbf{A}}\|_2,$$

also folgt aus

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_2 < 1/\|\mathbf{A}^{-1}\|_2$$

bereits, dass  $\tilde{\mathbf{A}}$  invertierbar ist. Wenn wir uns die Norm als Maß für den Abstand von Matrizen vorstellen, kann diese Aussage geometrisch interpretiert werden: Jede invertierbare Matrix ist der Mittelpunkt einer offenen Kugel mit Radius  $1/\|\mathbf{A}^{-1}\|_2$ , in der alle Matrizen ebenfalls invertierbar sind.

Wenn wir mit  $\tilde{\mathbf{x}}$  die Lösung des Gleichungssystems  $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}$  bezeichnen, haben wir unter der Annahme  $\|\mathbf{X}\|_2 < 1$  die Gleichungen

$$\begin{aligned} \mathbf{x} &= \mathbf{A}^{-1}\mathbf{b} = \mathbf{A}^{-1}\tilde{\mathbf{A}}\tilde{\mathbf{x}}, \\ \tilde{\mathbf{x}} - \mathbf{x} &= (\mathbf{I} - \mathbf{A}^{-1}\tilde{\mathbf{A}})\tilde{\mathbf{x}} = \mathbf{X}\tilde{\mathbf{x}} = \mathbf{X}(\mathbf{I} - \mathbf{X})^{-1}\mathbf{x}, \\ \frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2}{\|\mathbf{x}\|_2} &\leq \|\mathbf{X}\|_2 \|(\mathbf{I} - \mathbf{X})^{-1}\|_2 \leq \frac{\|\mathbf{X}\|_2}{1 - \|\mathbf{X}\|_2} \leq \frac{\kappa_2(\mathbf{A})}{1 - \|\mathbf{X}\|_2} \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2}, \end{aligned}$$

können also den relativen Fehler der Lösung durch den relativen Fehler der Matrix abschätzen.

In der Praxis tritt häufig der Fall auf, dass sowohl die rechte Seite  $\mathbf{b}$  als auch die Matrix  $\mathbf{A}$  gestört ist. Auch in diesem Fall lässt sich noch eine Aussage über den Einfluss der Störungen auf die Lösung treffen.

**Satz 2.15 (Gestörte lineare Gleichungssysteme)** Seien  $\mathbf{A}, \tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$ ,  $\mathbf{b}, \tilde{\mathbf{b}} \in \mathbb{R}^n$ .

Sei  $\mathbf{A}$  invertierbar und gelte

$$\|\mathbf{A}^{-1}(\mathbf{A} - \tilde{\mathbf{A}})\|_2 < 1.$$

Dann ist auch  $\tilde{\mathbf{A}}$  invertierbar und die Lösungen  $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^n$  der Gleichungssysteme

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$$

erfüllen die Abschätzung

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq \frac{\|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2}{1 - \|\mathbf{A}^{-1}(\mathbf{A} - \tilde{\mathbf{A}})\|_2} \left( \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2} + \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2} \right). \quad (2.16)$$

## 2 Lineare Gleichungssysteme

*Beweis.* Wir setzen  $\mathbf{X} := \mathbf{A}^{-1}(\mathbf{A} - \tilde{\mathbf{A}})$ . Nach Lemma 2.13 ist  $\mathbf{I} - \mathbf{X} = \mathbf{A}^{-1}\tilde{\mathbf{A}}$  invertierbar, also auch  $\tilde{\mathbf{A}}$ . Für die Lösungen der Gleichungssysteme ergeben sich aus (2.12) und (2.14) die Gleichungen

$$\begin{aligned} (\mathbf{I} - \mathbf{X})(\mathbf{x} - \tilde{\mathbf{x}}) &= \mathbf{A}^{-1}\tilde{\mathbf{A}}(\mathbf{x} - \tilde{\mathbf{x}}) = \mathbf{A}^{-1}(\mathbf{A}\mathbf{x} + (\tilde{\mathbf{A}} - \mathbf{A})\mathbf{x} - \tilde{\mathbf{A}}\tilde{\mathbf{x}}) \\ &= \mathbf{A}^{-1}(\mathbf{b} + (\tilde{\mathbf{A}} - \mathbf{A})\mathbf{x} - \tilde{\mathbf{b}}), \\ \|\mathbf{x} - \tilde{\mathbf{x}}\|_2 &= \|(\mathbf{I} - \mathbf{X})^{-1}\mathbf{A}^{-1}(\mathbf{b} - \tilde{\mathbf{b}} + (\tilde{\mathbf{A}} - \mathbf{A})\mathbf{x})\|_2 \\ &\leq \|(\mathbf{I} - \mathbf{X})^{-1}\|_2\|\mathbf{A}^{-1}\|_2(\|\mathbf{b} - \tilde{\mathbf{b}}\|_2 + \|\mathbf{A} - \tilde{\mathbf{A}}\|_2\|\mathbf{x}\|_2) \\ &\leq \frac{\|\mathbf{A}^{-1}\|_2}{1 - \|\mathbf{X}\|_2} \left( \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{A}\mathbf{x}\|_2} \|\mathbf{A}\|_2\|\mathbf{x}\|_2 + \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2} \|\mathbf{A}\|_2\|\mathbf{x}\|_2 \right) \\ &= \frac{\|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2}{1 - \|\mathbf{X}\|_2} \left( \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2} + \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2} \right) \|\mathbf{x}\|_2. \end{aligned}$$

Per Division durch  $\|\mathbf{x}\|_2$  folgt die Behauptung. ■

**Bemerkung 2.16 (Konditionszahl)** Falls  $\mathbf{A} \in \mathbb{R}^{n \times n}$  invertierbar ist, folgt mit der Substitution  $\mathbf{z} = \mathbf{A}\mathbf{y}$  die Gleichung

$$\begin{aligned} \alpha_2(\mathbf{A}) &= \min\{\|\mathbf{A}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} = \min\left\{ \frac{\|\mathbf{A}\mathbf{y}\|_2}{\|\mathbf{y}\|_2} : \mathbf{y} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \right\} \\ &= \min\left\{ \frac{\|\mathbf{z}\|_2}{\|\mathbf{A}^{-1}\mathbf{z}\|_2} : \mathbf{z} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \right\} = \frac{1}{\max\left\{ \frac{\|\mathbf{A}^{-1}\mathbf{z}\|_2}{\|\mathbf{z}\|_2} : \mathbf{z} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \right\}} \\ &= \frac{1}{\|\mathbf{A}^{-1}\|_2}, \end{aligned}$$

so dass sich die Konditionszahl als

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2$$

schreiben lässt. Indem wir (2.12) mit (2.16) kombinieren, folgt

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq \frac{\kappa_2(\mathbf{A})}{1 - \kappa_2(\mathbf{A})\frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2}} \left( \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2} + \frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\|\mathbf{A}\|_2} \right).$$

Auch bei einer Störung der Matrix ist also die Konditionszahl ein wichtiges Maß für die zu erwartende Fehlerverstärkung.

**Bemerkung 2.17 (Rückwärtsanalyse)** Bei der Realisierung eines numerischen Verfahrens auf einem Computer werden die reellen Zahlen durch Maschinenzahlen approximiert und die Grundrechenarten durch gerundete Operationen  $\oplus, \ominus, \odot, \oslash$ , für die unter normalen Bedingungen eine gewisse relative Genauigkeit garantiert ist.



Als Beispiel untersuchen wir das Lösen eines linearen Gleichungssystems  $Lx = b$  mit einer linken unteren Dreiecksmatrix. Im ersten Schritt würden wir theoretisch  $x_1 = b_1/\ell_{11}$  berechnen. Tatsächlich berechnet der Computer  $\tilde{x}_1 = b_1 \oslash \ell_{11}$ , garantiert aber, dass es ein kleines  $\epsilon_1 \in \mathbb{R}$  mit  $\tilde{x}_1 = (1 + \epsilon_1)b_1/\ell_{11}$  gibt. Wenn wir  $\tilde{b}_1 := (1 + \epsilon_1)b_1$  setzen, entspricht also die exakte Durchführung des ersten Schritts für das gestörte  $\tilde{b}_1$  der tatsächlichen Durchführung für das exakte  $b_1$ .

Im zweiten Schritt würden wir theoretisch  $b_i^{(1)} = b_i - \ell_{i1}x_1$  berechnen. Tatsächlich berechnet der Computer  $\tilde{b}_i^{(1)} = b_i \ominus \ell_{i1} \odot \tilde{x}_1$  und garantiert die Existenz kleiner Zahlen  $\epsilon_i, \epsilon_{i1} \in \mathbb{R}$  mit

$$\begin{aligned} \ell_{i1} \odot \tilde{x}_1 &= (1 + \epsilon_{i1})\ell_{i1}\tilde{x}_1, \\ \tilde{b}_i^{(1)} &= (1 + \epsilon_i)(b_i - \ell_{i1} \odot \tilde{x}_1) = (1 + \epsilon_i)b_i - (1 + \epsilon_i)(1 + \epsilon_{i1})\ell_{i1}\tilde{x}_1. \end{aligned}$$

Wenn wir  $\tilde{b}_i := (1 + \epsilon_i)b_i$  und  $\tilde{\ell}_{i1} := (1 + \epsilon_i)(1 + \epsilon_{i1})\ell_{i1}$  setzen, entspricht also die exakte Berechnung von  $\tilde{b}_i^{(1)}$  für die gestörte rechte Seite und die gestörte Matrix gerade der tatsächlichen Durchführung für das exakte Problem.

Im dritten Schritt wird der Computer  $\mathbf{L}_{**}\mathbf{x}_* = \tilde{\mathbf{b}}^{(1)}$  näherungsweise lösen, und wir können induktiv eine gestörte Matrix  $\tilde{\mathbf{L}}_{**}$  und einen gestörten Vektor  $\tilde{\mathbf{b}}_*$  so finden, dass die tatsächlich berechnete Lösung  $\tilde{\mathbf{x}}_*$  die exakte Lösung des Systems  $\tilde{\mathbf{L}}_{**}\tilde{\mathbf{x}}_* = \tilde{\mathbf{b}}_*$  ist.

Insgesamt ist also die tatsächlich berechnete Lösung des Gleichungssystems die exakte Lösung eines gestörten Gleichungssystems, und wir können Satz 2.15 verwenden, um die Genauigkeit abzuschätzen.

Diese Vorgehensweise bei der Analyse eines numerischen Verfahrens geht auf James H. Wilkinson zurück und ist als Rückwärtsanalyse bekannt: Statt direkt den Fehler zu analysieren, modifiziert man das Problem so, dass die gestörte Lösung dessen exakte Lösung ist und sich elegante mathematische Aussagen zu dessen Analyse verwenden lassen.

## 2.6 QR-Zerlegung

Für eine LR-Zerlegung  $(\mathbf{L}, \mathbf{R})$  der Matrix  $\mathbf{A}$  ist das System (2.4) äquivalent zu

$$\mathbf{L}\mathbf{y} = \mathbf{b}, \quad \mathbf{R}\mathbf{x} = \mathbf{y}.$$

Dank Lemma 2.10 erhalten wir für Störungen der rechten Seite die Abschätzung

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \leq \kappa_2(\mathbf{R}) \frac{\|\mathbf{y} - \tilde{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \leq \kappa_2(\mathbf{R})\kappa_2(\mathbf{L}) \frac{\|\mathbf{b} - \tilde{\mathbf{b}}\|_2}{\|\mathbf{b}\|_2}.$$

Aus (2.10) folgen unmittelbar

$$\alpha_2(\mathbf{A}) \geq \alpha_2(\mathbf{L})\alpha_2(\mathbf{R}), \quad \beta_2(\mathbf{A}) \leq \beta_2(\mathbf{L})\beta_2(\mathbf{R}), \quad \kappa_2(\mathbf{A}) \leq \kappa_2(\mathbf{L})\kappa_2(\mathbf{R}),$$

das Lösen mit Hilfe der LR-Zerlegung kann also zu einer ungünstigeren Fehlerfortpflanzung führen.

## 2 Lineare Gleichungssysteme

Unser Ziel ist es, eine Zerlegung der Matrix  $\mathbf{A}$  zu finden, mit deren Hilfe sich das Gleichungssystem ähnlich einfach wie mit der LR-Zerlegung lösen lässt, die aber die Konditionszahl möglichst unverändert lässt. Da die Konditionszahl von der Norm abhängt, bietet es sich an, nach Transformationen  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  zu suchen, die die Norm unverändert lassen, für die also

$$\|\mathbf{Q}\mathbf{y}\|_2 = \|\mathbf{y}\|_2 \quad \text{für alle } \mathbf{y} \in \mathbb{R}^n \quad (2.17)$$

gilt. Im Allgemeinen sind solche Transformationen nur schwer zu beschreiben, im Fall der euklidischen Norm können wir allerdings deren besondere Eigenschaften ausnutzen: Wenn wir das durch

$$\langle \mathbf{y}, \mathbf{z} \rangle_2 := \sum_{i=1}^n y_i z_i \quad \text{für alle } \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$$

definierte *euklidische Skalarprodukt* verwenden, gilt

$$\|\mathbf{y}\|_2^2 = \sum_{i=1}^n y_i^2 = \langle \mathbf{y}, \mathbf{y} \rangle_2 \quad \text{für alle } \mathbf{y} \in \mathbb{R}^n,$$

die Norm lässt sich also durch das Skalarprodukt darstellen. Eine nützliche Eigenschaft des Skalarprodukts besteht darin, dass es mit dem Transponieren von Matrizen verträglich ist: Wir haben

$$\langle \mathbf{B}\mathbf{y}, \mathbf{z} \rangle_2 = \langle \mathbf{y}, \mathbf{B}^* \mathbf{z} \rangle_2 \quad \text{für alle } \mathbf{B} \in \mathbb{R}^{n \times m}, \mathbf{y} \in \mathbb{R}^m, \mathbf{z} \in \mathbb{R}^n, \quad (2.18)$$

wobei  $\mathbf{B}^*$  die transponierte Matrix zu  $\mathbf{B}$  bezeichnet. Damit lässt sich die gewünschte Eigenschaft (2.17) in die Form

$$\langle \mathbf{y}, \mathbf{y} \rangle_2 = \|\mathbf{y}\|_2^2 = \|\mathbf{Q}\mathbf{y}\|_2^2 = \langle \mathbf{Q}\mathbf{y}, \mathbf{Q}\mathbf{y} \rangle_2 = \langle \mathbf{y}, \mathbf{Q}^* \mathbf{Q}\mathbf{y} \rangle_2 \quad \text{für alle } \mathbf{y} \in \mathbb{R}^n$$

bringen, und aus dieser Gleichung lässt sich schließen, dass  $\mathbf{Q}^* \mathbf{Q} = \mathbf{I}$  gelten muss.

**Definition 2.18 (Orthogonale Matrix)** Eine Matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  heißt orthogonal, falls  $\mathbf{Q}^* \mathbf{Q} = \mathbf{I}$  gilt, falls also ihre Inverse und ihre Transponierte übereinstimmen.

Unser Ziel ist es nun, die Matrix  $\mathbf{A}$  mit Hilfe einer orthogonalen Transformation  $\mathbf{Q}$  auf obere Dreiecksgestalt zu bringen, denn aus  $\mathbf{A} = \mathbf{Q}\mathbf{R}$  folgt insbesondere

$$\begin{aligned} \alpha_2(\mathbf{A}) &= \min\{\|\mathbf{A}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} \\ &= \min\{\|\mathbf{Q}\mathbf{R}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} \\ &= \min\{\|\mathbf{R}\mathbf{y}\|_2 : \mathbf{y} \in \mathbb{R}^n, \|\mathbf{y}\|_2 = 1\} = \alpha_2(\mathbf{R}), \\ \beta_2(\mathbf{A}) &= \dots = \beta_2(\mathbf{R}), \\ \kappa_2(\mathbf{A}) &= \beta_2(\mathbf{A})/\alpha_2(\mathbf{A}) = \beta_2(\mathbf{R})/\alpha_2(\mathbf{R}) = \kappa_2(\mathbf{R}), \end{aligned}$$

also wird die Konditionszahl und damit die Fehlerverstärkung durch diese Transformation nicht verschlechtert werden.

Wir beginnen mit einem einfachen Beispiel: Für einen Vektor  $\mathbf{y} \in \mathbb{R}^2$  suchen wir eine orthogonale Matrix  $\mathbf{Q} \in \mathbb{R}^{2 \times 2}$ , die seine zweite Komponente eliminiert, es soll also

$$0 = (\mathbf{Q}\mathbf{y})_2 = q_{21}y_1 + q_{22}y_2 \quad (2.19)$$

gelten. Wir verwenden dazu eine *Rotationsmatrix*

$$\mathbf{Q} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c^2 + s^2 = 1,$$

denn für diese Matrix gilt

$$\mathbf{Q}^* \mathbf{Q} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \begin{pmatrix} c^2 + s^2 & cs - sc \\ sc - cs & s^2 + c^2 \end{pmatrix} = \mathbf{I},$$

sie ist also orthogonal. Ihren Namen verdankt sie der Tatsache, dass sich ein Winkel  $\varphi$  finden lässt, für den  $c = \cos \varphi$  und  $s = \sin \varphi$  gelten, und dass die Anwendung der Matrix gerade eine Rotation der Ebene um diesen Winkel beschreibt.

Unser Ziel ist es, die Parameter  $c$  und  $s$  so zu wählen, dass (2.19) gilt, also

$$0 = -sy_1 + cy_2.$$

Dieses Ziel lässt sich einfach erreichen, indem wir

$$c := \frac{y_1}{r}, \quad s := \frac{y_2}{r}$$

setzen und den Skalierungsfaktor  $r$  so wählen, dass

$$1 = c^2 + s^2 = \frac{y_1^2 + y_2^2}{r^2}, \quad r = \pm \sqrt{y_1^2 + y_2^2}$$

gilt. Wenn wir allerdings  $r$  mit Hilfe dieser Gleichung berechnen, könnte es Schwierigkeiten geben, falls  $y_1^2$  oder  $y_2^2$  zu groß oder zu klein sind, um sich noch im Rechner darstellen zu lassen.

Dieses Problem lässt sich lösen, indem wir  $|y_1|$  oder  $|y_2|$  aus der Wurzel herausziehen: Falls  $|y_1| \geq |y_2|$  gilt, wählen wir für die Wurzel das Vorzeichen von  $y_1$  und erhalten

$$\begin{aligned} \tau &:= \frac{y_2}{y_1}, \\ r &= \operatorname{sgn}(y_1) \sqrt{y_1^2 + y_2^2} = \operatorname{sgn}(y_1) |y_1| \sqrt{1 + \tau^2} = y_1 \sqrt{1 + \tau^2}, \\ s &= \frac{y_2}{r} = \frac{y_2}{y_1 \sqrt{1 + \tau^2}} = \frac{\tau}{\sqrt{1 + \tau^2}}, \\ c &= \frac{y_1}{r} = \frac{1}{\sqrt{1 + \tau^2}} = \sqrt{\frac{1 + \tau^2 - \tau^2}{1 + \tau^2}} = \sqrt{1 - \frac{\tau^2}{1 + \tau^2}} = \sqrt{1 - s^2}. \end{aligned}$$

Wir stellen fest, dass in diesem Fall nach Voraussetzung  $|\tau| \leq 1$  gilt. Wir können auch direkt  $s = \tau c$  und damit  $2s^2 \leq s^2 + c^2 \leq 1$ , also  $s^2 \leq 1/2$  und  $|s| \leq \sqrt{1/2}$  ablesen.

## 2 Lineare Gleichungssysteme

Falls dagegen  $|y_1| \leq |y_2|$  gilt, geben wir der Wurzel das Vorzeichen von  $y_2$  und erhalten

$$\begin{aligned}\tau &:= \frac{y_1}{y_2}, \\ r &= \operatorname{sgn}(y_2) \sqrt{y_1^2 + y_2^2} = \operatorname{sgn}(y_2) |y_2| \sqrt{\tau^2 + 1} = y_2 \sqrt{1 + \tau^2}, \\ c &= \frac{y_1}{r} = \frac{y_1}{y_2 \sqrt{1 + \tau^2}} = \frac{\tau}{\sqrt{1 + \tau^2}}, \\ s &= \frac{y_2}{r} = \frac{1}{\sqrt{1 + \tau^2}} = \sqrt{\frac{1 + \tau^2 - \tau^2}{1 + \tau^2}} = \sqrt{1 - \frac{\tau^2}{1 + \tau^2}} = \sqrt{1 - c^2}.\end{aligned}$$

Auch hier gilt nach Voraussetzung  $|\tau| \leq 1$ , und wir können direkt  $c = \tau s$  und damit  $2c^2 \leq s^2 + c^2 \leq 1$  ablesen, so dass wir zu  $|c| \leq \sqrt{1/2}$  gelangen.

Ein Sonderfall bleibt noch zu behandeln: Falls  $y_1 = y_2 = 0$  gilt, können wir  $\tau$  nicht berechnen. Allerdings brauchen wir das auch nicht zu tun, denn *jede* Drehung des Nullvektors wird dafür sorgen, dass seine zweite Komponente gleich null ist. In diesem Fall setzen wir schlicht  $c = 1$  und  $s = 0$ , so dass die Givens-Rotation gerade die Identität ist.

**Definition 2.19 (Givens-Rotation)** Sei  $\mathbf{y} \in \mathbb{R}^2$ , und seien

$$\begin{cases} \tau = y_2/y_1, c = 1/\sqrt{\tau^2 + 1}, s = \tau c & \text{falls } |y_1| \geq |y_2|, y_1 \neq 0, \\ \tau = y_1/y_2, s = 1/\sqrt{\tau^2 + 1}, c = \tau s & \text{falls } |y_1| < |y_2|, \\ c = 1, s = 0 & \text{ansonsten, also falls } y_1 = y_2 = 0. \end{cases}$$

Dann bezeichnen wir

$$\mathbf{Q} := \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

als die Givens-Rotation zu dem Vektor  $\mathbf{y}$ . Wie wir bereits gesehen haben, ist sie eine orthogonale Matrix und erfüllt

$$\mathbf{Q}\mathbf{y} = \begin{pmatrix} cy_1 + sy_2 \\ 0 \end{pmatrix},$$

kann also verwendet werden, um zweidimensionale Vektoren auf die erste Koordinatenachse zu drehen und so eine Komponente des Vektors verschwinden zu lassen.

Mit Hilfe von Givens-Rotationen können wir eine beliebige Matrix  $\mathbf{A} \in \mathbb{R}^{m \times n}$  auf obere Dreiecksgestalt bringen. In einem ersten Schritt zeigen wir, dass wir in einem beliebigen Vektor alle Einträge bis auf einen eliminieren können:

**Lemma 2.20 (Elimination)** Sei  $\mathbf{y} \in \mathbb{R}^n$ . Dann existiert eine orthogonale Matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  so, dass

$$\mathbf{Q}\mathbf{y} = \begin{pmatrix} \gamma \\ \mathbf{0} \end{pmatrix}$$

für ein  $\gamma \in \mathbb{R}$  mit  $|\gamma| = \|\mathbf{y}\|_2$  gilt.

*Beweis.* Per Induktion über  $n$ . Für  $n = 1$  setzen wir  $q_{11} = 1$  und  $\gamma = y_1$  und sind fertig.

Sei nun  $n \in \mathbb{N}$  so gewählt, dass die Behauptung für alle Vektoren  $\mathbf{z} \in \mathbb{R}^{n-1}$  gilt, und sei  $\mathbf{y} \in \mathbb{R}^n$ . Gemäß Definition 2.19 finden wir Zahlen  $c, s, \hat{\gamma} \in \mathbb{R}$ , die die Gleichungen

$$\begin{pmatrix} cy_1 + sy_n \\ -sy_1 + cy_n \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} y_1 \\ y_n \end{pmatrix} = \begin{pmatrix} \hat{\gamma} \\ 0 \end{pmatrix}$$

und  $c^2 + s^2 = 1$  erfüllen. Wir zerlegen den Vektor  $\mathbf{y}$  in der Form

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \mathbf{y}_* \\ y_n \end{pmatrix}, \quad \mathbf{y}_* = \begin{pmatrix} y_2 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

und erhalten

$$\begin{pmatrix} c & & s \\ & \mathbf{I} & \\ -s & & c \end{pmatrix} \begin{pmatrix} y_1 \\ \mathbf{y}_* \\ y_n \end{pmatrix} = \begin{pmatrix} cy_1 + sy_n \\ \mathbf{y}_* \\ -sy_1 + cy_n \end{pmatrix} = \begin{pmatrix} \hat{\gamma} \\ \mathbf{y}_* \\ 0 \end{pmatrix}.$$

Dank der Induktionsvoraussetzung finden wir eine orthogonale Matrix  $\hat{\mathbf{Q}} \in \mathbb{R}^{(n-1) \times (n-1)}$  und ein  $\gamma \in \mathbb{R}$  mit

$$\hat{\mathbf{Q}} \begin{pmatrix} \hat{\gamma} \\ \mathbf{y}_* \end{pmatrix} = \begin{pmatrix} \gamma \\ \mathbf{0} \end{pmatrix}.$$

Nun setzen wir

$$\mathbf{Q} := \begin{pmatrix} \hat{\mathbf{Q}} & \\ & 1 \end{pmatrix} \begin{pmatrix} c & & s \\ & \mathbf{I} & \\ -s & & c \end{pmatrix}$$

und erhalten insgesamt

$$\mathbf{Q}\mathbf{y} = \begin{pmatrix} \hat{\mathbf{Q}} & \\ & 1 \end{pmatrix} \begin{pmatrix} c & & s \\ & \mathbf{I} & \\ -s & & c \end{pmatrix} \begin{pmatrix} y_1 \\ \mathbf{y}_* \\ y_n \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{Q}} & \\ & 1 \end{pmatrix} \begin{pmatrix} \hat{\gamma} \\ \mathbf{y}_* \\ 0 \end{pmatrix} = \begin{pmatrix} \gamma \\ \mathbf{0} \end{pmatrix}.$$

Als Produkt orthogonaler Matrizen ist auch  $\mathbf{Q}$  orthogonal, also ist der Induktionsbeweis abgeschlossen, und aus  $\|\mathbf{y}\|_2 = \|\mathbf{Q}\mathbf{y}\|_2 = |\gamma|$  folgt der Rest der Behauptung. ■

Mit Hilfe dieser Aussage können wir die Existenz einer QR-Zerlegung sogar für nicht-quadratische Matrizen beweisen:

**Satz 2.21 (QR-Zerlegung)** Sei  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Dann existieren eine orthogonale Matrix  $\mathbf{Q} \in \mathbb{R}^{m \times m}$  und eine obere Dreiecksmatrix  $\mathbf{R} \in \mathbb{R}^{m \times n}$  mit

$$\mathbf{A} = \mathbf{Q}\mathbf{R}. \quad (2.20)$$

*Beweis.* Wir bringen zunächst die erste Spalte in die gewünschte Form: Nach Lemma 2.20 existiert eine orthogonale Matrix  $\hat{\mathbf{Q}} \in \mathbb{R}^{m \times m}$  so, dass

$$\hat{\mathbf{Q}} \begin{pmatrix} a_{11} \\ \vdots \\ a_{m1} \end{pmatrix} = \begin{pmatrix} \gamma \\ \mathbf{0} \end{pmatrix}$$

## 2 Lineare Gleichungssysteme

für ein  $\gamma \in \mathbb{R}$  gilt.

Von diesem Zwischenergebnis ausgehend können wir nun die Behauptung per Induktion über  $n \in \mathbb{N}$  beweisen. Für  $n = 1$  setzen wir  $\mathbf{Q} = \widehat{\mathbf{Q}}$  und sind bereits fertig.

Sei nun  $n \in \mathbb{N}$  so gewählt, dass die Aussage für alle Matrizen  $\mathbf{B} \in \mathbb{R}^{\ell \times (n-1)}$  mit  $\ell \in \mathbb{N}$  gilt, und sei  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Gemäß unserer Vorbereitung gilt

$$\widehat{\mathbf{Q}}\mathbf{A} = \begin{pmatrix} \gamma & \mathbf{A}_{1*} \\ & \mathbf{A}_{**} \end{pmatrix}$$

für geeignete Teilmatrizen  $\mathbf{A}_{1*} \in \mathbb{R}^{1 \times (n-1)}$  und  $\mathbf{A}_{**} \in \mathbb{R}^{(m-1) \times (n-1)}$ . Falls  $m - 1 = 0$  oder  $n - 1 = 0$  gelten, sind wir damit bereits fertig. Anderenfalls benutzen wir die Induktionsvoraussetzung, um eine orthogonale Matrix  $\mathbf{P} \in \mathbb{R}^{(m-1) \times (m-1)}$  zu finden, die

$$\mathbf{A}_{**} = \mathbf{P}\mathbf{R}_{**}, \quad \mathbf{P}^*\mathbf{A}_{**} = \mathbf{R}_{**}$$

mit einer oberen Dreiecksmatrix  $\mathbf{R}_{**} \in \mathbb{R}^{(m-1) \times (n-1)}$  erfüllt. Wir definieren

$$\mathbf{Q} := \widehat{\mathbf{Q}}^* \begin{pmatrix} 1 & \\ & \mathbf{P} \end{pmatrix}$$

und erhalten

$$\mathbf{Q}^*\mathbf{A} = \begin{pmatrix} 1 & \\ & \mathbf{P}^* \end{pmatrix} \widehat{\mathbf{Q}}\mathbf{A} = \begin{pmatrix} 1 & \\ & \mathbf{P}^* \end{pmatrix} \begin{pmatrix} \gamma & \mathbf{A}_{1*} \\ & \mathbf{A}_{**} \end{pmatrix} = \begin{pmatrix} \gamma & \mathbf{A}_{1*} \\ & \mathbf{P}^*\mathbf{A}_{**} \end{pmatrix} = \mathbf{R} = \begin{pmatrix} \gamma & \mathbf{A}_{1*} \\ & \mathbf{R}_{**} \end{pmatrix} = \mathbf{R},$$

also folgt  $\mathbf{A} = \mathbf{Q}\mathbf{R}$ . Als Produkt orthogonaler Matrizen ist  $\mathbf{Q}$  ebenfalls orthogonal, und  $\mathbf{R}$  ist eine obere Dreiecksmatrix. ■

Zur Illustration untersuchen wir eine Matrix  $\mathbf{A} \in \mathbb{R}^{4 \times 3}$ . Wir stellen sie in der Form

$$\mathbf{A} = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix}$$

dar, wobei „ $\times$ “ für einen möglicherweise von null verschiedenen Eintrag steht. In einem ersten Schritt wenden wir eine Givens-Rotation  $\mathbf{Q}_{41}$  auf die erste und vierte Zeile an, um den Eintrag  $a_{41}$  zu eliminieren und erhalten so

$$\mathbf{A}^{(1)} := \mathbf{Q}_{41}\mathbf{A} = \begin{pmatrix} \otimes & \otimes & \otimes \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \otimes & \otimes \end{pmatrix}.$$

Hier markiert „ $\otimes$ “ von null verschiedene Einträge, die im letzten Arbeitsschritt verändert worden sind, und „0“ die in diesem Schritt entstandene Null.

Im nächsten Schritt benutzen wir eine Givens-Rotation  $\mathbf{Q}_{31}$  für die erste und dritte Zeile, um den Eintrag  $a_{31}^{(1)}$  zu beseitigen und gelangen zu

$$\mathbf{A}^{(2)} := \mathbf{Q}_{31}\mathbf{A}^{(1)} = \begin{pmatrix} \otimes & \otimes & \otimes \\ \times & \times & \times \\ 0 & \otimes & \otimes \\ & \times & \times \end{pmatrix}.$$

Entsprechend behandeln wir  $a_{21}^{(2)}$  mit einer Givens-Rotation  $\mathbf{Q}_{21}$ , die auf die erste und zweite Zeile wirkt:

$$\mathbf{A}^{(3)} := \mathbf{Q}_{21}\mathbf{A}^{(2)} = \begin{pmatrix} \otimes & \otimes & \otimes \\ 0 & \otimes & \otimes \\ & \times & \times \\ & \times & \times \end{pmatrix}.$$

In der zweiten Spalte dürfen wir nun Givens-Rotationen der Zeilen zwei bis vier verwenden, um  $a_{42}^{(3)}$  und  $a_{32}^{(4)}$  zu eliminieren, denn dabei gehen die Nulleinträge in der ersten Spalte nicht verloren:

$$\mathbf{A}^{(4)} := \mathbf{Q}_{42}\mathbf{A}^{(3)} = \begin{pmatrix} \times & \times & \times \\ & \otimes & \otimes \\ & \times & \times \\ & 0 & \otimes \end{pmatrix},$$

$$\mathbf{A}^{(5)} := \mathbf{Q}_{32}\mathbf{A}^{(4)} = \begin{pmatrix} \times & \times & \times \\ & \otimes & \otimes \\ & 0 & \otimes \\ & & \times \end{pmatrix}.$$

Mit einer letzten Givens-Rotation  $\mathbf{Q}_{43}$  für den Eintrag  $a_{43}^{(5)}$  erhalten wir die gewünschte Dreiecksform:

$$\mathbf{R} := \mathbf{Q}_{43}\mathbf{A}^{(5)} = \begin{pmatrix} \times & \times & \times \\ & \times & \times \\ & & \otimes \\ & & 0 \end{pmatrix}.$$

Damit haben wir das Ziel erreicht, es gilt

$$\mathbf{R} = \mathbf{Q}_{43}\mathbf{Q}_{32}\mathbf{Q}_{42}\mathbf{Q}_{21}\mathbf{Q}_{31}\mathbf{Q}_{41}\mathbf{A},$$

$$\underbrace{\mathbf{Q}_{41}^* \mathbf{Q}_{31}^* \mathbf{Q}_{21}^* \mathbf{Q}_{42}^* \mathbf{Q}_{32}^* \mathbf{Q}_{43}^*}_{=: \mathbf{Q}} \mathbf{R} = \mathbf{A},$$

und  $\mathbf{Q}$  ist als Produkt orthogonaler Matrizen selber orthogonal.

**Bemerkung 2.22 (Kompakte Darstellung)** Für die praktische Berechnung der QR-Zerlegung wäre es nützlich, wenn wir die gesamte Zerlegung, wie schon bei der LR-Zerlegung, in dem Speicher der ursprünglichen Matrix unterbringen könnten. Da eine

## 2 Lineare Gleichungssysteme

*Givens-Rotation einen Matrixeintrag eliminiert, müssten wir dazu diese Rotation durch eine einzige Zahl beschreiben.*

*Dazu nutzen wir aus, dass im Fall  $|y_1| \geq |y_2|$ ,  $y_1 \neq 0$  nach Definition 2.19  $c \in \mathbb{R}_{>0}$  gilt, so dass wir dank  $s^2 + c^2 = 1$  den Cosinus mit der Formel  $c = \sqrt{1 - s^2}$  aus dem Sinus  $s$  rekonstruieren können. Demzufolge genügt es,  $s$  abzuspeichern. Wir haben bereits gesehen, dass in diesem Fall  $|s| \leq \sqrt{1/2}$  gilt.*

*Im Fall  $|y_1| < |y_2|$  würden wir gerne wieder die Rollen von  $s$  und  $c$  vertauschen, also  $c$  abspeichern und  $s$  rekonstruieren, allerdings müssten wir uns dann noch zusätzlich merken, welcher der beiden Fälle aufgetreten ist. Dieses Problem lösen wir, indem wir in diesem Fall nicht  $c$ , sondern  $1/c$  speichern. Da  $|c| \leq \sqrt{1/2} < 1$  gilt, folgt  $|1/c| \geq \sqrt{2} > 1$ , so dass keine Verwechslungsgefahr besteht.*

*Wir beschreiben die Givens-Rotation also durch die Zahl*

$$\varrho := \begin{cases} s & \text{falls } |y_1| \geq |y_2|, y_1 \neq 0, \\ 1/c & \text{falls } |y_1| < |y_2|, \\ 1 & \text{ansonsten, also falls } y_1 = y_2 = 0. \end{cases}$$

*und rekonstruieren  $s$  und  $c$  mittels*

$$\begin{cases} s = \varrho, c = \sqrt{1 - s^2} & \text{falls } |\varrho| < 1, \\ c = 1/\varrho, s = \sqrt{1 - c^2} & \text{falls } |\varrho| > 1, \\ c = 1, s = 0 & \text{ansonsten, also falls } \varrho = 1. \end{cases}$$

*Die Zahl  $\varrho_{ij}$ , die die Givens-Rotation beschreibt, mit der der Eintrag  $a_{ij}$  eliminiert wird, können wir anschließend in dessen Speicherplatz aufbewahren und erhalten analog zu (2.7) die Darstellung*

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ \varrho_{21} & r_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ \varrho_{n1} & \cdots & \varrho_{n,n-1} & r_{nn} \end{pmatrix},$$

*der QR-Zerlegung.*

Nun können wir einen Algorithmus angeben, der die QR-Zerlegung einer beliebigen Matrix berechnet:

```

procedure qr_decomp( $m, n$ , var A);
for  $k = 1$  to  $\min\{m, n\}$  do
  for  $i \in \{k + 1, \dots, m\}$  do begin
    if  $a_{ik} = 0$  then begin
       $\varrho \leftarrow 1$ ;  $c \leftarrow 1$ ;  $s \leftarrow 0$ 
    end else if  $|a_{kk}| \geq |a_{ik}|$  then begin
       $\tau \leftarrow a_{ik}/a_{kk}$ ;  $\varrho \leftarrow \tau/\sqrt{\tau^2 + 1}$ ;  $s \leftarrow \varrho$ ;  $c \leftarrow \sqrt{1 - s^2}$ 
    end else begin

```



```

     $\tau \leftarrow a_{kk}/a_{ik}; \quad \varrho \leftarrow \sqrt{\tau^2 + 1}/\tau; \quad c \leftarrow 1/\varrho; \quad s \leftarrow \sqrt{1 - c^2}$ 
end;
 $a_{kk} \leftarrow ca_{kk} + sa_{ik}; \quad a_{ik} \leftarrow \varrho;$ 
for  $j \in \{k + 1, \dots, n\}$  do begin
     $\alpha \leftarrow a_{kj}; \quad a_{kj} \leftarrow c\alpha + sa_{ij}; \quad a_{ij} \leftarrow -s\alpha + ca_{ij}$ 
end
end
end

```

Um die Anzahl der benötigten Rechenoperationen abzuschätzen führen wir

$$\nu := \min\{m, n\}, \quad \mu := \max\{m, n\}$$

und erhalten die Schranke

$$\begin{aligned}
 & \sum_{k=1}^{\nu} (12(m-k) + 6(m-k)(n-k)) \\
 &= \sum_{k=1}^{\nu} (12(\nu-k) + 12(m-\nu) + 6(\nu-k)^2 + 6(\mu-\nu)(\nu-k)) \\
 &= 6\nu(\nu-1) + 12(m-\nu)\nu + \nu(\nu-1)(2\nu-1) + 3(\mu-\nu)\nu(\nu-1) \\
 &= 6\nu(\nu-1 + 2m - 2\nu) + \nu(\nu-1)(2\nu-1 + 3\mu - 3\nu) \\
 &< 6\nu(2m - \nu) + \nu^2(3\mu - \nu)
 \end{aligned} \tag{2.21}$$

für die Anzahl der arithmetischen Operationen. Im Fall einer quadratischen Matrix gilt  $\nu = \mu = n$  und es ergibt sich die Schranke  $6n^2 + 2n^3$ , die QR-Zerlegung benötigt also für große Matrizen ungefähr dreimal so viele Rechenoperationen wie die LR-Zerlegung.

