

NUMERISCHE METHODEN

Vorlesungsskript zur Veranstaltung

Numerische Methoden

gehalten von

Daniel Kressner

im FS 2010 an der ETH Zürich.

27. Mai 2010

- Dieses Skript basiert teilweise auf dem Vorlesungsskript der von *Prof. Dr. Christoph Schwab* gehaltenen Veranstaltung *Numerische Methoden* im FS 2007 an der ETH Zürich.
- Zur Vorbereitung des vorliegenden Skriptes wurden die folgenden Quellen benutzt:
 1. Dahmen, Wolfgang; Reusken, Arnold. Numerik für Ingenieure und Naturwissenschaftler. Springer, 2005.
 2. Deuffhard, Peter; Hohmann, Andreas. Numerische Mathematik. I. Eine algorithmisch orientierte Einführung. Walter de Gruyter & Co., Berlin, 1993.
 3. Golub, Gene H.; Van Loan, Charles F. Matrix computations. Third edition. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 1996.
 4. Higham, Nicholas J. Accuracy and stability of numerical algorithms. Second edition. SIAM, 2002.
 5. Hiptmair, Ralf. Numerik für CSE. Vorlesungsskript, ETH Zürich, 2007.
 6. Quarteroni, Alfio; Sacco, Riccardo; Saleri, Fausto. Numerische Mathematik 1 und 2. Springer, 2002.

Diese Liste ist *nicht* als Literaturempfehlung zur Prüfungsvorbereitung zu verstehen.

- Mit * gekennzeichnete Abschnitte enthalten Ergänzungsmaterial, das *nicht* Bestandteil der Prüfung sein wird.
- Dank an Lukas Wampfler und den Studenten der FS 2008/2010 für die zahlreichen Korrekturvorschläge. Weitere Korrekturvorschläge bitte unter <http://elbanet.ethz.ch/wikifarm/dkressner/index.php> eintragen oder direkt an

`kressner@math.ethz.ch`.

Inhaltsverzeichnis

0	Lineare Algebra*	1
0.1	Vektorräume	1
0.2	Matrizen	3
0.3	Matrixoperationen	5
0.4	Spezielle Matrizen	8
0.5	Eigenwerte und Eigenvektoren	9
0.6	Vektornormen	11
0.7	Matrixnormen	14
0.8	Orthogonalität	17
0.8.1	Orthogonales Komplement und Projektoren	18
0.8.2	Gram-Schmidt-Algorithmus	19
0.9	Jordan- und Schur-Normalform	21
0.10	Singulärwertzerlegung (SVD)	23
1	Computerarithmetik	27
1.1	Zahldarstellungen und Gleitpunktzahlen	29
1.2	Gleitpunktzahlensysteme	30
1.3	Runden	32
1.4	Rundungsfehleranalyse	34
1.5	Auslöschung	36
1.6	Landau-Symbole (Wiederholung)	38
2	Direkte Lösung Linearer Gleichungssysteme	41
2.1	Dreiecksmatrizen	41
2.2	Gauss'scher Algorithmus und LR -Zerlegung	43
2.3	Fehleranalyse	50
2.4	Störungsanalyse und Kondition	52
2.4.1	Kondition einer Funktion	53
2.4.2	Kondition einer Matrix	54
2.4.3	Störungsanalyse linearer Gleichungssysteme	55
2.4.4	Konvergente Folgen und Reihen von Matrizen	56
2.5	Pivotstrategien	59
2.6	Nachiteration	65
2.7	Cholesky-Zerlegung für SPD Matrizen	66
2.8	LR -Zerlegung für Band-Matrizen	69
3	Polynominterpolation	73
3.1	Lagrange-Interpolation	75
3.2	Neville's Schema	77
3.3	Newton-Interpolation	79
3.4	Hermite-Interpolation	81
3.5	Approximation und Kondition der Interpolation	83
3.6	Tschebyscheff-Interpolation	86
3.7	Spline-Interpolation	88
3.8	Bézier-Techniken*	93
4	Trigonometrische Interpolation	95
4.1	Fourier-Reihen*	95
4.2	Die trigonometrische Interpolationsaufgabe	97
4.3	Die schnelle Fourier-Transformation (FFT)	98
5	Numerische Integration	103
5.1	Interpolatorische Quadraturformeln	103
5.1.1	Geschlossene Newton-Cotes-Formeln	104
5.1.2	Offene Newton-Cotes-Formeln	107
5.1.3	Weitere Quadraturformeln*	107
5.2	Summierte Quadraturformeln	108
5.3	Das Rombergsche Integrationsverfahren	110
5.4	Gauss'sche Quadraturformeln	112
5.5	Verschiedenes*	117
5.5.1	Periodische Funktionen	117
5.5.2	Singuläre Integrale	119
5.5.3	Zweidimensionale Integrale	119
6	Lösung nichtlinearer Gleichungssysteme	121
6.1	Allgemeines zu Iterationsverfahren	121
6.2	Fixpunktiteration im Eindimensionalen	122
6.3	Fixpunktverfahren im Mehrdimensionalen	125
6.4	Nullstellen von Funktionen im Eindimensionalen	127
6.4.1	Bisektionsverfahren	128
6.4.2	Newton-Verfahren	128
6.4.3	Sekantenverfahren	130
6.4.4	Kondition von Nullstellen	132
6.5	Newton-Verfahren im Mehrdimensionalen	133
6.5.1	Grundlagen	133
6.5.2	Konvergenz und Abbruchkriterien	135
6.5.3	Globalisierungstechniken	138
6.5.4	Quasi-Newton-Verfahren*	139
7	Ausgleichsrechnung	141
7.1	Motivation: Gauss'sche Methode der kleinsten Quadrate	141
7.2	Normalgleichungen	143
7.3	Methode der Orthogonalisierung	146
7.4	Gram-Schmidt und modifizierter Gram-Schmidt	147
7.5	Householder-basierte QR -Zerlegung	150
7.5.1	Konstruktion	150
7.5.2	Anwendung auf Ausgleichsprobleme	155
7.6	Rechteckmatrizen mit Rangdefekt. Singulärwertzerlegung.	156

7.7	Niedrigrangapproximation*	157
8	Iterative Lösung Linearer Gleichungssysteme	159
8.1	Speicherung dicht- und dünnbesetzter Matrizen	159
8.2	Allgemeines zu Splitting-Verfahren	161
8.3	Jacobi- und Gauss-Seidel Verfahren	162
8.4	Relaxationsverfahren. JOR und SOR-Verfahren	165
8.5	Richardson-Verfahren*	167
8.6	Das CG-Verfahren	168
8.6.1	Minimierung in affinen Unterräumen	169
8.6.2	Konstruktion der A -orthogonalen Basis	170
8.6.3	Der Algorithmus	172
8.6.4	Geometrische Interpretation*	174
8.6.5	Vorkonditionierung*	174
8.7	Numerisches Beispiel	176
9	Eigenwertprobleme	181
9.1	Eigenwertabschätzungen*	182
9.2	Die Potenzmethode	184
9.3	Inverse Iteration	187
9.4	Krylovraum-Verfahren	189
9.4.1	Arnoldi-Verfahren	189
9.4.2	Extraktion von Ritz-Paaren	191
9.4.3	Lanczos-Verfahren	191
9.4.4	Konvergenzschranken*	192
9.4.5	Numerische Beispiele	194
Index		198

Kapitel 0

Lineare Algebra*

In diesem Kapitel rekapitulieren wir grundlegende Begriffe der linearen Algebra, die für die Analyse numerischer Methoden wichtig sind. Im folgenden verwenden wir die Konvention, dass Vektoren mit unterstrichenen Kleinbuchstaben ($\underline{v}, \underline{w}$) und Matrizen mit fettgestellten Grossbuchstaben (\mathbf{A}, \mathbf{B}) bezeichnet werden.

0.1 Vektorräume

Definition 0.1 Ein Vektorraum über einem Zahlkörper $\mathbb{K} (= \mathbb{R}, \mathbb{C})$ ist eine Menge $V \neq \emptyset$ versehen mit einer **Addition** “+”: $V \times V \rightarrow V$ (kommutativ + assoziativ) und einer **Skalarmultiplikation** “·”: $\mathbb{K} \times V \rightarrow V$, welche zusätzlich die folgenden Eigenschaften erfüllen:

- i. $\exists \underline{0} \in V : \forall \underline{v} \in V : \underline{v} + \underline{0} = \underline{v}$, (Nullvektor)
- ii. $\forall \underline{v} \in V : 0 \cdot \underline{v} = \underline{0}, 1 \cdot \underline{v} = \underline{v}$ für $0, 1 \in \mathbb{K}$,
- iii. $\forall \underline{v} \in V : \exists \underline{w} \in V : \underline{v} + \underline{w} = \underline{0}$, (inverses Element)
- iv. $\forall \alpha \in \mathbb{K} : \forall \underline{v}, \underline{w} \in V : \alpha(\underline{v} + \underline{w}) = \alpha\underline{v} + \alpha\underline{w}$,
 $\forall \alpha, \beta \in \mathbb{K} : \forall \underline{v} \in V : (\alpha + \beta)\underline{v} = \alpha\underline{v} + \beta\underline{v}$,
- v. $\forall \alpha, \beta \in \mathbb{K}, \forall \underline{v} \in V : (\alpha\beta)\underline{v} = \alpha(\beta\underline{v})$.

Beispiel 0.2 Abgesehen von \mathbb{R}^n und \mathbb{C}^n gibt es weitere kanonische Beispiele für Vektorräume, die wir im Verlaufe der Vorlesung wiederholt antreffen werden.

1. Die Menge aller *Polynome vom Grad höchstens n*:

$$V = \mathbb{P}_n := \left\{ p_n(x) = \sum_{k=0}^n a_k x^k \right\}.$$

2. Die Menge aller auf einem beschränkten Intervall $[a, b]$ p -mal stetig differenzierbaren Funktionen:

$$V = C^p([a, b]).$$

(Für $p = 0$ ist V die Menge der auf dem Intervall $[a, b]$ stetigen Funktionen.)

3. Die Menge aller auf einem beschränkten Intervall $[a, b]$ *stückweise linearen, stetigen Funktionen*

$$V = \{ f \in C^0([a, b]) : f|_{[x_{i-1}, x_i]} \text{ ist linear} \}$$

bezüglich einer festen Stützstellenmenge x_0, x_1, \dots, x_n mit $a = x_0 < x_1 < \dots < x_n = b$. Hierbei bezeichnet $f|_{[x_{i-1}, x_i]}$ die Einschränkung der Funktion f auf das Intervall $[x_{i-1}, x_i]$.

In allen obigen Beispielen ist V mit der bei Funktionen üblichen Addition und skalaren Multiplikation versehen. \diamond

Definition 0.3 Sei V Vektorraum. Eine Teilmenge $W \subseteq V$ heisst **Teilraum** (oder **Untervektorraum**) von V , wenn W selbst Vektorraum über \mathbb{K} ist (mit der von V induzierten Addition und Skalarmultiplikation).

Zum Nachweis eines Teilraums $W \subseteq V$ mit $W \neq \emptyset$ ist lediglich die Abgeschlossenheit von W bezüglich Addition und Skalarmultiplikation sowohl $\underline{0} \in V$ zu überprüfen. Alle anderen Eigenschaften von Definition 0.1 “erbt” W von V . Damit lassen sich mühelos die folgenden Beispiele zeigen.

Beispiel 0.4

1. Die Menge der stückweise linearen stetigen Funktionen (siehe Beispiel 0.2.3) bildet einen Teilraum von $V = C^0([a, b])$.
2. Die Menge \mathbb{P}_n (Polynome vom Grad höchstens n) bildet einen Teilraum von $C^0(\mathbb{R})$ (stetige Funktionen auf \mathbb{R}).
3. Sei V Vektorraum und $\underline{v}_1, \dots, \underline{v}_n \in V$. Dann ist

$$W := \text{span}\{\underline{v}_1, \dots, \underline{v}_n\} := \{ \underline{w} = \alpha_1 \underline{v}_1 + \dots + \alpha_n \underline{v}_n : \alpha_i \in \mathbb{K} \}$$

Teilraum von V . Wir nennen $\underline{v}_1, \dots, \underline{v}_n$ **erzeugendes System** von W .

4. Seien $W_1, \dots, W_m \subseteq W$ Teilräume eines Vektorraums V . Dann ist

$$S := \{ \underline{s} = \underline{w}_1 + \dots + \underline{w}_m \text{ mit } \underline{w}_i \in W_i \}$$

ebenfalls ein Teilraum von V .

S heisst **direkte Summe** der W_i , symbolisch $S = W_1 \oplus \dots \oplus W_m$, wenn die Zerlegung $\underline{s} = \underline{w}_1 + \dots + \underline{w}_m$ jedes Vektors $\underline{s} \in S$ eindeutig ist. \diamond

Definition 0.5 Sei V Vektorraum. Eine Menge von Vektoren $\{\underline{v}_1, \dots, \underline{v}_m\} \subset V$ heisst **linear unabhängig**, wenn

$$\alpha_1 \underline{v}_1 + \dots + \alpha_m \underline{v}_m = \underline{0} \implies \alpha_1 = \dots = \alpha_m = 0.$$

Eine linear unabhängige Menge $\{\underline{v}_1, \dots, \underline{v}_m\}$ heisst **Basis** von V , wenn zusätzlich $V = \text{span}\{\underline{v}_1, \dots, \underline{v}_m\}$ gilt.

Aus dem Austauschlemma von Steinitz folgt, dass jede Basis von V die gleiche Anzahl von Basiselementen besitzt. Diese Anzahl (m in der obigen Definition) wird als **Dimension** von V mit $\dim(V)$ bezeichnet. Gibt es keine endliche Basis, so wird V als unendlichdimensional bezeichnet.

Beispiel 0.6

1. Die **Einheitsvektoren**

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, e_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

bilden die kanonische Basis des \mathbb{R}^n und \mathbb{C}^n .

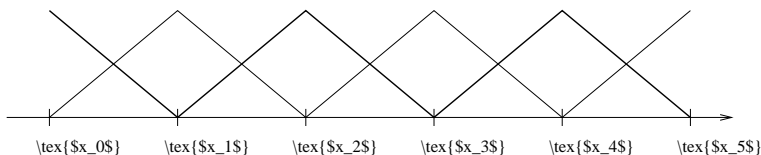
2. Die Monome $1, x, \dots, x^n$ bilden eine Basis von \mathbb{P}_n . Damit folgt $\dim(\mathbb{P}_n) = n + 1$.

3. Sei $V = \{f \in C^0([a, b]) : f|_{[x_{i-1}, x_i]}$ ist linear} mit $a = x_0 < x_1 < \dots < x_n = b$. Dann bilden die **Hutfunktionen**

$$b_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{für } x \in [x_{i-1}, x_i], \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & \text{für } x \in [x_i, x_{i+1}], \\ 0 & \text{sonst,} \end{cases} \quad (i = 1, \dots, n-1)$$

$$b_0(x) = \begin{cases} \frac{x_1-x}{x_1-x_0} & \text{für } x \in [x_0, x_1], \\ 0 & \text{sonst,} \end{cases} \quad b_n(x) = \begin{cases} \frac{x-x_{n-1}}{x_n-x_{n-1}} & \text{für } x \in [x_{n-1}, x_n], \\ 0 & \text{sonst,} \end{cases}$$

eine Basis von V . Es gilt also ebenfalls $\dim(V) = n + 1$. Die Funktionen b_0, b_1, \dots, b_n sind in der folgenden Abbildung für $n = 5$ dargestellt:



4. $C^p([a, b])$ besitzt keine endliche Basis, ist also unendlichdimensional. ◊

0.2 Matrizen

Sei $m, n \in \mathbb{N}$ und \mathbb{K} Körper. Die mn Zahlen $a_{ij} \in \mathbb{K}, i = 1, \dots, m, j = 1, \dots, n$ bilden eine $m \times n$ **Matrix** mit m Zeilen und n Spalten:

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}. \tag{0.1}$$

Falls $a_{ij} \in \mathbb{R}$ schreiben wir $A \in \mathbb{R}^{m \times n}$ und, falls $a_{ij} \in \mathbb{C}$, schreiben wir analog $A \in \mathbb{C}^{m \times n}$. Wir benutzen die Bezeichnungen:

MATLAB

```
% j-ter Spaltenvektor
A(:,j)
% i-ter Zeilenvektor
A(i,:)
% Diagonale
diag(A)
```

Definition 0.7 Sei A eine $m \times n$ Matrix, und i_p, j_q Indizes mit

$$1 \leq i_1 < \dots < i_k \leq m, \quad 1 \leq j_1 < \dots < j_\ell \leq n.$$

Dann heisst $S = (a_{i_p, j_q}), p = 1, \dots, k, q = 1, \dots, \ell$, die zugehörige **Untermatrix** von A .

Die folgende Illustration veranschaulicht den Begriff der Untermatrix:

MATLAB

```
S = A([3,5], [2,4,5])
```

Definition 0.8 Eine $m \times n$ Matrix A wird als **Blockmatrix** bezeichnet, wenn sie wie folgt in Untermatrizen partitioniert ist:

$$A = \begin{pmatrix} A_{11} & \dots & A_{1\ell} \\ \vdots & & \vdots \\ A_{k1} & \dots & A_{k\ell} \end{pmatrix},$$

mit $m_i \times n_j$ Matrizen A_{ij} und $m_1 + \dots + m_k = m, n_1 + \dots + n_\ell = n$.

Beispiel 0.9 $m = n = 3, k = \ell = 2, m_1 = n_1 = 2, m_2 = n_2 = 1$;

$$\left(\begin{array}{cc|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right) = \left(\begin{array}{cc|c} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right).$$

Weitere grundlegende MATLAB-Befehle zur Manipulation von Vektoren und Matrizen sind im folgenden Beispielcode illustriert.

MATLAB

```
v = rand(10,1); % Zufallsvektor der Laenge 10
length(v) % Laenge von v
v(2) % 2-te Komponente von v
```

```

v(2:5)           % Komponenten 2 bis 5 von v
v(1:2:end)      % alle Komponenten mit ungeradem Index
A = rand(10,8); % 10x8 Zufallsmatrix
[m,n] = size(A) % Anzahl Zeilen (m) und Spalten (n) von A
A(1,2)         % Eintrag (1,2) von A
A([3:5],[4:7]) % Untermatrix aus Zeilen 3,...,5
                % und Spalten 4,...,7 von A.
A(:,3) = v     % 3-te Spalte von A mit v ueberschrieben
A(6,:) = 7     % 6-te Zeile von A mit 7 ueberschrieben

```

Bemerkung 0.10 Sowohl MATLAB als auch dieses Skript verwenden die Konvention, dass ein Vektor $\underline{v} \in \mathbb{K}^n$ immer als Spaltenvektor, also als $n \times 1$ Matrix betrachtet wird.

0.3 Matrixoperationen

Wir rekapitulieren die wichtigsten Matrixoperationen.

Definition 0.11 (Matrixmultiplikation) Seien Matrizen $\mathbf{A} = (a_{ij}) \in \mathbb{K}^{m \times n}$ und $\mathbf{B} = (b_{jk}) \in \mathbb{K}^{n \times q}$ gegeben. Dann ist das Produkt der Matrizen, $\mathbf{C} := \mathbf{AB} \in \mathbb{K}^{m \times q}$ definiert durch seine Komponenten $\mathbf{C} = (c_{ik})$ mit

$$c_{ik} = \sum_{j=1}^n a_{ij} b_{jk}. \quad (0.2)$$

Partitioniert man \mathbf{B} in seine Spalten, $\mathbf{B} = (\underline{b}_1, \dots, \underline{b}_q)$ mit $\underline{b}_j \in \mathbb{K}^n$, so lässt sich (0.2) spaltenweise schreiben:

$$\mathbf{C} = \mathbf{AB} = \mathbf{A}(\underline{b}_1, \dots, \underline{b}_q) = (\mathbf{A}\underline{b}_1, \dots, \mathbf{A}\underline{b}_q),$$

das heisst, die Spalte $\underline{c}_j \in \mathbb{K}^m$ in $\mathbf{C} = (\underline{c}_1, \dots, \underline{c}_q)$ ist gegeben durch $\underline{c}_j = \mathbf{A}\underline{b}_j$.

BLAS (Basic Linear Algebra Subroutines)*

Die Speicherhierarchie moderner Rechner reicht von kleinen Speichern (Register, Cache) mit sehr kurzen Zugriffszeiten bis zu grossen Speichern (Hauptspeicher, Festplatte, Netzwerkspeicher) mit vergleichsweise langen Zugriffszeiten. Um diese Hierarchie optimal auszunutzen, sollten so viele Operationen wie möglich mit Daten im schnellen Speicher (Cache) durchgeführt werden, bevor neue Daten aus dem langsameren Speicher (Hauptspeicher) transferiert werden müssen. Eine Implementierung der Matrixmultiplikation nach (0.2) ist aus dieser Sicht unbefriedigend und würde zu inakzeptabel hohen Laufzeiten führen, da für jedes Element c_{ik} auf im Speicher weit auseinanderliegende Teile der Matrizen \mathbf{A} und \mathbf{B} zugegriffen werden muss. Um diesen Effekt zu vermeiden, werden die Matrizen \mathbf{A} und \mathbf{B} in Blockmatrizen partitioniert und die Matrixmultiplikation blockweise durchgeführt. Die Multiplikation der einzelnen Blöcke wird meist in hochoptimierten Assembler-Code implementiert. Viele Hardwarehersteller stellen solche effizienten Implementierungen von Vektor- und Matrixoperationen bereit und richten sich dabei nach dem sogenannten BLAS-Standard, dessen Funktionalität sich in 3 Stufen unterteilt:

BLAS level 1 enthält Vektor-Operationen, z.B. $\underline{y} \leftarrow \alpha \underline{x} + \underline{y}$.

BLAS level 2 enthält Matrix-Vektor-Operationen, z.B. $\underline{y} \leftarrow \alpha \mathbf{A}\underline{x} + \beta \underline{y}$.

BLAS level 3 enthält Matrix-Matrix-Operationen, z.B. $\mathbf{C} \leftarrow \alpha \mathbf{AB} + \beta \mathbf{C}$.

Dieses Modulkonzept erlaubt es numerische Algorithmen effizient zu implementieren, indem der Grossteil des Algorithmus als Vektor- und Matrixoperationen (vorzugsweise level 3) formuliert wird. Die meisten numerischen Programmpakete (MATLAB, Mathematica, ...) basieren auf BLAS.

Implementieren Sie niemals eine Matrixmultiplikation selbst.

Definition 0.12 Eine $n \times n$ Matrix \mathbf{A} heisst **invertierbar** oder **regulär**, wenn es eine $n \times n$ Matrix \mathbf{B} gibt, so dass $\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n$, wobei $\mathbf{I}_n = (\underline{e}_1, \dots, \underline{e}_n)$ die $n \times n$ **Einheitsmatrix** bezeichnet. Eine $n \times n$ Matrix \mathbf{A} heisst **singulär**, wenn sie nicht invertierbar ist.

Die Matrix \mathbf{B} in Definition 0.12 ist eindeutig festgelegt und wird als **Inverse** von \mathbf{A} bezeichnet. Wir schreiben $\mathbf{B} = \mathbf{A}^{-1}$.

Proposition 0.13 Eine Matrix $\mathbf{A} = (\underline{a}_1, \dots, \underline{a}_n)$ ist genau dann invertierbar, wenn ihre Spalten $\underline{a}_1, \dots, \underline{a}_n$ linear unabhängig sind.

Für die Zeilen von \mathbf{A} gilt eine zu Proposition 0.13 analoge Aussage.

Die beiden wichtigsten Rechenregeln bei der Arbeit mit Inversen lauten

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A}, \quad (\mathbf{AC})^{-1} = \mathbf{C}^{-1}\mathbf{A}^{-1}.$$

Definition 0.14 Sei $\mathbf{A} = (a_{ij}) \in \mathbb{K}^{m \times n}$. Dann bezeichnet $\mathbf{A}^T = (a_{ji}) \in \mathbb{K}^{n \times m}$ die **Transponierte** von \mathbf{A} .

Beispiel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}.$$

Einige Rechenregeln für die Arbeit mit der Transponierten:

$$\begin{aligned} (\mathbf{A}^T)^T &= \mathbf{A}, & (\mathbf{A} + \mathbf{B})^T &= \mathbf{A}^T + \mathbf{B}^T, \\ (\mathbf{AB})^T &= \mathbf{B}^T \mathbf{A}^T, & (\alpha \mathbf{A})^T &= \alpha \mathbf{A}^T \quad \forall \alpha \in \mathbb{K}. \end{aligned} \quad (0.3)$$

Ist \mathbf{A} invertierbar, so ist auch \mathbf{A}^T invertierbar und es gilt

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T =: \mathbf{A}^{-T}. \quad (0.4)$$

Für Matrizen mit komplexen Einträgen ist selten die Transponierte im Sinne von Definition 0.14 von Interesse, sondern die folgende Variante, bei der die Einträge zusätzlich konjugiert werden.

Definition 0.15 Sei $\mathbf{A} = (a_{ij}) \in \mathbb{C}^{m \times n}$. Dann bezeichnet $\mathbf{B} = \mathbf{A}^H := \overline{\mathbf{A}}^T = (\overline{a_{ji}}) \in \mathbb{C}^{n \times m}$ die **hermitesch Transponierte** von \mathbf{A} .

Die Rechenregeln (0.3) und (0.4) gelten analog für die hermitesch Transponierte, mit der Ausnahme

$$(\alpha \mathbf{A})^H = \overline{\alpha} \mathbf{A}^H \quad \forall \alpha \in \mathbb{C}.$$

Definition 0.16 Die **Spur** (Englisch: trace) einer $n \times n$ Matrix \mathbf{A} ist die Summe ihrer Diagonalelemente: $\text{spur}(\mathbf{A}) := \sum_{i=1}^n a_{ii}$.

Definition 0.17 Sei $A = (a_{ij})$ eine $n \times n$ Matrix. Dann bezeichnet

$$\det A = \sum_{\sigma \in S_n} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^n a_{i,\sigma(i)} \right)$$

die **Determinante** von A . Hierbei ist S_n die Menge aller Permutationen der Zahlen $\{1, \dots, n\}$ und sgn bezeichnet das Vorzeichen einer solchen Permutation.

Die Determinante erfüllt folgende Rechenregeln:

$$\det(A) = \det(A^T), \quad \det(AB) = \det(A) \det(B), \quad \det(A^{-1}) = 1/\det(A)$$

$$\det(A) = \overline{\det(A^H)}, \quad \det(\alpha A) = \alpha^n \det(A) \quad \forall \alpha \in \mathbb{K}.$$

Desweiteren gilt $\det(A) \neq 0$ genau dann, wenn A invertierbar ist.

```

MATLAB
C = A+B;    % Matrix-Addition
C = A*B;    % Matrix-Multiplikation
C = A.*B;   % ! komponentenweise Multiplikation
B = inv(A); % Inverse von A
B = A';    % hermitesch Transponierte von A
B = A.';   % Transponierte von A
trace(A)   % Spur von A
det(A)     % Determinante von A
    
```

Definition 0.18 Sei A eine $m \times n$ Matrix. Die Mengen

$$\operatorname{im}(A) := \{ \underline{y} \in \mathbb{K}^m : \underline{y} = A\underline{x}, \underline{x} \in \mathbb{K}^n \},$$

$$\operatorname{ker}(A) := \{ \underline{x} \in \mathbb{K}^n : A\underline{x} = \underline{0} \}$$

heissen **Bild** bzw. **Kern** von A .

Bild und Kern sind jeweils Teilräume des \mathbb{K}^m bzw. \mathbb{K}^n .

Definition 0.19 Der **Rang** einer Matrix A ist die Dimension ihres Bildes und wird mit $\operatorname{rang}(A) := \dim(\operatorname{im}(A))$ bezeichnet.

Aus der Definition ergibt sich, dass der Rang der Anzahl der linear unabhängigen Spalten von A entspricht. Weiterhin gelten die folgenden Eigenschaften:

$$\operatorname{rang}(A) = \operatorname{rang}(A^T) = \operatorname{rang}(A^H),$$

$$\operatorname{rang}(A) + \dim(\operatorname{ker}(A)) = n,$$

$$\operatorname{rang}(AB) \leq \operatorname{rang}(A) \operatorname{rang}(B).$$

Aus der letzten Eigenschaft folgt insbesondere, dass die Matrix $A = uv^H \in \mathbb{C}^{n \times n}$ mit Vektoren $u, v \in \mathbb{C}^n$ höchstens Rang 1 hat.

0.4 Spezielle Matrizen

Die numerische lineare Algebra basiert zu grossen Teilen auf Operationen mit Matrizen mit speziellen Eigenschaften. Als einfachstes Beispiel haben wir bereits die Einheitsmatrix I_n kennengelernt. Eine allgemeine $n \times n$ **Diagonalmatrix** bezeichnen wir mit

$$D = \operatorname{diag}(d_{11}, d_{22}, \dots, d_{nn}) := \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_{nn} \end{pmatrix}.$$

Wenn d_{11}, \dots, d_{nn} selber Matrizen sind, so heisst D **Blockdiagonalmatrix**. Eine linke bzw. rechte **Dreiecksmatrix** hat die Form

$$L = \begin{pmatrix} \ell_{11} & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \cdots & \ell_{n,n-1} & \ell_{nn} \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ 0 & \cdots & 0 & r_{nn} \end{pmatrix}.$$

Symbolisch schreiben wir auch

$$D = \diagdown, \quad L = \triangleleft, \quad R = \triangleright.$$

Oft werden linke (rechte) Dreiecksmatrizen auch als untere (obere) Dreiecksmatrizen bezeichnet. Invertierbare linke (rechte) Dreiecksmatrizen bilden eine multiplikative Gruppe.

Proposition 0.20

- i. Sind $A, B \in \mathbb{K}^{n \times n}$ linke (rechte) Dreiecksmatrizen, so ist AB auch eine linke (rechte) Dreiecksmatrix.
- ii. Ist $A \in \mathbb{K}^{n \times n}$ eine invertierbare linke (rechte) Dreiecksmatrix, so ist A^{-1} auch eine linke (rechte) Dreiecksmatrix.

Symbolisch lassen sich obige Aussagen wie folgt veranschaulichen:

$$\triangleleft \cdot \triangleleft = \triangleleft \quad \triangleleft^{-1} = \triangleleft \quad \triangleleft \cdot \triangleleft = \triangleleft \quad \triangleleft^{-1} = \triangleleft.$$

Proposition 0.21 Sei L linke Dreiecksmatrix, R rechte Dreiecksmatrix. Dann gilt

$$\det L = \ell_{11} \cdots \ell_{nn} = \prod_{i=1}^n \ell_{ii}, \quad \det R = r_{11} \cdots r_{nn} = \prod_{i=1}^n r_{ii}.$$

```

MATLAB
eye(n)    % nxn Einheitsmatrix
zeros(m,n) % mxn Matrix mit allen Eintraegen Null
ones(m,n) % mxn Matrix mit allen Eintraegen Eins
diag(v)   % Diagonalmatrix mit den Eintraegen des Vektors v auf
           % der Diagonalen
    
```

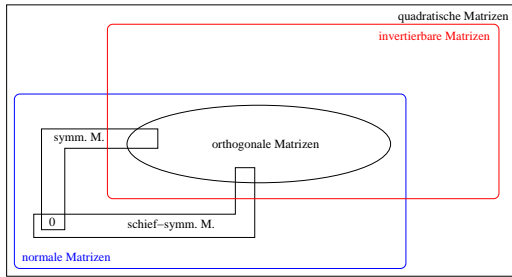
```

tril(A) % linke Dreiecksmatrix mit den Eintraegen von A im
        % unteren Dreieck
triu(A) % rechte Dreiecksmatrix mit den Eintraegen von A im
        % oberen Dreieck
    
```

Abgesehen von Matrizen mit Nulleinträgen gibt es weitere wichtige Klassen von Matrizen, deren Einträge gewisse Beziehungen erfüllen.

Definition 0.22

- $A \in \mathbb{R}^{n \times n}$ heisst **symmetrisch** $\iff A = A^T$,
- schiefsymmetrisch** $\iff A = -A^T$,
- orthogonal** $\iff A^{-1} = A^T \iff A^T A = A A^T = I$.
- $A \in \mathbb{C}^{n \times n}$ heisst **hermitesch** $\iff A^T = \bar{A} \iff A^H = A$,
- unitär** $\iff A^{-1} = A^H \iff A^H A = A A^H = I$,
- normal** $\iff A A^H = A^H A$.



0.5 Eigenwerte und Eigenvektoren

Definition 0.23 Sei $A \in \mathbb{C}^{n \times n}$. Dann ist $\lambda \in \mathbb{C}$ ein **Eigenwert** von A , wenn ein von Null verschiedener Vektor $\underline{x} \in \mathbb{C}^n$ existiert, so dass

$$A \underline{x} = \lambda \underline{x}. \tag{0.5}$$

Der Vektor \underline{x} heisst **Rechtseigenvektor** (meist nur **Eigenvektor**) von A .

Aus Gleichung (0.5) folgt, dass $A - \lambda I_n$ singularär ist, also gilt

$$p_A(\lambda) := \det(A - \lambda I) = 0.$$

Die Funktion $p_A(\lambda)$ ist ein Polynom vom Grad n in λ und heisst **charakteristisches Polynom** von A . Jede Nullstelle von $p_A(\lambda)$ ist ein Eigenwert von A , und umgekehrt. Daraus folgt, dass A genau n Eigenwerte (inklusive ihrer Vielfachheiten) hat. Die Menge

$$\Lambda(A) = \{\lambda \in \mathbb{C} : \lambda \text{ Eigenwert von } A\}$$

heisst **Spektrum** von A .

Proposition 0.24 Seien $\lambda_1, \dots, \lambda_n$ die Eigenwerte einer Matrix $A \in \mathbb{C}^{n \times n}$. Dann gilt:

- i. $\det(A) = p_A(0) = \lambda_1 \lambda_2 \dots \lambda_n$,
- ii. $\text{spur}(A) = \sum_{i=1}^n \lambda_i$,
- iii. $\Lambda(A) = \Lambda(A^T)$,
- iv. $\Lambda(A) = \overline{\Lambda(A^H)}$, d.h. $\lambda \in \Lambda(A) \iff \bar{\lambda} \in \Lambda(A^H)$.

```

MATLAB
eig(A) % Eigenwerte einer Matrix A
[V,D] = eig(A) % V nxn Matrix mit Eigenvektoren und
               % D Diagonalmatrix mit Eigenwerten von A.
               % Es gilt A*V = V*D.
    
```

Der **Spektralradius** von $A \in \mathbb{C}^{n \times n}$ ist der grösste Betrag aller Eigenwerte, also

$$\rho(A) := \max_{\lambda \in \Lambda(A)} |\lambda|.$$

Aus Proposition 0.24 folgt $\rho(A) = \rho(A^H)$. Die Eigenwerte von A^k sind die k -ten Potenzen der Eigenwerte von A , insbesondere gilt

$$\rho(A^k) = (\rho(A))^k, \quad k \in \mathbb{N}.$$

In der Numerik werden Ähnlichkeitstransformationen benutzt, um eine gegebene Matrix auf eine einfachere Gestalt zu bringen. Dabei wird ausgenutzt, dass eine solche Transformation das Spektrum einer Matrix nicht verändert.

Definition 0.25 Sei $A \in \mathbb{K}^{n \times n}$, und sei $C \in \mathbb{K}^{n \times n}$ invertierbar. Dann heisst die Matrix $C^{-1}AC$ **ähnlich** zu A .

Proposition 0.26 Sei $A \in \mathbb{K}^{n \times n}$, und sei $C \in \mathbb{K}^{n \times n}$ invertierbar. Dann gilt $p_{AC^{-1}}(\lambda) = p_{C^{-1}AC}(\lambda)$ und damit $\Lambda(A) = \Lambda(C^{-1}AC)$.

Beweis. Aus den Rechenregeln für die Determinante folgt

$$\begin{aligned}
 p_{C^{-1}AC}(\lambda) &= \det(C^{-1}AC - \lambda \underbrace{C^{-1}C}_I) \\
 &= \det(C^{-1}(A - \lambda I)C) \\
 &= \det(C^{-1}) \det(A - \lambda I) \det(C) \\
 &= \det(C)^{-1} \det(A - \lambda I) \det(C) \\
 &= p_A(\lambda).
 \end{aligned}$$

□

0.6 Vektornormen

Wir kommen jetzt zu den beiden wichtigsten Abschnitten in diesem Kapitel, den Vektor- und Matrixnormen. Erst diese werden es uns erlauben, den Fehler einer numerischen Berechnung sinnvoll zu quantifizieren.

Definition 0.27 Sei V Vektorraum über $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. Eine **Norm** auf V ist eine Abbildung $\|\cdot\| : V \rightarrow \mathbb{R}$ mit den folgenden Eigenschaften:

$$\forall \underline{x} \in V : \|\underline{x}\| \geq 0 \text{ und } \|\underline{x}\| = 0 \Leftrightarrow \underline{x} = 0. \tag{N1}$$

$$\forall \underline{x} \in V, \alpha \in \mathbb{K} : \|\alpha \underline{x}\| = |\alpha| \|\underline{x}\|. \tag{N2}$$

$$\forall \underline{x}, \underline{y} \in V : \|\underline{x} + \underline{y}\| \leq \|\underline{x}\| + \|\underline{y}\|. \tag{N3}$$

(N3) wird auch **Dreiecksungleichung** genannt. Es gilt die folgende Umkehrung.

Proposition 0.28 Sei $\|\cdot\| : V \rightarrow \mathbb{R}$ Norm auf einem Vektorraum V . Dann gilt $|\|\underline{x}\| - \|\underline{y}\|| \leq \|\underline{x} - \underline{y}\|$ für alle $\underline{x}, \underline{y} \in V$.

Beweis. Aus (N3) ergibt sich

$$\begin{aligned} \|\underline{x}\| - \|\underline{y}\| &= \|\underline{x} - \underline{y} + \underline{y}\| - \|\underline{y}\| \\ &\leq \|\underline{x} - \underline{y}\| + \|\underline{y}\| - \|\underline{y}\| = \|\underline{x} - \underline{y}\|. \end{aligned}$$

Durch Vertauschen von \underline{x} und \underline{y} folgt analog $\|\underline{y}\| - \|\underline{x}\| \leq \|\underline{x} - \underline{y}\|$ und damit die Behauptung. \square

Auf $V = \mathbb{R}^n$ und $V = \mathbb{C}^n$ sind in den meisten Anwendungen nur eine Klasse von Normen von Interesse, die sogenannten ℓ^p -Vektornormen $\|\cdot\|_p$ für $1 \leq p \leq \infty$. Für einen Vektor $\underline{x} = (x_1, \dots, x_n)^T \in \mathbb{C}^n$ ist diese wie folgt definiert:

$$\|\underline{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}, \quad 1 \leq p < \infty, \tag{0.6}$$

$$\|\underline{x}\|_\infty := \max\{|x_i| : i = 1, \dots, n\}. \tag{0.7}$$

Der wichtigsten Spezialfälle sind $p = 1$, $p = 2$ (**Euklidische Norm**), und $p = \infty$.

```

MATLAB
norm(v)          % 2-Norm eines Vektors v
norm(v,p)       % p-Norm eines Vektors v
norm(v,inf)     % oo-Norm eines Vektors v
    
```

Satz 0.29 Die in (0.6) und (0.7) für $1 \leq p \leq \infty$ definierte Abbildung $\|\cdot\|_p : \mathbb{C}^n \rightarrow \mathbb{R}$ ist eine Norm auf \mathbb{C}^n .

Beweis von Satz 0.29*

Der Beweis von (N1) und (N2) ist offensichtlich, aber um (N3) zeigen zu können, benötigen wir zunächst einige Hilfsresultate.

Lemma 0.30 (Young'sche Ungleichung) Sei $1 \leq p, q \leq \infty$ mit $\frac{1}{p} + \frac{1}{q} = 1$. Dann gilt für alle $a, b > 0$,

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}. \tag{0.8}$$

Beweis. Die Abbildung $x \mapsto \log x$ ist konkav, d.h.,

$$\forall \alpha, \beta \geq 0, \alpha + \beta = 1 : \forall u, v > 0 : \alpha \log u + \beta \log v \leq \log(\alpha u + \beta v)$$

Setzen wir $\alpha = \frac{1}{p}$, $\beta = \frac{1}{q}$ und $u = a^p$, $v = b^q$, so folgt

$$\log(ab) = \log a + \log b = \alpha \log(a^p) + \beta \log(b^q) \leq \log\left(\frac{1}{p} a^p + \frac{1}{q} b^q\right),$$

und damit (0.8). \square

Satz 0.31 (Hölder'sche Ungleichung) Sei $1 \leq p, q \leq \infty$ mit $\frac{1}{p} + \frac{1}{q} = 1$. Dann gilt für alle $\underline{x}, \underline{y} \in \mathbb{C}^n$,

$$\left| \sum_{i=1}^n x_i \bar{y}_i \right| \leq \|\underline{x}\|_p \|\underline{y}\|_q.$$

Beweis. Die Fälle $\underline{x} = \underline{0}$ und $\underline{y} = \underline{0}$ sind trivial. Seien also $\underline{x} \neq \underline{0}$, $\underline{y} \neq \underline{0}$. Setze

$$\hat{\underline{x}} = \underline{x} / \|\underline{x}\|_p, \quad \hat{\underline{y}} = \underline{y} / \|\underline{y}\|_q.$$

Aus Lemma 0.30 folgt

$$\begin{aligned} |\hat{x}_i| |\hat{y}_i| &\leq \frac{|\hat{x}_i|^p}{p} + \frac{|\hat{y}_i|^q}{q}, \quad i = 1, \dots, n, \\ \sum_{i=1}^n |\hat{x}_i| |\hat{y}_i| &\leq \frac{1}{p} \underbrace{\|\hat{\underline{x}}\|_p^p}_{=1} + \frac{1}{q} \underbrace{\|\hat{\underline{y}}\|_q^q}_{=1} = 1. \end{aligned}$$

\square

Damit können wir nun die Dreiecksungleichung (N3) für die $\|\cdot\|_p$ -Vektornorm und damit Satz 0.29 beweisen.

Satz 0.32 (Minkowski-Ungleichung) Seien $1 \leq p \leq \infty$, $\underline{x}, \underline{y} \in \mathbb{C}^n$. Dann gilt

$$\|\underline{x} + \underline{y}\|_p \leq \|\underline{x}\|_p + \|\underline{y}\|_p.$$

Beweis.

$$\begin{aligned} \|\underline{x} + \underline{y}\|_p^p &= \sum_{i=1}^n |x_i + y_i|^p \leq \sum_{i=1}^n (|x_i| + |y_i|) |x_i + y_i|^{p-1} \\ &\stackrel{\text{Hölder}}{\leq} (\|\underline{x}\|_p + \|\underline{y}\|_p) \left(\sum_{i=1}^n (|x_i + y_i|^{p-1})^q \right)^{\frac{1}{q}} \\ &= (\|\underline{x}\|_p + \|\underline{y}\|_p) \|\underline{x} + \underline{y}\|_p^{p/q} = (\|\underline{x}\|_p + \|\underline{y}\|_p) \|\underline{x} + \underline{y}\|_p^{p-1}, \end{aligned}$$

wobei benutzt wurde, dass $(p-1)q = p$, $\frac{1}{q} = (p-1)/p = 1 - \frac{1}{p}$. \square

Die Euklidische Norm hat die besondere Eigenschaft, dass sie durch das Standard-Skalarprodukt (auch: **Euklidisches Skalarprodukt**)

$$(\underline{x}, \underline{y}) = \underline{y}^H \underline{x} = \sum_{i=1}^n x_i \bar{y}_i$$

auf dem \mathbb{R}^n bzw. \mathbb{C}^n induziert wird. Wir erinnern an die Definition eines Skalar-

produktes.

Definition 0.33 Sei V ein Vektorraum über \mathbb{K} . Eine Abbildung $(\cdot, \cdot) : V \times V \rightarrow \mathbb{K}$ heisst **Skalarprodukt** auf V , falls gilt:

$$\forall \underline{x}, \underline{y}, \underline{z} \in V, \quad \forall \gamma, \lambda \in \mathbb{K} : (\gamma \underline{x} + \lambda \underline{y}, \underline{z}) = \gamma (\underline{x}, \underline{z}) + \lambda (\underline{y}, \underline{z}), \quad (S1)$$

$$(\underline{y}, \underline{x}) = \overline{(\underline{x}, \underline{y})} \quad (S2)$$

$$(\underline{x}, \underline{x}) > 0 \quad \text{für alle } \underline{x} \neq \underline{0}. \quad (S3)$$

Jedes Skalarprodukt auf V induziert eine Norm durch

$$\|\underline{x}\| = \sqrt{(\underline{x}, \underline{x})}.$$

Satz 0.34 (Cauchy-Schwarzsche Ungleichung) Sei V Vektorraum mit Skalarprodukt $(\cdot, \cdot) : V \times V \rightarrow \mathbb{K}$ und der dadurch induzierten Norm $\|\cdot\| : V \rightarrow \mathbb{K}$. Dann gilt für jede $\underline{x}, \underline{y} \in V$:

$$(\underline{x}, \underline{y}) \leq \|\underline{x}\| \|\underline{y}\|.$$

Gleichheit gilt genau dann, wenn \underline{x} und \underline{y} linear abhängig sind.

Beweis. Hier nur der Beweis der Ungleichung für $V = \mathbb{C}^n$ mit dem Euklidischen Skalarprodukt: Entspricht der Hölderschen Ungleichung (Satz 0.31) für den Spezialfall $p = q = 2$. \square

In der Praxis ist die Wahl der geeigneten Norm entscheidend für eine sinnvolle Interpretation des Fehlers. Vom Standpunkt der Theorie hingegen gibt es keinen signifikanten Unterschied zwischen zwei zueinander äquivalenten Normen.

Definition 0.35 Sei V Vektorraum. Zwei Normen $\|\cdot\|_V : V \rightarrow \mathbb{R}$ und $\|\cdot\|_W : V \rightarrow \mathbb{R}$ heissen **äquivalent**, wenn Konstanten $c > 0$ und $C > 0$ existieren, so dass

$$c \|\underline{x}\|_W \leq \|\underline{x}\|_V \leq C \|\underline{x}\|_W, \quad \forall \underline{x} \in V. \quad (0.9)$$

Es ist aus der Analysis bekannt, dass auf einem endlichdimensionalen Vektorraum alle Normen zueinander äquivalent sind, insbesondere sind also alle $\|\cdot\|_p$ -Vektornormen auf \mathbb{C}^n äquivalent. In der Numerik sind wir auch an den entsprechenden Äquivalenzkonstanten c, C in Abhängigkeit von der Dimension n interessiert.

Beispiel 0.36 Für alle $\underline{x} \in \mathbb{C}^n$ und alle $1 \leq p_1 \leq p_2 \leq \infty$ gilt

$$\|\underline{x}\|_{p_2} \leq \|\underline{x}\|_{p_1} \leq n^{\left(\frac{1}{p_1} - \frac{1}{p_2}\right)} \|\underline{x}\|_{p_2}.$$

Als Spezialfälle folgen

$$\begin{aligned} \|\underline{x}\|_2 &\leq \|\underline{x}\|_1 \leq \sqrt{n} \|\underline{x}\|_2, \\ \|\underline{x}\|_\infty &\leq \|\underline{x}\|_2 \leq \sqrt{n} \|\underline{x}\|_\infty, \\ \|\underline{x}\|_\infty &\leq \|\underline{x}\|_1 \leq n \|\underline{x}\|_\infty. \end{aligned}$$

\diamond

Verlust der Normäquivalenz im Unendlichdimensionalen*

Sei $V = C^0([0, 1])$, der Raum aller auf dem Intervall $[0, 1]$ stetigen Funktionen. V ist ein unendlichdimensionaler Vektorraum, auf dem die folgenden beiden Normen definiert werden können:

$$\|f\|_\infty = \sup_{x \in [0, 1]} |f(x)|, \quad \|f\|_2 = \left(\int_0^1 f(x)^2 dx \right)^{1/2}$$

für eine Funktion $f \in C^0([0, 1])$. Betrachte nun

$$f_\theta(x) = \begin{cases} (\theta - x)/\theta & \text{für } x \in [0, \theta], \\ 0 & \text{sonst,} \end{cases}$$

für ein $\theta \in (0, 1]$. Dann ist

$$\|f_\theta\|_\infty = 1, \quad \|f_\theta\|_2 = \sqrt{\frac{\theta}{3}}.$$

Also gibt es keine Konstanten $c, C > 0$ unabhängig von θ , so dass (0.9) für diese beiden Normen erfüllt sein könnte.

0.7 Matrixnormen

Analog zur Länge eines Vektors messen wir auch die Grösse von Matrizen durch Normen, den sogenannten Matrixnormen.

Definition 0.37 Eine Abbildung $\|\cdot\|_M : \mathbb{C}^{m \times n} \rightarrow \mathbb{R}$ heisst **Matrixnorm**, falls sie die folgenden Eigenschaften erfüllt:

$$\|\mathbf{A}\|_M \geq 0 \text{ und } \|\mathbf{A}\|_M = 0 \Leftrightarrow \mathbf{A} = \mathbf{0}, \quad (N1)$$

$$\|\alpha \mathbf{A}\|_M = |\alpha| \|\mathbf{A}\|_M \quad \forall \alpha \in \mathbb{C}, \quad \forall \mathbf{A} \in \mathbb{C}^{m \times n}, \quad (N2)$$

$$\|\mathbf{A} + \mathbf{B}\|_M \leq \|\mathbf{A}\|_M + \|\mathbf{B}\|_M \quad \forall \mathbf{A}, \mathbf{B} \in \mathbb{C}^{m \times n}. \quad (N3)$$

In der Numerik sind meist nur Matrixnormen von Interesse, mit denen sich obere Schranken für die Vektornorm eines Matrix-Vektor-Produkts angeben lassen.

Definition 0.38 Eine Matrixnorm $\|\cdot\|_M$ auf $\mathbb{C}^{m \times n}$ heisst **konsistent** (oder **verträglich**) mit den Vektornormen $\|\cdot\|_V : \mathbb{C}^n \rightarrow \mathbb{R}$ und $\|\cdot\|_W : \mathbb{C}^m \rightarrow \mathbb{R}$, falls

$$\|\mathbf{A}\underline{x}\|_W \leq \|\mathbf{A}\|_M \|\underline{x}\|_V \quad \forall \underline{x} \in \mathbb{C}^n.$$

Beispiel 0.39 Die **Frobeniusnorm** auf $\mathbb{C}^{m \times n}$ ist definiert durch

$$\|\mathbf{A}\|_F := \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{\frac{1}{2}} = \text{tr}(\mathbf{A}\mathbf{A}^H)^{\frac{1}{2}}.$$

$\|\cdot\|_F$ ist konsistent mit der Euklidischen Vektornorm $\|\cdot\|_2$ auf \mathbb{C}^n und \mathbb{C}^m :

$$\begin{aligned} \|\mathbf{A}\underline{x}\|_2^2 &= (\mathbf{A}\underline{x}, \mathbf{A}\underline{x}) = \underline{x}^H \mathbf{A}^H \mathbf{A} \underline{x} = \sum_{i=1}^n \left| \sum_{j=1}^m a_{ij} x_j \right|^2 \\ &\leq \sum_{i=1}^n \left(\sum_{j=1}^m |a_{ij}|^2 \sum_{j=1}^m |x_j|^2 \right) = \|\mathbf{A}\|_F^2 \|\underline{x}\|_2^2, \end{aligned}$$

wobei die Cauchy-Schwarzsche Ungleichung (Satz 0.34) angewandt wurde. \diamond

Definition 0.40 Seien $\|\cdot\|_V, \|\cdot\|_W$ Vektornormen auf \mathbb{C}^n bzw. \mathbb{C}^m . Dann heisst

$$\|\mathbf{A}\|_{VW} := \sup_{\underline{x} \neq 0} \frac{\|\mathbf{A}\underline{x}\|_W}{\|\underline{x}\|_V} = \sup_{\|\underline{x}\|_V=1} \|\mathbf{A}\underline{x}\|_W. \quad (0.10)$$

die durch $\|\cdot\|_V, \|\cdot\|_W$ induzierte Matrixnorm auf $\mathbb{C}^{m \times n}$.