Um das lineare Gleichungssystem  $Ax = b$  mit Hilfe der QR-Zerlegung zu lösen, verwenden wir

$$\mathbf{b} = \mathbf{Ax} = \mathbf{QRx} = \mathbf{Qy}, \quad \mathbf{y} = \mathbf{Rx}.$$

Die zweite Gleichung können wir wieder durch Rückwärtseinsetzen lösen, für die erste verwenden wir die Tatsache, dass die Transponierte einer orthogonalen Matrix ihre Inverse ist, dass also

$$\mathbf{y} = \mathbf{Q}^*\mathbf{b}$$

gilt. Wir benötigen demnach einen Algorithmus, mit dem wir die Transponierte der Matrix  $\mathbf{Q}$  unserer QR-Zerlegung auf einen Vektor anwenden können. Dazu müssen wir lediglich der Reihe nach die Givens-Rotationen anwenden, die schon bei der Transformation auf Dreiecksgestalt zum Einsatz kamen:

```

procedure qr_transform( $m, n, \mathbf{A}, \mathbf{var} b$ );
for  $k = 1$  to  $\min\{m, n\}$  do
    for  $i \in \{k + 1, \dots, m\}$  do begin
         $\varrho \leftarrow a_{ik};$ 

```

```

if  $\varrho = 1$  then begin
   $c \leftarrow 1; \quad s \leftarrow 0$ 
end else if  $|\varrho| < 1$  then begin
   $s \leftarrow \varrho; \quad c \leftarrow \sqrt{1 - s^2}$ 
end else begin
   $c \leftarrow 1/\varrho; \quad s \leftarrow \sqrt{1 - c^2}$ 
end;
 $\alpha \leftarrow b_k; \quad b_k \leftarrow c\alpha + sb_i; \quad b_i \leftarrow -s\alpha + cb_i$ 
end
end

```

Der Algorithmus überschreibt  $\mathbf{b}$  mit dem Vektor  $\mathbf{Q}^*\mathbf{b}$ .

QR-Zerlegungen lassen sich auch mit Hilfe effizienterer Verfahren konstruieren, beispielsweise mit *Householder-Spiegelungen*, die in einem Arbeitsschritt ganze Spalten der Matrix eliminieren. Dadurch lässt sich der Rechenaufwand auf ungefähr das Doppelte des Aufwands der LR-Zerlegung reduzieren. Der Preis für die höhere Stabilität der QR-Zerlegung im Vergleich zur LR-Zerlegung besteht also in dem höheren Rechenaufwand. In der Praxis hängt der Zeitaufwand der Algorithmen auf modernen Computern eher von der Anzahl und Reihenfolge von Speicherzugriffen ab, so dass eine gut implementierte QR-Zerlegung unter geeigneten Bedingungen ähnlich schnell wie eine LR-Zerlegung ablaufen kann.

## 2.7 Ausgleichsprobleme

In vielen praktischen Anwendungen erhalten wir lineare Gleichungen, die sich nicht exakt auflösen lassen. Als Beispiel untersuchen wir ein *überbestimmtes* System, bei dem wesentlich mehr Gleichungen als Unbekannte auftreten: Wir gehen davon aus, dass uns Paare von Werten  $(t_1, b_1), \dots, (t_m, b_m)$  zur Verfügung stehen und dass wir davon ausgehen dürfen, dass die Werte über eine Funktion zueinander in Beziehung gesetzt werden, dass also

$$b_i = y(t_i) \quad \text{für alle } i \in \{1, \dots, m\}$$

gilt, wobei  $y$  eine *unbekannte* Funktion ist, die wir bestimmen möchten.

Wir nehmen dazu an, dass sich  $y$  als Linearkombination bekannter Funktionen  $y_1, \dots, y_n$  mit  $n < m$  ergibt und bezeichnen die entsprechenden Faktoren mit  $x_1, \dots, x_n$ , so dass sich

$$y(t) = x_1 y_1(t) + x_2 y_2(t) + \dots + x_n y_n(t)$$

ergibt. Durch Einsetzen folgt

$$b_i = y(t_i) = x_1 y_1(t_i) + x_2 y_2(t_i) + \dots + x_n y_n(t_i) \quad \text{für alle } i \in \{1, \dots, m\},$$

wir haben also  $m$  Gleichungen für die  $n$  Unbekannten  $x_1, \dots, x_n$ , und die Anzahl der Gleichungen übersteigt die Anzahl der Unbekannten.

Wir können dieses Gleichungssystem kompakt ausdrücken, indem wir

$$\mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} y_1(t_1) & \dots & y_n(t_1) \\ \vdots & \ddots & \vdots \\ y_1(t_m) & \dots & y_n(t_m) \end{pmatrix}$$

eingeführen und zu der Gleichung

$$\mathbf{b} = \mathbf{A}\mathbf{x}$$

gelangen. Da in der Regel die gemessenen Werte  $b_1, \dots, b_m$  nicht perfekt zu unserem Ansatz für  $y$  passen werden, empfiehlt es sich, nicht nach einer exakten Lösung zu suchen, sondern lediglich nach der besten, die möglich ist: Wir suchen nach einem  $\mathbf{x} \in \mathbb{R}^n$  derart, dass

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n \quad (2.22)$$

gilt, der Lösungsvektor soll also derjenige Vektor sein, für den  $\mathbf{A}\mathbf{x}$  eine optimale Näherung von  $\mathbf{b}$  ergibt. Derartige Aufgabenstellungen bezeichnet man als *Ausgleichsprobleme*: Wir suchen  $\mathbf{x}$  so, dass alle Gleichungen möglichst „gleich gut“ erfüllt werden.

Anders als lineare Gleichungssysteme besitzen Ausgleichsprobleme immer eine Lösung, die allerdings nicht eindeutig bestimmt zu sein braucht. Um diese Eindeutigkeit sicher zu stellen, fordern wir, dass  $\mathbf{A}$  injektiv ist, dass also der Kern lediglich den Nullvektor enthält. In diesem Fall können wir die Lösung sogar direkt berechnen, indem wir uns wieder eine QR-Zerlegung zunutze machen: Es gelte  $\mathbf{A} = \mathbf{Q}\mathbf{R}$  mit einer orthogonalen Matrix  $\mathbf{Q} \in \mathbb{R}^{m \times m}$  und einer oberen Dreiecksmatrix  $\mathbf{R} \in \mathbb{R}^{m \times n}$ . Da  $\mathbf{Q}$  orthogonal ist, gilt  $\mathbf{Q}^* = \mathbf{Q}^{-1}$ , und wir erhalten

$$\|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2 = \|\mathbf{Q}\mathbf{R}\mathbf{z} - \mathbf{Q}\mathbf{Q}^*\mathbf{b}\|_2 = \|\mathbf{Q}(\mathbf{R}\mathbf{z} - \mathbf{Q}^*\mathbf{b})\|_2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n.$$

Da sich die Norm unter orthogonalen Transformationen nicht ändert, folgt daraus

$$\|\mathbf{A}\mathbf{z} - \mathbf{b}\|_2 = \|\mathbf{R}\mathbf{z} - \mathbf{Q}^*\mathbf{b}\|_2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n.$$

Da  $\mathbf{R}$  eine obere Dreiecksmatrix ist, sind alle Zeilen unterhalb der ersten  $n$  gleich null, wir können die Matrix also in der Form

$$\mathbf{R} = \begin{pmatrix} \widehat{\mathbf{R}} \\ \mathbf{0} \end{pmatrix}, \quad \widehat{\mathbf{R}} \in \mathbb{R}^{n \times n}$$

darstellen. Da  $\mathbf{R}$  injektiv ist, muss auch  $\widehat{\mathbf{R}}$  injektiv sein, und als quadratische Matrix damit auch regulär. Wir zerlegen den Vektor  $\mathbf{Q}^*\mathbf{b}$  entsprechend und erhalten

$$\mathbf{Q}^*\mathbf{b} = \begin{pmatrix} \widehat{\mathbf{b}} \\ \mathbf{b}_0 \end{pmatrix}, \quad \widehat{\mathbf{b}} \in \mathbb{R}^n, \quad \mathbf{b}_0 \in \mathbb{R}^{m-n}.$$

## 2 Lineare Gleichungssysteme

Ein Blick auf die Definition der euklidischen Norm (2.8) zeigt, dass

$$\|\mathbf{Rz} - \mathbf{Q}^*\mathbf{b}\|_2^2 = \left\| \begin{pmatrix} \widehat{\mathbf{R}}\mathbf{z} - \widehat{\mathbf{b}} \\ -\mathbf{b}_0 \end{pmatrix} \right\|_2^2 = \|\widehat{\mathbf{R}}\mathbf{z} - \widehat{\mathbf{b}}\|_2^2 + \|\mathbf{b}_0\|_2^2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n$$

gilt. Da  $\widehat{\mathbf{R}}$  regulär ist, können wir den ersten der beiden Summanden minimieren, indem wir das Gleichungssystem

$$\widehat{\mathbf{R}}\mathbf{x} = \widehat{\mathbf{b}} \quad (2.23)$$

lösen. Der zweite der Summanden ist von  $\mathbf{z}$  völlig unabhängig, wird sich also auch nicht reduzieren lassen. Mit dem durch (2.23) definierten  $\mathbf{x}$  folgt

$$\begin{aligned} \|\mathbf{Ax} - \mathbf{b}\|_2^2 &= \|\mathbf{Rx} - \mathbf{Q}^*\mathbf{b}\|_2^2 = \|\widehat{\mathbf{R}}\mathbf{x} - \widehat{\mathbf{b}}\|_2^2 + \|\mathbf{b}_0\|_2^2 = \|\mathbf{b}_0\|_2^2 \\ &\leq \|\widehat{\mathbf{R}}\mathbf{z} - \widehat{\mathbf{b}}\|_2^2 + \|\mathbf{b}_0\|_2^2 = \|\mathbf{Rz} - \mathbf{Q}^*\mathbf{b}\|_2^2 = \|\mathbf{Az} - \mathbf{b}\|_2^2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n, \end{aligned}$$

also haben wir die Lösung des Ausgleichsproblems gefunden. Da das System (2.23) eindeutig lösbar ist, ist  $\mathbf{x}$  auch die eindeutige Lösung des Ausgleichsproblems. Wir haben also eine einfache Lösungsstrategie gefunden: In einem ersten Schritt wird die QR-Zerlegung berechnet, in einem zweiten das System (2.23) durch Rückwärtseinsetzen gelöst.

**Bemerkung 2.23 (Normalengleichung)** *Es ist möglich, das Ausgleichsproblem zu lösen, ohne auf eine QR-Zerlegung zurückzugreifen: Wir können (2.23) mit der regulären Matrix  $\widehat{\mathbf{R}}^*$  multiplizieren, ohne das Ergebnis zu ändern, und erhalten*

$$\mathbf{A}^*\mathbf{Ax} = \mathbf{R}^*\mathbf{Q}^*\mathbf{QRx} = \mathbf{R}^*\mathbf{Rx} = \widehat{\mathbf{R}}^*\widehat{\mathbf{R}}\mathbf{x} = \widehat{\mathbf{R}}^*\widehat{\mathbf{b}} = \begin{pmatrix} \widehat{\mathbf{R}} \\ \mathbf{0} \end{pmatrix}^* \begin{pmatrix} \widehat{\mathbf{b}} \\ \mathbf{b}_0 \end{pmatrix} = \begin{pmatrix} \widehat{\mathbf{R}} \\ \mathbf{0} \end{pmatrix}^* \mathbf{Q}^*\mathbf{b} = \mathbf{A}^*\mathbf{b}.$$

Die so erhaltene Gleichung

$$\mathbf{A}^*\mathbf{Ax} = \mathbf{A}^*\mathbf{b} \quad (2.24)$$

bezeichnet man als Normalengleichung, da sie anschaulich bedeutet, dass

$$0 = \langle \mathbf{A}^*(\mathbf{Ax} - \mathbf{b}), \mathbf{z} \rangle_2 = \langle \mathbf{Ax} - \mathbf{b}, \mathbf{Az} \rangle_2 \quad \text{für alle } \mathbf{z} \in \mathbb{R}^n$$

gilt, dass also der Fehler  $\mathbf{Ax} - \mathbf{b}$  senkrecht auf dem Bild der Matrix  $\mathbf{A}$  steht und damit ein Normalenvektor auf diesem Teilraum ist.

Wenn man ausnutzt, dass die Matrix  $\mathbf{A}^*\mathbf{A}$  symmetrisch ist und sich deshalb effizient aufstellen lässt, kann das Lösen der Normalengleichung weniger Zeit erfordern als der Zugang über die QR-Zerlegung. Man kann allerdings nachrechnen, dass

$$\kappa_2(\mathbf{A}^*\mathbf{A}) = \kappa_2(\mathbf{A})^2$$

gilt, so dass die Verwendung der Normalengleichung das Risiko einer ungünstigen Fehlerverstärkung birgt. Für die etwas höhere Geschwindigkeit ist also möglicherweise eine geringere Genauigkeit in Kauf zu nehmen.

**Bemerkung 2.24 (Lösbarkeit)** Das Ausgleichsproblem (2.22) besitzt immer mindestens eine Lösung: Wir bezeichnen das Bild der transponierten Matrix  $\mathbf{A}^*$  mit

$$V := \{\mathbf{A}^* \mathbf{y} : \mathbf{y} \in \mathbb{R}^m\}$$

und führen die lineare Abbildung

$$\mathcal{L}: V \rightarrow V, \quad \mathbf{x} \mapsto \mathbf{A}^* \mathbf{A} \mathbf{x},$$

ein. Sie bildet den endlich-dimensionalen Raum  $V$  in sich ab, ist also nach dem Dimensionssatz genau dann invertierbar, wenn sie injektiv ist.

Sei  $\mathbf{x} \in V$  mit  $\mathcal{L}\mathbf{x} = \mathbf{0}$  gegeben. Daraus folgt mit (2.18) insbesondere

$$\|\mathbf{A}\mathbf{x}\|_2^2 = \langle \mathbf{A}\mathbf{x}, \mathbf{A}\mathbf{x} \rangle_2 = \langle \mathbf{x}, \mathbf{A}^* \mathbf{A} \mathbf{x} \rangle_2 = \langle \mathbf{x}, \mathcal{L}\mathbf{x} \rangle_2 = \langle \mathbf{x}, \mathbf{0} \rangle_2 = 0,$$

also  $\mathbf{A}\mathbf{x} = \mathbf{0}$ .

Da  $\mathbf{x} \in V$  gilt, finden wir ein  $\mathbf{y} \in \mathbb{R}^m$  mit  $\mathbf{x} = \mathbf{A}^* \mathbf{y}$ . Mit (2.18) gilt

$$\|\mathbf{x}\|_2^2 = \langle \mathbf{x}, \mathbf{x} \rangle_2 = \langle \mathbf{x}, \mathbf{A}^* \mathbf{y} \rangle_2 = \langle \mathbf{A}\mathbf{x}, \mathbf{y} \rangle_2 = \langle \mathbf{0}, \mathbf{y} \rangle_2 = 0,$$

so dass  $\mathcal{L}$  injektiv sein muss, und damit auch invertierbar.

Da die rechte Seite der Normalengleichung (2.24) offenbar ein Element des Raums  $V$  ist, folgt die Existenz eines  $\mathbf{x} \in V \subseteq \mathbb{R}^n$  mit  $\mathbf{A}^* \mathbf{A} \mathbf{x} = \mathcal{L}\mathbf{x} = \mathbf{A}^* \mathbf{b}$ .

Dieser Vektor  $\mathbf{x}$  besitzt noch eine weitere erstrebenswerte Eigenschaft: Nach Definition finden wir ein  $\mathbf{y} \in \mathbb{R}^m$  mit  $\mathbf{x} = \mathbf{A}^* \mathbf{y}$ . Falls  $\mathbf{z} \in \mathbb{R}^n$  ein Element des Kerns der Matrix  $\mathbf{A}$  ist, falls also  $\mathbf{A}\mathbf{z} = \mathbf{0}$  gilt, haben wir

$$\langle \mathbf{x}, \mathbf{z} \rangle_2 = \langle \mathbf{A}^* \mathbf{y}, \mathbf{z} \rangle_2 = \langle \mathbf{y}, \mathbf{A}\mathbf{z} \rangle_2 = \langle \mathbf{y}, \mathbf{0} \rangle_2 = 0,$$

also steht  $\mathbf{x}$  auf dem Kern der Matrix  $\mathbf{A}$  senkrecht. Daraus folgt, dass  $\mathbf{x}$  unter allen Lösungen der Normalengleichung diejenige mit der minimalen euklidischen Norm ist.



# 3 Nichtlineare Gleichungssysteme

## 3.1 Beispiel: Lagrange-Punkte im Gravitationsfeld

Als Beispiel für ein nichtlineares Gleichungssystem untersuchen wir die Suche nach den *Lagrange-Punkten* in einem Gravitationsfeld. Dabei handelt es sich um diejenigen Punkte, in denen sich die wirkenden Kräfte ausbalancieren.

Die klassische Aufgabenstellung lautet dabei wie folgt: Eine Reihe von Massen rotieren um den gemeinsamen Schwerpunkt. Gesucht sind diejenigen Punkte im Raum, an denen sich die von diesen Massen ausgeübten Gravitationsfelder gegenseitig aufheben, so dass ein hier platzierter Körper ruht.

Mathematisch beschreiben wir die Positionen des Lagrange-Punkts durch eine Funktion

$$y: \mathbb{R} \rightarrow \mathbb{R}^2,$$

die jedem Zeitpunkt  $t \in \mathbb{R}$  seine Position  $y(t)$  zuordnet.

Da das Gesamtsystem rotiert, empfiehlt es sich, ein rotierendes Koordinatensystem einzuführen, in dem die Massen ruhen. Dazu definieren wir die Matrix

$$Q_\alpha: \mathbb{R} \rightarrow \mathbb{R}^{2 \times 2}, \quad t \mapsto \begin{pmatrix} \cos(\alpha t) & -\sin(\alpha t) \\ \sin(\alpha t) & \cos(\alpha t) \end{pmatrix},$$

die eine Rotation mit Winkelgeschwindigkeit  $\alpha \in \mathbb{R}$  gegen den Uhrzeigersinn um den Nullpunkt beschreibt. In dem durch ihre Spalten gegebenen Koordinatensystem können wir die Positionen des Lagrange-Punkts in der Form

$$y(t) = Q_\alpha(t)\hat{y}(t) \quad \text{für alle } t \in \mathbb{R}$$

schreiben. Da  $Q_\alpha(t)$  eine orthogonale Matrix ist, gilt  $Q_\alpha(t)^{-1} = Q_\alpha(t)^*$ , also

$$\hat{y}(t) = Q_\alpha(t)^*y(t) \quad \text{für alle } t \in \mathbb{R}.$$

Damit der Lagrange-Punkt bezüglich des rotierenden Systems ruht, muss seine Ableitung verschwinden, es muss also nach der Produktregel die Gleichung

$$0 = \hat{y}'(t) = Q_\alpha(t)^*y'(t) + Q_\alpha'(t)^*y(t) \quad \text{für alle } t \in \mathbb{R}$$

gelten. Wir haben

$$Q_\alpha'(t) = \begin{pmatrix} -\alpha \sin(\alpha t) & -\alpha \cos(\alpha t) \\ \alpha \cos(\alpha t) & -\alpha \sin(\alpha t) \end{pmatrix},$$

### 3 Nichtlineare Gleichungssysteme

$$\begin{aligned} Q_\alpha(t)Q'_\alpha(t)^* &= \begin{pmatrix} \cos(\alpha t) & -\sin(\alpha t) \\ \sin(\alpha t) & \cos(\alpha t) \end{pmatrix} \begin{pmatrix} -\alpha \sin(\alpha t) & \alpha \cos(\alpha t) \\ -\alpha \cos(\alpha t) & -\alpha \sin(\alpha t) \end{pmatrix}, \\ &= \begin{pmatrix} 0 & \alpha \\ -\alpha & 0 \end{pmatrix} = -\alpha P \quad \text{für alle } t \in \mathbb{R}, \end{aligned}$$

wobei

$$P := \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

die 90-Grad-Drehung gegen den Uhrzeigersinn um den Nullpunkt beschreibt. Damit der Lagrange-Punkt bezüglich des rotierenden Koordinatensystems ruht, muss also

$$y'(t) = \alpha P y(t) \quad \text{für alle } t \in \mathbb{R}$$

gelten. Indem wir diese Gleichung differenzieren, erhalten wir mit

$$y''(t) = \alpha P y'(t) = \alpha^2 P^2 y(t) = -\alpha^2 y(t) \quad \text{für alle } t \in \mathbb{R}$$

eine Aussagen über die Beschleunigung, denen im Lagrange-Punkt befindliche Massen unterliegen müssen, um im rotierenden System auf einer festen Position zu verharren.

In unserem Fall entstehen diese Beschleunigungen durch Newtons Gravitationsgesetz. Wir gehen davon aus, dass Massen  $m_1, \dots, m_n$  an Positionen  $x_1(t), \dots, x_n(t)$  gegeben sind, und dass auch diese Positionen im rotierenden Koordinatensystem konstant sind, dass also

$$Q_\alpha(t)^* x_i(t) = Q_\alpha(0)^* x_i(0), \quad x_i(t) = Q_\alpha(t) x_i(0) \quad \text{für alle } i \in [1 : n], t \in \mathbb{R}$$

gilt. Nach dem Gravitationsgesetz üben sie auf eine an dem Punkt  $y(t)$  befindliche Masse die Beschleunigung

$$a(t) = \gamma \sum_{i=1}^n m_i \frac{x_i(t) - y(t)}{\|x_i(t) - y(t)\|_2^3} \quad \text{für alle } t \in \mathbb{R}$$

aus, so dass wir insgesamt die Gleichung

$$-\alpha^2 y(t) = y''(t) = a(t) = \gamma \sum_{i=1}^n m_i \frac{x_i(t) - y(t)}{\|x_i(t) - y(t)\|_2^3} \quad \text{für alle } t \in \mathbb{R}$$

erhalten. Da alle Massen in derselben Weise rotieren, genügt es, sich auf einen festen Zeitpunkt  $t = 0$  zu beschränken, und wir gelangen zu der Gleichung

$$0 = \alpha^2 y(0) + \gamma \sum_{i=1}^n m_i \frac{x_i(0) - y(0)}{\|x_i(0) - y(0)\|_2^3},$$

die die Position  $y(0)$  eines Lagrange-Punkts charakterisiert.

Lagrange-Punkte sind demnach Nullstellen der Funktion

$$f: \mathbb{R}^2 \rightarrow \mathbb{R}^2, \quad z \mapsto \alpha^2 z + \gamma \sum_{i=1}^n m_i \frac{x_i(0) - z}{\|x_i(0) - z\|_2^3},$$

und anders als im vorangehenden Kapitel ist diese Funktion nicht linear, so dass die bisher diskutierten Lösungsverfahren nicht zur Anwendung kommen können.



**Bemerkung 3.1 (Schwerpunkt)** In der Regel sollten auch die Massen  $m_1, \dots, m_n$  sich gegenseitig im Gleichgewicht halten, es sollte also

$$0 = \alpha^2 x_j(0) + \gamma \sum_{\substack{i=1 \\ i \neq j}}^n m_i \frac{x_i(0) - x_j(0)}{\|x_i(0) - x_j(0)\|_2^3} \quad \text{für alle } j \in [1 : n] \quad (3.1)$$

gelten. Indem wir die  $j$ -te Gleichung mit  $m_j$  multiplizieren und alle Gleichungen aufaddieren, erhalten wir

$$0 = \alpha^2 \sum_{j=1}^n m_j x_j(0) + \gamma \sum_{\substack{i,j=1 \\ i \neq j}}^n m_i m_j \frac{x_i(0) - x_j(0)}{\|x_i(0) - x_j(0)\|_2^3},$$

und da der zweite Summand aus Symmetriegründen verschwindet, folgt

$$0 = \sum_{j=1}^n m_j x_j(0),$$

der Nullpunkt muss also der Schwerpunkt unseres Systems sein.

**Bemerkung 3.2 (Zweikörperproblem)** Im Fall eines Zweikörperproblems können wir mit Hilfe der Gleichung (3.1) auch die Winkelgeschwindigkeit berechnen, die für ein stabiles Gleichgewicht erforderlich ist. Indem wir die beiden Gleichungen voneinander subtrahieren, erhalten wir

$$\begin{aligned} 0 &= \alpha^2 (x_1(0) - x_2(0)) + \gamma \left( m_2 \frac{x_2(0) - x_1(0)}{\|x_2(0) - x_1(0)\|_2^3} - m_1 \frac{x_1(0) - x_2(0)}{\|x_1(0) - x_2(0)\|_2^3} \right) \\ &= \alpha^2 (x_1(0) - x_2(0)) - \gamma \frac{m_1 + m_2}{\|x_2(0) - x_1(0)\|_2^3} (x_1(0) - x_2(0)), \end{aligned}$$

und daraus folgt

$$\alpha^2 = \gamma \frac{m_1 + m_2}{\|x_2(0) - x_1(0)\|_2^3}.$$

## 3.2 Bisektionsverfahren

Im Allgemeinen untersuchen wir nichtlineare Gleichungssysteme der Form

Gegeben eine stetige Funktion  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , finde  $\mathbf{x}^* \in \mathbb{R}^n$  mit

$$f(\mathbf{x}^*) = \mathbf{0}.$$

Da sich jedes nichtlineare Gleichungssystem einfach in ein derartiges Nullstellenproblem überführen lässt, ist dieser Problemtyp ausreichend allgemein.

### 3 Nichtlineare Gleichungssysteme

Wir untersuchen zunächst den eindimensionalen Fall: Seien  $a, b \in \mathbb{R}$  mit  $a < b$  gegeben, und sei  $f \in C[a, b]$  eine Funktion. Unser Ziel ist es, eine Nullstelle  $x^* \in [a, b]$  dieser Funktion zu finden, also eine Lösung des Nullstellenproblems

$$f(x^*) = 0.$$

Derartige Probleme lassen sich häufig nicht in endlich vielen Rechenschritten lösen, also verwenden wir einen alternativen Ansatz: Wir suchen lediglich nach einer *Näherungslösung*, die sich in endlich vielen Schritten bestimmen lässt. Dabei möchten wir natürlich dafür sorgen, dass diese Näherung hinreichend genau ist, wir wollen also den Fehler kontrollieren können.

Eine besonders einfache Technik ist das *Bisektionsverfahren*, das in jedem Schritt den Fehler mindestens halbiert und auf dem *Zwischenwertsatz für stetige Funktionen* beruht.

**Erinnerung 3.3 (Zwischenwertsatz)** Sei  $f : [a, b] \rightarrow \mathbb{R}$  eine stetige Funktion. Für jedes  $y \in \mathbb{R}$  zwischen  $f(a)$  und  $f(b)$  existiert ein  $x \in [a, b]$  mit  $f(x) = y$ .

Um das Bisektionsverfahren anwenden zu können, gehen wir davon aus, dass die Vorzeichen von  $f(a)$  und  $f(b)$  unterschiedlich sind. Dann liegt null zwischen beiden, und aus dem Zwischenwertsatz 3.3 folgt, dass ein  $x^* \in [a, b]$  mit  $f(x^*) = 0$  existieren muss.

Wie der Name schon andeutet, besteht die Idee des Bisektionsverfahrens darin, das Intervall zu unterteilen: Wir bezeichnen mit  $x = (a+b)/2$  seinen Mittelpunkt und untersuchen das Vorzeichen von  $f(x)$ . Falls es sich von dem Vorzeichen von  $f(a)$  unterscheidet, erfüllt auch das Intervall  $[a, x]$  unsere Voraussetzungen, so dass wir unsere Suche nach der Nullstelle mit diesem Intervall fortsetzen können. Anderenfalls, also falls  $f(a)$  und  $f(x)$  dasselbe Vorzeichen besitzen, muss sich das Vorzeichen von  $f(x)$  von dem von  $f(b)$  unterscheiden. Also erfüllt dann das Intervall  $[x, b]$  die Voraussetzungen, und wir setzen unsere Suche in diesem Intervall fort.

In beiden Fällen ist das neue Intervall halb so groß wie das ursprüngliche, wir haben also den Abstand zu der Nullstelle halbiert.

Wenn wir mit  $a^{(m)}$  und  $b^{(m)}$  die Grenzen der im  $m$ -ten Schritt verwendeten Intervalle bezeichnen, lässt sich das Verfahren kompakt in der Form

$$\begin{aligned} (a^{(0)}, b^{(0)}) &= (a, b), \\ x^{(m)} &= \frac{a^{(m)} + b^{(m)}}{2}, \\ (a^{(m+1)}, b^{(m+1)}) &= \begin{cases} (a^{(m)}, x^{(m)}) & \text{falls } f(a^{(m)})f(x^{(m)}) < 0, \\ (x^{(m)}, b^{(m)}) & \text{anderenfalls} \end{cases} \quad \text{für alle } m \in \mathbb{N}_0 \end{aligned}$$

zusammenfassen. Die Bedingung  $f(a^{(m)})f(x^{(m)}) < 0$  ist dabei gerade äquivalent dazu, dass sich die Vorzeichen der beiden Faktoren unterscheiden, dass also eine Nullstelle im Intervall  $[a^{(m)}, x^{(m)}]$  existieren muss.

Wenn wir die Werte der Funktion  $f$  im linken und rechten Endpunkt des jeweils aktuellen Intervalls speichern, können wir das Bisektionsverfahren so organisieren, dass nur eine Auswertung der Funktion pro Schritt erforderlich ist:

```

procedure bisection( $f, \epsilon, \text{var } a, b$ );
 $f_a \leftarrow f(a); f_b \leftarrow f(b);$ 
while  $b - a > \epsilon$  do begin
   $x \leftarrow (a + b)/2;$ 
   $f_x \leftarrow f(x);$ 
  if  $f_a f_x < 0$  do begin
     $b \leftarrow x; f_b \leftarrow f_x$ 
  else begin
     $a \leftarrow x; f_a \leftarrow f_x$ 
  end
end

```

Falls dieser Algorithmus mit  $f(a)f(b) < 0$  aufgerufen wird, berechnet er ein Intervall der Länge höchstens  $\epsilon > 0$ , das eine Nullstelle enthält. Da die Länge des Intervalls in jedem Schritt halbiert wird, sind dafür gerade

$$m := \lceil \log_2((b-a)/\epsilon) \rceil$$

Schritte erforderlich, da

$$(b^{(m)} - a^{(m)}) = (b-a)2^{-m} \leq (b-a)2^{-\log_2((b-a)/\epsilon)} = (b-a)\frac{\epsilon}{b-a} = \epsilon$$

gilt. Dafür benötigen wir  $m+2$  Auswertungen der Funktion  $f$  und  $2m$  Rechenoperationen, der Algorithmus wird also relativ schnell eine relativ gute Genauigkeit erreichen.

Da  $[a^{(m)}, b^{(m)}]$  eine Nullstelle enthalten muss, kann der Intervallmittelpunkt  $x^{(m)}$  höchstens einen Abstand von  $(b^{(m)} - a^{(m)})/2$  zu dieser Nullstelle haben, also folgt

$$|x^{(m)} - x^*| \leq \frac{\epsilon}{2}.$$

Wir können mit Hilfe des Mittelwertsatzes der Differentialrechnung relativ einfach

$$|f(x^{(m)})| = |f(x^{(m)}) - f(x^*)| \leq |f'(\eta)| |x^{(m)} - x^*| \leq \frac{|f'(\eta)|}{2} \epsilon$$

für einen Zwischenpunkt  $\eta \in [a^{(m)}, b^{(m)}]$  folgern. Natürlich können wir alternativ auch einfach  $f(x^{(m)})$  direkt berechnen.

Gegenüber anderen Algorithmen für das Lösen von Nullstellenproblemen bietet das Bisektionsverfahren den Vorteil sehr hoher Stabilität: Jeder Schritt muss das Intervall halbieren, und jedes so konstruierte Intervall muss eine Nullstelle enthalten. Leider lässt es sich nur auf reellwertige Funktionen auf geeigneten Intervallen anwenden.

### 3.3 Allgemeine Fixpunktiterationen

Das Bisektionsverfahren berechnet eine Folge von Näherungslösungen  $x^{(m)}$ , die zunehmend genauer werden und insbesondere gegen die exakte Lösung  $x^*$  konvergieren. Algorithmen, bei denen aus einer Näherung mit Hilfe einer geeigneten Rechenvorschrift eine verbesserte Näherung gewonnen wird, bezeichnet man als *Iterationsverfahren*.

Wir interessieren uns hier für eine relativ allgemeine Form dieser Verfahren:

### 3 Nichtlineare Gleichungssysteme

**Definition 3.4 (Iteration)** Sei  $U \subseteq \mathbb{R}^n$  eine abgeschlossene Teilmenge, und sei

$$\Phi : U \rightarrow U$$

eine stetige Abbildung. Dann bezeichnen wir  $\Phi$  als Iteration oder Iterationsabbildung auf  $U$ .

Eine gut gewählte Iteration sollte eine Folge von Näherungslösungen eines zu lösenden Problems berechnen:

**Definition 3.5 (Folge der Iterierten)** Sei  $\mathbf{x}^{(0)} \in U$ . Die durch

$$\mathbf{x}^{(m+1)} := \Phi(\mathbf{x}^{(m)}) \quad \text{für alle } m \in \mathbb{N}_0$$

definierte Folge  $(\mathbf{x}^{(m)})_{m \in \mathbb{N}_0}$  nennen wir die Folge der Iterierten zu dem Startwert  $\mathbf{x}^{(0)}$ .

Wir sind daran interessiert, eine Iteration so zu konstruieren, dass die Folge der Iterierten gegen die gesuchte Lösung  $\mathbf{x}^*$  konvergiert. Natürlich wünschen wir uns, dass die Anwendung der Iteration eine Näherungslösung verbessert, zumindest jedoch sollte sie sie möglichst nicht verschlechtern. Insbesondere sollte

$$\Phi(\mathbf{x}^*) = \mathbf{x}^*$$

gelten, damit wir die Lösung nicht wieder verlassen, sobald wir sie gefunden haben. Die Lösung muss also ein *Fixpunkt* der Iteration  $\Phi$  sein.

Für das folgende Beispiel benötigen wir den *Mittelwertsatz der Differentialrechnung*:

**Erinnerung 3.6 (Mittelwertsatz der Differentialrechnung)** Seien  $n \in \mathbb{N}$  und eine stetig differenzierbare Abbildung  $f : [a, b] \rightarrow \mathbb{R}^n$  gegeben. Dann existiert ein  $\eta \in (a, b)$  mit  $(b - a)f'(\eta) = f(b) - f(a)$ .

**Beispiel 3.7 (Nullstellensuche)** (vgl. [1, Beispiel 5.7]) Wir suchen nach einer Nullstelle  $x^* \in [1, 2]$  des Polynoms

$$p(x) = x^6 - x - 1.$$

Es gelten  $p(1) = -1$  und  $p(2) = 61$ , also muss in diesem Intervall mindestens eine Nullstelle existieren. Diese Nullstelle erfüllt

$$0 = (x^*)^6 - x^* - 1, \quad x^* = (x^*)^6 - 1,$$

also bietet es sich an,  $\Phi_1(x) := x^6 - 1$  als Iterationsabbildung zu verwenden. Wir untersuchen, mit Hilfe des Mittelwertsatzes der Differentialrechnung aus Erinnerung 3.6, wie sich der Fehler  $x^{(m)} - x^*$  im Laufe der Iteration verhält: Es gilt

$$|x^{(m+1)} - x^*| = |\Phi_1(x^{(m)}) - \Phi_1(x^*)| = |\Phi_1'(\eta)| |x^{(m)} - x^*|$$

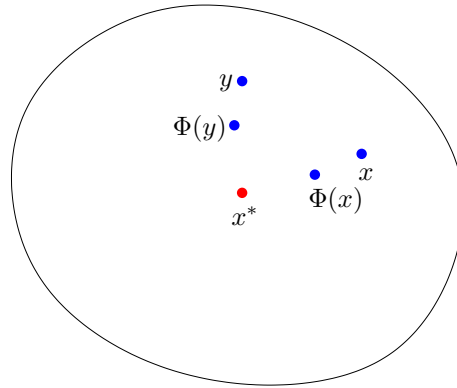


Abbildung 3.1: Wirkung einer Kontraktion

für ein  $\eta \in [1, 2]$ , und aus  $\Phi'_1(\eta) = 6\eta^5 \geq 6$  folgt

$$|x^{(m+1)} - x^*| \geq 6|x^{(m)} - x^*|,$$

der Fehler wird also mit jedem Schritt des Verfahrens mindestens um den Faktor sechs wachsen.

Wir können alternativ auch

$$0 = (x^*)^6 - x^* - 1, \quad (x^*)^6 = x^* + 1, \quad x^* = \sqrt[6]{x^* + 1}$$

verwenden und die Iteration  $\Phi_2(x) := \sqrt[6]{x+1}$  benutzen. Auch für sie gilt  $\Phi_2(x^*) = x^*$ , aber ihre Ableitung erfüllt

$$|\Phi'_2(x)| = \frac{1}{6}(x+1)^{-5/6} \leq \frac{1}{6} \quad \text{für alle } x \in [1, 2],$$

so dass wir

$$|x^{(m+1)} - x^*| = |\Phi'_2(\eta)| |x^{(m)} - x^*| \leq \frac{1}{6}|x^{(m)} - x^*|$$

für einen Zwischenpunkt  $\eta \in [1, 2]$  erhalten. Diese Iteration ist offensichtlich wesentlich attraktiver, weil sie den Fehler in jedem Schritt um einen Faktor sechs reduziert.

Es reicht also nicht aus, eine Iteration zu finden, die die gesuchte Lösung  $\mathbf{x}^*$  als Fixpunkt besitzt, wir müssen auch dafür sorgen, dass die Folge der Iterierten konvergiert.

**Satz 3.8 (Fixpunktsatz von Banach)** Sei  $\Phi$  eine Iteration auf einer abgeschlossenen Menge  $U \subseteq \mathbb{R}^n$ . Sei  $L \in [0, 1)$  so gegeben, dass

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \quad \text{für alle } \mathbf{x}, \mathbf{y} \in U \quad (3.2)$$

### 3 Nichtlineare Gleichungssysteme

gilt. Eine Abbildung mit dieser Eigenschaft bezeichnet man auch als Kontraktion. Dann besitzt  $\Phi$  genau einen Fixpunkt, und die Folge der Iterierten konvergiert für jeden Startwert  $\mathbf{x}^{(0)} \in U$  gegen diesen Fixpunkt. Den Fehler können wir durch

$$\|\mathbf{x}^{(m)} - \mathbf{x}^*\| \leq \frac{L^m}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|, \quad (3.3a)$$

$$\|\mathbf{x}^{(m)} - \mathbf{x}^*\| \leq \frac{1}{1-L} \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)}\| \quad \text{für alle } m \in \mathbb{N}_0 \quad (3.3b)$$

abschätzen und so die Qualität einer Näherung beurteilen.

*Beweis.* Sei  $\mathbf{x}^{(0)} \in U$ , und sei  $(\mathbf{x}^{(m)})_{m \in \mathbb{N}_0}$  die zugehörige Folge der Iterierten.

Wir zeigen zunächst

$$\|\mathbf{x}^{(m+1)} - \mathbf{x}^{(m)}\| \leq L^m \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \quad \text{für alle } m \in \mathbb{N}_0. \quad (3.4)$$

Der Induktionsanfang  $m = 0$  ist trivial. Gelte nun die Ungleichung für ein  $m \in \mathbb{N}_0$ . Dann folgt aus (3.2) und der Induktionsvoraussetzung die Abschätzung

$$\begin{aligned} \|\mathbf{x}^{(m+2)} - \mathbf{x}^{(m+1)}\| &= \|\Phi(\mathbf{x}^{(m+1)}) - \Phi(\mathbf{x}^{(m)})\| \leq L \|\mathbf{x}^{(m+1)} - \mathbf{x}^{(m)}\| \\ &\leq LL^m \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = L^{m+1} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|, \end{aligned}$$

und der Induktionsbeweis ist vollständig.

Sei nun  $n \in \mathbb{N}$ . Dann folgt aus (3.4) die Abschätzung

$$\begin{aligned} \|\mathbf{x}^{(m+n)} - \mathbf{x}^{(m)}\| &= \left\| \sum_{\ell=0}^{n-1} (\mathbf{x}^{(m+\ell+1)} - \mathbf{x}^{(m+\ell)}) \right\| \leq \sum_{\ell=0}^{n-1} \|\mathbf{x}^{(m+\ell+1)} - \mathbf{x}^{(m+\ell)}\| \\ &\leq \sum_{\ell=0}^{n-1} L^{m+\ell} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = L^m \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \sum_{\ell=0}^{n-1} L^\ell, \end{aligned}$$

und mit der geometrischen Summenformel

$$\sum_{\ell=0}^{n-1} L^\ell = \frac{1-L^n}{1-L} \leq \frac{1}{1-L}$$

erhalten wir die Abschätzung

$$\|\mathbf{x}^{(m+n)} - \mathbf{x}^{(m)}\| \leq \frac{L^m}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \quad \text{für alle } n \in \mathbb{N}. \quad (3.5)$$

Für jedes beliebige  $\epsilon \in \mathbb{R}_{>0}$  finden wir demnach ein  $m_0 \in \mathbb{N}$  mit

$$\frac{L^{m_0}}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \leq \epsilon,$$

und für alle  $m \in \mathbb{N}_{\geq m_0}$  und alle  $n \in \mathbb{N}$  gilt

$$\|\mathbf{x}^{(m+n)} - \mathbf{x}^{(m)}\| \leq \frac{L^m}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \leq \frac{L^{m_0}}{1-L} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \leq \epsilon,$$

### 3.3 Allgemeine Fixpunktiterationen

so dass wir feststellen dürfen, dass die Folge der Iterierten eine Cauchy-Folge ist. Da  $U$  eine abgeschlossene Teilmenge eines vollständigen Raums ist, muss diese Cauchy-Folge einen Grenzwert  $\mathbf{x}^* \in U$  besitzen. Da  $\Phi$  stetig ist, muss dieser Grenzwert auch

$$\Phi(\mathbf{x}^*) = \lim_{m \rightarrow \infty} \Phi(\mathbf{x}^{(m)}) = \lim_{m \rightarrow \infty} \mathbf{x}^{(m+1)} = \mathbf{x}^*$$

erfüllen, er ist also der gesuchte Fixpunkt. Um nachzuweisen, dass er auch der einzige ist, wählen wir ein  $\mathbf{y}^* \in U$  mit  $\Phi(\mathbf{y}^*) = \mathbf{y}^*$ . Dann gilt

$$\|\mathbf{x}^* - \mathbf{y}^*\| = \|\Phi(\mathbf{x}^*) - \Phi(\mathbf{y}^*)\| \leq L\|\mathbf{x}^* - \mathbf{y}^*\|,$$

also insbesondere

$$(1 - L)\|\mathbf{x}^* - \mathbf{y}^*\| \leq 0.$$

Da  $1 - L$  echt positiv ist, muss  $\|\mathbf{x}^* - \mathbf{y}^*\| = 0$  gelten, also auch  $\mathbf{x}^* = \mathbf{y}^*$ , also ist  $\mathbf{x}^*$  der einzige Fixpunkt.

Die Fehlerabschätzung (3.3a) erhalten wir, indem wir in (3.5) zu dem Grenzwert  $n \rightarrow \infty$  übergehen. Für den Nachweis der Fehlerabschätzung (3.3b) verwenden wir die Dreiecksungleichung wie folgt: Für alle  $m \in \mathbb{N}_0$  gilt

$$\begin{aligned} \|\mathbf{x}^{(m)} - \mathbf{x}^*\| &= \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)} + \mathbf{x}^{(m+1)} - \mathbf{x}^*\| \leq \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)}\| + \|\mathbf{x}^{(m+1)} - \mathbf{x}^*\| \\ &= \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)}\| + \|\Phi(\mathbf{x}^{(m)}) - \Phi(\mathbf{x}^*)\| \leq \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)}\| + L\|\mathbf{x}^{(m)} - \mathbf{x}^*\|, \end{aligned}$$

und indem wir den zweiten Term auf die linke Seite bringen folgt

$$(1 - L)\|\mathbf{x}^{(m)} - \mathbf{x}^*\| \leq \|\mathbf{x}^{(m)} - \mathbf{x}^{(m+1)}\|,$$

so dass wir den Beweis abschließen können, indem wir durch  $1 - L$  dividieren.  $\blacksquare$

Die Fehlerabschätzung (3.3a) bezeichnet man als *A-priori-Abschätzung*, weil sie bereits eine Aussage über den zu erwarteten Fehler trifft, bevor wir die entsprechende Iterierte überhaupt berechnet haben. Derartige Abschätzungen sind nützlich, um vorherzusagen, wieviele Schritte ein iterativer Algorithmus höchstens brauchen wird.

Die Fehlerabschätzung (3.3b) bezeichnet man dagegen als *A-posteriori-Abschätzung*, weil sie eine Aussage über den Fehler der  $m$ -ten Iterierten trifft, sobald sie berechnet wurde. Abschätzungen dieses Typs sind hilfreich, um während der Durchführung eines iterativen Algorithmus festzustellen, ob die berechnete Näherung bereits genau genug ist und der Algorithmus beendet werden kann.

Wir können Satz 3.8 auch auf unser Beispiel 3.7 anwenden: Die Iteration  $\Phi_2$  erfüllt

$$|\Phi_2(x) - \Phi_2(y)| = |\Phi_2'(\eta)| |x - y| \leq \frac{1}{6} |x - y| \quad \text{für alle } x, y \in [1, 2],$$

wobei  $\eta \in [1, 2]$  wieder von  $x$  und  $y$  abhängige Zwischenpunkte sind. Demzufolge ist  $\Phi_2$  auf  $U = [1, 2]$  eine Kontraktion mit  $L = 1/6$ . Da  $\Phi_2$  monoton ist, garantieren

$$\Phi_2(1) = \sqrt[6]{2} \geq 1, \quad \Phi_2(2) = \sqrt[6]{3} \leq 2,$$

dass  $\Phi_2$  das Intervall  $[1, 2]$  auf sich selbst abbildet, also können wir Satz 3.8 anwenden.

### 3.4 Newton-Verfahren

Wir kommen nun zurück zu der ursprünglichen Aufgabe, eine Nullstelle zu finden. Sei dazu  $U \subseteq \mathbb{R}^n$  eine offene Menge, und

$$f : U \rightarrow \mathbb{R}^n$$

eine zweimal stetig differenzierbare Funktion, die eine Nullstelle  $\mathbf{x}^* \in U$  besitzt, so dass

$$f(\mathbf{x}^*) = \mathbf{0} \tag{3.6}$$

gilt. Unser Ziel ist es nun, eine Iteration  $\Phi$  zu konstruieren, die diese Nullstelle als Fixpunkt besitzt und zu einem möglichst schnell konvergierenden Verfahren führt.

**Erinnerung 3.9 (Taylor)** Seien  $n \in \mathbb{N}$  und eine  $m$ -mal stetig differenzierbare Abbildung  $f : [a, b] \rightarrow \mathbb{R}^n$  gegeben. Dann existiert ein  $\eta \in [a, b]$  mit

$$f(b) = \sum_{k=0}^{m-1} f^{(k)}(a) \frac{(b-a)^k}{k!} + f^{(m)}(\eta) \frac{(b-a)^m}{m!}.$$

Zur Motivation untersuchen wir zunächst den Fall  $n = 1$ . Mit Hilfe des Satzes von Taylor aus Erinnerung 3.9 erhalten wir

$$0 = f(x^*) = f(x) + f'(x)(x^* - x) + \frac{f''(\eta)}{2}(x^* - x)^2 \tag{3.7}$$

für ein  $\eta \in [x, x^*]$ . Wir nehmen an, dass  $f'(x) \neq 0$  gilt, und dividieren durch diesen Wert, um

$$0 = \frac{f(x)}{f'(x)} + x^* - x + \frac{f''(\eta)}{2f'(x)}(x^* - x)^2$$

zu erhalten. In dieser Gleichung können wir  $f(x)$  und  $f'(x)$  praktisch berechnen, während  $x^*$  und  $f''(\eta)$  unbekannt sind, und indem wir alle berechenbaren Größen auf die linke Seite bringen folgt

$$x - \frac{f(x)}{f'(x)} = x^* + \frac{f''(\eta)}{2f'(x)}(x^* - x)^2. \tag{3.8}$$

Falls  $|x^* - x|$  klein ist, dürfen wir also davon ausgehen, dass die linke Seite eine gute Näherung der Nullstelle  $x^*$  ist.

**Definition 3.10 (Newton-Verfahren)** Sei  $U \subseteq \mathbb{R}$ , und sei  $f \in C^1(U)$  mit  $0 \notin f'(U)$ . Dann ist

$$\Phi : U \rightarrow \mathbb{R}, \quad x \mapsto x - \frac{f(x)}{f'(x)}, \tag{3.9}$$

die Rechenvorschrift des eindimensionalen Newton-Verfahrens.



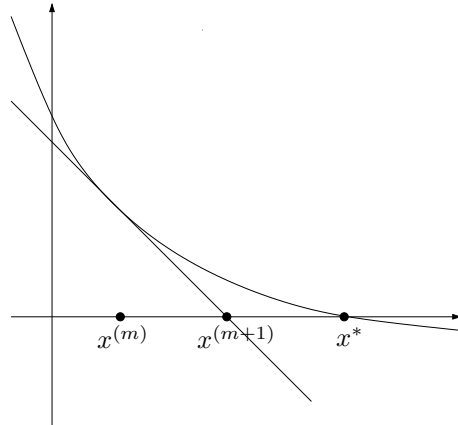


Abbildung 3.2: Geometrische Interpretation des Newton-Verfahrens

Das Newton-Verfahren lässt sich geometrisch interpretieren (vgl. Abbildung 3.2): Indem wir den dritten Term auf der rechten Seite der Gleichung (3.7) wegfassen lassen, approximieren wir die Funktion  $f$  durch ihre Tangente im Punkt  $x$ . Die Nullstelle dieser Tangente ist dann die nächste Iterierte  $\Phi(x)$ .

Um zu zeigen, dass  $\Phi$  eine Iteration ist, müssen wir nachweisen, dass es eine Selbstabbildung ist. Das ist nur sicher gestellt, wenn wir uns auf einer Menge bewegen, die hinreichend nahe an  $x^*$  ist, damit der zweite Term in (3.8) nicht stört.

**Lemma 3.11 (Konvergenz)** Sei  $r \in \mathbb{R}_{>0}$  gegeben, sei  $U := (x^* - r, x^* + r)$ , und gelte

- $f \in C^2(U)$ ,
- $|1/f'(x)| \leq C_1$  für alle  $x \in U$ ,
- $|f''(x)| \leq C_2$  für alle  $x \in U$  und
- $r \leq \frac{2}{C_1 C_2}$ .

Dann ist die in (3.9) definierte Abbildung  $\Phi$  eine Selbstabbildung auf dem Intervall  $U$ , und es gilt

$$|\Phi(x) - x^*| \leq \frac{C_1 C_2}{2} |x - x^*|^2 \quad \text{für alle } x \in U. \quad (3.10)$$

*Beweis.* Aus (3.8) folgt

$$|\Phi(x) - x^*| = \frac{|f''(\eta)|}{2|f'(x)|} |x^* - x|^2 \leq \frac{C_1 C_2}{2} |x^* - x|^2$$

für einen Zwischenpunkt  $\eta \in U$ , und dank unserer Voraussetzungen erhalten wir

$$|\Phi(x) - x^*| < \frac{C_1 C_2}{2} r^2 \leq \frac{C_1 C_2}{2} \frac{2}{C_1 C_2} r \leq r,$$

### 3 Nichtlineare Gleichungssysteme

also insbesondere  $\Phi(x) \in U$ . ■

Insbesondere wird das Newton-Verfahren konvergieren, falls der Startwert  $x^{(0)}$  in dem Intervall  $U$  liegt. Die Folge der Iterierten konvergiert um so schneller, je näher die Iterierten der Lösung liegen: Falls beispielsweise

$$|x^{(0)} - x^*| \leq \frac{1}{C_1 C_2}$$

gelten sollte, erhalten wir

$$\begin{aligned} |x^{(1)} - x^*| &\leq \frac{C_1 C_2}{2} |x^{(0)} - x^*|^2 \leq \frac{2}{4 C_1 C_2}, \\ |x^{(2)} - x^*| &\leq \frac{C_1 C_2}{2} |x^{(1)} - x^*|^2 \leq \frac{2}{16 C_1 C_2}, \\ |x^{(3)} - x^*| &\leq \frac{C_1 C_2}{2} |x^{(2)} - x^*|^2 \leq \frac{2}{256 C_1 C_2}, \end{aligned}$$

und mit einer einfachen Induktion folgt

$$|x^{(m)} - x^*| \leq \frac{2}{C_1 C_2} 2^{-2^m} \quad \text{für alle } m \in \mathbb{N}_0,$$

der Fehler wird also sehr schnell reduziert, insbesondere deutlich schneller als bei konventionellen Fixpunkt-Iterationen, bei der jeder Schritt nur eine Reduktion um einen festen Faktor  $L$  bewirkt.

Diese besonders schnelle Konvergenz verdanken wir der Tatsache, dass auf der rechten Seite der Abschätzung (3.10) das Quadrat des ursprünglichen Fehlers auftritt. Verfahren, die derartige Konvergenzabschätzungen der Form

$$\|\Phi(\mathbf{x}) - \mathbf{x}^*\| \leq C \|\mathbf{x} - \mathbf{x}^*\|^2$$

erfüllen, bezeichnen wir als *quadratisch konvergent*. Sie bieten uns die Möglichkeit, sehr genaue Näherungen der Lösung  $\mathbf{x}^*$  in relativ wenigen Iterationsschritten zu bestimmen.

Natürlich sind wir daran interessiert, die ermutigenden Ergebnisse von dem eindimensionalen Fall auf den Fall beliebiger Dimension zu übertragen. Wir fixieren eine Näherung  $\mathbf{x} \in \mathbb{R}^n$  und beschreiben das Geradenstück von  $\mathbf{x}$  zu der Lösung  $\mathbf{x}^*$  durch die Funktion

$$\gamma : [0, 1] \rightarrow \mathbb{R}^n, \quad t \mapsto \mathbf{x} + t(\mathbf{x}^* - \mathbf{x}),$$

mit deren Hilfe wir es uns ersparen können, im  $n$ -dimensionalen Raum arbeiten zu müssen. Stattdessen untersuchen wir die Funktion  $f$  nur noch entlang des durch  $\gamma$  beschriebenen Geradenstücks, beschränken uns also auf die Hilfsfunktion

$$g : [0, 1] \rightarrow \mathbb{R}^n, \quad t \mapsto f(\gamma(t)).$$

Die beiden Hilfsfunktionen besitzen die folgenden Eigenschaften:

$$\gamma(0) = \mathbf{x}, \quad \gamma(1) = \mathbf{x}^*, \quad (3.11a)$$

$$g(0) = f(\mathbf{x}), \quad g(1) = f(\mathbf{x}^*), \quad (3.11b)$$

$$\gamma'(t) = \mathbf{x}^* - \mathbf{x}, \quad g'(t) = Df(\gamma(t))(\mathbf{x}^* - \mathbf{x}) \quad \text{für alle } t \in [0, 1]. \quad (3.11c)$$

Die letzte Gleichung lässt sich leicht mit Hilfe der Kettenregel nachprüfen, die ersten beiden folgen direkt aus der Definition.

Bei der Herleitung des mehrdimensionalen Newton-Verfahrens gehen wir wie bei der eindimensionalen Variante vor, verwenden aber den Hauptsatz der Integral- und Differentialrechnung anstelle des Satzes von Taylor, um höhere Ableitungen zu vermeiden:

**Erinnerung 3.12 (Integral- und Differentialrechnung)** Seien  $n \in \mathbb{N}$  und eine stetig differenzierbare Funktion  $f : [a, b] \rightarrow \mathbb{R}^n$  gegeben. Dann gilt

$$f(b) - f(a) = \int_a^b f'(t) dt.$$

Den Ausgangspunkt der Herleitung des mehrdimensionalen Newton-Verfahrens bildet die unmittelbar aus diesem Hauptsatz und (3.11b) folgende Gleichung

$$\mathbf{0} = f(\mathbf{x}^*) = g(1) = g(0) + \int_0^1 g'(t) dt.$$

Im Allgemeinen steht uns  $g'(t)$  nicht zur Verfügung, da in seiner Definition die unbekannte Lösung  $\mathbf{x}^*$  vorkommt. Wir können allerdings als Approximation  $g'(0)$  verwenden und erhalten

$$\begin{aligned} \mathbf{0} &= g(0) + \int_0^1 g'(t) dt = g(0) + \int_0^1 g'(0) dt + \int_0^1 g'(t) - g'(0) dt \\ &= g(0) + g'(0) + \int_0^1 g'(t) - g'(0) dt. \end{aligned}$$

Durch Einsetzen von (3.11b) und (3.11c) ergibt sich

$$\mathbf{0} = f(\mathbf{x}) + Df(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) + \int_0^1 g'(t) - g'(0) dt.$$

Wir zuvor möchten wir den zweiten Term auf der rechten Seite verwenden, um  $\mathbf{x}^*$  zu approximieren. Dazu gehen wir davon aus, dass die Jacobi-Matrix  $Df(\mathbf{x})$  regulär ist, und multiplizieren die Gleichung mit ihrer Inversen, um

$$\mathbf{0} = Df(\mathbf{x})^{-1}f(\mathbf{x}) + \mathbf{x}^* - \mathbf{x} + Df(\mathbf{x})^{-1} \int_0^1 g'(t) - g'(0) dt$$

zu erhalten. Indem wir die praktisch berechenbaren Terme wieder auf die linke Seite bringen, folgt

$$\mathbf{x} - Df(\mathbf{x})^{-1}f(\mathbf{x}) = \mathbf{x}^* + Df(\mathbf{x})^{-1} \int_0^1 g'(t) - g'(0) dt. \quad (3.12)$$

Falls der zweiten Term auf der rechten Seite klein ist, können wir die linke Seite als Näherung der Nullstelle  $\mathbf{x}^*$  verwenden.

### 3 Nichtlineare Gleichungssysteme

**Definition 3.13 (Newton-Verfahren)** Sei  $U \subseteq \mathbb{R}^n$ , und sei  $Df(\mathbf{x})$  für alle  $\mathbf{x} \in U$  regulär. Dann ist

$$\Phi : U \rightarrow \mathbb{R}^d, \quad \mathbf{x} \mapsto \mathbf{x} - Df(\mathbf{x})^{-1}f(\mathbf{x}), \quad (3.13)$$

die Rechenvorschrift des mehrdimensionalen Newton-Verfahrens.

Nach Definition und (3.12) gilt

$$\Phi(\mathbf{x}) - \mathbf{x}^* = Df(\mathbf{x})^{-1} \int_0^1 g'(t) - g'(0) dt,$$

also müssen wir diesen Term beschränken, um eine Konvergenzaussage zu erhalten. Dazu untersuchen wir zunächst den Integralterm, der dank (3.11c) die Form

$$\int_0^1 g'(t) - g'(0) dt = \int_0^1 (Df(\gamma(t)) - Df(\gamma(0))) (\mathbf{x}^* - \mathbf{x}) dt$$

besitzt. Das Produkt aus der Matrix  $Df(\gamma(t)) - Df(\gamma(0))$  und dem Vektor  $\mathbf{x}^* - \mathbf{x}$  können wir mit Hilfe der Spektralnorm abschätzen, erhalten also

$$\|(Df(\gamma(t)) - Df(\gamma(0))) (\mathbf{x}^* - \mathbf{x})\|_2 \leq \|Df(\gamma(t)) - Df(\gamma(0))\|_2 \|\mathbf{x}^* - \mathbf{x}\|_2.$$

Wenn wir quadratische Konvergenz erhalten wollen, müssen wir aus dem ersten Faktor einen weiteren Term der Form  $\|\mathbf{x}^* - \mathbf{x}\|$  herausziehen. Das gelingt uns relativ einfach, wenn wir voraussetzen, dass die Jacobi-Matrix *Lipschitz-stetig* von ihrem Argument abhängt.

Indem wir das in Lemma 3.11 verwendete Intervall  $U$  durch eine Kugel

$$K(\mathbf{x}^*, r) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}^*\| < r\}$$

ersetzen, erhalten wir die folgende verallgemeinerte Form unserer Konvergenzaussage:

**Satz 3.14 (Konvergenz)** Sei  $r \in \mathbb{R}_{>0}$  gegeben, sei  $U := K(\mathbf{x}^*, r)$ , und gelte

- $f \in C^1(U, \mathbb{R}^d)$ ,
- $\|Df(\mathbf{x})^{-1}\|_2 \leq C_1$  für alle  $\mathbf{x} \in U$ ,
- $\|Df(\mathbf{x}) - Df(\mathbf{y})\|_2 \leq C_2 \|\mathbf{x} - \mathbf{y}\|_2$  für alle  $\mathbf{x}, \mathbf{y} \in U$ ,
- $r \leq \frac{2}{C_1 C_2}$ .

Dann ist die in (3.13) definierte Abbildung  $\Phi$  eine Selbstabbildung auf der Kugel  $U$ , und es gilt

$$\|\Phi(\mathbf{x}) - \mathbf{x}^*\|_2 \leq \frac{C_1 C_2}{2} \|\mathbf{x} - \mathbf{x}^*\|_2^2 \quad \text{für alle } \mathbf{x} \in U.$$

*Beweis.* Sei  $\mathbf{x} \in U$ . Aus (3.12) und unserer zweiten Voraussetzung folgt

$$\|\Phi(\mathbf{x}) - \mathbf{x}^*\|_2 \leq \left\| Df(\mathbf{x})^{-1} \int_0^1 g'(t) - g'(0) dt \right\|_2 \leq C_1 \left\| \int_0^1 g'(t) - g'(0) dt \right\|_2.$$

Wir wenden die Dreiecksungleichung für Integrale an und erhalten

$$\begin{aligned} \|\Phi(\mathbf{x}) - \mathbf{x}^*\|_2 &\leq C_1 \int_0^1 \|g'(t) - g'(0)\|_2 dt \\ &= C_1 \int_0^1 \|(Df(\gamma(t)) - Df(\gamma(0)))(\mathbf{x}^* - \mathbf{x})\|_2 dt \\ &\leq C_1 \int_0^1 \|Df(\gamma(t)) - Df(\gamma(0))\|_2 \|\mathbf{x}^* - \mathbf{x}\|_2 dt. \end{aligned}$$

Nun benutzen wir die dritte Voraussetzung und finden

$$\|Df(\gamma(t)) - Df(\gamma(0))\|_2 \leq C_2 \|\gamma(t) - \gamma(0)\|_2 = C_2 \|\mathbf{x} + t(\mathbf{x}^* - \mathbf{x}) - \mathbf{x}\|_2 = C_2 t \|\mathbf{x}^* - \mathbf{x}\|_2,$$

so dass sich insgesamt die Abschätzung

$$\|\Phi(\mathbf{x}) - \mathbf{x}^*\|_2 \leq C_1 \int_0^1 C_2 t \|\mathbf{x}^* - \mathbf{x}\|_2^2 dt = C_1 C_2 \|\mathbf{x}^* - \mathbf{x}\|_2^2 \int_0^1 t dt = \frac{C_1 C_2}{2} \|\mathbf{x}^* - \mathbf{x}\|_2$$

ergibt. Die restlichen Aussagen folgen wie im Beweis von Lemma 3.11.  $\blacksquare$

Wir erhalten also auch im mehrdimensionalen Fall quadratische Konvergenz, sogar unter etwas schwächeren Voraussetzungen, denn wir müssen nicht mehr fordern, dass  $f$  zweimal stetig differenzierbar ist, stattdessen genügt die Lipschitz-Stetigkeit der ersten Ableitung.

**Bemerkung 3.15 (Umsetzung)** *Die Durchführung des Newton-Verfahrens gemäß der Formel (3.13) erfordert die Berechnung der Inversen  $Df(\mathbf{x})^{-1}$  der Jacobi-Matrix im Punkt  $\mathbf{x}$ . Wesentlich günstiger und effizienter ist es, stattdessen ein lineares Gleichungssystem mit den in Kapitel 2 vorgestellten Verfahren zu lösen: Wenn wir die nächste Iterierte*

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - Df(\mathbf{x}^{(m)})^{-1} f(\mathbf{x}^{(m)})$$

*suchen, können wir auch*

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \mathbf{d}^{(m)}, \quad \mathbf{d}^{(m)} := -Df(\mathbf{x}^{(m)})^{-1} f(\mathbf{x}^{(m)})$$

*verwenden und die Newton-Richtung  $\mathbf{d}^{(m)} \in \mathbb{R}^n$  als Lösung des linearen Gleichungssystems*

$$Df(\mathbf{x}^{(m)}) \mathbf{d}^{(m)} = -f(\mathbf{x}^{(m)})$$

*bestimmen. Dieser Zugang ist in der Regel effizienter und numerisch stabiler als die Berechnung der Inversen.*

### 3 Nichtlineare Gleichungssysteme

**Bemerkung 3.16 (Gedämpftes Verfahren)** Bei Wahl eines Startwerts  $\mathbf{x}^{(0)}$ , der nicht die Voraussetzungen des Satzes 3.14 erfüllt, beispielsweise weil die Konstanten  $C_1$  und  $C_2$  nicht bekannt sind, kann die Newton-Iteration divergieren. In diesem Fall hilft es manchmal, die Newton-Richtung mit einem hinreichend kleinen Faktor zu multiplizieren, der dafür sorgt, dass der Fehler zumindest nicht größer als im vorangehenden Schritt werden kann. Man erhält das gedämpfte Newton-Verfahren

$$Df(\mathbf{x}^{(m)})\mathbf{d}^{(m)} = -f(\mathbf{x}^{(m)}), \quad \mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \sigma^{(m)}\mathbf{d}^{(m)}.$$

Es gibt verschiedene Strategien für die geschickte Wahl des Dämpfungsparameters  $\sigma^{(m)} \in \mathbb{R}$ , die zu je nach Anwendung unterschiedlich guten Ergebnissen führen.

## 4 Eigenwertprobleme

### 4.1 Beispiel: Schwingende Saite

Eine Variante des linearen Gleichungssystems ist das *Eigenwertproblem*. Als Beispiel verwenden wir die *Wellengleichung*

$$\frac{\partial^2 u}{\partial x^2}(x, t) = \frac{\partial^2 u}{\partial t^2}(x, t) \quad \text{für alle } x \in [0, 1], t \in \mathbb{R}, \quad (4.1)$$

die das Verhalten einer schwingenden Saite modelliert, auf die keine externen Kräfte einwirken. Die Funktion  $u$  beschreibt dabei die Auslenkung der Saite aus der Ruhelage: Wenn wir uns die Saite im Ruhezustand als waagrecht liegend vorstellen, beschreibt  $u(x, t)$ , wie sehr sie in einem Punkt  $x$  zu einem Zeitpunkt  $t$  nach oben ausgelenkt ist. Um zu einem eindeutig lösbaeren Problem zu gelangen, müssen wir Randbedingungen ergänzen. Wir entscheiden uns dafür, die Saite in den Punkten  $x = 0$  und  $x = 1$  in der Ruhelage einzuspannen, verwenden also

$$u(0) = 0, \quad u(1) = 0. \quad (4.2)$$

Wir lösen die Gleichung mit Hilfe einer Separation der Variablen: Mit dem Ansatz

$$u(x, t) = u_0(x) \sin \omega t \quad \text{für alle } x \in [0, 1], t \in \mathbb{R}$$

nimmt (4.1) die Form

$$u_0''(x) \sin \omega t = -\omega^2 u_0(x) \sin \omega t, \quad \text{für alle } x \in [0, 1], t \in \mathbb{R}$$

an, und indem wir den Sinus-Term eliminieren folgt

$$-u_0''(x) = \omega^2 u_0(x) \quad \text{für alle } x \in [0, 1]. \quad (4.3)$$

Es lässt sich nachweisen, dass diese Gleichung (mit den durch (4.2) gegebenen Randbedingungen) genau dann lösbar ist, wenn  $\omega = \pi \ell$  für ein  $\ell \in \mathbb{Z}$  gilt, und in diesem Fall gilt gerade  $u_0(x) = \sin(\omega x)$ .

Wir sind allerdings daran interessiert, Gleichungen dieses Typs im Allgemeinen zu lösen. Dazu ersetzen wir wieder das Intervall  $[0, 1]$  durch eine diskrete Punktmenge  $\{t_0, t_1, \dots, t_n, t_{n+1}\}$ , die Funktion  $u_0$  durch einen Vektor  $\mathbf{x} \in \mathbb{R}^n$  mit  $u_0(t_i) \approx x_i$ , und den Differentialoperator durch die Matrix  $\mathbf{A}$  aus (2.3), so dass wir

$$\mathbf{A}\mathbf{x} = \omega^2 \mathbf{x}$$

## 4 Eigenwertprobleme

erhalten. Die triviale Lösung der Gleichung ist  $\mathbf{x} = \mathbf{0}$ , jeden nicht-trivialen Lösungsvektor  $\mathbf{x} \neq \mathbf{0}$  bezeichnen wir als *Eigenvektor* zu dem *Eigenwert*  $\lambda = \omega^2$ .

Im Allgemeinen untersuchen wir Eigenwertprobleme der Form

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}, \quad \mathbf{x} \neq \mathbf{0}, \quad (4.4)$$

die in vielen Gebieten der Mechanik und Elektrodynamik eine Rolle spielen, weil sich mit ihnen beispielsweise Resonanzphänomene wie die schwingende Saite untersuchen lassen.

### 4.2 Vektoriteration

Wir können Eigenwertprobleme einfach in die Form

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$$

eines linearen Gleichungssystems bringen, allerdings können wir dieses System nicht mit den zuvor behandelten Verfahren lösen, weil uns erstens  $\lambda$  in der Regel nicht bekannt ist und zweitens für jeden Eigenwert  $\lambda$  die Matrix auf der linken Seite nicht regulär ist, so dass sich unsere bisherigen Techniken nicht anwenden lassen. Es lässt sich sogar beweisen, dass es kein Verfahren gibt, das in endlich vielen Schritten das Eigenwertproblem löst. Deshalb müssen wir wie schon in Kapitel 3 auf iterative Näherungsverfahren zurückgreifen.

Der einfachste Lösungsansatz ist die *Vektoriteration*: Wir gehen zur Vereinfachung davon aus, dass  $\mathbf{A} = \mathbf{A}^*$  gilt, dass wir es also mit einer symmetrischen Matrix zu tun haben. Derartige Matrizen lassen sich mit der *Hauptachsentransformation* auf Diagonalgestalt bringen, genauer gesagt gibt es eine orthogonale Matrix  $\mathbf{U} \in \mathbb{R}^{n \times n}$  derart, dass

$$\mathbf{U}^* \mathbf{A} \mathbf{U} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} =: \mathbf{D}$$

gilt. Wir dürfen sogar die Reihenfolge der Diagonalelemente der Matrix beliebig vorgeben und entscheiden uns für eine dem Betrag nach absteigende Sortierung, gehen also von

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| \quad (4.5)$$

aus. Die Vektoriteration wird durch die folgende Beobachtung motiviert: Wenn wir einen Vektor  $\hat{\mathbf{x}}^{(0)} \in \mathbb{R}^n$  mit der  $m$ -ten Potenz der Matrix  $\mathbf{D}$  multiplizieren, erhalten wir neue Vektoren

$$\hat{\mathbf{x}}^{(m)} := \mathbf{D}^m \hat{\mathbf{x}}^{(0)} = \begin{pmatrix} \lambda_1^m \hat{x}_1^{(0)} \\ \vdots \\ \lambda_n^m \hat{x}_n^{(0)} \end{pmatrix} \quad \text{für alle } m \in \mathbb{N}_0.$$

Da die Eigenwerte dem Betrag nach absteigend sortiert sind, werden sich für große Werte von  $m$  die ersten Komponenten gegenüber dem Rest durchsetzen.



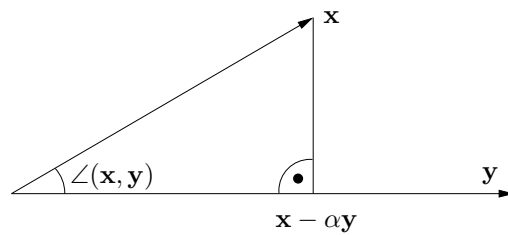


Abbildung 4.1: Definition des Winkels

Besonders einfach ist der Fall eines *dominanten Eigenwerts*: Falls

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n| \quad (4.6)$$

gilt, dürfen wir davon ausgehen, dass  $|\lambda_1|^m$  schneller als die Potenzen aller anderen Eigenwerte wachsen wird, so dass die Vektoren  $\hat{\mathbf{x}}^{(m)}$  in einem geeigneten Sinn gegen ein Vielfaches des ersten Einheitsvektors

$$\hat{\mathbf{e}}^{(1)} := \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

konvergieren dürften. Der erste Einheitsvektor ist offenbar gerade ein Eigenvektor zu dem Eigenwert  $\lambda_1$ , also erhalten wir Konvergenz gegen einen Eigenvektor.

In der Praxis kennen wir  $\mathbf{D}$  nicht und arbeiten stattdessen direkt mit der Matrix  $\mathbf{A}$ : Für einen gegebenen Anfangsvektor  $\mathbf{x}^{(0)}$  definieren wir

$$\mathbf{x}^{(m)} := \mathbf{A}^m \mathbf{x}^{(0)} \quad \text{für alle } m \in \mathbb{N}_0 \quad (4.7)$$

und stellen fest, dass

$$\mathbf{U}^* \mathbf{x}^{(m)} = \mathbf{U}^* \mathbf{A}^m \mathbf{x}^{(0)} = \mathbf{U}^* \mathbf{A}^m \mathbf{U} \mathbf{U}^* \mathbf{x}^{(0)} = \mathbf{D}^m \mathbf{U}^* \mathbf{x}^{(0)} \quad \text{für alle } m \in \mathbb{N}_0$$

gilt. Es bietet sich an,

$$\hat{\mathbf{x}}^{(0)} := \mathbf{U}^* \mathbf{x}^{(0)}$$

zu setzen und so

$$\hat{\mathbf{x}}^{(m)} = \mathbf{U}^* \mathbf{x}^{(m)} \quad \text{für alle } m \in \mathbb{N}_0$$

zu erhalten. Die Vektoren  $\mathbf{x}^{(m)}$  können wir dabei mit (4.7) direkt berechnen, die Vektoren  $\hat{\mathbf{x}}^{(m)}$  dienen als Hilfsmittel für die Untersuchung der Konvergenz.