Dass die in (0.10) definierte Funktion $\|\cdot\|_{VW}$ tatsächlich eine Matrixnorm im Sinne von Definition 0.37 ist, folgt aus den Eigenschaften (N1), (N2), (N3) der Vektornorm $\|\cdot\|_W$. Da $\|\mathbf{A}\|_{VW}$ nicht über die Einträge von \mathbf{A} sondern mittels der Operation Matrix-Vektor-Produkt definiert wird, nennt man eine induzierte Norm auch **Operatornorm**. Aus (0.10) folgt sofort, dass $\|\cdot\|_{VW}$ mit den Vektornormen $\|\cdot\|_V$ und $\|\cdot\|_W$ konsistent ist, d.h.

$$\forall \underline{x} \in \mathbb{C}^n : \quad \|\mathbf{A}\underline{x}\|_W \leq \|\mathbf{A}\|_{VW} \|\underline{x}\|_V,$$

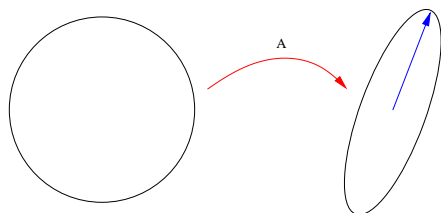
Bemerkung 0.41 Für jede induzierte Norm $\|\cdot\|_{VW}$ hat die Einheitsmatrix \mathbf{I}_n Norm 1. Auf der anderen Seite gilt $\|\mathbf{I}_n\|_F = \sqrt{n}$. Also ist die Frobeniusnorm keine induzierte Norm.

Beispiel 0.42 ($\|\cdot\|_p$ - Matrixnormen) Sei $1 \leq p \leq \infty$. Dann ist die von der $\|\cdot\|_p$ Vektornorm induzierte Matrixnorm definiert durch

$$\|\mathbf{A}\|_p := \max_{\underline{x} \neq 0} \frac{\|\mathbf{A}\underline{x}\|_p}{\|\underline{x}\|_p} = \max_{\|\underline{x}\|_p=1} \|\mathbf{A}\underline{x}\|_p. \quad (0.11)$$

Übung: Zeige, dass die $\|\cdot\|_p$ - Matrixnorm wohldefiniert ist (warum kann man sup durch max in (0.11) ersetzen?).

Die rechte Seite von (0.11) lässt sich für $\mathbf{A} \in \mathbb{R}^{2 \times 2}$ geometrisch gut veranschaulichen. Für $p = 2$ bildet die Menge $\{\underline{x} \in \mathbb{R}^2 : \|\underline{x}\|_2 = 1\}$ den Einheitskreis. Dieser wird von \mathbf{A} auf eine Ellipse abgebildet. Der längere Radius dieser Ellipse ist das Supremum von $\{\|\mathbf{A}\underline{x}\|_2 : \underline{x} \in \mathbb{R}^2, \|\underline{x}\|_2 = 1\}$ und damit die 2-Norm von \mathbf{A} :



Analog bildet für $p \in \{1, \infty\}$ die Matrix $\mathbf{A} \in \mathbb{R}^{2 \times 2}$ ein Quadrat auf ein Parallelogram ab. Nur für $p \in \{1, \infty\}$ lassen sich auch einfache Formeln für die Berechnung von $\|\mathbf{A}\|_p$ angeben:

$$\|\mathbf{A}\|_1 = \max_{j=1, \dots, n} \sum_{i=1}^m |a_{ij}| \quad (\text{Spaltensummennorm})$$

$$\|\mathbf{A}\|_\infty = \max_{i=1, \dots, m} \sum_{j=1}^n |a_{ij}| \quad (\text{Zeilensummennorm})$$

MATLAB	
norm(A,1)	% 1-Norm
norm(A,2)	% 2-Norm
norm(A,inf)	% oo-Norm
norm(A,'fro')	% Frobeniusn.

Eine in der Praxis überaus wichtige Eigenschaft von Matrixnormen ist *Submultiplikativität*.

Definition 0.43 Betrachte eine Klasse von Matrixnormen $\|\cdot\|_M : \mathbb{C}^{k \times l} \rightarrow \mathbb{R}$, die für alle $k, l \geq 1$ definiert ist. Dann heisst $\|\cdot\|_M$ **submultiplikativ**, falls

$$\forall \mathbf{A} \in \mathbb{C}^{m \times n}, \mathbf{B} \in \mathbb{C}^{n \times q} : \|\mathbf{AB}\|_M \leq \|\mathbf{A}\|_M \|\mathbf{B}\|_M.$$

Die meisten, aber nicht alle Matrixnormen sind submultiplikativ.

Beispiel 0.44 Für eine $m \times n$ Matrix \mathbf{A} , sei $\|\mathbf{A}\|_{\max} := \max_{i=1, \dots, m} \max_{j=1, \dots, n} |a_{ij}|$. Seien

$$\mathbf{A} = \mathbf{B} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{AB} = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}.$$

Also haben wir $\|\mathbf{AB}\|_{\max} = 2, \|\mathbf{A}\|_{\max} \|\mathbf{B}\|_{\max} = 1$, und damit ist $\|\cdot\|_{\max}$ nicht submultiplikativ. \diamond

Proposition 0.45 Sei wieder $1 \leq p \leq \infty$. Dann ist die von der $\|\cdot\|_p$ -Vektornorm induzierte Matrixnorm submultiplikativ.

Beweis.

$$\begin{aligned} \|\mathbf{AB}\|_p &= \max_{\underline{x} \neq 0} \frac{\|\mathbf{AB}\underline{x}\|_p}{\|\underline{x}\|_p} = \max_{\substack{\underline{x} \neq 0 \\ \underline{y} \neq 0}} \frac{\|\mathbf{A}\underline{y}\|_p}{\|\underline{y}\|_p} \frac{\|\underline{y}\|_p}{\|\underline{x}\|_p} \\ &\leq \max_{\substack{\underline{x} \neq 0 \\ \underline{y} \neq 0}} \frac{\|\mathbf{A}\underline{y}\|_p}{\|\underline{y}\|_p} \max_{\underline{x} \neq 0} \frac{\|\underline{y}\|_p}{\|\underline{x}\|_p} \leq \|\mathbf{A}\|_p \|\mathbf{B}\|_p. \end{aligned}$$

□

Proposition 0.46 Die Frobeniusnorm $\|\cdot\|_F$ ist submultiplikativ.

Beweis. Sei

$$\mathbf{A} = (\underline{a}_1, \dots, \underline{a}_n) \in \mathbb{C}^{m \times n}, \mathbf{B} = \begin{pmatrix} \underline{b}_1^\top \\ \vdots \\ \underline{b}_n^\top \end{pmatrix} \in \mathbb{C}^{n \times k}.$$

Dann hat das Matrixprodukt \mathbf{AB} folgende Darstellung als Summe von Rang-1 Matrizen, die durch Multiplikation von Spaltenvektoren \underline{a}_i von \mathbf{A} mit Zeilenvektoren \underline{b}_j^\top von \mathbf{B} gewonnen werden:

$$\mathbf{AB} = \underline{a}_1 \underline{b}_1^\top + \dots + \underline{a}_n \underline{b}_n^\top.$$

Anwendung der Dreiecksungleichung (N3) für die Frobeniusnorm ergibt

$$\begin{aligned} \|\mathbf{AB}\|_F &\leq \|\underline{a}_1 \underline{b}_1^T\|_F + \dots + \|\underline{a}_n \underline{b}_n^T\|_F \\ &= \|\underline{a}_1\|_2 \|\underline{b}_1\|_2 + \dots + \|\underline{a}_n\|_2 \|\underline{b}_n\|_2 \\ &\leq \left(\sum_{i=1}^n \|\underline{a}_i\|_2^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^n \|\underline{b}_i\|_2^2\right)^{\frac{1}{2}} \\ &= \|\mathbf{A}\|_F \|\mathbf{B}\|_F. \end{aligned}$$

□

Wie bei den Vektornormen lassen sich auch bei den Matrixnormen gewisse Normäquivalenzen zeigen. Zum Beispiel gilt für $\mathbf{A} \in \mathbb{C}^{m \times n}$:

$$\begin{aligned} \|\mathbf{A}\|_2 &\leq \|\mathbf{A}\|_F \leq \sqrt{\text{rang}(\mathbf{A})} \|\mathbf{A}\|_2, \\ \frac{1}{\sqrt{n}} \|\mathbf{A}\|_\infty &\leq \|\mathbf{A}\|_2 \leq \sqrt{m} \|\mathbf{A}\|_\infty, \\ \frac{1}{\sqrt{m}} \|\mathbf{A}\|_1 &\leq \|\mathbf{A}\|_2 \leq \sqrt{n} \|\mathbf{A}\|_1. \end{aligned}$$

0.8 Orthogonalität

Definition 0.47 Sei V Vektorraum mit Skalarprodukt (\cdot, \cdot) . Zwei Vektoren \underline{x} und \underline{y} heißen **orthogonal** (oder **senkrecht**) bezüglich (\cdot, \cdot) , falls

$$(\underline{x}, \underline{y}) = 0$$

gilt. Symbolisch schreiben wir $\underline{x} \perp \underline{y}$.

Im folgenden werden wir uns der Einfachheit auf den Fall beschränken, dass V endlichdimensional ist. Analog zu Definition 0.47 heisst eine Basis $\{\underline{x}_1, \dots, \underline{x}_n\}$ von V **Orthogonalbasis**, falls

$$\underline{x}_i \perp \underline{x}_j, \quad \forall i \neq j.$$

Gilt zusätzlich $\|\underline{x}_i\| = \sqrt{(\underline{x}_i, \underline{x}_i)} = 1$ für alle i , dann heisst $\{\underline{x}_1, \dots, \underline{x}_n\}$ **Orthonormalbasis**.

Lemma 0.48 Sei $\mathbf{Q} = (\underline{q}_1, \dots, \underline{q}_n) \in \mathbb{C}^{n \times n}$. \mathbf{Q} ist genau dann unitär, wenn die Spalten $\underline{q}_1, \dots, \underline{q}_n$ eine Orthonormalbasis von \mathbb{C}^n bezüglich dem Euklidischen Skalarprodukt bilden.

Beweis. Folgt sofort aus der Beziehung $\mathbf{Q}^H \mathbf{Q} = \mathbf{I}_n$. □

Die für die Numerik wichtigste Eigenschaft unitärer Matrizen ist, dass sie die Euklidische Norm eines Vektors \underline{x} nicht verändern.

Lemma 0.49 Sei $\mathbf{Q} \in \mathbb{C}^{n \times n}$ unitär. Dann gilt $\|\mathbf{Q}\underline{x}\|_2 = \|\underline{x}\|_2$ für alle $\underline{x} \in \mathbb{C}^n$.

Beweis. $\|\mathbf{Q}\underline{x}\|_2 = \sqrt{(\mathbf{Q}\underline{x})^H \mathbf{Q}\underline{x}} = \sqrt{\underline{x}^H \mathbf{Q}^H \mathbf{Q} \underline{x}} = \sqrt{\underline{x}^H \underline{x}} = \|\underline{x}\|_2$. □

Als Korollar folgt, dass unitäre Matrizen die 2- und Frobeniusnorm einer Matrix erhalten.¹

Proposition 0.50 Seien $\mathbf{U} \in \mathbb{C}^{m \times m}$ und $\mathbf{V} \in \mathbb{C}^{n \times n}$ unitär. Dann gilt $\|\mathbf{UA}\|_2 = \|\mathbf{AV}\|_2 = \|\mathbf{A}\|_2$ und $\|\mathbf{UA}\|_F = \|\mathbf{AV}\|_F = \|\mathbf{A}\|_F$ für alle $\mathbf{A} \in \mathbb{C}^{m \times n}$.

Beweis. Das Resultat für die 2-Norm von \mathbf{UA} folgt mit Lemma 0.49 direkt aus der Definition:

$$\|\mathbf{UA}\|_2 = \max_{\|\underline{x}\|_2=1} \|\mathbf{UA}\underline{x}\|_2 = \max_{\|\underline{x}\|_2=1} \|\mathbf{A}\underline{x}\|_2 = \|\mathbf{A}\|_2.$$

Für die Frobeniusnorm partitionieren wir $\mathbf{A} = (\underline{a}_1, \dots, \underline{a}_n)$ und erhalten

$$\|\mathbf{UA}\|_F^2 = \|\mathbf{U}\underline{a}_1\|_2^2 + \dots + \|\mathbf{U}\underline{a}_n\|_2^2 = \|\underline{a}_1\|_2^2 + \dots + \|\underline{a}_n\|_2^2 = \|\mathbf{A}\|_F^2.$$

Der Beweis für \mathbf{AV} erfolgt analog. □

0.8.1 Orthogonales Komplement und Projektoren

Definition 0.51 Sei W Unterraum eines Vektorraums V . Dann bezeichnet

$$W^\perp = \{\underline{y} \in V : \underline{y} \perp \underline{x} \text{ für alle } \underline{x} \in W\}$$

das **orthogonale Komplement** von W (in V).

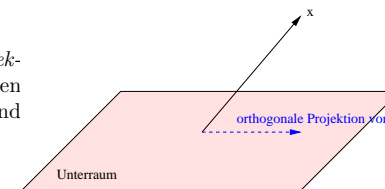
Jeder Vektor $\underline{x} \in V$ lässt sich als Summe von Vektoren des Unterraums und dessen orthogonalem Komplement schreiben:

$$\underline{x} = \underline{x}_W + \underline{x}_{W^\perp}, \quad \underline{x}_W \in W, \quad \underline{x}_{W^\perp} \in W^\perp. \tag{0.12}$$

Diese Zerlegung ist eindeutig (Beweis Übung). Damit lässt sich V als direkte Summe (siehe Beispiel 0.4.4) schreiben:

$$V = W \oplus W^\perp.$$

Der Vektor \underline{x}_W heisst **orthogonale Projektion** von \underline{x} auf W . Dieser hat von allen Vektoren aus W den geringsten Abstand zu \underline{x} .



Lemma 0.52 Sei W Unterraum eines Vektorraums V und sei $\underline{x} \in V$. Dann gilt für die orthogonale Projektion \underline{x}_W von \underline{x} auf W :

$$\|\underline{x} - \underline{x}_W\|_2 = \min_{\underline{w} \in W} \|\underline{x} - \underline{w}\|_2.$$

¹Matrixnormen mit dieser Eigenschaft heissen **unitär invariant**.

Beweis. Aus der Zerlegung (0.12) folgt

$$\|\underline{x} - \underline{w}\|_2^2 = \|(\underline{x}_W - \underline{w}) + \underline{x}_{W^\perp}\|_2^2 = \|\underline{x}_W - \underline{w}\|_2^2 + \|\underline{x}_{W^\perp}\|_2^2 \geq \|\underline{x}_{W^\perp}\|_2^2$$

für alle $w \in W$. Dabei wurde ausgenutzt, dass die $\underline{x}_W - \underline{w} \in W$ und $\underline{x}_{W^\perp} \in W^\perp$ orthogonal zueinander stehen (Satz des Pythagoras). Auf der anderen Seite gilt $\|\underline{x} - \underline{x}_W\|_2 = \|\underline{x}_{W^\perp}\|_2$. Damit minimiert \underline{x}_W den Ausdruck $\|\underline{x} - \underline{w}\|_2$. \square

Um \underline{x}_W zu berechnen, benötigt man lediglich eine Orthonormalbasis für W .

Satz 0.53 Sei W k -dimensionaler Unterraum von \mathbb{C}^n und $\underline{x} \in \mathbb{C}^n$. Bilden die Spalten von $\mathbf{W} \in \mathbb{C}^{n \times k}$ eine Orthonormalbasis für W , so ist $\mathbf{W}\mathbf{W}^H \underline{x}$ die orthogonale Projektion von \underline{x} auf W und $(\mathbf{I}_n - \mathbf{W}\mathbf{W}^H)\underline{x}$ die orthogonale Projektion von \underline{x} auf W^\perp .

Beweis. Für einen Vektor $\underline{w} \in W$ gibt es $\tilde{\underline{w}} \in \mathbb{C}^k$, so dass $\underline{w} = \mathbf{W}\tilde{\underline{w}}$. Also gilt

$$\mathbf{W}\mathbf{W}^H \underline{w} = \mathbf{W} \underbrace{\mathbf{W}^H \mathbf{W}}_{=\mathbf{I}_k} \tilde{\underline{w}} = \underline{w},$$

und damit

$$(\underline{w}, (\mathbf{I}_n - \mathbf{W}\mathbf{W}^H)\underline{x}) = \underline{w}^H (\mathbf{I}_n - \mathbf{W}\mathbf{W}^H)\underline{x} = 0, \quad \forall \underline{w} \in W \Rightarrow (\mathbf{I}_n - \mathbf{W}\mathbf{W}^H)\underline{x} \in W^\perp.$$

Mit $\mathbf{W}\mathbf{W}^H \underline{x} \in \text{im}(\mathbf{W}) = W$ folgt

$$\underline{x} = \mathbf{W}\mathbf{W}^H \underline{x} + (\mathbf{I}_n - \mathbf{W}\mathbf{W}^H)\underline{x},$$

eine Zerlegung von \underline{x} im Sinne von (0.12). \square

0.8.2 Gram-Schmidt-Algorithmus

Seien $k \geq 1$ linear unabhängige Vektoren

$$\underline{x}_1, \dots, \underline{x}_k \in \mathbb{C}^n$$

gegeben. (Die lineare Unabhängigkeit impliziert $k \leq n$.) Der Gram-Schmidt-Algorithmus erzeugt aus diesen Vektoren eine neue Familie von k Vektoren

$$\underline{q}_1, \dots, \underline{q}_k \in \mathbb{C}^n$$

derart, so dass $\|\underline{q}_i\|_2 = 1$ und

$$\text{span}\{\underline{x}_1, \dots, \underline{x}_\ell\} = \text{span}\{\underline{q}_1, \dots, \underline{q}_\ell\}, \quad \forall 1 \leq \ell \leq k. \quad (0.13)$$

Weiterhin sind die \underline{q}_i paarweise orthogonal:

$$\underline{q}_i \perp \underline{q}_j, \quad i \neq j. \quad (0.14)$$

Die Idee des Gram-Schmidt-Algorithmus im ℓ -ten Schritt ist wie folgt: Man erhält den ℓ -ten Vektor \underline{q}_ℓ durch orthogonale Projektion von \underline{x}_ℓ auf das orthogonale Komplement von $\text{im}(\underline{q}_1, \dots, \underline{q}_{\ell-1})$ und Normalisierung:

$$\hat{\underline{q}}_\ell = (\mathbf{I} - (\underline{q}_1, \dots, \underline{q}_{\ell-1})(\underline{q}_1, \dots, \underline{q}_{\ell-1})^H)\underline{x}_\ell, \\ \underline{q}_\ell = \hat{\underline{q}}_\ell / \|\hat{\underline{q}}_\ell\|_2.$$

Dies führt zu dem folgenden Algorithmus.

Algorithmus 0.54 (Gram-Schmidt)

Input: Linear unabhängige Vektoren $\underline{x}_1, \dots, \underline{x}_k \in \mathbb{C}^n$.

Output: Orthonormalbasis

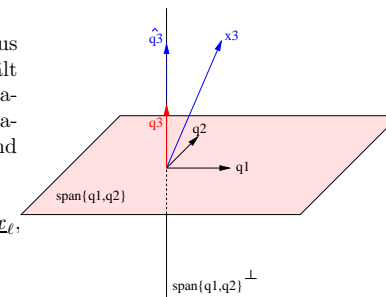
$\underline{q}_1, \dots, \underline{q}_k$, so dass (0.13) erfüllt ist.

for $\ell = 1, \dots, k$ **do**

$$\hat{\underline{q}}_\ell := \underline{x}_\ell - \sum_{j=1}^{\ell-1} (\underline{q}_j, \underline{x}_\ell) \underline{q}_j$$

$$\underline{q}_\ell := \frac{\hat{\underline{q}}_\ell}{\|\hat{\underline{q}}_\ell\|_2}$$

end for



Bemerkung 0.55 Algorithmus 0.54 lässt sich in dieser Form ohne Änderung auf einen beliebigen Vektorraum V übertragen.

Dass die von Algorithmus 0.54 erzeugten Vektoren tatsächlich die gewünschten Eigenschaften erfüllen, lässt sich einfach verifizieren. Die Orthogonalität (0.14) etwa folgt direkt aus der Konstruktion von $\hat{\underline{q}}_\ell$. Numerisch ist Algorithmus 0.54 problematisch; das werden wir später, in Abschnitt 7.4, noch näher diskutieren.

Als eine Anwendung des Gram-Schmidt Algorithmus können wir jede Orthonormalbasis eines Teilraums des \mathbb{C}^n zu einer unitären Matrix ergänzen.

Lemma 0.56 Sei $r < m$. Falls $\mathbf{V}_1 = (\underline{v}_1, \dots, \underline{v}_r) \in \mathbb{C}^{m \times r}$ orthonormale Spalten

hat, d.h. $\underline{v}_i^H \underline{v}_j = \delta_{ij}$ mit dem Kronecker-Symbol $\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$, dann existiert

$\mathbf{V}_2 := (\underline{v}_{r+1}, \dots, \underline{v}_m) \in \mathbb{C}^{m \times (m-r)}$ derart, dass die Matrix

$$\mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2)$$

unitär ist, d.h. $\mathbf{V}^H \mathbf{V} = \mathbf{I}_m$ und $\text{im}(\mathbf{V}_1)^\perp = \text{im}(\mathbf{V}_2)$.

Beweis. Aus dem Basisergänzungssatz folgt die Existenz von Vektoren $\underline{x}_{r+1}, \dots, \underline{x}_m \in \mathbb{C}^m$, so dass

$$\{\underline{v}_1, \dots, \underline{v}_r, \underline{x}_{r+1}, \dots, \underline{x}_m\}$$

eine Basis des \mathbb{C}^m bilden. Der Gram-Schmidt-Algorithmus reproduziert in den ersten r Schritten die Vektoren $\underline{v}_1, \dots, \underline{v}_r$ (Beweis Übung). In den letzten $m - r$ Schritten werden Vektoren $\underline{v}_{r+1}, \dots, \underline{v}_m$ erzeugt, die wegen (0.13) und (0.14) die geforderten Eigenschaften erfüllen. \square

0.9 Jordan- und Schur-Normalform

Die Transformation von Matrizen auf einfachere Gestalt ist ein entscheidendes Hilfsmittel in der Theorie und Numerik der linearen Algebra. In diesem Abschnitt rekapitulieren wir zwei Normalformen, die über Ähnlichkeitstransformation $X^{-1}AX$ erreicht werden können. Aus Proposition 0.26 wissen wir bereits, dass eine solche Transformation die Eigenwerte von A nicht verändert.

Die folgende Normalform ist vor allem für die Theorie von Interesse.

Satz 0.57 (Jordan'sche Normalform) Für jede Matrix $A \in \mathbb{C}^{n \times n}$ existiert eine nichtsinguläre Matrix $X \in \mathbb{C}^{(n \times n)}$ derart, dass

$$X^{-1}AX = \text{diag}(J_1, \dots, J_t)$$

die sogenannte **Jordan'sche Normalform** von A ist, wobei die **Jordan-Blöcke** J_i gegeben sind durch

$$J_i = \begin{pmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots \\ 0 & \cdots & \cdots & 0 & \lambda_i \end{pmatrix} \in \mathbb{C}^{n_i \times n_i}$$

mit $\sum_{i=1}^t n_i = n$ und $\Lambda(A) = \{\lambda_1, \dots, \lambda_t\}$.

Wir bemerken zur Jordan'schen Normalform:

- Die Eigenwerte λ_i sind in den verschiedenen Jordanblöcken nicht notwendig verschieden. Die Anzahl der Jordanblöcke zu einem Eigenwert λ_i ist die **geometrische Vielfachheit** von λ_i und entspricht der Dimension von $\ker(A - \lambda_i I_n)$. Die Summe der Grössen n_i aller Jordanblöcke zu einem Eigenwert λ_i ist die **algebraische Vielfachheit** von λ_i und entspricht der Vielfachheit der Nullstelle λ_i des charakteristischen Polynoms $\det(A - \lambda I_n)$.
- Eine Matrix heisst **diagonalisierbar**, wenn alle Jordanblöcke der Grösse 1 sind, also bei allen Eigenwerten die algebraischen und geometrischen Vielfachheiten identisch sind.
- Aus verschiedenen Gründen lässt sich die Jordan'sche Normalform numerisch nur schwer oder gar nicht berechnen. Zum einen kann eine beliebig kleine Störung der Einträge von A (etwa durch Rundungsfehler) die Jordan-Normalform komplett verändern. Zum anderen kann die Transformationsmatrix X sehr schlecht konditioniert sein, also $\|X\|_2 \|X^{-1}\|_2$ sehr gross sein (siehe Abschnitt 2.4 zum Begriff der Kondition einer Matrix).

In der **Numerik** verlangt man deshalb weniger als die Jordan'sche Normalform und schränkt die zulässigen Transformationen auf unitäre Matrizen ein. Dies wird damit "bezahlt", dass die transformierte Normalform der Matrix weniger Nulleinträge hat.

Satz 0.58 (Schur-Form) Sei $A \in \mathbb{C}^{n \times n}$ und seien $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ die Eigenwerte von A . Dann existiert eine unitäre Matrix $Q \in \mathbb{C}^{n \times n}$, so dass

$$Q^H A Q = \begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} =: T,$$

d.h., T ist rechte Dreiecksmatrix, mit den Eigenwerten von A auf der Diagonalen.

Beweis. Der Beweis erfolgt per Induktion über n . Für $n = 1$ ist die Aussage mit $Q = 1$ offensichtlich erfüllt. Sei die Aussage für alle Matrizen $B \in \mathbb{C}^{(n-1) \times (n-1)}$ erfüllt. Betrachte nun für $A \in \mathbb{C}^{n \times n}$ den Eigenvektor \underline{u}_1 zu λ_1 . O.B.d.A. können wir $\|\underline{u}_1\|_2 = 1$ annehmen. Wegen Lemma 0.56 existieren Vektoren $\underline{u}_2, \dots, \underline{u}_n \in \mathbb{C}^n$, so dass $Q_0 = (\underline{u}_1, \dots, \underline{u}_n)$ unitär ist. Wegen $A\underline{u}_1 = \lambda_1 \underline{u}_1$ gilt

$$A Q_0 = Q_0 \begin{pmatrix} \lambda_1 & * \\ \underline{0} & A_1 \end{pmatrix}.$$

Aus der Induktionsvoraussetzung folgt die Existenz einer unitären Matrix $Q_1 \in \mathbb{C}^{(n-1) \times (n-1)}$, so dass $T_1 = Q_1^H A_1 Q_1$ rechte Dreiecksmatrix ist, mit $\lambda_2, \dots, \lambda_n$ auf der Diagonalen. Mit

$$Q = Q_0 \begin{pmatrix} 1 & \underline{0} \\ \underline{0} & Q_1 \end{pmatrix}$$

folgt die Behauptung. \square

Bemerkung 0.59 Da die Eigenwerte einer reellen Matrix komplex sein können, gibt es für $A \in \mathbb{R}^{n \times n}$ i.a. keine reelle orthogonale Matrix Q , so dass $Q^T A Q$ in rechter Dreiecksform ist. MATLAB gibt für reelles A die reelle Schur-Form zurück, bei der T nur fast rechte Dreiecksform hat, mit 2×2 -Blöcken auf der Diagonalen.

<code>[Q,T] = schur(A)</code>	MATLAB	<code>% (reelle) Schur-Form von A</code>
<code>[Q,T] = schur(A,'complex')</code>		<code>% komplexe Schur-Form von A</code>
		<code>% (selbst wenn A reell ist)</code>

Korollar 0.60 (Spektralzerlegung) Ist $A \in \mathbb{C}^{n \times n}$ eine normale Matrix, dann gibt es eine unitäre Matrix Q , so dass $T = Q^H A Q$ diagonal ist.

Beweis. Aus der Normalität von A folgt, dass die Schur-Form T von A die Beziehung $T T^H = T^H T$ erfüllt. Mit der Partitionierung $T = \begin{pmatrix} \lambda_1 & \underline{w}^H \\ \underline{0} & T_1 \end{pmatrix}$ folgt daraus $|\lambda_1|^2 + \|\underline{w}\|_2^2 = |\lambda_1|^2$ und damit $\underline{w} = \underline{0}$. Wiederholtes Anwenden dieses Arguments auf T_1 impliziert die Diagonalität von T . \square

Die Umkehrung von Korollar 0.60 gilt auch: Jede unitär diagonalisierbare Matrix ist normal.

0.10 Singulärwertzerlegung (SVD)

Die **SVD** einer (nicht notwendig quadratischen!) Matrix \mathbf{A} ist die mit Abstand wichtigste Matrixzerlegung in Anwendungen der (numerischen) linearen Algebra. Dementsprechend firmiert die SVD – je nach Anwendung – unter weiteren Bezeichnungen und Akronymen, z.B.: PCA – Principal Component Analysis, LSI – Latent Semantic Indexing, POD – Proper Orthogonal Decomposition.

Satz 0.61 (Singulärwertzerlegung) Sei $\mathbf{A} \in \mathbb{C}^{m \times n}$, $r = \text{rang}(\mathbf{A}) \leq \min\{m, n\}$. Dann existieren reelle Werte $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ sowie unitäre Matrizen $\mathbf{U} \in \mathbb{C}^{m \times m}$ und $\mathbf{V} \in \mathbb{C}^{n \times n}$, so dass

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^H, \tag{0.15}$$

wobei

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \mathbf{0} \\ & & \sigma_r & \\ \mathbf{0} & & & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{m \times n}.$$

Beweis. Für $\mathbf{A} = \mathbf{0}$ ist der Satz offensichtlich mit $\mathbf{U} = \mathbf{I}_m, \mathbf{V} = \mathbf{I}_n$ erfüllt. O.b.d.A. können wir also $\mathbf{A} \neq \mathbf{0}$ annehmen.

Wir erinnern an die Definition der 2-Norm von \mathbf{A} :

$$\|\mathbf{A}\|_2 = \max_{\|\underline{x}\|_2=1} \|\mathbf{A}\underline{x}\|_2.$$

Also existiert ein Vektor $\underline{v}_1 \in \mathbb{C}^n$ mit $\|\underline{v}_1\|_2 = 1$, so dass $\|\mathbf{A}\underline{v}_1\|_2 = \|\mathbf{A}\|_2$. Da $\sigma_1 := \|\mathbf{A}\|_2 \neq 0$ folgt mit $\underline{u}_1 := \mathbf{A}\underline{v}_1/\sigma_1$ die Beziehung

$$\underline{u}_1^H \mathbf{A} \underline{v}_1 = \sigma_1.$$

Nach Lemma 0.56 existieren $\mathbf{U}_2 \in \mathbb{C}^{m \times (m-1)}, \mathbf{V}_2 \in \mathbb{C}^{n \times (n-1)}$ derart, dass die Matrizen

$$\mathbf{U} = (\underline{u}_1, \mathbf{U}_2) \in \mathbb{C}^{m \times m}, \quad \mathbf{V} = (\underline{v}_1, \mathbf{V}_2) \in \mathbb{C}^{n \times n}$$

unitär sind. Weiterhin gilt

$$\begin{aligned} \mathbf{U}^H \mathbf{A} \mathbf{V} &= (\underline{u}_1, \mathbf{U}_2)^H \mathbf{A} (\underline{v}_1, \mathbf{V}_2) \\ &= \left(\begin{array}{c|c} \underline{u}_1^H \mathbf{A} \underline{v}_1 & \underline{u}_1^H \mathbf{A} \mathbf{V}_2 \\ \hline \mathbf{U}_2^H \mathbf{A} \underline{v}_1 & \mathbf{U}_2^H \mathbf{A} \mathbf{V}_2 \end{array} \right) =: \left(\begin{array}{c|c} \sigma_1 & \underline{w}^H \\ \hline \mathbf{0} & \mathbf{B} \end{array} \right) =: \mathbf{A}_1. \end{aligned}$$

Da (nachrechnen!)

$$\left\| \mathbf{A}_1 \begin{pmatrix} \sigma_1 \\ \underline{w} \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \sigma_1^2 + \underline{w}^H \underline{w} \\ \mathbf{B} \underline{w} \end{pmatrix} \right\|_2^2 = (\sigma_1^2 + \underline{w}^H \underline{w})^2 + \|\mathbf{B} \underline{w}\|_2^2 \geq (\sigma_1^2 + \underline{w}^H \underline{w})^2,$$

gilt auf der einen Seite

$$\|\mathbf{A}_1\|_2^2 = \sup_{\underline{0} \neq \underline{z} \in \mathbb{C}^n} \frac{\|\mathbf{A}_1 \underline{z}\|_2^2}{\|\underline{z}\|_2^2} \geq \frac{\left\| \mathbf{A}_1 \begin{pmatrix} \sigma_1 \\ \underline{w} \end{pmatrix} \right\|_2^2}{\left\| \begin{pmatrix} \sigma_1 \\ \underline{w} \end{pmatrix} \right\|_2^2} \geq \frac{(\sigma_1^2 + \underline{w}^H \underline{w})^2}{\sigma_1^2 + \underline{w}^H \underline{w}} = \sigma_1^2 + \underline{w}^H \underline{w}. \tag{0.16}$$

Wegen der Invarianz der 2-Norm unter unitären Transformationen (siehe Proposition 0.50) gilt auf der anderen Seite $\sigma_1 = \|\mathbf{A}\|_2 = \|\mathbf{U}^H \mathbf{A} \mathbf{V}\|_2 = \|\mathbf{A}_1\|_2$. Zusammen mit (0.16) folgt

$$\sigma_1^2 = \|\mathbf{A}_1\|_2^2 \geq \sigma_1^2 + \underline{w}^H \underline{w},$$

also $\|\underline{w}\|_2^2 = \underline{w}^H \underline{w} = 0$ und damit $\underline{w} = \mathbf{0}$. Somit ergibt sich

$$\mathbf{A}_1 = \left(\begin{array}{c|c} \sigma_1 & \underline{0}^T \\ \hline \underline{0} & \mathbf{B} \end{array} \right).$$

Per Induktion (ähnlich wie im Beweis von Satz 0.58) ist damit die Behauptung bewiesen. \square

Man kann zeigen, dass die Werte $\sigma_1, \dots, \sigma_r$ in der SVD eindeutig bestimmt sind. Gilt ausserdem $\sigma_1 > \sigma_2 > \dots > \sigma_r$, sind sogar die Vektoren $\underline{u}_1, \dots, \underline{u}_r, \underline{v}_1, \dots, \underline{v}_r$ bis auf skalare Vielfache vom Betrag 1 eindeutig bestimmt.

Definition 0.62 Betrachte eine SVD (0.15) von $\mathbf{A} \in \mathbb{C}^{m \times n}$ und partitioniere

$$\mathbf{U} = (\underline{u}_1, \dots, \underline{u}_r, \underline{u}_{r+1}, \dots, \underline{u}_m), \quad \mathbf{V} = (\underline{v}_1, \dots, \underline{v}_r, \underline{v}_{r+1}, \dots, \underline{v}_n). \tag{0.17}$$

Dann heissen

- $\sigma_1, \dots, \sigma_r$ die **Singulärwerte**,
- $\underline{u}_1, \dots, \underline{u}_r$ die **linken Singulärvektoren**,
- $\underline{v}_1, \dots, \underline{v}_r$ die **rechten Singulärvektoren**

von \mathbf{A} .

Im Gegensatz zur Schur-Form, können bei reellem $\mathbf{A} \in \mathbb{R}^{n \times n}$ auch die Transformationsmatrizen \mathbf{U}, \mathbf{V} in der SVD reell (und damit orthogonal) gewählt werden.

Bemerkung 0.63 Statt der vollen SVD von \mathbf{A} betrachtet man oft die sogenannte **kompakte SVD** $\mathbf{A} = \mathbf{U}_0 \mathbf{\Sigma}_0 \mathbf{V}_0^H$, die sich aus der SVD mit der Partitionierung (0.17) mittels

$$\begin{aligned} \mathbf{U}_0 &= (\underline{u}_1, \dots, \underline{u}_r), \\ \mathbf{\Sigma}_0 &= \text{diag}(\sigma_1, \dots, \sigma_r), \\ \mathbf{V}_0 &= (\underline{v}_1, \dots, \underline{v}_r) \text{ ergibt.} \end{aligned}$$

```

MATLAB
s = svd(A) % Singulaerwerte von A
[U,D,V] = svd(A) % SVD von A
[U,D,V] = svd(A,0) % kompakte SVD
    
```

Die SVD erlaubt es auf einfache Weise, viele theoretisch und praktisch relevante Aussagen über \mathbf{A} zu treffen. So folgt z.B. sofort, dass \mathbf{A} sich als Summe von r Matrizen vom Rang 1 schreiben lässt:

$$\mathbf{A} = \sigma_1 \underline{u}_1 \underline{u}_1^H + \dots + \sigma_r \underline{u}_r \underline{u}_r^H.$$

Weiterhin können aus \mathbf{U}, \mathbf{V} Orthonormalbasen für das Bild und den Kern von \mathbf{A} bzw. \mathbf{A}^H gewonnen werden.

Proposition 0.64 Betrachte eine SVD von $\mathbf{A} \in \mathbb{C}^{m \times n}$ mit der Partitionierung (0.17).

Dann gilt

$$\begin{aligned}\ker(\mathbf{A}) &= \text{span}\{\underline{v}_{r+1}, \dots, \underline{v}_n\}, \\ \text{im}(\mathbf{A}) &= \text{span}\{\underline{u}_1, \dots, \underline{u}_r\}, \\ \ker(\mathbf{A}^H) &= \text{span}\{\underline{u}_{r+1}, \dots, \underline{u}_m\}, \\ \text{im}(\mathbf{A}^H) &= \text{span}\{\underline{v}_1, \dots, \underline{v}_r\}.\end{aligned}$$

Beweis. Die aus der SVD folgende Beziehungen $\mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$ und $\mathbf{A}^H\mathbf{U} = \mathbf{V}\mathbf{\Sigma}^H$ spaltenweise aufgeschrieben:

$$\begin{aligned}\mathbf{A}\underline{v}_i &= \begin{cases} \sigma_i \underline{u}_i & 1 \leq i \leq r, \\ 0 & r+1 \leq i \leq n, \end{cases} \\ \mathbf{A}^H \underline{u}_i &= \begin{cases} \sigma_i \underline{v}_i & 1 \leq i \leq r, \\ 0 & r+1 \leq i \leq m. \end{cases}\end{aligned}$$

Daraus folgt z.B. sofort, dass $\text{span}\{\underline{v}_{r+1}, \dots, \underline{v}_n\} \subseteq \ker(\mathbf{A})$. Da aber die Dimension des Kerns $n-r$ ist, muss Gleichheit gelten. Die anderen Beziehungen folgen analog. \square

Zwischen den Singulärwerten bzw. -vektoren von \mathbf{A} und den Eigenwerten bzw. -vektoren von $\mathbf{A}^H\mathbf{A}$ und $\mathbf{A}\mathbf{A}^H$ gibt es enge Zusammenhänge. Aus $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$, folgen die Spektralzerlegungen

$$\begin{aligned}\mathbf{U}^H\mathbf{A}\mathbf{A}^H\mathbf{U} &= \mathbf{\Sigma}\mathbf{\Sigma}^H = \text{diag}(\sigma_1^2, \dots, \sigma_r^2, 0, \dots, 0), \\ \mathbf{V}^H\mathbf{A}^H\mathbf{A}\mathbf{V} &= \mathbf{\Sigma}^H\mathbf{\Sigma} = \text{diag}(\sigma_1^2, \dots, \sigma_r^2, 0, \dots, 0).\end{aligned}$$

Insbesondere ist $\lambda \neq 0$ genau dann ein Eigenwert von $\mathbf{A}^H\mathbf{A}$ (oder $\mathbf{A}\mathbf{A}^H$), wenn $\sqrt{\lambda}$ ein Singulärwert von \mathbf{A} ist.

Kapitel 1

Computerarithmetik

Since none of the numbers which we take out from logarithmic and trigonometric tables admit of absolute precision, but all are to a certain extent approximate only, the results of all calculations performed by the aid of these numbers can only be approximately true.

It may happen, that in special cases the effect of the errors of the tables is so augmented that we may be obliged to reject a method, otherwise the best, and substitute another in its place.

— CARL F. GAUSS, *Theoria Motus* (1809)²

MATLAB's creator Dr. Cleve Moler used to advise foreign visitors not to miss the country's two most awesome spectacles: the Grand Canyon, and meetings of IEEE p754.

— WILLIAM M. KAHAN³

Trotz des massiven technologischen Fortschritts des letzten Jahrhunderts hat das Zitat von Gauss nichts an Bedeutung verloren. Nicht nur Logarithmentabellen sondern auch Computer haben endliche Speicherkapazitäten. Eine reelle Zahl lässt sich im allgemeinen nicht beliebig genau im Computer darstellen. Dadurch ergeben sich Rundungsfehler, die sich im Verlauf einer Rechnung verstärken können und – im schlimmsten Fall – das gewünschte Ergebnis bis zur Unkenntlichkeit verfälschen können.

Beispiel 1.1 Die folgende Darstellung der Euler'schen Zahl e ist aus der Analysis bekannt:

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n.$$

Man erwartet also, dass der Ausdruck $e_n = \left(1 + \frac{1}{n}\right)^n$ für grösser werdendes n immer bessere Approximationen von e liefert. Dies wäre in exakter Arithmetik auch tatsächlich zu beobachten. Für den auf dem Computer (also in inexakter Arithmetik) berechneten Wert \hat{e}_n ist dies allerdings nicht mehr der Fall:

```

MATLAB
% Approximation von e
for i=1:15
    n = 10^i; en = (1+1/n)^n;
    fprintf('10^%2d %20.15f ...
           %20.15f\n', ...
           i, en, en-exp(1));
end

```

n	Berechnetes \hat{e}_n	Fehler $\hat{e}_n - e$
10^1	2.593742460100002	-0.124539368359044
10^2	2.704813829421529	-0.013467999037517
10^3	2.716923932235520	-0.001357896223525
10^4	2.718145926824356	-0.000135901634689
10^5	2.718268237197528	-0.000013591261517
10^6	2.718280469156428	-0.000001359302618
10^7	2.718281693980372	-0.000000134478673
10^8	2.718281786395798	-0.000000042063248
10^9	2.718282030814509	0.000000202355464
10^{10}	2.718282053234788	0.000000224775742
10^{11}	2.718282053357110	0.000000224898065
10^{12}	2.718523496037238	0.000241667578192
10^{13}	2.716110034086901	-0.002171794372145
10^{14}	2.716110034087023	-0.002171794372023
10^{15}	3.035035206549262	0.316753378090216

Ab $n = 10^8$ nimmt die Genauigkeit sogar wieder ab, bis bei $n = 10^{15}$ nicht eine Dezimalstelle von e richtig berechnet wird. ◊

Beispiel 1.2 Die Potenzreihe der Exponentialfunktion konvergiert für jedes x :

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \dots$$

Auf dem Rechner können wir natürlich nur eine Partialsumme

$$s_i(x) = \sum_{k=0}^i \frac{x^k}{k!}$$

berechnen. Aufgrund der Restglieddarstellung der Taylor-Entwicklung gibt es ein ξ mit $0 < |\xi| < |x|$, so dass

$$e^x - s_i(x) = \frac{e^\xi x^{i+1}}{(i+1)!}.$$

Wählen wir also als Abbruchkriterium $|x|^{i+1}/(i+1)! \leq \text{tol} \cdot s_i(x)$ für eine (kleine) vorgegebene Toleranz tol , so folgt für negative x

$$|e^x - s_i(x)| \leq \frac{|x|^{i+1}}{(i+1)!} \leq \text{tol} \cdot s_i(x) \approx \text{tol} \cdot e^x.$$

Also ist der relative Fehler $|e^x - s_i(x)|/|e^x|$ approximativ durch tol beschränkt. Für positive x findet man eine ähnliche Schranke über eine genauere Analyse des Restglieds (Übung).

Tabelle 1.1 listet die Ausgabe der MATLAB-Funktion für $\text{tol} = 10^{-8}$. Im Widerspruch zu den obigen theoretischen Aussagen ist der relative Fehler für negative x in Computerarithmetik *nicht* durch tol beschränkt. ◊

Ziel dieses Kapitels ist, die in den Beispielen 1.1 und 1.2 auftretenden Phänomene zu verstehen und zu vermeiden lernen. Dazu ist es notwendig, zunächst einmal zu klären wie eigentlich reelle Zahlen im Computer dargestellt bzw. approximiert werden.

```

MATLAB
function y = expeval(x,tol)
% Approximation von e^x
s=1; k=1;
term=1;
while (abs(term)>tol*abs(s))
    term = term*x/k;
    s = s + term;
    k = k+1;
end
y = s;

```

²Translated and quoted in Higham (2002).

³See <http://www.cs.berkeley.edu/~wkahan/ieee754status/754story.html>.

x	Berechnetes $\hat{s}_t(x)$	$\exp(x)$	$\frac{ \exp(x) - \hat{s}_t(x) }{\exp(x)}$
-20	5.6218844674e-09	2.0611536224e-09	1.727542676201181
-18	1.5385415977e-08	1.5229979745e-08	0.010205938187564
-16	1.1254180496e-07	1.1253517472e-07	0.000058917020257
-14	8.3152907681e-07	8.3152871910e-07	0.000000430176956
-12	6.1442133148e-06	6.1442123533e-06	0.000000156480737
-10	4.5399929556e-05	4.5399929762e-05	0.00000004544414
-8	3.3546262817e-04	3.3546262790e-04	0.00000000788902
-6	2.4787521758e-03	2.4787521767e-03	0.000000000333306
-4	1.8315638879e-02	1.8315638889e-02	0.000000000530694
-2	1.3533528320e-01	1.3533528324e-01	0.000000000273603
0	1.0000000000e+00	1.0000000000e+00	0.000000000000000
2	7.3890560954e+00	7.3890560989e+00	0.00000000479969
4	5.4598149928e+01	5.4598150033e+01	0.00000001923058
6	4.0342879295e+02	4.0342879349e+02	0.00000001344248
8	2.9809579808e+03	2.9809579870e+03	0.00000002102584
10	2.2026465748e+04	2.2026465795e+04	0.000000002143800
12	1.6275479114e+05	1.6275479142e+05	0.000000001723845
14	1.2026042798e+06	1.2026042842e+06	0.000000003634135
16	8.8861105010e+06	8.8861105205e+06	0.000000002197990
18	6.5659968911e+07	6.5659969137e+07	0.000000003450972
20	4.8516519307e+08	4.8516519541e+08	0.000000004828738

Tabelle 1.1. Numerischer Fehler bei der Auswertung der Potenzreihe für e^x .

1.1 Zahldarstellungen und Gleitpunktzahlen

Die Grundlage für Zahlendarstellungen auf Rechnern bildet der folgende Satz aus der Analysis über die Darstellung reeller Zahlen in einer **Basis** $\beta \in \mathbb{N}$, $\beta \geq 2$, den wir ohne Beweis angeben.

Satz 1.3 Jede reelle von Null verschiedene Zahl $x \in \mathbb{R}$ lässt sich in der Form

$$x = \pm \beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \frac{d_3}{\beta^3} + \dots \right) \quad (1.1)$$

darstellen; mit den **Ziffern** $d_1, d_2, \dots, \in \{0, 1, 2, \dots, \beta - 1\}$, wobei $d_1 \neq 0$, und dem **Exponenten** $e \in \mathbb{Z}$.

Die sogenannte wissenschaftliche Darstellung (1.1) wird eindeutig, wenn man zusätzlich noch fordert, dass eine unendliche Teilmenge $\mathbb{N}_1 \subset \mathbb{N}$ mit $d_k \neq \beta - 1$ für alle $k \in \mathbb{N}_1$ existiert. Diese Forderung ist notwendig, da ansonsten $x = +10^1 \cdot 0.1$ und $x = +10^0 \cdot 0.999 \dots$ zwei verschiedene Darstellungen der gleichen Zahl wären. Da wir aber auf dem Rechner sowieso nur mit einer endlichen Anzahl von Ziffern arbeiten können, ist dieser Aspekt für uns nicht weiter von Interesse.

Beispiel 1.4 (Basen)

System	β	Zifferndarstellung
Dezimal	10	0, 1, 2, ..., 9
Binär	2	0, 1
Oktal	8	0, 1, 2, ..., 7
Hexadezimal	16	0, 1, 2, ..., 9, A, B, C, D, E, F

Finger oder Fäuste*

Das im Alltag verbreitetste Zahlensystem ist das Dezimalsystem ($\beta = 10$). Allerdings gibt es Ausnahmen, so benutzen etwa einige Naturvölker zum Zählen auch die Zehen ($\beta = 20$). Weitere in einigen Regionen der Welt gebräuchliche Basen sind 4, 8, 12 sowie 16, siehe <http://de.wikipedia.org/wiki/Zahlensystem>. In der Computertechnik hat sich aus naheliegenden Gründen das Binärsystem $\beta = 2$ durchgesetzt. (Bei der Darstellung wird zur Übersichtlichkeit oft auf das Hexadezimalsystem zurückgegriffen.) Dem Ende der 1950-iger Jahre entwickelten auf dem Ternärsystem ($\beta = 3$) beruhenden Rechner der Moskauer Staatsuniversität war dagegen nur kurzer Erfolg beschieden. Das Binärsystem wartet mit einigen unerwarteten Effekten auf. So ist etwa die Dezimalzahl 0.1 nicht durch einen endlichen Binärbruch darstellbar. In der Finanzwirtschaft setzt man bei Spezialanwendungen deswegen gelegentlich das Dezimalsystem ein, welches aber meist softwareseitig emuliert wird und dementsprechend langsam ist. Prozessoren, deren Hardware Dezimaloperationen unterstützt, sind selten; ein Beispiel ist der IBM Power6 Prozessor.