„Echte Konvergenz“ der Form  $\mathbf{x}^{(m)} \rightarrow \mathbf{x}^*$  dürfen wir dabei nicht erwarten, denn selbst wenn  $\mathbf{x}^{(0)}$  bereits ein Eigenvektor zu einem Eigenwert  $\lambda_1$  wäre, würde  $\mathbf{x}^{(m)} = \lambda^m \mathbf{x}^{(0)}$

#### 4 Eigenwertprobleme

gelten, die Länge des Vektors könnte sich also ändern. Da die Länge von Eigenvektoren keine Rolle spielt, bietet es sich an, einen Konvergenzbegriff zu verwenden, der sie nicht berücksichtigt. Ein nützliches Hilfsmittel ist dabei der *Winkel* zwischen zwei Vektoren, der durch

$$\sin \angle(\mathbf{x}, \mathbf{y}) = \min \left\{ \frac{\|\mathbf{x} - \alpha \mathbf{y}\|_2}{\|\mathbf{x}\|_2} : \alpha \in \mathbb{R} \right\} \quad \text{für alle } \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \mathbf{y} \in \mathbb{R}^n$$

definiert ist: Für das minimale  $\alpha$  steht  $\mathbf{x} - \alpha \mathbf{y}$  senkrecht auf  $\alpha \mathbf{y}$ , so dass wir ein rechtwinkliges Dreieck mit Hypotenuse  $\mathbf{x}$  und Kathete  $\mathbf{x} - \alpha \mathbf{y}$  erhalten (vgl. Abbildung 4.1). Wie üblich definieren wir Cosinus und Tangens durch

$$\cos \angle(\mathbf{x}, \mathbf{y}) := \sqrt{1 - \sin^2 \angle(\mathbf{x}, \mathbf{y})}, \quad \tan \angle(\mathbf{x}, \mathbf{y}) := \frac{\sin \angle(\mathbf{x}, \mathbf{y})}{\cos \angle(\mathbf{x}, \mathbf{y})}.$$

und erhalten das folgende Konvergenzresultat für unsere Iteration:

**Satz 4.1 (Konvergenz)** *Der Vektor  $\mathbf{e}^{(1)} := \mathbf{U}\hat{\mathbf{e}}^{(1)}$  ist ein Eigenvektor zu dem Eigenwert  $\lambda_1$ . Es gelte*

$$\tan \angle(\mathbf{x}^{(0)}, \mathbf{e}^{(1)}) < \infty,$$

$\mathbf{x}^{(0)}$  soll also nicht senkrecht auf  $\mathbf{e}^{(1)}$  stehen. Dann haben wir

$$\tan \angle(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \left( \frac{|\lambda_2|}{|\lambda_1|} \right)^m \tan \angle(\mathbf{x}^{(0)}, \mathbf{e}^{(1)}) \quad \text{für alle } m \in \mathbb{N}_0.$$

*Beweis.* Die Multiplikation mit einer orthogonalen Matrix lässt nach (2.17) die Norm unverändert, also gilt

$$\tan \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)}) = \tan \angle(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \quad \text{für alle } m \in \mathbb{N}_0, \quad (4.8)$$

so dass es genügt, die transformierten Vektoren zu untersuchen. Für alle  $\alpha \in \mathbb{R}$  gilt

$$\hat{\mathbf{x}}^{(m)} - \alpha \hat{\mathbf{e}}^{(1)} = \begin{pmatrix} \lambda_1^m \hat{x}_1^{(0)} - \alpha \\ \lambda_2^m \hat{x}_2^{(0)} \\ \vdots \\ \lambda_n^m \hat{x}_n^{(0)} \end{pmatrix},$$

und die Norm dieses Vektors ist offenbar gerade für  $\alpha_m := \lambda_1^m \hat{x}_1^{(0)}$  minimal. Damit folgen

$$\begin{aligned} \sin^2 \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)}) &= \frac{\|\hat{\mathbf{x}}^{(m)} - \alpha_m \hat{\mathbf{e}}^{(1)}\|_2^2}{\|\hat{\mathbf{x}}^{(m)}\|_2^2} = \frac{\sum_{i=2}^n |\lambda_i|^{2m} |\hat{x}_i^{(0)}|^2}{\sum_{i=1}^n |\lambda_i|^{2m} |\hat{x}_i^{(0)}|^2}, \\ \cos^2 \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)}) &= 1 - \sin^2 \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)}) = \frac{|\lambda_1|^{2m} |\hat{x}_1^{(0)}|^2}{\sum_{i=1}^n |\lambda_i|^{2m} |\hat{x}_i^{(0)}|^2}, \\ \tan^2 \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)}) &= \frac{\sin^2 \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)})}{\cos^2 \angle(\hat{\mathbf{x}}^{(m)}, \hat{\mathbf{e}}^{(1)})} = \frac{\sum_{i=2}^n |\lambda_i|^{2m} |\hat{x}_i^{(0)}|^2}{|\lambda_1|^{2m} |\hat{x}_1^{(0)}|^2} \quad \text{für alle } m \in \mathbb{N}_0. \end{aligned}$$

Dank (4.5) folgt daraus insbesondere

$$\begin{aligned}\tan^2 \angle(\widehat{\mathbf{x}}^{(m)}, \widehat{\mathbf{e}}^{(1)}) &= \frac{\sum_{i=2}^n |\lambda_i|^{2m} |\widehat{x}_i^{(0)}|^2}{|\lambda_1|^{2m} |\widehat{x}_1^{(0)}|^2} \leq \frac{|\lambda_2|^{2m} \sum_{i=2}^n |\widehat{x}_i^{(0)}|^2}{|\lambda_1|^{2m} |\widehat{x}_1^{(0)}|^2} \\ &= \left( \frac{|\lambda_2|}{|\lambda_1|} \right)^{2m} \tan^2 \angle(\widehat{\mathbf{x}}^{(0)}, \widehat{\mathbf{e}}^{(1)}) \quad \text{für alle } m \in \mathbb{N}_0,\end{aligned}$$

und mit (4.8) erhalten wir das gewünschte Resultat.  $\blacksquare$

**Bemerkung 4.2 (Diagonalisierbare Matrizen)** *Das Resultat des Satzes 4.1 lässt sich auf diagonalisierbare Matrizen übertragen: Falls eine reguläre Matrix  $\mathbf{T} \in \mathbb{R}^{n \times n}$  so existiert, dass*

$$\mathbf{T}^{-1} \mathbf{A} \mathbf{T} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} = \mathbf{D}$$

*gilt, erhalten wir ein Konvergenzresultat für die transformierten Vektoren  $\widehat{\mathbf{x}}^{(m)} = \mathbf{T}^{-1} \mathbf{x}^{(m)}$ . Allerdings müssen wir anstelle von (4.8) Ungleichungen der Form*

$$\sin \angle(\mathbf{x}, \mathbf{y}) \leq \kappa_2(\mathbf{T}) \sin \angle(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}), \quad \sin \angle(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) \leq \kappa_2(\mathbf{T}) \sin \angle(\mathbf{x}, \mathbf{y})$$

*in Kauf nehmen, wenn wir daraus Konvergenzaussagen für den Winkel zwischen  $\mathbf{x}^{(m)}$  und  $\mathbf{e}^{(1)} := \mathbf{T} \widehat{\mathbf{e}}^{(1)}$  herleiten wollen.*

In der Praxis hat die bisher betrachtete Folge  $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots$  den Nachteil, dass sie für  $|\lambda_1| > 1$  zu Vektoren mit sehr großen und für  $|\lambda_1| < 1$  zu Vektoren mit sehr kleinen Einträgen führt, die Schwierigkeiten mit dem im Rechner darstellbaren endlichen Zahlenbereich nach sich ziehen. Da uns nur das Verhältnis der Komponenten zueinander, beziehungsweise der Winkel zwischen  $\mathbf{x}^{(m)}$  und  $\mathbf{e}^{(1)}$ , interessiert, können wir die Vektoren so skalieren, dass dieses Problem ausgeschlossen ist. Wir erhalten die folgende Variante der Vektoriteration:

$$\mathbf{y}^{(m)} = \mathbf{A} \mathbf{x}^{(m-1)}, \quad \gamma^{(m)} = \|\mathbf{y}^{(m)}\|_2, \quad \mathbf{x}^{(m)} = \mathbf{y}^{(m)} / \gamma^{(m)} \quad \text{für alle } m \in \mathbb{N} \quad (4.9)$$

Indem wir in jedem Schritt die Norm berechnen und durch sie dividieren, ist sicher gestellt, dass die Vektoren  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$  Einheitsvektoren sind, und damit sind ihre Komponenten insbesondere weder zu klein noch zu groß. Die Vektoren aus (4.7) und (4.9) unterscheiden sich lediglich durch die Skalierungsfaktoren, also bleibt die Aussage des Satzes 4.1 unverändert gültig.

Der bisherige Algorithmus hat den Nachteil, dass wir die Anzahl der Iterationsschritte vorgeben müssen. Praktischer wäre ein *adaptiver* Algorithmus, dem wir nur mitteilen, wie genau die Näherung mindestens sein sollte, und der dann die Anzahl der Iterationsschritte automatisch steuert.

Eine Möglichkeit besteht darin, zu den Näherungsvektoren einen genäherten Eigenwert zu konstruieren und zu prüfen, wie groß der Fehler der definierenden Gleichung (4.4) für diese Werte ist.

#### 4 Eigenwertprobleme

**Definition 4.3 (Rayleigh-Quotient)** Der Rayleigh-Quotient zu einer Matrix  $\mathbf{A}$  ist gegeben durch

$$\Lambda_A : \mathbb{R}^n \setminus \{\mathbf{0}\} \rightarrow \mathbb{R}, \quad \mathbf{x} \mapsto \frac{\langle \mathbf{A}\mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2}.$$

Falls  $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  ein Eigenvektor der Matrix  $\mathbf{A}$  zu einem Eigenwert  $\lambda \in \mathbb{R}$  ist, gilt

$$\Lambda_A(\mathbf{x}) = \frac{\langle \mathbf{A}\mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2} = \frac{\langle \lambda\mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2} = \lambda \frac{\langle \mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2} = \lambda,$$

der Rayleigh-Quotient reproduziert also zu einem gegebenen Eigenvektor den passenden Eigenwert. Um auch eine Aussage für *Näherungen* von Eigenvektoren zu erhalten benötigen wir das folgende Hilfsmittel:

**Erinnerung 4.4 (Cauchy-Schwarz)** Für das Skalarprodukt gilt die Cauchy-Schwarz-Ungleichung

$$|\langle \mathbf{x}, \mathbf{y} \rangle_2| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \quad \text{für alle } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \quad (4.10)$$

Mit Hilfe dieser Ungleichung lässt sich die Genauigkeit der durch den Rayleigh-Quotienten berechnen Näherung des Eigenwerts wie folgt abschätzen:

**Lemma 4.5 (Rayleigh-Quotient)** Sei  $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  eine Näherung eines Eigenvektors  $\mathbf{e} \in \mathbb{R}^n$  der Matrix  $\mathbf{A}$  für den Eigenwert  $\lambda \in \mathbb{R}$ . Dann gilt

$$|\Lambda_A(\mathbf{x}) - \lambda| \leq \|\mathbf{A} - \lambda\mathbf{I}\|_2 \sin \angle(\mathbf{x}, \mathbf{e}) \leq \|\mathbf{A} - \lambda\mathbf{I}\|_2 \tan \angle(\mathbf{x}, \mathbf{e}).$$

*Beweis.* Da  $\mathbf{e}$  ein Eigenvektor ist, gilt insbesondere

$$(\mathbf{A} - \lambda\mathbf{I})\alpha\mathbf{e} = \mathbf{A}\alpha\mathbf{e} - \lambda\alpha\mathbf{e} = \mathbf{0} \quad \text{für alle } \alpha \in \mathbb{R},$$

und wir erhalten mit der Cauchy-Schwarz-Ungleichung und (2.12) die Abschätzung

$$\begin{aligned} |\Lambda_A(\mathbf{x}) - \lambda| &= \left| \frac{\langle \mathbf{A}\mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2} - \frac{\langle \lambda\mathbf{x}, \mathbf{x} \rangle_2}{\langle \mathbf{x}, \mathbf{x} \rangle_2} \right| = \frac{|\langle (\mathbf{A} - \lambda\mathbf{I})\mathbf{x}, \mathbf{x} \rangle_2|}{\|\mathbf{x}\|_2^2} \\ &= \frac{|\langle (\mathbf{A} - \lambda\mathbf{I})(\mathbf{x} - \alpha\mathbf{e}), \mathbf{x} \rangle_2|}{\|\mathbf{x}\|_2^2} \leq \frac{\|(\mathbf{A} - \lambda\mathbf{I})(\mathbf{x} - \alpha\mathbf{e})\|_2 \|\mathbf{x}\|_2}{\|\mathbf{x}\|_2^2} \\ &\leq \frac{\|\mathbf{A} - \lambda\mathbf{I}\|_2 \|\mathbf{x} - \alpha\mathbf{e}\|_2}{\|\mathbf{x}\|_2} \leq \|\mathbf{A} - \lambda\mathbf{I}\|_2 \frac{\|\mathbf{x} - \alpha\mathbf{e}\|_2}{\|\mathbf{x}\|_2} \quad \text{für alle } \alpha \in \mathbb{R}. \end{aligned}$$

Durch Wahl des optimalen  $\alpha \in \mathbb{R}$  folgt der erste Teil der Behauptung, wegen  $\cos \angle(\mathbf{x}, \mathbf{e}) \leq 1$  der zweite.  $\blacksquare$

Für symmetrische (oder allgemeiner für sogenannte *normale*) Matrizen lässt sich diese Abschätzung erheblich verbessern:

**Bemerkung 4.6 (Quadratische Konvergenz)** Falls  $\mathbf{A} = \mathbf{A}^*$  gilt, können wir

$$\langle (\mathbf{A} - \lambda \mathbf{I})(\mathbf{x} - \alpha \mathbf{e}), \mathbf{x} \rangle_2 = \langle \mathbf{x} - \alpha \mathbf{e}, (\mathbf{A} - \lambda \mathbf{I})\mathbf{x} \rangle_2$$

ausnutzen, um auch im zweiten Argument  $\alpha \mathbf{e}$  abzuziehen und so zu der verbesserten Abschätzung

$$|\Lambda_A(\mathbf{x}) - \lambda| \leq \|\mathbf{A} - \lambda \mathbf{I}\|_2 \sin^2 \angle(\mathbf{x}, \mathbf{e})$$

zu gelangen. In diesem Fall kann die Näherung des Eigenwerts also unter Umständen wesentlich schneller als die des Eigenvektors konvergieren.

Indem wir mit Hilfe des Rayleigh-Quotienten eine Näherung des Eigenwerts berechnen, können wir die Vektoriteration so gestalten, dass sie erst beendet wird, wenn eine gegebene Genauigkeit erreicht ist:

```

procedure power_adaptive( $\epsilon$ ,  $\mathbf{A}$ , var  $\mathbf{x}$ );
 $\gamma \leftarrow \|\mathbf{x}\|_2$ ;    $\mathbf{x} \leftarrow \mathbf{x}/\gamma$ ;
 $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ ;
 $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ ;
while  $\|\lambda \mathbf{x} - \mathbf{y}\|_2 > \epsilon \|\mathbf{y}\|_2$  do begin
   $\gamma \leftarrow \|\mathbf{y}\|_2$ ;    $\mathbf{x} \leftarrow \mathbf{y}/\gamma$ ;
   $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ ;
   $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ 
end

```

Um mit möglichst wenigen Matrix-Vektor-Multiplikationen auskommen zu können, verwenden wir das in  $\mathbf{y}$  gespeicherte Produkt  $\mathbf{A}\mathbf{x}$  in diesem Algorithmus mehrfach: Für die Berechnung des Rayleigh-Quotienten, für die Prüfung auf Konvergenz, und für die Bestimmung der nächsten Iterierten. Bei der Berechnung des Rayleigh-Quotienten im Inneren der Schleife können wir uns die Division durch  $\|\mathbf{x}\|_2^2$  sparen, weil nach unserer Konstruktion  $\mathbf{x}$  an diesem Punkt des Algorithmus' ein Einheitsvektor ist.

### 4.3 Inverse Iteration

Die Vektoriteration eignet sich für die Berechnung des größten Eigenwerts, in der Praxis ist man allerdings häufig eher an den kleinsten Eigenwerten interessiert, beispielsweise an der niedrigsten Frequenz eines Systems, bei der Resonanzeffekte zu erwarten sind.

Falls  $\mathbf{A}$  regulär ist, können wir

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x} \quad \iff \quad \mathbf{x} = \lambda \mathbf{A}^{-1} \mathbf{x} \quad \iff \quad \frac{1}{\lambda} \mathbf{x} = \mathbf{A}^{-1} \mathbf{x}$$

ausnutzen: Jeder Eigenvektor der Matrix  $\mathbf{A}$  zu einem Eigenwert  $\lambda$  ist auch ein Eigenvektor der Matrix  $\mathbf{A}^{-1}$  zu dem Eigenwert  $1/\lambda$ . Damit ist insbesondere der betragskleinste Eigenwert von  $\mathbf{A}$  gerade der Kehrwert des betragsgrößten Eigenwerts von  $\mathbf{A}^{-1}$ , so dass wir ihn mit Hilfe der *inversen Iteration* berechnen können, die durch

$$\mathbf{y}^{(m)} = \mathbf{A}^{-1} \mathbf{x}^{(m-1)}, \quad \gamma^{(m)} = \|\mathbf{y}^{(m)}\|_2, \quad \mathbf{x}^{(m)} = \mathbf{y}^{(m)} / \gamma^{(m)} \quad \text{für alle } m \in \mathbb{N}$$

#### 4 Eigenwertprobleme

gegeben ist. Da uns die Inverse  $\mathbf{A}^{-1}$  in der Regel nicht zur Verfügung steht, bestimmen wir  $\mathbf{y}^{(m)}$  als Lösung des linearen Gleichungssystems

$$\mathbf{A}\mathbf{y}^{(m)} = \mathbf{x}^{(m-1)},$$

beispielsweise mit Hilfe einer LR- oder QR-Zerlegung.

Mit Vektoriteration und inverser Iteration können wir die größten und kleinsten Eigenwerte berechnen, also stellt sich die Frage, ob sich auch Eigenwerte zwischen beiden Extremen behandeln lassen. Dazu modifizieren wir die inverse Iteration: Falls  $\mu \in \mathbb{R}$  kein Eigenwert der Matrix  $\mathbf{A}$  ist, ist die Matrix  $\mathbf{A} - \mu\mathbf{I}$  regulär. Falls  $\lambda \in \mathbb{R}$  ein Eigenwert von  $\mathbf{A}$  mit Eigenvektor  $\mathbf{x} \in \mathbb{R}^n$  ist, gilt

$$\begin{aligned}\mathbf{A}\mathbf{x} = \lambda\mathbf{x} &\iff (\mathbf{A} - \mu\mathbf{I})\mathbf{x} = (\lambda - \mu)\mathbf{x} \\ &\iff \mathbf{x} = (\lambda - \mu)(\mathbf{A} - \mu\mathbf{I})^{-1}\mathbf{x} \\ &\iff \frac{1}{\lambda - \mu}\mathbf{x} = (\mathbf{A} - \mu\mathbf{I})^{-1}\mathbf{x},\end{aligned}$$

also ist  $1/(\lambda - \mu)$  ein Eigenwert der Matrix  $(\mathbf{A} - \mu\mathbf{I})^{-1}$ . Der betragsgrößte Eigenwert dieser Matrix korrespondiert also mit dem Eigenwert der Matrix  $\mathbf{A}$ , der  $\mu$  am nächsten liegt. Wir können uns also aussuchen, an welchen Eigenwerten wir interessiert sind, und diese Eigenwerte gezielt berechnen.

Das resultierende Verfahren

$$\mathbf{y}^{(m)} = (\mathbf{A} - \mu\mathbf{I})^{-1}\mathbf{x}^{(m-1)}, \quad \gamma^{(m)} = \|\mathbf{y}^{(m)}\|_2, \quad \mathbf{x}^{(m)} = \mathbf{y}^{(m)}/\gamma^{(m)} \quad \text{für alle } m \in \mathbb{N}$$

trägt den Namen *inverse Iteration mit Shift*, und der die „Verschiebung“ der Eigenwerte beschreibende Parameter  $\mu$  wird als *Shift-Parameter* bezeichnet.

Bei der Umsetzung der inversen Iteration zeigt sich, dass es sehr nützlich ist, dass wir das Lösen linearer Gleichungssysteme in eine relativ zeitaufwendige Vorbereitungsphase, in unserem Fall die Berechnung der LR- oder QR-Zerlegung, und eine effiziente eigentliche Lösungsphase, nämlich das Vorwärts- und Rückwärtseinsetzen, zerlegt haben. Deshalb brauchen wir die Zerlegung nur einmal zu berechnen und können dann beliebig viele Iterationsschritte mit relativ geringem Aufwand durchführen.

```
procedure invit_adaptive( $\epsilon, \mu, \mathbf{A}, \mathbf{var} \mathbf{x}$ );  
  Berechne eine Faktorisierung der Matrix  $\mathbf{A} - \mu\mathbf{I}$ ;  
   $\gamma \leftarrow \|\mathbf{x}\|_2$ ;  $\mathbf{x} \leftarrow \mathbf{x}/\gamma$ ;  
  Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ ;  
   $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ ;  
  while  $\|\lambda\mathbf{x} - \mathbf{y}\|_2 > \epsilon\|\mathbf{y}\|_2$  do begin  
     $\gamma \leftarrow \|\mathbf{y}\|_2$ ;  $\mathbf{x} \leftarrow \mathbf{y}/\gamma$ ;  
    Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ ;  
     $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$   
  end
```

Bei diesem Algorithmus ist zu beachten, dass der Rayleigh-Quotient  $\lambda$  nun gegen einen Eigenwert der Matrix  $(\mathbf{A} - \mu\mathbf{I})^{-1}$  konvergieren wird, so dass der korrespondierende Eigenwert der ursprünglichen Matrix  $\mathbf{A}$  mit Hilfe der Formel  $1/\lambda + \mu$  rekonstruiert werden muss.

Die Untersuchung der Konvergenz der inversen Iteration lässt sich auf die der Vektoriteration zurückführen: Wir setzen wieder voraus, dass  $\mathbf{A} = \mathbf{A}^*$  gilt und sich die Matrix mit einer orthogonalen Matrix  $\mathbf{U} \in \mathbb{R}^{n \times n}$  diagonalisieren lässt, dass also

$$\mathbf{U}^* \mathbf{A} \mathbf{U} = \mathbf{D} = \text{diag}(\lambda_1, \dots, \lambda_n)$$

gilt. Da wir die Vektoriteration auf  $(\mathbf{A} - \mu\mathbf{I})^{-1}$  anwenden, müssen wir die Eigenwerte nun gemäß

$$|\lambda_1 - \mu| < |\lambda_2 - \mu| \leq \dots \leq |\lambda_n - \mu|$$

anordnen und voraussetzen, dass genau ein einfacher Eigenwert  $\mu$  am nächsten liegt. Wir bezeichnen wieder mit  $\mathbf{e}^{(1)} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  einen Eigenvektor der Matrix  $\mathbf{A}$  zu dem Eigenwert  $\lambda_1$  und mit  $\hat{\mathbf{x}}^{(0)} = \mathbf{U}^* \mathbf{x}^{(0)}$  die Transformation des Anfangsvektors.

**Folgerung 4.7 (Konvergenz)** *Es gelte*

$$\tan \angle(\mathbf{x}^{(0)}, \mathbf{e}^{(1)}) < \infty.$$

*Dann erfüllen die Iterierten der inversen Iteration mit Shift die Abschätzung*

$$\tan \angle(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \left( \frac{|\lambda_1 - \mu|}{|\lambda_2 - \mu|} \right)^m \tan \angle(\mathbf{x}^{(0)}, \mathbf{e}^{(1)}) \quad \text{für alle } m \in \mathbb{N}_0.$$

*Beweis.* Da die Eigenwerte der Matrix  $(\mathbf{A} - \mu\mathbf{I})^{-1}$  durch

$$\frac{1}{|\lambda_1 - \mu|} > \frac{1}{|\lambda_2 - \mu|} \geq \dots \geq \frac{1}{|\lambda_n - \mu|}$$

gegeben sind, folgt die Aussage direkt aus Satz 4.1. ■

Der große Vorteil der inversen Iteration mit Shift besteht darin, dass man durch geschickte Wahl des Parameters  $\mu$  die Konvergenzgeschwindigkeit verbessern kann: Je näher  $\mu$  dem Eigenwert  $\lambda_1$  ist, desto schneller wird die Folge gegen den Eigenvektor konvergieren.

Falls  $\mathbf{x}^{(m)}$  schon eine relativ gute Näherung eines Eigenvektors ist, dürfen wir laut Lemma 4.5 darauf hoffen, dass  $\Lambda_A(\mathbf{x}^{(m)})$  eine gute Näherung des entsprechenden Eigenwerts ist. Da eine gute Näherung eines Eigenwerts auch ein guter Shift-Parameter ist, bietet es sich an,  $\mu^{(m)} = \Lambda_A(\mathbf{x}^{(m-1)})$  zu verwenden, und wir erhalten die *Rayleigh-Iteration*:

$$\begin{aligned} \mu^{(m)} &= \Lambda_A(\mathbf{x}^{(m-1)}), \\ \mathbf{y}^{(m)} &= (\mathbf{A} - \mu^{(m)}\mathbf{I})^{-1} \mathbf{x}^{(m-1)}, \quad \mathbf{x}^{(m)} = \frac{\mathbf{y}^{(m)}}{\|\mathbf{y}^{(m)}\|_2} \quad \text{für alle } m \in \mathbb{N}. \end{aligned}$$

Da der Shift-Parameter nicht mehr von  $m$  unabhängig ist, muss in jedem Schritt der Iteration ein anderes lineares Gleichungssystem gelöst werden:

#### 4 Eigenwertprobleme

```

procedure invit_rayleigh( $\epsilon, \mu, \mathbf{A}, \text{var } \mathbf{x}$ );
 $\gamma \leftarrow \|\mathbf{x}\|_2$ ;    $\mathbf{x} \leftarrow \mathbf{x}/\gamma$ ;
Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ ;
 $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ ;    $\mu \leftarrow 1/\lambda + \mu$ ;
while  $\|\lambda\mathbf{x} - \mathbf{y}\|_2 > \epsilon\|\mathbf{y}\|_2$  do begin
   $\gamma \leftarrow \|\mathbf{y}\|_2$ ;    $\mathbf{x} \leftarrow \mathbf{y}/\gamma$ ;
  Löse  $(\mathbf{A} - \mu\mathbf{I})\mathbf{y} = \mathbf{x}$ ;
   $\lambda \leftarrow \langle \mathbf{y}, \mathbf{x} \rangle_2$ ;    $\mu \leftarrow 1/\lambda + \mu$ 
end

```

In diesem Algorithmus ist  $\lambda$  jeweils der Rayleigh-Quotient der Matrix  $(\mathbf{A} - \mu\mathbf{I})^{-1}$ , also eine Näherung eines ihrer Eigenwerte. Aus dieser Näherung rekonstruieren wir mit der bereits erwähnten Formel  $1/\lambda + \mu$  eine Näherung des korrespondierenden Eigenwerts der Matrix  $\mathbf{A}$ , der uns dann im nächsten Schritt der Iteration als Shift-Parameter dient. Da sich der Shift-Parameter in jedem Schritt des Verfahrens ändert, können wir nicht mehr die für das Lösen des linearen Gleichungssystems erforderlichen Operationen aus der zentralen Schleife heraushalten, so dass ein Schritt der Rayleigh-Iteration wesentlich zeitaufwendiger als einer der einfachen inversen Iteration werden kann.

Allerdings besitzt die Rayleigh-Iteration auch einen sehr erheblichen Vorteil: Falls uns bereits eine hinreichend gute Näherung zur Verfügung steht, konvergiert sie besonders schnell.

**Satz 4.8 (Konvergenz)** *Seien  $(\mathbf{x}^{(m)})_{m \in \mathbb{N}_0}$  die Vektoren der Rayleigh-Iteration. Falls*

$$\tan \angle(\mathbf{x}^{(m-1)}, \mathbf{e}^{(1)}) \leq \theta := \frac{|\lambda_1 - \lambda_2|}{2\|\mathbf{A} - \lambda_1\mathbf{I}\|_2}$$

*gilt, erhalten wir die Fehlerabschätzung*

$$\tan \angle(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \frac{2\|\mathbf{A} - \lambda_1\mathbf{I}\|_2}{|\lambda_1 - \lambda_2|} \tan^2 \angle(\mathbf{x}^{(m-1)}, \mathbf{e}^{(1)}).$$

*Insbesondere erhalten wir  $\tan \angle(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \theta$ , so dass sich die Abschätzung wiederholt anwenden lässt.*

*Beweis.* Gelte  $\tan \angle(\mathbf{x}^{(m-1)}, \mathbf{e}^{(1)}) \leq \theta$ . Aus Lemma 4.5 erhalten wir so

$$\begin{aligned} |\lambda_1 - \mu^{(m)}| &\leq \|\mathbf{A} - \lambda_1\mathbf{I}\|_2 \tan \angle(\mathbf{x}^{(m-1)}, \mathbf{e}^{(1)}) \leq |\lambda_1 - \lambda_2|/2, \\ |\lambda_2 - \mu^{(m)}| &\geq |\lambda_2 - \lambda_1| - |\lambda_1 - \mu^{(m)}| \geq |\lambda_1 - \lambda_2|/2. \end{aligned}$$

Aufgrund der Folgerung 4.7 ergibt sich

$$\tan \angle(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \frac{|\lambda_1 - \mu^{(m)}|}{|\lambda_2 - \mu^{(m)}|} \tan \angle(\mathbf{x}^{(m-1)}, \mathbf{e}^{(1)}) \leq \frac{\|\mathbf{A} - \lambda_1\mathbf{I}\|_2}{|\lambda_1 - \lambda_2|/2} \tan^2 \angle(\mathbf{x}^{(m-1)}, \mathbf{e}^{(1)}),$$

und das ist bereits die zu beweisende Aussage. ■



Die *quadratische Konvergenz* der Rayleigh-Iteration kann sehr nützlich sein, falls uns gute Ausgangsvektoren zur Verfügung stehen. Falls beispielsweise

$$\tan(\mathbf{x}^{(0)}, \mathbf{e}^{(1)}) \leq \frac{|\lambda_1 - \lambda_2|}{4\|\mathbf{A} - \lambda_1 \mathbf{I}\|_2}$$

gilt, erhalten wir mit einer einfachen Induktion

$$\tan(\mathbf{x}^{(m)}, \mathbf{e}^{(1)}) \leq \frac{|\lambda_1 - \lambda_2|}{2^{2^{m+1}}\|\mathbf{A} - \lambda_1 \mathbf{I}\|_2} \quad \text{für alle } m \in \mathbb{N}_0,$$

jeder Schritt *verdoppelt* also die Anzahl der korrekt berechneten Stellen der Lösung, so dass wir sehr schnelle Konvergenz erwarten dürfen, sobald eine hinreichend gute Näherung des Eigenvektors berechnet wurde.

## 4.4 Orthogonale Iteration

Die bisher besprochenen Verfahren setzen voraus, dass der betragsgrößte Eigenwert einfach ist. Da das in der Praxis oft nicht gewährleistet ist, verallgemeinern wir nun die Vektoriteration so, dass auch mehrfache Eigenwerte behandelt werden können.

Wir beschränken uns auf den Fall der Vektoriteration, Varianten wie die inverse und die Rayleigh-Iteration lassen sich entsprechend konstruieren. Anstelle von (4.6) setzen wir nun voraus, dass für ein  $k \in \{1, \dots, n-1\}$  die Ungleichungskette

$$|\lambda_1| \geq \dots \geq |\lambda_k| > |\lambda_{k+1}| \geq \dots \geq |\lambda_n| \quad (4.11)$$

gilt, es muss also erst der Betrag des  $k$ -ten Eigenwerts echt größer als der des  $(k+1)$ -ten Eigenwerts sein. Insbesondere sind  $k$ -fache Eigenwerte zugelassen.

Unter dieser schwächeren Voraussetzung können wir nicht mehr Konvergenz gegen einen Eigenvektor zu  $\lambda_1$  erwarten, stattdessen gegen einen Vektor aus dem durch die Eigenvektoren  $\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(k)} \in \mathbb{R}^n$  zu den ersten  $k$  Eigenwerten  $\lambda_1, \dots, \lambda_k$  aufgespannten *Teilraum*.

Ein einzelner Vektor aus einem  $k$ -dimensionalen Teilraum hilft uns kaum weiter, besser wäre eine *Basis* des Teilraums, also  $k$  linear unabhängige Vektoren. Wir können sie berechnen, indem wir die Vektoriteration mit mehreren Vektoren gleichzeitig durchführen. Um unnötige Indizes einzusparen, fassen wir diese Vektoren zu Matrizen  $\mathbf{X}^{(m)} \in \mathbb{R}^{n \times k}$  zusammen, so dass

$$\mathbf{X}^{(m)} := \mathbf{A}^m \mathbf{X}^{(0)} \quad \text{für alle } m \in \mathbb{N}_0 \quad (4.12)$$

an die Stelle der Gleichung (4.7) tritt.

Die ersten  $k$  Eigenvektoren fassen wir ebenfalls zu einer Matrix

$$\mathbf{E}^{(k)} := \mathbf{U} \widehat{\mathbf{E}}^{(k)}, \quad \widehat{\mathbf{E}}^{(k)} := \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \in \mathbb{R}^{n \times k}$$

#### 4 Eigenwertprobleme

zusammen und verwenden zur Vereinfachung der Analyse wieder transformierte Größen

$$\widehat{\mathbf{X}}^{(m)} := \mathbf{U}^* \mathbf{X}^{(m)} \quad \text{für alle } m \in \mathbb{N}_0.$$

Wie schon im Fall der Vektoriteration dürfen wir nur auf Konvergenz hoffen, wenn die erste Matrix  $\mathbf{X}^{(0)}$  Anteile der gesuchten Eigenvektoren aufweist. Diese Eigenschaft lässt sich elegant beschreiben, indem wir die *Projektion*

$$\mathbf{\Pi} := \mathbf{E}^{(k)} (\mathbf{E}^{(k)})^* = \mathbf{U} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{U}^*$$

auf den gesuchten Raum einführen: Wenn  $\mathbf{\Pi X}^{(0)}$  injektiv ist (also alle Spalten linear unabhängig sind), gilt dasselbe auch für alle anderen durch den Algorithmus berechneten Matrizen.

**Satz 4.9 (Konvergenz)** *Sei  $\mathbf{\Pi X}^{(0)}$  injektiv. Für jedes  $m \in \mathbb{N}_0$  ist dann auch  $\mathbf{\Pi X}^{(m)}$  injektiv und es existiert ein  $\mathbf{\Lambda} \in \mathbb{R}^{k \times k}$  mit*

$$\frac{\|(\mathbf{A X}^{(m)} - \mathbf{X}^{(m)} \mathbf{\Lambda}) \mathbf{y}\|_2}{\|\mathbf{\Pi X}^{(m)} \mathbf{y}\|_2} \leq \left( \frac{|\lambda_{k+1}|}{|\lambda_k|} \right)^m \frac{\|(\mathbf{A X}^{(0)} - \mathbf{X}^{(0)} \mathbf{\Lambda}) \mathbf{y}\|_2}{\|\mathbf{\Pi X}^{(0)} \mathbf{y}\|_2} \quad \text{für alle } \mathbf{y} \in \mathbb{R}^k. \quad (4.13)$$

*Beweis.* Wir zerlegen die Matrizen  $\mathbf{D}$  und  $\widehat{\mathbf{X}}^{(m)}$  in der Form

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_k & \\ & \mathbf{D}_\perp \end{pmatrix}, \quad \mathbf{D}_k := \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{pmatrix}, \quad \mathbf{D}_\perp := \begin{pmatrix} \lambda_{k+1} & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix},$$

$$\widehat{\mathbf{X}}^{(m)} = \begin{pmatrix} \widehat{\mathbf{X}}_k^{(m)} \\ \widehat{\mathbf{X}}_\perp^{(m)} \end{pmatrix}, \quad \widehat{\mathbf{X}}_k^{(m)} := \begin{pmatrix} \hat{x}_{11}^{(m)} & \cdots & \hat{x}_{1k}^{(m)} \\ \vdots & \ddots & \vdots \\ \hat{x}_{k1}^{(m)} & \cdots & \hat{x}_{kk}^{(m)} \end{pmatrix}, \quad \widehat{\mathbf{X}}_\perp^{(m)} := \begin{pmatrix} \hat{x}_{k+1,1}^{(m)} & \cdots & \hat{x}_{k+1,k}^{(m)} \\ \vdots & \ddots & \vdots \\ \hat{x}_{n1}^{(m)} & \cdots & \hat{x}_{nk}^{(m)} \end{pmatrix}.$$

Da  $\mathbf{\Pi X}^{(0)}$  injektiv ist, gilt dasselbe für

$$\mathbf{U}^* \mathbf{\Pi X}^{(0)} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \widehat{\mathbf{X}}^{(0)} = \begin{pmatrix} \widehat{\mathbf{X}}_k^{(0)} \\ \mathbf{0} \end{pmatrix},$$

also muss  $\widehat{\mathbf{X}}_k^{(0)}$  injektiv sein. Da diese Matrix quadratisch ist, ist sie auch invertierbar. Wir erhalten

$$\mathbf{U}^* \mathbf{\Pi X}^{(m)} = \mathbf{U}^* \mathbf{\Pi A}^m \mathbf{X}^{(0)} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{D}_k & \\ & \mathbf{D}_\perp \end{pmatrix}^m \begin{pmatrix} \widehat{\mathbf{X}}_k^{(0)} \\ \widehat{\mathbf{X}}_\perp^{(0)} \end{pmatrix} = \begin{pmatrix} \mathbf{D}_k^m \widehat{\mathbf{X}}_k^{(0)} \\ \mathbf{0} \end{pmatrix}, \quad (4.14)$$

und da nach Voraussetzung alle Diagonalelemente von  $\mathbf{D}_k$  ungleich null sind, muss  $\mathbf{D}_k^m \widehat{\mathbf{X}}_k^{(0)}$  regulär sein, also ist  $\mathbf{\Pi X}^{(m)}$  injektiv.

Für den Nachweis der zweiten Aussage untersuchen wir

$$\begin{aligned} \mathbf{U}^*(\mathbf{A}\mathbf{X}^{(m)} - \mathbf{X}^{(m)}\mathbf{\Lambda}) &= \mathbf{D}\widehat{\mathbf{X}}^{(m)} - \widehat{\mathbf{X}}^{(m)}\mathbf{\Lambda} = \begin{pmatrix} \mathbf{D}_k\widehat{\mathbf{X}}_k^{(m)} - \widehat{\mathbf{X}}_k^{(m)}\mathbf{\Lambda} \\ \mathbf{D}_\perp\widehat{\mathbf{X}}_\perp^{(m)} - \widehat{\mathbf{X}}_\perp^{(m)}\mathbf{\Lambda} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{D}_k^{m+1}\widehat{\mathbf{X}}_k^{(0)} - \mathbf{D}_k^m\widehat{\mathbf{X}}_k^{(0)}\mathbf{\Lambda} \\ \mathbf{D}_\perp^{m+1}\widehat{\mathbf{X}}_\perp^{(0)} - \mathbf{D}_\perp^m\widehat{\mathbf{X}}_\perp^{(0)}\mathbf{\Lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{D}_k^m(\mathbf{D}_k\widehat{\mathbf{X}}_k^{(0)} - \widehat{\mathbf{X}}_k^{(0)}\mathbf{\Lambda}) \\ \mathbf{D}_\perp^m(\mathbf{D}_\perp\widehat{\mathbf{X}}_\perp^{(0)} - \widehat{\mathbf{X}}_\perp^{(0)}\mathbf{\Lambda}) \end{pmatrix} \end{aligned}$$

Wir haben bereits gesehen, dass  $\widehat{\mathbf{X}}_k^{(0)}$  invertierbar ist, also können wir

$$\mathbf{\Lambda} := (\widehat{\mathbf{X}}_k^{(0)})^{-1}\mathbf{D}_k\widehat{\mathbf{X}}_k^{(0)}$$

setzen, um die erste Zeile zu eliminieren und

$$\mathbf{U}^*(\mathbf{A}\mathbf{X}^{(m)} - \mathbf{X}^{(m)}\mathbf{\Lambda}) = \begin{pmatrix} \mathbf{0} \\ \mathbf{D}_\perp^m(\mathbf{D}_\perp\widehat{\mathbf{X}}_\perp^{(0)} - \widehat{\mathbf{X}}_\perp^{(0)}\mathbf{\Lambda}) \end{pmatrix} \quad (4.15)$$

zu erhalten. Für jeden Vektor  $\mathbf{y} \in \mathbb{R}^k$  ergibt sich aus (4.14), (4.15) und (2.17)

$$\begin{aligned} \frac{\|(\mathbf{A}\mathbf{X}^{(m)} - \mathbf{X}^{(m)}\mathbf{\Lambda})\mathbf{y}\|_2}{\|\mathbf{\Pi}\mathbf{X}^{(m)}\mathbf{y}\|_2} &= \frac{\|\mathbf{U}^*(\mathbf{A}\mathbf{X}^{(m)} - \mathbf{X}^{(m)}\mathbf{\Lambda})\mathbf{y}\|_2}{\|\mathbf{U}^*\mathbf{\Pi}\mathbf{X}^{(m)}\mathbf{y}\|_2} = \frac{\|\mathbf{D}_\perp^m(\mathbf{D}_\perp\widehat{\mathbf{X}}_\perp^{(0)} - \widehat{\mathbf{X}}_\perp^{(0)}\mathbf{\Lambda})\mathbf{y}\|_2}{\|\mathbf{D}_k^m\widehat{\mathbf{X}}_k^{(0)}\mathbf{y}\|_2} \\ &\leq \frac{|\lambda_{k+1}|^m \|\mathbf{D}_\perp\widehat{\mathbf{X}}_\perp^{(0)} - \widehat{\mathbf{X}}_\perp^{(0)}\mathbf{\Lambda}\mathbf{y}\|_2}{|\lambda_k|^m \|\widehat{\mathbf{X}}_k^{(0)}\mathbf{y}\|_2} \\ &= \left(\frac{|\lambda_{k+1}|}{|\lambda_k|}\right)^m \frac{\|(\mathbf{A}\mathbf{X}^{(0)} - \mathbf{X}^{(0)}\mathbf{\Lambda})\mathbf{y}\|_2}{\|\mathbf{\Pi}\mathbf{X}^{(0)}\mathbf{y}\|_2}, \end{aligned}$$

also die gewünschte Abschätzung. ■

Natürlich sollten wir hier auch wieder darauf achten, dass die Vektoren nicht zu große oder zu kleine Koeffizienten enthalten. Dieses Ziel lässt sich wie bei der einfachen Vektoriteration durch eine Skalierung der Vektoren sicherstellen. Allerdings würde eine Skalierung alleine noch nicht ausreichen, um zu einem sinnvollen Ergebnis zu gelangen: Falls etwa  $|\lambda_1| > |\lambda_2|$  gilt, werden alle Spalten unseres bisherigen Verfahrens gegen den durch den ersten Eigenvektor aufgespannten Raum konvergieren und damit insbesondere ihre lineare Unabhängigkeit einbüßen. Deshalb müssen wir zusätzlich dafür sorgen, dass alle Vektoren linear unabhängig bleiben.

Diese Aufgabe können wir lösen, indem wir die in Abschnitt 2.6 diskutierte QR-Zerlegung etwas modifizieren. Wir haben dort gesehen, dass wir eine Matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  mit einer Reihe  $\mathbf{G}_1, \dots, \mathbf{G}_\ell$  von Givens-Rotationen in eine obere Dreiecksmatrix  $\mathbf{R}$  überführen können:

$$\mathbf{G}_\ell \dots \mathbf{G}_1 \mathbf{A} = \mathbf{R}.$$

In unserem Fall gilt  $n \geq m$ , so dass wir

$$\mathbf{R} = \begin{pmatrix} \widehat{\mathbf{R}} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \widehat{\mathbf{R}}, \quad \widehat{\mathbf{R}} \in \mathbb{R}^{m \times m}$$

#### 4 Eigenwertprobleme

erhalten und zu der Gleichung

$$\mathbf{A} = \mathbf{G}_1^* \dots \mathbf{G}_\ell^* \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \widehat{\mathbf{R}} = \mathbf{Q}\widehat{\mathbf{R}}, \quad \mathbf{Q} := \mathbf{G}_1^* \dots \mathbf{G}_\ell^* \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix}$$

gelangen. Der Unterschied zwischen der bereits bekannten QR-Zerlegung und dieser *dünnen* QR-Zerlegung besteht darin, dass  $\mathbf{Q} \in \mathbb{R}^{n \times m}$  eine rechteckige Matrix ist, während  $\widehat{\mathbf{R}} \in \mathbb{R}^{m \times m}$  quadratisch ist. Die wichtigen Eigenschaften

$$\mathbf{Q}^* \mathbf{Q} = \mathbf{I}, \quad \|\mathbf{Q}\mathbf{y}\|_2 = \|\mathbf{y}\|_2 \quad \text{für alle } \mathbf{y} \in \mathbb{R}^m$$

bleiben glücklicherweise erhalten, so dass auch die rechteckige Matrix  $\mathbf{Q}$  weiterhin als orthogonal bezeichnet werden kann.

Die Spalten einer orthogonalen Matrix bilden eine orthonormale Basis und können insbesondere nicht gegen denselben Raum konvergieren. Deshalb bietet es sich an, unseren Algorithmus so zu modifizieren, dass die Matrizen  $\mathbf{X}^{(m)}$  durch orthogonale Matrizen  $\mathbf{Q}^{(m)}$  ersetzt werden. Wir suchen also orthogonale Matrizen  $\mathbf{Q}^{(m)} \in \mathbb{R}^{n \times k}$  und Dreiecksmatrizen  $\mathbf{R}^{(m)} \in \mathbb{R}^{k \times k}$ , die

$$\mathbf{X}^{(m)} = \mathbf{Q}^{(m)} \mathbf{R}^{(m)} \quad \text{für alle } m \in \mathbb{N}_0 \quad (4.16)$$

erfüllen. Wir könnten diese QR-Zerlegungen direkt berechnen, müssten dann aber mit den instabilen Matrizen  $\mathbf{X}^{(m)}$  arbeiten. Stattdessen gehen wir wie bei der Vektoriteration vor und führen die Orthogonalisierung in jedem Schritt durch: Um von  $\mathbf{Q}^{(m)}$  zu  $\mathbf{Q}^{(m+1)}$  zu gelangen, berechnen wir

$$\mathbf{Y}^{(m+1)} := \mathbf{A}\mathbf{Q}^{(m)} \quad (4.17a)$$

und konstruieren anschließend eine dünne QR-Zerlegung

$$\mathbf{Y}^{(m+1)} = \mathbf{Q}^{(m+1)} \widehat{\mathbf{R}}^{(m+1)}. \quad (4.17b)$$

Mit (4.12) und (4.16) erhalten wir

$$\begin{aligned} \mathbf{X}^{(m+1)} &= \mathbf{A}\mathbf{X}^{(m)} = \mathbf{A}\mathbf{Q}^{(m)} \mathbf{R}^{(m)} = \mathbf{Y}^{(m+1)} \mathbf{R}^{(m)} = \mathbf{Q}^{(m+1)} \widehat{\mathbf{R}}^{(m+1)} \mathbf{R}^{(m)} \\ &= \mathbf{Q}^{(m+1)} \mathbf{R}^{(m+1)} \quad \text{mit} \quad \mathbf{R}^{(m+1)} := \widehat{\mathbf{R}}^{(m+1)} \mathbf{R}^{(m)}, \end{aligned}$$

wir können also den Schritt von  $\mathbf{Q}^{(m)}$  zu  $\mathbf{Q}^{(m+1)}$  vollziehen, ohne  $\mathbf{X}^{(m+1)}$  zu berechnen. Die durch (4.17) beschriebene Vorschrift zur Berechnung der Folge  $(\mathbf{Q}^{(m)})_{m=0}^\infty$  bezeichnen wir als die *orthogonale Iteration*.

Der Rayleigh-Quotient, den wir bisher für die Beurteilung der Genauigkeit der Approximation verwendet haben, wird in diesem Fall durch die *Rayleigh-Ritz-Matrix*

$$\mathbf{\Lambda}^{(m)} := (\mathbf{Q}^{(m)})^* \mathbf{A}\mathbf{Q}^{(m)} \in \mathbb{R}^{k \times k} \quad \text{für alle } m \in \mathbb{N}_0$$

ersetzt, und wir erhalten ein Abbruchkriterium der Form

$$\|\mathbf{A}\mathbf{Q}^{(m)} - \mathbf{Q}^{(m)} \mathbf{\Lambda}^{(m)}\|_2 \leq \epsilon.$$

Der resultierende Algorithmus hat die folgende Form:

```

procedure orthoit_adaptive( $\epsilon$ ,  $\mathbf{A}$ , var  $\mathbf{X}$ );
Berechne  $\mathbf{Q}\hat{\mathbf{R}} = \mathbf{X}$ ;
 $\mathbf{Y} \leftarrow \mathbf{A}\mathbf{Q}$ ;
 $\mathbf{\Lambda} \leftarrow \mathbf{Q}^*\mathbf{Y}$ ;
while  $\|\mathbf{Q}\mathbf{\Lambda} - \mathbf{Y}\|_2 > \epsilon$  do begin
  Berechne  $\mathbf{Q}\hat{\mathbf{R}} = \mathbf{Y}$ ;
   $\mathbf{Y} \leftarrow \mathbf{A}\mathbf{Q}$ ;
   $\mathbf{\Lambda} \leftarrow \mathbf{Q}^*\mathbf{Y}$ ;
end;
 $\mathbf{X} \leftarrow \mathbf{Q}$ 

```

Bei genauerer Betrachtung zeigt sich, dass es sich um eine relativ einfache Verallgemeinerung der Vektoriteration handelt: Statt mit einzelnen Vektoren wird mit  $k$ -spaltigen Matrizen gearbeitet, und statt die Iterierten zu Einheitsvektoren zu machen, werden Orthonormalbasen verwendet. Das Ergebnis ist eine Basis des von den Eigenvektoren zu den  $k$  betragsgrößten Eigenwerten aufgespannten Teilraums. Mehrfache oder eng beieinander liegende Eigenwerte führen bei dieser Technik nicht mehr zu Problemen, und der Rechenaufwand pro Schritt verhält sich wie  $nk^2$ , ist also für moderate Werte von  $k$  akzeptabel.

Für den Konvergenzbeweis benötigen wir die folgende nützliche Eigenschaft orthogonaler Matrizen:

**Lemma 4.10 (Orthogonale Projektion)** Sei  $\mathbf{Q} \in \mathbb{R}^{n \times m}$  orthogonal. Dann gelten

$$\|\mathbf{x} - \mathbf{Q}\mathbf{y}\|_2^2 = \|\mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}\|_2^2 + \|\mathbf{Q}^*\mathbf{x} - \mathbf{y}\|_2^2 \quad \text{für alle } \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m, \quad (4.18a)$$

$$\|\mathbf{Q}^*\mathbf{x}\|_2 \leq \|\mathbf{x}\|_2 \quad \text{für alle } \mathbf{x} \in \mathbb{R}^n. \quad (4.18b)$$

*Beweis.* Seien  $\mathbf{x} \in \mathbb{R}^n$  und  $\mathbf{y} \in \mathbb{R}^m$  gegeben. Aus der Beziehung zwischen euklidischer Norm und euklidischem Skalarprodukt sowie der ersten binomischen Gleichung folgt

$$\begin{aligned}
\|\mathbf{x} - \mathbf{Q}\mathbf{y}\|_2^2 &= \|\mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x} + \mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y})\|_2^2 \\
&= \langle \mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x} + \mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y}), \mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x} + \mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y}) \rangle_2 \\
&= \langle \mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}, \mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x} \rangle_2 + 2\langle \mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}, \mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y}) \rangle_2 \\
&\quad + \langle \mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y}), \mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y}) \rangle_2 \\
&= \|\mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}\|_2^2 + 2\langle \mathbf{Q}^*(\mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}), \mathbf{Q}^*\mathbf{x} - \mathbf{y} \rangle_2 \\
&\quad + \|\mathbf{Q}(\mathbf{Q}^*\mathbf{x} - \mathbf{y})\|_2^2 \\
&= \|\mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}\|_2^2 + 2\langle \mathbf{Q}^*\mathbf{x} - \mathbf{Q}^*\mathbf{x}, \mathbf{Q}^*\mathbf{x} - \mathbf{y} \rangle_2 + \|\mathbf{Q}^*\mathbf{x} - \mathbf{y}\|_2^2 \\
&= \|\mathbf{x} - \mathbf{Q}\mathbf{Q}^*\mathbf{x}\|_2^2 + \|\mathbf{Q}^*\mathbf{x} - \mathbf{y}\|_2^2.
\end{aligned}$$

Die Abschätzung (4.18b) folgt, indem wir  $\mathbf{y} = \mathbf{0}$  einsetzen. ■

Die Aussage (4.18a) zeigt, dass es zu einem gegebenen Vektor  $\mathbf{x}$  genau einen Vektor aus dem Bild der Matrix  $\mathbf{Q}$  gibt, der den geringsten Abstand zu ihm hat: Nur für  $\mathbf{y} = \mathbf{Q}^*\mathbf{x}$  nimmt die rechte Seite ihr Minimum an, also ist  $\mathbf{Q}\mathbf{Q}^*\mathbf{x}$  die bestmögliche Approximation des Vektors  $\mathbf{x}$  im Bild der Matrix  $\mathbf{Q}$ .

#### 4 Eigenwertprobleme

**Folgerung 4.11 (Konvergenz)** Sei  $\mathbf{\Pi X}^{(0)}$  injektiv. Dann existiert eine Konstante  $C \in \mathbb{R}_{\geq 0}$  mit

$$\|\mathbf{A Q}^{(m)} - \mathbf{Q}^{(m)} \mathbf{\Lambda}^{(m)}\|_2 \leq C \left( \frac{|\lambda_{k+1}|}{|\lambda_k|} \right)^m \quad \text{für alle } m \in \mathbb{N}_0.$$

*Beweis.* Sei  $m \in \mathbb{N}_0$ . Nach Satz 4.9 ist  $\mathbf{\Pi X}^{(m)}$  injektiv, also muss wegen (4.16) die Matrix  $\mathbf{R}^{(m)}$  invertierbar sein. Es folgt  $\mathbf{Q}^{(m)} = \mathbf{X}^{(m)} (\mathbf{R}^{(m)})^{-1}$ . Indem wir für einen beliebigen Vektor  $\mathbf{z} \in \mathbb{R}^k$  den Vektor  $\mathbf{y} := (\mathbf{R}^{(m)})^{-1} \mathbf{z}$  in die rechte Seite der Abschätzung (4.13) einsetzen, folgt

$$\begin{aligned} \frac{\|(\mathbf{A X}^{(m)} - \mathbf{X}^{(m)} \mathbf{\Lambda}) \mathbf{y}\|_2}{\|\mathbf{\Pi X}^{(m)} \mathbf{y}\|_2} &= \frac{\|(\mathbf{A X}^{(m)} - \mathbf{X}^{(m)} \mathbf{\Lambda}) (\mathbf{R}^{(m)})^{-1} \mathbf{z}\|_2}{\|\mathbf{\Pi X}^{(m)} (\mathbf{R}^{(m)})^{-1} \mathbf{z}\|_2} \\ &= \frac{\|(\mathbf{A Q}^{(m)} - \mathbf{Q}^{(m)} \mathbf{R}^{(m)} \mathbf{\Lambda} (\mathbf{R}^{(m)})^{-1}) \mathbf{z}\|_2}{\|\mathbf{\Pi Q}^{(m)} \mathbf{z}\|_2} \\ &= \frac{\|(\mathbf{A Q}^{(m)} - \mathbf{Q}^{(m)} \tilde{\mathbf{\Lambda}}^{(m)}) \mathbf{z}\|_2}{\|\mathbf{\Pi Q}^{(m)} \mathbf{z}\|_2} \end{aligned}$$

mit der Matrix

$$\tilde{\mathbf{\Lambda}}^{(m)} := \mathbf{R}^{(m)} \mathbf{\Lambda} (\mathbf{R}^{(m)})^{-1}.$$

Nach Definition gilt  $\mathbf{\Pi} = \mathbf{E}^{(k)} (\mathbf{E}^{(k)})^*$  mit der orthogonalen Matrix  $\mathbf{E}^{(k)}$ , also lässt sich (4.18b) anwenden, um für den Nenner

$$\|\mathbf{\Pi Q}^{(m)} \mathbf{z}\|_2 \leq \|\mathbf{Q}^{(m)} \mathbf{z}\|_2 = \|\mathbf{z}\|_2$$

zu folgern. Da  $\mathbf{Q}^{(m)}$  orthogonal ist, erhalten wir mit (4.18a) für den Zähler

$$\begin{aligned} \|\mathbf{A Q}^{(m)} \mathbf{z} - \mathbf{Q}^{(m)} \tilde{\mathbf{\Lambda}}^{(m)} \mathbf{z}\|_2 &\geq \|\mathbf{A Q}^{(m)} \mathbf{z} - \mathbf{Q}^{(m)} (\mathbf{Q}^{(m)})^* \mathbf{A Q}^{(m)} \mathbf{z}\|_2 \\ &= \|(\mathbf{A Q}^{(m)} - \mathbf{Q}^{(m)} \mathbf{\Lambda}^{(m)}) \mathbf{z}\|_2. \end{aligned}$$

Insgesamt haben wir also

$$\frac{\|(\mathbf{A Q}^{(m)} - \mathbf{Q}^{(m)} \mathbf{\Lambda}^{(m)}) \mathbf{z}\|_2}{\|\mathbf{z}\|_2} \leq \frac{\|(\mathbf{A Q}^{(m)} - \mathbf{Q}^{(m)} \tilde{\mathbf{\Lambda}}^{(m)}) \mathbf{z}\|_2}{\|\mathbf{\Pi Q}^{(m)} \mathbf{z}\|_2}$$

bewiesen. Indem wir diese Abschätzung mit (4.13) kombinieren und zu dem Maximum über alle  $\mathbf{y} \neq \mathbf{0}$  übergehen, erhalten wir die gewünschte Aussage mit der Konstanten

$$C := \sup \left\{ \frac{\|(\mathbf{A X}^{(0)} - \mathbf{X}^{(0)} \mathbf{\Lambda}) \mathbf{y}\|_2}{\|\mathbf{\Pi X}^{(0)} \mathbf{y}\|_2} : \mathbf{y} \in \mathbb{R}^k \setminus \{\mathbf{0}\} \right\}$$

und der Definition 2.11 der Spektralnorm. Da  $\mathbf{\Pi X}^{(0)}$  injektiv ist, ist  $C$  endlich. ■

## 4.5 QR-Iteration

Wir wenden uns nun der Aufgabe zu, *alle* Eigenwerte und Eigenvektoren einer Matrix zu berechnen. Dazu untersuchen wir die Matrizen  $\widehat{\mathbf{Q}}^{(m)}$  der orthogonalen Iteration etwas genauer. Die Matrizen der orthogonalen Iteration sind durch die Gleichung

$$\mathbf{Q}^{(m)} \widehat{\mathbf{R}}^{(m)} = \mathbf{A} \mathbf{Q}^{(m-1)} \quad \text{für alle } m \in \mathbb{N}$$

definiert. Wir wählen ein  $\ell \in \{1, \dots, k-1\}$  und untersuchen, welche Eigenschaften die ersten  $\ell$  Spalten dieser Matrizen aufweisen. Präziser formuliert zerlegen wir die Matrizen in der Form

$$\mathbf{Q}^{(m)} = \begin{pmatrix} \mathbf{Q}_\ell^{(m)} & \mathbf{Q}_*^{(m)} \end{pmatrix}, \quad \mathbf{Q}_\ell^{(m)} \in \mathbb{R}^{n \times \ell}, \quad \mathbf{Q}_*^{(m)} \in \mathbb{R}^{n \times (k-\ell)} \quad \text{für alle } m \in \mathbb{N}_0,$$

und interessieren uns für das Verhalten der Matrizen  $\mathbf{Q}_\ell^{(m)}$ . Da  $\widehat{\mathbf{R}}^{(m)}$  eine obere Dreiecksmatrix ist, gilt

$$\widehat{\mathbf{R}}^{(m)} = \begin{pmatrix} \widehat{\mathbf{R}}_{\ell\ell}^{(m)} & \widehat{\mathbf{R}}_{\ell*}^{(m)} \\ & \widehat{\mathbf{R}}_{**}^{(m)} \end{pmatrix}, \quad \widehat{\mathbf{R}}_{\ell\ell}^{(m)} \in \mathbb{R}^{\ell \times \ell}, \quad \widehat{\mathbf{R}}_{**}^{(m)} \in \mathbb{R}^{(k-\ell) \times (k-\ell)} \quad \text{für alle } m \in \mathbb{N}_0,$$

und wir erhalten

$$\begin{aligned} \begin{pmatrix} \mathbf{A} \mathbf{Q}_\ell^{(m-1)} & \mathbf{A} \mathbf{Q}_*^{(m-1)} \end{pmatrix} &= \mathbf{A} \begin{pmatrix} \mathbf{Q}_\ell^{(m-1)} & \mathbf{Q}_*^{(m-1)} \end{pmatrix} = \mathbf{A} \mathbf{Q}^{(m-1)} = \mathbf{Q}^{(m)} \widehat{\mathbf{R}}^{(m)} \\ &= \begin{pmatrix} \mathbf{Q}_\ell^{(m)} & \mathbf{Q}_*^{(m)} \end{pmatrix} \begin{pmatrix} \widehat{\mathbf{R}}_{\ell\ell}^{(m)} & \widehat{\mathbf{R}}_{\ell*}^{(m)} \\ & \widehat{\mathbf{R}}_{**}^{(m)} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{Q}_\ell^{(m)} \widehat{\mathbf{R}}_{\ell\ell}^{(m)} & \mathbf{Q}_\ell^{(m)} \widehat{\mathbf{R}}_{\ell*}^{(m)} + \mathbf{Q}_*^{(m)} \widehat{\mathbf{R}}_{**}^{(m)} \end{pmatrix}. \end{aligned}$$

Der linken Spalte dieser Gleichung entnehmen wir

$$\mathbf{A} \mathbf{Q}_\ell^{(m-1)} = \mathbf{Q}_\ell^{(m)} \widehat{\mathbf{R}}_{\ell\ell}^{(m)} \quad \text{für alle } m \in \mathbb{N},$$

die ersten  $\ell$  Spalten der Matrizen  $\widehat{\mathbf{Q}}^{(m)}$  werden also so berechnet, als würden wir eine orthogonale Iteration mit nur diesen Spalten durchführen. Insbesondere können wir die Konvergenzaussage der Folgerung 4.11 anwenden, falls

$$|\lambda_1| \geq \dots \geq |\lambda_\ell| > |\lambda_{\ell+1}| \geq \dots \geq |\lambda_n|$$

gilt, um zu folgern, dass

$$\|\mathbf{A} \mathbf{Q}_\ell^{(m)} - \mathbf{Q}_\ell^{(m)} \boldsymbol{\Lambda}_\ell^{(m)}\|_2 \leq C \left( \frac{|\lambda_{\ell+1}|}{|\lambda_\ell|} \right)^m \quad \text{für alle } m \in \mathbb{N}$$

mit einer geeigneten Konstanten  $C \in \mathbb{R}_{\geq 0}$  und  $\boldsymbol{\Lambda}_\ell^{(m)} := (\mathbf{Q}_\ell^{(m)})^* \mathbf{A} \mathbf{Q}_\ell^{(m)}$  gilt.

Mit Hilfe dieser Eigenschaft lässt sich eine nützliche Aussage über die Konvergenz gewinnen, indem wir ausnutzen, dass die ersten  $\ell$  Spalten der Matrix  $\mathbf{Q}^{(m)}$ , die in der

#### 4 Eigenwertprobleme

Matrix  $\mathbf{Q}_\ell^{(m)}$  zusammengefasst sind, und die letzten  $k - \ell$  Spalten, die wir in der Matrix  $\mathbf{Q}_*^{(m)}$  untergebracht haben, senkrecht aufeinander stehen. In Formeln ausgedrückt gilt

$$\begin{aligned} \begin{pmatrix} (\mathbf{Q}_\ell^{(m)})^* \mathbf{Q}_\ell^{(m)} & (\mathbf{Q}_\ell^{(m)})^* \mathbf{Q}_*^{(m)} \\ (\mathbf{Q}_*^{(m)})^* \mathbf{Q}_\ell^{(m)} & (\mathbf{Q}_*^{(m)})^* \mathbf{Q}_*^{(m)} \end{pmatrix} &= \begin{pmatrix} \mathbf{Q}_\ell^{(m)} & \mathbf{Q}_*^{(m)} \end{pmatrix}^* \begin{pmatrix} \mathbf{Q}_\ell^{(m)} & \mathbf{Q}_*^{(m)} \end{pmatrix} \\ &= (\mathbf{Q}^{(m)})^* \mathbf{Q}^{(m)} = \mathbf{I} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \end{aligned}$$

und komponentenweise gelesen erhalten wir

$$(\mathbf{Q}_\ell^{(m)})^* \mathbf{Q}_\ell^{(m)} = \mathbf{I}, \quad (\mathbf{Q}_*^{(m)})^* \mathbf{Q}_*^{(m)} = \mathbf{I}, \quad (\mathbf{Q}_*^{(m)})^* \mathbf{Q}_\ell^{(m)} = \mathbf{0}. \quad (4.19)$$

Also sind  $\mathbf{Q}_\ell^{(m)}$  und  $\mathbf{Q}_*^{(m)}$  orthogonale Matrizen, deren Bilder senkrecht aufeinander stehen.

Indem wir (4.19) und (4.18b) mit der Folgerung 4.11 kombinieren, erhalten wir die Abschätzung

$$\begin{aligned} \|(\mathbf{Q}_*^{(m)})^* \mathbf{A} \mathbf{Q}_\ell^{(m)}\|_2 &= \|(\mathbf{Q}_*^{(m)})^* \mathbf{A} \mathbf{Q}_\ell^{(m)} - (\mathbf{Q}_*^{(m)})^* \mathbf{Q}_\ell^{(m)} \mathbf{\Lambda}_\ell^{(m)}\|_2 \\ &= \|(\mathbf{Q}_*^{(m)})^* (\mathbf{A} \mathbf{Q}_\ell^{(m)} - \mathbf{Q}_\ell^{(m)} \mathbf{\Lambda}_\ell^{(m)})\|_2 \\ &\leq \|\mathbf{A} \mathbf{Q}_\ell^{(m)} - \mathbf{Q}_\ell^{(m)} \mathbf{\Lambda}_\ell^{(m)}\|_2 \\ &\leq C \left( \frac{|\lambda_{\ell+1}|}{|\lambda_\ell|} \right)^m \quad \text{für alle } m \in \mathbb{N}_0. \end{aligned}$$

Unter diesen Bedingungen muss also der linke untere Block der Matrix

$$\mathbf{A}^{(m)} := (\mathbf{Q}^{(m)})^* \mathbf{A} \mathbf{Q}^{(m)} = \begin{pmatrix} (\mathbf{Q}_\ell^{(m)})^* \mathbf{A} \mathbf{Q}_\ell^{(m)} & (\mathbf{Q}_\ell^{(m)})^* \mathbf{A} \mathbf{Q}_*^{(m)} \\ (\mathbf{Q}_*^{(m)})^* \mathbf{A} \mathbf{Q}_\ell^{(m)} & (\mathbf{Q}_*^{(m)})^* \mathbf{A} \mathbf{Q}_*^{(m)} \end{pmatrix}$$

gegen null konvergieren. Falls sogar

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_k| \quad (4.20)$$

gelten sollte, können wir unsere Argumentation auf jedes beliebige  $\ell \in \{1, \dots, k-1\}$  anwenden und folgern, dass die Matrix sogar gegen eine obere Dreiecksmatrix konvergieren wird, an der wir Eigenwerte und Eigenvektoren relativ einfach ablesen können.

Unsere Aufgabe, alle Eigenwerte und Eigenvektoren der Matrix  $\mathbf{A}$  zu berechnen, lässt sich aufgrund dieser Beobachtung relativ einfach lösen: Wir verwenden die orthogonale Iteration mit  $k = n$  Vektoren, beispielsweise mit  $\mathbf{Q}^{(0)} = \mathbf{I}$ . Bei dieser Wahl ist sicher gestellt, dass sämtliche Eigenräume im Bild der Matrix  $\mathbf{Q}^{(0)}$  vorkommen. Sofern die Voraussetzung (4.20) erfüllt ist, werden dann die Matrizen  $\mathbf{A}^{(m)}$  gegen obere Dreiecksform konvergieren, so dass wir ihre Eigenwerte und Eigenvektoren einfach bestimmen können. Da  $\mathbf{Q}^{(m)}$  in diesem Fall wieder eine quadratische orthogonale Matrix ist, muss

$$\mathbf{Q}^{(m)} \mathbf{A}^{(m)} (\mathbf{Q}^{(m)})^* = \mathbf{A} \quad (4.21)$$



gelten, so dass wir die Eigenwerte und Eigenvektoren der ursprünglichen Matrix  $\mathbf{A}$  aus denen der Matrix  $\mathbf{A}^{(m)}$  rekonstruieren können.

Für den Algorithmus wäre es sehr hilfreich, wenn uns die Matrix  $\mathbf{A}^{(m)}$  zur Verfügung stehen würde, denn wie bereits gesehen lässt sich an ihr einfach ablesen, wie gut die durch die Matrix  $\mathbf{Q}^{(m)}$  beschriebene Approximation der Eigenvektoren bereits konvergiert ist. Besonders günstig wäre es, wenn wir  $\mathbf{A}^{(m+1)}$  direkt aus  $\mathbf{A}^{(m)}$  berechnen könnten, denn dann ließe sich Speicherplatz sparen.

Um dieses Ziel zu erreichen führen wir orthogonale Matrizen  $\widehat{\mathbf{Q}}^{(m)} \in \mathbb{R}^{n \times n}$  ein, die den Wechsel zwischen den einzelnen orthonormalen Basen gemäß

$$\mathbf{Q}^{(m)} = \mathbf{Q}^{(m-1)} \widehat{\mathbf{Q}}^{(m)} \quad \text{für alle } m \in \mathbb{N}$$

beschreiben. Mit dieser Wahl erhalten wir

$$\begin{aligned} \mathbf{A}^{(m)} &= (\mathbf{Q}^{(m)})^* \mathbf{A} \mathbf{Q}^{(m)} = (\widehat{\mathbf{Q}}^{(m)})^* (\mathbf{Q}^{(m-1)})^* \mathbf{A} \mathbf{Q}^{(m-1)} \widehat{\mathbf{Q}}^{(m)} \\ &= (\widehat{\mathbf{Q}}^{(m)})^* \mathbf{A}^{(m-1)} \widehat{\mathbf{Q}}^{(m)} \quad \text{für alle } m \in \mathbb{N}. \end{aligned}$$

Natürlich müssen wir auch dazu in der Lage sein, die Matrizen  $\widehat{\mathbf{Q}}^{(m)}$  aus den Matrizen  $\mathbf{A}^{(m-1)}$  direkt zu bestimmen. Nach Definition gilt

$$\begin{aligned} \mathbf{Q}^{(m)} \widehat{\mathbf{R}}^{(m)} &= \mathbf{A} \mathbf{Q}^{(m-1)}, \\ \mathbf{Q}^{(m-1)} \widehat{\mathbf{Q}}^{(m)} \widehat{\mathbf{R}}^{(m)} &= \mathbf{A} \mathbf{Q}^{(m-1)}, \\ \widehat{\mathbf{Q}}^{(m)} \widehat{\mathbf{R}}^{(m)} &= (\mathbf{Q}^{(m-1)})^* \mathbf{A} \mathbf{Q}^{(m-1)}, \\ \widehat{\mathbf{Q}}^{(m)} \widehat{\mathbf{R}}^{(m)} &= \mathbf{A}^{(m-1)} \quad \text{für alle } m \in \mathbb{N}, \end{aligned}$$

also lässt sich  $\widehat{\mathbf{Q}}^{(m)}$  bestimmen, indem wir eine gewöhnliche QR-Zerlegung der Matrix  $\mathbf{A}^{(m-1)}$  berechnen. Die nächste Iterierte ergibt sich in diesem Fall aus

$$\begin{aligned} \mathbf{A}^{(m)} &= (\widehat{\mathbf{Q}}^{(m)})^* \mathbf{A}^{(m-1)} \widehat{\mathbf{Q}}^{(m)} = (\widehat{\mathbf{Q}}^{(m)})^* \widehat{\mathbf{Q}}^{(m)} \widehat{\mathbf{R}}^{(m)} \widehat{\mathbf{Q}}^{(m)} \\ &= \widehat{\mathbf{R}}^{(m)} \widehat{\mathbf{Q}}^{(m)} \quad \text{für alle } m \in \mathbb{N}, \end{aligned}$$

wir müssen also lediglich die beiden Faktoren der QR-Zerlegung in vertauschter Reihenfolge wieder multiplizieren. Damit haben wir eine einfache Variante der *QR-Iteration* erhalten:

```
procedure qrit_simple(var  $\mathbf{A}$ ,  $\widehat{\mathbf{Q}}$ );
while  $\mathbf{A}$  nicht obere Dreiecksmatrix do begin
  Berechne  $\widehat{\mathbf{Q}} \widehat{\mathbf{R}} = \mathbf{A}$ ;
   $\mathbf{A} \leftarrow \widehat{\mathbf{R}} \widehat{\mathbf{Q}}$ ;    $\mathbf{Q} \leftarrow \mathbf{Q} \widehat{\mathbf{Q}}$ 
end
```

In dieser einfachen Form weist der Algorithmus eine Reihe von Nachteilen auf, die ihn für den praktischen Einsatz ungeeignet erscheinen lassen. Es ist allerdings möglich, ihn durch geeignete Modifikationen erheblich zu verbessern.

## 4.6 Praktische QR-Iteration\*

Die uns bisher vorliegende Variante der QR-Iteration hat mehrere praktische Nachteile:

- Die Durchführung eines einzelnen Schritts erfordert die Berechnung einer QR-Zerlegung, also im allgemeinen Fall einen zu  $n^3$  proportionalen Aufwand.
- Es wird nicht ausgenutzt, dass eventuell Teilmatrizen bereits konvergiert sein könnten, obwohl die Gesamtmatrix noch nicht die gewünschte Dreiecksform erreicht hat.
- Die Konvergenz ist bei nahe beieinander liegenden Eigenwerten relativ langsam.

Praktische Umsetzungen der QR-Iteration verwenden einfache Modifikationen des grundlegenden Verfahrens, um diese Nachteile zu beseitigen oder wenigstens zu mindern.

**Bemerkung 4.12 (Hessenberg-Form)** *Eine Matrix  $\mathbf{H}$  ist in Hessenberg-Form, wenn alle Einträge unterhalb der unteren Nebendiagonalen gleich null sind:*

$$\mathbf{H} = \begin{pmatrix} \times & \dots & \dots & \dots & \times \\ \times & \times & \dots & \dots & \times \\ & \times & \times & \dots & \times \\ & & \ddots & \ddots & \vdots \\ & & & \times & \times \end{pmatrix}.$$

Für eine derartige Matrix lässt sich die QR-Zerlegung in lediglich  $n - 1$  Givens-Rotationen berechnen, da nur  $n - 1$  Elemente unterhalb der Diagonalen eliminiert werden müssen. Ein Schritt der QR-Iteration benötigt dann nur  $\sim n^2$  Operationen, und die neue Matrix wird wieder in Hessenberg-Form sein.

Mit einer Reihe von Givens-Rotationen lässt sich jede beliebige Ausgangsmatrix  $\mathbf{A}$  in eine Hessenberg-Matrix  $\mathbf{H}$  überführen, so dass die folgenden Iterationsschritte sehr effizient durchgeführt werden können. Zu Illustration untersuchen wir eine Matrix der Form

$$\mathbf{A} = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}.$$

In einem ersten Schritt wenden wir eine Givens-Rotation  $\mathbf{G}_1$  auf die zweite und vierte Zeile an, um  $a_{41}$  zu eliminieren:

$$\mathbf{G}_1^* \mathbf{A} = \begin{pmatrix} \times & \times & \times & \times \\ \otimes & \otimes & \otimes & \otimes \\ \times & \times & \times & \times \\ 0 & \otimes & \otimes & \otimes \end{pmatrix}.$$

Um die Form (4.21) zu erhalten, müssen wir von rechts mit der adjungierten Matrix multiplizieren, also die Givens-Rotation auf die zweite und vierte Spalte anwenden, so dass sich

$$\mathbf{B}^{(1)} := \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 = \begin{pmatrix} \times & \otimes & \times & \otimes \\ \times & \otimes & \times & \otimes \\ \times & \otimes & \times & \otimes \\ 0 & \otimes & \times & \otimes \end{pmatrix}$$

ergibt. Erfreulicherweise bleibt die Null in der ersten Spalte erhalten, da diese Spalte nicht verändert wird. Wir verfahren entsprechend mit der zweiten und dritten Spalte, um  $b_{31}^{(1)}$  mit einer Givens-Rotation  $\mathbf{G}_2$  zu beseitigen:

$$\mathbf{G}_2^* \mathbf{B}^{(1)} = \begin{pmatrix} \times & \times & \times & \times \\ \otimes & \otimes & \otimes & \otimes \\ 0 & \otimes & \otimes & \otimes \\ 0 & \times & \times & \times \end{pmatrix}, \quad \mathbf{B}^{(2)} := \mathbf{G}_2^* \mathbf{B}^{(1)} \mathbf{G}_2 = \begin{pmatrix} \times & \otimes & \otimes & \times \\ \times & \otimes & \otimes & \times \\ 0 & \otimes & \otimes & \times \\ 0 & \otimes & \otimes & \times \end{pmatrix}.$$

Schließlich benutzen wir eine Givens-Rotation  $\mathbf{G}_3$ , um durch eine Kombination der dritten und vierten Spalte auch den letzten störenden Eintrag  $b_{42}^{(2)}$  zu einer Null zu machen und erreichen so die gewünschte Hessenberg-Form:

$$\mathbf{G}_3^* \mathbf{B}^{(2)} = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \otimes & \otimes & \otimes \\ 0 & 0 & \otimes & \otimes \end{pmatrix}, \quad \mathbf{H} := \mathbf{G}_3^* \mathbf{B}^{(2)} \mathbf{G}_3 = \begin{pmatrix} \times & \times & \otimes & \otimes \\ \times & \times & \otimes & \otimes \\ 0 & \times & \otimes & \otimes \\ 0 & 0 & \otimes & \otimes \end{pmatrix}.$$

Insgesamt haben wir eine Hessenberg-Matrix mit

$$\mathbf{H} = \mathbf{G}_3^* \mathbf{G}_2^* \mathbf{G}_1^* \mathbf{A} \mathbf{G}_1 \mathbf{G}_2 \mathbf{G}_3 = \mathbf{P}^* \mathbf{A} \mathbf{P}, \quad \mathbf{P} := \mathbf{G}_1 \mathbf{G}_2 \mathbf{G}_3$$

erhalten. Die nützliche Transformation auf Hessenberg-Form können wir elegant in unsere bisherige QR-Iteration integrieren, indem wir  $\mathbf{Q}^{(0)} := \mathbf{P}$  als Ausgangspunkt verwenden und so  $\mathbf{A}^{(0)} = \mathbf{H}$  erhalten.