1.2 Gleitpunktzahlensysteme

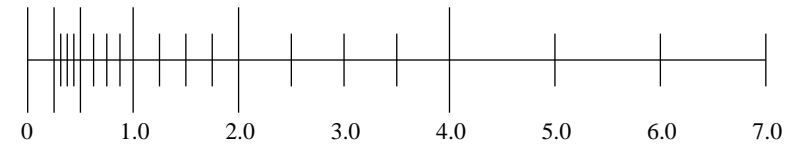
Auf Digitalrechnern können nur endlich viele Zahlen $\mathbb{F} \subset \mathbb{R}$ gespeichert und verarbeitet werden. Abgesehen von Spezialanwendungen (Bsp: Fixpunktzahlen in der Digitaltechnik) haben sich die Gleitpunktzahlen durchgesetzt.

Definition 1.5 (Gleitpunktzahlen $\mathbb{F}(\beta, t, e_{\min}, e_{\max})$) Sei $\beta \in \mathbb{N}$, $\beta \geq 2$ (**Basis**) und $t \in \mathbb{N}$ (**Mantissenlänge**), sowie $e_{\min} < 0 < e_{\max}$ mit $e_{\min}, e_{\max} \in \mathbb{Z}$ (**Exponentenschranken**). Dann ist die Menge $\mathbb{F} = \mathbb{F}(\beta, t, e_{\min}, e_{\max})$ wie folgt definiert:

$$\mathbb{F} := \left\{ \pm \beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \dots + \frac{d_t}{\beta^t} \right) : \begin{array}{l} d_1, \dots, d_t \in \{0, \dots, \beta - 1\}, \\ d_1 \neq 0, \\ e \in \mathbb{Z}, e_{\min} \leq e \leq e_{\max}. \end{array} \right\} \cup \{0\}.$$

Offensichtlich ist \mathbb{F} eine endliche Teilmenge von \mathbb{R} . Es ist wichtig zu bemerken, dass die Elemente vom \mathbb{F} nicht gleichmässig verteilt sind.

Beispiel: Nichtnegative Gleitpunktzahlen für $\beta = 2, t = 3, e_{\min} = -1, e_{\max} = 3$:



Lemma 1.6 In $\mathbb{F} = \mathbb{F}(\beta, t, e_{\min}, e_{\max})$ gilt

$$x_{\min}(\mathbb{F}) := \min\{x \in \mathbb{F} : x > 0\} = \beta^{e_{\min}-1}, \quad (1.2)$$

$$x_{\max}(\mathbb{F}) := \max\{x \in \mathbb{F}\} = \beta^{e_{\max}}(1 - \beta^{-t}). \quad (1.3)$$

Beweis. Betrachtet man Definition 1.5, so besteht die einzige Schwierigkeit im Beweis von (1.2)–(1.3) darin, den Bereich der Mantisse zu begrenzen. Sei dazu $x \in \mathbb{F}$ mit Mantisse $d = \sum_{k=1}^t d_k \beta^{-k}$. Dann gilt wegen $d_1 \neq 0$:

$$\beta^{-1} \leq d \leq \sum_{k=1}^t \beta^{-k}(\beta - 1) = 1 - \beta^{-t} \quad (1.4)$$

$$\begin{array}{ccc} \uparrow & & \uparrow \\ d_1 \geq 1 & & d_k \leq \beta - 1. \end{array}$$

□

Lemma 1.7 Der Abstand zwischen einer Gleitpunktzahl $x \neq 0$ und der nächstgelegenen Gleitpunktzahl ist mindestens $\beta^{-1}\epsilon_M|x|$ und höchstens $\epsilon_M|x|$, wobei $\epsilon_M = \beta^{1-t}$ der Abstand von 1 zur nächstgrösseren Gleitpunktzahl ist.

Beweis. Übung. □

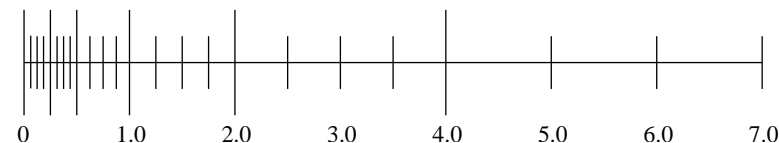
Aus Lemma 1.7 folgt, dass für eine feste Basis β die relative Genauigkeit von \mathbb{F} lediglich von der Mantissenlänge t abhängt; e_{\min}, e_{\max} bestimmen nur den Bereich von \mathbb{F} , nicht aber die Genauigkeit von \mathbb{F} .

Zwischen 0 und der kleinsten nichtnegativen Gleitpunktzahl ist eine "Lücke" von $\beta^{e_{\min}-1}$, die durch Hinzufügen der sogenannten **denormalisierten Zahlen** geschlossen werden kann:

$$\widehat{\mathbb{F}} := \mathbb{F} \cup \left\{ \pm \beta^{e_{\min}} \left(\frac{d_2}{\beta^2} + \dots + \frac{d_t}{\beta^t} \right) : d_2, \dots, d_t \in \{0, \dots, \beta - 1\} \right\}, \quad (1.5)$$

wobei noch der Fall $d_2 = \dots = d_t = 0$ ausgeschlossen ist. In der Notation von Definition 1.5 werden also noch Zahlen hinzugefügt, die minimalen Exponenten e_{\min} und $d_1 = 0$ haben. Gleitpunktzahlen in \mathbb{F} heissen **normalisiert** und in $\widehat{\mathbb{F}} \setminus \mathbb{F}$ heissen **denormalisiert**.

Beispiel: $\widehat{\mathbb{F}}$ für $\beta = 2, t = 3, e_{\min} = -1, e_{\max} = 3$:



Es ist zu beachten, dass für denormalisierte Zahlen, Lemma 1.7 *nicht* gilt; der denormalisierte Bereich hat niedrigere relative Genauigkeit.

Die im IEEE 754(r)-Standard festgelegte Gleitpunktarithmetik ist am weitesten verbreitet. Abgesehen vom oben beschriebenen Gleitpunktformat legt sie auch Regeln für Operationen auf \mathbb{F} fest. Ausserdem gibt sie eine einheitliche Richtlinie zur Behandlung von Ausnahmen ($1/0, \infty \cdot \infty, \infty - \infty$) vor. Die zwei im IEEE-Standard von 1995 festgelegten Grundformate zur Basis $\beta = 2$ sind:

Typ	Grösse	Mantisse	Exponent	x_{\min}	x_{\max}
Single	32 bits	23 + 1 bit	8 bits	10^{-38}	10^{+38}
Double	64 bits	52 + 1 bit	11 bits	10^{-308}	10^{+308}

Dabei wird ein Bit der Mantisse für das Vorzeichen verwendet. Auf der anderen Seite wird d_1 *nicht* gespeichert, da im normalisierten Bereich bei $\beta = 2$ immer $d_1 = 1$ gilt. Also stehen effektiv $t = 24$ bzw. $t = 53$ Stellen für die Mantisse zur Verfügung. Doppelte Genauigkeit ist Standard auf den meisten verbreiteten Hauptprozessoren (CPU). Auf dem cell processor (Playstation 3) sowie Graphikkarten findet man hingegen nur einfache Genauigkeit. Manche Prozessoren rechnen intern im Register mit 80 bit, was bei hintereinanderausgeführten Operationen zu höherer Genauigkeit führen kann, aber auch zu recht unschönen Phänomenen (Heisenberg bug) da die Registerbelegung in höheren Programmiersprachen nicht kontrollierbar ist.

Beispiel 1.8 (Detaillierte Darstellung im Rechner*) Als Beispiel betrachten wir die Darstellung der Dezimalzahl $x = 6.5$ in doppelter Genauigkeit. Zunächst einmal die Binärdarstellung von x :

$$\begin{aligned} x &= (-1)^0 \cdot \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{16} \right) 2^3 \\ &= (-1)^0 \cdot \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{16} \right) 2^{1025-1022}. \end{aligned}$$

Die Verschiebung im Exponenten wird durchgeführt, um das Vorzeichen im Exponenten einzusparen. Abgespeichert wird das Vorzeichen, der verschobene Exponent 1025, sowie die Mantisse ohne d_1 im Binärformat:

0 100 0000 0001 1010 0000 0000 ... 0000 0000 0000.

Diese Darstellung kann in MATLAB im Hexadezimalformat überprüft werden:
`format hex; 6.5 % ans = 401a000000000000` ◇

In MATLAB werden grundsätzlich alle Operationen in doppelter Genauigkeit durchgeführt. Variablen in einfacher Genauigkeit können mittels `single()` erzeugt werden.

	MATLAB
realmin	% ans = 2.2251e-308
realmax	% ans = 1.7977e+308
1/0	% ans = Inf
3*Inf	% ans = Inf
-1/0	% ans = -Inf
0/0	% ans = NaN
Inf - Inf	% ans = NaN

1.3 Runden

Im allgemeinen kann eine reelle Zahl nicht genau in \mathbb{F} dargestellt werden. Die Abbildung

$$rd : \text{Bereich}(\mathbb{F}) \rightarrow \mathbb{F}$$

heisst **Runden** und ordnet $x \in \text{Bereich}(\mathbb{F})$ die in \mathbb{F} nächstgelegene Zahl $rd(x)$ zu. Dabei bezeichnet $\text{Bereich}(\mathbb{F})$ die Menge aller $x \in \mathbb{R}$, deren Betrag zwischen $x_{\min}(\mathbb{F})$ und $x_{\max}(\mathbb{F})$ (siehe Lemma 1.6) liegt.

Die Abbildung rd ist nicht eindeutig bestimmt, wenn x genau in die Mitte zwischen zwei Gleitpunktzahlen fällt. Es existieren verschiedene "tie breaking"-Strategien, um die Abbildung in solchen (bei $\beta = 2$ nicht seltenen) Fällen eindeutig zu machen. IEEE 754 rundet zu dem Wert, bei dem die letzte Ziffer der Mantisse gerade (bei $\beta = 2$ also Null) ist. Das folgende Beispiel verdeutlicht diese Konvention.

Beispiel 1.9 Sei $\beta = 10, t = 3$. Dann sind $rd(0.9996) = 1.0, rd(0.3345) = 0.334, rd(0.3355) = 0.336$. ◇

Der folgende Satz gibt eine nützliche Abschätzung für den relativen Fehler von rd .

Satz 1.10 Für $x \in \text{Bereich}(\mathbb{F})$ gilt

$$\text{rd}(x) = x(1 + \delta), \quad |\delta| < u,$$

wobei $u = u(\mathbb{F}) := \frac{1}{2}\beta^{1-t}$.

Beweis. O.B.d.A. können wir $x > 0$ annehmen. Dann lässt sich x wie folgt darstellen (siehe Satz 1.3):

$$x = \mu \cdot \beta^{e-t}, \quad \beta^{t-1} \leq \mu < \beta^t.$$

Also liegt x zwischen den benachbarten Gleitpunktzahlen $y_1 = \lfloor \mu \rfloor \beta^{e-t}$, $y_2 = \lceil \mu \rceil \beta^{e-t}$, und damit ist $\text{rd}(x) \in \{y_1, y_2\}$. Da x zur näheren Gleitpunktzahl gerundet wird, haben wir $|\text{rd}(x) - x| \leq |y_2 - y_1|/2$. Zusammen mit Lemma 1.7 ergibt sich

$$|\text{rd}(x) - x| \leq \frac{1}{2}\beta^{-t}|y_1| = \frac{1}{2}\beta^{-t}\lfloor \mu \rfloor \beta^{e-t} < \frac{1}{2}\beta^{e-t}.$$

Wegen

$$\left| \frac{\text{rd}(x) - x}{x} \right| < \frac{\frac{\beta^{e-t}}{2}}{\mu \cdot \beta^{e-t}} \leq \frac{1}{2}\beta^{1-t} = u$$

ist damit die Behauptung gezeigt. \square

Die im Satz 1.10 verwendete Variable u ist die prominente Grösse jeder Rundungsfehleranalyse und heisst **Rundungseinheit** (engl. “unit roundoff”).

Gelegentlich ist die folgende Variation von Satz 1.10 nützlicher.

Satz 1.12 Für $x \in \text{Bereich}(\mathbb{F})$ gilt

$$\text{rd}(x) = \frac{x}{1 + \delta}, \quad |\delta| < u.$$

Beweis. Übung. \square

Es symbolisiere ‘ \circ ’ eine Grundrechenoperation:

$$\circ \in \{+, -, *, /\}.$$

Setzen wir formal $\pm\infty = x/0$ für $x \in \mathbb{R}$, gilt (mit der Ausnahme $0/0$)

$$\circ : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \cup \{\pm\infty\}.$$

Allerdings haben wir

$$\circ : \mathbb{F} \times \mathbb{F} \not\rightarrow \mathbb{F} \cup \{\pm\infty\}.$$

Um “zurück” nach \mathbb{F} zu kommen, müssen wir runden. Eine sinnvolle Annahme wäre, dass Elementaroperationen das Ergebnis *exakt* berechnen und *danach* erst runden. In der Praxis stellt man eine leicht schwächere Forderung an die Computerarithmetik, die aus der Kombination dieser Annahme mit Satz 1.10 folgt.

Definition 1.13 (Standardmodell der Rundung)

$$\text{rd}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \leq u, \quad \circ \in \{+, -, *, /\}. \quad (1.6)$$

Alternativ zu (1.6) kann man auch

$$\text{rd}(x \circ y) = \frac{x \circ y}{1 + \delta'}, \quad |\delta'| \leq u, \quad \circ \in \{+, -, *, /\} \quad (1.7)$$

benutzen (siehe auch Satz 1.12).

Beachte, dass Definition 1.13 *nicht* Kommutativität impliziert: $\text{rd}(x \circ y) \neq \text{rd}(y \circ x)$. In der Praxis ist diese aber fast immer erfüllt. Anders sieht es mit der Assoziativität aus; diese ist selbst unter der stärkeren Annahme “exakte Rechnung, gerundetes Ergebnis” verletzt.

Beispiel 1.14 Sei $\beta = 10$, $t = 2$. Dann haben wir

$$\begin{aligned} \text{rd}(\text{rd}(70 + 74) + 74) &= \text{rd}(140 + 74) = 210 \\ &\neq \text{rd}(70 + \text{rd}(74 + 74)) = \text{rd}(70 + 150) = 220 \end{aligned}$$

sowie

$$\begin{aligned} \text{rd}(\text{rd}(110 - 99) - 10) &= \text{rd}(11 - 10) = 1 \\ &\neq \text{rd}(110 + \text{rd}(-99 - 10)) = \text{rd}(110 - 110) = 0. \end{aligned}$$

\diamond

Idealerweise möchte man auch für elementare Funktionen $f \in \{\exp, \sin, \cos, \tan, \dots\}$ eine mit (1.6) vergleichbare Eigenschaft:

$$\text{rd}(f(x)) = f(x)(1 + \delta), \quad |\delta| \leq u. \quad (1.8)$$

Die Implementierung eines solchen numerischen Verfahrens zur Berechnung von $f(x)$ ist alles andere als trivial, insbesondere weil nicht vorhersagbar ist, wieviele Stellen von $f(x)$ genau berechnet werden müssen, um nach dem Runden (1.8) zu erfüllen (in der Literatur als *Table Maker’s Dilemma* bekannt).

1.4 Rundungsfehleranalyse

Das Standardmodell der Rundung erlaubt es uns, auf einfache Weise die Fortpflanzung von Rundungsfehlern in einer Rechnung zu kontrollieren. Als wichtiges Beispiel betrachten wir die Berechnung des euklidischen Skalarproduktes

$$\underline{x}^T \underline{y} = \sum_{k=1}^n x_k y_k.$$

Für die partielle Summe $s_i = \sum_{k=1}^i x_k y_k$ gilt in Computerarithmetik

$$\begin{aligned} \hat{s}_1 &= \text{rd}(x_1 y_1) = x_1 y_1 (1 + \delta_1), \\ \hat{s}_2 &= \text{rd}(\hat{s}_1 + x_2 y_2) = (\hat{s}_1 + x_2 y_2 (1 + \delta_2)) (1 + \delta_3) \\ &= x_1 y_1 (1 + \delta_1) (1 + \delta_3) + x_2 y_2 (1 + \delta_2) (1 + \delta_3), \end{aligned}$$

wobei für alle δ_i gilt $|\delta_i| \leq u$. Um die Notation zu vereinfachen, lassen wir die Indizes weg; im folgenden steht δ für eine beliebige, unbestimmte Zahl mit $|\delta| \leq u$. Mit dieser Konvention erhalten wir

$$\begin{aligned}\widehat{s}_3 &= \text{rd}(\widehat{s}_2 + x_3 y_3) = (\widehat{s}_2 + x_3 y_3(1 + \delta))(1 + \delta) \\ &= x_1 y_1(1 + \delta)^3 + x_2 y_2(1 + \delta)^3 + x_3 y_3(1 + \delta)^2.\end{aligned}$$

Induktiv ergibt sich also für $s_n = \underline{x}^T \underline{y}$:

$$\widehat{s}_n = x_1 y_1(1 + \delta)^n + x_2 y_2(1 + \delta)^n + x_3 y_3(1 + \delta)^{n-1} + \dots + x_n y_n(1 + \delta)^2. \quad (1.9)$$

Folgendes Resultat kann eingesetzt werden, um diesen Ausdruck zu vereinfachen.

Lemma 1.15 Sei $|\delta_i| \leq u$ für $i = 1, \dots, n$. Gilt $nu < 1$, so ist

$$\prod_{i=1}^n (1 + \delta_i) = 1 + \theta_n, \quad |\theta_n| \leq \frac{nu}{1 - nu} =: \gamma_n. \quad (1.10)$$

Beweis. Der Beweis erfolgt per Induktion. Induktionsschritt:

$$\begin{aligned}\prod_{i=1}^n (1 + \delta_i) &= (1 + \delta_n)(1 + \theta_{n-1}) =: 1 + \theta_n, \\ \theta_n &= \delta_n + (1 + \delta_n)\theta_{n-1}, \\ |\theta_n| &\leq u + (1 + u) \frac{(n-1)u}{1 - (n-1)u} \\ &= \frac{nu}{1 - (n-1)u} \leq \gamma_n.\end{aligned}$$

□

Die Anwendung von Lemma 1.15 auf (1.9) ergibt

$$\widehat{s}_n = x_1 y_1(1 + \theta_n) + x_2 y_2(1 + \theta'_n) + x_3 y_3(1 + \theta_{n-1}) + \dots + x_n y_n(1 + \theta_2), \quad (1.11)$$

mit $|\theta_j| \leq \gamma_j$ und $|\theta'_n| \leq \gamma_n$. Mit der Abschätzung $\gamma_j \leq \gamma_n$ erhalten wir

$$\text{rd}(\underline{x}^T \underline{y}) = (\underline{x} + \Delta \underline{x})^T \underline{y} = \underline{x}^T (\underline{y} + \Delta \underline{y}), \quad |\Delta \underline{x}| \leq \gamma_n |\underline{x}|, \quad |\Delta \underline{y}| \leq \gamma_n |\underline{y}|, \quad (1.12)$$

wobei $|\underline{x}|$ der Vektor mit den Komponenten $|x_i|$ sei und Ungleichungen zwischen Vektoren (und Matrizen) komponentenweise zu verstehen sind. Es ist zu bemerken, dass die Schranken in (1.12) im Gegensatz zu (1.11) *unabhängig* von der Reihenfolge der Summation sind.

Das Resultat (1.12) gibt den sogenannten **Rückwärtsfehler** an. Es sagt aus, dass das berechnete Ergebnis (Skalarprodukt) das exakte Ergebnis von leicht gestörten Eingabedaten (\underline{x} und \underline{y}) ist. Da die Eingabedaten sowieso nie exakt sind (allein schon das Abspeichern der Daten im Rechner erzeugt einen relativen Fehler von u), führt ein Algorithmus mit kleinem Rückwärtsfehler die Rechnung im Prinzip gerade so genau durch wie es eben die Daten zulassen. Der erzeugte Fehler im berechneten Ergebnis heisst **Vorwärtsfehler**. Aus (1.12) ergibt sich sofort eine Schranke für den Vorwärtsfehler im Skalarprodukt:

$$|\underline{x}^T \underline{y} - \text{rd}(\underline{x}^T \underline{y})| \leq \gamma_n \sum_{i=1}^n |x_i y_i| = \gamma_n |\underline{x}|^T |\underline{y}|. \quad (1.13)$$

Gilt $|\underline{x}^T \underline{y}| \approx |\underline{x}|^T |\underline{y}|$, so ist der relative Fehler $|\underline{x}^T \underline{y} - \text{rd}(\underline{x}^T \underline{y})|/|\underline{x}^T \underline{y}|$ klein. Gilt allerdings $|\underline{x}^T \underline{y}| \ll |\underline{x}|^T |\underline{y}|$, so ist kein kleiner relativer Fehler zu erwarten.

Als erste Anwendung der erhaltenen Schranken für das Skalarprodukt betrachten wir das Matrix-Vektor-Produkt

$$\underline{y} = \mathbf{A} \underline{x}, \quad \underline{x} \in \mathbb{R}^n, \quad \underline{y} \in \mathbb{R}^m, \quad \mathbf{A} \in \mathbb{R}^{m \times n}.$$

Bezeichnet \underline{a}_i^T die i -te Spalte von \mathbf{A} , so haben wir $y_i = \underline{a}_i^T \underline{x}$ und aus (1.12) folgt

$$\widehat{y}_i = (\underline{a}_i + \Delta \underline{a}_i)^T \underline{x}, \quad |\Delta \underline{a}_i| \leq \gamma_n |\underline{a}_i|.$$

Der Rückwärtsfehler erfüllt also

$$\widehat{\underline{y}} = (\mathbf{A} + \Delta \mathbf{A}) \underline{x}, \quad |\Delta \mathbf{A}| \leq \gamma_n |\mathbf{A}|, \quad (1.14)$$

und der sich daraus ergebende Vorwärtsfehler

$$|\widehat{\underline{y}} - \underline{y}| \leq \gamma_n |\mathbf{A}| |\underline{x}|.$$

Normweise Vorwärtsfehlerschranken ergeben sich direkt für die 1- und ∞ -Norm:

$$\|\widehat{\underline{y}} - \underline{y}\|_p \leq \gamma_n \|\mathbf{A}\|_p \|\underline{x}\|_p, \quad p \in \{1, \infty\}.$$

Mit den in Abschnitt 0.7 angegebenen Matrixnormäquivalenzen folgt ausserdem

$$\|\widehat{\underline{y}} - \underline{y}\|_2 \leq \min\{m, n\}^{1/2} \gamma_n \|\mathbf{A}\|_2 \|\underline{x}\|_2.$$

1.5 Auslöschung

Numerische Auslöschung tritt beim Subtrahieren von zwei fast gleichen, *bereits rundungsfehlerbehafteten* Zahlen in Gleitpunktarithmetik auf. Sie ist eine der Hauptursachen für die massive Verstärkung von Rundungsfehlern im Laufe einer Rechnung.

Beispiel 1.16 Betrachte

$$f(x) = \frac{1 - \cos(x)}{x^2}, \quad x = 1.2 \times 10^{-5}.$$

Dann ist $\cos(x)$ auf 10 Dezimalen gerundet

$$c = 0.9999\ 9999\ 99,$$

und

$$1 - c = 0.0000\ 0000\ 01.$$

Damit ist $(1 - c)/x^2 = 10^{-10}/1.44 \times 10^{-10} = 0.6944 \dots$. Da aber $0 \leq f(x) < \frac{1}{2}$ für $x \neq 0$, ist dieses Ergebnis offensichtlich vollkommen falsch. ◇

Allgemeiner seien $a, b \in \mathbb{R}$ gegeben, und

$$\widehat{a} = \text{rd}(a) = a(1 + \delta_a), \quad \widehat{b} = \text{rd}(b) = b(1 + \delta_b)$$

mit $|\delta_a| \leq u$, $|\delta_b| \leq u$. Dann gilt für $x = a - b$ und $\widehat{x} = \widehat{a} - \widehat{b}$

$$\left| \frac{x - \widehat{x}}{x} \right| = \left| \frac{-a\delta_a + b\delta_b}{a - b} \right| \leq u \frac{|a| + |b|}{|a - b|} \quad (1.15)$$

x	Alg. 1	Alg. 2
10^{-3}	1.0005236	1.0005002
10^{-4}	1.0001659	1.0000499
10^{-5}	1.0013580	1.0000050
10^{-6}	0.9536743	1.0000005
10^{-7}	1.1920929	1.0000001

Tabelle 1.2. Ergebnisse von Algorithmus 1 und 2 in einfacher Genauigkeit. Korrekt berechnete Stellen sind kursiv dargestellt.

Der relative Fehler in der Berechnung $x = a - b$ ist gross, falls

$$|a - b| \ll |a| + |b|.$$

Es ist zu betonen, dass in obiger Analyse a und b bereits Rundungsfehlerbehaftet sind. Die Subtraktion selbst geschieht bei $a \approx b$ *rundungsfehlerfrei*; sie verstärkt lediglich die bereits vorhandenen Fehler.

Rundungsfehler können sich auch wieder auslöschen, d.h., ein sehr ungenaues Zwischenergebnis führt nicht zwingend auf ein sehr ungenaues Endergebnis. Dies zeigt das folgende Beispiel.

Beispiel 1.17 (Berechnung von $(e^x - 1)/x$ für $x \downarrow 0$) Wir berechnen

$$f(x) = (e^x - 1)/x = \sum_{i=0}^{\infty} \frac{x^i}{(i+1)!} \quad (1.16)$$

in einfacher Genauigkeit (IEEE single) auf 2 Arten.

MATLAB	MATLAB
<pre>% Algorithmus 1 if x == 0, f = 1; else f = (exp(x) - 1) / x; end</pre>	<pre>% Algorithmus 2 y = exp(x); if y == 1, f = 1; else f = (y - 1) / log(y); end</pre>

Die erhaltenen Ergebnisse finden sich in Tabelle 1.2⁴. Um zu verstehen, wieso Algorithmus 1 wesentlich schlechtere Resultate liefert, betrachten wir $x = 9.0 \times 10^{-8}$ und nehmen an, dass die Implementierungen von $\exp(\cdot)$ und $\log(\cdot)$ das Fehlermodell (1.8) erfüllen. Die ersten 9 Dezimalstellen des exakten Ergebnisses sind

$$\frac{e^x - 1}{\log e^x} = 1.00000005.$$

Algorithmus 1 ergibt

$$\text{rd}\left(\frac{e^x - 1}{x}\right) = \text{rd}\left(\frac{1.19209290 \times 10^{-7}}{9.0 \times 10^{-8}}\right) = 1.32454766;$$

⁴Werden die Algorithmen in einem m-file verwendet, so kompiliert bzw. optimiert MATLAB den Code (JIT), was zur Folge hat, dass Teile des Codes allenfalls in hoeherer Genauigkeit durchgeführt werden. Um die Ergebnisse in Tabelle 1.2 zu reproduzieren muss vorher JIT mittels `feature accel off` ausgeschaltet werden.

Algorithmus 2 dagegen ergibt

$$\text{rd}\left(\frac{e^x - 1}{\log e^x}\right) = \text{rd}\left(\frac{1.19209290 \times 10^{-7}}{1.19209282 \times 10^{-7}}\right) = 1.00000006.$$

Beachte: Algorithmus 2 berechnet aufgrund von Auslöschung sehr ungenaue Werte für $e^x - 1 = 9.00000041 \times 10^{-8}$ und $\log e^x = 9 \times 10^{-8}$; die Rundungsfehler heben sich aber beim Dividieren heraus!

Dieses Phänomen kann durch die Rundungsfehleranalyse von Algorithmus 2 erklärt werden. Wir haben $\hat{y} = e^x(1 + \delta)$, $|\delta| \leq u$. Falls $\hat{y} = 1$, folgt damit

$$e^x(1 + \delta) = 1 \iff x = -\log(1 + \delta) = -\delta + \delta^2/2 - \delta^3/3 + \dots,$$

so dass folgt

$$\hat{f} = \text{rd}(1 + x/2 + x^2/6 + \dots|_{x=-\delta+O(\delta^2)}) = 1. \quad (1.17)$$

Falls $\hat{y} \neq 1$, folgt

$$\hat{f} = \text{rd}((\hat{y} - 1)/\log \hat{y}) = \frac{(\hat{y} - 1)(1 + \delta_1)}{\log \hat{y}(1 + \delta_2)} (1 + \delta_3), \quad |\delta_i| \leq u. \quad (1.18)$$

Definiere

$$v := \hat{y} - 1.$$

Dann gilt

$$\begin{aligned} g(\hat{y}) &:= \frac{\hat{y} - 1}{\log \hat{y}} = \frac{v}{\log(1 + v)} = \frac{v}{v - v^2/2 + v^3/3 - \dots} \\ &= \frac{1}{1 - v/2 + v^2/3 - \dots} = 1 + \frac{v}{2} + O(v^2). \end{aligned}$$

Falls x klein ist, ist $y \sim 1$ und

$$g(\hat{y}) - g(y) \approx \frac{\hat{y} - y}{2} \approx \frac{e^x \delta}{2} \approx \frac{\delta}{2} \approx g(y) \frac{\delta}{2}$$

(1.18) \implies

$$\left| \frac{\hat{f} - f}{f} \right| \leq 3.5u. \quad (1.19)$$

Also sind in Algorithmus 2, $\hat{y} - 1$ und $\log \hat{y}$ ungenau. Aber: $(\hat{y} - 1)/\log \hat{y}$ ist eine sehr genaue Approximation an $(y - 1)/\log y$ bei $y = 1$, da $g(y) := (y - 1)/\log y$ dort **“langsam variiert”**, denn $g(\cdot)$ hat bei $y = 1$ eine hebbare Singularität: es gilt $g'(1) = 1$, und wir sagen, $g(\cdot)$ ist bei $y = 1$ **“gut konditioniert”**. \diamond

1.6 Landau-Symbole (Wiederholung)

Die **Landau-Symbole** $O(\cdot)$ und $o(\cdot)$ werden verwendet um das asymptotische Verhalten einer Funktion $f(x)$ für $x \rightarrow a$ zu beschreiben.

Definition 1.18 Sei $f, g: \mathbb{R} \rightarrow \mathbb{R}$ und $a \in \mathbb{R} \cup \{\pm\infty\}$. Dann schreibt man

$$f(x) = O(g(x)) \text{ für } x \rightarrow a, \quad \text{wenn } \limsup_{x \rightarrow a} \left| \frac{f(x)}{g(x)} \right| < \infty,$$

$$f(x) = o(g(x)) \text{ für } x \rightarrow a, \quad \text{wenn } \lim_{x \rightarrow a} \left| \frac{f(x)}{g(x)} \right| = 0.$$

Sei $p(x) = a_l x^l + a_{l+1} x^{l+1} \dots + a_u x^u$ Polynom mit $a_l \neq 0, a_u \neq 0$ und $u \geq l$. Dann gilt

$$\begin{aligned} p(x) &= O(x^j) \text{ f\"ur } x \rightarrow \infty \quad \forall j \geq u, \\ p(x) &= o(x^j) \text{ f\"ur } x \rightarrow \infty \quad \forall j > u, \\ p(x) &= O(x^j) \text{ f\"ur } x \rightarrow 0 \quad \forall j \leq l, \\ p(x) &= o(x^j) \text{ f\"ur } x \rightarrow 0 \quad \forall j < l. \end{aligned}$$

Im allgemeinen gelten die folgenden leicht beweisbaren Rechenregeln:

$$\begin{aligned} f(x) = O(g(x)) &\implies c \cdot f(x) = O(g(x)), \\ f_1(x) = O(g_1(x)), f_2(x) = O(g_2(x)) &\implies f_1(x)f_2(x) = O(g_1(x)g_2(x)). \end{aligned}$$

Analoge Aussagen gelten f\"ur "o".

Kapitel 2

Direkte Lösung Linearer Gleichungssysteme

The closer one looks the more subtle and remarkable Gaussian elimination appears.
 —LOYD N. TREFETHEN, *Three Mysteries of Gaussian Elimination* (1985)

Wir betrachten lineare Gleichungssysteme in der Form

$$A\underline{x} = \underline{b}; \tag{2.1}$$

hier ist $A \in \mathbb{R}^{n \times n}$ eine reguläre Matrix, $\underline{b} \in \mathbb{R}^n$ ist gegeben, und $\underline{x} \in \mathbb{R}^n$ ist die gesuchte Lösung. Die Matrix A und die Vektoren \underline{x} , \underline{b} haben die Komponenten

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

Wir erinnern daran, dass (2.1) unter den gegebenen Voraussetzungen immer eindeutig lösbar ist.

2.1 Dreiecksmatrizen

Bevor wir den allgemeinen Fall behandeln, betrachten wir zunächst einmal zwei Spezialfälle von Matrizen A . Wir sagen, dass A eine *linke Dreiecksmatrix* (oft auch: *untere Dreiecksmatrix*) ist, falls

$$a_{ij} = 0 \quad \text{für alle } i, j \text{ mit } i < j.$$

Analog sprechen wir von A als einer *rechten Dreiecksmatrix* (oft auch: *obere Dreiecksmatrix*), falls

$$a_{ij} = 0 \quad \text{für alle } i, j \text{ mit } i > j.$$

Linke Dreiecksmatrizen werden meist mit L bezeichnet und rechte Dreiecksmatrizen mit R . Die Namensgebung ist aus der Struktur der Matrix leicht verständlich:

$$L = \begin{pmatrix} \ell_{11} & 0 & \cdots & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & \\ \ell_{31} & \ell_{32} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nn-1} & \ell_{nn} \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & \cdots & r_{2n} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & r_{nn} \end{pmatrix}.$$

(Im englischsprachigen Raum werden Matrizen dieses Typs typischerweise mit L und U für *lower* und *upper* bezeichnet.) Hat die Matrix A in (2.1) linke oder rechte Dreiecksgestalt, dann ist das Lösen des Gleichungssystems besonders einfach. Beim Lösen von

$$L\underline{x} = \underline{b}$$

spricht man von **Vorwärtssubstitution** und beim Lösen von

$$R\underline{x} = \underline{b}$$

spricht man von **Rückwärtssubstitution**. Die Namensgebung folgt aus der Tatsache, dass man beim Lösen von $L\underline{x} = \underline{b}$ die Unbekannten x_i sukzessive “vorwärts” bestimmt d.h. zuerst $x_1 = b_1/a_{11}$, mit dessen Hilfe man x_2 bestimmt, dann x_3 u.s.w. Beim Lösen von $R\underline{x} = \underline{b}$ werden die Unbekannten x_i sukzessive “rückwärts” bestimmt, d.h. zuerst $x_n = b_n/a_{nn}$, dann damit x_{n-1} , dann x_{n-2} u.s.w. Dieses Vorwärts- und Rückwärtseinsetzen formalisieren wir in den folgenden beiden Algorithmen.

<p>Algorithmus 2.1 Vorwärtssubstitution Input: Reguläre Linksdreiecksmatrix $A \in \mathbb{R}^{n \times n}$, $\underline{b} \in \mathbb{R}^n$. Output: Lösung \underline{x} von $A\underline{x} = \underline{b}$.</p> <pre> for $i = 1, 2, \dots, n$ do $x_i := \frac{1}{a_{ii}} \left(b_i - \sum_{k=1}^{i-1} a_{ik}x_k \right)$ end for</pre>	<p>Algorithmus 2.2 Rückwärtssubstitution Input: Reguläre Rechtsdreiecksmatrix $A \in \mathbb{R}^{n \times n}$, $\underline{b} \in \mathbb{R}^n$. Output: Lösung \underline{x} von $A\underline{x} = \underline{b}$.</p> <pre> for $i = n, n-1, \dots, 1$ do $x_i := \frac{1}{a_{ii}} \left(b_i - \sum_{k=i+1}^n a_{ik}x_k \right)$ end for</pre>
--	--

Man überzeugt sich leicht davon, dass in beiden Algorithmen für jedes i auf der rechten Seite der Zuweisung nur Objekte stehen, die bereits in einem vorangehenden Schritt bestimmt wurden.

Bemerkung 2.3 In MATLAB werden Gleichungssysteme mit dem “backslash” - Operator gelöst; also $\mathbf{x} = \mathbf{A} \setminus \mathbf{b}$. Dabei wird automatisch geprüft ob A linke oder rechte Dreiecksform hat. Ist dies der Fall, so wird eine Variante von Algorithmus 2.1 bzw. 2.2 verwendet. Das genaue Verhalten von \setminus kann mit der Funktion `linsolve` kontrolliert werden.

Für Algorithmus 2.1 führen wir exemplarisch eine Aufwands- und Fehleranalyse durch. Der Aufwand eines Algorithmus setzt sich aus der Anzahl von Operationen $+$, $-$, $*$, $/$ und elementaren Funktionsauswertungen zusammen. Jede Operation bzw. Auswertung zählt als ein **flop** (floating point operation – Gleitpunktoperation).⁵ In der i -ten Schleife von Algorithmus 2.1 werden 1 Division, $i - 1$ Multiplikationen, sowie $i - 1$ Additionen/Subtraktionen durchgeführt, zusammen also $2i - 1$ flops. Damit ist der Gesamtaufwand von Algorithmus 2.1

$$\sum_{i=1}^n (2i - 1) = n^2 \text{ flops.} \tag{2.2}$$

⁵Während Addition und Multiplikation in etwa gleich schnell sind und in einem Prozessorzyklus abgearbeitet werden, brauchen Division und die Auswertung elementarer Funktionen – abhängig vom Prozessor – mehrere Zyklen. Aus Bequemlichkeit verzichtet man aber auf die separate Zählung dieser Operationen, wenn sie keinen dominanten Anteil der Rechnung darstellen.

Zur Fehleranalyse haben wir folgendes Resultat.

Lemma 2.4 Die von Algorithmus 2.1 berechnete Lösung $\hat{\underline{x}}$ erfüllt

$$(\mathbf{A} + \Delta\mathbf{A})\hat{\underline{x}} = \underline{b}, \quad |\Delta\mathbf{A}| \leq \gamma_n |\mathbf{A}|,$$

wobei γ_n wie in Lemma 1.15 definiert ist.

Beweis. Um die Notation zu vereinfachen, bezeichne θ_i eine unbestimmte, generische Variable von der nur bekannt ist, dass sie $|\theta_i| \leq \gamma_i$ erfüllt. Mit den Modellen (1.6) und (1.7) gilt für die erste Schleife von Algorithmus 2.1

$$a_{11}(1 + \theta_1)\hat{x}_1 = b_1,$$

und für die zweite Schleife

$$a_{22}(1 + \theta_2)\hat{x}_2 = b_2 - a_{21}\hat{x}_1(1 + \theta_1).$$

Allgemeiner gilt in der i -ten Ausführung der Schleife

$$a_{ii}(1 + \theta_i)\hat{x}_i = b_i - a_{i1}\hat{x}_1(1 + \theta_{i-1}) - a_{i2}\hat{x}_2(1 + \theta_{i-2}) - \dots - a_{i,i-1}\hat{x}_{i-1}(1 + \theta_1).$$

Durch Setzen von $\hat{a}_{ii} := a_{ii}(1 + \theta_i)$ und $\hat{a}_{ik} := a_{ik}(1 + \theta_{i-k})$ folgt das Resultat. \square

Die Aussage von Lemma 2.4 über den Rückwärtsfehler bei der Lösung von Dreieckssystemen führt zu einer Schranke für den Vorwärtsfehler; dies behandeln wir in allgemeinerer Form in Abschnitt 2.4.3.

Ganz analog können (2.2) und Lemma 2.4 für die Rückwärtssubstitution, Algorithmus 2.2, gezeigt werden.

2.2 Gauss'scher Algorithmus und LR -Zerlegung

Im vorhergehenden Abschnitt haben wir gesehen, dass gestaffelte Gleichungssysteme (d.h. solche, bei denen die Matrix \mathbf{A} linke oder rechte Dreiecksgestalt hat) besonders einfach aufzulösen sind. Der Gauss'sche Algorithmus führt nun den allgemeinen Fall auf diese beiden Fälle zurück, indem er eine beliebige Matrix \mathbf{A} in ein Produkt aus einer Links- und einer Rechtsdreiecksmatrix zerlegt:

$$\mathbf{A} = \mathbf{L}\mathbf{R}. \quad (2.3)$$

Ist eine solche Zerlegung bekannt, dann kann das Gleichungssystem (2.1) mithilfe einer Vorwärts- und einer Rückwärtssubstitution gelöst werden. Führt man nämlich den Hilfsvektor $\underline{y} = \mathbf{R}\underline{x}$ ein, so ergibt sich $\underline{b} = \mathbf{A}\underline{x} = \mathbf{L}\mathbf{R}\underline{x} = \mathbf{L}(\mathbf{R}\underline{x}) = \mathbf{L}\underline{y}$; dies führt auf folgende Vorgehensweise:

- löse das Gleichungssystem $\mathbf{L}\underline{y} = \underline{b}$ für \underline{y} mithilfe von Algorithmus 2.1;
- löse das Gleichungssystem $\mathbf{R}\underline{x} = \underline{y}$ für \underline{x} mithilfe von Algorithmus 2.2.

Definition 2.5 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Dann besitzt \mathbf{A} eine LR -Zerlegung, falls es eine Rechtsdreiecksmatrix \mathbf{R} und eine Linksdreiecksmatrix \mathbf{L} mit $\ell_{11} = \dots = \ell_{nn} = 1$ gibt, so dass $\mathbf{A} = \mathbf{L}\mathbf{R}$.

Hat eine reguläre Matrix \mathbf{A} eine LR -Zerlegung, so ist diese eindeutig:

Satz 2.6 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ regulär und habe eine LR -Zerlegung $\mathbf{L}\mathbf{R} = \mathbf{A}$. Dann ist $\mathbf{R}_{ii} \neq 0$ für $i = 1, \dots, n$, und die Zerlegung ist eindeutig.

Beweis. Wegen $0 \neq \det \mathbf{A} = \det \mathbf{L} \cdot \det \mathbf{R} = \prod_{i=1}^n \mathbf{R}_{ii}$ folgt die erste Behauptung. Seien $\mathbf{L}\mathbf{R} = \mathbf{A} = \mathbf{L}'\mathbf{R}'$ zwei LR -Zerlegungen von \mathbf{A} . Dann sind nach obiger Überlegung \mathbf{R} und \mathbf{R}' invertierbar (ihre Determinanten verschwinden nicht). Somit gilt

$$\mathbf{L}'\mathbf{R}' = \mathbf{L}\mathbf{R} \implies \mathbf{L}^{-1}\mathbf{L}' = \mathbf{R}(\mathbf{R}')^{-1}.$$

Nach Proposition 0.20 sind $\mathbf{L}^{-1}\mathbf{L}'$ und $\mathbf{R}(\mathbf{R}')^{-1}$ Links- bzw. Rechtsdreiecksmatrizen. Weiterhin lässt sich leicht zeigen, dass die Diagonaleinträge von $\mathbf{L}^{-1}\mathbf{L}' \in \mathcal{L}_n^1$ ebenfalls alle 1 sind. Die einzige Matrix, die zugleich Linksdreiecksmatrix und Rechtsdreiecksmatrix ist, sowie Einsen auf der Diagonalen hat, ist die Identität. Also ist $\mathbf{R} = \mathbf{R}'$ und $\mathbf{L} = \mathbf{L}'$. \square

Bemerkung 2.7 Die Voraussetzung der Regularität von \mathbf{A} ist wesentlich für die Eindeutigkeitsaussage, wie das Beispiel

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

zeigt.

Die LR -Zerlegung einer Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ geschieht in $n - 1$ Schritten, mittels der allenfalls bereits bekannten Gauss-Eliminierung. Zur Motivation des Algorithmus schreiben wir das Gleichungssystem (2.1) aus:

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

Es wird nun von der zweiten, dritten, etc. Zeile ein geeignetes Vielfaches der ersten Zeile subtrahiert, um die Variable x_1 in Zeilen 2 bis n zu eliminieren. Wir definieren also (für $a_{11} \neq 0$)

$$l_{i1} := \frac{a_{i1}}{a_{11}}, \quad i = 2, \dots, n$$

ecksmatrix ergibt: Setzt man

$$\mathbf{L}_k := \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -l_{nk} & & & 1 \end{pmatrix}, \quad (2.5)$$

so kann man nachrechnen (Übung!), dass gilt:

$$\mathbf{A}^{(k)} = \mathbf{L}_k \mathbf{A}^{(k-1)} \quad \text{und} \quad \underline{\mathbf{b}}^{(k)} = \mathbf{L}_k \underline{\mathbf{b}}^{(k-1)}, \quad k = 1, \dots, n-1. \quad (2.6)$$

Man erhält also

$$\mathbf{A}^{(n-1)} = \mathbf{L}_{n-1} \mathbf{L}_{n-2} \cdots \mathbf{L}_1 \mathbf{A}^{(0)} = \mathbf{L}_{n-1} \mathbf{L}_{n-2} \cdots \mathbf{L}_1 \mathbf{A}.$$

Da alle auftretenden Linksdreiecksmatrizen \mathbf{L}_k regulär sind können wir dies umschreiben als

$$\mathbf{L} \mathbf{R} = \mathbf{A},$$

wobei

$$\mathbf{L} := \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \cdots \mathbf{L}_{n-1}^{-1}, \quad \mathbf{R} = \mathbf{A}^{(n-1)}.$$

Hier ist \mathbf{R} eine Rechtsdreiecksmatrix nach Konstruktion (vgl. (2.4)), und \mathbf{L} ist eine Linksdreiecksmatrix nach Proposition 0.20. Die Beziehung

$$\mathbf{L}_k^{-1} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & l_{nk} & & & 1 \end{pmatrix} \quad (2.7)$$

lässt sich leicht über $\mathbf{L}_k^{-1} \mathbf{L}_k = \mathbf{I}$ nachweisen. Damit gilt für die Einträge in \mathbf{L} ganz explizit (Übung!):

$$\mathbf{L} = \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \cdots \mathbf{L}_{n-1}^{-1} = \begin{pmatrix} 1 & & & & & \\ l_{21} & \ddots & & & & \\ l_{31} & \ddots & 1 & & & \\ \vdots & & l_{k+1,k} & 1 & & \\ \vdots & & \vdots & & \ddots & \\ l_{n1} & \cdots & l_{nk} & \cdots & l_{n,n-1} & 1 \end{pmatrix} \quad (2.8)$$

Wir haben also eine explizite Konstruktion einer LR -Zerlegung von \mathbf{A} gefunden: Die Einträge der Linksdreiecksmatrix \mathbf{L} sind die Faktoren l_{ik} , die im Laufe des Algorithmus bestimmt werden und die Rechtsdreiecksmatrix \mathbf{R} ist gerade das Endschema des Gauss'schen Algorithmus, die Matrix $\mathbf{A}^{(n-1)}$. Wir können unser Vorgehen in dem folgenden abstrakten Algorithmus, der *Gauss'schen Eliminierung ohne Pivotsuche* festhalten:

Algorithmus 2.8 (Rohfassung der LR -Zerlegung ohne Pivotsuche)

Input: Reguläre Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$.

Output: LR -Zerlegung $\mathbf{A} = \mathbf{L} \mathbf{R}$ mit $\mathbf{R} = \mathbf{A}^{(n-1)}$ und \mathbf{L} wie in (2.8).

$\mathbf{A}^{(0)} := \mathbf{A}$

for $k = 1, \dots, n-1$ **do**

Bestimme Matrix \mathbf{L}_k (vgl. (2.5)) durch Berechnung der Faktoren

$$l_{ik} := \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k+1, \dots, n.$$

Setze $\mathbf{A}^{(k)} := \mathbf{L}_k \mathbf{A}^{(k-1)}$.

end for

Mit Hilfe der Konstruktion der LR -Zerlegung lassen sich Bedingungen für die Existenz der LR -Zerlegung angeben.

Satz 2.9 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ regulär. Dann hat \mathbf{A} eine LR -Zerlegung genau dann, wenn alle führenden Hauptabschnittsmatrizen $\mathbf{S}_k = (a_{ij})$, $i, j = 1, \dots, k$, regulär sind.

Beweis. Partitioniere für eine gegebene LR -Zerlegung $\mathbf{A} = \mathbf{L} \mathbf{R}$:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{12} & \mathbf{L}_{22} \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{pmatrix}, \quad \mathbf{L}_{11}, \mathbf{R}_{11} \in \mathbb{R}^{k \times k}.$$

Dann gilt

$$\mathbf{S}_k = \mathbf{L}_{11} \mathbf{R}_{11} \Rightarrow \det(\mathbf{S}_k) = \det(\mathbf{L}_{11}) \det(\mathbf{R}_{11}) = \det(\mathbf{R}_{11}),$$

wobei ausgenutzt wurde, dass \mathbf{L} nur Einsen auf der Diagonale hat. Da $0 \neq \det(\mathbf{A}) = \det(\mathbf{R}) = \prod_{i=1}^n r_{ii}$, folgt auch $\det(\mathbf{R}_{11}) = \prod_{i=1}^k r_{ii} \neq 0$ und damit die Regularität von \mathbf{S}_k .

Um die Rückrichtung zu beweisen, benutzen wir obige Konstruktion der LR -Zerlegung. Dazu muss im k -ten Schritt von Algorithmus 2.8 $a_{kk}^{(k-1)} \neq 0$ erfüllt sein. Nehmen wir an, dass die ersten $k-1$ Schritte von Algorithmus 2.8 erfolgreich ausgeführt wurden (also $a_{11} \neq 0, a_{22} \neq 0, \dots, a_{k-1,k-1} \neq 0$), so gilt

$$\mathbf{S}_k = \begin{pmatrix} 1 & & & & \\ l_{21} & \ddots & & & \\ \vdots & \ddots & 1 & & \\ l_{k1} & \cdots & l_{k,k-1} & 1 & \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & & \vdots \\ a_{kk}^{(k-1)} \end{pmatrix}.$$

Also folgt aus $0 \neq \det(\mathbf{S}_k) = a_{11} a_{22}^{(1)} \cdots a_{kk}^{(k-1)}$, dass $a_{kk}^{(k-1)} \neq 0$ und damit kann der k -te Schritt von Algorithmus 2.8 ausgeführt werden. \square

Für eine Computerimplementierung von Algorithmus 2.8 müssen die Matrixmultiplikationen noch ausgeschrieben werden, da eine explizite Matrix-Multiplikation ohne Ausnutzung der Struktur \mathbf{L}_k viel zu teuer wäre. Ausserdem man direkt "auf"

der Matrix A operieren, d.h. sie während des Algorithmus verändern. Dies geschieht aus Speicherplatzgründen, weil man nicht Speicher für die n Matrizen $A^{(0)}, A^{(1)}, \dots$ bereitstellen kann/will.