**Bemerkung 4.13 (Tridiagonalmatrix)** Falls  $\mathbf{A}$  selbstadjungiert ist, überträgt sich diese Eigenschaft auf  $\mathbf{H}$ , so dass wir sogar die Tridiagonalfom

$$\mathbf{H} = \begin{pmatrix} \times & \times & & \\ \times & \times & \times & \\ & \times & \times & \times \\ & & \times & \times \end{pmatrix}$$

erhalten, für die sich ein QR-Schritt sogar mit  $\sim n$  Operationen durchführen lässt.

**Bemerkung 4.14 (Deflation)** Falls wir die QR-Iteration mit Hessenberg-Matrizen durchführen, lässt sich einfach anhand der unteren Nebendiagonalelemente  $a_{i+1,i}^{(m)}$  ablesen, welche Teilmatrizen bereits konvergiert sind. Diese Matrizen brauchen wir nicht weiter zu bearbeiten. Beispielsweise können wir im  $m$ -ten Schritt

$$\alpha := \min\{i \in \{1, \dots, n-1\} : |a_{i+1,i}^{(m)}| > \epsilon(|a_{ii}^{(m)}| + |a_{i+1,i+1}^{(m)}|)\},$$

#### 4 Eigenwertprobleme

$$\beta := \min\{i \in \{\alpha + 1, \dots, n - 1\} : |a_{i+1,i}^{(m)}| \leq \epsilon(|a_{ii}^{(m)}| + |a_{i+1,i+1}^{(m)}|)\}$$

setzen und lediglich die Zeilen und Spalten zwischen  $\alpha$  und  $\beta$  bearbeiten, um Rechenoperationen zu sparen.

**Bemerkung 4.15 (Shift-Strategie)** *Wir haben bereits bei der Rayleigh-Iteration gesehen, dass sich durch eine geschickte Wahl eines Shift-Parameters die Konvergenz eines Eigenwertverfahrens erheblich verbessern lässt. Die QR-Iteration lässt sich leicht entsprechend modifizieren: Wir wählen im  $m$ -ten Schritt ein  $\mu^{(m)}$  und führen den QR-Schritt in der Form*

$$\begin{aligned}\widehat{\mathbf{Q}}^{(m)}\widehat{\mathbf{R}}^{(m)} &= \mathbf{A}^{(m-1)} - \mu^{(m)}\mathbf{I}, \\ \mathbf{A}^{(m)} &= \widehat{\mathbf{R}}^{(m)}\widehat{\mathbf{Q}}^{(m)} + \mu^{(m)}\mathbf{I}\end{aligned}$$

durch. Eine besonders einfache Shift-Strategie ist die Wahl des rechten unteren Diagonalelements als  $\mu^{(m)}$ , das gerade dem Rayleigh-Quotienten für die letzte Spalte der Matrix  $\mathbf{Q}^{(m)}$  entspricht.

# 5 Approximation von Funktionen

## 5.1 Beispiel: Computergrafik

Techniken, um aus einer mathematischen Beschreibung eines Objekts ein Bild des Objekts zu erzeugen, finden vielfältige Anwendungen, seit Computer leistungsfähig genug geworden sind, um sie in vertretbarer Zeit auszuführen: Maschinen werden per *Computer Aided Design (CAD)* im Rechner interaktiv entworfen, bevor sie in der Realität hergestellt werden. Computersimulationen ahmen das Verhalten der Natur nach und ermöglichen es, kostspielige Experimente durch wesentlich günstiger Berechnungen zu ersetzen. Und nicht zuletzt sorgen Computerspiele und Filme mit dreidimensionaler Grafik für Unterhaltung.

Während auch die moderne Computergrafik im Wesentlichen darauf basiert, schnell sehr viele geeignet eingefärbte Dreiecke darstellen zu können, wird für die Beschreibung der durch diese Dreiecke dargestellten Geometrie in der Regel mit Kurven und gebogenen Flächen gearbeitet, die wesentlich weniger Speicherplatz erfordern und bei Bedarf in eine beliebig gute Approximation durch Dreiecke oder ebene Geradenstücke umgerechnet werden können.

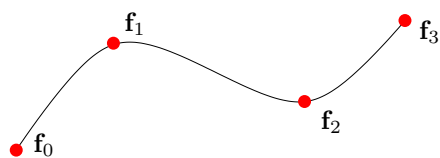


Abbildung 5.1: Kubische Interpolation

Ein Beispiel sind *Spline-Kurven*, bei denen kubische Polynome eingesetzt werden, um eine Kurve im  $d$ -dimensionalen Raum durch *Kontrollpunkte*  $f_0, \dots, f_m \in \mathbb{R}^d$  zu beschreiben, durch die die Kurve verlaufen muss.

Mit Hilfe geeigneter Algorithmen lassen sich solche Kurven auf dem Bildschirm darstellen und den Vorstellungen eines Ingenieurs oder Künstlers anpassen, mit einem Drucker ausgeben oder als Bestandteil einer Computeranimation einsetzen. Die Technik lässt sich einfach erweitern, um gebogene Flächen darzustellen, und aus mehreren solchen Flächen lassen sich komplexere Objekte zusammensetzen.

## 5.2 Polynominterpolation

Wir sind daran interessiert, einen möglicherweise komplizierten Kurvenverlauf durch einige Punkte zu beschreiben, die auf der Kurve liegen. Unsere Aufgabe besteht also darin, aus diesen Punkten die Kurve zu rekonstruieren.

Wir beschränken uns hier auf die *Polynominterpolation*: Wir bezeichnen den Raum der Polynome höchstens  $m$ -ten Grades mit

$$\Pi_m := \text{span}\{1, x, x^2, \dots, x^m\}$$

und suchen nach einer Lösung der folgenden *Interpolationsaufgabe*:

Für gegebene Werte  $f_0, \dots, f_m \in \mathbb{R}$  und paarweise verschiedene *Stützstellen*  $x_0, \dots, x_m \in \mathbb{R}$  finde ein Polynom  $p \in \Pi_m$ , das

$$p(x_i) = f_i \quad \text{für alle } i \in \{0, \dots, m\} \quad (5.1)$$

erfüllt.

Bevor wir uns Gedanken über die konkrete Berechnung des Polynoms  $p$  machen, sollten wir zunächst klären, ob die Interpolationsaufgabe überhaupt lösbar ist.

Einen besonders eleganten Zugang bieten die *Lagrange-Polynome*: Für jedes  $i \in \{0, \dots, m\}$  ist die Funktion

$$\ell_i : \mathbb{R} \rightarrow \mathbb{R}, \quad x \mapsto \prod_{\substack{k=0 \\ k \neq i}}^m \frac{x - x_k}{x_i - x_k}, \quad (5.2)$$

ein Polynom höchstens  $m$ -ten Grades, es gilt also  $\ell_i \in \Pi_m$ , da  $\ell_i$  das Produkt von  $m$  linearen Polynomen ist.

Lagrange-Polynome haben eine entscheidende Eigenschaft:

**Lemma 5.1 (Lagrange-Eigenschaft)** Falls  $x_0, \dots, x_m$  paarweise verschieden sind, gilt

$$\ell_i(x_j) = \begin{cases} 1 & \text{falls } i = j, \\ 0 & \text{ansonsten} \end{cases} \quad \text{für alle } i, j \in \{0, \dots, m\}. \quad (5.3)$$

*Beweis.* Seien  $i, j \in \{0, \dots, m\}$  gewählt. Falls  $i = j$  gilt, erhalten wir

$$\ell_i(x_i) = \prod_{\substack{k=0 \\ k \neq i}}^m \frac{x_i - x_k}{x_i - x_k} = 1,$$

also das gewünschte Ergebnis.

Falls  $i \neq j$  gilt, muss in dem Produkt ein Faktor mit  $k = j$  vorkommen, und in diesem Fall wird wegen  $x_j - x_k = 0$  auch das gesamte Produkt gleich null sein. ■

**Lemma 5.2 (Lösbarkeit)** Für beliebige  $f_0, \dots, f_m \in \mathbb{R}$  löst das Polynom

$$p = \sum_{k=0}^m f_k \ell_k \quad (5.4)$$

das Interpolationsproblem (5.1). Es ist die einzige Lösung dieses Problems.

*Beweis.* Sei  $i \in \{0, \dots, m\}$ . Dank (5.3) erhalten wir

$$p(x_i) = \sum_{k=0}^m f_k \ell_k(x_i) = f_i \ell_i(x_i) = f_i,$$

da alle Summanden mit Ausnahme des  $i$ -ten gleich null sind.

Um zu zeigen, dass  $p$  die einzige Lösung des Interpolationsproblems ist, untersuchen wir die lineare Abbildung

$$\Phi : \Pi_m \rightarrow \mathbb{R}^{m+1}, \quad q \mapsto \begin{pmatrix} q(x_0) \\ \vdots \\ q(x_m) \end{pmatrix}.$$

Wir haben bereits gezeigt, dass wir zu beliebigen Werten  $f_0, \dots, f_m \in \mathbb{R}$  ein Polynom mit (5.1) finden können, also muss  $\Phi$  surjektiv sein.

Da  $\Pi_m$  nach Definition höchstens  $(m+1)$ -dimensional sein kann, muss die lineare Abbildung  $\Phi$  infolge des Dimensionssatzes der linearen Algebra auch injektiv sein.

Falls nun  $q \in \Pi_m$  ein weiteres Polynom ist, dass

$$q(x_i) = f_i \quad \text{für alle } i \in \{0, \dots, m\}$$

erfüllt, muss für  $r := p - q$  gerade

$$r(x_i) = p(x_i) - q(x_i) = f_i - f_i = 0 \quad \text{für alle } i \in \{0, \dots, m\}$$

gelten, also  $\Phi(r) = \mathbf{0}$ , und da  $\Phi$  injektiv ist, folgt daraus  $r = 0$ , also  $p = q$ . ■

## 5.3 Auswertung per Neville-Aitken-Verfahren

Im Prinzip könnten wir die Gleichung (5.4) verwenden, um das Interpolationspolynom  $p$  an beliebigen Stellen  $x \in \mathbb{R}$  auszuwerten. Allerdings ist dieser Ansatz nicht allzu effizient, so dass es sich empfiehlt, nach Alternativen zu suchen.

Wir stellen fest, dass wir den Fall  $m = 0$  besonders einfach behandeln können: Ein Polynom  $p \in \Pi_0$  ist konstant, und  $p = f_0$  ist die Lösung der Interpolationsaufgabe. Die Idee des *Neville-Aitken-Verfahrens* besteht darin, von konstanten Polynomen ausgehend Polynome höheren Grades zu konstruieren, bis das gesuchte Polynom  $m$ -ten Grades gefunden ist.

## 5 Approximation von Funktionen

Für alle  $i, j \in \{0, \dots, m\}$  mit  $i \leq j$  existiert nach Lemma 5.2 genau ein Polynom  $p_{i,j} \in \Pi_{j-i}$ , das

$$p_{i,j}(x_k) = f_k \quad \text{für alle } k \in \{i, \dots, j\} \quad (5.5)$$

erfüllt. Diese Polynome stehen zueinander in einer Beziehung, die sich für praktische Berechnungen ausnutzen lässt:

**Lemma 5.3 (Aitken-Rekurrenz)** *Seien  $i, j \in \{0, \dots, m\}$  mit  $i < j$  gegeben. Dann gilt*

$$p_{i,j}(x) = \frac{x - x_i}{x_j - x_i} p_{i+1,j}(x) + \frac{x_j - x}{x_j - x_i} p_{i,j-1}(x) \quad \text{für alle } x \in \mathbb{R}. \quad (5.6)$$

*Beweis.* Wir führen den Beweis per Induktion über  $j - i \in \mathbb{N}$ .

Seien  $i, j \in \{0, \dots, m\}$  mit  $j - i = 1$  gegeben. Dann folgen  $p_{i+1,j} = p_{j,j} = f_j$  und  $p_{i,j-1} = p_{i,i} = f_i$ , so dass wir

$$p_{i,j}(x_i) = \frac{x_j - x_i}{x_j - x_i} p_{i+1,j}(x_i) = f_i, \quad p_{i,j}(x_j) = \frac{x_j - x_i}{x_j - x_i} p_{i+1,j}(x_j) = f_j$$

erhalten und (5.6) bewiesen ist.

Sei nun  $n \in \mathbb{N}$  so gewählt, dass (5.6) für  $j - i \leq n$  gilt. Wir wählen  $i, j \in \{0, \dots, m\}$  mit  $j - i = n + 1$ . Sei  $k \in \{i, \dots, j\}$ . Wir unterscheiden drei verschiedene Fälle.

Falls  $k = i$  gilt, erhalten wir

$$p_{i,j}(x_k) = p_{i,j}(x_i) = f_i = \frac{x_i - x_i}{x_j - x_i} p_{i+1,j}(x_i) + \frac{x_j - x_i}{x_j - x_i} p_{i,j-1}(x_i).$$

Falls  $k = j$  gilt, ergibt sich

$$p_{i,j}(x_k) = p_{i,j}(x_j) = f_j = \frac{x_j - x_i}{x_j - x_i} p_{i+1,j}(x_j) + \frac{x_j - x_j}{x_j - x_i} p_{i,j-1}(x_j),$$

und für  $i < k < j$  erhalten wir mit Hilfe der Induktionsvoraussetzung  $p_{i+1,j}(x_k) = f_k$  und  $p_{i,j-1}(x_k) = f_k$ , so dass

$$p_{i,j}(x_k) = f_k = \frac{x_k - x_i + x_j - x_k}{x_j - x_i} f_k = \frac{x_k - x_i}{x_j - x_i} p_{i+1,j}(x_k) + \frac{x_j - x_k}{x_j - x_i} p_{i,j-1}(x_k),$$

folgt und (5.6) auch für  $j - i = n + 1$  bewiesen ist. ■

Mit Hilfe der Aitken-Rekurrenz können wir wie angedeutet vorgehen: Um das Interpolationspolynom  $p$  in einem Punkt  $x \in \mathbb{R}$  auszuwerten, bestimmen wir zunächst die konstanten Polynome  $p_{i,i} = f_i$ . Dann verwenden wir (5.6), um lineare Polynome zu konstruieren, anschließend quadratische, und in dieser Weise können wir fortfahren, bis



wir den Polynomgrad  $m$  erreicht haben und damit  $p_{0,m}(x) = p(x)$  berechnet ist. Anschaulich lassen sich die Abhängigkeiten zwischen den einzelnen Werten, und damit die Rechenvorschrift, in Form des folgenden Dreiecksschemas darstellen:

$$\begin{array}{ccccccc}
 f_0 = p_{0,0}(x) & & & & & & \\
 & \searrow & & & & & \\
 f_1 = p_{1,1}(x) & \rightarrow & p_{0,1}(x) & & & & \\
 & \searrow & & \searrow & & & \\
 f_2 = p_{2,2}(x) & \rightarrow & p_{1,2}(x) & \rightarrow & p_{0,2}(x) & & \\
 & \searrow & & \searrow & & \searrow & \\
 f_3 = p_{3,3}(x) & \rightarrow & p_{2,3}(x) & \rightarrow & p_{1,3}(x) & \rightarrow & p_{0,3}(x) = p(x)
 \end{array}$$

Unser Ziel ist es nun, diesen Algorithmus mit möglichem geringem Speicherbedarf umzusetzen. Wir stellen fest, dass der Wert  $p_{3,3}(x)$  nur für die Berechnung von  $p_{2,3}(x)$  benötigt wird, wir können also den Speicherplatz, den dieser Wert belegt hat, nun mit dem Wert  $p_{2,3}(x)$  überschreiben. Der Wert  $p_{2,2}(x)$  wird nicht mehr benötigt, sobald  $p_{2,3}(x)$  und  $p_{1,2}(x)$  berechnet wurden, also können wir den Speicherplatz für  $p_{1,2}(x)$  wiederverwenden. Entsprechend können wir fortfahren und spaltenweise von links nach rechts und innerhalb der Spalten von unten nach oben die Werte überschreiben, bis  $p(x)$  berechnet wurde:

```

function neville( $x$ , var  $f$ );
for  $n = 1, \dots, m$  do
  for  $j = m, \dots, n$  do begin
     $i \leftarrow j - n$ ;
     $f_j \leftarrow ((x - x_i)f_j + (x_j - x)f_{j-1}) / (x_j - x_i)$ 
  end
return  $f_m$ 

```

Die Variable  $n$  gibt in diesem Algorithmus gerade die Differenz  $j - i = n$  an, also den Grad der jeweils auszuwertenden Polynome. Die innere Schleife überschreibt  $f_j$  mit dem Wert  $p_{i,j}(x)$ , indem die im vorangehenden Schritt in  $f_j$  und  $f_{j-1}$  gespeicherten Werte von  $p_{i+1,j}(x)$  und  $p_{i,j-1}(x)$  kombiniert werden. In der inneren Schleife fallen gerade 7 Rechenoperationen an, insgesamt benötigt der Algorithmus

$$\sum_{n=1}^m 7(m-n+1) = \sum_{n=1}^m 7n = \frac{7}{2}m(m+1)$$

Rechenoperationen, die Anzahl wächst also quadratisch mit dem Grad des zu berechnenden Polynoms.

## 5.4 Auswertung mit Newtons dividierten Differenzen

Ein quadratisch mit  $m$  wachsender Rechenaufwand ist akzeptabel, wenn wir das Interpolationspolynom  $p$  nur in wenigen Punkten auswerten wollen. Wenn wir dagegen

## 5 Approximation von Funktionen

beispielsweise den Verlauf einer Kurve auf dem Bildschirm darstellen wollen, müssen wir in der Regel das zugehörige Polynom sehr häufig auswerten, also sind wir daran interessiert, den Rechenaufwand pro Auswertung so weit wie möglich zu reduzieren. Das ist möglich, falls wir geeignete Hilfsgrößen im Voraus berechnen, ähnlich der Vorgehensweise bei der Behandlung linearer Gleichungssysteme, bei der wir mit einer LR- oder QR-Zerlegung das Lösen des Systems erheblich beschleunigen konnten.

Ein sinnvoller Zugang besteht darin, eine alternative Darstellung des Polynoms  $p$  zu finden, die sich effizient auswerten lässt. Die *Newton-Darstellung* beruht auf der Beobachtung, dass für  $i, j \in \{0, \dots, m\}$  mit  $i < j$  die Gleichung

$$p_{i,j}(x) = p_{i,j-1}(x) + d_{i,j}(x - x_i) \dots (x - x_{j-1}) \quad \text{für alle } x \in \mathbb{R} \quad (5.7)$$

gelten muss, wenn wir die Konstanten  $d_{i,j}$  durch

$$d_{i,j} := \begin{cases} f_i & \text{falls } i = j, \\ \frac{f_j - p_{i,j-1}(x_j)}{(x_j - x_i) \dots (x_j - x_{j-1})} & \text{falls } i < j \end{cases} \quad \text{für alle } i, j \in \{0, \dots, m\} \text{ mit } i \leq j$$

definieren und so dafür sorgen, dass die rechte Seite der Gleichung in dem Punkt  $x_j$  den richtigen Wert annimmt. Dass sie auch in den Punkten  $x_i, \dots, x_{j-1}$  die richtigen Werte annimmt, folgt daraus, dass  $p_{i,j-1}$  es tut und der rechte Summand in diesen Punkten gleich null ist.

Indem wir die *Newton-Polynome*

$$n_{i,j} : \mathbb{R} \rightarrow \mathbb{R}, \quad x \mapsto \begin{cases} 1 & \text{falls } i = j, \\ \prod_{k=i}^{j-1} (x - x_k) & \text{ansonsten} \end{cases} \quad \text{für alle } i, j \in \{0, \dots, m\} \text{ mit } i \leq j$$

eingeführen, können wir die Gleichung in der Form

$$p_{i,j}(x) = p_{i,j-1}(x) + n_{i,j}(x)d_{i,j}, \quad \text{für alle } x \in \mathbb{R}$$

darstellen, und mit einer einfachen Induktion über  $j - i \in \mathbb{N}_0$  erhalten wir

$$p_{i,j}(x) = \sum_{k=i}^j d_{i,k} n_{i,k}(x) \quad \text{für alle } x \in \mathbb{R}. \quad (5.8)$$

Unser Ziel ist es nun, die Auswertung dieses Polynoms besonders effizient zu gestalten. Dazu nutzen wir aus, dass

$$n_{i,j}(x) = (x - x_i)n_{i+1,j}(x) \quad \text{für alle } x \in \mathbb{R}, \quad i, j \in \{0, \dots, m\}, \quad i < j$$

gilt, und erhalten aus (5.8) mit  $i = 0$  und  $j = m$  die Gleichung

$$\begin{aligned} p(x) &= p_{0,m}(x) \\ &= d_{0,0}n_{0,0}(x) + \sum_{k=1}^m d_{0,k}n_{0,k}(x) \end{aligned}$$

$$= d_{0,0} + (x - x_0) \sum_{k=1}^m d_{0,k} n_{1,k}(x) \quad \text{für alle } x \in \mathbb{R}.$$

Die rechte Summe ist ein Polynom der Ordnung  $m - 1$ , so dass sich uns wieder die Möglichkeit eröffnet, Polynome höherer Ordnung aus solchen niedrigerer Ordnung zusammen zu setzen: Wir definieren

$$s_i = \sum_{k=i}^m d_{0,k} n_{i,k} \quad \text{für alle } i \in \{0, \dots, m\}$$

und stellen fest, dass  $s_i \in \Pi_{m-i}$  sowie  $s_0 = p$  gelten. Wie zuvor können wir diese Polynome mit Hilfe der Formel

$$\begin{aligned} s_i(x) &= \sum_{k=i}^m d_{0,k} n_{i,k}(x) = d_{0,i} + (x - x_i) \sum_{k=i+1}^m d_{0,k} n_{i+1,k}(x) \\ &= d_{0,i} + (x - x_i) s_{i+1}(x) \quad \text{für alle } i \in \{0, \dots, m-1\}, x \in \mathbb{R} \end{aligned}$$

bestimmen, die sich ideal als Grundlage eines effizienten Algorithmus eignet: Wir bestimmen zuerst  $s_m(x) = d_{0,m}$ , berechnen dann mit Hilfe der Formel  $s_{m-1}(x)$ , und fahren so fort, bis  $s_0(x)$  bekannt ist. Mit dem Vektor  $\mathbf{d} = (d_{0,0}, \dots, d_{0,m})$  lässt sich der Algorithmus wie folgt zusammenfassen:

```

function eval_newton( $x$ ,  $\mathbf{d}$ );
 $s \leftarrow d_m$ ;
for  $i = m - 1$  downto 0 do
     $s \leftarrow d_i + (x - x_i)s$ ;
return  $s$ 

```

Die Variable  $s$  enthält dabei zunächst den Wert  $s_m(x)$ , dann den Wert  $s_{m-1}(x)$ , und so weiter, bis  $s_0(x) = p(x)$  zurückgegeben werden kann. Wir können leicht erkennen, dass der Algorithmus lediglich  $3m$  Rechenoperationen benötigt, also *wesentlich* schneller als die Methode von Neville und Aitken ist.

Es bleibt noch die Frage, wie sich die Größen  $d_{0,i}$  effizient berechnen lassen, die wir für diesen Algorithmus benötigen. Dazu führen wir die folgende Sprechweise ein: Für jedes Polynom  $q \in \Pi_m$  existiert genau ein Satz von Koeffizienten  $a_0, \dots, a_m \in \mathbb{R}$ , für den

$$q(x) = \sum_{i=0}^m a_i x^i \quad \text{für alle } x \in \mathbb{R}$$

gilt. Dementsprechend bezeichnen wir für jedes  $i \in \{0, \dots, m\}$  die Zahl  $a_i$  als den  *$i$ -ten Koeffizienten* des Polynoms  $q$ .

Aus der Gleichung (5.7) folgt, dass der  $(j - i)$ -te Koeffizient des Polynoms  $p_{i,j}$  gerade  $d_{i,j}$  ist. Für  $i, j \in \{0, \dots, m\}$  mit  $i < j$  existieren Koeffizienten  $a_0, \dots, a_m \in \mathbb{R}$ ,  $b_0, \dots, b_{m-1} \in \mathbb{R}$  und  $c_0, \dots, c_{m-1} \in \mathbb{R}$ , die

$$p_{i,j}(x) = \sum_{k=0}^{j-i} a_k x^k, \quad p_{i+1,j}(x) = \sum_{k=0}^{j-i-1} b_k x^k, \quad p_{i,j-1}(x) = \sum_{k=0}^{j-i-1} c_k x^k \quad \text{für alle } x \in \mathbb{R}$$

## 5 Approximation von Funktionen

erfüllen. Durch Einsetzen in die Formel (5.6) der Aitken-Rekurrenz erhalten wir

$$\begin{aligned}
 \sum_{k=0}^{j-i} a_k x^k &= p_{i,j}(x) = \frac{x-x_i}{x_j-x_i} p_{i+1,j}(x) + \frac{x_j-x}{x_j-x_i} p_{i,j-1}(x) \\
 &= \frac{x}{x_j-x_i} \sum_{k=0}^{j-i-1} b_k x^k - \frac{x_i}{x_j-x_i} \sum_{k=0}^{j-i-1} b_k x^k \\
 &\quad + \frac{x_j}{x_j-x_i} \sum_{k=0}^{j-i-1} c_k x^k - \frac{x}{x_j-x_i} \sum_{k=0}^{j-i-1} c_k x^k \\
 &= \sum_{k=0}^{j-i-1} \frac{b_k - c_k}{x_j-x_i} x^{k+1} + \sum_{k=0}^{j-i-1} \frac{x_j c_k - x_i b_k}{x_j-x_i} x^k \\
 &= \sum_{k=1}^{j-i} \frac{b_{k-1} - c_{k-1}}{x_j-x_i} x^k + \sum_{k=0}^{j-i-1} \frac{x_j c_k - x_i b_k}{x_j-x_i} x^k \quad \text{für alle } x \in \mathbb{R}
 \end{aligned}$$

und folgern per Koeffizientenvergleich, dass

$$a_{j-i} = \frac{b_{j-i-1} - c_{j-i-1}}{x_j - x_i}$$

gelten muss. Mit  $a_{j-i} = d_{i,j}$ ,  $b_{j-i-1} = d_{i+1,j}$  und  $c_{j-i-1} = d_{i,j-1}$  erhalten wir

$$d_{i,j} = \frac{d_{i+1,j} - d_{i,j-1}}{x_j - x_i} \quad \text{für alle } i, j \in \{0, \dots, m\}, i < j. \quad (5.9)$$

Dieser Gleichung verdanken die Koeffizienten  $d_{i,j}$  den Namen *Newtons dividierte Differenzen*. Wir können bei der Berechnung wie im Fall des Neville-Verfahrens vorgehen und ein Dreiecksschema verwenden:

$$\begin{array}{ccccccc}
 f_0 = d_{0,0} & & & & & & \\
 & \searrow & & & & & \\
 f_1 = d_{1,1} & \rightarrow & d_{0,1} & & & & \\
 & \searrow & & \searrow & & & \\
 f_2 = d_{2,2} & \rightarrow & d_{1,2} & \rightarrow & d_{0,2} & & \\
 & \searrow & & \searrow & & \searrow & \\
 f_3 = d_{3,3} & \rightarrow & d_{2,3} & \rightarrow & d_{1,3} & \rightarrow & d_{0,3}
 \end{array}$$

Bei der praktischen Umsetzung können wir wieder durch geschicktes Überschreiben von Variablen dafür sorgen, dass nach dem  $n$ -ten Schritt die Werte  $d_{j-n,j}$  in den Einträgen  $f_j$  für  $j \in \{m-n, \dots, m\}$  vorliegen:

```

procedure newton_differenzen( $x$ , var  $f$ );
for  $n = 1$  to  $m$  do
  for  $j = m$  downto  $n$  do begin
     $i \leftarrow j - n$ ;
     $f_j \leftarrow (f_j - f_{j-1}) / (x_j - x_i)$ 
  end

```

Dieser Algorithmus überschreibt  $f_0, \dots, f_m$  mit  $d_{0,0}, \dots, d_{0,m}$  und benötigt dafür

$$\sum_{n=1}^m 3(m-n+1) = \sum_{n=1}^m 3n = \frac{3}{2}m(m+1)$$

arithmetische Operationen, so dass sein Aufwand wieder quadratisch mit dem Grad  $m$  anwächst. Allerdings muss dieser Algorithmus nur *einmal* durchgeführt werden, um die zu  $p$  gehörenden Koeffizienten zu berechnen, die Auswertung des Polynoms kann dann in der bereits beschriebenen Weise mit  $3m$  Operationen erfolgen.

## 5.5 Approximation von Funktionen

Interpolation wird in der Mathematik häufig eingesetzt, um Funktionen zu approximieren. Dabei ist man in der Regel nicht nur daran interessiert, das Polynom  $p$  zu finden, sondern auch daran, abzuschätzen, wie gut es die ursprüngliche Funktion approximiert.

Um diese Frage zu untersuchen wählen wir ein nicht-leeres Intervall  $[a, b]$  sowie paarweise verschiedene Interpolationspunkte  $x_0, \dots, x_m \in [a, b]$  und untersuchen die folgende Variante der Interpolationsaufgabe:

Sei  $f \in C[a, b]$  gegeben. Gesucht ist ein  $p \in \Pi_m$ , das

$$p(x_i) = f(x_i) \quad \text{für alle } i \in \{0, \dots, m\} \quad (5.10)$$

erfüllt.

Da diese Aufgabe ein Spezialfall der Aufgabe (5.1) ist, übertragen sich alle Aussagen über Existenz und Eindeutigkeit der Lösung, und auch die praktische Berechnung des Polynoms  $p$  lässt sich mit den bisher besprochenen Verfahren effizient durchführen.

Mit Hilfe einer Variante des Mittelwertsatzes können wir eine erste Abschätzung für den Approximationsfehler gewinnen:

**Lemma 5.4 (Interpolationsfehler)** *Sei  $f \in C^{m+1}[a, b]$ , sei  $p$  die Lösung der Aufgabe (5.10), und sei  $x \in [a, b]$ . Dann existiert ein  $\eta \in [a, b]$  mit*

$$f(x) - p(x) = (x - x_0) \dots (x - x_m) \frac{f^{(m+1)}(\eta)}{(m+1)!}. \quad (5.11)$$

*Beweis.* (siehe [3]) Falls ein  $i \in \{0, \dots, m\}$  mit  $x = x_i$  existieren sollte, folgt aus der definierenden Bedingung (5.10) bereits  $f(x_i) - p(x_i) = 0$ , also ist die linke Seite der Gleichung (5.11) gleich null. Da auf dem Produkt auf der rechten Seite auch ein Term  $x - x_i$  auftritt, ist auch die rechte Seite gleich null, also gilt insbesondere die Gleichheit.

Sei nun  $x \in [a, b] \setminus \{x_0, \dots, x_m\}$ . Wir untersuchen die Hilfsfunktion

$$g : [a, b] \rightarrow \mathbb{R}, \quad y \mapsto f(y) - p(y) - (y - x_0) \dots (y - x_m)R,$$

bei der wir  $R \in \mathbb{R}$  so wählen, dass  $g(x) = 0$  gilt. Das ist möglich, da nach Voraussetzung  $(x - x_0) \dots (x - x_m) \neq 0$  gilt.

## 5 Approximation von Funktionen

Wir stellen fest, dass auch

$$g(x_i) = f(x_i) - p(x_i) - (x_i - x_0) \dots (x_i - x_m) R = 0 \quad \text{für alle } i \in \{0, \dots, m\}$$

gilt, also besitzt  $g$  mindestens  $m + 2$  paarweise verschiedene Nullstellen. Nach dem Satz von Rolle muss dann  $g'$  noch mindestens  $m + 1$  paarweise verschiedene Nullstellen besitzen, und mit einer einfachen Induktion folgt, dass die  $(m + 1)$ -te Ableitung  $g^{(m+1)}$  immer noch mindestens eine Nullstelle  $\eta \in [a, b]$  besitzen muss. Wir erhalten

$$0 = g^{(m+1)}(\eta) = f^{(m+1)}(\eta) - p^{(m+1)}(\eta) - (m + 1)! R,$$

und da  $p \in \Pi_m$  nach Voraussetzung gilt, haben wir  $p^{(m+1)}(\eta) = 0$  und somit

$$R = \frac{f^{(m+1)}(\eta)}{(m + 1)!}.$$

Da  $x$  nach Konstruktion eine Nullstelle der Funktion  $g$  ist, folgt

$$0 = g(x) = f(x) - p(x) - (x - x_0) \dots (x - x_m) \frac{f^{(m+1)}(\eta)}{(m + 1)!},$$

und das ist die gesuchte Gleichung für den Fehler. ■

Diese Darstellung des Interpolationsfehlers hat den Vorteil, eine *Gleichung* zu bieten statt einer Abschätzung, aber sie hat auch den Nachteil, dass die bloße Existenz des entscheidenden Zwischenpunkts  $\eta$  bewiesen wird, aber keine Aussage über seine konkrete Position oder seine Abhängigkeit von  $f$  und  $x$ .

Deshalb geht man in der Praxis zu Abschätzungen für den maximalen Fehler über: Wir definieren für Funktionen  $g \in C[a, b]$  die *Maximumnorm* durch

$$\|g\|_{\infty, [a, b]} := \max\{|g(x)| : x \in [a, b]\}$$

und schätzen in (5.11) die rechte Seite durch diese Norm ab. Damit erhalten wir die von  $x$  unabhängige Fehlerschranke

$$\|f - p\|_{\infty, [a, b]} \leq \|\omega\|_{\infty, [a, b]} \frac{\|f^{(m+1)}\|_{\infty, [a, b]}}{(m + 1)!}, \quad (5.12)$$

bei der das *Stützstellenpolynom*  $\omega \in \Pi_{m+1}$  durch

$$\omega(x) := (x - x_0) \dots (x - x_m) \quad \text{für alle } x \in \mathbb{R}$$

definiert ist. Zwar haben wir so die Fehler*gleichung* durch eine Fehler*schranke* ersetzt, aber diese Schranke ist so formuliert, dass der Einfluss des Verfahrens von dem Einfluss der interpolierten Funktion getrennt ist: Das Stützstellenpolynom hängt nur von der Wahl der Interpolationspunkte ab, während der zweite Faktor nur von dem Verhalten der Funktion  $f$  abhängt. Damit ist es beispielsweise möglich, nach besonders guten Interpolationspunkten zu suchen.

**Beispiel 5.5 (Tschebyscheff-Interpolation)** Für jedes Stützstellenpolynom gilt

$$\|\omega\|_{\infty,[a,b]} \geq 2 \left( \frac{b-a}{4} \right)^{m+1}.$$

Falls wir die Tschebyscheff-Interpolationspunkte

$$\hat{x}_i := \frac{b+a}{2} + \frac{b-a}{2} \cos \left( \pi \frac{2i+1}{2m+2} \right) \quad \text{für alle } i \in \{0, \dots, m\}$$

für  $[a, b]$  verwenden, erhalten wir

$$\|\hat{\omega}\|_{\infty,[a,b]} = 2 \left( \frac{b-a}{4} \right)^{m+1},$$

in diesem Sinn sind diese Interpolationspunkte also die bestmögliche Wahl.

Es ist auch möglich, eine Fehlerabschätzung anzugeben, die es nicht erfordert, zusätzliche Differenzierbarkeitsanforderungen an die Funktion  $f$  zu stellen:

**Lemma 5.6 (Bestapproximation)** Die Lebesgue-Konstante der Interpolation ist gegeben durch

$$\Lambda_m := \max \left\{ \sum_{k=0}^m |\ell_k(x)| : x \in [a, b] \right\}.$$

Für das durch (5.10) definierte Interpolationspolynom  $p \in \Pi_m$  gilt

$$\|f - p\|_{\infty,[a,b]} \leq (\Lambda_m + 1) \|f - q\|_{\infty,[a,b]} \quad \text{für alle } q \in \Pi_m,$$

der Interpolationsfehler lässt sich also durch den Fehler der bestmöglichen polynomiellen Approximation abschätzen.

*Beweis.* Wir definieren den Interpolationsoperator

$$\mathfrak{I}_m : C[a, b] \rightarrow \Pi_m, \quad g \mapsto \sum_{k=0}^m g(x_k) \ell_k,$$

der gemäß (5.4) jeder Funktion  $g \in C[a, b]$  das Polynom  $\mathfrak{I}_m[g] \in \Pi_m$  zuordnet, das diese Funktion in den Punkten  $x_0, \dots, x_m$  interpoliert.

Für jede Funktion  $g \in C[a, b]$  gilt

$$\begin{aligned} \|\mathfrak{I}_m[g]\|_{\infty,[a,b]} &= \max \{ |\mathfrak{I}_m[g](x)| : x \in [a, b] \} \\ &= \max \left\{ \left| \sum_{k=0}^m g(x_k) \ell_k(x) \right| : x \in [a, b] \right\} \\ &\leq \max \left\{ \sum_{k=0}^m |g(x_k)| |\ell_k(x)| : x \in [a, b] \right\} \end{aligned}$$

## 5 Approximation von Funktionen

$$\leq \|g\|_{\infty,[a,b]} \max \left\{ \sum_{k=0}^m |\ell_k(x)| : x \in [a,b] \right\} = \Lambda_m \|g\|_{\infty,[a,b]},$$

wir können also das Maximum des Interpolationspolynoms durch das Maximum der interpolierten Funktion abschätzen.

Sei nun  $q \in \Pi_m$ . Da  $q^{(m+1)} = 0$  gilt, folgt aus (5.11) insbesondere  $q = \mathfrak{I}_m[q]$ , und wir erhalten

$$\begin{aligned} \|f - \mathfrak{I}_m[f]\|_{\infty,[a,b]} &= \|f - q + \mathfrak{I}_m[q] - \mathfrak{I}_m[f]\|_{\infty,[a,b]} \\ &\leq \|f - q\|_{\infty,[a,b]} + \|\mathfrak{I}_m[q] - \mathfrak{I}_m[f]\|_{\infty,[a,b]} \\ &= \|f - q\|_{\infty,[a,b]} + \|\mathfrak{I}_m[f - q]\|_{\infty,[a,b]} \\ &\leq \|f - q\|_{\infty,[a,b]} + \Lambda_m \|f - q\|_{\infty,[a,b]} = (1 + \Lambda_m) \|f - q\|_{\infty,[a,b]}, \end{aligned}$$

und die gewünschte Aussage ist bewiesen. ■

**Bemerkung 5.7 (Tschebyscheff)** Falls wir Tschebyscheff-Interpolationspunkte verwenden, erfüllt die Lebesgue-Konstante die Abschätzung

$$\Lambda_m \leq \frac{2}{\pi} \log(m+1) + 1 \leq m+1$$

(siehe [2]), wächst also nur sehr langsam mit dem Polynomgrad  $m$ . Man kann zeigen, dass sich durch eine andere Wahl der Interpolationspunkte lediglich die Konstante vor dem logarithmischen Faktor etwas reduzieren lässt.

**Bemerkung 5.8 (Steuerung des Fehlers)** In der Regel sind wir daran interessiert, den Interpolationsfehler nicht zu groß werden zu lassen. Wie wir an (5.12) sehen, gibt es im Wesentlichen zwei Möglichkeiten, um sicher zu stellen, dass der Fehler unter Kontrolle bleibt:

Falls  $f \in C^\infty[a,b]$  gilt und die Ableitungen der Funktion nicht zu schnell wachsen, dürfen wir darauf hoffen, dass der Fehler sinkt, wenn wir  $m$  erhöhen. Verfahren, die auf diesem Weg den Fehler steuern nennt man Spektralverfahren oder  $p$ -Methoden.

Falls  $f$  dagegen nicht beliebig oft differenzierbar ist oder die Ableitungen schnell wachsen, können wir den Fehler mit Hilfe des Faktors  $\|\omega\|_{\infty,[a,b]}$  zu reduzieren versuchen. Gemäß Beispiel 5.5 lässt er sich nicht beliebig weit reduzieren, aber da

$$\begin{aligned} |\omega(x)| &= |x - x_0| |x - x_1| \dots |x - x_m| \\ &\leq (b-a)(b-a) \dots (b-a) = (b-a)^{m+1} \quad \text{für alle } x \in [a,b] \end{aligned}$$

gilt, können wir durch Wahl eines hinreichend kleinen Intervalls  $[a,b]$  auch für beliebig gewählte Interpolationspunkte einen beliebig geringen Fehler erreichen. Verfahren, die durch die Verkleinerung der Länge des Interpolationsintervalls den Fehler reduzieren, bezeichnet man als  $h$ -Methoden.



# 6 Numerische Integration

## 6.1 Beispiel: Wahrscheinlichkeiten

Die Lösung mancher mathematischer Probleme erfordert es, Integrale von Funktionen zu berechnen, deren Stammfunktion nicht explizit ausgewertet werden kann.

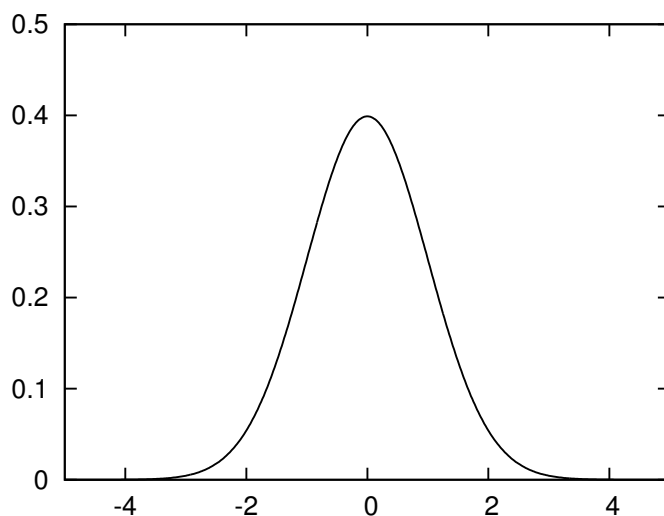


Abbildung 6.1: Dichtefunktion der Normalverteilung

Ein Beispiel aus der Stochastik ist die Berechnung der Wahrscheinlichkeit eines normalverteilten Ereignisses: Das Integral

$$P(a \leq x \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-x^2/2} dx$$

gibt die Wahrscheinlichkeit an, mit der eine normalverteilte Zufallsvariable  $x$  einen Wert zwischen  $a$  und  $b$  annimmt. Da keine Methode bekannt ist, mit der sich die Stammfunktion von  $e^{-x^2}$  explizit berechnen lässt, wird dieses Integral in der Regel numerisch approximiert.

Ähnliche Aufgaben treten auch bei anderen Wahrscheinlichkeitsverteilungen auf, beispielsweise bei der Analyse stochastischer Prozesse, mit denen das Verhalten von Aktienkursen modelliert wird, allerdings ist in diesem Fall der Integrand häufig komplizierter und das Integrationsgebiet nicht nur eindimensional.

## 6.2 Quadraturformeln

Bei der numerischen Integration sind wir daran interessiert, die folgende Aufgabe möglichst effizient zu lösen:

Gegeben sind ein nicht-leeres Intervall  $[a, b]$  und eine stetige Funktion  $f \in C[a, b]$ , zu approximieren ist das Integral

$$\int_a^b f(x) dx.$$

Um die Notation etwas abzukürzen, schreiben wir das Integral als Abbildung, die jeder Funktion  $f \in C[a, b]$  ihr Integral zuordnet:

$$\mathcal{I}_{[a,b]} : C[a, b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{I}_{[a,b]}(f) := \int_a^b f(x) dx.$$

Da das Integral in der Regel als Verallgemeinerung der Summe definiert wird, bietet es sich an, auch bei der numerischen Approximation von einer Summe auszugehen.

**Definition 6.1 (Quadraturformel)** Seien  $m \in \mathbb{N}_0$ , Punkte  $x_0, \dots, x_m \in [a, b]$  und  $w_0, \dots, w_m \in \mathbb{R}$  gegeben. Dann definiert

$$\mathcal{Q}_{[a,b]} : C[a, b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{Q}_{[a,b]}(f) := \sum_{i=0}^m w_i f(x_i),$$

eine Quadraturformel, die jeder Funktion  $f \in C[a, b]$  eine Approximation des Integrals durch Funktionswerte in den Quadraturpunkten  $x_0, \dots, x_m$  und mit den Quadraturgewichten  $w_0, \dots, w_m$  zuordnet.

Wir sind daran interessiert, Quadraturformeln zu finden, die mit möglichst wenigen Funktionsauswertungen eine möglichst hohe Genauigkeit erreichen.

**Beispiel 6.2 (Mittelpunktregel)** Eine besonders einfache Quadraturformel ist die Mittelpunktregel, die für das Intervall  $[-1, 1]$  durch

$$\mathcal{M}_{[-1,1]} : C[-1, 1] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{M}_{[-1,1]}(f) := 2f(0),$$

definiert ist. Um den Approximationsfehler abschätzen zu können, setzen wir voraus, dass  $f \in C^2[-1, 1]$  gilt. Dann folgt mit partieller Integration

$$\begin{aligned} \mathcal{I}_{[-1,1]}(f) - \mathcal{M}_{[-1,1]}(f) &= \int_{-1}^1 f(x) dx - 2f(0) \\ &= \int_{-1}^0 f(x) dx - f(0) + \int_0^1 f(x) dx - f(0) \\ &= - \int_{-1}^0 (x+1)f'(x) dx - \int_0^1 (x-1)f'(x) dx \end{aligned}$$

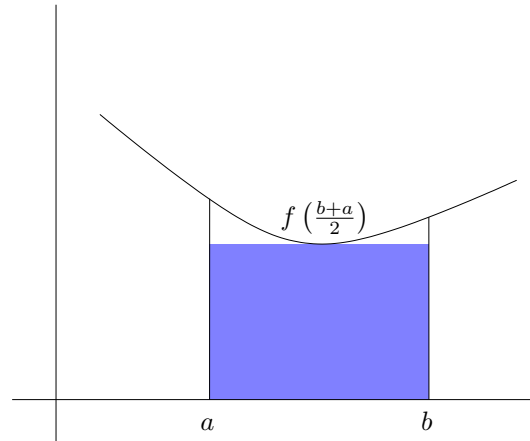


Abbildung 6.2: Idee der Mittelpunkregel: Die Fläche unter der Kurve wird durch ein Rechteck angenähert.

$$\begin{aligned}
 &= \int_{-1}^0 \frac{(x+1)^2}{2} f''(x) dx - \frac{f'(0)}{2} + \int_0^1 \frac{(x-1)^2}{2} f''(x) dx + \frac{f'(0)}{2} \\
 &= \int_{-1}^0 \frac{(x+1)^2}{2} f''(x) dx + \int_0^1 \frac{(x-1)^2}{2} f''(x) dx,
 \end{aligned}$$

und mit dem Mittelwertsatz der Integralrechnung müssen  $\eta_- \in [-1, 0]$  und  $\eta_+ \in [0, 1]$  mit

$$\begin{aligned}
 \int_{-1}^0 \frac{(x+1)^2}{2} f''(x) dx &= f''(\eta_-) \int_{-1}^0 \frac{(x+1)^2}{2} dx = \frac{f''(\eta_-)}{6}, \\
 \int_0^1 \frac{(x-1)^2}{2} f''(x) dx &= f''(\eta_+) \int_0^1 \frac{(x-1)^2}{2} dx = \frac{f''(\eta_+)}{6}
 \end{aligned}$$

existieren, so dass wir mit dem Zwischenwertsatz für stetige Funktionen ein  $\eta \in [\eta_-, \eta_+]$  finden, das

$$\mathcal{I}_{[-1,1]}(f) - \mathcal{M}_{[-1,1]}(f) = \frac{f''(\eta)}{3}$$

erfüllt. Für ein allgemeines Intervall ist die Mittelpunkregel durch

$$\mathcal{M}_{[a,b]} : C[a,b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{M}_{[a,b]}(f) := (b-a) f\left(\frac{b+a}{2}\right),$$

gegeben, und für jedes  $f \in C^2[a,b]$  existiert ein  $\eta \in [a,b]$  mit

$$\mathcal{I}_{[a,b]}(f) - \mathcal{M}_{[a,b]}(f) = \frac{(b-a)^3}{24} f''(\eta). \quad (6.1)$$

Man sieht, dass die Mittelpunkregel durchaus eine gute Näherung des Integrals sein kann, falls  $f''$  und das Intervall  $[a,b]$  nicht zu groß sind.

## 6 Numerische Integration

Wenn wir die Mittelpunkregel in der Form

$$\mathcal{M}_{[a,b]}(f) = (b-a)f\left(\frac{b+a}{2}\right) = \int_a^b f\left(\frac{b+a}{2}\right) dx \quad \text{für alle } f \in C[a,b]$$

schreiben, können wir ein allgemeines Konstruktionsprinzip ablesen: Wir haben die Funktion  $f$  durch eine konstante Funktion approximiert, nämlich durch das interpolierende Polynom im Punkt  $(b+a)/2$ , und dann diese Funktion integriert, um eine Approximation des Integrals zu erhalten.

Diesen Zugang können wir verallgemeinern: Wir gehen von paarweise verschiedenen Interpolationspunkten  $x_0, \dots, x_m \in [a, b]$  aus. Für ein  $f \in C[a, b]$  bezeichnen wir mit  $p \in \Pi_m$  das Polynom, das  $f$  in diesen Punkten interpoliert (vgl. (5.10)) und integrieren es. Dank (5.4) erhalten wir

$$\int_a^b f(x) dx \approx \int_a^b p(x) dx = \int_a^b \sum_{i=0}^m f(x_i) \ell_i(x) dx = \sum_{i=0}^m f(x_i) \int_a^b \ell_i(x) dx,$$

und dieser Ausdruck entpuppt sich bei näherer Betrachtung als Quadraturformel.

**Definition 6.3 (Interpolatorische Quadratur)** Seien  $x_0, \dots, x_m \in [a, b]$  paarweise verschiedene Punkte, und seien  $\ell_0, \dots, \ell_m \in \Pi_m$  die korrespondierenden Lagrange-Polynome (vgl. (5.2)). Mit

$$w_i := \int_a^b \ell_i(x) dx \quad \text{für alle } i \in \{0, \dots, m\}$$

erhalten wir die zu den Punkten  $x_0, \dots, x_m$  gehörende interpolatorische Quadraturformel

$$\mathcal{Q}_{[a,b]} : C[a, b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{Q}_{[a,b]}(f) := \sum_{i=0}^m w_i f(x_i).$$

**Bemerkung 6.4 (Berechnung der Gewichte)** Da die Interpolation eines Polynoms  $m$ -ten Grades in paarweise verschiedenen Punkten  $x_0, \dots, x_m$  nach dem Identitätssatz (zweite Aussage des Lemmas 5.2) das Polynom exakt reproduziert, muss eine interpolatorische Quadraturformel solche Polynome exakt integrieren.

Diese Eigenschaft können wir nutzen, um die Quadraturgewichte zu berechnen, ohne die Lagrange-Polynome explizit integrieren zu müssen: Für jedes  $i \in [0 : m]$  besitzt das Monom  $\mu_i : x \mapsto x^i$  die Stammfunktion  $x \mapsto \frac{x^{i+1}}{i+1}$ , so dass

$$\sum_{j=0}^m w_j x_j^i = \mathcal{Q}_{[a,b]}(\mu_i) = \int_a^b \mu_i(x) dx = \frac{b^{i+1} - a^{i+1}}{i+1} \quad \text{für alle } i \in [0 : m]$$

gelten muss. Da die  $x_j$  bekannt sind, handelt es sich bei diesen Gleichungen um ein lineares Gleichungssystem für die unbekanntenen Gewichte  $w_0, \dots, w_m$ , das wir mit den bereits behandelten Verfahren lösen können.

Ein besonders einfacher Zugang besteht darin, die Interpolationspunkte *äquidistant* zu wählen, also das Intervall  $[a, b]$  in gleich große Teile zu zerlegen.

**Beispiel 6.5 (Newton-Cotes-Quadraturformeln)** Für jedes  $m \in \mathbb{N}$  definieren die durch

$$x_i := \frac{m-i}{m}a + \frac{i}{m}b \quad \text{für alle } i \in \{0, \dots, m\}$$

gegebenen Interpolationspunkte eine interpolatorische Quadraturformel, die wir als Newton-Cotes-Quadraturformel  $m$ -ten Grades bezeichnen. Die Newton-Cotes-Formel ersten Grades beispielsweise ist durch

$$\mathcal{Q}_{[a,b]} : C[a,b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{Q}_{[a,b]}(f) := \frac{b-a}{2}(f(a) + f(b)),$$

gegeben und trägt auch den Namen Trapezregel (vgl. Abbildung 6.3).

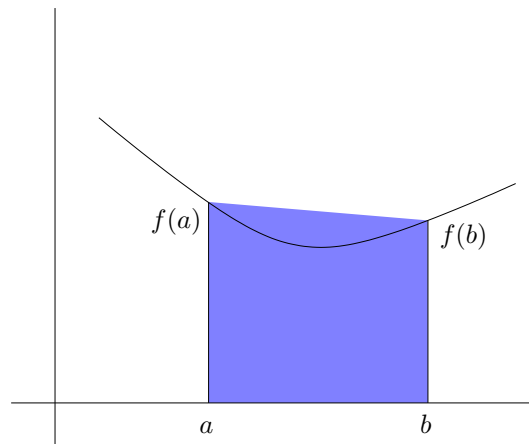


Abbildung 6.3: Idee der Trapezregel: Die Fläche unter der Kurve wird durch ein Trapez angenähert.

**Beispiel 6.6 (Simpson-Regel)** Die Newton-Cotes-Quadraturformel zweiten Grades ist auch als Simpson-Regel bekannt. Für das Intervall  $[-1, 1]$  können wir sie mit der in Bemerkung 6.4 beschriebenen Methode konstruieren: Die Quadraturpunkte sind nach Definition  $x_0 = -1$ ,  $x_1 = 0$  und  $x_2 = 1$ . Damit erhalten wir das Gleichungssystem

$$\begin{aligned} w_0 + w_1 + w_2 &= w_0x_0^0 + w_1x_1^0 + w_2x_2^0 = \int_{-1}^1 \mu_0(x) dx = 2, \\ -w_0 + w_2 &= w_0x_0^1 + w_1x_1^1 + w_2x_2^1 = \int_{-1}^1 \mu_1(x) dx = 0, \end{aligned}$$

## 6 Numerische Integration

$$w_0 + w_2 = w_0 x_0^2 + w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^1 \mu_2(x) dx = 2/3$$

für die Gewichte  $w_0, w_1$  und  $w_2$ . Aus der zweiten Zeile folgt  $w_0 = w_2$ , Einsetzen in die dritte Zeile ergibt  $w_0 = w_2 = 1/3$ , und mit der ersten Zeile erhalten wir das letzte Gewicht  $w_1 = 4/3$ .

Also hat die Simpson-Regel für das Intervall  $[-1, 1]$  die Form

$$\mathcal{S}_{[-1,1]}: C[-1, 1] \rightarrow \mathbb{R}, \quad f \mapsto \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1).$$

Eine interessante Eigenschaft der Simpson-Regel besteht darin, dass sie zwar für quadratische Polynome konstruiert ist, aber auch kubische Polynome noch exakt integriert.

Es stellt sich die Frage, wie wir dafür sorgen können, dass ein Integral mit einer hinreichenden Genauigkeit approximiert wird. Wie man an der Formel (6.1) sieht, hängt der Fehler von der Ableitung des Integranden  $f$  und der Größe des Intervalls ab. Den Integranden können wir nicht ändern, aber die Größe des Intervalls lässt sich reduzieren, indem wir das Integral nicht auf dem gesamten Intervall  $[a, b]$  approximieren, sondern auf Teilintervallen. Besonders einfach ist in diesem Fall wieder die äquidistante Unterteilung in  $\ell \in \mathbb{N}$  Intervalle, die durch

$$h := \frac{b-a}{\ell}, \quad y_i := a + ih \quad \text{für alle } i \in \{0, \dots, \ell\}$$

definiert wird. Wir erhalten

$$\mathcal{I}_{[a,b]}(f) = \int_a^b f(x) dx = \sum_{i=1}^{\ell} \int_{y_{i-1}}^{y_i} f(x) dx = \sum_{i=1}^{\ell} \mathcal{I}_{[y_{i-1}, y_i]}(f) \approx \sum_{i=1}^{\ell} \mathcal{Q}_{[y_{i-1}, y_i]}(f),$$

wobei  $\mathcal{Q}_{[y_{i-1}, y_i]}$  jeweils eine Quadraturformel auf dem Teilintervall  $[y_{i-1}, y_i]$  ist. Wir können leicht nachprüfen, dass

$$\mathcal{Q}_{[a,b],\ell}(f) := \sum_{i=1}^{\ell} \mathcal{Q}_{[y_{i-1}, y_i]}(f) \quad \text{für alle } f \in C[a, b]$$

eine Quadraturformel ist, die wir als *summierte Quadraturformel* bezeichnen.

**Beispiel 6.7 (Summierte Mittelpunkregel)** Aus der Mittelpunkregel können wir in der beschriebenen Weise die summierte Mittelpunkregel konstruieren, die durch

$$\mathcal{M}_{[a,b],\ell}(f) = \sum_{i=1}^{\ell} \mathcal{M}_{[y_{i-1}, y_i]}(f) = h \sum_{i=1}^{\ell} f\left(\frac{y_i + y_{i-1}}{2}\right) \quad \text{für alle } f \in C[a, b]$$

definiert ist. Für  $f \in C^2[a, b]$  erhalten wir durch Einsetzen der Gleichung (6.1) die Fehlergleichung

$$\mathcal{I}_{[a,b]}(f) - \mathcal{M}_{[a,b],\ell}(f) = \frac{(b-a)^3}{24 \ell^3} \sum_{i=1}^{\ell} f''(\eta_i)$$

mit Zwischenpunkten  $\eta_i \in [y_{i-1}, y_i]$ , und mit dem Mittelwertsatz für stetige Funktionen folgt

$$\mathcal{I}_{[a,b]}(f) - \mathcal{M}_{[a,b],\ell}(f) = \frac{(b-a)^3}{24 \ell^2} f''(\eta) \quad (6.2)$$

mit einem  $\eta \in [a, b]$ . Man sieht, dass wir jede beliebige Genauigkeit erreichen können, falls  $f''$  beschränkt ist und wir die Anzahl  $\ell$  der Teilintervalle groß genug wählen. Asymptotisch ist zu erwarten, dass eine Verdoppelung der Anzahl der Teilintervalle den Fehler viertelt.

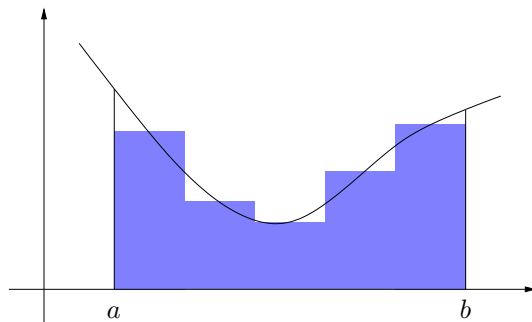


Abbildung 6.4: Summierte Mittelpunktmittelwertregel mit  $\ell = 5$ .

### 6.3 Fehleranalyse

Um den Quadraturfehler allgemein analysieren zu können, bietet es sich an, Quadraturformeln auf einem festen *Referenzintervall* zu definieren und daraus dann per Transformation Quadraturformeln auf allgemeinen Intervallen zu gewinnen. Wir verwenden wieder das Intervall  $[-1, 1]$  und wollen das Integral

$$\mathcal{I} : C[-1, 1] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{I}(f) := \int_{-1}^1 f(x) dx,$$

approximieren. Dazu gehen wir davon aus, dass eine Quadraturformel

$$\mathcal{Q} : C[-1, 1] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{Q}(f) = \sum_{i=0}^m w_i f(x_i),$$

mit Gewichten  $w_0, \dots, w_m$  und Quadraturpunkten  $x_0, \dots, x_m$  gegeben ist.

Die bisher diskutierten Quadraturformeln sind alle dazu in der Lage, Polynome bis zu einem gewissen Grad exakt zu integrieren, beispielsweise lineare Polynome bei Mittelpunkt- und Trapezregel.

**Definition 6.8 (Exakte Quadraturformel)** Sei  $n \in \mathbb{N}_0$ . Wir nennen die Quadraturformel  $\mathcal{Q}$  exakt von Grad  $n$ , falls

$$\mathcal{Q}(p) = \mathcal{I}(p) \quad \text{für alle } p \in \Pi_n$$

gilt, falls also Polynome bis zum Grad  $n$  exakt integriert werden.

Da nach Lemma 5.2 das Interpolationsproblem genau eine Lösung hat, muss die Interpolation eines Polynoms  $p \in \Pi_m$  in den  $m + 1$  Punkten  $x_0, \dots, x_m$  wieder  $p$  ergeben, also sind interpolatorische Quadraturformeln mindestens von Grad  $m$  exakt.

**Bemerkung 6.9 (Gauß-Quadratur)** Das Beispiel der Mittelpunkregel zeigt, dass interpolatorische Quadraturformeln bei geeigneter Wahl der Quadraturpunkte auch von Grad  $m + 1$  exakt sein können. Bei der Gauß-Quadratur ist es sogar möglich, Polynome bis zu einem Grad von  $2m + 1$  exakt zu integrieren, und es lässt sich zeigen, dass ein noch höherer Grad mit  $m + 1$  Quadraturpunkten nicht zu erreichen ist.

Die Eigenschaft der Exaktheit lässt sich ausnutzen: Wir approximieren den Integranden  $f$  durch ein Polynom  $p \in \Pi_n$  und wissen, dass  $p$  exakt integriert wird, so dass wir lediglich den Approximationsfehler beschränken müssen. Den Fehler messen wir wieder mit der Maximumnorm

$$\|g\|_{\infty, [a, b]} := \max\{|g(x)| : x \in [a, b]\} \quad \text{für alle } g \in C[a, b].$$

Den Einfluss einer Störung des Integranden auf das Ergebnis der Quadraturformel können wir wie folgt beschränken:

**Lemma 6.10 (Stabilität)** Wir nennen

$$C_Q := \sum_{i=0}^m |w_i|$$

die Stabilitätskonstante der Quadraturformel  $\mathcal{Q}$ . Es gelten

$$|\mathcal{Q}(f)| \leq C_Q \|f\|_{\infty, [-1, 1]}, \quad |\mathcal{I}(f)| \leq 2 \|f\|_{\infty, [-1, 1]} \quad \text{für alle } f \in C[-1, 1].$$

*Beweis.* Sei  $f \in C[-1, 1]$ . Mit Hilfe der Dreiecksungleichung erhalten wir

$$|\mathcal{Q}(f)| = \left| \sum_{i=0}^m w_i f(x_i) \right| \leq \sum_{i=0}^m |w_i| |f(x_i)| \leq \sum_{i=0}^m |w_i| \|f\|_{\infty, [-1, 1]} = C_Q \|f\|_{\infty, [-1, 1]}.$$

Aus der Monotonie des Integrals folgt dagegen

$$|\mathcal{I}(f)| = \left| \int_{-1}^1 f(x) dx \right| \leq \int_{-1}^1 |f(x)| dx \leq \int_{-1}^1 \|f\|_{\infty, [-1, 1]} dx = 2 \|f\|_{\infty, [-1, 1]}.$$

■

Mit Hilfe dieser Stabilitätsabschätzung können wir den Quadraturfehler allgemein beschränken:



**Lemma 6.11 (Fehler auf dem Referenzintervall)** Sei  $\mathcal{Q}$  exakt von Grad  $n \in \mathbb{N}_0$ . Es gilt

$$|\mathcal{I}(f) - \mathcal{Q}(f)| \leq (2 + C_Q) \frac{\|f^{(n+1)}\|_{\infty,[-1,1]}}{(n+1)!} \quad \text{für alle } f \in C^{n+1}[-1,1]. \quad (6.3)$$

*Beweis.* Sei  $f \in C^{n+1}[-1,1]$ . Wir bezeichnen mit  $p \in \Pi_n$  das Taylor-Polynom  $n$ -ten Grades im Entwicklungspunkt null, gegeben durch

$$p(x) = \sum_{i=0}^n x^i \frac{f^{(i)}(0)}{i!} \quad \text{für alle } x \in [-1,1].$$

Nach dem Satz von Taylor existiert für jedes  $x \in [-1,1]$  ein  $\eta \in [-1,1]$ , das

$$|f(x) - p(x)| = |x|^{n+1} \frac{|f^{(n+1)}(\eta)|}{(n+1)!} \leq \frac{\|f^{(n+1)}\|_{\infty,[-1,1]}}{(n+1)!}$$

erfüllt, also gilt insgesamt

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

Da  $\mathcal{Q}$  exakt von Grad  $n$  ist, gilt  $\mathcal{Q}(p) = \mathcal{I}(p)$ , und dank Lemma 6.10 folgt daraus

$$\begin{aligned} |\mathcal{I}(f) - \mathcal{Q}(f)| &= |\mathcal{I}(f) - \mathcal{I}(p) + \mathcal{Q}(p) - \mathcal{Q}(f)| = |\mathcal{I}(f - p) + \mathcal{Q}(p - f)| \\ &\leq |\mathcal{I}(f - p)| + |\mathcal{Q}(p - f)| \leq 2\|f - p\|_{\infty,[-1,1]} + C_Q\|f - p\|_{\infty,[-1,1]} \\ &= (2 + C_Q)\|f - p\|_{\infty,[-1,1]} \leq (2 + C_Q) \frac{\|f^{(n+1)}\|_{\infty,[-1,1]}}{(n+1)!}. \end{aligned}$$

■

Wie wir bereits gesehen haben, ist die Genauigkeit der Approximation von der Größe des Intervalls abhängig. Um diese Abhängigkeit genauer untersuchen zu können, führen wir die Transformation

$$\Phi_{[a,b]} : [-1,1] \rightarrow [a,b], \quad \hat{x} \mapsto \Phi_{[a,b]}(\hat{x}) := \frac{b+a}{2} + \frac{b-a}{2}\hat{x},$$

ein und erhalten mit der Transformationsformel für Integrale

$$\begin{aligned} \mathcal{I}_{[a,b]}(f) &= \int_a^b f(x) dx = \int_{-1}^1 |\Phi'_{[a,b]}(\hat{x})| f(\Phi_{[a,b]}(\hat{x})) d\hat{x} \\ &= \frac{b-a}{2} \mathcal{I}(f \circ \Phi_{[a,b]}) \quad \text{für alle } f \in C[a,b]. \end{aligned} \quad (6.4)$$

Indem wir das Integral durch die Quadraturformel ersetzen, können wir aus der Quadraturformel  $\mathcal{Q}$  auf  $[-1,1]$  eine Quadraturformel  $\mathcal{Q}_{[a,b]}$  auf dem allgemeinen Intervall  $[a,b]$  konstruieren:

**Definition 6.12 (Transformierte Quadraturformel)** Seien  $a, b \in \mathbb{R}$  mit  $a < b$  gegeben. Dann bezeichnen wir

$$\mathcal{Q}_{[a,b]} : C[a, b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{Q}_{[a,b]}(f) := \frac{b-a}{2} \mathcal{Q}(f \circ \Phi_{[a,b]}),$$

als die zu  $\mathcal{Q}$  gehörende transformierte Quadraturformel auf dem Intervall  $[a, b]$ .

Offenbar gilt

$$\mathcal{Q}_{[a,b]}(f) = \frac{b-a}{2} \sum_{i=0}^m w_i f \circ \Phi_{[a,b]}(x_i) = \sum_{i=0}^m \frac{b-a}{2} w_i f(\Phi_{[a,b]}(x_i)) \quad \text{für alle } f \in C[a, b],$$

die transformierte Quadraturformel verwendet also gerade die mit  $(b-a)/2$  skalierten Quadraturgewichte und die mit  $\Phi_{[a,b]}$  transformierten Quadraturpunkte der ursprünglichen Quadraturformel.

Um aus Lemma 6.11 eine Aussage für die transformierte Quadraturformel  $\mathcal{Q}_{[a,b]}$  zu erhalten, müssen wir untersuchen, wie sich die Ableitungen der transformierten Funktion  $f \circ \Phi_{[a,b]}$  verhalten.

**Lemma 6.13 (Skalierung)** Sei  $n \in \mathbb{N}_0$ , und sei  $f \in C^n[a, b]$ . Dann erfüllt  $\hat{f} := f \circ \Phi_{[a,b]} \in C^n[-1, 1]$  die Gleichung

$$\hat{f}^{(n)}(\hat{x}) = \left(\frac{b-a}{2}\right)^n f^{(n)}(\Phi_{[a,b]}(\hat{x})) \quad \text{für alle } \hat{x} \in [-1, 1].$$

*Beweis.* Per Induktion über  $n \in \mathbb{N}_0$ . Für  $n = 0$  ist die Aussage trivial.

Sei nun  $n \in \mathbb{N}_0$  so gewählt, dass die Aussage gilt. Sei  $f \in C^{n+1}[a, b]$  und  $\hat{f} = f \circ \Phi_{[a,b]}$ . Mit der Induktionsvoraussetzung erhalten wir

$$\hat{f}^{(n)}(\hat{x}) = \left(\frac{b-a}{2}\right)^n f^{(n)} \circ \Phi_{[a,b]}(\hat{x}) \quad \text{für alle } \hat{x} \in [-1, 1],$$

und mit der Kettenregel folgt

$$\begin{aligned} \hat{f}^{(n+1)}(\hat{x}) &= \left(\frac{b-a}{2}\right)^n (f^{(n)} \circ \Phi_{[a,b]})'(\hat{x}) = \left(\frac{b-a}{2}\right)^n f^{(n+1)}(\Phi_{[a,b]}(\hat{x})) \Phi'_{[a,b]}(\hat{x}) \\ &= \left(\frac{b-a}{2}\right)^{n+1} f^{(n+1)}(\Phi_{[a,b]}(\hat{x})) \quad \text{für alle } \hat{x} \in [-1, 1]. \end{aligned}$$

Das ist die gewünschte Aussage. ■

Nun können wir ausnutzen, dass die beiden Seiten der Gleichung (6.3) sich bei der Transformation des Intervalls unterschiedlich verhalten:

**Lemma 6.14 (Quadraturfehler)** Sei  $\mathcal{Q}$  exakt von Grad  $n \in \mathbb{N}_0$ . Es gilt

$$|\mathcal{I}_{[a,b]}(f) - \mathcal{Q}_{[a,b]}(f)| \leq (2 + C_Q) \left(\frac{b-a}{2}\right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty, [a,b]}}{(n+1)!} \quad \text{für alle } f \in C^{n+1}[a, b].$$

*Beweis.* Sei  $f \in C^{n+1}[a, b]$ , und sei wieder  $\hat{f} := f \circ \Phi_{[a,b]}$  die transformierte Funktion auf dem Referenzintervall  $[-1, 1]$ .

Nach Transformationsformel (6.4) und Definition 6.12 gilt

$$|\mathcal{I}_{[a,b]}(f) - \mathcal{Q}_{[a,b]}(f)| = \frac{b-a}{2} |\mathcal{I}(\hat{f}) - \mathcal{Q}(\hat{f})|.$$

Wir wenden Lemma 6.11 an und erhalten

$$|\mathcal{I}_{[a,b]}(f) - \mathcal{Q}_{[a,b]}(f)| \leq (2 + C_Q) \frac{b-a}{2} \frac{\|\hat{f}^{(n+1)}\|_{\infty,[-1,1]}}{(n+1)!}.$$

Den letzten Faktor können wir mit Lemma 6.13 berechnen und erhalten

$$\begin{aligned} |\mathcal{I}_{[a,b]}(f) - \mathcal{Q}_{[a,b]}(f)| &\leq (2 + C_Q) \frac{b-a}{2} \left(\frac{b-a}{2}\right)^{n+1} \frac{\|f^{(n+1)} \circ \Phi_{[a,b]}\|_{\infty,[-1,1]}}{(n+1)!} \\ &= (2 + C_Q) \left(\frac{b-a}{2}\right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty,[a,b]}}{(n+1)!}, \end{aligned}$$

indem wir im letzten Schritt ausnutzen, dass  $\Phi_{[a,b]}$  eine bijektive Abbildung des Referenzintervalls  $[-1, 1]$  auf  $[a, b]$  ist. ■

Diesem Lemma lässt sich entnehmen, dass wir den Fehler beliebig reduzieren können, indem wir das Integrationsintervall verkleinern. Um trotzdem das ursprüngliche Integral zu berechnen, verwenden wir wie zuvor eine *summierte Quadraturformel*.

**Definition 6.15 (Summierte Quadraturformel)** Seien  $a, b \in \mathbb{R}$  mit  $a < b$  gegeben, sei  $\ell \in \mathbb{N}$ . Wir setzen

$$h := \frac{b-a}{\ell}, \quad y_j := a + jh \quad \text{für alle } j \in \{0, \dots, \ell\}$$

und definieren mit

$$\mathcal{Q}_{[a,b],\ell} : C[a, b] \rightarrow \mathbb{R}, \quad f \mapsto \mathcal{Q}_{[a,b],\ell} := \sum_{j=1}^{\ell} \mathcal{Q}_{[y_{j-1}, y_j]}(f),$$

die zu  $\mathcal{Q}$  gehörende summierte Quadraturformel auf dem  $\ell$ -fach unterteilten Intervall  $[a, b]$ . Hier ist  $\mathcal{Q}_{[y_{j-1}, y_j]}$  jeweils die aus Definition 6.12 bekannte transformierte Quadraturformel auf dem Teilintervall  $[y_{j-1}, y_j]$ .

**Satz 6.16 (Genauigkeit)** Sei  $\mathcal{Q}$  exakt von Grad  $n \in \mathbb{N}_0$ , sei  $\ell \in \mathbb{N}$ . Es gilt

$$|\mathcal{I}_{[a,b]}(f) - \mathcal{Q}_{[a,b],\ell}(f)| \leq \frac{2 + C_Q}{\ell^{n+1}} \left(\frac{b-a}{2}\right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty,[a,b]}}{(n+1)!} \quad \text{für alle } f \in C^{n+1}[a, b].$$

Insbesondere können wir jede beliebige Genauigkeit erreichen, indem wir  $\ell$  hinreichend groß wählen.