Algorithmus 2.10 (LR-Zerlegung ohne Pivotisierung)
Input: Reguläre Matrix $A \in \mathbb{R}^{n \times n}$.
Output: **LR**-Zerlegung $A = LR$, wobei der linke bzw. rechte Dreiecksanteil von A mit L bzw. R überschrieben wird.

```

for k = 1, ..., n - 1 do
  for i = k + 1, ..., n do
     $a_{ik} \leftarrow \frac{a_{ik}}{a_{kk}}$ 
    for j = k + 1, ..., n do
       $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$ 
    end for
  end for
end for
    
```

MATLAB

```

function A = lr(A)
% Berechnet LR-Zerlegung von A.
for k = 1:n
  A(k+1:n,k) = A(k+1:n,k) / A(k,k);
  A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - A(k+1:n,k) * A(k,k+1:n);
end
    
```

In der formulierten Form geht die Matrix A in Algorithmus 2.10 verloren, da sie im strikten linken Dreiecksanteil mit L und im rechten Dreiecksanteil mit R überschrieben wird. Genauer gilt:

$$\tilde{a}_{ij} = \begin{cases} r_{ij} & \text{falls } j \geq i \\ l_{ij} & \text{falls } j < i. \end{cases} \quad (2.9)$$

Der Aufwand von Algorithmus 2.10 ergibt sich als

$$\sum_{k=1}^{n-1} (1 + 2(n - k))(n - k) = \frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n = \frac{2}{3}n^3 + O(n^2) \text{ flops.}$$

Insgesamt wird zum Lösen eines linearen Gleichungssystems (2.1) folgender Algorithmus durchgeführt:

Algorithmus 2.11 (Gauss-Algorithmus ohne Pivotsuche)

1. Bestimme **LR**-Zerlegung von A mithilfe von Algorithmus 2.10.
2. Löse $L\underline{y} = \underline{b}$ mithilfe der Vorwärtssubstitution (Algorithmus 2.1).
3. Löse $R\underline{x} = \underline{y}$ mithilfe der Rückwärtssubstitution (Algorithmus 2.2).

LR -Zerlegung (Alg. 2.10)	$\frac{2}{3}n^3 + O(n^2)$
Vorwärtssubst. (Alg. 2.1 unter Ausnutzung von $l_{ii} = 1$)	$n^2 - n$
Rückwärtssubst. (Alg. 2.2)	n^2
Gesamtkosten	$\frac{2}{3}n^3 + O(n^2)$

Tabelle 2.1. Kosten für das Lösen eines linearen Gleichungssystems mit Algorithmus 2.11.

In Tabelle 2.1 sind die Gesamtkosten beim Lösen eines linearen Gleichungssystems mithilfe von Algorithmus 2.11 zusammengestellt. Wie man sieht, werden die Kosten (für grosse n) durch die **LR**-Zerlegung dominiert. Bereits für $n = 100$ machen die Vorwärts- und Rückwärtssubstitutionen zusammen nur 3% der Gesamtkosten aus. Ein positiver Nebeneffekt ist, dass, falls eine **LR**-Zerlegung erst einmal aufgestellt ist, das lineare Gleichungssystem (2.1) für viele verschiedene rechte Seiten \underline{b} billig gelöst werden kann.

Bemerkung 2.12 In der **LR**-Zerlegung in Algorithmus 2.10 haben wir nicht den Fall abgefangen, dass ein Pivotelement 0 sein könnte. Algorithmus 2.11 versagt deshalb, wenn die Bedingungen von Satz 2.9 nicht erfüllt sind, z.B. bei dem trivialen Beispiel

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

Der Behandlung solcher Fälle werden wir uns im Abschnitt 2.5 zuwenden.

2.3 Fehleranalyse

Als Faustregel gilt in der Numerik:

In der Nähe eines unzulässigen Inputs eines Algorithmus treten in endlicher Arithmetik numerische Probleme auf.

Ganz konkret bedeutet dies, dass wir Probleme bei Algorithmus 2.11 erwarten können, wenn eine der führenden Hauptabschnittsmatrizen von A nahezu singular ist. Diese Erwartung wird erfüllt, wie das folgende Beispiel zeigt.

Beispiel 2.13 Wir bestimmen in 4-stelliger dezimaler Gleitpunktarithmetik \mathbb{F} (d.h. $\beta = 10, t = 4$) die Lösung \underline{x} des folgenden linearen Gleichungssystems:

$$A = \begin{pmatrix} 3.1 \cdot 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}, \quad \underline{b} = \begin{pmatrix} -3 \\ -7 \end{pmatrix}.$$

Es ist dann $l_{21} = 1/(3.1 \cdot 10^{-4}) \approx 3.226 \cdot 10^3$ und die **LR**-Zerlegung von A ist

$$L = \begin{pmatrix} 1 & 0 \\ 3.226 \cdot 10^3 & 1 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 3.1 \cdot 10^{-4} & 1 \\ 0 & 1 - l_{21} \end{pmatrix} \approx \begin{pmatrix} 3.1 \cdot 10^{-4} & 1 \\ 0 & -3.225 \cdot 10^3 \end{pmatrix}.$$

Die Lösung von $\mathbf{L}\underline{y} = \underline{b}$ führt dann auf

$$\underline{y} = \mathbf{L}^{-1}\underline{b} = \begin{pmatrix} -3 \\ 9.671 \cdot 10^3 \end{pmatrix}$$

und damit ist die Lösung \underline{x} von $\mathbf{R}\underline{x} = \underline{y}$

$$\underline{x} = \mathbf{R}^{-1}\underline{y} = \begin{pmatrix} \frac{1}{3.1 \cdot 10^{-4}}(-3 - (-2.999)) \\ -2.999 \end{pmatrix} = \begin{pmatrix} -3.226 \\ -2.999 \end{pmatrix}.$$

Das "exakte" Ergebnis ist $\underline{x} = (-4.001246 \dots, -2.99875 \dots)^\top$. Hier sind also beim Rückwärtseinsetzen in der x_1 -Komponente durch Auslöschung alle Ziffern verloren gegangen. Der Grund ist die schlechte Pivotwahl. Wir starteten mit einem sehr kleinen Pivotelement und erhielten dadurch sehr grosse (und auch sehr kleine) Einträge in der \mathbf{LR} -Zerlegung. Dies führt dann zu Auslöschung bei der Vorwärts- und Rückwärtssubstitution. \diamond

Beispiel 2.13 zeigt, dass grosse Einträge in \mathbf{L} und \mathbf{R} zu Instabilitäten bei Vorwärts- und Rückwärtssubstitution führen können. Formal kann dies durch folgende Fehleranalyse begründet werden.

Lemma 2.14 Wenn in Algorithmus 2.10 kein Pivotelement 0 auftritt, dann gilt für die berechneten Faktoren $\widehat{\mathbf{L}}$ und $\widehat{\mathbf{R}}$ der \mathbf{LR} -Zerlegung:

$$\widehat{\mathbf{L}}\widehat{\mathbf{R}} = \mathbf{A} + \Delta\mathbf{A}, \quad |\Delta\mathbf{A}| \leq \gamma_n |\widehat{\mathbf{L}}| |\widehat{\mathbf{R}}|.$$

Beweis. Analog zum Beweis von Lemma 2.4. \square

Die Kombination von Lemma 2.14 mit den Fehlerschranken für Vorwärts- und Rückwärtssubstitution (Lemma 2.4) ergibt folgende Rückwärtsfehlerschranke für die Lösung eines linearen Gleichungssystems.

Satz 2.15 Die von Algorithmus 2.11 berechnete Lösung $\widehat{\underline{x}}$ des linearen Gleichungssystems $\mathbf{A}\underline{x} = \underline{b}$ erfüllt:

$$(\mathbf{A} + \Delta\mathbf{A})\widehat{\underline{x}} = \underline{b}, \quad |\Delta\mathbf{A}| \leq \gamma_{3n} |\widehat{\mathbf{L}}| |\widehat{\mathbf{R}}|,$$

wobei $\widehat{\mathbf{L}}$ und $\widehat{\mathbf{R}}$ die von Algorithmus 2.10 berechneten Faktoren der \mathbf{LR} -Zerlegung sind.

Beweis. Übung. \square

Für die Matrix \mathbf{A} aus Beispiel 2.13 ergibt sich

$$|\widehat{\mathbf{L}}| |\widehat{\mathbf{R}}| = \begin{pmatrix} 3.1 \cdot 10^{-4} & 1 \\ 1.00006 & 6.451 \cdot 10^3 \end{pmatrix}.$$

Bei $\beta = 10$, $t = 4$ gilt $\gamma_2 \approx 10^{-3}$ d.h., der Rückwärtsfehler zerstört alle Stellen des (2,2)-Eintrags von \mathbf{A} .

Satz 2.15 gibt ein Resultat für den komponentenweisen Rückwärtsfehler von \mathbf{A} an. Für die weitere Analyse ist es bequemer, den normweisen Rückwärtsfehler $\|\Delta\mathbf{A}\|$ für eine Matrixnorm $\|\cdot\|$ zu betrachten. Zum Beispiel folgt für die 2-Norm:

$$\|\Delta\mathbf{A}\|_2 \leq n\gamma_{3n} \|\widehat{\mathbf{L}}\| \|\widehat{\mathbf{R}}\|_2 \approx 3n^2 u \|\widehat{\mathbf{L}}\| \|\widehat{\mathbf{R}}\|_2.$$

Unter Umständen überschätzt Satz 2.15 den tatsächlichen Rückwärtsfehler grob. Ausserdem wird es nicht für jeden Algorithmus zur Lösung von linearen Gleichungssystemen möglich sein, **a priori** eine solche Analyse durchzuführen. Um trotzdem **a posteriori** einschätzen zu können, ob ein Algorithmus einen kleinen Rückwärtsfehler liefert, benötigt man lediglich das **Residuum** $\underline{r} = \underline{b} - \mathbf{A}\widehat{\underline{x}}$.

Satz 2.16 Sei $\widehat{\underline{x}} \neq 0$ die berechnete Lösung des linearen Gleichungssystems $\mathbf{A}\underline{x} = \underline{b}$. Dann gilt

$$\min \{ \|\Delta\mathbf{A}\|_2 : (\mathbf{A} + \Delta\mathbf{A})\widehat{\underline{x}} = \underline{b} \} = \frac{\|\underline{r}\|_2}{\|\widehat{\underline{x}}\|_2},$$

wobei $\underline{r} = \underline{b} - \mathbf{A}\widehat{\underline{x}}$.

Beweis. Aus $(\mathbf{A} + \Delta\mathbf{A})\widehat{\underline{x}} = \underline{b}$ folgt

$$\|\Delta\mathbf{A}\|_2 \|\widehat{\underline{x}}\|_2 \geq \|\Delta\mathbf{A}\widehat{\underline{x}}\|_2 = \|\underline{r}\|_2.$$

Damit gilt $\min \{ \|\Delta\mathbf{A}\|_2 : (\mathbf{A} + \Delta\mathbf{A})\widehat{\underline{x}} = \underline{b} \} \geq \|\underline{r}\|_2 / \|\widehat{\underline{x}}\|_2$. Um Gleichheit zu zeigen, benötigt man lediglich eine zulässige Störung $\Delta_0\mathbf{A}$ mit $\|\Delta_0\mathbf{A}\|_2 = \|\underline{r}\|_2 / \|\widehat{\underline{x}}\|_2$. Diese ist durch

$$\Delta_0\mathbf{A} = \frac{1}{\|\widehat{\underline{x}}\|_2^2} \underline{r} \widehat{\underline{x}}^\top$$

gegeben, da $(\mathbf{A} + \Delta_0\mathbf{A})\widehat{\underline{x}} = \mathbf{A}\widehat{\underline{x}} + \underline{r} = \underline{b}$. \square

2.4 Störungsanalyse und Kondition

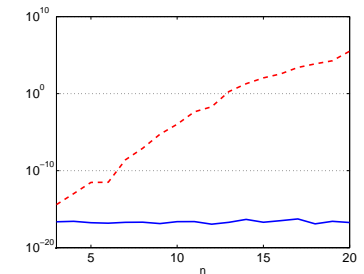
Die Rückwärtsfehleranalyse von Satz 2.15 sagt aus, dass $\widehat{\underline{x}}$ die exakte Lösung eines gestörten linearen Gleichungssystems ist. Das Resultat sagt nichts über den Vorwärtsfehler, also die Genauigkeit von $\widehat{\underline{x}}$, aus. Dieser kann wesentlich höher sein.

Beispiel 2.17 Sei $\mathbf{A} = (a_{ij})$ mit $a_{ij} = \frac{1}{i+j-1}$ die $n \times n$ **Hilbert-Matrix** und $\underline{b} = (1, \dots, 1)^\top$. Wir berechnen die Lösung von $\mathbf{A}\underline{x} = \underline{b}$ mit dem \(\backslash\)-Operator und vergleichen diese mit der "exakten" Lösung, die in 100-stelliger Dezimalarithmetik berechnet wird (Symbolic-Toolbox von MATLAB).

```

MATLAB
digits(100); rw = []; vw = [];
for n = 3:20,
    A = hilib(n); b = ones(n,1);
    x = A \ b;
    rw = [rw; norm(A*x-b) / ...
          norm(x)];
    xe = vpa(A) \ vpa(b);
    vw = [vw; norm( ...
          double( x-xe ) / norm(x) ];
end

```



Die gestrichelte rote Linie gibt den relativen Fehler von \hat{x} und die blaue Linie den Rückwärtsfehler (vgl. Satz 2.16) für steigendes n an. Der Vorwärtsfehler wächst offenbar sehr schnell obwohl \hat{x} die exakte Lösung eines nur sehr leicht gestörten Systems ist. \diamond

2.4.1 Kondition einer Funktion

Das in Beispiel 2.17 beobachtete Phänomen wollen wir im folgenden näher betrachten. Dazu ist es hilfreich eine numerische Berechnung in einem abstrakten Rahmen zu betrachten. Eine Berechnung kann als Funktion $f: \text{Input} \mapsto \text{Output}$ verstanden werden. Wir nehmen an, dass die Menge aller Inputs einen endlichdimensionalen Vektorraum V_i mit Norm $\|\cdot\|_{V_i}$ bildet. Entsprechend bildet die Menge aller Outputs einen endlichdimensionalen Vektorraum V_o mit Norm $\|\cdot\|_{V_o}$. Die Berechnung ist also die Auswertung von $f(x) \in V_o$ für ein $x \in V_i$ (x kann eine Matrix sein; dann wäre $V_i = \mathbb{R}^{n \times n}$). Schon aufgrund von Rundungsfehlern bei der Darstellung von x in der Menge \mathbb{F} der Gleitpunktzahlen muss mit einem Fehler in x gerechnet werden. Es stellt sich die Frage wie sich ein solcher Fehler Δx bei der Auswertung von $f(x + \Delta x)$ verstärkt.

Hierzu nehmen wir an, dass $f: V_i \rightarrow V_o$ an der Stelle $x \in V_i$ zweimal stetig (Fretchet) differenzierbar ist, d.h. die Differentiale f' , f'' existieren in einer hinreichend grossen Umgebung von x . Dann gilt formal nach Taylor:

$$x \mapsto f(x), \quad x + \Delta x \mapsto f(x + \Delta x) = f(x) + f'(x) \Delta x + O(\|\Delta x\|_{V_i}^2)$$

für $\Delta x \rightarrow 0$. Die Grösse des Differentials f' an der Stelle x , d.h. $\|f'(x)\|_{\mathcal{L}(V_i, V_o)}$, ist ein Mass für die **absolute Sensitivität** von $f(x)$ gegen eine (kleine) Störung Δx von x .

Für die Analyse der Fehlerfortpflanzung in Gleitpunktarithmetik ist es oftmals vernünftiger, den sog. **relativen Fehler** zu betrachten, denn er ist unabhängig von der Skalierung von x oder $f(x)$: Für $f(x) \neq 0$, $x \neq 0$, $\Delta x \rightarrow 0$ gilt

$$\begin{aligned} \frac{\|f(x + \Delta x) - f(x)\|_{V_o}}{\|f(x)\|_{V_o}} &= \frac{\|f'(x)\Delta x\|_{V_o} + O(\|\Delta x\|_{V_i}^2)}{\|f(x)\|_{V_o}} \implies \\ \underbrace{\frac{\|f(x + \Delta x) - f(x)\|_{V_o}}{\|f(x)\|_{V_o}}}_{\text{rel. Fehler in } f} &\leq \left(\frac{\|f'(x)\|_{\mathcal{L}(V_i, V_o)}}{\|f(x)\|_{V_o}} \|x\|_{V_i} \right) \underbrace{\frac{\|\Delta x\|_{V_i}}{\|x\|_{V_i}}}_{\text{rel. Störung in } x} + O(\|\Delta x\|_{V_i}^2). \end{aligned}$$

Definition 2.18 Die (relative) **Kondition der Funktion** $x \mapsto f(x)$ an der Stelle x ist

$$\text{cond}(f, x) := \frac{\|f'(x)\|_{\mathcal{L}(V_i, V_o)}}{\|f(x)\|_{V_o}} \|x\|_{V_i}.$$

Als einfaches Beispiel betrachten wir die Subtraktion zweier Zahlen $a, b \in \mathbb{R}$. Dann sind $V_i = \mathbb{R}^2$ (als Norm wählen wir $\|\cdot\|_2$) und $V_o = \mathbb{R}$. Ausserdem gilt $f: V_i \rightarrow V_o$ mit $f: (a, b) \mapsto a - b$ und

$$\|f'(a_0, b_0)\|_2 = \left\| \left(\frac{\partial f}{\partial a} \Big|_{(a_0, b_0)}, \frac{\partial f}{\partial b} \Big|_{(a_0, b_0)} \right) \right\|_2 = \|(1, -1)\|_2 = \sqrt{2}.$$

Also ist die Kondition von f an der Stelle (a_0, b_0)

$$\text{cond}(f, (a_0, b_0)) = \frac{\sqrt{2}}{|a_0 - b_0|} \sqrt{a_0^2 + b_0^2}.$$

Wie erwartet wird die Kondition gross, wenn $a_0 \approx b_0$.

Bemerkung 2.19 In vielen Anwendungen ist f nicht explizit bekannt (z.B. Eigenwertberechnung). Trotzdem kann man oft mit dem Satz über implizite Funktionen Aussagen über f' treffen.

2.4.2 Kondition einer Matrix

Für den Rest dieses Abschnitts nehmen wir durchweg an, dass $\|\cdot\|_M$ eine **submultiplikative Matrixnorm** und $\|\cdot\|_V$ eine dazu konsistente Vektornorm ist. Wir lassen zur Vereinfachung der Notation die Subskripten durchweg fallen – es sollte aus dem Zusammenhang klar sein, was gemeint ist.

Sei $\mathbf{X} \in \mathbb{R}^{n \times n}$ regulär und $f(\mathbf{X}) = \mathbf{X}^{-1}$. Wir betrachten die Kondition dieser Matrixfunktion an der Stelle $\mathbf{X} = \mathbf{A}$. Sei dazu $\Delta \mathbf{A} \in \mathbb{R}^{n \times n}$ eine “kleine” Störung von \mathbf{A} :

$$\|\Delta \mathbf{A}\| \ll \|\mathbf{A}\|.$$

Zunächst betrachten wir $n = 1$. Dann ist $0 \neq \mathbf{A} \in \mathbb{R}$, $\Delta \mathbf{A} \in \mathbb{R}$ und

$$f(\mathbf{A} + \Delta \mathbf{A}) = f(\mathbf{A}) + \Delta \mathbf{A} f'(\mathbf{A}) + O(\Delta \mathbf{A}^2).$$

Für $f(\mathbf{A}) = \mathbf{A}^{-1}$ folgt also

$$\begin{aligned} (\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1} &= \Delta \mathbf{A} (-\mathbf{A}^{-2}) + O(\Delta \mathbf{A}^2) \implies \\ \frac{|(\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1}|}{|\mathbf{A}^{-1}|} &= |\mathbf{A} [(\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1}]| \\ &= |\mathbf{A} \Delta \mathbf{A} (-\mathbf{A}^{-2}) + O(\Delta \mathbf{A}^2)| \\ &\leq |\mathbf{A}| |\mathbf{A}^{-1}| \frac{|\Delta \mathbf{A}|}{|\mathbf{A}|} + O(\Delta \mathbf{A}^2). \end{aligned}$$

Dabei haben wir die Ungleichung in einer Form geschrieben, das sie dem Fall $n > 1$ so ähnlich wie möglich aussieht.

Proposition 2.20 (Störungsschranke bei Matrixinversion) Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ regulär und $\Delta \mathbf{A} \in \mathbb{R}^{n \times n}$ Störung von \mathbf{A} mit $\|\mathbf{A}^{-1}\| \|\Delta \mathbf{A}\| < 1$. Dann gilt

$$\underbrace{\frac{\|(\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1}\|}{\|\mathbf{A}^{-1}\|}}_{\text{Rel. Fehler in } \mathbf{A}^{-1}} \leq \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \underbrace{\frac{\|\Delta \mathbf{A}\|}{\|\mathbf{A}\|}}_{\text{Rel. Fehler in } \mathbf{A}} \left(1 + O(\|\Delta \mathbf{A}\|)\right).$$

Beweis. Wir werden in den nächsten beiden Abschnitten sehen, dass die Annahme $\|\mathbf{A}^{-1}\| \|\Delta \mathbf{A}\| < 1$ die Existenz von $(\mathbf{A} + \Delta \mathbf{A})^{-1}$ nach sich zieht. Also gilt

$$[(\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1}](\mathbf{A} + \Delta \mathbf{A}) = \mathbf{I} - \mathbf{A}^{-1}(\mathbf{A} + \Delta \mathbf{A}) = -\mathbf{A}^{-1} \Delta \mathbf{A},$$

und wir haben

$$(\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1} = -\mathbf{A}^{-1} \Delta \mathbf{A} (\mathbf{A} + \Delta \mathbf{A})^{-1}.$$

Damit folgt

$$\begin{aligned} \|(\mathbf{A} + \Delta\mathbf{A})^{-1} - \mathbf{A}^{-1}\| &\leq \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\| \|(\mathbf{A} + \Delta\mathbf{A})^{-1} - \mathbf{A}^{-1} + \mathbf{A}^{-1}\| \\ &\leq \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\| (\|(\mathbf{A} + \Delta\mathbf{A})^{-1} - \mathbf{A}^{-1}\| + \|\mathbf{A}^{-1}\|) \\ \implies \|(\mathbf{A} + \Delta\mathbf{A})^{-1} - \mathbf{A}^{-1}\| (1 - \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\|) &\leq \|\mathbf{A}^{-1}\|^2 \|\Delta\mathbf{A}\| \\ \implies \frac{\|(\mathbf{A} + \Delta\mathbf{A})^{-1} - \mathbf{A}^{-1}\|}{\|\mathbf{A}^{-1}\|} &\leq \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\| (1 - \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\|)^{-1} \\ &= \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} (1 - \|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\|)^{-1} \\ &= \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} \sum_{i=0}^{\infty} (\|\mathbf{A}^{-1}\| \|\Delta\mathbf{A}\|)^i. \end{aligned}$$

Die Behauptung ergibt sich aus der Abschätzung der geometrischen Reihe mittels $1 + O(\|\Delta\mathbf{A}\|)$. \square

Die obigen Betrachtungen zeigen, dass der Ausdruck $\|\mathbf{A}\| \|\mathbf{A}^{-1}\|$ die *Kondition der Funktion* $f(\mathbf{X}) = \mathbf{X}^{-1}$ an der Stelle $\mathbf{X} = \mathbf{A}$ ist. Wegen dessen Wichtigkeit bezeichnet man $\|\mathbf{A}\| \|\mathbf{A}^{-1}\|$ kurz als **Kondition** (auch Konditionszahl) von \mathbf{A} , d.h.

$$\kappa(\mathbf{A}) := \|\mathbf{A}\| \|\mathbf{A}^{-1}\|. \quad (2.10)$$

Beachte, dass die Kondition von der verwendeten Norm $\|\cdot\|$ abhängt. Wir schreiben $\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2$.

MATLAB	
<code>cond(A)</code>	% Kondition von A in der 2-Norm
<code>cond(A,p)</code>	% Kondition von A in der p-Norm mit p = 1, 2, inf
<code>cond(A,'fro')</code>	% Kondition von A in der Frobenius-Norm
<code>condest(A)</code>	% Schaetzer fuer Kondition von A in der 1-Norm

2.4.3 Störungsanalyse linearer Gleichungssysteme

Wir wollen uns jetzt dem eigentlichen Ziel unserer Betrachtungen zuwenden, dem Einfluss von Störungen auf die Lösung eines linearen Gleichungssystems. Betrachte für $\mathbf{A} \in \mathbb{R}^{n \times n}$ regulär, $\mathbf{b} \in \mathbb{R}^n$:

$$\mathbf{A} \mathbf{x} = \mathbf{b}. \quad (2.11)$$

Als einfache Vorbetrachtung nehmen wir an, dass *nur* die rechte Seite gestört ist. Sei $\hat{\mathbf{b}} = \mathbf{b} + \Delta\mathbf{b}$ Störung von \mathbf{b} und $\hat{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x} = \mathbf{A}^{-1} \hat{\mathbf{b}}$ Lösung von

$$\mathbf{A} \hat{\mathbf{x}} = \hat{\mathbf{b}}. \quad (2.12)$$

Dann gilt $\Delta\mathbf{x} = \mathbf{A}^{-1} \Delta\mathbf{b}$ und damit

$$\|\Delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \|\Delta\mathbf{b}\|. \quad (2.13)$$

Der absolute Fehler in \mathbf{x} ist also gross, falls $\|\mathbf{A}^{-1}\|$ gross ist. Für den relativen Fehler folgt mit $\|\mathbf{x}\| \geq \|\mathbf{b}\|/\|\mathbf{A}\|$, dass

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \underbrace{\|\mathbf{A}\| \|\mathbf{A}^{-1}\|}_{\kappa(\mathbf{A})} \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}. \quad (2.14)$$

Was passiert wenn in (2.12) auch \mathbf{A} gestört wird? Der folgende Satz zeigt, dass sich die Schranke nur unwesentlich ändert, vorausgesetzt die Störung von \mathbf{A} ist klein.

Satz 2.21 Sei $\mathbf{A} \in \mathbb{C}^{n \times n}$ invertierbar und sei $\Delta\mathbf{A} \in \mathbb{R}^{n \times n}$ so dass

$$\|\mathbf{A}^{-1} \Delta\mathbf{A}\| < 1. \quad (2.15)$$

Dann ist

$$(\mathbf{A} + \Delta\mathbf{A}) \hat{\mathbf{x}} = \mathbf{b} + \Delta\mathbf{b} \quad (2.16)$$

eindeutig lösbar und es gilt

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}^{-1}\| \|\mathbf{A}\|}{1 - \|\mathbf{A}^{-1} \Delta\mathbf{A}\|} \left(\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \right). \quad (2.17)$$

Der Beweis von Satz 2.21 wird im nächsten Abschnitt gegeben.

2.4.4 Konvergente Folgen und Reihen von Matrizen

Zum Beweis von Satz 2.21 sowie später für die Konvergenz iterativer Verfahren benötigen wir die Verallgemeinerung geometrischer Reihen auf Matrizen. Wir erinnern daran, dass im Folgenden $\|\cdot\|$ immer eine **submultiplikative Matrixnorm** ist.

Motivation: sei $q \in \mathbb{C}$ beliebig, fest gegeben. Dann gilt für alle $m \in \mathbb{N}$

$$(1 - q)(1 + q + q^2 + \dots + q^m) = 1 - q^{m+1}.$$

Falls $|q| < 1$, gilt $|q|^{m+1} \rightarrow 0$ für $m \rightarrow \infty$ und

$$1 = \lim_{m \rightarrow \infty} (1 - q^{m+1}) = (1 - q) \lim_{m \rightarrow \infty} \sum_{i=0}^m q^i = (1 - q) \sum_{i=0}^{\infty} q^i,$$

d.h. es gilt die wohlbekannt Summationsformel für die geometrische Reihe:

$$(1 - q)^{-1} = \sum_{i=0}^{\infty} q^i.$$

Sei nun $\mathbf{B} \in \mathbb{R}^{n \times n}$ eine Matrix. Dann gilt für alle $m \in \mathbb{N}$

$$(\mathbf{I} - \mathbf{B})(\mathbf{I} + \mathbf{B} + \mathbf{B}^2 + \dots + \mathbf{B}^m) = \mathbf{I} - \mathbf{B}^{m+1}.$$

Falls $\|\mathbf{B}\| < 1$ gilt, so folgt für $m \rightarrow \infty$

$$\|\mathbf{B}^{m+1}\| = \|\mathbf{B}^m \mathbf{B}\| \leq \|\mathbf{B}^m\| \|\mathbf{B}\| \leq \dots \leq \|\mathbf{B}\|^{m+1} \rightarrow 0.$$

Somit existiert der Grenzwert der **Neumann'schen Reihe**

$$\lim_{m \rightarrow \infty} \sum_{i=0}^m \mathbf{B}^i = \sum_{i=0}^{\infty} \mathbf{B}^i \quad (2.18)$$

und es gilt

$$(\mathbf{I} - \mathbf{B})^{-1} = \sum_{i=0}^{\infty} \mathbf{B}^i.$$

Insbesondere ist in diesem Fall $\mathbf{I} - \mathbf{B}$ invertierbar.

Das obige Argument zeigt, dass $\|\mathbf{B}\| < 1$ hinreichend ist für die Konvergenz der Neumann'schen Reihe *in dieser Norm*. Damit ist das Konvergenzkriterium abhängig von der Wahl der Norm $\|\cdot\|$. Gibt es eine hinreichende *und* notwendige Bedingung für die Konvergenz der Neumann'schen Reihe? Für die Formulierung einer solchen Bedingung benutzt man den Spektralradius $\rho(\mathbf{B})$ einer Matrix \mathbf{B} – wir erinnern daran, dass $\rho(\cdot)$ *keine Matrixnorm* ist (Übung!). Der Beweis des folgenden Satzes zeigt aber, dass es für jedes $\varepsilon > 0$ eine (von ε abhängige) Matrixnorm $\|\cdot\|_{\varepsilon}$ gibt, so dass $\rho(\mathbf{B}) \leq \|\mathbf{B}\|_{\varepsilon} \leq \rho(\mathbf{B}) + \varepsilon$.

Satz 2.22 Sei $\mathbf{B} \in \mathbb{C}^{n \times n}$. Dann gilt $\mathbf{B}^m \rightarrow \mathbf{0}$ für $m \rightarrow \infty$ genau dann, wenn $\rho(\mathbf{B}) < 1$.

Beweis. “ \Rightarrow ”: Sei $\lambda \in \mathbb{C}$ Eigenwert von \mathbf{B} mit Eigenvektor $\underline{v} \in \mathbb{C}^n$. Dann gilt für jede von einer Vektornorm induzierten Matrixnorm $\|\cdot\|$, dass

$$\lambda \underline{v} = \mathbf{B} \underline{v} \Rightarrow |\lambda| \|\underline{v}\| = \|\mathbf{B} \underline{v}\| \leq \|\mathbf{B}\| \|\underline{v}\| \Rightarrow |\lambda| \leq \|\mathbf{B}\|.$$

Daher gilt $\rho(\mathbf{B}) \leq \|\mathbf{B}\|$ und

$$\rho(\mathbf{B})^m = \rho(\mathbf{B}^m) \leq \|\mathbf{B}^m\| \leq \|\mathbf{B}\|^m \xrightarrow{m \rightarrow \infty} 0.$$

Also folgt $\rho(\mathbf{B}) < 1$.

“ \Leftarrow ”: Wähle $\varepsilon > 0$ so, dass $\rho(\mathbf{B}) + \varepsilon < 1$ und betrachte die Jordan-Normalform von $\varepsilon^{-1} \mathbf{B}$:

$$\varepsilon^{-1} \mathbf{B} = \mathbf{X} \mathbf{J} \mathbf{X}^{-1}.$$

Definiere für eine beliebige Matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ die submultiplikative Matrixnorm

$$\|\mathbf{A}\|_{\varepsilon} := \|\mathbf{X}^{-1} \mathbf{A} \mathbf{X}\|_{\infty}.$$

Zerlegen wir $\mathbf{J} = \varepsilon^{-1} \mathbf{D} + \mathbf{N}$ mit $\mathbf{D} = \text{diag}(\lambda_1, \dots, \lambda_n)$, so gilt

$$\|\mathbf{B}\|_{\varepsilon} := \|\mathbf{X}^{-1} \mathbf{B} \mathbf{X}\|_{\infty} = \|\mathbf{D} + \varepsilon \mathbf{N}\|_{\infty} \leq \rho(\mathbf{B}) + \varepsilon < 1.$$

Mit $\|\mathbf{B}^m\|_{\varepsilon} \leq \|\mathbf{B}\|_{\varepsilon}^m$ folgt die Behauptung. \square

Korollar 2.23 Sei $\mathbf{B} \in \mathbb{C}^{n \times n}$ mit $\rho(\mathbf{B}) < 1$. Dann gilt:

1. $(\mathbf{I} - \mathbf{B})^{-1}$ existiert;
2. die Neumann-Reihe $\sum_{j=0}^{\infty} \mathbf{B}^j$ konvergiert, und

$$(\mathbf{I} - \mathbf{B})^{-1} = \sum_{j=0}^{\infty} \mathbf{B}^j. \quad (2.19)$$

Mithilfe der Neumann-Reihe können wir jetzt den Beweis von Satz 2.21 abschliessen.

Beweis von Satz 2.21. Mit der Regularität von \mathbf{A} folgt

$$\begin{aligned} (\mathbf{A} + \Delta \mathbf{A})^{-1} &= (\mathbf{A}(\mathbf{I} + \mathbf{A}^{-1} \Delta \mathbf{A}))^{-1} \\ &= (\mathbf{I} - \mathbf{B})^{-1} \mathbf{A}^{-1}, \text{ wobei } \mathbf{B} := -\mathbf{A}^{-1} \Delta \mathbf{A} \end{aligned}$$

Aus den obigen Betrachtungen folgt mit $\|\mathbf{B}\| = \|\mathbf{A}^{-1} \Delta \mathbf{A}\| < 1$, dass $(\mathbf{I} - \mathbf{B})^{-1}$ existiert, also auch

$$(\mathbf{A} + \Delta \mathbf{A})^{-1} = (\mathbf{I} - \mathbf{B})^{-1} \mathbf{A}^{-1}.$$

Damit folgt

$$\begin{aligned} \hat{\underline{x}} - \underline{x} &= (\mathbf{A} + \Delta \mathbf{A})^{-1} (\underline{b} + \Delta \underline{b}) - \mathbf{A}^{-1} \underline{b} \\ &= [(\mathbf{A} + \Delta \mathbf{A})^{-1} - \mathbf{A}^{-1}] \underline{b} + (\mathbf{A} + \Delta \mathbf{A})^{-1} \Delta \underline{b} \\ &= \left[(\mathbf{I} + \underbrace{\mathbf{A}^{-1} \Delta \mathbf{A}}_{-\mathbf{B}})^{-1} \mathbf{A}^{-1} - \mathbf{A}^{-1} \right] \underline{b} + (\mathbf{I} + \mathbf{A}^{-1} \Delta \mathbf{A})^{-1} \mathbf{A}^{-1} \Delta \underline{b} \\ &= \left[\left(\sum_{k=0}^{\infty} \mathbf{B}^k \right) - \mathbf{I} \right] \mathbf{A}^{-1} \underline{b} + \left[\sum_{k=0}^{\infty} (-\mathbf{A}^{-1} \Delta \mathbf{A})^k \right] \mathbf{A}^{-1} \Delta \underline{b} \\ &= \left[\left(\sum_{k=1}^{\infty} (-\mathbf{A}^{-1} \Delta \mathbf{A})^k \right) \underbrace{\mathbf{A}^{-1} \underline{b}}_{\underline{x}} + \left[\sum_{k=0}^{\infty} (-\mathbf{A}^{-1} \Delta \mathbf{A})^k \right] \mathbf{A}^{-1} \Delta \underline{b} \right], \end{aligned}$$

und somit

$$\begin{aligned} \|\underline{x} - \hat{\underline{x}}\| &\leq \left[\sum_{k=1}^{\infty} \|\mathbf{A}^{-1} \Delta \mathbf{A}\|^k \right] \|\underline{x}\| \\ &\quad + \left[\sum_{k=0}^{\infty} \|\mathbf{A}^{-1} \Delta \mathbf{A}\|^k \right] \|\mathbf{A}^{-1}\| \|\Delta \underline{b}\|. \end{aligned}$$

Mit $\|\underline{b}\| = \|\mathbf{A} \underline{x}\| \leq \|\mathbf{A}\| \|\underline{x}\| \Rightarrow \|\underline{x}\|^{-1} \leq \|\mathbf{A}\| / \|\underline{b}\|$ folgt

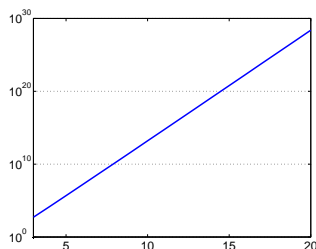
$$\begin{aligned} \frac{\|\underline{x} - \hat{\underline{x}}\|}{\|\underline{x}\|} &\leq \|\mathbf{A}^{-1} \Delta \mathbf{A}\| \left[\sum_{k=0}^{\infty} \|\mathbf{A}^{-1} \Delta \mathbf{A}\|^k \right] \\ &\quad + \left[\sum_{k=0}^{\infty} \|\mathbf{A}^{-1} \Delta \mathbf{A}\|^k \right] \|\mathbf{A}^{-1}\| \|\Delta \underline{b}\| \|\mathbf{A}\| / \|\underline{b}\|, \end{aligned}$$

woraus folgt

$$\begin{aligned} \frac{\|\underline{x} - \hat{\underline{x}}\|}{\|\underline{x}\|} &\leq \frac{1}{1 - \|\mathbf{A}^{-1} \Delta \mathbf{A}\|} \left[\underbrace{\|\mathbf{A}^{-1} \Delta \mathbf{A}\|}_{\leq \|\mathbf{A}^{-1}\| \|\mathbf{A}\| \|\Delta \mathbf{A}\| / \|\mathbf{A}\|} + \underbrace{\|\mathbf{A}\| \|\mathbf{A}^{-1}\|}_{\kappa(\mathbf{A})} \frac{\|\Delta \underline{b}\|}{\|\underline{b}\|} \right] \\ &\leq \frac{\|\mathbf{A}^{-1}\| \|\mathbf{A}\|}{1 - \|\mathbf{A}^{-1} \Delta \mathbf{A}\|} \left[\frac{\|\Delta \mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\Delta \underline{b}\|}{\|\underline{b}\|} \right]. \end{aligned} \quad (2.20)$$

\square

Obige Analyse erklärt das starke Wachstum des Vorwärtsfehlers in Beispiel 2.17. Für wachsendes n wächst die Kondition der Hilbert-Matrix exponentiell!⁶ Der folgende Plot zeigt die Konditionszahl für die 2-Norm in Abhängigkeit von n .



2.5 Pivotstrategien

Der Fehler einer berechneten Lösung eines linearen Gleichungssystems setzt sich aus zwei Komponenten zusammen. Einerseits kann die Matrix \mathbf{A} sehr schlecht konditioniert sein; dagegen kann man – ist die Matrix einmal aufgestellt – auf algorithmischer Seite wenig machen⁷. Andererseits kann aber auch der *Algorithmus* numerisch instabil sein, d.h., der Rückwärtsfehler ist wesentlich höher als die durch Rundungsfehler verursachten Fehler in den Daten. Wir sagen, dass ein Algorithmus zur Lösung des Gleichungssystems $\mathbf{A}\underline{x} = \underline{b}$ **numerisch rückwärtsstabil** ist, wenn für das berechnete $\hat{\underline{x}}$ gilt

$$(\mathbf{A} + \Delta\mathbf{A})\hat{\underline{x}} = \underline{b}, \quad \|\Delta\mathbf{A}\|_2 \lesssim u\|\mathbf{A}\|_2.$$

Nach Satz 2.16 ist dies äquivalent zu

$$\|\underline{x}\|_2 = \|\underline{b} - \mathbf{A}\underline{x}\|_2 \lesssim u\|\mathbf{A}\|_2\|\underline{x}\|_2.$$

Offensichtlich ist der Gauss-Algorithmus ohne Pivotisierung kein stabiler Algorithmus für Beispiel 2.13. Das Problem in Beispiel 2.13 war ein kleines Pivotelement, welches grosse Einträge in den Faktoren \mathbf{L} und \mathbf{R} nach sich zieht. Nach Satz 2.15 führen grosse Einträge in \mathbf{L} oder \mathbf{R} zu einem grossen Rückwärtsfehler.

Um das Problem der kleinen Pivotelemente in den Griff zu bekommen, wird deshalb nicht eine \mathbf{LR} -Zerlegung der Matrix \mathbf{A} gesucht, sondern die \mathbf{LR} -Zerlegung einer Matrix \mathbf{A}_{per} , die durch geeignetes Vertauschen von Zeilen von \mathbf{A} entstanden ist. Man beachte, dass für das Lösen von Gleichungssystemen das Vertauschen von Zeilen keine Rolle spielt (wenn man beim Vektor auf der rechten Seite die entsprechende Vertauschung ebenfalls durchführt). Dass Zeilenvertauschen numerisch vorteilhaft sein kann, zeigt folgende Fortsetzung von Beispiel 2.13:

⁶Siehe Beckermann, B. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. Numer. Math. 85 (2000), no. 4, 553–577.

⁷Frühe Arbeiten empfehlen die vorherige Anwendung einer Diagonalskalierung $\mathbf{D}_1\mathbf{A}\mathbf{D}_2$, um die Konditionszahl zu minimieren. Moderne Implementierungen führen meist **keine** Skalierung durch, siehe Abschnitt 7.8 in [Higham 2002] für eine ausführliche Diskussion.

Beispiel 2.24 Wir lösen das lineare Gleichungssystem aus Beispiel 2.13, indem wir die beiden Zeilen von $\mathbf{A}\underline{x} = \underline{b}$ vertauschen, d.h. wir betrachten

$$\mathbf{A}_{per} = \begin{pmatrix} 1 & 1 \\ 3.1 \cdot 10^{-4} & 1 \end{pmatrix}, \quad \underline{b}_{per} = \begin{pmatrix} -7 \\ -3 \end{pmatrix}.$$

Nun ist $l_{21} = 3.1 \cdot 10^{-4}$ und

$$\mathbf{L}_{per} = \begin{pmatrix} 1 & 0 \\ 3.1 \cdot 10^{-4} & 1 \end{pmatrix}, \quad \mathbf{R}_{per} = \begin{pmatrix} 1 & 1 \\ 0 & 1 - l_{21} \end{pmatrix} \approx \begin{pmatrix} 1 & 1 \\ 0 & 0.9997 \end{pmatrix}.$$

Für die Lösungen \underline{y} , \underline{x} von $\mathbf{L}\underline{y} = \underline{b}$, $\mathbf{R}\underline{x} = \underline{y}$ erhalten wir damit

$$\underline{y} = \begin{pmatrix} -7 \\ -2.998 \end{pmatrix}, \quad \underline{x} = \begin{pmatrix} -4.001 \\ -2.999 \end{pmatrix}.$$

Wir erhalten also das korrekte Ergebnis bis auf Rundungsgenauigkeit. Da die Permutationsmatrix

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

bei Multiplikation von links an die Matrix \mathbf{A} die Zeilen 1 und 2 vertauscht, haben wir also folgende Zerlegung erhalten:

$$\mathbf{L}_{per}\mathbf{R}_{per} = \mathbf{A}_{per} = \mathbf{P}\mathbf{A}.$$

◇

In Beispiel 2.24 konnte das lineare Gleichungssystem numerisch stabil gelöst werden, indem die beiden Zeilen des Gleichungssystems vertauscht wurden. Für eine allgemeine Matrix \mathbf{A} ist die richtige Zeilenanordnung natürlich nicht im Voraus bekannt. Sie muss also während des Algorithmus mitbestimmt werden. Man geht algorithmisch wie folgt vor (siehe Algorithmus 2.27 unten): In jedem Eliminationsschritt wird nicht einfach $a_{kk}^{(k-1)}$ als Pivot benutzt, sondern es wird in der Spalte k das **betragsgrösste Element** $a_{ik}^{(k-1)}$ mit Zeilenindex $i \geq k$ gesucht; anschliessend werden die Zeilen k und i vertauscht und dann der Eliminationsschritt durchgeführt.

Das Vertauschen von Zeilen in einer Matrix beschreibt man formal am besten mit *Permutationsmatrizen*.

Definition 2.25 (Permutationsmatrix) Sei $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ eine Permutation der Zahlen $1, \dots, n$ (d.h. π ist eine bijektive Abbildung) und bezeichne $\mathbf{e}_1, \dots, \mathbf{e}_n$ die n Einheitsvektoren: $(\mathbf{e}_i)_j = \delta_{ij}$. Dann heisst

$$\mathbf{P}_\pi := (\mathbf{e}_{\pi(1)}, \mathbf{e}_{\pi(2)}, \dots, \mathbf{e}_{\pi(n)})$$

die zu π gehörige Permutationsmatrix.

Lemma 2.26 (Eigenschaften von Permutationsmatrizen) Sei $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ eine Permutation und \mathbf{P}_π die zugehörige Permutationsmatrix. Dann gilt:

1. $\mathbf{P}_\pi \mathbf{e}_i = \mathbf{e}_{\pi(i)}$ für $i \in \{1, \dots, n\}$.
2. $\mathbf{P}_\pi^{-1} = \mathbf{P}_\pi^\top$.

3. Die Matrix $\mathbf{P}_\pi \mathbf{A}$ entsteht aus \mathbf{A} durch Zeilenpermutation, d.h. die i -te Zeile von \mathbf{A} ist die $\pi(i)$ -te Zeile von $\mathbf{P}_\pi \mathbf{A}$, $i = 1, \dots, n$.

4. Sei $\bar{\pi}$ eine weitere Permutation. Dann gilt $\mathbf{P}_{\bar{\pi} \circ \pi} = \mathbf{P}_{\bar{\pi}} \mathbf{P}_\pi$.

Beweis. Übung. \square

Der Algorithmus zur **LR**-Zerlegung mit Pivotsuche ist damit wie folgt.

Algorithmus 2.27 (LR-Zerlegung mit Spaltenpivotsuche)

Input: Reguläre Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$.

Output: LR-Zerlegung $\mathbf{PA} = \mathbf{LR}$ mit Permutationsmatrix \mathbf{P} .

$\mathbf{A}^{(0)} := \mathbf{A}$

for $k = 1, \dots, n - 1$ **do**

Suche $i \in \{k, \dots, n\}$ mit $|a_{ik}^{(k-1)}| \geq |a_{i'k}^{(k-1)}|$ für alle $i' \in \{k, \dots, n\}$

Setze $\pi_k : \{1, \dots, n\} \rightarrow \{1, \dots, k - 1, i, k + 1, \dots, i - 1, k, i + 1, \dots, n\}$ (Permutation, die i und k vertauscht).

$\mathbf{A}^{(k-1)} \leftarrow \mathbf{P}_{\pi_k} \mathbf{A}^{(k-1)}$

Bestimme Matrix \mathbf{L}_k (vgl. (2.5)) durch Berechnung der Faktoren

$$l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k + 1, \dots, n.$$

Setze $\mathbf{A}^{(k)} := \mathbf{L}_k \mathbf{A}^{(k-1)}$.

end for

Setze $\pi := \pi_{n-1} \circ \pi_{n-2} \circ \dots \circ \pi_1$, $\mathbf{P} := \mathbf{P}_\pi$.

Setze $\mathbf{R} := \mathbf{A}^{(n-1)}$.

Bemerkung 2.28 Der Algorithmus wird in der Praxis etwas anders realisiert: wie beim Fall ohne Pivotsuche, Algorithmus 2.8, operiert man direkt auf der Matrix \mathbf{A} und erhält am Schluss das Endschema \mathbf{R} anstelle von \mathbf{A} . Ausserdem wird wie in Algorithmus 2.10 der Linksdreiecksfaktor \mathbf{L} ebenfalls im unteren Teil von \mathbf{A} abgespeichert. Die Permutationsmatrix \mathbf{P} wird nicht explizit aufgestellt, sondern es wird nur ein Vektor mit natürlichen Zahlen zurückgegeben, der angibt, wie die Zeilen von \mathbf{A} permutiert werden.

Wir führen das Vorgehen an einem einfachen Beispiel vor.

Beispiel 2.29

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 2 \\ 2 & -7 & 2 \\ 1 & 24 & 0 \end{pmatrix}$$

Bei der Pivotsuche in der 1. Spalte, stossen wir auf die 2. Zeile. Man vertauscht also die 1. und 2. Zeile:

$$\mathbf{A}^{(0)} = \begin{pmatrix} 2 & -7 & 2 \\ 1 & 2 & 2 \\ 1 & 24 & 0 \end{pmatrix}$$

und führt dann den Eliminationsschritt durch. Es ist $l_{21} = 0.5$, $l_{31} = 0.5$ und damit

$$\mathbf{A}^{(1)} = \begin{pmatrix} 2 & -7 & 2 \\ 0 & 5.5 & 1 \\ 0 & 27.5 & -1 \end{pmatrix}.$$

Bei der Pivotsuche in der 2. Spalte müssen wir nun nur die Elemente $a_{22}^{(1)}$ und $a_{32}^{(1)}$ vergleichen. Da $a_{32}^{(1)}$ betragsmässig grösser ist als $a_{22}^{(1)}$, vertauschen wir die 2. und die 3. Zeile:

$$\mathbf{A}^{(1)} = \begin{pmatrix} 2 & -7 & 2 \\ 0 & 27.5 & -1 \\ 0 & 5.5 & 1 \end{pmatrix}.$$

Beim nächsten Eliminationsschritt entsteht $l_{32} = \frac{5.5}{27.5} = 0.2$. Wir erhalten als Endschema und als Matrix \mathbf{L}

$$\mathbf{R} = \mathbf{A}^{(2)} = \begin{pmatrix} 2 & -7 & 2 \\ 0 & 27.5 & -1 \\ 0 & 0 & 1.2 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0.2 & 1 \end{pmatrix}.$$

Es bleibt, die Permutationsmatrix \mathbf{P} zu bestimmen. In Algorithmus 2.27 werden die Permutationen π_1, π_2 aufgestellt mit

$$\begin{aligned} \pi_1(1) &= 2, & \pi_1(2) &= 1, & \pi_1(3) &= 3 \\ \pi_2(1) &= 1, & \pi_2(2) &= 3, & \pi_2(3) &= 2. \end{aligned}$$

Damit ergibt sich für $\pi = \pi_2 \circ \pi_1$:

$$\pi(1) = 3, \quad \pi(2) = 1, \quad \pi(3) = 2$$

und somit für die Permutationsmatrix \mathbf{P}_π :

$$\mathbf{P} = \mathbf{P}_\pi = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

Man hätte die Permutation π auch weniger formal durch Verfolgen der Zeilenvertauschungen erhalten können: die ursprünglich 1. Zeile ist zur 3. Zeile geworden (im ersten Schritt wurden Zeilen 1 und 2 vertauscht, in zweiten Schritt Zeilen 2 und 3), die ursprüngliche 2. Zeile ist zur 1. Zeile geworden, und die ursprünglich 3. Zeile ist am Schluss die 2. Zeile. Die Permutation π ist damit $\pi(1) = 3$, $\pi(2) = 1$, $\pi(3) = 2$. Man überzeugt sich davon, dass in der Tat $\mathbf{LR} = \mathbf{PA}$. \diamond

Gleichungssystemen werden mittels Algorithmus 2.27 wie folgt gelöst.