## 6 Numerische Integration

*Beweis.* Sei  $f \in C^{n+1}[a, b]$ . Wie in Definition 6.15 setzen wir

$$h := \frac{b-a}{\ell}, \quad y_j := a + jh \quad \text{für alle } j \in \{0, \dots, \ell\}.$$

Insbesondere gilt damit

$$y_j - y_{j-1} = h = (b-a)/\ell \quad \text{für alle } j \in \{1, \dots, \ell\},$$

und mit der Dreiecksungleichung und Lemma 6.14 erhalten wir

$$\begin{aligned} |\mathcal{I}_{[a,b]}(f) - \mathcal{Q}_{[a,b],\ell}(f)| &= \left| \sum_{j=1}^{\ell} \mathcal{I}_{[y_{j-1}, y_j]}(f) - \sum_{j=1}^{\ell} \mathcal{Q}_{[y_{j-1}, y_j]}(f) \right| \\ &= \left| \sum_{j=1}^{\ell} \mathcal{I}_{[y_{j-1}, y_j]}(f) - \mathcal{Q}_{[y_{j-1}, y_j]}(f) \right| \\ &\leq \sum_{j=1}^{\ell} |\mathcal{I}_{[y_{j-1}, y_j]}(f) - \mathcal{Q}_{[y_{j-1}, y_j]}(f)| \\ &\leq (2 + C_Q) \sum_{j=1}^{\ell} \left( \frac{y_j - y_{j-1}}{2} \right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty, [y_{j-1}, y_j]}}{(n+1)!} \\ &\leq (2 + C_Q) \sum_{j=1}^{\ell} \left( \frac{b-a}{2\ell} \right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty, [a,b]}}{(n+1)!} \\ &= \frac{2 + C_Q}{\ell^{n+2}} \ell \left( \frac{b-a}{2} \right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty, [a,b]}}{(n+1)!} \\ &= \frac{2 + C_Q}{\ell^{n+1}} \left( \frac{b-a}{2} \right)^{n+2} \frac{\|f^{(n+1)}\|_{\infty, [a,b]}}{(n+1)!}, \end{aligned}$$

also das gewünschte Ergebnis. ■

Wenn  $x_0, \dots, x_m \in [-1, 1]$  die Quadraturpunkte und  $w_0, \dots, w_m \in \mathbb{R}$  die Quadraturgewichte der Quadraturformel  $\mathcal{Q}$  für das Referenzintervall  $[-1, 1]$  sind, lässt sich die korrespondierende summierte Quadraturformel auf einem Intervall  $[a, b]$  mit dem folgenden einfachen Algorithmus auswerten:

```

function integrate( $\ell, a, b, f$ );
 $q \leftarrow 0$ ;     $h \leftarrow (b-a)/\ell$ ;
for  $j \in \{1, \dots, \ell\}$  do begin
     $y \leftarrow a + h(j-1)$ ;
    for  $i \in \{0, \dots, m\}$  do begin
         $x \leftarrow y + h(x_i + 1)/2$ ;     $q \leftarrow q + w_i f(x)$ 
    end
end;
return  $qh/2$ 

```

**Bemerkung 6.17 (Effizienz)** Satz 6.16 legt nahe, dass sich der Quadraturfehler proportional zu  $1/\ell^{n+1}$  verhalten wird, wenn wir die Anzahl  $\ell$  der Teilintervalle erhöhen. Da die Auswertung der summierten Quadraturformel  $\ell$  Auswertungen der transformierten Quadraturformel erfordert, ist die Anzahl der Rechenoperationen proportional zu  $\ell(m+1)$ .

Wenn wir also mit möglichst geringem Rechenaufwand eine hohe Genauigkeit erreichen wollen, empfiehlt es sich, Quadraturformeln zu verwenden, die von möglichst hohem Grad exakt sind, denn in diesem Fall gewinnen wir am meisten, wenn wir  $\ell$  erhöhen.

Allerdings erfordert Satz 6.16, dass der Integrand  $f$  hinreichend oft differenzierbar ist. Falls beispielsweise  $f$  nur zweimal differenzierbar ist, wird auch eine Quadraturformel fünften Grades keinen großen Vorteil gegenüber der Mittelpunktregel bieten.



# 7 Gewöhnliche Differentialgleichungen

Viele zeitabhängige Vorgänge in den Natur-, Ingenieur- und Wirtschaftswissenschaften werden beschrieben, indem man den Zustand, den ein System zu einem Zeitpunkt einnimmt, zu der Veränderung des Zustands in Beziehung setzt. Mathematisch werden solche Vorgänge durch gewöhnliche Differentialgleichungen beschrieben, deren Lösungen sich häufig nicht in geschlossener Form darstellen lassen, so dass numerische Näherungsverfahren zum Einsatz kommen müssen.

Bei deren Entwicklung können wir auf die im vorigen Kapitel eingeführten Quadraturverfahren zurückgreifen: Die Differentialgleichung kann äquivalent in eine Integralgleichung überführt werden, und das auftretende Integral kann mit einer Quadraturformel approximiert werden.

## 7.1 Beispiel: Numerische Simulationen

Viele physikalische Systeme sind durch einen *Zustand* beschrieben, der sich mit der Zeit verändert. Ein einfaches Beispiel ist ein idealisiertes Federpendel, dessen Auslenkung  $x(t)$  und Geschwindigkeit  $v(t)$  zu einem Zeitpunkt  $t$  durch die Gleichungen

$$x'(t) = v(t), \quad v'(t) = -cx(t) \quad \text{für alle } t \in \mathbb{R}$$

gegeben sind, bei denen die *Federkonstante* die Materialeigenschaften des Pendels beschreibt. Wenn wir beide Größen zu einem Vektor

$$y(t) := \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} \quad \text{für alle } t \in \mathbb{R}$$

zusammenfassen, der den Zustand des Systems zum Zeitpunkt  $t$  vollständig beschreibt, können wir das System kompakt in der Form

$$y'(t) = \begin{pmatrix} & 1 \\ -c & \end{pmatrix} y(t) \quad \text{für alle } t \in \mathbb{R} \quad (7.1)$$

schreiben, die Ableitung der Funktion zum Zeitpunkt  $t$  hängt also von dem Wert der Funktion zu diesem Zeitpunkt ab.

Auch in der Biologie lassen sich gelegentlich Abhängigkeiten zwischen dem aktuellen Zustand eines Systems und seiner Veränderung formulieren. Ein Beispiel sind die *Lotka-Volterra-Gleichungen*

$$b'(t) = b(t)(\epsilon_1 - \gamma_1 r(t)), \quad r'(t) = r(t)(\gamma_2 b(t) - \epsilon_2) \quad \text{für alle } t \in \mathbb{R},$$

die die Wechselwirkung zwischen einer Population von Beutetieren und einer von Raubtieren modellieren. Dabei gibt  $b(t)$  die (gemittelte) Anzahl der Beutetiere zu einem Zeitpunkt  $t$  an, während  $r(t)$  die Anzahl der Raubtiere ist.  $\epsilon_1 > 0$  gibt die Rate an, mit der sich die Beutetiere vermehren, während  $\epsilon_2 > 0$  die Rate angibt, mit der die Raubtiere aussterben, sofern sie keine Nahrung finden.  $\gamma_1$  und  $\gamma_2$  beschreiben, um wie viel sich die Beutetierpopulation reduziert und die Raubtierpopulation vermehrt, wenn Beutetiere gefressen werden.

Auch hier können wir den Zustand durch einen Vektor

$$y(t) := \begin{pmatrix} b(t) \\ r(t) \end{pmatrix} \quad \text{für alle } t \in \mathbb{R}$$

beschreiben, der die Gleichung

$$y'(t) = \begin{pmatrix} y_1(t)(\epsilon_1 - \gamma_1 y_2(t)) \\ y_2(t)(\gamma_2 y_1(t) - \epsilon_2) \end{pmatrix} \quad \text{für alle } t \in \mathbb{R} \quad (7.2)$$

erfüllt.

Während sich für die Gleichung (7.1) eine exakte Lösung angeben lässt, ist für die Gleichung (7.2) keine bekannt, so dass Näherungsverfahren zum Einsatz kommen.

## 7.2 Integralgleichung

Wir stellen uns die Aufgabe, relativ allgemeine explizite *Anfangswertprobleme* zu lösen:

Sei  $n \in \mathbb{N}$ . Gegeben ein Anfangswert  $y_0 \in \mathbb{R}^n$  und eine stetige Funktion  $f: [a, b] \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ .

Gesucht ist eine Funktion  $y \in C^1([a, b], \mathbb{R}^n)$  mit

$$y(a) = y_0, \quad (7.3a)$$

$$y'(t) = f(t, y(t)) \quad \text{für alle } t \in [a, b]. \quad (7.3b)$$

Bei der Behandlung dieser Aufgabe können wir auf die bei der numerischen Integration gesammelte Erfahrung zurückgreifen, wenn wir die Aufgabe von einer Differential- in eine Integralgleichung überführen.

**Satz 7.1 (Integralgleichung)** *Eine Lösung  $y \in C^1([a, b], \mathbb{R}^n)$  des Anfangswertproblems (7.3) erfüllt die Integralgleichung*

$$y(t) = y_0 + \int_a^t f(s, y(s)) ds \quad \text{für alle } t \in [a, b]. \quad (7.4)$$

*Umgekehrt ist eine Lösung  $y \in C([a, b], \mathbb{R}^n)$  dieser Integralgleichung stetig differenzierbar und Lösung des Anfangswertproblems (7.3).*



*Beweis.* Sei zunächst  $y \in C^1([a, b], \mathcal{V})$  eine Lösung des Anfangswertproblems (7.3). Mit dem Hauptsatz der Integral- und Differentialrechnung folgt

$$y(t) = y(a) + \int_a^t y'(s) ds = y_0 + \int_a^t f(s, y(s)) ds \quad \text{für alle } t \in [a, b],$$

also die Integralgleichung (7.4).

Sei nun  $y \in C([a, b], \mathcal{V})$  eine Lösung der Integralgleichung (7.4). Aus dieser Gleichung folgt unmittelbar  $y(a) = y_0$ , wir müssen also nur noch nachweisen, dass  $y$  stetig differenzierbar ist und die Ableitung den richtigen Wert aufweist.

Seien dazu  $t \in [a, b)$  und  $h \in \mathbb{R}_{>0}$  mit  $t + h \in [a, b]$  gegeben. Es gilt

$$\begin{aligned} f(t, y(t)) - \frac{y(t+h) - y(t)}{h} &= f(t, y(t)) - \frac{1}{h} \left( \int_a^{t+h} f(s, y(s)) ds - \int_a^t f(s, y(s)) ds \right) \\ &= f(t, y(t)) - \frac{1}{h} \int_t^{t+h} f(s, y(s)) ds \\ &= \frac{1}{h} \int_t^{t+h} f(t, y(t)) - f(s, y(s)) ds. \end{aligned}$$

Mit der Dreiecksungleichung für Integrale und dem Mittelwertsatz der Integralrechnung finden wir ein  $\eta_h \in [t, t+h]$  mit

$$\begin{aligned} \left\| f(t, y(t)) - \frac{y(t+h) - y(t)}{h} \right\| &= \frac{1}{h} \left\| \int_t^{t+h} f(t, y(t)) - f(s, y(s)) ds \right\| \\ &\leq \frac{1}{h} \int_t^{t+h} \|f(t, y(t)) - f(s, y(s))\| ds \\ &= \frac{h}{h} \|f(t, y(t)) - f(\eta_h, y(\eta_h))\| \\ &= \|f(t, y(t)) - f(\eta_h, y(\eta_h))\|. \end{aligned}$$

Da  $y$  und  $f$  stetig sind und  $\eta_h \in [t, t+h]$  gilt, haben wir

$$\lim_{h \rightarrow 0} y(\eta_h) = \lim_{\eta_h \rightarrow t} y(\eta_h) = y(t), \quad \lim_{h \rightarrow 0} f(\eta_h, y(\eta_h)) = f(t, y(t)),$$

so dass insgesamt

$$\lim_{h \rightarrow 0} \left\| f(t, y(t)) - \frac{y(t+h) - y(t)}{h} \right\| = 0$$

folgt, also

$$\lim_{h \rightarrow 0} \frac{y(t+h) - y(t)}{h} = f(t, y(t)).$$

Damit ist  $y$  in  $t$  differenzierbar mit der Ableitung  $y'(t) = f(t, y(t))$ .

Für  $t = b$  können wir entsprechend mit dem linksseitigen Differenzenquotienten  $y'(b) = \lim_{h \rightarrow 0} \frac{y(b) - y(b-h)}{h}$  verfahren. ■

## 7 Gewöhnliche Differentialgleichungen

Wenn wir den Operator

$$\Phi: C([a, b], \mathcal{V}) \rightarrow C([a, b], \mathcal{V}), \quad y \mapsto \left( t \mapsto y_0 + \int_a^t f(s, y(s)) ds \right),$$

definieren, der den Raum  $C([a, b], \mathcal{V})$  der stetigen Funktionen auf sich selbst abbildet, können wir die Integralgleichung (7.4) kompakt in der Form

$$y = \Phi[y]$$

einer Fixpunktgleichung schreiben. Derartige Gleichungen sind uns schon in Abschnitt 3.3 begegnet, in dem uns der Fixpunktsatz 3.8 eine Methode zu deren Behandlung bot: Falls  $\Phi$  eine Kontraktion bezüglich einer geeigneten Norm ist, können wir ausgehend von einem beliebigen Näherung  $y^{(0)} \in C([a, b], \mathcal{V})$  die durch

$$y^{(m+1)} := \Phi[y^{(m)}] \quad \text{für alle } m \in \mathbb{N}_0$$

definierte Folge  $(y^{(m)})_{m=0}^\infty$  der Näherungslösungen einführen, die gegen den gesuchten Fixpunkt konvergieren wird. Wir haben den Fixpunktsatz zwar nur für Teilmengen des Raums  $\mathbb{R}^n$  formuliert, aber er gilt tatsächlich auch in allgemeinen vollständigen Räumen, und  $C([a, b], \mathcal{V})$  ist ein solcher Raum.

Unter gewissen Bedingungen lässt sich zeigen, dass  $\Phi$  bezüglich einer geschickt gewählten Norm tatsächlich eine Kontraktion ist und deshalb die Integralgleichung (7.4) genau eine Lösung besitzt.

Die zugehörige Fixpunktiteration ist als *Picard-Iteration* bekannt und kann in der Form

$$y^{(m+1)}(t) = y_0 + \int_a^t f(s, y^{(m)}(s)) ds \quad \text{für alle } m \in \mathbb{N}_0, t \in [a, b]$$

geschrieben werden. In der Praxis kommt sie eher selten zum Einsatz, denn sie erfordert es, eine Approximation der gesamten Lösung  $y^{(m)}$  in jedem Schritt abzuspeichern und zu verarbeiten.

### 7.3 Zeitschrittverfahren

Das in (7.4) auftretende Integral können wir mit einer Quadraturformel approximieren, um

$$y(b) \approx \sum_{i=0}^m w_i f(s_i, y(s_i))$$

zu erhalten, wobei  $s_0, \dots, s_m$  die Quadraturpunkte und  $w_0, \dots, w_m$  die Quadraturgewichte sind. Leider kennen wir allerdings  $y$  nur im Anfangszeitpunkt  $a$ , so dass sich lediglich die *Rechteckregel*

$$y(b) \approx y_0 + (b - a)f(a, y_0)$$

unmittelbar anwenden lässt. Dabei handelt es sich um die einzige Quadraturformel mit  $m = 0$  und  $s_0 = a$ , die immerhin konstante Polynome exakt integriert. Sie erreicht allerdings keine besonders hohe Genauigkeit.

**Lemma 7.2 (Rechteckregel)** Sei  $y \in C^2([a, b], \mathcal{V})$ , und sei

$$\tilde{y}(b) := y_0 + (b - a)f(a, y_0)$$

die mit der Rechteckregel berechnete Näherung des Zustands  $y(b)$ . Dann gilt

$$\|y(b) - \tilde{y}(b)\| \leq \frac{(b - a)^2}{2} \|y''\|_{\infty, [a, b]}.$$

*Beweis.* Mit der Integralgleichung (7.4) und dem Hauptsatz der Integral- und Differentialrechnung erhalten wir

$$\begin{aligned} y(b) - \tilde{y}(b) &= y_0 + \int_a^b f(s, y(s)) ds - (y_0 + (b - a)f(a, y_0)) \\ &= \int_a^b f(s, y(s)) - f(a, y(a)) ds = \int_a^b y'(s) - y'(a) ds \\ &= \int_a^b \int_a^s y''(r) dr ds. \end{aligned}$$

Mit der Dreiecksungleichung für Integrale folgt

$$\begin{aligned} \|y(b) - \tilde{y}(b)\| &= \left\| \int_a^b \int_a^s y''(r) dr ds \right\| \leq \int_a^b \int_a^s \|y''(r)\| dr ds \\ &\leq \int_a^b \int_a^s \|y''\|_{\infty, [a, b]} dr ds = \|y''\|_{\infty, [a, b]} \int_a^b (s - a) ds \\ &= \|y''\|_{\infty, [a, b]} \left[ \frac{(s - a)^2}{2} \right]_{s=a}^b = \frac{(b - a)^2}{2} \|y''\|_{\infty, [a, b]}. \end{aligned}$$

■

Um die Genauigkeit zu steuern, können wir wie im Fall der summierten Quadraturverfahren vorgehen: Wir zerlegen das Intervall  $[a, b]$  in Teilintervalle  $[y_{j-1}, y_j]$  und wenden unser Näherungsverfahren auf jedem Teilintervall an.

**Definition 7.3 (Explizites Euler-Verfahren)** Seien  $a = t_0 < t_1 < \dots < t_k = b$  gegeben. Das explizite Euler-Verfahren berechnet mit der Regel

$$\begin{aligned} \tilde{y}(t_0) &:= y_0, \\ \tilde{y}(t_{i+1}) &:= \tilde{y}(t_i) + (t_{i+1} - t_i)f(t_i, \tilde{y}(t_i)) \quad \text{für alle } i \in [0 : k - 1] \end{aligned}$$

Näherungslösungen für  $y(t_0), \dots, y(t_k)$ .

Im Vergleich zu summierten Quadraturformeln stellen sich bei der Fehleranalyse zusätzliche Herausforderungen: Nur auf dem ersten Intervall  $[t_0, t_1]$  verwendet unsere Quadraturformel den exakten Anfangswert  $y(t_0) = y_0$ . Auf allen folgenden Intervallen werden die im vorigen Schritt berechneten Näherungswerte  $\tilde{y}(t_i)$  verwendet, so dass sich einmal aufgetretene Fehler fortpflanzen können.

## 7 Gewöhnliche Differentialgleichungen

Wenn es ein *explizites* Euler-Verfahren gibt, stellt sich natürlich die Frage, ob es auch ein *implizites* gibt. Dem ist in der Tat so: Wenn wir statt des linken Endpunkts des Intervalls den rechten verwenden, erhalten wir

$$y(b) \approx y_0 + (b - a)f(b, y(b)).$$

Da  $y(b)$  auf der rechten Seite auftritt, ist die resultierende Näherungslösung  $\tilde{y}(b)$  nur noch implizit als Lösung der Fixpunktgleichung

$$\tilde{y}(b) = y_0 + (b - a)f(b, \tilde{y}(b))$$

gegeben. Im Allgemeinen muss diese Gleichung mit Hilfe eines geeigneten Iterationsverfahrens näherungsweise gelöst werden. Unter geeigneten Voraussetzungen lässt sich zeigen, dass die Lösung eine Lemma 7.2 entsprechende Fehlerabschätzung erfüllt, auch hier müssen wir also das Intervall in Teilintervalle zerlegen und schrittweise vorgehen.

**Definition 7.4 (Implizites Euler-Verfahren)** Seien  $a = t_0 < t_1 < \dots < t_k = b$  gegeben. Das implizite Euler-Verfahren berechnet mit der Regel

$$\begin{aligned} \tilde{y}(t_0) &:= y_0, \\ \tilde{y}(t_{i+1}) &= \tilde{y}(t_i) + (t_{i+1} - t_i)f(t_{i+1}, \tilde{y}(t_{i+1})) \quad \text{für alle } i \in [0 : k - 1] \end{aligned}$$

Näherungslösungen für  $y(t_0), \dots, y(t_k)$ .

Die Implementierung eines impliziten Verfahrens ist häufig erheblich aufwendiger als die eines expliziten, und das resultierende Programm benötigt häufig auch erheblich mehr Rechenzeit pro Schritt. Es gibt allerdings Differentialgleichungen, bei denen wir bei impliziten Verfahren deutlich weniger Schritte als bei expliziten durchführen müssen, so dass die für die gesamte Simulation erforderliche Zeit bei impliziten Verfahren deutlich geringer als bei expliziten ausfallen kann.

Wenn wir schon dazu bereit sind, auch den rechten Randpunkt des Intervalls für unsere Quadraturformel zu verwenden, können wir auch gleich beide benutzen. Die Trapezregel führt zu

$$\tilde{y}(b) \approx y_0 + \frac{b - a}{2}(f(a, y_0) + f(b, y(b))), \quad (7.5)$$

also wieder zu einer impliziten Formel. Die Näherungslösung ist über die Fixpunktgleichung

$$\tilde{y}(b) = y_0 + \frac{b - a}{2}(f(a, y_0) + f(b, \tilde{y}(b)))$$

gegeben, die wiederum im Allgemeinen mit Hilfe eines Iterationsverfahrens gelöst werden muss. Unter geeigneten Annahmen erreicht die Trapezregel eine wesentlich höhere Genauigkeit als die Rechteckregel, es ist aber immer noch erforderlich, zu Teilintervallen über zu gehen, um die Genauigkeit steuern zu können.

**Definition 7.5 (Crank-Nicolson-Verfahren)** Seien  $a = t_0 < t_1 < \dots < t_k = b$  gegeben. Das Crank-Nicolson-Verfahren berechnet mit der Regel

$$\begin{aligned}\tilde{y}(t_0) &= y_0, \\ \tilde{y}(t_{i+1}) &= \tilde{y}(t_i) + \frac{t_{i+1} - t_i}{2} (f(t_i, \tilde{y}(t_i)) + f(t_{i+1}, \tilde{y}(t_{i+1}))) \quad \text{für alle } i \in [0 : k - 1]\end{aligned}$$

Näherungslösungen für  $y(t_0), \dots, y(t_k)$ .

Dieses Verfahren ist bei Anwendungen besonders interessant, bei denen man jedem Zustand  $y(t)$  eine Energie zuordnen kann. Nach den üblichen Regeln der Physik ist die Energie eine Erhaltungsgröße, sollte also konstant sein. Das Crank-Nicolson-Verfahren kann in manchen Fällen diese Eigenschaft auch für die Näherungslösungen gewährleisten.

Wir können aus dem implizite Crank-Nicolson-Verfahren ein explizites Verfahren machen, indem wir auf der rechten Seite der Gleichung (7.5) den unbekanntem Wert  $y(b)$  durch eine mit Hilfe des expliziten Euler-Verfahrens berechnete Näherung

$$\hat{y}(b) := y_0 + (b - a)f(a, y_0)$$

ersetzen, um zu

$$\begin{aligned}y(b) &\approx y_0 + \frac{b - a}{2} (f(a, y_0) + f(b, \hat{y}(b))) \\ &\approx y_0 + \frac{b - a}{2} (f(a, y_0) + f(b, (y_0 + (b - a)f(a, y_0))))\end{aligned}$$

zu gelangen. Die rechte Seite lässt sich nun wieder explizit auswerten.

**Definition 7.6 (Heun-Verfahren)** Seien  $a = t_0 < t_1 < \dots < t_k = b$  gegeben. Das Heun-Verfahren berechnet mit der Regel

$$\begin{aligned}\tilde{y}(t_0) &= y_0, \\ \hat{y}(t_{i+1}) &:= \tilde{y}(t_i) + (t_{i+1} - t_i)f(t_i, \tilde{y}(t_i)), \\ \tilde{y}(t_{i+1}) &:= \tilde{y}(t_i) + \frac{t_{i+1} - t_i}{2} (f(t_i, \tilde{y}(t_i)) + f(t_{i+1}, \hat{y}(t_{i+1}))) \quad \text{für alle } i \in [0 : k - 1]\end{aligned}$$

Näherungslösungen für  $y(t_0), \dots, y(t_k)$ .

Auch das Heun-Verfahren erreicht in der Regel eine erheblich höhere Genauigkeit als das explizite Euler-Verfahren, bietet aber nicht mehr die Energieerhaltung, die das Crank-Nicolson-Verfahren auszeichnet.

Neben der Trapezregel haben wir im vorigen Kapitel auch die Mittelpunkregel kennen gelernt. Wenn wir sie auf die Integralgleichung (7.4) anwenden, erhalten wir

$$y(b) \approx y_0 + (b - a)f\left(\frac{b+a}{2}, y\left(\frac{b+a}{2}\right)\right).$$

## 7 Gewöhnliche Differentialgleichungen

Auch hier stehen wir vor dem Problem, dass uns  $y\left(\frac{b+a}{2}\right)$  nicht bekannt ist. Analog zu dem Heun-Verfahren lösen wir das Problem näherungsweise mit Hilfe des expliziten Euler-Verfahrens, indem wir

$$\hat{y}\left(\frac{b+a}{2}\right) := y_0 + \frac{b+a}{2}f(a, y_0)$$

als Näherung von  $y\left(\frac{b+a}{2}\right)$  verwenden.

**Definition 7.7 (Runge-Verfahren)** Seien  $a = t_0 < t_1 < \dots < t_k = b$  gegeben. Das Runge-Verfahren berechnet mit der Regel

$$\begin{aligned} \tilde{y}(t_0) &= y_0, \\ t_{i+1/2} &:= \frac{t_{i+1} + t_i}{2}, \\ \hat{y}(t_{i+1/2}) &:= \tilde{y}(t_i) + (t_{i+1/2} - t_i)f(t_i, \tilde{y}(t_i)), \\ \tilde{y}(t_{i+1}) &:= \tilde{y}(t_i) + (t_{i+1} - t_i)f(t_{i+1/2}, \hat{y}(t_{i+1/2})) \quad \text{für alle } i \in [0 : k-1] \end{aligned}$$

Näherungslösungen für  $y(t_0), \dots, y(t_k)$ .

Sowohl das Heun- als auch das Runge-Verfahren werten die rechte Seite  $f$  für mehrere Zeitpunkte und Zustände aus, bevor die Näherung des neuen Zustands ermittelt wird. Diese Vorgehensweise verallgemeinern *Runge-Kutta-Verfahren*, bei denen  $s$  Zwischenergebnisse berechnet werden. Ein solches Verfahren ist von der Gestalt

$$\begin{aligned} k_j &= f\left(t_i + c_j(t_{i+1} - t_i), \tilde{y}(t_i) + (t_{i+1} - t_i) \sum_{\ell=1}^s a_{j\ell} k_\ell\right) \quad \text{für alle } j \in [1 : s], \\ \tilde{y}(t_{i+1}) &:= \tilde{y}(t_i) + (t_{i+1} - t_i) \sum_{j=1}^s b_j k_j. \end{aligned}$$

Falls  $a_{j\ell} = 0$  für alle  $\ell \geq j$  gilt, falls  $k_j$  also nur von  $k_1, \dots, k_{j-1}$  abhängt, handelt es sich um ein explizites Verfahren, ansonsten um ein implizites.

Runge-Kutta-Verfahren lassen sich kompakt durch die Koeffizienten  $a_{11}, \dots, a_{ss}$ ,  $b_1, \dots, b_s$  und  $c_1, \dots, c_s$  darstellen, die traditionell in einem *Butcher-Tableau* der Form

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

zusammengefasst werden. Zu dem expliziten und dem impliziten Euler-Verfahren gehören die Tableaus

$$\frac{0}{1} \left| \begin{array}{c} 0 \\ 1 \end{array} \right. \quad \text{und} \quad \frac{1}{1} \left| \begin{array}{c} 1 \\ 1 \end{array} \right. ,$$

zu Crank-Nicolson, Heun und Runge gehören

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}, \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \quad \text{und} \quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}.$$

Das *klassische Runge-Kutta-Verfahren vierter Ordnung* ist durch das Tableau

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

gegeben. Es handelt sich um ein explizites Verfahren, das aus der Simpson-Regel hergeleitet werden kann.

## 7.4 Verfeinerungen und Variationen\*

Aufgrund der großen Bedeutung numerischer Verfahren für gewöhnliche Differentialgleichungen besteht erhebliches Interesse daran, sie schneller, zuverlässiger und genauer zu machen. In der Regel lassen sich nicht alle Ziele gleichzeitig erreichen, so dass sorgfältig geplant werden sollte, welche Modifikationen zum Einsatz kommen.

Eine sehr häufig verwendete Modifikation ist die *adaptive Schrittweitensteuerung*: Lemma 7.2 legt nahe, dass der auf einem Intervall auftretende Fehler von dem Produkt zwischen dessen Länge und dem Maximum der zweiten Ableitung abhängt. Falls die zweite Ableitung klein ist, dürften wir also mit größeren Intervallen rechnen und so Rechenzeit sparen.

Der Ansatz der adaptiven Schrittweitensteuerung besteht nun darin, die Genauigkeit der Approximation auf einem Intervall zu schätzen. Falls die Schätzung nahe legt, dass die Approximation zu ungenau ist, wird das Intervall verkürzt. Falls die Approximation zu genau ist, wird es verlängert.

Ein besonders elegante Einsatz sind dabei *eingebettete Runge-Kutta-Fehlberg-Verfahren*, bei denen aus den in einem Runge-Kutta-Verfahren hoher Genauigkeit berechnen Hilfsgrößen  $k_1, \dots, k_s$  auch eine Näherung niedrigerer Genauigkeit gewonnen wird. Aus dem Vergleich beider Näherungen kann dann eine verbesserte Schrittweite ermittelt werden.

Beispielsweise können wir das explizite Euler-Verfahren in dem Heun-Verfahren einbetten: Wie in letzterem berechnen wir

$$k_1 := f(t_i, \tilde{y}(t_i)), \quad k_2 := f(t_{i+1}, \tilde{y}(t_i) + (t_{i+1} - t_i)k_1).$$

Dann können wir mit

$$\tilde{y}_{\text{Euler}}(t_{i+1}) := \tilde{y}(t_i) + (t_{i+1} - t_i)k_1, \quad \tilde{y}_{\text{Heun}}(t_{i+1}) := \tilde{y}(t_i) + \frac{t_{i+1} - t_i}{2}(k_1 + k_2)$$

## 7 Gewöhnliche Differentialgleichungen

sowohl die schlechtere Näherung des Euler-Verfahrens als auch die des Heun-Verfahrens berechnen, ohne zusätzliche — potentiell rechenintensive — Auswertungen der rechten Seite  $f$  zu benötigen.

Während bei Anwendungen, in denen die Lösung über lange Zeitintervall fast linear verläuft, die adaptive Schrittweitensteuerung die Rechenzeit drastisch verkürzen kann, ist es häufig schwierig, die Zuverlässigkeit des geschätzten Fehlers zu beweisen. Wir gewinnen also Geschwindigkeit potentiell auf Kosten der Zuverlässigkeit.

Es lässt sich beweisen, dass ein explizites Runge-Kutta-Verfahren, dessen Fehler mit der  $m$ -ten Potenz der Schrittweite sinkt, mindestens  $m$ -stufig sein muss, dass also jeder Schritt des Verfahrens mindestens  $m$  Auswertungen der Funktion  $f$  erfordert. Dadurch werden Verfahren höherer Ordnung relativ aufwendig und zeitintensiv.

Einen sehr eleganten Ausweg bieten *Mehrschrittverfahren*, bei denen nicht nur die Näherungslösung im aktuellen Zeitpunkt  $\tilde{y}(t_i)$  verwendet wird, sondern auch Näherungen zu früheren Zeitpunkten. Das klassische *Adams-Bashforth-Verfahren* kann mit Hilfe der Integralformulierung hergeleitet werden: Es gilt

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(s, y(s)) ds.$$

Unser Interesse gilt der Funktion

$$g: [a, b] \rightarrow \mathbb{R}^n, \quad s \mapsto f(s, y(s)) = y'(s).$$

Wir wählen  $m \in \mathbb{N}$ , nehmen  $i \geq m$  an, und interpolieren  $g$  in den  $m + 1$  Punkten  $t_i, t_{i-1}, \dots, t_{i-m}$ . Das resultierende Interpolationspolynom  $p \in \Pi_m$  können wir mit den zu den Interpolationspunkten gehörenden Lagrange-Polynomen  $\ell_{i,0}, \dots, \ell_{i,m}$  in der Form

$$p = \sum_{j=0}^m g(t_{i-j}) \ell_{i,j} = \sum_{j=0}^m f(t_{i-j}, y(t_{i-j})) \ell_{i,j}$$

darstellen. Wenn wir  $g$  in unserem Integral durch  $p$  ersetzen, erhalten wir

$$\begin{aligned} y(t_{i+1}) &\approx y(t_i) + \int_{t_i}^{t_{i+1}} p(s) ds \\ &= y(t_i) + \int_{t_i}^{t_{i+1}} \sum_{j=0}^m f(t_{i-j}, y(t_{i-j})) \ell_{i,j}(s) ds \\ &= y(t_i) + \sum_{j=0}^m f(t_{i-j}, y(t_{i-j})) \int_{t_i}^{t_{i+1}} \ell_{i,j}(s) ds. \end{aligned}$$

Die Integrale der Lagrange-Polynome können wir im Vorfeld berechnen und in Konstanten

$$b_{i,j} := \int_{t_i}^{t_{i+1}} \ell_{i,j}(s) ds \quad \text{für alle } j \in [0 : m]$$



speichern, so dass wir zu

$$y(t_{i+1}) \approx y(t_i) + \sum_{j=0}^m b_{i,j} f(t_{i-j}, y(t_{i-j}))$$

gelangen. Falls wir nun annehmen, dass die Schrittweite unseres Verfahrens konstant ist, dass also  $t_{i+1} - t_i$  für alle  $i \in [0 : k - 1]$  denselben Wert besitzt, lässt sich nachprüfen, dass  $b_{i,j}$  nur von  $j$  abhängt, nicht aber von  $i$ , so dass wir nur  $m + 1$  Konstanten berechnen und abspeichern müssen.

Um zu einem numerischen Näherungsverfahren zu gelangen, ersetzen wir die Werte der Lösung  $y$  durch Approximationen und gelangen zu dem Adams-Bashforth-Verfahren:

$$\begin{aligned} f_i &:= f(t_i, y(t_i)) && \text{für alle } i \in [0 : m], \\ \tilde{y}(t_{i+1}) &:= \tilde{y}(t_i) + \sum_{j=0}^m b_{i,j} f_{i-j}, \\ f_{i+1} &:= f(t_{i+1}, \tilde{y}(t_{i+1})) && \text{für alle } i \in [m : k - 1]. \end{aligned}$$

Der Vorteil dieses Verfahrens ist offensichtlich: Wir können mit dem Parameter  $m$  die Ordnung fast beliebig festlegen, und da wir die Werte  $f_i$  abspeichern, muss in jedem Schritt die Funktion nur einmal ausgewertet werden. Tatsächlich brauchen wir nicht alle  $f_i$  abspeichern, wir benötigen lediglich die letzten  $m + 1$  von ihnen, so dass wir beispielsweise  $f_{i-m}$  mit  $f_{i+1}$  überschreiben können.

Auch dieses Verfahren hat allerdings Nachteile: Einerseits müssen uns neben dem Anfangswert  $y(a)$  auch die Werte  $y(t_i)$  für  $i \in [0 : m]$  vorliegen, gegebenenfalls in genäherter Form. Andererseits lässt sich die aufwendige Neuberechnung der Konstanten  $b_{i,j}$  in jedem Schritt nur vermeiden, wenn wir mit einer festen Schrittweite arbeiten, also auf eine adaptive Schrittweitensteuerung verzichten.



# Index

- Aitken-Rekurrenz, 88
- Ausgleichsproblem, 43
  
- Bisektionsverfahren, 50
- Bubblesort, 7
  
- Cauchy-Folge, 55
- Cauchy-Schwarz-Ungleichung, 68
- Crank-Nicolson-Verfahren, 117
  
- Diskretisierung, 16
- Divide and Conquer, 7
- Dividierte Differenzen, 92
- Dreiecksmatrix, 17
  
- Eigenvektor, 64
- Eigenwert, 64
- Euler-Verfahren
  - explizit, 115
  - implizit, 116
  
- FFT, 12
- Fixpunkt, 52
- Fixpunktsatz von Banach, 53
  
- Givens-Rotation, 36
  
- Hauptsatz der Integral- und Differentialrechnung, 59
- Hessenberg-Form, 82
- Heun-Verfahren, 117
  
- Interpolationsfehler, 93
- Interpolatorische Quadratur, 100
- Inverse Iteration, 69
- Iteration, 52
- Iterationsverfahren, 51
  
- Konditionszahl, 28
  
- Kontraktion, 54
- Konvergenz, quadratisch, 58
  
- Lagrange-Polynome, 86
- Lebesgue-Konstante, 95
- LR-Zerlegung, 20
- LR-Zerlegung mit Pivotsuche, 24
  
- Maximumnorm, 94
- Mehrschrittverfahren, 120
- Mittelpunktregel, 98
- Mittelwertsatz
  - Differentialrechnung, 52
  
- Neumannsche Reihe, 29
- Newton-Cotes-Quadratur, 101
- Newton-Polynome, 90
- Newton-Richtung, 61
- Newton-Verfahren, 56, 60
- Newton-Verfahren, Konvergenz, 57, 60
- Normalengleichung, 44
  
- orthogonale Iteration, 76
  
- Permutation, 23
- Permutationsmatrix, 23
- Picard-Iteration, 114
- Polynominterpolation, 86
  
- QR-Iteration, 81
- QR-Zerlegung, 37
  - dünn, 76
- Quadratur, interpolatorisch, 100
- Quadraturformel, 98
- Quadraturformel, exakt, 104
- Quadraturformel, induziert, 106
- Quadraturformel, summiert, 107
  
- Rayleigh-Iteration, 71

## INDEX

- Rayleigh-Quotient, 68
- Rayleigh-Ritz-Matrix, 76
- Rechteckregel, 115
- Rückwärtsanalyse, 32
- Rückwärtseinsetzen, 19
- Runge-Kutta-Verfahren, 118
- Runge-Verfahren, 118
  
- Satz von Taylor, 56
- Schrittweitensteuerung, 119
- Simpson-Regel, 101
- Sortieren durch Einfügen, 6
- Spektralnorm, 29
- Stabilität, Interpolation, 95
- Stabilität, Quadratur, 104
- Stützstellenpolynom, 94
  
- Trapezregel, 101
- Tridiagonalmatrix, 83
- Tschebyscheff-Interpolation, 95
  
- Vektoriteration, 64
- Vorwärtseinsetzen, 19
  
- Wellengleichung, 63
  
- Zwischenwertsatz, 50

# Literaturverzeichnis

- [1] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer, 2006.
- [2] T. J. Rivlin. *The Chebyshev Polynomials*. Wiley-Interscience, New York, 1990.
- [3] J. Stoer. *Einführung in die Numerische Mathematik I*. Springer, 5th edition edition, 1989.