1. Bestimme $\mathbf{P}, \mathbf{L}, \mathbf{R}$ mittels Algorithmus 2.27 so, dass $\mathbf{PA} = \mathbf{LR}$.
2. Setze $\underline{b}' := \mathbf{P}\underline{b}$ und löse $\mathbf{L}\underline{y} = \underline{b}'$ mithilfe von Algorithmus 2.1.
3. Löse $\mathbf{R}\underline{x} = \underline{y}$ mithilfe von Algorithmus 2.2.

Im 2. Schritt wird die Multiplikation $\mathbf{P}\underline{b}$ nicht als Matrix-Vektor Multiplikation ausgeführt, sondern es werden natürlich nur die entsprechenden Einträge von \underline{b} vertauscht.

Der folgende Satz liefert den formalen Beweis, dass Algorithmus 2.27 tatsächlich die LR-Zerlegung einer Zeilenpermutation von \mathbf{A} liefert,

Satz 2.30 Zu jeder regulären Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ existiert eine Permutationsmatrix \mathbf{P}_π , so dass

$$\mathbf{L}\mathbf{R} = \mathbf{P}_\pi \mathbf{A}$$

eine $\mathbf{L}\mathbf{R}$ -Zerlegung ist. Zudem kann \mathbf{P}_π so gewählt werden, dass gilt

$$|l_{ij}| \leq 1 \quad \forall i, j \in \{1, \dots, n\}.$$

Beweis. Die $\mathbf{L}\mathbf{R}$ -Zerlegung und die Permutationsmatrix, deren Existenz im Satz behauptet wird, konstruieren wir mithilfe von Algorithmus 2.27. Im ersten Schritt von Algorithmus 2.27 wird (falls nötig) eine Zeilenvertauschung von zwei Zeilen durchgeführt, so dass die neue Matrix

$$\mathbf{A}^{(0)} := \mathbf{P}_{\pi_1} \mathbf{A},$$

so dass $a_{11}^{(0)}$ das betragsmässig grösste Element der ersten Spalte von $\mathbf{A}^{(0)}$ ist. Die Permutationsmatrix \mathbf{P}_{π_1} vermittelt dabei die Vertauschung der beiden Zeilen ($\mathbf{P}_{\pi_1} = \mathbf{I}$, falls keine Vertauschung nötig ist). Zudem ist $a_{11}^{(0)} \neq 0$, denn sonst wäre die erste Spalte identisch Null im Widerspruch zur Annahme, dass \mathbf{A} regulär ist. Weil $a_{11}^{(0)}$ das betragsgrösste Element in der ersten Spalte von $\mathbf{A}^{(0)}$ ist, gilt für die Einträge $l_{i1} = a_{i1}^{(0)}/a_{11}^{(0)}$, dass $|l_{i1}| \leq 1$. Wir erhalten also nach dem ersten Eliminationsschritt mit \mathbf{L}_1 gegeben durch (2.5):

$$\mathbf{A}^{(1)} = \mathbf{L}_1 \mathbf{A}^{(0)} = \mathbf{L}_1 \mathbf{P}_{\pi_1} \mathbf{A} = \left(\begin{array}{c|ccc} a_{11}^{(1)} & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \mathbf{B}^{(1)} \end{array} \right).$$

Aus der Regularität von \mathbf{L}_1 , \mathbf{P}_{π_1} und \mathbf{A} folgt also $0 \neq \det \mathbf{A}^{(1)} = a_{11}^{(1)} \det \mathbf{B}^{(1)}$. Mithin ist die Untermatrix $\mathbf{B}^{(1)}$ regulär. Wir können also mit Algorithmus 2.27 fortfahren und erhalten schliesslich

$$\mathbf{R} = \mathbf{A}^{(n-1)} = \mathbf{L}_{n-1} \mathbf{P}_{\pi_{n-1}} \mathbf{L}_{n-2} \mathbf{P}_{\pi_{n-2}} \cdots \mathbf{L}_1 \mathbf{P}_{\pi_1} \mathbf{A}, \quad (2.21)$$

wobei die Matrizen \mathbf{L}_k alle Einträge haben, die betragsmässig durch 1 beschränkt sind. Um die Matrizen \mathbf{L}_k von den Permutationsmatrizen \mathbf{P}_{π_k} zu separieren, schieben wir in der Darstellung von \mathbf{R} aus (2.21) zwischen die Faktoren \mathbf{L}_k und \mathbf{P}_{π_k} die Identität $\mathbf{P}_k^{-1} \mathbf{P}_k$, wobei die Permutationsmatrix \mathbf{P}_k gegeben ist durch: $\mathbf{P}_k := \mathbf{P}_{\pi_{n-1}} \mathbf{P}_{\pi_{n-2}} \cdots \mathbf{P}_{\pi_{k+1}}$ ($\mathbf{P}_{n-1} = \mathbf{I}$). Wir erhalten damit

$$\begin{aligned} \mathbf{R} &= \mathbf{L}_{n-1} \mathbf{P}_{n-1}^{-1} \mathbf{P}_{n-1} \mathbf{P}_{\pi_{n-1}} \mathbf{L}_{n-2} \mathbf{P}_{n-2}^{-1} \mathbf{P}_{n-2} \mathbf{P}_{\pi_{n-2}} \mathbf{L}_{n-3} \mathbf{P}_{n-3}^{-1} \mathbf{P}_{n-3} \mathbf{P}_{\pi_{n-3}} \\ &\quad \mathbf{L}_{n-4} \mathbf{P}_{n-4}^{-1} \mathbf{P}_{n-4} \cdots \mathbf{L}_1 \mathbf{P}_1^{-1} \mathbf{P}_1 \mathbf{P}_{\pi_1} \mathbf{A}. \end{aligned}$$

Weil $\mathbf{P}_k \mathbf{P}_{\pi_k} = \mathbf{P}_{k-1}$, ergibt sich mit der Abkürzung

$$\tilde{\mathbf{L}}_k := \mathbf{P}_k \mathbf{L}_k \mathbf{P}_k^{-1} \quad (2.22)$$

für \mathbf{R} :

$$\mathbf{R} = \tilde{\mathbf{L}}_{n-1} \tilde{\mathbf{L}}_{n-2} \cdots \tilde{\mathbf{L}}_1 \mathbf{P}_0 \mathbf{A}.$$

Da \mathbf{P}_0 als Verkettung von Permutationsmatrizen eine Permutationsmatrix ist, bleibt zu zeigen, dass das Produkt der Matrizen $\tilde{\mathbf{L}}_k$ tatsächlich eine Linksdreiecksmatrix ist. Sei $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ eine beliebige Permutation, die nur die Zahlen $\geq k+1$ permutiert (d.h. $\pi(j) = j$ für $j \leq k$). Dann überzeugt man sich davon, dass für eine Matrix \mathbf{L}_k von der Form (2.5) gilt:

$$\tilde{\mathbf{L}}_k = \mathbf{P}_\pi \mathbf{L}_k \mathbf{P}_\pi^{-1} = \mathbf{P}_\pi \mathbf{L}_k \mathbf{P}_\pi^\top = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{\pi_k(k+1),k} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{\pi_k(n),k} & & & 1 \end{pmatrix}. \quad (2.23)$$

Die oben eingeführten Permutationen \mathbf{P}_k sind genau von der Art, dass bei den zugehörige Permutationen der Zahlen 1 bis n nur die Zahlen $\geq k+1$ permutiert werden. Also haben die Matrizen $\tilde{\mathbf{L}}_k$ aus (2.22) die Darstellung (2.23). Daraus folgt, dass

$$\mathbf{L} := \tilde{\mathbf{L}}_1^{-1} \tilde{\mathbf{L}}_2^{-1} \cdots \tilde{\mathbf{L}}_{n-1}^{-1}$$

untere Dreiecksmatrix mit Einsen auf der Diagonale ist. Zusätzlich haben wir die explizite Darstellung

$$\mathbf{L} = \begin{pmatrix} 1 & & & & \\ l_{\pi_1(2),1} & 1 & & & \\ l_{\pi_1(3),1} & l_{\pi_2(3),2} & 1 & & \\ \vdots & \vdots & & \ddots & \\ l_{\pi_1(n),1} & l_{\pi_2(n),2} & \cdots & l_{\pi_{n-1}(n),n-1} & 1 \end{pmatrix}$$

Die Permutationsmatrix \mathbf{P}_0 gehört nach Definition zu einer Permutation der Zahlen 1 bis n , die gerade die Vertauschung der Zeilenindizes während Algorithmus 2.27 angibt. \square

Satz 2.15 über den Rückwärtsfehler beim Lösen von Gleichungssystemen verändert sich nicht durch Spaltenpivotisierung. Allerdings ist jetzt sichergestellt, dass die Einträge von \mathbf{L} betragsmässig nicht grösser als 1 sind. Über die Grösse der Einträge von \mathbf{R} lässt sich leider keine einfache Aussage treffen. In der Praxis gilt fast immer $\|\mathbf{R}\|_2 \lesssim \|\mathbf{A}\|_2$. Eine der seltenen Ausnahmen ist die **Wilkinson-Matrix**

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & 1 & \ddots & \vdots & 1 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ -1 & \cdots & -1 & 1 & 1 \\ -1 & \cdots & -1 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

deren $\mathbf{L}\mathbf{R}$ -Zerlegung gegeben ist durch

$$\mathbf{W} = \mathbf{L}\mathbf{R}, \quad l_{ij} = \begin{cases} 1 & \text{falls } i = j, \\ -1 & \text{falls } i > j, \\ 0 & \text{sonst,} \end{cases} \quad r_{ij} = \begin{cases} 1 & \text{falls } i = j, \\ 2^{i-1} & \text{falls } j = n, \\ 0 & \text{sonst.} \end{cases}$$

Beachte, dass Algorithmus 2.27 bei dieser Matrix keine Zeilen vertauscht. Die Einträge von \mathbf{R} wachsen trotzdem exponentiell mit n .

Bemerkung 2.31 (Vollpivotsuche) Die Spaltenpivotsuche bei der \mathbf{LR} -Zerlegung verursacht zusätzliche Kosten $O(n^2)$, die im Verhältnis zu den Gesamtkosten $O(n^3)$ relativ klein sind. (Übung: Geben Sie genau an, wieviele Vergleiche bei der Spaltenpivotsuche gemacht werden müssen.)

Alternativ zur Spaltenpivotsuche kann auch eine Vollpivotsuche durchgeführt werden. Dabei wird in jedem Schritt des Gauss-Algorithmus das betragsmässig grösste Element der Restmatrix gesucht und durch Zeilen- und Spaltenvertauschungen zum Pivot gemacht. Man erhält auf diese Weise eine Zerlegung

$$P_\pi \mathbf{A} P_{\pi'} = \mathbf{LR}$$

der Matrix \mathbf{A} , wobei die Permutationen π , π' die Zeilen- und Spaltenvertauschungen repräsentieren. Die Einträge der Faktoren \mathbf{L} , \mathbf{R} sind betragsmässig kleiner als bei Spaltenpivotsuche, so dass man erwartet, dass Gauss-Elimination mit Vollpivotsuche numerisch stabiler ist. Insbesondere wird z.B. die Instabilität für die Wilkinson-Matrix vermieden. Sie wird jedoch aus Kostengründen nur sehr selten eingesetzt, denn der zusätzliche Aufwand ist nun $O(n^3)$.

2.6 Nachiteration

Ist die Genauigkeit einer berechneten Lösung unbefriedigend, so lässt sich diese unter Umständen auf sehr einfache Weise verbessern. Wir nehmen folgende Situation an:

1. Der von unserem Löser berechnete Vektor $\hat{\mathbf{x}}$ erfüllt

$$\|\hat{\mathbf{b}} - \mathbf{A}\hat{\mathbf{x}}\|_2 \leq u_{\text{grob}} \|\mathbf{A}\|_2 \|\hat{\mathbf{x}}\|_2$$

für beliebige rechte Seiten $\hat{\mathbf{b}} \in \mathbb{R}^n$.

2. Das berechnete Residuum $\hat{\mathbf{r}} = \text{rd}(\hat{\mathbf{b}} - \mathbf{A}\hat{\mathbf{x}})$ erfüllt

$$\|\hat{\mathbf{r}} - \mathbf{r}\|_2 \leq u_{\text{fein}} \|\mathbf{A}\|_2 \|\hat{\mathbf{x}}\|_2$$

mit $u_{\text{fein}} \ll u_{\text{grob}}$.

Verschiedene Szenarien sind hier vorstellbar:

- Der eingesetzte Löser ist nicht rückwärtsstabil.
- Die zur Berechnung des Residuum $O(n^2)$ Operationen werden in sehr hoher (meist auch teurer) Genauigkeit ausgeführt (z.B. softwareseitig realisierte vierfache Genauigkeit).
- Der Lösung des Gleichungssystems wird in einfacher Genauigkeit (z.B. auf Graphikkarte oder cell processor) und nur das Residuum wird in doppelter Genauigkeit (z.B. auf Hauptprozessor) berechnet.

Eine "verbesserte" Lösung wird durch die Korrektur

$$\mathbf{x}_{\text{neu}} = \hat{\mathbf{x}} + \hat{\mathbf{x}}_{\text{korr}}, \quad (2.24)$$

erzielt, wobei $\hat{\mathbf{x}}_{\text{korr}}$ die berechnete Lösung der Korrekturgleichung $\mathbf{A}\hat{\mathbf{x}}_{\text{korr}} = \hat{\mathbf{r}}$ ist. Nach der ersten Annahme erfüllt diese $\|\hat{\mathbf{r}} - \mathbf{A}\hat{\mathbf{x}}_{\text{korr}}\|_2 \leq u_{\text{grob}} \|\mathbf{A}\|_2 \|\hat{\mathbf{x}}_{\text{korr}}\|_2$.

Wann ist \mathbf{x}_{neu} tatsächlich eine verbesserte Lösung? Um diese Frage zu klären, betrachten wir das Residuum:

$$\begin{aligned} \|\hat{\mathbf{b}} - \mathbf{A}\mathbf{x}_{\text{neu}}\|_2 &= \|\hat{\mathbf{b}} - \mathbf{A}(\hat{\mathbf{x}} + \hat{\mathbf{x}}_{\text{korr}})\|_2 = \|\hat{\mathbf{r}} - \hat{\mathbf{r}} + \hat{\mathbf{r}} - \mathbf{A}\hat{\mathbf{x}}_{\text{korr}}\|_2 \\ &\leq \|\hat{\mathbf{r}} - \hat{\mathbf{r}}\|_2 + \|\hat{\mathbf{r}} - \mathbf{A}\hat{\mathbf{x}}_{\text{korr}}\|_2 \\ &\leq u_{\text{fein}} \|\mathbf{A}\|_2 \|\hat{\mathbf{x}}\|_2 + u_{\text{grob}} \|\mathbf{A}\|_2 \|\hat{\mathbf{x}}_{\text{korr}}\|_2 \\ &\lesssim (u_{\text{fein}} + u_{\text{grob}}^2 \kappa(\mathbf{A})) \|\mathbf{A}\|_2 \|\hat{\mathbf{x}}\|_2, \end{aligned} \quad (2.25)$$

wobei in die letzte Ungleichung die Abschätzung

$$\|\hat{\mathbf{x}}_{\text{korr}}\|_2 \leq \|\mathbf{A}^{-1}\|_2 \|\hat{\mathbf{r}}\|_2 \approx \|\mathbf{A}^{-1}\|_2 \|\mathbf{r}\|_2 \leq u_{\text{grob}} \kappa(\mathbf{A}) \|\hat{\mathbf{x}}\|_2$$

eingegangen ist. Die Schranke (2.25) lässt die folgende Interpretation zu

Die Genauigkeit von \mathbf{x}_{neu} gegenüber $\hat{\mathbf{x}}$ wird verbessert, wenn $u_{\text{grob}} \kappa(\mathbf{A}) < 1$ gilt.

Ist $\kappa(\mathbf{A}) \approx 1$ und $u_{\text{fein}} = u_{\text{grob}}^2$, so reicht also ein Korrekturschritt aus, um die maximale, durch u_{fein} bestimmte Genauigkeit zu erreichen. Anderenfalls muss der Korrekturschritt wiederholt angewendet werden. Kombiniert mit der \mathbf{LR} -Zerlegung führt dies auf den folgenden Algorithmus.

Algorithmus 2.32 (Iterative Verbesserung)

Input: (Approximative) \mathbf{LR} -Zerlegung (allenfalls mit Pivotisierung) einer Matrix \mathbf{A} , rechte Seite $\hat{\mathbf{b}}$

Output: Lösung \mathbf{x} von $\mathbf{A}\mathbf{x} = \hat{\mathbf{b}}$.

- (i) Bestimme Lösung \mathbf{x} von $\mathbf{LR}\mathbf{x} = \mathbf{P}\hat{\mathbf{b}}$ mithilfe von Algorithmen 2.1, 2.2.
- (ii) Bestimme das Residuum $\mathbf{r} = \hat{\mathbf{b}} - \mathbf{A}\mathbf{x}$ (evtl. in erhöhter Rechengenauigkeit).
- (iii) Falls $\|\mathbf{r}\|_2$ grösser als vorgegebene Toleranz, finde Korrektur $\hat{\mathbf{x}}_{\text{korr}}$ als Lösung von $\mathbf{LR}\hat{\mathbf{x}}_{\text{korr}} = \mathbf{r}$ mithilfe von Algorithmen 2.1, 2.2.
- (iv) Setze $\mathbf{x} := \mathbf{x} + \hat{\mathbf{x}}_{\text{korr}}$ und gehe zu (ii).

Da wir bereits gesehen hatten, dass die Vorwärts- und Rückwärtssubstitution relativ billig im Vergleich zur \mathbf{LR} -Zerlegung sind, ist die Nachiteration vom Standpunkt des Aufwandes eine attraktive Möglichkeit, die Genauigkeit einer Lösung zu verbessern.

Ein Beispiel für den Einsatz von Algorithmus 2.32: Die \mathbf{LR} -Zerlegung mit Pivotisierung sowie Schritt (i) werden in *single precision* durchgeführt ($u_{\text{grob}} \approx 10^{-8}$). Das Matrix-Vektor-Produkt $\mathbf{A}\mathbf{x}$ sowie die Subtraktion $\hat{\mathbf{b}} - \mathbf{A}\mathbf{x}$ in Schritt (ii) werden in *double precision* durchgeführt ($u_{\text{grob}} \approx 10^{-16}$). Schritt (iii) wird wieder komplett in *single precision* durchgeführt. Die Addition $\mathbf{x} + \hat{\mathbf{x}}_{\text{korr}}$ in Schritt (iv) wird hingegen in *double precision* durchgeführt.

2.7 Cholesky-Zerlegung für SPD Matrizen

In Anwendungen hat die Matrix \mathbf{A} eines Gleichungssystems fast immer zusätzliche Eigenschaften, die zur effizienteren Lösung ausgenutzt werden können. Eine häufige

Eigenschaft ist positive Definitheit, die sich zum Beispiel aus physikalischen Modellen ergibt, bei denen der Ausdruck $\underline{x}^T \mathbf{A} \underline{x}$ eine Energie darstellt.

Definition 2.33 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Eine Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ heisst

1. **positiv definit**, falls $\underline{x}^T \mathbf{A} \underline{x} > 0 \quad \forall \underline{x} \in \mathbb{R}^n \setminus \{0\}$;
2. **positiv semi-definit**, falls $\underline{x}^T \mathbf{A} \underline{x} \geq 0 \quad \forall \underline{x} \in \mathbb{R}^n$.

Eine symmetrische, positiv definite Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ heisst kurz **SPD**.

Eine symmetrische Matrix ist genau dann positiv definit (bzw. semi-definit), wenn alle Eigenwerte positiv (bzw. nichtnegativ) sind. Einige einfachere notwendige Bedingungen finden sich im folgenden Satz.

Satz 2.34 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ SPD. Dann gilt:

1. \mathbf{A} ist regulär (invertierbar);
2. $a_{ii} > 0$ für alle $i \in \{1, \dots, n\}$;
3. $|a_{ij}| < \frac{1}{2}(a_{ii} + a_{jj})$ für $i \neq j$ und damit $\max_{ij} |a_{ij}| = \max_i a_{ii}$;
4. ist $\mathbf{X} \in \mathbb{R}^{n \times n}$ invertierbar, so ist $\mathbf{X}^T \mathbf{A} \mathbf{X}$ wieder SPD;
5. ist \mathbf{A} eine Blockdiagonalmatrix $\mathbf{A} = \text{diag}(\mathbf{A}_1, \mathbf{A}_2)$, so sind sowohl \mathbf{A}_1 als auch \mathbf{A}_2 SPD.

Beweis. Übung. \square

Als Spezialfall folgt aus Satz 2.34.4, dass $\mathbf{X}^T \mathbf{X}$ für jede invertierbare Matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ SPD ist. Im folgenden werden wir die Umkehrung zeigen: jede SPD Matrix lässt sich in der Form $\mathbf{X}^T \mathbf{X}$ schreiben.

Satz 2.35 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ SPD. Dann existiert eine rechte Dreiecksmatrix $\mathbf{R} \in \mathbb{R}^{n \times n}$, so dass $\mathbf{A} = \mathbf{R}^T \mathbf{R}$.

Beweis. Wir partitionieren die Matrix \mathbf{A} wie folgt:

$$\mathbf{A} = \left(\begin{array}{c|c} a_{11} & \underline{z}^T \\ \hline \underline{z} & \mathbf{B} \end{array} \right),$$

wobei $\underline{z}^T = (a_{12}, \dots, a_{1n})$. Im ersten Schritt der Gauss-Elimination erhält man

$$\mathbf{A}^{(1)} = \mathbf{L}_1 \mathbf{A} = \left(\begin{array}{c|c} a_{11} & \underline{z}^T \\ 0 & \mathbf{B}' \\ \vdots & \\ 0 & \end{array} \right), \quad \mathbf{L}_1 = \left(\begin{array}{cccc} 1 & & & \\ -l_{21} & 1 & & \\ \vdots & & \ddots & \\ -l_{n1} & & & 1 \end{array} \right), \quad (2.26)$$

wobei $l_{i1} = a_{i1}/a_{11}$. Man beachte, dass nach Satz 2.34 $a_{11} > 0$ gilt. Eine Rechnung zeigt nun, dass

$$\mathbf{L}_1 \mathbf{A} \mathbf{L}_1^T = \mathbf{A}^{(1)} \mathbf{L}_1^T = \left(\begin{array}{c|ccc} a_{11} & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \mathbf{B}' & \\ 0 & & & \end{array} \right).$$

Hier ist insbesondere die Matrix \mathbf{B}' unverändert aus (2.26) übernommen. Nach Satz 2.34 ist \mathbf{B}' wieder SPD. Mithin kann man dieselbe Argumentation für \mathbf{B}' wiederholen. Induktiv schliesst man dann, dass

$$\mathbf{L}_{n-1} \cdots \mathbf{L}_1 \mathbf{A} \mathbf{L}_1^T \mathbf{L}_2^T \cdots \mathbf{L}_{n-1}^T = \left(\begin{array}{cccc} a_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{nn} \end{array} \right) =: \mathbf{D}.$$

Nach Konstruktion sind die Matrizen \mathbf{L}_k , $k = 1, \dots, n-1$ gerade die Linksdreiecksmatrizen, die in Algorithmus 2.11 berechnet werden. Das hier vorgestellte Induktionsargument zeigt, dass der Algorithmus nicht abbricht, weil in jedem Eliminationsschritt das Pivotelement nicht verschwindet. Die Pivotelemente sind nach Satz 2.34 sogar strikt positiv! Mit $\mathbf{L} = (\mathbf{L}_{n-1} \cdots \mathbf{L}_1)^{-1}$ erhält man $\mathbf{A} = \mathbf{L} \mathbf{D} \mathbf{L}^T$. Durch Setzen von $\mathbf{D}^{1/2} := \text{diag}(\sqrt{a_{11}}, \sqrt{d_{22}}, \dots, \sqrt{d_{nn}})$ und $\mathbf{R} = \mathbf{D}^{1/2} \mathbf{L}^T$ ist die Behauptung bewiesen. \square

Ist \mathbf{A} SPD, dann ist die Zerlegung $\mathbf{A} = \mathbf{R}^T \mathbf{R}$, wobei \mathbf{R} rechte Dreiecksmatrix mit positiven Diagonaleinträgen, eindeutig und heisst **Cholesky-Zerlegung** von \mathbf{A} . Um die Cholesky-Zerlegung zu berechnen, könnte man zuerst die **LR**-Zerlegung von \mathbf{A} berechnen, und dann eine Diagonalskalierung wie im Beweis von Satz 2.35 vornehmen. Es geht aber auch einfacher und direkter.

Algorithmus 2.36 (Cholesky-Zerlegung)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$.

Output: Cholesky-Faktor \mathbf{R} , so dass $\mathbf{A} = \mathbf{R}^T \mathbf{R}$.

```

for  $j = 1, \dots, n$  do
  for  $i = 1, \dots, j-1$  do
     $r_{ij} = \left( a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj} \right) / r_{ii}$ 
  end for
   $r_{jj} = \left( a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2 \right)^{1/2}$ 
end for
```

Algorithmus 2.36 benötigt $1/3n^3 + O(n^2)$ flops und ist damit ungefähr halb so teuer wie die **LR**-Zerlegung. Die Reduktion um den Faktor 2 liegt daran, dass wegen der Symmetrie der Matrix und der Zerlegung nur ein Faktor der Zerlegung berechnet werden muss.

Bemerkung 2.37 Algorithmus 2.36 zur Bestimmung der Cholesky-Zerlegung kann auch dazu benutzt werden, zu prüfen, ob eine gegebene symmetrische Matrix positiv

definiert ist (ohne dass die Eigenwerte berechnet werden müssen). Man führt Algorithmus 2.36 durch; bricht er ab, weil eine Division durch Null auftritt oder weil eine Wurzel aus einer negativen Zahl gezogen werden soll, dann war die Matrix nicht SPD. Andernfalls ist sie SPD, vorausgesetzt, dass auch r_{nn} positiv ist.

Algorithmus 2.36 führt keine Pivottisierung durch. Die **LR**-Zerlegung mit Spaltenpivotsuche würde hingegen Zeilen vertauschen; bei einer SPD Matrix ist das Diagonalelement *nicht* notwendigerweise das grösste Element einer Spalte. Zum Beispiel ist

$$\begin{pmatrix} \theta^2 & \theta \\ \theta & 2 \end{pmatrix}$$

für jedes $\theta > 0$ SPD. Überraschenderweise ist Algorithmus 2.36 in Kombination mit Algorithmen 2.1, 2.2 trotzdem rückwärtsstabil, wie der folgende Satz zeigt.

Satz 2.38 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ SPD und $\hat{\mathbf{x}}$ die mittels Cholesky-Zerlegung berechnete Lösung von $\mathbf{A}\mathbf{x} = \mathbf{b}$. Dann gilt

$$(\mathbf{A} + \Delta\mathbf{A})\hat{\mathbf{x}} = \mathbf{b}, \quad \|\Delta\mathbf{A}\|_2 \leq 4n(3n+1)u\|\mathbf{A}\|_2.$$

Beweisskizze. Analog zu Satz 2.15 lässt sich zeigen:

$$(\mathbf{A} + \Delta\mathbf{A})\hat{\mathbf{x}} = \mathbf{b}, \quad |\Delta\mathbf{A}| \leq \gamma_{3n+1}|\hat{\mathbf{R}}^T|\hat{\mathbf{R}}| \leq 4(3n+1)|\mathbf{R}^T|\mathbf{R}|.$$

Die Behauptung folgt aus

$$\|\mathbf{R}^T\mathbf{R}\|_2 = \|\mathbf{R}\|_2^2 \leq n\|\mathbf{R}\|_2^2 = n\|\mathbf{A}\|_2,$$

wobei $\|\mathbf{A}\|_2 = \|\mathbf{R}^T\mathbf{R}\|_2 = \|\mathbf{R}\|_2^2$ aus der Singulärwertzerlegung von \mathbf{R} folgt. \square

2.8 LR-Zerlegung für Band-Matrizen

Die bisher entwickelten Algorithmen zur **LR**- oder Cholesky-Zerlegung gingen von einer *vollbesetzten* Matrix \mathbf{A} aus. In der Praxis (z.B. in der Strukturmechanik und bei der Diskretisierung von partiellen Differentialgleichungen) sind die auftretenden Matrizen oft *schwach besetzt* (engl. *sparse*), d.h. viele Einträge von \mathbf{A} sind gleich Null. Dies kann in zweierlei Hinsicht ausgenutzt werden:

1. Speichersparnis: Man speichert nicht die gesamte Matrix \mathbf{A} ab, sondern nur die von Null verschiedenen Einträge und deren Positionen ab.
2. Die Matrizen \mathbf{L} , \mathbf{R} der **LR**-Zerlegung von \mathbf{A} sind unter Umständen ebenfalls schwach besetzt. Auch hier kann Speicher und Rechenzeit eingespart werden, indem nur die nicht-trivialen Einträge von \mathbf{L} und \mathbf{R} berechnet werden.

Im folgenden illustrieren wir die zu erwartenden Ersparnisse am einfachsten nicht-trivialen Typ von schwach besetzten Matrizen vor: Bandmatrizen. Selbstverständlich decken diese nicht alle in der Praxis auftretenden Fälle von schwach besetzten Matrizen ab.⁸

⁸Siehe T. A. Davis. *Direct Methods for Sparse Linear Systems*, SIAM, 2006 für aktuelle Entwicklungen zur **LR**-Zerlegung von schwachbesetzten Matrizen.

Definition 2.39 Eine Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ heisst **Bandmatrix** mit **Bandbreite** $p+q+1$, falls es $p, q \in \mathbb{N} \cup \{0\}$ gibt, so dass

$$a_{ij} = 0 \quad \text{für } j > i + p \text{ oder } i > j + q.$$

Die Zahl p heisst die obere Bandbreite und q die untere Bandbreite.

Bandmatrizen haben also höchstens auf den p Nebendiagonalen über und auf den q Nebendiagonalen unter der Hauptdiagonalen nichtverschwindende Einträge:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,p+1} & 0 & \cdots & \cdots & 0 \\ a_{21} & a_{22} & \cdots & \cdots & a_{2,p+2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & & & & 0 \\ a_{q+1,1} & a_{q+1,2} & & \ddots & & & & 0 \\ 0 & a_{q+2,2} & & & & & & a_{n-p,n} \\ \vdots & 0 & \ddots & & & & & \vdots \\ \vdots & \vdots & \ddots & & & & & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,n-q} & \cdots & a_{n,n-1} & a_{nn} \end{pmatrix} \quad (2.27)$$

Um diese Matrix darzustellen, brauchen wir nur $(p+q+1)n - \frac{p(p+1)}{2} - \frac{q(q+1)}{2}$ reelle Zahlen abzuspeichern. Auch die **LR**-Zerlegung (ohne Spaltenpivotsuche!) einer Bandmatrix erbt die spezielle Struktur.

Satz 2.40 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ eine Bandmatrix mit oberer Bandbreite p und unterer Bandbreite q und existiere die **LR**-Zerlegung $\mathbf{LR} = \mathbf{A}$. Dann haben \mathbf{L} , \mathbf{R} Bandstruktur, und es gilt:

$$\begin{aligned} l_{ij} &= 0 & \text{falls } j > i \text{ oder } j < i - q \\ r_{ij} &= 0 & \text{falls } j < i \text{ oder } j > i + p. \end{aligned}$$

Beweisskizze. Die Aussage des Satzes folgt durch sorgfältige Untersuchung von Algorithmus 2.10. Man sieht recht einfach, dass die Aussage richtig ist für die erste Zeile von \mathbf{R} und die erste Spalte von \mathbf{L} . Dann schliesst man induktiv für die weiteren Zeilen/Spalten mithilfe von Algorithmus 2.10. \square

Satz 2.40 sagt aus, dass die Matrizen \mathbf{L} , \mathbf{R} der **LR**-Zerlegung ohne Pivottisierung(!) der Bandmatrix \mathbf{A} aus (2.27) folgende Struktur haben:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & & & & & 0 \\ l_{q+1,1} & l_{q+1,2} & & \ddots & & & & 0 \\ 0 & l_{q+2,2} & & & \ddots & & & \vdots \\ \vdots & \vdots & \ddots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & l_{n,n-q} & \cdots & l_{n,n-1} & 0 \\ 0 & 0 & \cdots & 0 & l_{n,n-q} & \cdots & l_{n,n-1} & 1 \end{pmatrix}$$

<i>LR</i> -Zerlegung (Algorithmus 2.41)	$\approx 2nqp$
Vorwärtssubst. (Alg. 2.1; Ausnutzen der Bandstruktur von \mathbf{L})	$\approx 2nq$
Rückwärtssubst. (Alg. 2.2; Ausnutzen der Bandstruktur von \mathbf{R})	$\approx 2np$
Gesamtkosten	$\approx 2n(pq + p + q)$

Tabelle 2.2. *Kosten beim Lösen von Gleichungssystemen mit Bandmatrizen*

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1,p+1} & 0 & \cdots & \cdots & 0 \\ 0 & r_{22} & \cdots & \cdots & r_{2,p+2} & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & & & & & 0 \\ \vdots & \vdots & \ddots & & & & & 0 \\ \vdots & \vdots & \ddots & & & & & r_{n-p,n} \\ \vdots & \vdots & \ddots & & & & & \vdots \\ \vdots & \vdots & \ddots & & & & & \vdots \\ \vdots & \vdots & \ddots & & & & & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & & r_{nn} \end{pmatrix}$$

Die Tatsache, dass die Matrizen \mathbf{L} und \mathbf{R} auch wieder schwach besetzt sind, wird bei Algorithmen zur Bestimmung von *LR*-Zerlegungen von Bandmatrizen ausgenutzt. Es brauchen insbesondere nur die nicht-trivialen Einträge von \mathbf{L} , \mathbf{R} berechnet zu werden. Dies führt auf die folgende Variante von Algorithmus 2.10, bei dem die beiden inneren Schleifen verkürzt werden können.

Algorithmus 2.41 (*LR*-Zerlegung für Bandmatrizen)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ mit oberer Bandbreite p und unterer Bandbreite q .

Output: Die Matrix \mathbf{A} wird mit ihrer *LR*-Zerlegung überschrieben.

```

for  $k = 1, \dots, n - 1$  do
  for  $i = k + 1, \dots, \min\{n, k + q\}$  do
     $a_{ik} := a_{ik} / a_{kk}$ 
    for  $j = k + 1, \dots, \min\{n, k + p\}$  do
       $a_{ij} := a_{ij} - a_{ik} a_{kj}$ 
    end for
  end for
end for

```

Die Bandstruktur von \mathbf{L} und \mathbf{R} wird ebenfalls bei der Vorwärts- und Rückwärts-substitution ausgenutzt. Die sich ergebenden Kosten der Algorithmen sind in Tabelle 2.2 zusammengestellt. Es wird vereinfachend $n \gg \max\{p, q\}$ angenommen.

Kapitel 3

Polynominterpolation

Das Ziel der *Polynominterpolation* ist für gegebene Datenpunkte

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array}$$

mit **Stützstellen** (oder auch **Knoten**) $x_0, \dots, x_n \in \mathbb{R}$ und Funktionswerten $y_0, \dots, y_n \in \mathbb{R}$ ein **Polynom**

$$p_n(x) = a_0 + a_1x + \cdots + a_nx^n = \sum_{k=0}^n a_kx^k, \quad a_0, a_1, \dots, a_n \in \mathbb{R}, \quad (3.1)$$

zu finden, so dass die **Interpolationsbedingungen**

$$p_n(x_i) = y_i, \quad i = 0, 1, \dots, n, \quad (3.2)$$

erfüllt sind. Schreiben wir (3.2) aus, so erhalten wir die $n+1$ in a_0, \dots, a_n linearen Gleichungen

$$\begin{aligned} a_0 + a_1x_0 + \cdots + a_nx_0^n &= y_0, \\ a_0 + a_1x_1 + \cdots + a_nx_1^n &= y_1, \\ &\vdots \\ a_0 + a_1x_n + \cdots + a_nx_n^n &= y_n. \end{aligned}$$

In Matrix-Vektor-Schreibweise ergibt sich das lineare Gleichungssystem:

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \quad (3.3)$$

Es lässt sich zeigen, dass (3.3) genau dann eindeutig lösbar ist, wenn $x_i \neq x_j$ für alle $i \neq j$ gilt. Dies werden wir nicht hier beweisen, sondern später auf einfacherem Weg die Lösbarkeit der Interpolationsaufgabe zeigen.

Mit der Lösung von (3.3) sind jetzt die Koeffizienten a_0, a_1, \dots, a_n bestimmt; es bleibt noch zu zeigen, wie das Polynom (3.1) an einer Stelle x_* ausgewertet wird.

Die naive Variante benötigt $n-1$ Potenzen. Effizienter ist das **Horner-Schema**, das auf einer geschickten Klammerung von (3.1) basiert:

$$p_n(x) = \left(\dots \left((a_nx + a_{n-1})x + a_{n-2} \right)x + a_{n-3} \right)x + \cdots + a_1 \right)x + a_0.$$

Die Auswertung dieses Ausdrucks von innen nach aussen führt auf den folgenden Algorithmus.

Algorithmus 3.1

Input: $a_0, \dots, a_n \in \mathbb{R};$
 $x_* \in \mathbb{R}.$

Output: $p_n(x_*) = y_*$
mit p wie
in (3.1).

```

 $y_* := a_n.$ 
for  $j = n-1, \dots, 0$ 
do
   $y_* = a_j + y_*x_*$ 
end for

```

```

MATLAB
function y = horner(a,x)
% Input: Vektor a mit Polynomkoeffizienten.
% Vektor x der Laenge k.
% Output: Vektor y mit Werten des Polynoms
% an den Stellen x(1), ..., x(k).
n = length(a)-1; k = length(x);
y(1:k,1) = a(n+1);
for j = n:-1:1,
  y = a(j) + y.*x;
end

```

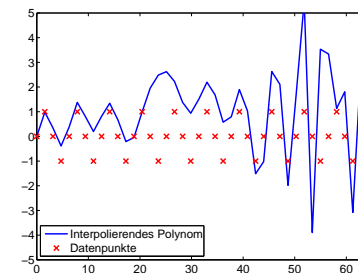
Die auf der rechten Seite aufgeführte MATLAB-Funktion wurde so geschrieben, dass sie das Polynom gleichzeitig an mehreren Stellen auswerten kann. Der Operator $.*$ führt dabei die elementweise Multiplikation zweier Vektoren durch. Siehe auch die MATLAB-Funktion `polyval`.⁹

Beispiel 3.2 Das folgende MATLAB-Skript interpoliert $\sin(x)$ im Intervall $[0, 20\pi]$.

```

MATLAB
n = 40;
x = linspace(0,20*pi,n+1)';
y = sin(x);
V = zeros(n+1);
for i = 1:n+1,
  V(:,i) = x.^(i-1);
end
a = V \ y;
f = horner(a,x);
plot(x,f,'b-',x,y,'rx');

```



Offensichtlich werden – wegen Rundungsfehlern – die Interpolationsbedingungen nicht erfüllt. Ursachen sind die hohe Konditionszahl von V ($\text{cond}(V) = 1.4e+73$) sowie die Größenunterschiede in den Koeffizienten ($a_1 \approx -7, a_n \approx 10^{-58}$). \diamond

Das in Beispiel 3.2 beobachtete Verhalten liegt an der unglücklich gewählten Formulierung des Interpolationsproblems. Die Menge \mathbb{P}_n aller Polynome vom Grad höchstens n bildet einen Vektorraum der Dimension $n+1$, siehe Beispiel 0.2. Das Polynom p wird in (3.1) bezüglich der Basis $\{1, x, \dots, x^n\}$ dargestellt. Wir können aber genausogut eine beliebige andere Basis

$$b_0(x), b_1(x), \dots, b_n(x)$$

von \mathbb{P}_n wählen. In dieser abstrakteren Formulierung lautet die Darstellung

$$p_n(x) = a_0b_0(x) + a_1b_1(x) + \cdots + a_nb_n(x)$$

⁹Die Koeffizienten werden bei `polyval` in umgekehrter Reihenfolge abgespeichert.

und die Interpolationsbedingungen führen auf das lineare Gleichungssystem

$$\begin{pmatrix} b_0(x_0) & b_1(x_0) & \cdots & b_n(x_0) \\ b_0(x_1) & b_1(x_1) & \cdots & b_n(x_1) \\ \vdots & \vdots & & \vdots \\ b_0(x_n) & b_1(x_n) & \cdots & b_n(x_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}. \quad (3.4)$$

3.1 Lagrange-Interpolation

Abgesehen von den in Beispiel 3.2 illustrierten numerischen Problemen, ist ein weiterer Nachteil der Monombasis, dass das Gleichungssystem (3.4) voll besetzt ist, also kostet dessen Lösung mittels **LR**-Zerlegung $O(n^3)$ flops.

Ein wesentlich einfacheres Gleichungssystem ziehen die **Lagrange-Polynome** nach sich:

$$\ell_k(x) = \frac{\prod_{\substack{i=0 \\ i \neq k}}^n (x - x_i)}{\prod_{\substack{i=0 \\ i \neq k}}^n (x_k - x_i)} \in \mathbb{P}_n, \quad k = 0, 1, \dots, n. \quad (3.5)$$

Es gilt offensichtlich

$$\ell_k(x_j) = \begin{cases} 1, & \text{für } j = k, \\ 0, & \text{sonst.} \end{cases} \quad (3.6)$$

Man beachte, dass die Lagrange-Polynome im Gegensatz zur Monombasis von den Stützstellen abhängen.

Satz 3.3 Sei $x_0, \dots, x_n \in \mathbb{R}$ mit $x_i \neq x_j$ für alle $i \neq j$. Dann bilden die in (3.5) definierten Lagrange-Polynome $\ell_0(x), \dots, \ell_n(x)$ eine Basis des \mathbb{P}_n .

Beweis. Da bereits $\dim(\mathbb{P}_n) = n + 1$ bekannt ist, bleibt lediglich zu zeigen, dass die Lagrange-Polynome linear unabhängig sind. Seien $a_0, \dots, a_n \in \mathbb{R}$, so dass

$$p_n(x) = a_0 \ell_0(x) + a_1 \ell_1(x) + \cdots + a_n \ell_n(x) \equiv 0.$$

Dann folgt wegen (3.6), dass $0 = p_n(x_j) = a_j$ und damit $a_0 = \cdots = a_n = 0$. \square

Wegen (3.6) wird das Gleichungssystem (3.4) trivial; die Systemmatrix ist die Identität! Also gilt $a_j = y_j$ für $j = 0, \dots, n$ und das interpolierende Polynom besitzt die Darstellung

$$p_n(x) = y_0 \ell_0(x) + y_1 \ell_1(x) + \cdots + y_n \ell_n(x). \quad (3.7)$$

Als Korollar folgt, dass die Polynominterpolationsaufgabe immer eindeutig lösbar ist, vorausgesetzt die Stützstellen sind paarweise verschieden.

Eine Anwendung der Interpolation ist der "Ersatz" einer komplizierten oder sogar unbekanntem Funktion f durch ein Polynom. Der folgende Satz liefert eine Aussage über den dabei auftretenden Approximationsfehler.

Satz 3.4 Seien $x_0 < x_1 < \cdots < x_n$ Stützstellen; sei weiter $x_* \in [x_0, x_n]$ beliebig, und

$$f \in C^{n+1}([x_0, x_n]). \quad (3.8)$$

Dann gilt für das Interpolationspolynom $p_n \in \mathbb{P}_n$ mit $p_n(x_i) = f(x_i)$ für $i = 0, \dots, n$ folgende Fehlerdarstellung:

$$E_n[f](x_*) := f(x_*) - p_n(x_*) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x_*), \quad (3.9)$$

mit einer (von x_* abhängigen) Zwischenstelle $\xi \in [x_0, x_n]$ und

$$\omega_{n+1}(x) := (x - x_0)(x - x_1) \cdots (x - x_n). \quad (3.10)$$

Beweis. Für $x_* = x_i$ gilt $E_n[f](x_i) = 0$. Sei nun $x_* \neq x_i$. Definiere für $x \in I := [x_0, x_n]$ die Funktion

$$g(x) := E_n[f](x) - \omega_{n+1}(x) E_n[f](x_*) / \omega_{n+1}(x_*). \quad (3.11)$$

Wegen $f \in C^{n+1}(I)$ und $\omega_{n+1} \in \mathbb{P}_{n+1}$ gilt $g \in C^{n+1}(I)$. Desweiteren besitzt g mindestens $n + 2$ verschiedene Nullstellen in I , denn

$$\begin{aligned} g(x_i) &= E_n[f](x_i) - \omega_{n+1}(x_i) E_n[f](x_*) / \omega_{n+1}(x_*) = 0, \quad i = 0, \dots, n, \\ g(x_*) &= E_n[f](x_*) - \omega_{n+1}(x_*) E_n[f](x_*) / \omega_{n+1}(x_*) = 0. \end{aligned}$$

Also hat $g' = \frac{dg}{dx}$ mindestens $n + 1 = n + 2 - 1$ Nullstellen, und $g^{(j)} = \frac{d^j g}{dx^j}$ hat mindestens $n + 2 - j$ Nullstellen für $j = 1, \dots, n + 1$. Also hat $g^{(n+1)}$ mindestens 1 Nullstelle. Sei ξ diese Nullstelle. Sie hängt, wie $g(x)$, von x_* ab, d.h. $\xi = \xi(x_*)$. Wegen $E_n^{(n+1)}[f](x) = f^{(n+1)}(x)$ und $\omega_{n+1}^{(n+1)}(x) \equiv (n+1)!$ folgt aus (3.11), dass

$$0 = g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)! E_n[f](x_*) / \omega_{n+1}(x_*).$$

Auflösen nach $E_n[f](x_*)$ gibt (3.9). \square

Für die naive Auswertung des Interpolationspolynoms $p_n(x)$ in der Lagrange-Darstellung (3.7) sind pro Auswertung $O(n^2)$ flops notwendig. Dieser Aufwand kann verringert werden, wenn vorher die folgenden Größen berechnet werden:

$$\lambda_j = \frac{1}{(x_j - x_0) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)}, \quad j = 0, \dots, n.$$

Dann lassen sich die Lagrange-Polynome wie folgt schreiben:

$$\ell_j(x) = \omega_{n+1}(x) \frac{\lambda_j}{x - x_j},$$

wobei $\omega_{n+1}(x)$ wie in (3.10) definiert ist. Es folgt

$$p_n(x) = \sum_{j=0}^n y_j \ell_j(x) = \omega_{n+1}(x) \sum_{j=0}^n \frac{\lambda_j y_j}{x - x_j}. \quad (3.12)$$

Pro Auswertung werden also nur noch $O(n)$ Operationen benötigt. Die Formel (3.12) lässt sich noch vereinfachen. Betrachten wir die Interpolation für $y_j \equiv 1$, so gilt natürlich $p_n \equiv 1$ und (3.12) reduziert sich auf

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

Einsetzen dieser Beziehung in (3.12) ergibt die **baryzentrische Interpolationsformel**

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

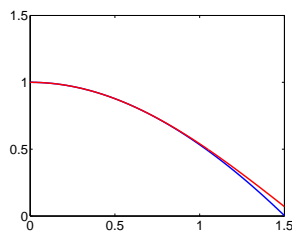
Beispiel Sei $x_0 = 0, x_1 = 0.2, x_2 = 0.4, x_3 = 0.6$ und $y_j = \cos(x_j)$. Für die Parameter der baryzentrischen Interpolationsformel ergibt sich

$$\begin{aligned} \lambda_0 &= \frac{1}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} = -20.8333, \\ \lambda_1 &= \frac{1}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} = 62.5000, \\ \lambda_2 &= \frac{1}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} = -62.5000, \\ \lambda_3 &= \frac{1}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} = 20.8333. \end{aligned}$$

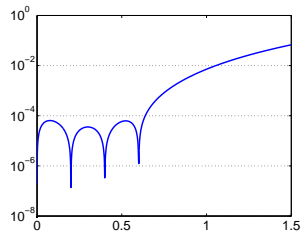
Das interpolierende Polynom in der Lagrange-Darstellung ist

$$\begin{aligned} p_3(x) &= y_0 \ell_0(x) + y_1 \ell_1(x) + y_2 \ell_2(x) + y_3 \ell_3(x) \\ &= y_0 \lambda_0 (x - x_1)(x - x_2)(x - x_3) + y_1 \lambda_1 (x - x_0)(x - x_2)(x - x_3) \\ &\quad + y_2 \lambda_2 (x - x_0)(x - x_1)(x - x_3) + y_3 \lambda_3 (x - x_0)(x - x_1)(x - x_2) \\ &= -20.8333(x - x_1)(x - x_2)(x - x_3) + 61.2542(x - x_0)(x - x_2)(x - x_3) \\ &\quad - 57.5663(x - x_0)(x - x_1)(x - x_3) + 17.1945(x - x_0)(x - x_1)(x - x_2). \end{aligned}$$

cos (rot) und $p_3(x)$ (blau)



Interpolationsfehler



3.2 Neville's Schema

Für die Auswertung des Interpolationspolynoms $p_n(x)$ an nur *einer* Stelle bietet sich als Alternative das folgende Resultat an.

Satz 3.5 (Aitken-Neville) Seien Datenpunkte $(x_i, y_i), i = 0, \dots, n$ gegeben mit paarweise verschiedenen x_j . Sei die Familie von Polynomen $P_{ik}(x) \in \mathbb{P}_k$ gegeben durch

$$P_{i0}(x) := y_i, \quad i = 0 : n \tag{3.13}$$

$$P_{ik}(x) := [(x - x_i) P_{i+1,k-1}(x) - (x - x_{i+k}) P_{i,k-1}(x)] / (x_{i+k} - x_i) \tag{3.14}$$

$$k = 1, \dots, n, \quad i = 0, \dots, n - k.$$

Dann gilt für das Interpolationspolynom $p_n(x)$ die Beziehung

$$p_n(x) = P_{0n}(x) \in \mathbb{P}_n.$$

Beweis. Die Beziehung (3.13) besagt, dass jedes $P_{i0} \in \mathbb{P}_0$, das ja ein Polynom vom Grad null ist, genau ein y_i "interpoliert". Sei nun $k = 1$. Wegen $P_{i0} \in \mathbb{P}_0$ folgt aus (3.14) sofort $P_{i1} \in \mathbb{P}_1, i = 0 : n - 1$. Weiter ist

$$P_{01}(x_0) = -(x_0 - x_1) P_{0,0}(x_0) / (x_1 - x_0) = P_{00}(x_0) = y_0,$$

$$P_{01}(x_1) = (x_1 - x_0) P_{1,0}(x_1) / (x_1 - x_0) = P_{10}(x_1) = y_1,$$

und, allgemeiner,

$$P_{i1}(x_i) = y_i, \quad P_{i1}(x_{i+1}) = y_{i+1}, \quad i = 0, \dots, n - 1.$$

Per Induktion nach k gilt für $P_{ik}(x) \in \mathbb{P}_k, k = 1, \dots, n$,

$$P_{ik}(x_j) = y_j, \quad j = i, \dots, i + k, \quad i = 0, \dots, n - k$$

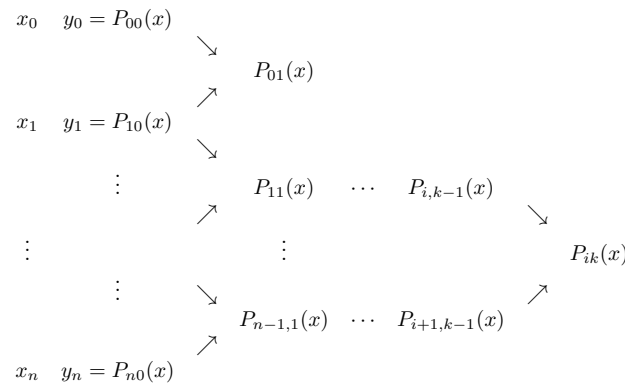
denn $P_{i,k-1}(x)$ interpoliert in $x_j, j = i, \dots, i + k - 1, P_{i+1,k-1}$ in x_j für $j = i + 1, \dots, i + k$. Damit gilt

$$P_{ik}(x_i) = -(x_i - x_{i+k}) P_{i,k-1}(x_i) / (x_{i+k} - x_i) = P_{i,k-1}(x_i) = y_i,$$

$$P_{ik}(x_{i+k}) = (x_{i+k} - x_i) P_{i+1,k-1}(x_{i+k}) / (x_{i+k} - x_i) = y_{i+k}.$$

Also interpoliert $P_{ik} \in \mathbb{P}_k$ in x_j für $j = i, \dots, i + k$. Für $k = n$ und $i = 0$ folgt die Behauptung. \square

Aus der Rekursionsformel von Satz (3.5) folgt sofort ein Algorithmus, das sogenannte **Aitken-Neville Schema**, zur Berechnung von $p_n(x)$.



Beispiel Sei $x_0 = 0, x_1 = 0.2, x_2 = 0.4, x_3 = 0.6$ und $y_j = \cos(x_j)$. Neville's Schema zur Auswertung an der Stelle $x = \pi/4$:

0	1.0000			
0.2	0.9801	0.9217		
0.4	0.9211	0.8074	0.6972	0.7059
0.6	0.8253	0.7366		

3.3 Newton-Interpolation

Seien wieder $(x_i, y_i), i = 0, \dots, n$ mit paarweise verschiedenen x_i gegeben und $p_n \in \mathbb{P}_n$ das zugehörige eindeutige interpolierende Polynom. Weiterhin bezeichne $p_{n-1} \in \mathbb{P}_{n-1}$ das in $(x_i, y_i), i = 0, \dots, n-1$ interpolierende Polynom. Wie bei Neville-Aitken wollen wir von p_{n-1} auf p_n schliessen. Ansatz:

$$p_n(x) = p_{n-1}(x) + q_n(x), \quad q_n(x) \in \mathbb{P}_n. \tag{3.15}$$

Dann gilt

$$q_n(x_i) = p_n(x_i) - p_{n-1}(x_i) = y_i - y_i = 0, \quad i = 0, \dots, n-1.$$

Damit ist q_n bis auf ein skalares Vielfaches eindeutig bestimmt; es gibt ein $a_n \in \mathbb{R}$, so dass

$$q_n(x) = a_n(x-x_0)(x-x_1)\dots(x-x_{n-1}) = a_n\omega_n(x). \tag{3.16}$$

Aus der verbleibenden Interpolationsbedingung $p_n(x_n) = y_n$ erhält man

$$p_{n-1}(x_n) + a_n\omega_n(x_n) = y_n \iff a_n = (y_n - p_{n-1}(x_n))/\omega_n(x_n). \tag{3.17}$$

Definition 3.6 Seien $f[x_0] = y_0, f[x_1] = y_1, \dots, f[x_n] = y_n$. Definiere rekursiv

$$f[x_i, x_{i+1}, \dots, x_{j-1}, x_j] = \frac{f[x_{i+1}, \dots, x_{j-1}, x_j] - f[x_i, x_{i+1}, \dots, x_{j-1}]}{x_j - x_i} \tag{3.18}$$

für $i < j$. Dann bezeichnet $f[x_0, \dots, x_k]$ mit $0 \leq k \leq n$ die k -te **dividierte Differenz** zu $(x_i, y_i), i = 0, \dots, n$.

Lemma 3.7 Sei a_n wie in (3.17) definiert. Dann gilt $a_n = f[x_0, \dots, x_n]$.

Beweis. Sei die Behauptung für $n-1$ Stützstellen erfüllt. Dann gilt einerseits

$$p_{n-1}(x) = \tilde{p}_{n-2}(x) + f[x_0, \dots, x_{n-1}](x-x_1)\dots(x-x_{n-1}),$$

wobei $\tilde{p}_{n-2}(x) \in \mathbb{P}_{n-2}$ das in x_1, \dots, x_{n-1} interpolierende Polynom bezeichnet.¹⁰ Auf der anderen Seite gilt auch

$$\tilde{p}_{n-1}(x) = \tilde{p}_{n-2}(x) + f[x_1, \dots, x_n](x-x_1)\dots(x-x_{n-1}),$$

¹⁰An dieser Stelle wird $f[x_1, \dots, x_{n-1}, x_0] = f[x_0, x_1, \dots, x_{n-1}]$ verwendet (Beweis Übung).

wobei $\tilde{p}_{n-1}(x) \in \mathbb{P}_{n-1}$ das in x_1, \dots, x_n interpolierende Polynom bezeichnet. Als Spezialfälle für $x = x_n$ folgen

$$p_{n-1}(x_n) = \tilde{p}_{n-2}(x_n) + f[x_0, \dots, x_{n-1}](x_n - x_1)\dots(x_n - x_{n-1}),$$

$$y_n = \tilde{p}_{n-1}(x_n) = \tilde{p}_{n-2}(x_n) + f[x_1, \dots, x_n](x_n - x_1)\dots(x_n - x_{n-1}).$$

Eingesetzt in (3.17) ergibt sich

$$a_n = \frac{y_n - p_{n-1}(x_n)}{\omega_n(x_n)} = \frac{f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]}{x_n - x_0},$$

und damit folgt die Behauptung. \square

Mit Lemma 3.7 folgt

$$p_n(x) = p_{n-1}(x) + f[x_0, \dots, x_n]\omega_n(x). \tag{3.19}$$

Wiederholtes Anwenden von (3.19) ergibt die **Newton'sche Interpolationsformel**

$$p_n(x) = \sum_{k=0}^n f[x_0, \dots, x_k]\omega_k(x), \tag{3.20}$$

wobei – wie oben –

$$\omega_k(x) = (x-x_0)(x-x_1)\dots(x-x_{k-1})$$

mit der Konvention $\omega_0(x) \equiv 1$. Es lässt sich ganz analog ein Horner-Schema entwickeln, das (3.20) in $O(n)$ flops pro Stützstelle auswertet.

Beispiel 3.8 Sei $n = 1$. Dann gilt

- Lagrange:** $P_1(x) = \frac{x-x_1}{x_0-x_1} y_0 + \frac{x-x_0}{x_1-x_0} y_1,$
- Neville-Aitken:** $P_1(x) = ((x-x_0)y_1 - (x-x_1)y_0)/(x_1-x_0) \quad \diamond$
- Newton:** $P_1(x) = y_0 + \frac{y_1-y_0}{x_1-x_0} (x-x_0)$

Die dividierten Differenzen lassen sich (ähnlich wie bei dem Neville-Schema) rekursiv mit einem Tableau, dem sogenannten dividierten Differenzenschema, berechnen:

x_0	$f[x_0]$			
x_1	$> f[x_0, x_1]$			
x_2	$> f[x_1, x_2]$	$> f[x_0, x_1, x_2]$		
x_3	$> f[x_2, x_3]$	$> f[x_1, x_2, x_3]$	$> f[x_0, x_1, x_2, x_3]$	

Dabei wird das Tableau, Spalte für Spalte, von links nach rechts mittels der Rekursion (3.18) aufgebaut.

Beispiel Sei $x_0 = 0, x_1 = 0.2, x_2 = 0.4, x_3 = 0.6$ und $y_j = \cos(x_j)$. Dividiertes Differenzenschema:

0	1.0000			
		-0.0995		
0.2	0.9801		-0.4888	
		-0.2950		0.0480
0.4	0.9211		-0.4600	
		-0.4790		
0.6	0.8253			

Das entsprechende Interpolationspolynom in der Newton-Darstellung:

$$p_3(x) = 1 - 0.0995(x - x_0) - 0.4888(x - x_0)(x - x_1) + 0.0480(x - x_0)(x - x_1)(x - x_2).$$

Mit den dividierten Differenzen lässt sich auch eine genaue Darstellung für den Interpolationsfehler in einer Stelle x_* finden. Sei p_n das in x_0, \dots, x_n interpolierende Polynom. Um den Fehler in $x_* \neq x_j$ zu bestimmen, interpolieren wir die Fehlerfunktion $E(x) = f(x) - p_n(x)$ an den Stützstellen x_0, \dots, x_n, x_* . Das entsprechende Polynom hat die Darstellung

$$\tilde{p}_{n+1}(x) = \left[\sum_{k=0}^n E[x_0, \dots, x_k] \omega_k(x) \right] + E[x_0, \dots, x_n, x_*] \omega_{n+1}(x).$$

Wegen $E(x_0) = \dots = E(x_n) = 0$ folgt $E[x_0, \dots, x_k] = 0$ für $k = 0, \dots, n$ und damit

$$\tilde{p}_{n+1}(x) = E[x_0, \dots, x_n, x_*] \omega_{n+1}(x). \tag{3.21}$$

Wegen der Linearität der dividierten Differenzen folgt

$$E[x_0, \dots, x_n, x_*] = f[x_0, \dots, x_n, x_*] - p_n[x_0, \dots, x_n, x_*] = f[x_0, \dots, x_n, x_*]$$

Im zweiten Schritt wurde benutzt, dass die $(n + 1)$ -te dividierte Differenz eines Polynoms vom Grad höchstens n immer verschwindet. Da $\tilde{p}_{n+1}(x_*) = E(x_*)$ ergibt sich mit (3.21) die folgende Fehlerdarstellung.

Lemma 3.9 Sei p_n das eine Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ in paarweise verschiedenen Stützstellen x_0, \dots, x_n interpolierende Polynom. Dann gilt für jedes $x_* \in \mathbb{R}$,

$$f(x_*) - p_n(x_*) = f[x_0, \dots, x_n, x_*] \omega_{n+1}(x_*).$$

3.4 Hermite-Interpolation

Bei der **Hermite-Interpolation** können zusätzlich zu den Funktionswerten von p_n noch dessen Ableitungen an den Stützstellen vorgeben werden. Natürlich muss sich dann der Grad des Polynoms entsprechend erhöhen. Dazu führen wir folgende praktische Notation ein. Wir lassen zu, dass in der Folge der Stützstellen

$$x_0 \leq x_1 \leq x_2 \leq \dots \leq x_n \tag{3.22}$$

Werte mehrfach auftreten. Sind an einer Stützstelle x_i der Funktionswert $y_i = f(x_i)$ und die Ableitungen $f'(x_i), \dots, f^{(k)}(x_i)$ bis zu einem Grad k gegeben, so soll x_i

in der Folge gerade $(k + 1)$ -mal auftauchen. Gleiche Stützstellen werden wie folgt durchnummeriert:

$$d_i = \max\{j : x_i = x_{i-j}\}.$$

Damit definieren wir die linearen Abbildungen

$$\mu_i : C^n([x_0, x_n]) \rightarrow \mathbb{R}, \quad \mu_i(f) = f^{(d_i)}(x_i)$$

und die Hermite-Interpolationsaufgabe lautet formal wie folgt:

$$\mu_i(p_n) = \mu_i(f), \quad i = 0, \dots, n. \tag{3.23}$$

Satz 3.10 Zu jeder Funktion $f \in C^n([t_0, t_n])$ und jeder Folge $x_0 \leq x_1 \leq x_2 \leq \dots \leq x_n$ gibt es genau ein Polynom $p_n \in \mathbb{P}_n$, so dass (3.23) erfüllt ist.

Beweis. Die Abbildung $\mu : \mathbb{P}_n \rightarrow \mathbb{R}^{n+1}$ mit $\mu : p \mapsto (\mu_0(p), \dots, \mu_n(p))$ ist offensichtlich linear. Sie ist auch injektiv, denn aus $\mu(p) = 0$ folgt, dass p mindestens $n + 1$ Nullstellen besitzt (mit Vielfachheit gezählt), also folgt $p \equiv 0$. Eine lineare injektive Abbildung zwischen gleich-dimensionalen Räumen ist aber immer auch bijektiv. \square

Die Idee der Newton-Interpolation lässt sich auf einfache Weise auf die Hermite-Interpolation verallgemeinern. Mit der mehrfachen Zählung der Stützstellen in (3.22), wählt man als Basis-Funktionen wieder

$$\omega_j(x) = (x - x_0)(x - x_1) \dots (x - x_{j-1}), \quad j = 0, \dots, n,$$

mit der Konvention $\omega_0(x) \equiv 1$, und stellt p_n dar als

$$p_n(x) = \sum_{j=0}^n f[x_0, x_1, \dots, x_j] \omega_j(x).$$

Hierbei muss aber die Definition der dividierten Differenz $f[x_i, \dots, x_j]$ erweitert werden. Gilt $x_i \neq x_j$, so ist $f[x_i, \dots, x_j]$ im Sinne von Definition 3.6 zu verstehen. Gilt allerdings $x_i = x_{i+1} = \dots = x_j$, so setzen wir

$$f[x_i, \dots, x_j] = \frac{f^{(j-i)}(x_i)}{(j-i)!} \tag{3.24}$$

Im dividierte Differenzenschema wird einfach an den entsprechenden Stellen die Ableitung geteilt durch $(j - i)!$ eingetragen.

Bemerkung 3.11 Für den Spezialfall $x_0 = \dots = x_n$ hat das Hermite-Interpolationspolynom wegen (3.24) die Form einer abgebrochenen Taylor-Reihe um $x = x_0$:

$$p_n(x) = \sum_{j=0}^n \frac{f^{(j)}(x_0)}{j!} (x - x_0)^j.$$

Beispiel 3.12

$f(0)$	$f(1/2)$	$f'(1/2)$	$f''(1/2)$	$f(1)$
1	3/2	1/2	0	5/2

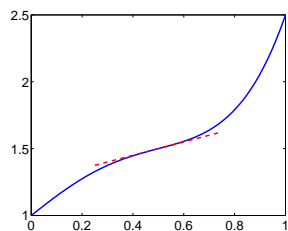
Das interpolierende Polynom hat die Form

$$p_4(x) = f[0] + f[0, 1/2]x + f[0, 1/2, 1/2]x(x-1/2) + f[0, 1/2, 1/2, 1/2]x(x-1/2)^2 + f[0, 1/2, 1/2, 1/2, 1]x(x-1/2)^3.$$

Die dividierten Differenzen ergeben sich aus der oberen Reihe des Schemas (vorgegebene Daten sind fettgedruckt):

0	1				
1/2	3/2	> 1			
1/2	3/2	> 1/2	> -1		
1/2	3/2	> 1/2	> 2	> 4	
1/2	3/2	> 1/2	> 0	> 6	
1	5/2	> 2	> 3		

Also gilt $p_4(x) = 1 + x - x(x-1/2) + 2x(x-1/2)^2 + 4x(x-1/2)^3$.



3.5 Approximation und Kondition der Interpolation

Wir betrachten jetzt die Approximationsgüte des Interpolationspolynoms p_n in den Stützstellen

$$a := x_0 < x_1 < \dots < x_n := b \tag{3.25}$$

zu einer gegebenen Funktion $f \in C^0([a, b])$. Dazu bezeichnen wir die Abbildung von f auf das eindeutig bestimmte Interpolationspolynom p_n ($p_n(x_i) = f(x_i)$ für $i = 0, \dots, n$) mit

$$I_n : C^0([a, b]) \rightarrow \mathbb{P}_n, \quad I_n[f] = p_n. \tag{3.26}$$

Wie messen den Approximationsfehler in der L_∞ -Norm auf $[a, b]$:

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

Aus Satz 3.4 folgt sofort die Abschätzung

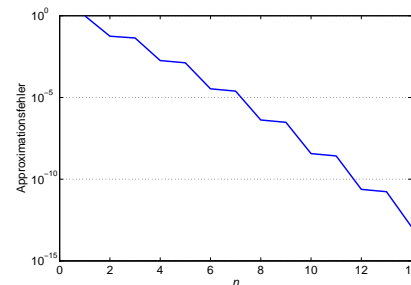
$$\|f - I_n[f]\|_\infty \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \|\omega_{n+1}\|_\infty, \tag{3.27}$$

vorrausgesetzt natürlich, dass zusätzlich $f \in C^{n+1}([a, b])$ gilt. Ob $\|f - I_n[f]\|_\infty \rightarrow 0$ für $n \rightarrow \infty$ gilt, hängt wesentlich vom Verhalten der Ableitungen $f^{(n+1)}$ ab.

Beispiel 3.13 Sei $f(x) = \sin x$ und $[a, b] = [0, \pi]$. Wähle äquidistante Stützstellen $x_j = j\pi/n$ für $j = 0, \dots, n$. Dann gilt wegen $\|f^{(n+1)}\|_\infty = 1$:

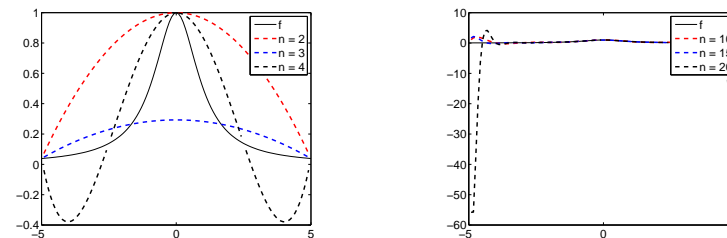
$$\begin{aligned} \|f - I_n[f]\|_\infty &\leq \frac{1}{(n+1)!} \max_{x \in [0, \pi]} \left| (x-0)\left(x-\frac{\pi}{n}\right)\left(x-2\frac{\pi}{n}\right) \cdots (x-\pi) \right| \\ &\leq \frac{1}{n+1} \left(\frac{\pi}{n}\right)^{n+1} \xrightarrow{n \rightarrow \infty} 0. \end{aligned}$$

Die folgende Abbildung zeigt den tatsächlichen Verlauf von $\|f - I_n[f]\|_\infty$ für $n = 1, \dots, 14$:



Die beobachtete Konvergenz entspricht qualitativ der angegebenen Schranke. ◇

Beispiel 3.14 (Runge) Sei $f(x) = (1+x^2)^{-1}$ und $[a, b] = [-5, 5]$. Wähle äquidistante Stützstellen $x_j = 5 + \frac{10}{n}j$ für $j = 0, \dots, n$. Die folgenden Abbildungen zeigen die interpolierenden Polynome $I_n[f]$ für $n = 2, 3, 4, 10, 15, 20$.



Offenbar divergiert der Fehler an den Rändern des Intervalls! Das Phänomen erklärt sich aus dem Wachstum der Ableitungen von f . ◇

Beispiel 3.14 steht nur scheinbar im Widerspruch zum Satz von Weierstrass (konstruktiv bewiesen durch Bernstein), der besagt, dass sich jede stetige Funktion $f \in C^0([a, b])$ beliebig genau gleichmäßig durch ein Polynom approximieren lässt. Oder formal ausgedrückt:

$$\inf_{q \in \mathbb{P}_n} \|f - q\|_\infty \xrightarrow{n \rightarrow \infty} 0.$$

Es stellt sich die Frage, wie weit die Polynominterpolation für eine gegebene Folge von Stützstellen von dieser Bestapproximation entfernt ist. Dazu betrachten wir im folgenden die **Kondition der Polynominterpolation**.

Wir schreiben

$$I_n[f](x) = \sum_{k=0}^n f(x_k) \ell_k(x)$$

mit den in (3.5) definierten Lagrange-Polynomen. Der Interpolationsoperator I_n ist linear:

$$I_n[\alpha f + \beta g] = \alpha I_n[f] + \beta I_n[g], \quad \alpha, \beta \in \mathbb{R}, \quad f, g \in C^0([a, b]). \quad (3.28)$$

Für eine “kleine” Störung $\delta \in C^0([a, b])$ gilt also für $\hat{f} = f + \delta$:

$$\begin{aligned} \|I_n[\hat{f}] - I_n[f]\|_\infty &= \|I_n[\delta]\|_\infty = \left\| \sum_{k=0}^n \delta(x_k) \ell_k(x) \right\|_\infty \\ &\leq \max_{0 \leq k \leq n} |\delta(x_k)| \left\| \sum_{k=0}^n |\ell_k(x)| \right\|_\infty \\ &\leq \|\delta\|_\infty \left\| \sum_{k=0}^n |\ell_k(x)| \right\|_\infty. \end{aligned} \quad (3.29)$$

Die Störung δ im “Input” f kann also bei der Polynominterpolation durch den Faktor $\left\| \sum_{k=0}^n |\ell_k(x)| \right\|_\infty$ verstärkt werden. Dies motiviert die folgende Definition.

Definition 3.15 Seien $a = x_0 < x_1 < \dots < x_n = b$ Stützstellen und $\ell_j(x) \in \mathbb{P}_n$ die entsprechenden Lagrange-Polynome. Dann heisst

$$\Lambda_n = \Lambda_n(\{x_0, \dots, x_n\}) := \left\| \sum_{k=0}^n |\ell_k(x)| \right\|_\infty \quad (3.30)$$

Lebesgue-Konstante von $\{x_0, \dots, x_n\}$.

Wegen (3.29) ist die Lebesgue-Konstante ein Mass für die absolute Kondition der Polynominterpolation. Darüber hinaus ist sie auch ein Mass für die Entfernung der Polynominterpolation vom Ideal der Bestapproximation.

Satz 3.16 Es gilt für $a = x_0 < x_1 < x_2 < \dots < x_n = b$ die Quasioptimalität

$$\|f - I_n[f]\|_\infty \leq (1 + \Lambda_n) \inf_{q \in \mathbb{P}_n} \|f - q\|_\infty. \quad (3.31)$$

Beweis. Dies ist eine direkte Folgerung aus (3.29):

$$\begin{aligned} \|f - I_n[f]\|_\infty &= \|f - q + q - I_n[f]\|_\infty \\ &= \|(f - q) - I_n[f - q]\|_\infty \\ &\leq \|f - q\|_\infty + \|I_n[f - q]\|_\infty \\ &\leq \|f - q\|_\infty + \left\| \sum_{j=0}^n |\ell_j(x)| \right\|_\infty \|f - q\|_\infty \\ &= (1 + \Lambda_n) \|f - q\|_\infty. \end{aligned}$$

□

Für $n = 1, 2, 3, \dots$ und äquidistante Stützstellen $x_i = a + i(b - a)/n$, $i = 0, \dots, n$ gilt

$$\Lambda_n \cong \frac{2^{n+1}}{en \log n}, \quad n \rightarrow \infty.$$

Die Λ_n können also – wie in Beispiel 3.14 beobachtet – für äquidistante Stützstellen sehr stark wachsen.

3.6 Tschebyscheff-Interpolation

Im folgenden werden wir Folgen von Stützstellen konstruieren, die wesentlich niedrigere Lebesgue-Konstanten als äquidistante Stützstellen nach sich ziehen. Den Einfluss der Stützstellen in den Approximationsfehler lässt sich am einfachsten in Abschätzung (3.27) quantifizieren. Es scheint vielversprechend, die Stützstellen so zu wählen, dass

$$\|\omega_{n+1}\|_\infty = \max_{x \in [a, b]} |(x - x_0)(x - x_1) \cdots (x - x_n)|$$

minimiert wird. (Da die kritischen Bereiche von ω_{n+1} an den Rändern des Intervalls liegen, wird man erwarten, dass dies nur über eine stärkere Konzentration der Stützstellen an den Rändern erfolgen kann.)

Es wird sich zeigen, dass die Nullstellen der im folgenden eingeführten Tschebyscheff-Polynome $\|\omega_{n+1}\|_\infty$ minimieren. Um die Diskussion zu vereinfachen ziehen wir uns mittels einer affinen Abbildung auf das Intervall $[-1, 1]$ zurück:

$$t : [a, b] \rightarrow [-1, 1], \quad t : x \mapsto \tilde{x} = \frac{2x - a - b}{b - a}.$$

Deren Umkehrabbildung hat die Form

$$t^{-1} : [-1, 1] \rightarrow [a, b], \quad t^{-1} : \tilde{x} \mapsto x = \frac{1 - \tilde{x}}{2}a + \frac{1 + \tilde{x}}{2}b. \quad (3.32)$$

Definition 3.17 Seien $T_0(x) = 1$ und $T_1(x) = x$. Dann ist das k -te Tschebyscheff-Polynom $T_k(x)$ rekursiv definiert als

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x).$$

Die folgenden Eigenschaften der Tschebyscheff-Polynome sind leicht nachzuprüfen.

Lemma 3.18

1. Die Tschebyscheff-Polynome haben ganzzahlige Koeffizienten.
2. Der höchste Koeffizient von T_n ist $a_n = 2^{n-1}$.
3. T_n ist eine gerade Funktion, falls n gerade, und eine ungerade Funktion, falls n ungerade.
4. Es gilt $T_k(x) = \cos(k \arccos x)$ für $-1 \leq x \leq 1$.

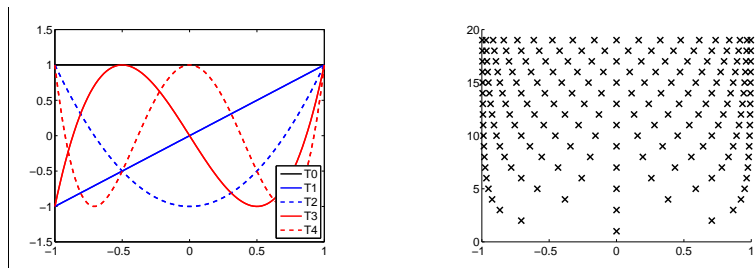


Abbildung 3.1. Links: Die Tschebyscheff-Polynome T_0, \dots, T_4 . Rechts: Die Nullstellen der Tschebyscheff-Polynome T_1, \dots, T_{20} .

5. $|T_n(x)| \leq 1$ für $x \in [-1, 1]$.
6. $|T_n(x)|$ nimmt den Wert 1 an den Punkten $\bar{x}_j = \cos(j\pi/n)$ für $j = 0, \dots, n$ an.
7. Die Nullstellen von $T_n(x)$ sind

$$x_j = \cos\left(\frac{2j-1}{2n}\pi\right), \quad j = 1, \dots, n. \tag{3.33}$$

Der folgende Satz beschreibt die für uns wichtigste Eigenschaft der Tschebyscheff-Polynome.

Satz 3.19 Das Tschebyscheff-Polynom T_n hat minimale L_∞ -Norm auf dem Intervall $[-1, 1]$ unter allen Polynomen vom Grad n mit höchstem Koeffizienten $a_n = 2^{n-1}$.

Beweis. Der Beweis erfolgt über Widerspruch. Sei $p_n \in \mathbb{P}_n$ ein Polynom mit höchstem Koeffizienten $a_n = 2^{n-1}$ und nehmen wir an, dass $\|p_n\|_\infty < \|T_n\|_\infty = 1$ gilt. Dann ist $T_n - p_n$ ein Polynom vom Grad höchstens $n - 1$. Wegen Lemma 3.18 gilt an den Punkten $\bar{x}_j = \cos(j\pi/n)$:

$$\begin{aligned} T_n(\bar{x}_{2k}) = 1, \quad p_n(\bar{x}_{2k}) < 1 &\implies p_n(\bar{x}_{2k}) - T_n(\bar{x}_{2k}) < 0, \\ T_n(\bar{x}_{2k+1}) = -1, \quad p_n(\bar{x}_{2k+1}) > -1 &\implies p_n(\bar{x}_{2k+1}) - T_n(\bar{x}_{2k+1}) > 0. \end{aligned}$$

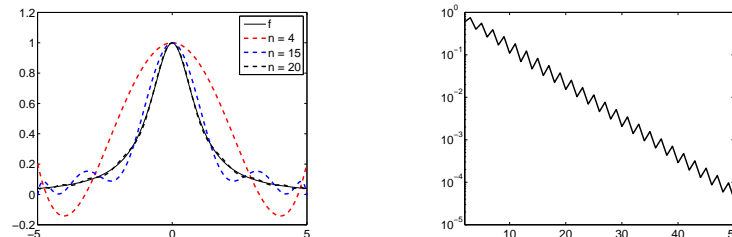
Damit ist $T_n - p_n$ an den $n + 1$ Punkten $\bar{x}_j, j = 0, \dots, n$, abwechselnd positiv und negativ, und hat daher mindestens n Nullstellen in $[-1, 1]$. Dies ist ein Widerspruch, da der Grad von $T_n - p_n$ höchstens $n - 1$ ist. \square

Als Korollar von Satz 3.19 folgt, dass die Nullstellen x_0, \dots, x_n des $(n + 1)$ -ten Tschebyscheff-Polynoms T_{n+1} den Ausdruck

$$\|\omega_{n+1}\|_\infty = \max_{x \in [-1, 1]} |(x - x_0)(x - x_1) \cdots (x - x_n)|$$

minimieren. Für ein allgemeines Intervall $[a, b]$ sind die Stützstellen mittels der in (3.32) definierten Abbildung t^{-1} zu transformieren.

Beispiel 3.20 (Fortsetzung von Beispiel 3.14) Sei $f(x) = (1+x^2)^{-1}$ und $[a, b] = [-5, 5]$. Wähle als Stützstellen $x_j = 5 \cos\left(\frac{2j+1}{2n+2}\pi\right)$ mit $j = 0, \dots, n$. Die folgenden Abbildungen zeigen die interpolierenden Polynome $I_n[f]$ für $n = 4, 10, 20$ sowie den Fehler $\|f - I_n[f]\|_\infty$ für $n = 2, \dots, 50$.



Offenbar konvergiert nun $I_n[f]$ gegen f . \diamond

Im allgemeinen ist $I_n[f]$ selbst mit den Tschebyscheff-Nullstellen als Stützstellen nicht die beste Polynomapproximation von f . Allerdings kann man für die Lebesgue-Konstante zeigen¹¹:

$$\Lambda_n < \frac{2}{\pi} \log(1+n) + 1. \tag{3.34}$$

Damit ist die Approximationsgüte für nicht allzu grosse n nur unwesentlich vom Ideal ($\Lambda_n = 1$) entfernt.

Bemerkung 3.21 Um wirklich die beste Polynomapproximation zu finden, verwendet man den iterativen Remez-Algorithmus, der allerdings auf keine geschlossene Darstellung für das entsprechende Polynom führt. In Anbetracht von (3.34) kann der Remez-Algorithmus die Approximationsgüte nur noch unwesentlich verbessern; er wird deshalb in der numerischen Analysis fast nie verwendet.

3.7 Spline-Interpolation

Trotz der durch die Tschebyscheff-Interpolation erzielte Verbesserung der Approximationseigenschaft, bleibt die Approximation einer Funktion $f : [a, b] \rightarrow \mathbb{R}$ durch Polynome hohen Grades aus verschiedenen Gründen problematisch:

- Die angegebenen a priori Fehlerschranken stellen hohe Glattheitsforderungen an f .
- Die Konvergenzgeschwindigkeit $I_n[f] \rightarrow f$ ist oft unbefriedigend.
- Zu jedem fest vorgegebene Schema von Stützstellenfolgen $x_0^{(k)}, \dots, x_k^{(k)}, k = 1, 2, 3, \dots$ gibt es eine stetige Funktion f , so dass $\|f - I_k[f]\|_\infty$ divergiert.
- Polynome von hohem Grad oszillieren stark. Dies wirkt sich bei vielen graphischen Anwendungen unangenehm aus (z.B. Computer-Graphik, Computer aided design (CAD)).

¹¹Siehe zum Beispiel Q. Chen und I. Babuška: Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. Comput. Methods Appl. Mech. Engrg. 128 (1995), no. 3-4, 405-417.

Mit einer einfachen Idee lassen sich diese Probleme vermeiden: Anstatt den Polynomgrad zu erhöhen unterteilt man das Intervall $[a, b]$ in viele Teilintervalle und interpoliert auf jedem Teilintervall. Ganz so einfach ist die Vorgehensweise leider nicht; man würde Funktionen erhalten, deren Ableitungen an den Anschlussstellen unstetig sind. Diese Unstetigkeiten in der Ableitung, wie sie z.B. bei stückweise linearer Interpolation auftritt (siehe Abb. 3.2), sind bei vielen Anwendungen äusserst störend. Die Grundidee der **Spline-Interpolation** besteht nun darin, auf jedem

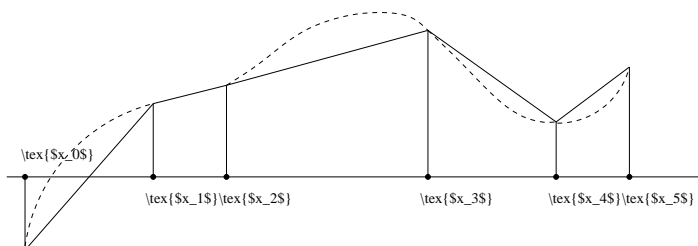


Abbildung 3.2. Stückweise lineare Interpolationsfunktion; an den Stützstellen ist diese Funktion nicht differenzierbar.

Teilintervall ein Polynom zu verwenden, dessen Grad “unnötig” hoch ist. Die dadurch gewonnenen Freiheiten werden wir – ähnlich wie bei der Hermite-Interpolation – dazu verwenden, eine gewisse Differenzierbarkeit an den Stützstellen zu erzwingen.

Vorerst werden wir nur ein Teilintervall $[x_{j-1}, x_j]$ mit $x_{j-1} < x_j$ betrachten und ein Polynom s_j suchen, so dass

$$s_j(x_{j-1}) = y_{j-1}, \quad s_j(x_j) = y_j, \quad s'_j(x_{j-1}) = c_{j-1}, \quad s'_j(x_j) = c_j \quad (3.35)$$

für vorgegebene $y_{j-1}, y_j, c_{j-1}, c_j \in \mathbb{R}$ erfüllt ist. Aus Abschnitt 3.4 zur Hermite-Interpolation wissen wir bereits, dass es genau ein solches $s_j \in \mathbb{P}_3$ gibt. Im Gegensatz zu Abschnitt 3.4 werden wir eine Lagrange-ähnliche Darstellung von s_j bevorzugen und die folgende Basis wählen:

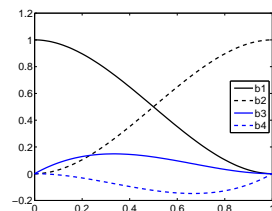
$$b_1(x) = \phi\left(\frac{x_j-x}{h_j}\right) \quad b_2(x) = \phi\left(\frac{x-x_{j-1}}{h_j}\right) \\ b_3(x) = -h_j\psi\left(\frac{x_j-x}{h_j}\right) \quad b_4(x) = h_j\psi\left(\frac{x-x_{j-1}}{h_j}\right)$$

mit

$$h_j = x_j - x_{j-1}, \quad \phi(\xi) = 3\xi^2 - 2\xi^3, \quad \psi(\xi) = \xi^3 - \xi^2.$$

Die rechte Abbildung zeigt die 4 Basisfunktionen für $x_{j-1} = 0, x_j = 1$. Es gilt offenbar

$$b_1(x_{j-1}) = 1, \quad b_1(x_j) = 0, \quad b'_1(x_{j-1}) = 0, \quad b'_1(x_j) = 0, \\ b_2(x_{j-1}) = 0, \quad b_2(x_j) = 1, \quad b'_2(x_{j-1}) = 0, \quad b'_2(x_j) = 0, \\ b_3(x_{j-1}) = 0, \quad b_3(x_j) = 0, \quad b'_3(x_{j-1}) = 1, \quad b'_3(x_j) = 0, \\ b_4(x_{j-1}) = 0, \quad b_4(x_j) = 0, \quad b'_4(x_{j-1}) = 0, \quad b'_4(x_j) = 1.$$



Also hat das Polynom $s_j \in \mathbb{P}_3$ welches die Interpolationsbedingungen (3.35) erfüllt die Form

$$s_j(x) = y_{j-1}b_1(x) + y_jb_2(x) + c_{j-1}b_3(x) + c_jb_4(x). \quad (3.36)$$

Wir betrachten jetzt eine Funktion s auf dem gesamten Intervall $[a, b]$, die stückweise auf den Teilintervallen wie folgt definiert wird:

$$s(x)|_{[x_{j-1}, x_j]} = s_j(x). \quad (3.37)$$

Per Definition gilt $s(x) \in C^1([a, b])$. Nach Erfüllung der Interpolationsbedingungen $s(x_j) = y_j$ stehen uns immer noch die $n + 1$ freien Parameter c_0, c_1, \dots, c_n zur Verfügung. Diese werden wir dazu benutzen, um $s(x) \in C^2([a, b])$ zu erzwingen:

$$s''_{j+1}(x_j) = s''_j(x_j), \quad j = 1, \dots, n - 1. \quad (3.38)$$

Nun haben wir immer noch $n + 1 - (n - 1) = 2$ freie Parameter zur Verfügung, mit denen wir zwei weitere Bedingungen an s stellen können.

Obige Vorbetrachtungen motivieren die folgende Definition.

Definition 3.22

- a) $s : [a, b] \rightarrow \mathbb{R}$ heisst **kubischer Spline** zu den Knoten $a = x_0 < x_1 < \dots < x_n = b$, wenn gilt:
 - (i) $s(x)$ ist zweimal stetig differenzierbar auf $[a, b]$
 - ii) $s(x)$ ist auf $[x_i, x_{i+1}]$ ein Polynom vom Grad höchstens 3.
- b) s heisst **interpolierender Spline** zu den Stützwerten y_0, \dots, y_n , wenn gilt $s(x_j) = y_j$ für $j = 0, 1, \dots, n$.
- c) s heisst **vollständiger interpolierender Spline**, falls neben a) und b) gilt: $s'(a) = c_0, s'(b) = c_n$ für vorgegebene c_0, c_n .
- d) s heisst **periodischer interpolierender Spline** mit Periode $b - a$, falls neben a) und b) gilt: $s^{(j)}(a) = s^{(j)}(b)$ für $j = 0, 1, 2$.
- e) s heisst **natürlicher interpolierender Spline**, wenn neben a) und b) gilt: $s''(a) = s''(b) = 0$.

Um kubische Interpolationssplines tatsächlich ausrechnen und zeichnen zu können, müssen wir die unbekannt Parameter c_0, \dots, c_n bestimmen. Dazu betrachten wir (3.38) näher.

Lemma 3.23 Die Bedingung (3.38) ist genau dann erfüllt, wenn

$$\frac{1}{h_j}c_{j-1} + \left(\frac{2}{h_j} + \frac{2}{h_{j+1}}\right)c_j + \frac{1}{h_{j+1}}c_{j+1} = 3\left(\frac{y_j - y_{j-1}}{h_j^2} + \frac{y_{j+1} - y_j}{h_{j+1}^2}\right) \quad (3.39)$$

für $j = 1, \dots, n - 1$ gilt.

Beweis. Wir haben

$$\begin{aligned} h_j^2 b_1''(x) &= \phi''\left(\frac{x_j - x}{h_j}\right) = 6 - 12\left(\frac{x_j - x}{h_j}\right), \\ h_j^2 b_2''(x) &= \phi''\left(\frac{x - x_{j-1}}{h_j}\right) = 6 - 12\left(\frac{x - x_{j-1}}{h_j}\right), \\ h_j b_3''(x) &= -\psi''\left(\frac{x_j - x}{h_j}\right) = -6\left(\frac{x_j - x}{h_j}\right) + 2, \\ h_j b_4''(x) &= \psi''\left(\frac{x - x_{j-1}}{h_j}\right) = 6\left(\frac{x - x_{j-1}}{h_j}\right) - 2, \end{aligned}$$

und damit gilt wegen der Darstellung (3.36)

$$\begin{aligned} s_j''(x_j) &= \frac{6}{h_j^2} y_{j-1} - \frac{6}{h_j^2} y_j + \frac{2}{h_j} c_{j-1} + \frac{4}{h_j} c_j, \\ s_{j+1}''(x_j) &= -\frac{6}{h_{j+1}^2} y_j + \frac{6}{h_{j+1}^2} y_{j+1} - \frac{4}{h_{j+1}} c_j - \frac{2}{h_{j+1}} c_{j+1}. \end{aligned}$$

Daraus folgt die Äquivalenz der Bedingung $s_j''(x_j) = s_{j+1}''(x_j)$ zu (3.39). \square

Für die Bestimmung von c_0, \dots, c_n sind also die $n-1$ Bedingungen (3.39) zu erfüllen. Zusätzlich fügt man je nach Spline-Art noch die in Definition 3.22 angegebenen Bedingungen hinzu.

- a) **Vollständiger Spline.** Hier sind c_0 und c_n vorgegeben und man erhält für c_1, \dots, c_{n-1} das $(n-1) \times (n-1)$ LGS

$$\begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \ddots & \ddots & & \\ & & \ddots & \ddots & \beta_{n-1} & \\ & & & & \beta_{n-1} & \alpha_{n-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 - \frac{1}{h_1} c_0 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} - \frac{1}{h_n} c_n \end{pmatrix} \quad (3.40)$$

mit

$$\alpha_j = \frac{2}{h_j} + \frac{2}{h_{j+1}}, \quad \beta_j = \frac{1}{h_j}, \quad d_j = 3 \left(\frac{y_j - y_{j-1}}{h_j^2} + \frac{y_{j+1} - y_j}{h_{j+1}^2} \right). \quad (3.41)$$

- b) **Periodischer Spline.** Damit der interpolierende Spline überhaupt die Periode $b - a = x_n - x_0$ haben kann, muss für die Daten $y_0 = y_n$ gelten. Für die Periodizität der ersten Ableitung muss $c_0 = c_n$ gesetzt werden. Für die Periodizität der zweiten Ableitung, muss $s_1''(x_0) = s_n''(x_n)$ gelten. Analog zu dem Beweis von Lemma 3.23 ist die letzte Forderung äquivalent zu der Bedingung

$$\frac{1}{h_n} c_{n-1} + \left(\frac{2}{h_n} + \frac{2}{h_1} \right) c_0 + \frac{1}{h_1} c_1 = 3 \left(\frac{y_0 - y_{n-1}}{h_n^2} + \frac{y_1 - y_0}{h_1^2} \right),$$

Zusammen mit (3.39) ergibt sich damit das $n \times n$ Gleichungssystem

$$\begin{pmatrix} \alpha_0 & \beta_1 & & & \beta_n \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{n-1} \\ \beta_n & & & & \alpha_{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}, \quad (3.42)$$

für $n > 2$, wobei zusätzlich zu (3.41) noch

$$\alpha_0 = \frac{2}{h_n} + \frac{2}{h_1}, \quad d_0 = 3 \left(\frac{y_0 - y_{n-1}}{h_n^2} + \frac{y_1 - y_0}{h_1^2} \right)$$

gesetzt werden. (Bemerkung: Für $n = 2$ muss β_2 in (3.42) durch $\beta_1 + \beta_2$ ersetzt werden.)

- c) **Natürlicher Spline.** Bei natürlichen Splines muss $s_1''(x_0) = s_n''(x_n) = 0$ erfüllt sein. Daraus folgt das $(n+1) \times (n+1)$ LGS

$$\begin{pmatrix} \tilde{\alpha}_0 & \beta_1 & & & \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \alpha_{n-1} & \beta_n \\ & & & & \beta_n & \tilde{\alpha}_n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = \begin{pmatrix} \tilde{d}_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ \tilde{d}_n \end{pmatrix}. \quad (3.43)$$

wobei zusätzlich zu (3.41) noch

$$\tilde{\alpha}_0 = \frac{2}{h_1}, \quad \tilde{d}_0 = 3 \frac{y_1 - y_0}{h_1^2}, \quad \tilde{\alpha}_n = \frac{2}{h_n}, \quad \tilde{d}_n = 3 \frac{y_n - y_{n-1}}{h_n^2}$$

gesetzt werden.

Die Existenz und Eindeutigkeit der Lösungen von (3.40), (3.42) respektive (3.43) muss noch gesichert werden. Dazu nutzen wir die folgende besondere Eigenschaft der Systemmatrizen.

Definition 3.24 Eine Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ heisst strikt zeilendiagonaldominant, wenn

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1, \dots, n$$

gilt.

Satz 3.25 Eine diagonaldominante Matrix ist regulär.

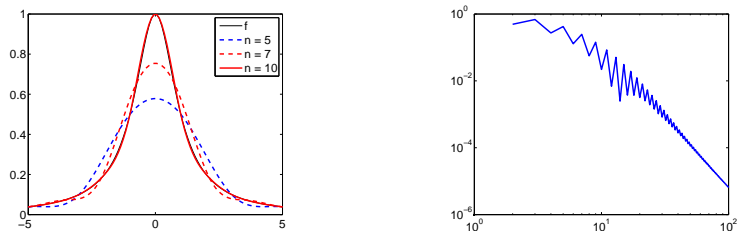
Beweis. Übung. \square

Man sieht leicht ein, dass die Systemmatrizen in (3.40), (3.42) und (3.43) allesamt strikt zeilendiagonaldominant sind; und damit sind die LGS eindeutig lösbar. Mit der **LR**- bzw. Choleskyzerlegung für Bandmatrizen (siehe Abschnitt 2.8), kostet die Berechnung von c_0, \dots, c_n nur $O(n)$ flops. Für (3.42) verwendet man die **Sherman-Morrison-Formel**

$$(\mathbf{A} + \underline{u} \underline{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1} \underline{u} \underline{v}^T \mathbf{A}^{-1}}{1 + \underline{v}^T \mathbf{A}^{-1} \underline{u}}, \quad (3.44)$$

um die Lösung auf ein Tridiagonalsystem zurückzuführen.

Beispiel 3.26 Sei wieder $f(x) = (1+x^2)^{-1}$ und $[a, b] = [-5, 5]$. Wähle äquidistante Stützstellen $x_j = 5 + \frac{10}{n}j$ für $j = 0, \dots, n$. Die folgenden Abbildungen zeigen die vollständigen Splines für $n = 5, 7, 10$, sowie den Verlauf des Fehlers in der L_∞ -Norm im Vergleich zu n .



Satz 3.27 Sei $f \in C^4([a, b])$ und s der in den Stützstellen $a, a + h, a + 2h, \dots, b$ mit $h = (b - a)/n$ interpolierende vollständige Spline. Dann gilt

$$\|f - s\|_\infty \leq \frac{5}{384} h^4 \|f^{(4)}\|_\infty.$$

3.8 Bézier-Techniken*

Ziel ist jetzt die Approximation (nicht die Interpolation!) einer **Kurve** durch stückweise Polynome. Wie bereits oben erwähnt garantieren die Bernstein-Polynome gleichmässige Konvergenz mit steigendem Polynomgrad.

Satz 3.28 Sei $f \in C^0([0, 1])$ und

$$p_n(x) := \sum_{j=0}^n f\left(\frac{j}{n}\right) \binom{n}{j} x^j (1-x)^{n-j}, \quad 0 \leq x \leq 1. \quad (3.45)$$

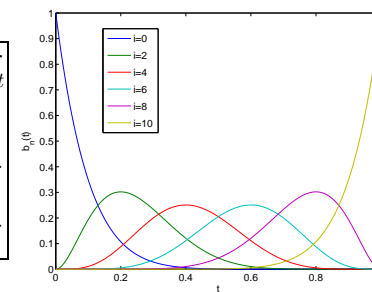
Dann konvergiert p_n gleichmässig gegen f für $n \rightarrow \infty$. Wenn $f \in C^m([0, 1])$, dann konvergiert $p_n^{(k)}$ gleichmässig gegen $f^{(k)}$ für $n \rightarrow \infty$.

Im allgemeinen ist die Konvergenz der Polynome in (3.45) zu langsam, um für die numerische Mathematik von Interesse zu sein. Allerdings haben diese Polynome gewisse formerhaltende Eigenschaften und lassen sich schnell auswerten. Ihr Hauptanwendungsbereich liegt in der Computergraphik.

Definition 3.29 Die **Bernstein-Polynome** vom Grad n sind definiert als:

$$b_{i,n}(t) := \binom{n}{i} t^i (1-t)^{n-i} \quad i = 0, \dots, n.$$

Wir setzen $b_{i,n} \equiv 0$ für $i < 0$ oder $i > n$.



Lemma 3.30 (Eigenschaften von Bernstein-Polynomen)

- $t = 0$ ist i -fache Nullstelle von $b_{i,n}$, $t = 1$ ist $n - i$ -fache Nullstelle von $b_{i,n}$;
- $\sum_{i=0}^n b_{i,n} \equiv 1$ (Zerlegung der Eins);
- $b_{i,n}(t) = (1-t)b_{i,n-1} + tb_{i-1,n-1}(t)$;
- $b_{i,n}(t) \geq 0 \quad \forall 0 \leq t \leq 1$.

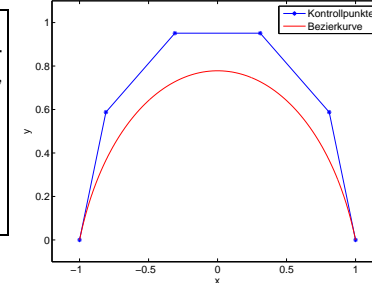
Aus Lemma 3.30.3 folgt eine einfache Rekursion zur Auswertung von $b_{i,n}(t)$.

```

MATLAB
function V = bernstein(n,x)
V = [ones(1,length(x)); ...
     zeros(n,length(x))];
for j=1:n
    for k=j+1:-1:2
        V(k,:) = x.*V(k-1,:) + (1-x).*V(k,:);
    end
    V(1,:) = (1-x).*V(1,:);
end
    
```

Die Bernstein-Polynome eignen sich auch zur Beschreibung von *Kurven* im \mathbb{R}^p . Wir führen hier nur die Definition der sogenannten **Bézier-Kurven** an und verweisen auf die Literatur.¹²

Definition 3.31 (Bézier-Kurven)
Die **Bézier-Kurve** zu **Kontrollpunkten** $\underline{d}_i \in \mathbb{R}^m$, $i = 0, \dots, n$, $m \in \mathbb{N}$, ist

$$\gamma : \begin{cases} [0, 1] & \mapsto \mathbb{R}^d \\ t & \mapsto \sum_{i=0}^n \underline{d}_i b_{i,n}(t). \end{cases}$$


¹²Siehe zum Beispiel Abschnitt 8.7.1 in Quarteroni, Sacco, Saleri: Numerische Mathematik 2.

Kapitel 4

Trigonometrische Interpolation

Schwingungen spielen in elektrischen, akustischen und mechanischen Systemen eine zentrale Rolle. In Systemen ohne Dämpfung lassen sich Schwingungen durch periodische Funktionen beschreiben. Es liegt nun nahe, in diesem Kontext zur Interpolation statt Polynome deren periodische Gegenstücke – trigonometrische Funktionen – als Interpolationsbasis zu verwenden.

4.1 Fourier-Reihen*

Die fundamentalsten periodischen Funktionen sind Sinus und Cosinus. Die eulersche Identität

$$e^{i\varphi} = \cos(\varphi) + i \sin(\varphi),$$

verknüpft die trigonometrischen Funktionen eng mit den komplexen Zahlen. Wir werden deshalb im folgenden *komplexwertige* periodische Funktionen mit Periode 2π betrachten:

$$f : \mathbb{R} \rightarrow \mathbb{C}, \quad f(x) = f(x + 2\pi).$$

Zur Behandlung von f genügt es offensichtlich, sich auf das Grundintervall $[0, 2\pi]$ zurückzuziehen. Auf diesem definieren wir das folgende, sogenannte **L₂-Skalarprodukt**¹³:

$$\langle f, g \rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x) \overline{g(x)} \, dx.$$

Dieses induziert (siehe Abschnitt 0.6) die **L₂-Norm**

$$\|f\|_2 = \left(\frac{1}{2\pi} \int_0^{2\pi} |f(x)|^2 \, dx \right)^{1/2}.$$

Man überprüft leicht, dass die **Grundschnwingungen**

$$e_j(x) = e^{ijx} = \cos(jx) + i \sin(jx)$$

¹³Eigentlich dürften wir hier nur Funktionen betrachten für die das L₂-Skalarprodukt wohldefiniert ist. Wir verzichten aber im Rahmen dieser Vorlesung auf die korrekte formale Einführung des Vektorraums L₂, da die hierzu notwendigen mathematischen Hilfsmittel (Masstheorie) den Rahmen der Vorlesung sprengen würden.

ein Orthonormalsystem bezüglich $\langle \cdot, \cdot \rangle$ bilden:

$$\langle e_j, e_k \rangle = \frac{1}{2\pi} \int_0^{2\pi} e^{ijx} e^{-ikx} \, dx = \delta_{jk} = \begin{cases} 1, & \text{für } j = k, \\ 0, & \text{sonst.} \end{cases}$$

Der sogenannte **Fourier-Koeffizient**

$$\hat{f}(k) := \langle f, e_k \rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} \, dx \in \mathbb{C} \quad (4.1)$$

kann als Stärke der k -ten Grundschnwingung in f interpretiert werden.

Die **Fourier-Teilsumme**

$$S_n(f, x) := \sum_{|k| \leq n} \hat{f}(k) e^{ikx} = \sum_{|k| \leq n} \langle f, e_k \rangle e_k. \quad (4.2)$$

konvergiert (im Sinne der L₂-Norm) für $n \rightarrow \infty$ gegen f :

$$f(x) = \lim_{n \rightarrow \infty} S_n(f, x) = \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{ikx}. \quad (4.3)$$

Die Funktion $S_n(f, \cdot)$ ist die orthogonale Projektion von f auf den $(2n+1)$ -dimensionalen Unterraum $U_{2n+1} = \text{span} \{e_k : |k| \leq n\}$ (vgl. Abschnitt 0.8.1).

Lemma 4.1 Die in (4.2) definierte Funktion $S_n(f, \cdot) \in U_{2n+1}$ erfüllt das *Minimierungsproblem*

$$\|f - S_n(f, \cdot)\|_2 = \min\{\|f - u\|_2 : u \in U_{2n+1}\}$$

Beweis. Da die Funktionen e_k eine Orthonormalbasis bilden, haben wir

$$\|f - S_n(f, \cdot)\|_2^2 = \left\| \sum_{|k| > n} \langle f, e_k \rangle e_k \right\|_2^2 = \sum_{|k| > n} |\langle f, e_k \rangle|^2.$$

Auf der anderen Seite gilt für eine beliebige Funktion $u \in U_{2n+1}$:

$$\begin{aligned} \|f - u\|_2^2 &= \left\| \sum_{k=-\infty}^{\infty} \langle f - u, e_k \rangle e_k \right\|_2^2 = \sum_{|k| \leq n} |\langle f - u, e_k \rangle|^2 + \sum_{|k| > n} |\langle f - u, e_k \rangle|^2 \\ &= \sum_{|k| \leq n} |\langle f - u, e_k \rangle|^2 + \sum_{|k| > n} |\langle f, e_k \rangle|^2 \geq \sum_{|k| > n} |\langle f, e_k \rangle|^2. \end{aligned}$$

□

Die Approximation von f durch $S_n(f)$ lässt sich auch als Interpolation interpretieren. Anstatt die Funktionswerte von f an Stützstellen abzugleichen, wird hier eine Funktion konstruiert, bei der die ersten $2n + 1$ Fourierkoeffizienten mit denen von f übereinstimmen.

Für eine *reellwertige Funktion* f setzen wir

$$a_0 = 2\langle f, e_0 \rangle, \quad a_k = \langle f, e_k \rangle + \langle f, e_{-k} \rangle, \quad b_k = i(\langle f, e_k \rangle - \langle f, e_{-k} \rangle)$$

und erhalten aus (4.3) die reelle Darstellung

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx)).$$

4.2 Die trigonometrische Interpolationsaufgabe

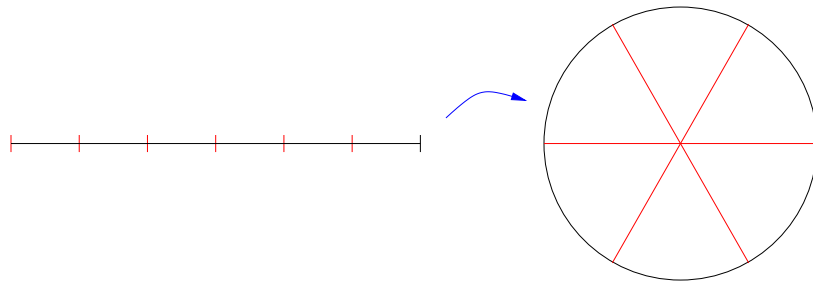
In Anwendungen kann natürlich nicht wie in Abschnitt 4.1 angenommen werden, dass die Funktion f auf dem gesamten Intervall $[0, 2\pi]$ vorliegt. Zum Beispiel liegen Audiodaten auf CDs nur mit einer Abtastrate von 44.1 kHz vor. Wir betrachten deshalb nun die äquidistanten Stützstellen

$$x_j = 2\pi j/n, \quad j = 0, 1, \dots, n-1$$

im Intervall $[0, 2\pi]$. Dies entspricht einer Aufteilung des Einheitskreises in gleichlange Bogensegmente:

$$\omega_n^j = e^{-2\pi i j/n}, \quad j = 0, 1, \dots, n-1, \tag{4.4}$$

wobei $\omega_n = e^{-2\pi i/n}$ die n -te Einheitswurzel bezeichnet.



Das Ziel ist jetzt eine gegebene periodische Funktion, deren Werte an den Stützstellen bekannt sind, durch eine einfachere *periodische* Funktion zu "ersetzen", d.h. zu interpolieren. Wir verwenden dazu als Interpolationsraum

$$T_n := \left\{ \sum_{j=0}^{n-1} c_j e^{ijx} : c_j \in \mathbb{C} \right\}.$$

Mit der Variablensubstitution $z = e^{ix}$ sieht man leicht ein, dass jedes Element in T_n ein Polynom vom Grad höchstens $n-1$ in z ist. Also ist T_n Vektorraum der Dimension n . Damit wissen wir auch bereits, dass die folgende Interpolationsaufgabe eindeutig lösbar ist.

Aufgabe 4.2 Finde $t_n \in T_n$ so dass

$$t_n(x_j) = y_j, \quad j = 0, \dots, n-1, \tag{4.5}$$

für $x_j = 2\pi j/n$ und $y_j \in \mathbb{C}$.

Wie am Anfang von Kapitel 3 schreiben wir die Gleichungen (4.5) in Matrix-Vektor-Form. Mit

$$t_n(x) = c_0 + c_1 e^{ix} + \dots + c_{n-1} e^{i(n-1)x} = \sum_{j=0}^{n-1} c_j e^{ijx}$$

entspricht (4.5) dem linearen Gleichungssystem

$$\mathbf{F}_n \underline{c} = \underline{y} \tag{4.6}$$

mit den Vektoren $\underline{c} = (c_0, \dots, c_{n-1})^T$, $\underline{y} = (y_0, \dots, y_{n-1})^T$ und der Matrix

$$\mathbf{F}_n = \begin{pmatrix} 1 & e^{ix_0} & \dots & e^{i(n-1)x_0} \\ 1 & e^{ix_1} & \dots & e^{i(n-1)x_1} \\ \vdots & \vdots & & \vdots \\ 1 & e^{ix_{n-1}} & \dots & e^{i(n-1)x_{n-1}} \end{pmatrix} = (f_{jk})_{j,k=0}^{n-1} = (\omega_n^{jk})_{j,k=0}^{n-1}. \tag{4.7}$$

Es ist dabei zu beachten, dass – im Gegensatz zur üblichen Konvention – die Nummerierung der Vektor- und Matrixeinträge in diesem Abschnitt mit 0 beginnt.

4.3 Die schnelle Fourier-Transformation (FFT)

Zur Lösung des Gleichungssystems (4.6) lohnt es sich, die Matrix \mathbf{F}_n etwas näher zu betrachten. Das wesentliche Hilfsmittel sind die folgende Identitäten.

Lemma 4.3 Sei $\omega_n = e^{-2\pi i/n}$. Dann gelten

$$\omega_n^k = \omega_n^{k+n} \quad \forall k \in \mathbb{Z}, \quad \omega_n^n = 1, \quad \omega_n^{n/2} = -1, \tag{4.8}$$

sowie

$$\sum_{j=0}^{n-1} \omega_n^{jk} = \begin{cases} n & \text{falls } k = 0, \\ 0 & \text{falls } k = 1, \dots, n-1. \end{cases} \tag{4.9}$$

Beweis. Die Eigenschaften (4.8) ergeben sich sofort aus den Eigenschaften der Exponentialfunktion. Für (4.9) wenden wir die geometrische Reihe

$$\sum_{j=0}^{n-1} q^j = \frac{1-q^n}{1-q}$$

auf $q = \omega_n^k$ mit $k = 1, \dots, n-1$ an:

$$\sum_{j=0}^{n-1} (\omega_n^k)^j = \frac{1 - (\omega_n^k)^n}{1 - \omega_n^k} = 0.$$

□

Wir bezeichnen jetzt mit a_{jk} das Element (j, k) (wieder mit 0 beginnend gezählt!) der Matrix $\mathbf{F}_n^H \mathbf{F}_n$ und erhalten aus (4.9):

$$a_{jk} = \sum_{i=0}^{n-1} \overline{f_{ij}} f_{ik} = \sum_{i=0}^{n-1} \omega_n^{-ij} \omega_n^{ik} = \sum_{i=0}^{n-1} \omega_n^{i(k-j)} = \begin{cases} n & \text{falls } k = j, \\ 0 & \text{falls } k \neq j. \end{cases}$$

Es gilt also $\mathbf{F}_n^H \mathbf{F}_n = n \mathbf{I}_n$ oder, anders ausgedrückt, $\frac{1}{\sqrt{n}} \mathbf{F}_n$ ist unitär! Damit gilt

$$\mathbf{F}_n^{-1} = \frac{1}{n} \mathbf{F}_n^H = \frac{1}{n} \overline{\mathbf{F}_n}.$$

Um das Gleichungssystem in (4.6) zu lösen, braucht man also lediglich mit $\frac{1}{n}\overline{\mathbf{F}_n}$ zu multiplizieren (bzw. mit $\frac{1}{n}\mathbf{F}_n$ multiplizieren und zwei Mal das komplex Konjugierte nehmen). Dies reduziert den Aufwand zur Lösung bereits auf $O(n^2)$ flops. Das folgende Resultat erlaubt es, den Aufwand noch weiter zu reduzieren.

Satz 4.4 Sei $n \geq 2$ gerade. Sei \mathbf{P}_n die zu der Permutation $\xi : \{0, \dots, n-1\} \rightarrow \{0, \dots, n-1\}$ mit

$$\xi : 0 \mapsto 0, 1 \mapsto 2, \dots, \frac{n}{2} - 1 \mapsto n - 2, \frac{n}{2} \mapsto 1, \frac{n}{2} + 1 \mapsto 3, \dots, n - 1 \mapsto n - 1$$

gehörige Permutationsmatrix (siehe Definition 2.25). Dann gilt

$$\mathbf{P}_n^T \mathbf{F}_n = \begin{pmatrix} \mathbf{F}_{n/2} & \mathbf{F}_{n/2} \\ \mathbf{F}_{n/2} \Omega_{n/2} & -\mathbf{F}_{n/2} \Omega_{n/2} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{n/2} & \\ & \mathbf{F}_{n/2} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{n/2} & \mathbf{I}_{n/2} \\ \Omega_{n/2} & -\Omega_{n/2} \end{pmatrix}$$

mit

$$\Omega_{n/2} = \text{diag}(\omega_n^0, \omega_n^1, \dots, \omega_n^{n/2-1}).$$

Beweis. Durch Ausnutzung der Eigenschaften der Potenzen von ω_n . \square

Beispiel 4.5 Wir illustrieren die Aussage von Satz 4.4 für $n = 6$. Laut Definition (4.7) hat \mathbf{F}_6 die Form

$$\mathbf{F}_6 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_6 & \omega_6^2 & \omega_6^3 & \omega_6^4 & \omega_6^5 \\ 1 & \omega_6^2 & \omega_6^4 & \omega_6^6 & \omega_6^8 & \omega_6^{10} \\ 1 & \omega_6^3 & \omega_6^6 & \omega_6^9 & \omega_6^{12} & \omega_6^{15} \\ 1 & \omega_6^4 & \omega_6^8 & \omega_6^{12} & \omega_6^{16} & \omega_6^{20} \\ 1 & \omega_6^5 & \omega_6^{10} & \omega_6^{15} & \omega_6^{20} & \omega_6^{25} \end{pmatrix},$$

wobei $\omega_6 = e^{-2\pi i/6} = e^{\pi i/3}$. Wir können die Permutationsmatrix \mathbf{P}_6 aus Satz 4.4 explizit nach Definition 2.25 angeben (die passende Numerierung der Matrixeinträge verwenden):

$$\mathbf{P}_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Wichtig ist aber allein zu wissen, dass \mathbf{P}_6^T angewandt auf \mathbf{F}_6 die Zeilen gemäss der Inversen von der in Satz 4.4 angegebenen Permutation,

$$\xi : 0 \mapsto 0, 1 \mapsto 2, 2 \mapsto 4, 3 \mapsto 1, 4 \mapsto 3, 5 \mapsto 5,$$

vertauscht:

$$\mathbf{P}_6^T \mathbf{F}_6 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_6^2 & \omega_6^4 & \omega_6^6 & \omega_6^8 & \omega_6^{10} \\ 1 & \omega_6^4 & \omega_6^8 & \omega_6^{12} & \omega_6^{16} & \omega_6^{20} \\ 1 & \omega_6^6 & \omega_6^{12} & \omega_6^{18} & \omega_6^{24} & \omega_6^0 \\ 1 & \omega_6^8 & \omega_6^{16} & \omega_6^{24} & \omega_6^0 & \omega_6^2 \\ 1 & \omega_6^{10} & \omega_6^{20} & \omega_6^0 & \omega_6^2 & \omega_6^4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_6^2 & \omega_6^4 & \omega_6^6 & \omega_6^8 & \omega_6^{10} \\ 1 & \omega_6^4 & \omega_6^8 & \omega_6^{12} & \omega_6^{16} & \omega_6^{20} \\ 1 & \omega_6^6 & \omega_6^{12} & \omega_6^{18} & \omega_6^{24} & \omega_6^0 \\ 1 & \omega_6^8 & \omega_6^{16} & \omega_6^{24} & \omega_6^0 & \omega_6^2 \\ 1 & \omega_6^{10} & \omega_6^{20} & \omega_6^0 & \omega_6^2 & \omega_6^4 \end{pmatrix}.$$

Hierbei wurde auf der rechten Seite mehrfach die Beziehung (4.8) aus Lemma 4.3 ausgenutzt. Da $\omega_3^k = \omega_6^{2k}$ für jedes $k \in \mathbb{Z}$:

$$\mathbf{P}_6^T \mathbf{F}_6 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_3^1 & \omega_3^2 & \omega_3^3 & \omega_3^4 & \omega_3^5 \\ 1 & \omega_3^2 & \omega_3^4 & \omega_3^8 & \omega_3^{16} & \omega_3^1 \\ 1 & \omega_3^3 & \omega_3^6 & \omega_3^{12} & \omega_3^9 & \omega_3^{18} \\ 1 & \omega_3^4 & \omega_3^8 & \omega_3^{16} & \omega_3^9 & \omega_3^{18} \\ 1 & \omega_3^5 & \omega_3^{10} & \omega_3^6 & \omega_3^{12} & \omega_3^3 \end{pmatrix} = \begin{pmatrix} \mathbf{F}_3 & \mathbf{F}_3 \\ \mathbf{F}_3 \Omega_3 & -\mathbf{F}_3 \Omega_3 \end{pmatrix}$$

mit

$$\mathbf{F}_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega_3^1 & \omega_3^2 \\ 1 & \omega_3^2 & \omega_3^1 \end{pmatrix}, \quad \Omega_3 = \begin{pmatrix} 1 & & \\ & \omega_6 & \\ & & \omega_6^2 \end{pmatrix}.$$

\diamond

Satz 4.4 lässt sich dazu einsetzen, um eine Matrix-Vektormultiplikation $\mathbf{F}_n \underline{y}$ für gerade n auf eine Rekursion zurückzuführen. Dazu partitionieren wir den Vektor $\underline{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ so dass $\underline{y}_1, \underline{y}_2 \in \mathbb{C}^{n/2}$. Mittels Satz lässt sich $\mathbf{F}_n \underline{y}$ dann wie folgt schreiben:

$$\mathbf{F}_n \underline{y} = \mathbf{P}_n \begin{pmatrix} \mathbf{F}_{n/2} & \mathbf{F}_{n/2} \\ \mathbf{F}_{n/2} \Omega_{n/2} & -\mathbf{F}_{n/2} \Omega_{n/2} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{n/2} & \mathbf{I}_{n/2} \\ \Omega_{n/2} & -\Omega_{n/2} \end{pmatrix} \begin{pmatrix} \underline{y}_1 \\ \underline{y}_2 \end{pmatrix} = \mathbf{P}_n \begin{pmatrix} \mathbf{F}_{n/2}(\underline{y}_1 + \underline{y}_2) \\ \mathbf{F}_{n/2} \Omega_{n/2}(\underline{y}_1 - \underline{y}_2) \end{pmatrix}.$$

Dass heisst, die Multiplikation mit \mathbf{F}_n kann im wesentlichen auf zwei Multiplikationen mit $\mathbf{F}_{n/2}$ zurückgeführt werden. Rekursives Anwenden dieser Beziehung führt für den Fall, dass n eine Potenz von 2 ist, auf den folgenden Algorithmus¹⁴.

Algorithmus 4.6

Input: Vektor $\underline{y} \in \mathbb{C}^n$ mit n Potenz von 2.

Output: Matrix-Vektor-Produkt $\underline{z} = \mathbf{F}_n \underline{y}$.

Partit. $\underline{y} = \begin{pmatrix} \underline{y}_1 \\ \underline{y}_2 \end{pmatrix}$ mit $\underline{y}_1, \underline{y}_2 \in \mathbb{C}^{n/2}$.

$\underline{z}_1 = \underline{y}_1 + \underline{y}_2$, $\underline{z}_2 = \mathbf{F}_{n/2}(\underline{y}_1 - \underline{y}_2)$.

Rekursiv: $\underline{z}_1 \leftarrow \mathbf{F}_{n/2} \underline{z}_1$

Rekursiv: $\underline{z}_2 \leftarrow \mathbf{F}_{n/2} \underline{z}_2$.

$\underline{z} = \mathbf{P}_n \begin{pmatrix} \underline{z}_1 \\ \underline{z}_2 \end{pmatrix}$

```
MATLAB
function z = myfft(y)
n = length(y);
if (n == 1), z = y; return; end
omega = exp(-2*pi*i/n);
z1 = y(1:n/2)+y(n/2+1:n);
z2 = (omega.^(0:(n/2-1))) .* ...
      (y(1:n/2)-y(n/2+1:n));
z = [myfft(z1); myfft(z2)];
z = reshape(...
      reshape(z,n/2,2)',n,1);
```

Die Aufwandsanalyse von rekursiven Algorithmen kann oft mit dem Master-Theorem (siehe Wikipedia-Eintrag) “erschlagen” werden. Im Fall von Algorithmus 4.6 können

¹⁴Es lohnt sich an einem kleinem Beispiel nachzuvollziehen, wie mittels `reshape` die Multiplikation mit \mathbf{P}_n implementiert wurde.

wir den Aufwand aber recht einfach abschätzen. Bezeichne $A(n)$ die benötigten flops von Algorithmus 4.6 für einen Vektor der Länge n . Ignoriert man den Aufwand für die Berechnung der Potenzen von ω_n (diese können zum Beispiel vorher berechnet und in einer Tabelle abgelegt werden), so gilt¹⁵ $A(n) = 5n + 2A(n/2)$. Rekursives Anwenden dieser Formel führt auf

$$A(n) = 5n + 5n + 4A(n/4) = \dots = 5kn + 2^k A(n/2^k).$$

Für $k = \log_2 n$ haben wir $A(n/2^k) = A(n/n) = A(1) = 0$, also folgt

$$A(n) = 5n \log_2 n.$$

Diesem im Vergleich zu $O(n^2)$ bedeutend geringeren Aufwand trägt der Name **schnelle Fourier-Transformation** (Englisch: Fast Fourier Transform – FFT) Rechnung. Der zweite Teil des Namens leitet sich aus der Tatsache ab, dass $\underline{z} = \mathbf{F}_n \underline{y}$ als **diskrete Fourier-Transformation** von \underline{y} bezeichnet wird, weil die k -te Komponente von \underline{z} , als diskretes Analogon des k -ten Fourierkoeffizienten $\hat{f}(k)$ (siehe (4.1)) aufgefasst werden kann.

Bemerkung 4.7 Die in MATLAB enthaltene Funktion `fft` basiert auf dem Softwarepaket FFTW (Fastest Fourier Transform in the West, siehe <http://www.fftw.org/>). Für $n = 2^k$ entspricht dies im Prinzip Algorithmus 4.6. Für $n \neq 2^k$ müssen andere Zerlegungen von n verwendet werden; die asymptotische Laufzeit ist immer noch $O(n \log_2 n)$, aber die Konstante kann sich besonders für Werte von n mit grossen Primfaktoren deutlich erhöhen.

Bemerkung 4.8 Für die obige Herleitung der FFT ist die Annahme wesentlich, dass die Stützstellen x_j gleichmässig im Intervall $[0, 2\pi]$ verteilt sind. Für Algorithmen und Software für ungleichmässig verteilte Stützstellen, siehe <https://www-user.tu-chemnitz.de/~potts/nfft/>.

¹⁵Eine komplexe Addition oder Subtraktion wird mit 2 flops und eine komplexe Multiplikation mit 6 flops gezählt.

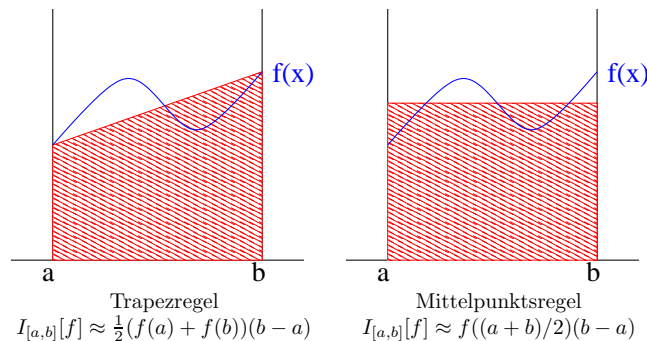
Kapitel 5

Numerische Integration

Dieses Kapitel beschäftigt sich mit der näherungsweise Berechnung des bestimmten Integrals

$$I_{[a,b]}[f] = \int_a^b f(x) dx.$$

In der Praxis ist die Funktion f oft zu komplex, um eine explizite Formel für die Stammfunktion angeben zu können. Die folgenden beiden Ideen zur Approximation von $I_{[a,b]}[f]$ liegen nahe:



Natürlich wird die Approximationsgüte für grössere Intervalle $[a, b]$ recht ungenügend sein; wir werden im Folgenden verschiedene Techniken entwickeln, um diese zu verbessern.

5.1 Interpolatorische Quadraturformeln

Trapez- und Mittelpunktsregel lassen sich beide als **Integral der linearen Interpolanten** von f interpretieren. Diese Vorgehensweise lässt sich auf Polynome beliebigen Grades verallgemeinern. Für gegebene Stützstellen

$$a \leq x_0 < \dots < x_n \leq b$$

betrachten wir das interpolierende Polynom vom Grad n in der Lagrange-Darstellung (siehe Abschnitt 3.1):

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

Dann ergibt das bestimmte Integral von p_n eine Approximation zu $I_{[a,b]}[f]$:

$$Q_{[a,b]}^{(n)}[f] := \int_a^b \sum_{j=0}^n f(x_j) \ell_j(x) dx = \sum_{j=0}^n f(x_j) \int_a^b \ell_j(x) dx = \sum_{j=0}^n \alpha_j f(x_j), \quad (5.1)$$

wobei die Skalare $\alpha_j = \int_a^b \ell_j(x) dx$ die **Gewichte** der **Quadraturformel** $Q_{[a,b]}^{(n)}[f]$ sind. Als Korollar von Satz 3.4 erhalten wir die folgende Fehlerabschätzung:

$$\left| \int_a^b f(x) dx - Q_{[a,b]}^{(n)}[f] \right| \leq (b-a) \max_{x \in [a,b]} \prod_{j=0}^n |x - x_j| \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \\ \leq (b-a)^{n+2} \frac{\|f^{(n+1)}\|_\infty}{(n+1)!}, \quad (5.2)$$

vorausgesetzt, dass f genügend oft differenzierbar ist. Wie wir später sehen werden, ist diese Abschätzung unbefriedigend grob.

Je nachdem ob die Stützstellen die Intervallenden enthalten, unterscheidet man zwischen offenen und geschlossenen Quadraturformeln.

5.1.1 Geschlossene Newton-Cotes-Formeln

Die **geschlossenen Newton-Cotes-Formeln** erhält man mit der Stützstellenwahl

$$x_j = a + jh, \quad j = 0, \dots, n, \quad h = \frac{b-a}{n}.$$

Zur Berechnung der Gewichte $\alpha_j = \int_a^b \ell_j(x) dx$ geht man wie folgt vor: Jedes $x \in [a, b]$ ist darstellbar als $x = a + th$ mit einem $t \in [0, n]$. Durch die affin lineare Transformation $x \rightarrow t = \frac{x-a}{h}$ wird das Intervall $[a, b]$ auf $[0, n]$ abgebildet. Damit gilt

$$\ell_j(x) = \prod_{\substack{i=0 \\ i \neq j}}^n \frac{x - x_i}{x_j - x_i} = \prod_{\substack{i=0 \\ i \neq j}}^n \frac{a + th - a - ih}{a + jh - a - ih} = \prod_{\substack{i=0 \\ i \neq j}}^n \frac{t - i}{j - i}.$$

Wegen $dx/dt = 1/h$ erhält man also

$$\alpha_j = \int_a^b \ell_j(x) dx = h \int_0^n \prod_{\substack{i=0 \\ i \neq j}}^n \frac{t - i}{j - i} dt, \quad j = 0, \dots, n.$$

Die skalierten Gewichte α_j/h sind unabhängig von der Wahl des Intervalls und können also ein für allemal berechnet und tabelliert werden. Die konkrete Berechnung ist recht aufwändig und soll nur für $n = 2$ demonstriert werden.

n		ξ_j	$\alpha_j/(b-a)$	$I_{[a,b]}[f] - Q_{[a,b]}^{(n)}[f]$
1	Trapezregel	0, 1	$\frac{1}{2}, \frac{1}{2}$	$-\frac{1}{12}(b-a)^3 f^{(2)}(\xi)$
2	Simpson-Regel	$0, \frac{1}{2}, 1$	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	$-\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$
3	$\frac{3}{8}$ -Regel	$0, \frac{1}{3}, \frac{2}{3}, 1$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$-\frac{3}{80}\left(\frac{b-a}{3}\right)^5 f^{(4)}(\xi)$
4	Milne-Regel	$0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$-\frac{8}{945}\left(\frac{b-a}{4}\right)^7 f^{(6)}(\xi)$

Tabelle 5.1. Geschlossene Newton-Cotes-Formeln ($x_j = a + \xi_j(b-a)$).

Beispiel 5.1 Geschlossene Newton-Cotes-Formel für $n = 2$:

$$\alpha_0 = h \int_0^2 \frac{t-1}{0-1} \frac{t-2}{0-2} dt = \frac{h}{2} \int_0^2 (t^2 - 3t + 2) dt = \frac{1}{3} h,$$

$$\alpha_1 = h \int_0^2 \frac{t-0}{1-0} \frac{t-2}{1-2} dt = -h \int_0^2 (t^2 - 2t) dt = \frac{4}{3} h,$$

$$\alpha_2 = h \int_0^2 \frac{t-0}{2-0} \frac{t-1}{2-1} dt = \frac{h}{2} \int_0^2 (t^2 - t) dt = \frac{1}{3} h.$$

Die erhaltene Quadraturformel

$$Q_{[a,b]}^{(2)}[f] = \frac{(b-a)}{6} \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$

nennt man **Simpson-Regel**. \diamond

Tabelle 5.1 gibt die Stützstellen $x_j = a + \xi_j(b-a)$ und Gewichte für $n \leq 4$ an. In der vierten Spalte findet sich der Quadraturfehler. Dabei ist $\xi \in [a, b]$ irgendein (von f abhängiger) Zwischenwert. Man überzeugt sich leicht, dass der angegebene Quadraturfehler wesentlich kleiner als die aus dem Interpolationsfehler gewonnene Schranke (5.2) ist. Insbesondere gehen bei der Simpson- und Milne-Regel eine um eine Ordnung höhere Ableitung von f ein. Der folgende Satz zeigt die Fehlerdarstellung für $n = 1$ und $n = 2$.

Satz 5.2 Es gelten die folgenden Darstellungen des Quadraturfehlers,

i) für die Trapezregel, wenn $f \in C^2[a, b]$:

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

ii) für die Simpson-Regel, wenn $f \in C^4[a, b]$:

$$I_{[a,b]}[f] - \frac{b-a}{6} \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right] = -\frac{1}{90} \left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi),$$

mit gewissen Zwischenstellen $\xi \in [a, b]$.

Beweis.

i) Aus Satz 3.4 folgt

$$f(x) - p_1(x) = \frac{f''(\xi_x)}{2}(x-a)(x-b).$$

für ein $\xi_x \in [a, b]$. Man beachte, dass ξ_x von x abhängt. Integration auf beiden Seiten ergibt

$$I_{[a,b]}[f] - Q_{[a,b]}^{(1)}[f] = \int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b) dx. \quad (5.3)$$

Sei

$$c = \frac{\int_a^b \frac{f''(\xi_x)}{2}(x-a)(x-b) dx}{\frac{1}{2} \int_a^b (x-a)(x-b) dx} = \frac{\int_a^b f''(\xi_x)(x-a)(x-b) dx}{-\frac{1}{6}(b-a)^3}.$$

Wegen $(x-a)(x-b) \leq 0$ für x in $[a, b]$ folgt

$$\min_{x \in [a,b]} f''(x) \leq c \leq \max_{x \in [a,b]} f''(x).$$

Da f'' stetig ist, muss es aufgrund des Zwischenwertsatzes ein (von x unabhängiges) $\xi \in [a, b]$ geben, so dass $c = f''(\xi)$ gilt. Einsetzen in (5.3) ergibt die Behauptung.

ii) Da $(x-a)(x-\frac{a+b}{2})(x-b)$ in $[a, b]$ einen Vorzeichenwechsel hat, kann die Methode aus i) hier nicht angewendet werden. Wir geben nur die Beweisidee an.

Die Newton'sche Darstellung des Interpolationsfehlers lautet

$$f(x) - p_2(x) = f\left[a, \frac{a+b}{2}, b, x\right] (x-a) \left(x - \frac{a+b}{2}\right) (x-b),$$

siehe Lemma 3.9. Daraus folgt

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

Nach dem verallgemeinerten Mittelwertsatz der Integralrechnung folgt

$$\begin{aligned} I_{[a,b]}[f] - Q_{[a,b]}^{(2)}[f] &= \int_a^b f\left[a, \frac{a+b}{2}, \frac{a+b}{2}, b, x\right] (x-a) \left(x - \frac{a+b}{2}\right)^2 (x-b) dx \\ &= \frac{f^{(4)}(\xi)}{4!} \int_a^b (x-a) \left(x - \frac{a+b}{2}\right)^2 (x-b) dx. \end{aligned}$$

\square

Bei den geschlossenen Newton-Cotes-Formeln treten für $n = 8$ **negative** Gewichte α_i auf. Dies führt zu numerischer Auslöschung bei dem in der Praxis relevanten Fall eines sehr kleinen Intervalls.

5.1.2 Offene Newton-Cotes-Formeln

Die **offenen Newton-Cotes-Formeln** erhält man mit der Stützstellenwahl

$$x_j = a + (j+1)h, \quad j = 0, \dots, n, \quad h = \frac{b-a}{n+2}.$$

Zur Berechnung der Gewichte α_j geht man wie in Abschnitt 5.1.1 vor. Man erhält die folgenden Quadraturformeln:

$$Q_{[a,b]}^{(0)}[f] = (b-a) f\left(\frac{a+b}{2}\right) \quad (\text{Mittelpunktsregel})$$

$$Q_{[a,b]}^{(1)}[f] = \frac{b-a}{2} [f(a+h) + f(b-h)]$$

$$Q_{[a,b]}^{(2)}[f] = \frac{b-a}{3} \left[2f(a+h) - f\left(\frac{a+b}{2}\right) + 2f(b-h) \right]$$

$$Q_{[a,b]}^{(3)}[f] = \frac{b-a}{24} [11f(a+h) + f(a+2h) + f(b-2h) + 11f(b-h)].$$

Hier treten schon ab $n=2$ negative Gewichte auf.

Satz 5.3 Für den Quadraturfehler der Mittelpunktsregel gilt

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

mit einer gewissen Zwischenstelle $\xi \in [a, b]$.

Beweis. Wir verwenden eine analoge Schlussweise wie im Beweis von Satz 5.2 ii):

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

□

5.1.3 Weitere Quadraturformeln*

Im Gegensatz zu den Newton-Cotes-Formeln verwenden die sogenannten **Besselschen Formeln** auch Stützstellen ausserhalb von $[a, b]$; z.B.:

$$Q_{[a,b]}^{(3)}[f] = \frac{b-a}{24} [-f(2a-b) + 13f(a) + 13f(b) - f(2b-a)].$$

Die **Hermiteschen Formeln** verwenden Ableitungswerte; z.B.:

$$Q_{[a,b]}^{(3)}[f] = \frac{b-a}{2} [f(a) + f(b)] + \frac{(b-a)^2}{12} [f'(a) - f'(b)].$$

5.2 Summierte Quadraturformeln

Unter anderem weil für grössere n **negative** Gewichte α_i auftreten, ist die Erhöhung von n kein sinnvoller Weg zur Verbesserung der Approximationsgüte der Quadratur. Stattdessen unterteilen wir das Intervall in N Teilintervalle und wenden die Quadraturformel auf jedes Teilintervall einzeln an.

Achtung: Im folgenden bezeichnen x_j , $j = 0, \dots, N$, die Intervallenden der Teilintervalle. Bei gleichmässiger Unterteilung des Intervalls $[a, b]$ erhalten wir

$$x_j = a + jh, \quad j = 0, \dots, N, \quad h = \frac{b-a}{N}.$$

Anwendung einer Quadraturformel $Q_{[x_i, x_{i+1}]}^{(n)}[f]$ auf die Teilintervalle und Aufaddierung der Einzelbeiträge ergibt die **summierte** Quadraturformel:

$$Q_h^{(n)}[f] := \sum_{i=0}^{N-1} Q_{[x_i, x_{i+1}]}^{(n)}[f]. \quad (5.4)$$

Für $n=1$ erhält man die **summierte Trapezregel**

$$Q_h^{(1)}[f] = \sum_{i=0}^{N-1} \frac{x_{i+1} - x_i}{2} [f(x_i) + f(x_{i+1})] = \frac{h}{2} \left[f(a) + 2 \sum_{i=1}^{N-1} f(x_i) + f(b) \right],$$

und für $n=2$ die **summierte Simpson-Regel**

$$\begin{aligned} Q_h^{(2)}[f] &= \sum_{i=0}^{N-1} \frac{x_{i+1} - x_i}{6} [f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})] \\ &= \frac{h}{6} \left[f(a) + 2 \sum_{i=1}^{N-1} f(x_i) + 4 \sum_{i=0}^{N-1} f\left(\frac{x_i + x_{i+1}}{2}\right) + f(b) \right]. \end{aligned}$$

Satz 5.4 Für die summierten Trapez- und Simpsonregeln gelten die folgenden Fehlerschätzungen

$$\begin{aligned} |I_{[a,b]}[f] - Q_h^{(1)}[f]| &\leq \frac{h^2}{12} (b-a) \max_{x \in [a,b]} |f''(x)|, \quad f \in C^2[a, b], \\ |I_{[a,b]}[f] - Q_h^{(2)}[f]| &\leq \frac{h^4}{2880} (b-a) \max_{x \in [a,b]} |f^{(4)}(x)|, \quad f \in C^4[a, b]. \end{aligned}$$

Beweis. Anwendung von Satz 5.2 auf jedes Teilintervall ergibt

$$\begin{aligned} |I_{[a,b]}[f] - Q_h^{(1)}[f]| &= \left| \sum_{i=0}^{N-1} \frac{(x_{i+1} - x_i)^3}{12} f''(\xi_i) \right| \\ &\leq \sum_{i=0}^{N-1} \frac{h^3}{12} |f''(\xi_i)| \leq \frac{h^2}{12} (b-a) \max_{x \in [a,b]} |f''(x)|. \end{aligned}$$

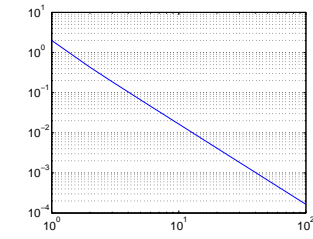
Der Beweis für die Simpson-Regel erfolgt analog. \square

Beispiel 5.5 Wir betrachten die summierte Trapez- und Simpsonregel zur Approximation von

$$\int_0^\pi \sin(x) \, dx = 2.$$

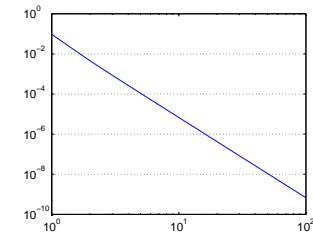
```
MATLAB
mn = 100; err = zeros(mn,1);
for n = 1:mn,
    err(n) = ...
        abs(2-trapez(@sin,0,pi,n));
end
loglog(1:mn,err),

function T = trapez(fun,a,b,n)
x = linspace(a,b,2*n+1);
f = feval(fun,x);
f(1) = f(1)/2; f(end) = f(end)/2;
T = (b-a) * sum(f) / n;
```



Fehler der summierten Trapezregel vs. n .

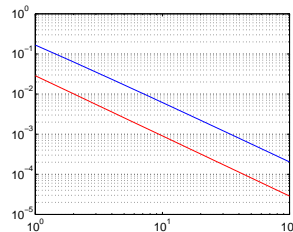
```
MATLAB
function T = simpson(fun,a,b,n)
x = linspace(a,b,2*n+1);
f = feval(fun,x);
f(2:2:end-1) = 4*f(2:2:end-1);
f(3:2:end-2) = 2*f(3:2:end-2);
T = (b-a) * sum(f) / n / 6;
```



Fehler der summ. Simpsonregel vs. n .

Wie zu erwarten konvergiert der Fehler bei der Simpsonregel wesentlich schneller gegen 0. Auch bezüglich der Anzahl der Funktionsauswertungen ist die Simpson- der Trapezregel weit überlegen. Sind die Glattheitsforderungen an f nicht erfüllt, so gehen die überragenden Vorteile der Simpsonregel verloren. Betrachte dazu

$$\int_0^1 \sqrt{x} \, dx = \frac{2}{3}.$$



Fehler der summ. Trapezregel (blau) und Simpsonregel (rot) vs. n . \diamond

Summierte Quadraturformeln lassen sich natürlich auch für offene Newton-Cotes-Formeln aufstellen. Zum Beispiel **summierte Mittelpunktsregel**:

$$Q_h^{(0)}[f] = \sum_{i=0}^{N-1} (x_{i+1} - x_i) f\left(\frac{x_i + x_{i+1}}{2}\right).$$

5.3 Das Rombergsche Integrationsverfahren

Das Prinzip der Extrapolation liefert in der Numerik eine wichtige Methode zur Genauigkeitsverbesserung. Aus einer gegebenen Folge von Approximationen wird durch geschickte Linearkombination eine genauere Approximation gewonnen. Zentrale Annahme bei Extrapolationsverfahren ist die Existenz einer *asymptotischen Entwicklung des Fehlers in h* .

Konkret betrachten wir nun die summierte Trapezregel $T(h) = Q_h^{(1)}$. Für $f \in C^{2p+2}([a, b])$ gilt

$$I - T(h) = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots + c_p h^{2p} + O(h^{2p+2}), \quad \text{für } h \rightarrow 0, \quad (5.5)$$

wobei $I = I_{[a,b]}[f]$.¹⁶ Wichtig für die folgende Argumentation ist keineswegs die genaue Kenntnis der Koeffizienten c_k sondern lediglich die Tatsache, dass die c_k *nicht* von h abhängen. Dann ergibt sich

$$I - T(h/2) = \frac{c_1}{4} h^2 + \tilde{c}_2 h^4 + \tilde{c}_3 h^6 + \dots + \tilde{c}_p h^{2p} + O(h^{2p+2}). \quad (5.6)$$

Linearkombination von (5.5) und (5.6) ergibt

$$I - \left[\frac{4}{3} T(h/2) - \frac{1}{3} T(h) \right] = \hat{c}_1 h^4 + \dots + \hat{c}_p h^{2p} + O(h^{2p+2}).$$

Also nähert $\frac{4}{3} T(h/2) - \frac{1}{3} T(h)$ das Integral I mit einer Genauigkeit von $O(h^4)$ an! Da bei Halbierung der Schrittweite h in der summierten Trapezregel die Anzahl der Funktionsauswertungen lediglich verdoppelt wird, ist die dadurch erreichte Fehlerreduktion eine sehr effiziente Genauigkeitssteigerung. Man beachte auch, dass die zur Bestimmung von $T(h)$ berechneten Funktionswerte bei der Bestimmung von $T(h/2)$ wiederverwendet werden können. Die Idee kann systematisch fortgesetzt werden. Setzt man $T_1(h) = \frac{4}{3} T(h/2) - \frac{1}{3} T(h)$, so gilt für $T_2(h) = \frac{16}{15} T_1(h/2) - \frac{1}{15} T_1(h)$ die Fehlerdarstellung

$$I - T_2(h) = \tilde{c}_1 h^6 + \dots + \tilde{c}_p h^{2p} + O(h^{2p+2}).$$

Allgemein berechnet man nach dem **Romberg-Schema** zunächst alle

$$T_0(2^{-i}h) := T(2^{-i}h), \quad i = 0, \dots, m, \quad (5.7)$$

mit der summierten Trapezregel, und danach rekursiv für $k = 1, 2, \dots, m$:

$$T_k(2^{-i}h) := \frac{4^k T_{k-1}(2^{-(i+1)}h) - T_{k-1}(2^{-i}h)}{4^k - 1}, \quad i = 0, \dots, m - k. \quad (5.8)$$

¹⁶Siehe zum Beispiel [Stoer/Bulirsch'83].

Man baut also sukzessiv, Spalte für Spalte, folgendes Tableau auf:

0	1	2	m
$T_0(h)$			
$T_0(2^{-1}h)$	$T_1(h)$		
$T_0(2^{-2}h)$	$T_1(2^{-1}h)$	$T_2(h)$	
\vdots	\vdots	\vdots	\ddots
$T_0(2^{-m}h)$	$T_1(2^{-(m-1)}h)$	$T_2(2^{-(m-2)}h)$	\dots $T_m(h)$

Als Approximation von I wählt man dann $T_m(h)$.

Das Romberg-Schema erinnert stark an Neville's Schema (siehe Abschnitt 3.2). Tatsächlich ist es ein Spezialfall. Um dies einzusehen, betrachte man die Stützstellen

$$\xi_0 = h^2, \quad \xi_1 = \left(\frac{h}{2}\right)^2, \quad \xi_2 = \left(\frac{h}{2^2}\right)^2, \quad \dots, \quad \xi_m = \left(\frac{h}{2^m}\right)^2,$$

und die zu interpolierenden Funktionswerte

$$\begin{aligned} y_0 &= T(\sqrt{\xi_0}) = T_0(h), \\ y_1 &= T(\sqrt{\xi_1}) = T_0(2^{-1}h), \\ &\vdots \\ y_m &= T(\sqrt{\xi_m}) = T_0(2^{-m}h). \end{aligned}$$

Jetzt wird das Neville-Schema angewendet, um das entsprechende Interpolationspolynom an der Stelle $\xi = 0$ auszuwerten.¹⁷ Zuerst setzen wir $P_{i0}(0) = y_i$; die erste Spalte des Neville-Schemas entspricht damit der ersten Spalte des Romberg-Schemas. Zur Berechnung der nächsten Spalten wenden wir die Rekursion (3.14) für $k = 1, \dots, m$ an:

$$P_{ik}(0) = \frac{(0 - \xi_i)P_{i+1,k-1}(0) - (0 - \xi_{i+k})P_{i,k-1}(0)}{\xi_{i+k} - \xi_i}.$$

Mit der Induktionsvoraussetzung $P_{i,k-1}(0) = T_{k-1}(2^{-i}h)$ für $i = 0, \dots, m - (k - 1)$, folgt

$$\begin{aligned} P_{ik}(0) &= \frac{-(2^{-i}h)^2 T_{k-1}(2^{-(i+1)}h) + (2^{-(i+k)}h)^2 T_{k-1}(2^{-i}h)}{(2^{-(i+k)}h)^2 - (2^{-i}h)^2} \\ &= \frac{-4^{-i} T_{k-1}(2^{-(i+1)}h) + 4^{-(i+k)} T_{k-1}(2^{-i}h)}{4^{-(i+k)} - 4^{-i}} \\ &= \frac{-T_{k-1}(2^{-(i+1)}h) + 4^{-k} T_{k-1}(2^{-i}h)}{4^{-k} - 1} \\ &= \frac{4^k T_{k-1}(2^{-(i+1)}h) - T_{k-1}(2^{-i}h)}{4^k - 1} = T_k(2^{-i}h), \end{aligned}$$

wobei im letzten Schritt die Definition (5.8) von $T_k(2^{-i}h)$ angewandt wurde.

Für $T_m(h)$ ergibt sich die folgende Fehlerschranke.

¹⁷Da $\xi = 0$ ausserhalb des Bereichs $[4^{-m}h^2, h^2]$ der Stützstellen liegt, spricht man in diesem Zusammenhang auch von **Extrapolation**.

Satz 5.6 Sei $f \in C^{2m+2}[a, b]$. Für den durch das Romberg-Schema berechneten Wert $T_m(h)$ gilt $I - T_m(h) = O((2^{-m}h)^{2m+2})$.

Beispiel 5.7 Wir betrachten wieder $\int_0^\pi \sin(x) dx$. Das Romberg-Schema liefert für $h = \pi/2$ das folgende Tableau:

	0	1	2	3
$2^{-1}\pi$	1.5708			
$2^{-2}\pi$	1.8961	2.004559		
$2^{-3}\pi$	1.9742	2.000269	$2 - 1.6 \cdot 10^{-5}$	
$2^{-4}\pi$	1.9936	2.000016	$2 - 2.5 \cdot 10^{-7}$	$2 + 1.6 \cdot 10^{-8}$

Für das Integral $\int_0^1 \sqrt{x} dx$ liefert das Romberg-Schema für $h = 1/2$ hingegen:

	0	1	2	3
2^{-1}	0.6036			
2^{-2}	0.6433	0.6565		
2^{-3}	0.6581	0.6631	0.6635	
2^{-4}	0.6636	0.6654	0.6656	0.6656

Hier verbessert Extrapolation die Approximation nur unwesentlich. \diamond

5.4 Gauss'sche Quadraturformeln

Die Ordnung einer Quadraturformel gibt an bis zu welchem Grad Polynome *exakt* integriert werden.

Definition 5.8 Eine Quadraturformel $Q_{[a,b]}$ hat **Ordnung $n + 1$** , wenn

$$I_{[a,b]}(p_n) = Q_{[a,b]}[p_n], \quad \forall p_n \in \mathbb{P}_n.$$

Alle interpolatorischen Quadraturformeln

$$Q^{(n)}[f] = \sum_{j=0}^n \alpha_j f(x_j), \quad \text{mit } \alpha_j = \int_a^b \ell_j(x) dx, \quad (5.9)$$

zu $n + 1$ verschiedenen Stützstellen $x_0, \dots, x_n \in [a, b]$ sind nach Konstruktion mindestens von Ordnung $n + 1$ (da Polynome vom Grad n exakt interpoliert werden). Für den Spezialfall der Newton-Cotes-Formeln mit *geradem* $n \geq 0$ haben wir gesehen¹⁸, dass Polynome aus \mathbb{P}_{n+1} exakt integriert werden, also Ordnung $n + 2$ erreicht wird. Man kann sogar noch höhere Ordnung erreichen, wenn man die Stützstellen geschickt wählt. Intuitiv vermutet man, dass wir bis zu Ordnung $2n + 2$ erzielen können, da die $n + 1$ Stützstellen und die $n + 1$ Gewichte zusammen $2n + 2$ Freiheitsgrade ergeben. (Die Situation erinnert an Tschebyscheff-Interpolation.¹⁹)

¹⁸Siehe Satz 5.2 für $n = 2$. Der Nachweis, dass eine Quadraturformel Ordnung k hat, ist wesentlich einfacher als die genaue Darstellung des Quadraturfehlers.

¹⁹Es ist aber zu beachten, dass Tschebyscheff-Knoten zu *keiner* höheren Ordnung führen würden.

Wir wollen im folgenden zeigen, dass es tatsächlich interpolatorische Quadraturformeln zu $n + 1$ Stützstellen gibt, welche die Maximalordnung $2n + 2$ haben. Sie heißen **Gauss'sche Quadraturformeln**. Dazu müssen wir die Stützstellen x_0, \dots, x_n in (5.9) so bestimmen, dass $Q^{(n)}$ alle Polynome vom Grad höchstens $2n + 1$ exakt integriert. Wegen Linearität reicht es aus irgendeine Basis von \mathbb{P}_{2n+1} zu überprüfen. Die Monombasis führt zu den $2n + 2$ Gleichungen

$$I_{[-1,1]}[x^k] = \alpha_0 x_0^k + \dots + \alpha_n x_n^k, \quad k = 0, \dots, 2n + 1, \quad (5.10)$$

die erfüllt sein müssen, um Ordnung $2n + 2$ zu erreichen. Der Einfachheit halber nehmen wir hier und im folgenden an, dass über das Intervall $[a, b] = [-1, 1]$ integriert wird. Die Lösung des nichtlinearen Gleichungssystems (5.10) ist für allgemeines n recht mühsam und mathematisch unbefriedigend. Es gibt einen wesentlich eleganteren Zugang, der im folgenden beschrieben werden soll. Die Grundidee dazu liefert der folgende Satz aus der Algebra.

Satz 5.9 Sei $p \in \mathbb{P}_{2n+1}$ und $q \in \mathbb{P}_{n+1}$. Dann gibt es genau ein $h \in \mathbb{P}_n$ und ein $r \in \mathbb{P}_n$, so dass $p = hq + r$.

Beweis. Polynomdivision. \square

Wir werden $q \in \mathbb{P}_{n+1}$ geschickt wählen. Um ein beliebiges Polynom $p \in \mathbb{P}_{2n+1}$ exakt zu integrieren, muss gelten

$$\begin{aligned} 0 &= \int_{-1}^1 p(x) \, dx - \sum_{j=0}^n \alpha_j p(x_j) \\ &= \int_{-1}^1 h(x)q(x) \, dx - \sum_{j=0}^n \alpha_j h(x_j)q(x_j) + \left(\int_{-1}^1 r(x) \, dx - \sum_{j=0}^n \alpha_j r(x_j) \right) \end{aligned} \quad (5.11)$$

Der Term in Klammern macht uns dabei am wenigsten Sorgen: bei der Wahl einer interpolatorischen Quadraturformel in $n + 1$ Stützstellen wird $r \in \mathbb{P}_n$ exakt integriert. Der Abgleich der anderen Terme wird mit der Wahl von q als Legendre-Polynom erreicht.

Definition 5.10 Das Legendre-Polynom $q_{n+1} \in \mathbb{P}_{n+1}$ erfüllt

$$\int_{-1}^1 q_{n+1}(x)h(x) \, dx = 0 \quad \forall h \in \mathbb{P}_n, \quad q_{n+1}(1) = 1. \quad (5.12)$$

Definieren wir das L_2 -Skalarprodukt

$$(p, q) = \int_{-1}^1 p(x)q(x) \, dx$$

auf dem Vektorraum \mathbb{P}_{n+1} , so sagt (5.12) aus, dass q_{n+1} orthogonal zu dem Unterraum \mathbb{P}_n sein muss. Die Existenz der Legendre-Polynome kann also einfach mit dem Gram-Schmidt-Orthogonalisierungsverfahren angewandt auf die Monombasis $1, x, \dots, x^{n+1}$ (siehe Abschnitt 0.8.2) nachgewiesen werden.

Satz 5.11 Die Legendre-Polynome q_0, q_1, \dots erfüllen die Dreitermrekursion

$$q_{n+1}(x) = \frac{2n+1}{n+1} x q_n(x) - \frac{n}{n+1} q_{n-1}(x), \quad q_0(x) = 1, \quad q_1(x) = x.$$

Beweis. Siehe Quarterioni et al. \square

Die ersten fünf Legendre-Polynome lauten nach Satz 5.11

$$\begin{aligned} q_0(x) &= 1 \\ q_1(x) &= x \\ q_2(x) &= \frac{1}{2}(3x^2 - 1) \\ q_3(x) &= \frac{1}{2}(5x^3 - 3x) \\ q_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3) \\ q_5(x) &= \frac{1}{8}(63x^5 - 70x^3 + 15x). \end{aligned}$$

Wählen wir $q = q_{n+1}$, so verschwindet der erste Term in (5.11). Wählen wir x_0, \dots, x_{n+1} als die Nullstellen von q_{n+1} , so verschwindet auch der zweite Term. Es lohnt sich also, die Nullstellen von q_{n+1} etwas näher zu betrachten.

Satz 5.12 Das Legendre-Polynom q_{n+1} hat genau $n + 1$ einfache Nullstellen in $(-1, 1)$.

Beweis. Wir definieren die Menge

$$N := \{\lambda \in (-1, 1) : \lambda \text{ ist Nullstelle ungerader Vielfachheit von } q_{n+1}\}$$

und setzen

$$\begin{aligned} h(x) &:= 1 && \text{für } N = \emptyset, \quad \text{und} \\ h(x) &:= \prod_{i=1}^m (x - \lambda_i) && \text{für } N = \{\lambda_1, \dots, \lambda_m\}. \end{aligned}$$

Dann ist das Polynom $q_{n+1} \cdot h \in \mathbb{P}_{n+m+1}$ reell und hat in $(-1, 1)$ keinen Vorzeichenwechsel. Es gilt

$$(q_{n+1}, h) = \int_{-1}^1 q_{n+1}(x)h(x) \, dx \neq 0.$$

Für $m \leq n$ wäre dies aber ein Widerspruch zu $q_{n+1} \perp \mathbb{P}_n$. Also ist $m > n$ und damit hat q_{n+1} mindestens $n + 1$ Nullstellen in $(-1, 1)$. Aus dem Fundamentalsatz der Algebra folgt weiterhin, dass q_{n+1} nicht nur mindestens sondern genau $n + 1$ Nullstellen in $(-1, 1)$ hat. \square

Insgesamt erhalten wir das folgende Resultat.

Satz 5.13 Seien $x_0, \dots, x_n \in (-1, 1)$ die Nullstellen des Legendre-Polynoms q_{n+1} und ℓ_0, \dots, ℓ_n die entsprechende Lagrange-Basis. Mit der Wahl $\alpha_j = \int_{-1}^1 \ell_j(x) \, dx$ hat die Gauss'sche Quadraturformel

$$Q^{(n)} = \alpha_0 f(x_0) + \dots + \alpha_n f(x_n)$$

die Ordnung $2n + 2$.

Für $n = 1$ und $n = 2$ ergeben sich die Quadraturformeln

$$Q^{(1)}[f] = f(-\sqrt{1/3}) + f(\sqrt{1/3}),$$

$$Q^{(2)}[f] = \frac{1}{9} \{5f(-\sqrt{3/5}) + 8f(0) + 5f(\sqrt{3/5})\}.$$

Wegen

$$0 < \int_{-1}^1 \ell_i(x)^2 dx = \sum_{j=0}^n \alpha_j \underbrace{\ell_i(x_j)^2}_{\delta_{ij}} = \alpha_i$$

folgt, dass die Gewichte immer positiv sind. Wir erhalten also keine numerische Instabilitäten für grössere n wie bei den Newton-Cotes-Formeln! Für allgemeine n lassen sich die Gewichte auch aus dem linearen Gleichungssystem

$$\sum_{j=0}^n \alpha_j x_j^i = \int_{-1}^1 x^i dx = \frac{1}{i+1} (1 - (-1)^{i+1}), \quad i = 0, \dots, n$$

bestimmen. Die verlässliche Bestimmung der Nullstellen von q_{n+1} ist schwieriger und wird entscheidend durch das folgende Resultat vereinfacht.

Satz 5.14 (Golub/Welsh) Die Nullstellen x_0, \dots, x_{n+1} sind die Eigenwerte der Matrix

$$J = \begin{pmatrix} 0 & b_1 & & & \\ b_1 & 0 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & & b_n & 0 \\ & & & & 0 \end{pmatrix}$$

wobei

$$b_j = j \left(4 \sum_{i=1}^n i^2 - 1 \right)^{-1/2}.$$

MATLAB

```
function [x,a]=gaussQuad(n)
% Berechnet Knoten x_j und Gewichte a_j der Gauss'schen Quadratur.
b = 1:n; b = b./sqrt(4*b.*b-1);
J=diag(b,-1)+diag(b,1); [ev,ew]=eig(J);
x=diag(ew); a=(2*(ev(1,:)).*ev(1,:));
```

Der folgende Satz, den wir ohne Beweis angeben, zeigt, dass die Gauss-Quadratur nicht nur von hoher Ordnung ist sondern auch zu einem kleineren Quadraturfehler als eine geschlossene Newton-Cotes-Formel mit der gleichen Anzahl der Stützstellen führt.

Satz 5.15 (Fehler der Gauss'schen Quadratur) Für $f \in C^{2n+2}[-1, 1]$ existiert ein $\xi \in (-1, 1)$, so dass

$$R^{(n)}[f] := Q^{(n)}[f] - \int_{-1}^1 f(x) dx = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_{-1}^1 \prod_{j=0}^n (x-x_j)^2 dx$$

$$= \frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3} f^{(2n+2)}(\xi).$$

Beispiel: Für $n = 1$ ist die geschlossene Newton-Cotes-Formel die Trapezregel und aus Satz 5.2 ergibt sich bei $[a, b] = [-1, 1]$ ein Fehler von $2/3 f''(\xi)$. Bei der Gauss-Quadratur ergibt sich für $n = 1$ aus Satz 5.15 hingegen ein Fehler von nur $1/135 f^{(4)}(\xi)$. \diamond

Satz 5.16 (Konvergenz der Gauss'schen Quadratur) Für jede Funktion $f \in C^0[-1, 1]$ konvergiert $Q^{(n)}[f] \rightarrow \int_{-1}^1 f(x) dx$ für $n \rightarrow \infty$.

Beweis. Für die Gewichte der Gauss-Legendre-Formel gilt

$$Q^{(n)}[f] = \sum_{i=0}^n \alpha_i^{(n)} f(x_i^{(n)}), \quad \alpha_i^{(n)} > 0, \quad \sum_{i=0}^n \alpha_i^{(n)} = 2.$$

Sei $\varepsilon > 0$ beliebig vorgegeben. Nach dem Weierstrass'schen Approximationssatz gibt es ein $p_\varepsilon \in \mathbb{P}_m$ (für $m \equiv m(\varepsilon)$ hinreichend gross), so dass

$$\max_{-1 \leq x \leq 1} |f(x) - p_\varepsilon(x)| \leq \frac{\varepsilon}{4}.$$

Es gilt $R^{(n)}(p_\varepsilon) = 0$ für $2n+2 > m$. Für solche n ist also

$$|I[f] - Q^{(n)}[f]| \leq \underbrace{|I[f - p_\varepsilon]|}_{\leq \frac{\varepsilon}{4 \cdot 2}} + \underbrace{|I[p_\varepsilon] - Q^{(n)}[p_\varepsilon]|}_{=0} + \underbrace{|Q^{(n)}[p_\varepsilon - f]|}_{\leq \frac{\varepsilon}{4 \cdot 2}} \leq \varepsilon.$$

Wegen der beliebigen Wahl von $\varepsilon > 0$ haben wir also Konvergenz $Q^{(n)}[f] \rightarrow I[f]$ für $n \rightarrow \infty$. \square

Gauss'sche Quadraturformeln über einem beliebigen (beschränkten) Intervall $[a, b]$ gewinnt man durch Anwendung der Koordinatentransformation $\varphi: [-1, 1] \rightarrow [a, b]$,

$$\boxed{y = \varphi(x) = \frac{b-a}{2} x + \frac{b+a}{2}}. \quad (5.13)$$

Es ist dann

$$\int_a^b f(y) dy = \frac{b-a}{2} \int_{-1}^1 f(\varphi(x)) dx$$

$$= \frac{b-a}{2} \sum_{i=0}^n \alpha_i f(\varphi(x_i)) + \frac{b-a}{2} R^{(n)}[f(\varphi(\cdot))]$$

wobei

$$R^{(n)}[f(\varphi(\cdot))] = \frac{2^{(2n+3)}(n+1)!^4}{(2n+3)(2n+2)!^3} \underbrace{\frac{d^{2n+2}}{dx^{2n+2}} f(\varphi(\zeta))}_{\left(\frac{b-a}{2}\right)^{2n+2} f^{(2n+2)}(\varphi(\zeta))}.$$

D.h. die Stützstellen und Gewichte der Quadraturformel $(2n+2)$ -ter Ordnung über $[a, b]$,

$$Q^{(n)}[f] = \sum_{i=0}^n \tilde{\alpha}_i f(\tilde{x}_i),$$

sind gegeben durch

$$\tilde{x}_i = \frac{b-a}{2} x_i + \frac{b+a}{2}, \quad \tilde{\alpha}_i = \frac{b-a}{2} \alpha_i, \quad i = 0, \dots, n.$$

Für $n = 1$ und $n = 2$ erhalten wir mit $c = \frac{b+a}{2}$ und $h = \frac{b-a}{2}$ die Quadraturformeln

$$Q^{(1)}[f] = \frac{b-a}{2} \{f(c-h\sqrt{1/3}) + f(c+h\sqrt{1/3})\},$$

$$Q^{(2)}[f] = \frac{b-a}{18} \{5f(c-h\sqrt{3/5}) + 8f(c) + 5f(c+h\sqrt{3/5})\}.$$

Die zugehörigen summierten Gauss'schen Quadraturformeln haben die Gestalt ($x_j = a + jh$, $h = (b-a)/N$):

$$Q_h^{(1)}[f] = \frac{h}{2} \sum_{j=0}^{N-1} \{f(x_j + h') + f(x_{j+1} - h')\}$$

mit $h' = (\frac{1}{2} - \frac{1}{2\sqrt{3}})h \sim 0.2113249h$,

$$Q_h^{(2)}[f] = \frac{h}{18} \sum_{j=0}^{N-1} \{5f(x_j + h') + 8f(x_j + \frac{1}{2}h) + 5f(x_{j+1} - h')\}$$

mit $h' = (\frac{1}{2} - \frac{1}{2}\sqrt{\frac{2}{5}})h \sim 0.1127012h$.

Bemerkung 5.17 Es kann keine Quadraturformel in $n + 1$ Punkten x_0, \dots, x_n geben, die noch höhere Ordnung hat. Wäre nämlich $Q^{(n)}$ von Ordnung höher als $2n + 2$, d.h. insbesondere also exakt für das Polynom

$$p(x) = \prod_{i=0}^n (x - x_i)^2 \in \mathbb{P}_{2n+2},$$

so ergäbe sich der Widerspruch

$$0 < \int_a^b p(x) dx = Q^{(n)}[p] = 0.$$

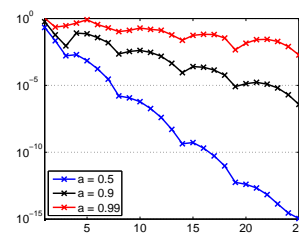
5.5 Verschiedenes*

5.5.1 Periodische Funktionen

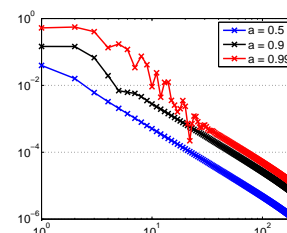
Die summierte Trapezregel eignet sich hervorragend zur Integration periodischer Funktionen. Als Beispiel betrachten wir die Funktion

$$f(x) = \frac{1}{\sqrt{1 - a \sin(x-1)}}.$$

Diese Funktion ist periodisch auf dem Intervall $[0, 2\pi]$. Für $a = 1$ besitzt die Funktion eine Singularität an der Stelle $2\pi/4 + 1 \approx 2.57$.



Quadraturfehler vs. n für $\int_0^{2\pi} f(x) dx$



Quadraturfehler vs. n für $\int_0^{\pi} f(x) dx$

Auf der linken Seite ist das Intervall so gewählt, dass genau über eine Periode integriert wird. Es ist zu beobachten, dass der Quadraturfehler *exponentiell schnell* gegen 0 konvergiert (schneller als jedes Polynom in n^{-1}). Auf der rechten Seite (man beachte, dass hier im Gegensatz zur linken Seite die n -Achse logarithmisch skaliert ist), konvergiert der Quadraturfehler hingegen nur algebraisch gegen 0 ($O(n^{-2})$ – wie aus Satz 5.4 zu erwarten).

Was steckt aber hinter der schnellen Konvergenz im periodischen Fall? Um dies zu klären, betrachten wir die *trigonometrische Interpolation* von $f(x)$ (siehe Abschnitt 4.2):

$$t_n(x) = \sum_{j=0}^n c_j e^{ijx}, \quad c_j \in \mathbb{C}.$$

Für eine periodische Funktion $f \in C^\infty([-\infty, \infty])$ mit Periode 2π konvergiert $t_n(x)$ exponentiell gegen f für $n \rightarrow \infty$.²⁰ Auf der einen Seite gilt

$$\int_0^{2\pi} e^{ikx} dx = \begin{cases} 0 & \text{für } k = 1, \dots, n-1, \\ 2\pi & \text{für } k = 0, \end{cases}$$

und auf der anderen Seite gilt für $h = 2\pi/n$,

$$Q_h^{(1)}[e^{ikx}] = \frac{2\pi}{n} \sum_{j=0}^n e^{2\pi ijk/n} = \begin{cases} 0 & \text{für } k = 1, \dots, n-1, \\ 2\pi & \text{für } k = 0, \end{cases},$$

wobei wir Lemma 4.3 verwendet haben. Damit integriert die summierte Trapezregel mit $n + 1$ gleichverteilten Stützstellen das gesamte trigonometrische Polynom $t_n(x)$ *exakt!* Aus der exponentiellen Konvergenz von t_n folgt so die exponentielle Konvergenz von $Q_h^{(1)}$.

Im periodischen Fall wäre es eher schädlich, die Trapezregel durch eine Quadraturformel höherer Ordnung zu ersetzen.

Bemerkung 5.18 Entscheidend ist auch hier wieder einmal die Glattheit von f . Ist f nicht genügend oft differenzierbar, so geht die exponentielle Konvergenz verloren (siehe obiges Beispiel für $a \rightarrow 1$).

²⁰Siehe z.B. Abschnitt 10.9 in Quarteroni et al.

5.5.2 Singuläre Integrale

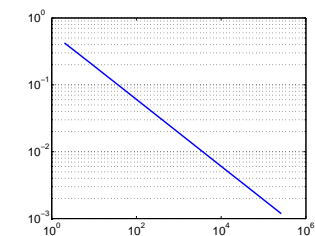
Integrierbare Singularitäten lassen sich auch numerisch approximieren. Zum Beispiel konvergiert die summierte Mittelpunktsregel für

$$\int_0^1 \frac{1}{\sqrt{x}} dx = 2.$$

```

MATLAB
function singmidpoint,
mn = 2.^[1:18]; err = [];
for n = mn,
    err = [ err; abs(2 - ...
        midpoint(@sing,0,1,n) ) ];
end
loglog(mn,err),
function y = sing(x)
y = 1./sqrt(x);
function M = midpoint(fun,a,b,n)
h = (b-a)/n; x = a+h/2:h:b-h/2;
f = feval(fun,x); M = h * sum(f);

```



Fehler der summierten Mittelpunktsregel vs. n .

Die Konvergenz ist aber sehr langsam. Dies kann entweder mit Variablentransformation oder mit adaptiver Quadratur vermieden werden.

5.5.3 Zweidimensionale Integrale

In der Praxis spielen Integrale im 2D, 3D oder, allgemeiner, in hochdimensionalen Räumen \mathbb{R}^d eine wichtige Rolle. Mittels einer Gebietesunterteilung und einer allfälligen Transformation zieht man sich dabei typischerweise auf ein Standardgebiet, z.B. den Einheitswürfel, zurück und entwickelt Quadraturformeln für das Standardgebiet. Die prinzipielle Vorgehensweise zur Entwicklung solcher Formeln soll im 2D für das Einheitsquadrat und das Einheitsdreieck demonstriert werden.

Einheitsquadrat. Wir betrachten das Integral

$$\int_0^1 \int_0^1 f(x, y) dx dy. \quad (5.14)$$

Sei

$$Q_m[g] = \sum_{i=0}^m \alpha_i g(x_i)$$

eine Quadraturformel für $\int_0^1 g(x) dx$. Bezeichnen wir das eindimensionale Teilintegral von (5.14) mit

$$F(y) = \int_0^1 f(x, y) dx,$$

so erhalten wir mit der Anwendung von Q_m in x - und y -Richtung die folgende Produkt-Quadraturregel $Q_{m \times m}[f]$:

$$\begin{aligned} \int_0^1 \int_0^1 f(x, y) dx dy &= \int_0^1 F(y) dy \approx \sum_{j=0}^m \alpha_j F(x_j) \\ &= \sum_{j=0}^m \alpha_j \int_0^1 f(x, x_j) dx \approx \sum_{j=0}^m \alpha_j \sum_{i=0}^m \alpha_i f(x_i, x_j) \\ &= \sum_{i,j=0}^m \alpha_i \alpha_j f(x_i, x_j) =: Q_{m \times m}[f]. \end{aligned}$$

Bemerkung 5.19 Für den Einheitswürfel im \mathbb{R}^d erhält man mit obiger Strategie m^d aufzusummierende Terme, der Aufwand wächst also sehr schnell mit der Dimension an. Diesem Fluch der Dimension kann mit der Monte-Carlo-Integration oder mit Sparse Grids begegnet werden. Beide Ansätze sind auf eine gewisse Glattheit der zu integrierenden Funktion f angewiesen.

Einheitsdreieck Der obige Zugang lässt sich nicht sinnvoll auf Dreiecke erweitern. Stattdessen sucht man hier (ähnlich wie bei der Gauss-Quadratur) Quadraturformeln die alle Monome der Form $x^{k_1} y^{k_2}$, $k_1 + k_2 \leq M$ für ein gewisses M exakt integrieren. Typische Formeln für das Einheitsdreieck $\{(x, y) : x + y \leq 1\}$ sind:

1. $Q[f] = \frac{1}{2} f(1/3, 1/3)$,
2. $Q[f] = \frac{1}{6} [f(0, 0) + f(1, 0) + f(0, 1)]$,
3. $Q[f] = \frac{1}{6} [f(1/2, 0) + f(0, 1/2) + f(1/2, 1/2)]$,
4. $Q[f] = \frac{1}{6} [f(1/6, 1/6) + f(2/3, 1/6) + f(1/6, 2/3)]$.

Man rechnet einfach nach, dass die Monome $1, x, y$ durch 1 und 2 exakt integriert werden ($M = 1$) und dass die Monome $1, x, y, xy, x^2, y^2$ durch 3 und 4 exakt integriert werden ($M = 2$).

Kapitel 6

Lösung nichtlinearer Gleichungssysteme

Dieses Kapitel behandelt die numerische Lösung nichtlinearer Gleichungen, wie z.B.

$$\frac{1}{4} \sin(\pi x) \cos(\pi x) - \frac{3x - 1}{3} = 0,$$

und Systeme nichtlinearer Gleichungen, wie z.B.

$$\begin{pmatrix} 6x_1 - \cos x_1 - 2x_2 \\ 8x_2 - x_1 x_2^2 - \sin x_1 \end{pmatrix} = \mathbf{0}.$$

6.1 Allgemeines zu Iterationsverfahren

Sei $F : D \rightarrow \mathbb{R}^n$ mit $D \subseteq \mathbb{R}^n$. Ausgehend von einem Startwert $x^{(0)} \in D$ konstruiert ein **Iterationsverfahren** zur Lösung von $F(x) = 0$ eine Folge $x^{(1)}, x^{(2)}, \dots \in D$.

Definition 6.1 Ein Iterationsverfahren heisst **konvergent**, wenn die Folge $x^{(k)}$ gegen ein $x^* \in D$ konvergiert und $F(x^*) = 0$.

Die Konvergenz eines Verfahrens hängt wesentlich von der Wahl des Startwerts $x^{(0)}$ ab.

Definition 6.2 Ein Iterationsverfahren **konvergiert lokal** gegen $x^* \in D$, falls es eine offene Umgebung $U \subseteq D$ um x^* gibt, so dass das Verfahren für jeden Startwert $x^{(0)} \in U$ gegen x^* konvergiert. Ein Iterationsverfahren **konvergiert global** gegen $x^* \in D$, wenn das Verfahren für jeden Startwert $x^{(0)} \in D$ gegen x^* konvergiert.

Wie wir später sehen werden, ist globale Konvergenz oft nur sehr schwierig zu erreichen bzw. nachzuweisen. Man wird sich deshalb mit Verfahren begnügen müssen, die zumindest lokal konvergieren. Zur Klassifizierung der Konvergenzgeschwindigkeit führen wir die folgenden Begrifflichkeiten ein.

Definition 6.3 Eine Folge $x^{(k)} \in \mathbb{R}^n, k = 0, 1, 2, \dots$, konvergiert **linear** gegen $x^* \in \mathbb{R}^n$, falls es $L < 1$ und $k_0 \in \mathbb{N}_0$ gibt, so dass

$$\|x^{(k+1)} - x^*\| \leq L \|x^{(k)} - x^*\|, \quad \forall k \geq k_0, \quad (6.1)$$

für eine Vektornorm $\|\cdot\|$ auf \mathbb{R}^n .

Die Wahl der Norm ist für die qualitative Aussage von Definition 6.3 unwesentlich²¹. Lineare Konvergenz ist in der Praxis oft unbefriedigend langsam. Ist insbesondere das kleinste zulässige L in (6.1) sehr nahe bei 1, so wird die Folge fast stagnieren.

Definition 6.4 Eine konvergente Folge $x^{(k)}, k = 0, 1, 2, \dots \in \mathbb{R}^n$ konvergiert in **p-ter Ordnung** mit $p > 1$ gegen x^* , falls es ein $C > 0$ gibt, so dass

$$\|x^{(k+1)} - x^*\| \leq C \|x^{(k)} - x^*\|^p.$$

Beispiel 6.5 Betrachte die folgende Iteration zur Berechnung von \sqrt{a} für $a > 0$:

$$x^{(k+1)} = \frac{1}{2} \left(x^{(k)} + \frac{a}{x^{(k)}} \right)$$

Wir haben

$$|x^{(k+1)} - \sqrt{a}| = \frac{1}{2x^{(k)}} |x^{(k)} - \sqrt{a}|^2 \leq \frac{1}{2a} |x^{(k)} - \sqrt{a}|^2,$$

für $k \geq 1$. Damit folgt quadratische Konvergenz. Iterierte für $a = 2$ und $x^{(0)} = 2$:

k	$x^{(k)}$	$e^{(k)} := x^{(k)} - \sqrt{2}$	$\log \frac{ e^{(k)} }{ e^{(k-1)} } : \log \frac{ e^{(k-1)} }{ e^{(k-2)} }$
0	2.0000000000000000	0.58578643762690485	
1	1.5000000000000000	0.08578643762690485	
2	1.4166666666666666	0.00245310429357137	1.850
3	1.41421568627450966	0.00000212390141452	1.984
4	1.41421356237468987	0.0000000000159472	2.000
5	1.41421356237309492	0.0000000000000022	0.630

In jedem Iterationsschritt verdoppelt sich also die Anzahl der gültigen Dezimalstellen, bis schliesslich für $k = 5$ der Einfluss der Rundungsfehler die Konvergenz begrenzt. \diamond

6.2 Fixpunktiteration im Eindimensionalen

Viele der im weiteren besprochenen Verfahren zur Lösung einer nichtlinearen Gleichung $f(x) = 0$ lassen sich als Spezialfall einer Fixpunktiteration auffassen. Zu einer **Fixpunktgleichung**

$$\Phi(x) = x \quad (6.2)$$

²¹Gilt (6.1) für eine gewählte Vektornorm, so folgt aus der Äquivalenz der Vektornormen auf dem \mathbb{R}^n , dass (6.1) auch für jede beliebige andere Vektornorm gilt. Qualitative Konvergenzaussagen sind also unabhängig von der Wahl der Norm.

und einem Startwert $x^{(0)} \in \mathbb{R}$ lautet die zugehörige **Fixpunktiteration**

$$x^{(k+1)} = \Phi(x^{(k)}), \quad k = 0, 1, 2, \dots \tag{6.3}$$

Definition 6.6 Eine Fixpunktgleichung (6.2) heisst **konsistent**, wenn

$$f(x) = 0 \iff \Phi(x) = x.$$

Zu einer gegebenen Gleichung $f(x) = 0$ ist die Umformulierung als konsistente Fixpunktgleichung nicht eindeutig. Unterschiedliche Formulierungen führen zu unterschiedlich guten Iterationen.

Beispiel 6.7 Die nichtlineare Gleichung $xe^x - 1 = 0$ hat die Lösung²²

$$x^* = 0.56714329040978 \dots$$

Konsistente Fixpunktformulierungen dieser Gleichung sind z.B.

$$\Phi_1(x) = e^{-x}, \quad \Phi_2(x) = \frac{1+x}{1+e^x}, \quad \Phi_3(x) = x+1 - xe^x. \tag{6.4}$$

Die folgenden Tabellen zeigen das Verhalten der entsprechenden Fixpunktiterationen für den Startwert $x^{(0)} = 0.5$.

k	$x^{(k+1)} := \Phi_1(x^{(k)})$	$x^{(k+1)} := \Phi_2(x^{(k)})$	$x^{(k+1)} := \Phi_3(x^{(k)})$
0	0.5000000000000000	0.5000000000000000	0.5000000000000000
1	0.606530659712633	0.566311003197218	0.675639364649936
2	0.545239211892605	0.567143165034862	0.347812678511202
3	0.579703094878068	0.567143290409781	0.855321409174107
4	0.560064627938902	0.567143290409784	-0.156505955383169
5	0.571172148977215	0.567143290409784	0.977326422747719
6	0.564862946980323	0.567143290409784	-0.619764251895580
7	0.568438047570066	0.567143290409784	0.713713087416146
8	0.566409452746921	0.567143290409784	0.256626649129847
9	0.567559634262242	0.567143290409784	0.924920676910549
10	0.566907212935471	0.567143290409784	-0.407422405542253

Die Verläufe der entsprechenden Fehler:

k	$ x^{(k+1)} - x^* $	$ x^{(k+1)} - x^* $	$ x^{(k+1)} - x^* $
0	0.067143290409784	0.067143290409784	0.067143290409784
1	0.039387369302849	0.000832287212566	0.108496074240152
2	0.021904078517179	0.000000125374922	0.219330611898582
3	0.012559804468284	0.000000000000003	0.288178118764323
4	0.007078662470882	0.000000000000000	0.723649245792953
5	0.004028858567431	0.000000000000000	0.410183132337935
6	0.002280343429460	0.000000000000000	1.186907542305364
7	0.001294757160282	0.000000000000000	0.146569797006362
8	0.000733837662863	0.000000000000000	0.310516641279937
9	0.000416343852458	0.000000000000000	0.357777386500765
10	0.000236077474313	0.000000000000000	0.974565695952037

²²Die **Lambert-W-Funktion** $x = W(y)$ ist als die reelle Lösung von $xe^x = y$ definiert. Für das Beispiel hat man also $x^* = W(1) \approx 0.56714329040978$.

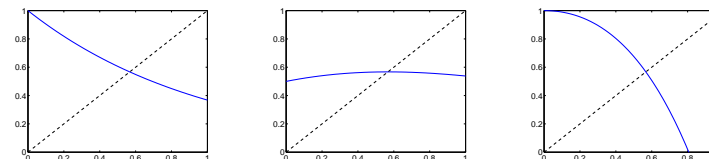


Abbildung 6.1. Von links nach rechts: Φ_1, Φ_2, Φ_3 wie in (6.4).

Offensichtlich ist in Beispiel 6.7 die Fixpunktiteration mit Φ_1 langsamer konvergent als diejenige mit Φ_2 , und die Fixpunktiteration mit Φ_3 ist vollkommen unbrauchbar. Konsistenz allein ist also noch kein Garant für Konvergenz. Wie man sich geometrisch (siehe Abbildung 6.1) klar machen kann, hängt die lokale Konvergenz von der Steigung von Φ um x^* ab.

Satz 6.8 Sei $\Phi : \mathbb{R} \rightarrow \mathbb{R}$ und $\Phi(x^*) = x^*$. Ist Φ in einer Umgebung von x^* stetig differenzierbar und gilt $|\Phi'(x^*)| < 1$, dann existiert ein $\epsilon > 0$, so dass die Fixpunktiteration für jeden Startwert $x^{(0)}$ mit $|x^{(0)} - x^*| \leq \epsilon$ linear gegen x^* konvergiert.

Beweis. Wegen der Stetigkeit von Φ' existiert ein $\epsilon > 0$, so dass

$$L = \max_{x \in B_\epsilon} |\Phi'(x)| < 1, \quad B_\epsilon := [x^* - \epsilon, x^* + \epsilon]. \tag{6.5}$$

Nach dem Mittelwertsatz existiert für jedes $x, y \in B_\epsilon$ ein $\xi \in B_\epsilon$, so dass

$$|\Phi(x) - \Phi(y)| = |\Phi'(\xi)(x - y)| \leq L|x - y|.$$

Für $x^{(0)} \in B_\epsilon$ erhält man

$$|x^{(1)} - x^*| = |\Phi(x^{(0)}) - \Phi(x^*)| \leq L|x^{(0)} - x^*| < \epsilon,$$

die nächste Iterierte $x^{(1)}$ liegt also auch in B_ϵ . So fortfahrend kann man zeigen, dass alle Iterierten in B_ϵ liegen und

$$|x^{(k+1)} - x^*| = |\Phi(x^{(k)}) - \Phi(x^*)| \leq L|x^{(k)} - x^*| \leq L^{k+1}|x^{(0)} - x^*|$$

für alle $k \geq 0$. Also folgt (lineare) Konvergenz $x^{(k)} \xrightarrow{k \rightarrow \infty} x^*$. \square

Der lineare Konvergenzfaktor L in (6.5) kann beliebig nahe an $|\Phi'(x^*)|$ gewählt werden, mittels einer genügend kleinen Wahl von ϵ . Im Fall $\Phi'(x^*) = 0$ kann noch mehr gezeigt werden (vgl. Φ_2 aus Beispiel 6.7).

Satz 6.9 Sei $\Phi : \mathbb{R} \rightarrow \mathbb{R}$ mit $\Phi(x^*) = x^*$. Sei Φ $(m + 1)$ -Mal stetig in einer Umgebung von x^* differenzierbar und

$$\Phi'(x^*) = \Phi''(x^*) = \dots = \Phi^{(m)}(x^*) = 0.$$

Dann konvergiert die Fixpunktiteration lokal in $(m + 1)$ -ter Ordnung gegen x^* .

Beweis. Mit der Taylor-Entwicklung von Φ um x^* haben wir

$$\begin{aligned} x^{(k+1)} - x^* &= \Phi(x^{(k)}) - \Phi(x^*) \\ &= \sum_{l=1}^m \frac{1}{l!} \Phi^{(l)}(x^*) (x^{(k)} - x^*)^l + O(|x^{(k)} - x^*|^{m+1}) \\ &= O(|x^{(k)} - x^*|^{m+1}), \end{aligned}$$

und damit – laut Definition – Konvergenz von Ordnung $m+1$. \square

Für Φ_2 in Beispiel 6.7 impliziert Satz 6.9 also lokal quadratische Konvergenz.

6.3 Fixpunktverfahren im Mehrdimensionalen

Im \mathbb{R}^n folgt die Konvergenz der Fixpunktiteration aus dem allgemeineren **Banach'schen Fixpunktsatz**, der hier zur Vollständigkeit für den Spezialfall \mathbb{R}^n angegeben ist.

Satz 6.10 Sei $E \subseteq \mathbb{R}^n$ abgeschlossen und $\Phi : E \rightarrow E$. Ferner sei Φ eine **Kontraktion** auf E , d.h. es gibt ein $L < 1$, so dass

$$\|\Phi(\underline{x}) - \Phi(\underline{y})\| \leq L \|\underline{x} - \underline{y}\|, \quad \forall \underline{x}, \underline{y} \in E, \quad (6.6)$$

für eine Vektornorm $\|\cdot\|$ auf \mathbb{R}^n . Dann gelten die folgenden Aussagen.

1. Es existiert genau ein $\underline{x}^* \in E$ mit $\Phi(\underline{x}^*) = \underline{x}^*$.

2. Für beliebiges $x^{(0)} \in E$ konvergiert

$$\underline{x}^{(k+1)} = \Phi(\underline{x}^{(k)}), \quad k = 0, 1, 2, \dots$$

gegen \underline{x}^* .

3. Es gilt die a priori Fehlerabschätzung

$$\|\underline{x}^{(k)} - \underline{x}^*\| \leq \frac{L^k}{1-L} \|\underline{x}^{(1)} - \underline{x}^{(0)}\|.$$

4. Es gilt die a posteriori Fehlerabschätzung

$$\|\underline{x}^{(k)} - \underline{x}^*\| \leq \frac{L}{1-L} \|\underline{x}^{(k)} - \underline{x}^{(k-1)}\|.$$

Beweis. Wegen der Kontraktionseigenschaft gilt

$$\begin{aligned} \|\underline{x}^{(k+1)} - \underline{x}^{(k)}\| &= \|\Phi(\underline{x}^{(k)}) - \Phi(\underline{x}^{(k-1)})\| \\ &\leq L \|\underline{x}^{(k)} - \underline{x}^{(k-1)}\| \leq \dots \leq L^k \|\underline{x}^{(1)} - \underline{x}^{(0)}\|. \end{aligned}$$

Hiermit ergibt sich

$$\begin{aligned} \|\underline{x}^{(m+k)} - \underline{x}^{(k)}\| &= \|\underline{x}^{(m+k)} - \underline{x}^{(m+k-1)} + \underline{x}^{(m+k-1)} + \dots + \underline{x}^{(k+1)} - \underline{x}^{(k)}\| \\ &\leq \|\underline{x}^{(m+k)} - \underline{x}^{(m+k-1)}\| + \dots + \|\underline{x}^{(k+1)} - \underline{x}^{(k)}\| \\ &\leq (L^{m+k-1} + \dots + L^k) \|\underline{x}^{(1)} - \underline{x}^{(0)}\| \\ &\leq L^k \frac{1-L^m}{1-L} \|\underline{x}^{(1)} - \underline{x}^{(0)}\| \\ &\leq \frac{L^k}{1-L} \|\underline{x}^{(1)} - \underline{x}^{(0)}\|. \end{aligned} \quad (6.7)$$

Wegen $L < 1$ folgt, dass $\underline{x}^{(k)}$ eine Cauchy-Folge ist. Wegen der Abgeschlossenheit von E existiert ein $\underline{x}^* \in E$, so dass

$$\lim_{k \rightarrow \infty} \underline{x}^{(k)} = \underline{x}^*.$$

Für diesen Grenzwert gilt²³

$$\begin{aligned} \underline{x}^* - \Phi(\underline{x}^*) &= \underline{x}^* - \Phi(\lim_{k \rightarrow \infty} \underline{x}^{(k)}) = \underline{x}^* - \lim_{k \rightarrow \infty} \Phi(\underline{x}^{(k)}) \\ &= \underline{x}^* - \lim_{k \rightarrow \infty} \underline{x}^{(k+1)} = \underline{x}^* - \underline{x}^* = 0 \end{aligned}$$

und damit ist $\Phi(\underline{x}^*) = \underline{x}^*$. Erfülle $\underline{x}^{**} \in E$ ebenfalls $\Phi(\underline{x}^{**}) = \underline{x}^{**}$. Dann folgt aus der Kontraktionseigenschaft

$$\|\underline{x}^* - \underline{x}^{**}\| = \|\Phi(\underline{x}^*) - \Phi(\underline{x}^{**})\| \leq L \|\underline{x}^* - \underline{x}^{**}\|.$$

Da $L < 1$, kann dies nur für $\underline{x}^* = \underline{x}^{**}$ gelten. Die a priori Fehlerabschätzung folgt aus (6.7) für $m \rightarrow \infty$. Der Nachweis der a posteriori Fehlerabschätzung ist Übung. \square

Die Kontraktionseigenschaft (6.6) ist im allgemeinen schwer überprüfbar. Um eine einfacher nachzuprüfende Forderung zu erhalten, betrachten wir das Differential von

$$\Phi(\underline{x}) = \begin{pmatrix} \Phi_1(\underline{x}) \\ \vdots \\ \Phi_n(\underline{x}) \end{pmatrix} = \begin{pmatrix} \Phi_1(x_1, \dots, x_n) \\ \vdots \\ \Phi_n(x_1, \dots, x_n) \end{pmatrix}, \quad \Phi_j : \mathbb{R}^n \rightarrow \mathbb{R},$$

an der Stelle \underline{x}^* :

$$\Phi'(\underline{x}^*) = \begin{pmatrix} \frac{\partial}{\partial x_1} \Phi_1(\underline{x}^*) & \cdots & \frac{\partial}{\partial x_n} \Phi_1(\underline{x}^*) \\ \vdots & & \vdots \\ \frac{\partial}{\partial x_1} \Phi_n(\underline{x}^*) & \cdots & \frac{\partial}{\partial x_n} \Phi_n(\underline{x}^*) \end{pmatrix}.$$

Korollar 6.11 Sei $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\Phi(\underline{x}^*) = \underline{x}^*$ für ein $\underline{x}^* \in \mathbb{R}^n$, und Φ stetig differenzierbar in einer Umgebung von \underline{x}^* . Für die von einer Vektornorm $\|\cdot\|$ auf \mathbb{R}^n induzierte Matrixnorm gelte

$$\|\Phi'(\underline{x}^*)\| < 1.$$

Dann existiert ein $\epsilon > 0$, so dass mit $E = B_\epsilon := \{\underline{x} \in \mathbb{R}^n : \|\underline{x} - \underline{x}^*\| \leq \epsilon\}$ die Voraussetzungen von Satz 6.10 erfüllt sind.

²³Bei der Vertauschung von Φ mit \lim wird ausgenutzt, dass die Kontraktionseigenschaft von Φ Stetigkeit impliziert.

Beweis. Es genügt, die Kontraktivität von Φ zu zeigen. Wegen Stetigkeit von Φ' gibt es ein $\epsilon > 0$, so dass $\|\Phi'(\xi)\| \leq L < 1$ für alle $\xi \in B_\epsilon$. Für $\underline{x}, \underline{y} \in B_\epsilon$ gilt

$$\|\Phi(\underline{x}) - \Phi(\underline{y})\| = \left\| \int_0^1 \Phi'(y + t(\underline{x} - \underline{y}))(\underline{x} - \underline{y}) dt \right\| \leq L \|\underline{x} - \underline{y}\|.$$

□

Beispiel 6.12 Betrachte das Gleichungssystem

$$\begin{aligned} x_1 - c(\cos x_1 - \sin x_2) &= 0, \\ (x_1 - x_2) + c \sin x_2 &= 0. \end{aligned}$$

Eine konsistente Fixpunktgleichung ist

$$\begin{aligned} c(\cos x_1 - \sin x_2) &= x_1 \\ c(\cos x_1 - 2 \sin x_2) &= x_2, \end{aligned}$$

mit dem Differential

$$\Phi' \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = -c \begin{pmatrix} \sin x_1 & \cos x_2 \\ \sin x_1 & 2 \cos x_2 \end{pmatrix}.$$

Für $c < 1/3$ gilt $\|\Phi'(\underline{x})\|_\infty < 1$ für alle $\underline{x} \in \mathbb{R}^2$ und damit konvergiert die Fixpunktiteration linear für alle Startwerte. Z.B. für $c = 1/4$:

k	$x_1^{(k)}$	$x_2^{(k)}$	$e^{(k)} = \ x^{(k)} - x^*\ _\infty$	$e^{(k)}/e^{(k-1)}$
0	0	0	0.2041	-
1	0.2500	0.2422	0.0788	0.3859
2	0.1823	0.1259	0.0375	0.4763
3	0.2145	0.1815	0.0180	0.4804
4	0.1992	0.1548	0.0086	0.4788
5	0.2065	0.1676	0.0041	0.4796
6	0.2030	0.1615	0.0020	0.4792
7	0.2047	0.1644	0.0010	0.4794
8	0.2039	0.1630	0.0005	0.4793
9	0.2043	0.1637	0.0002	0.4794

◇

Bemerkung 6.13 Gilt $\Phi'(\underline{x}^*) = 0$, so kann man ähnlich wie in Satz 6.9 lokal quadratische Konvergenz zeigen.

6.4 Nullstellen von Funktionen im Eindimensionalen

In diesem Abschnitt werden wir Klassen von Iterationsverfahren zur Bestimmung der Nullstelle(n) einer skalaren Funktion f behandeln.

6.4.1 Bisektionsverfahren

Die Bisektion ist eines der einfachsten (und robustesten) Verfahren und eignet sich für stetiges $f : \mathbb{R} \rightarrow \mathbb{R}$. Das unterliegende mathematische Resultat ist in der folgenden Proposition enthalten.

Proposition 6.14 Sei $f \in C^0([a, b])$ mit $f(a)f(b) < 0$. Dann existiert $x^* \in (a, b)$ mit $f(x^*) = 0$.

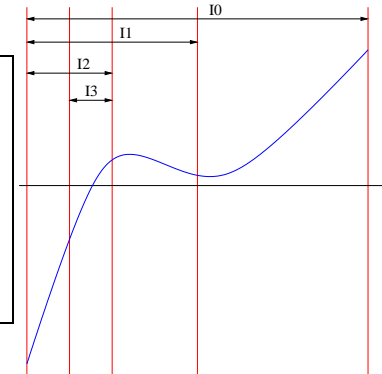
Beweis. Zwischenwertsatz. □

Ausgehend von einem Intervall $I_0 = [a^{(0)}, b^{(0)}]$ mit $f(a^{(0)})f(b^{(0)}) < 0$, erzeugt die **Bisektion** eine Folge $\{I_k\}_{k=0}^\infty$ von Intervallen $I_k = [a^{(k)}, b^{(k)}] \subset I_{k-1}$ mit $|I_k| = (b^{(0)} - a^{(0)})/2^k \rightarrow 0$, die eine Nullstelle x^* von f enthalten.

Algorithmus 6.15 (Bisektion)

```

for  $k = 0, 1, \dots$  do
   $x^{(k)} = \frac{1}{2}(a^{(k)} + b^{(k)})$ 
  if  $f(a^{(k)})f(x^{(k)}) \leq 0$  then
     $a^{(k+1)} = a^{(k)}, b^{(k+1)} = x^{(k)}$ 
  else
     $a^{(k+1)} = x^{(k)}, b^{(k+1)} = b^{(k)}$ 
  end if
end for
    
```



Bemerkung 6.16 Der Abstand zwischen den Intervallmitten $x^{(k)}$ und der Nullstelle x^* nimmt bei der Bisektion nicht notwendig monoton ab. Für den Fehler $e^{(k)} = |x^{(k)} - x^*|$ gilt im Allgemeinen nicht $e^{(k+1)} \leq L_k e^{(k)}$ mit $L_k < 1$.

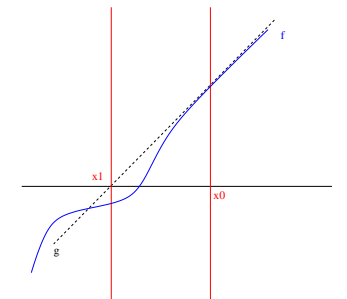
Die Bisektion ist sehr robust, konvergiert aber nur linear und besitzt keine sinnvolle Erweiterung auf mehrdimensionale Probleme. In der Praxis setzt man die Bisektion oft dazu ein, um einen Startwert genügend nahe bei x^* zu erhalten, von dem ein lokal schneller konvergentes Verfahren gestartet wird.

6.4.2 Newton-Verfahren

Das Newton-Verfahren ist ein (lokal) schneller konvergentes Verfahren als die Bisektion, setzt aber voraus, dass (i) f differenzierbar ist, (ii) $f'(x^*) \neq 0$ (keine doppelte Nullstelle). Wir geben 3 Herleitungen für das Newton-Verfahren an.

1. Graphisch Ausgehend von einem Startwert $x^{(0)} \in \mathbb{R}$, wird die Funktion f durch die Tangente g an $(x^{(0)}, f(x^{(0)}))$ approximiert. Die Nullstelle der Tangente bestimmt die nächste Iterierte $x^{(1)}$. Die Gerade g hat die Steigung $f'(x^{(0)})$ und geht durch den Punkt $(x^{(0)}, f(x^{(0)}))$. Damit ist g eindeutig bestimmt:

$$g(x) = f'(x^{(0)})(x - x^{(0)}) + f(x^{(0)}).$$



Die Nullstelle von g ist also

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})}$$

Wiederholung der beschriebenen Prozedur ergibt die **Newton-Iteration**

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \quad (6.8)$$

2. Fixpunktgleichung Unter der getroffenen Voraussetzung $f'(x^*) \neq 0$ ist x^* genau dann eine Nullstelle von f , wenn die Fixpunktgleichung

$$x = \Phi(x) := x - \frac{f(x)}{f'(x)}$$

erfüllt ist. Die zugehörige Fixpunktiteration hat die Form

$$x^{(k+1)} = \Phi(x^{(k)}) = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

Dabei gilt

$$\Phi'(x^*) = 1 - \frac{f'(x^*)f'(x^*) - f(x^*)f''(x^*)}{(f'(x^*))^2} = 0;$$

zusätzlich wird hier vorausgesetzt, dass f zwei Mal stetig differenzierbar ist. Nach Satz 6.9 konvergiert die Fixpunktiteration also lokal *quadratisch*.

3. Taylorentwicklung Betrachte für einen gegebenen Startwert $x^{(0)}$ die Taylorentwicklung um $x^{(0)}$:

$$f(x) = f(x^{(0)}) + (x - x^{(0)})f'(x^{(0)}) + \frac{1}{2}(x - x^{(0)})^2 f''(\xi_k). \quad (6.9)$$

für ein ξ_k zwischen $x^{(0)}$ und x . Ignorieren wir den quadratischen Term so gilt

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})f'(x^{(0)}).$$

Also folgt

$$x^* \approx x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})} =: x^{(1)}.$$

Wiederholtes Anwenden dieser Prozedur ergibt (wieder einmal) die Iteration

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}.$$

Aus der Taylorentwicklung (6.9) um $x^{(k)}$ und Einsetzen von $x = x^*$ erhält man die Fehlerdarstellung

$$0 = f(x^*) = f(x^{(k)}) + (x^* - x^{(k)})f'(x^{(k)}) + \frac{1}{2}(x^* - x^{(k)})^2 f''(\xi_k)$$

und damit

$$-\frac{f(x^{(k)})}{f'(x^{(k)})} + x^{(k)} - x^* = \frac{1}{2}(x^* - x^{(k)})^2 \frac{f''(\xi_k)}{f'(x^{(k)})}.$$

Wegen der Definition von $x^{(k+1)}$ gilt also

$$x^{(k+1)} - x^* = \frac{1}{2}(x^* - x^{(k)})^2 \frac{f''(\xi_k)}{f'(x^{(k)})}.$$

Hier zeigt sich auch wieder die lokale quadratische Konvergenz.

Beispiel 6.17 Die Funktion $f(x) = x^6 - x - 1$ hat eine Nullstelle x^* im Intervall $[0, 2]$, die berechnet werden soll. Das Newton-Verfahren (6.8) ergibt in diesem Fall

$$x^{(k+1)} = x^{(k)} - \frac{(x^{(k)})^6 - x^{(k)} - 1}{6(x^{(k)})^5 - 1}.$$

Dabei hängt die globale Konvergenz stark von der Wahl des richtigen Startwertes ab. Zum Beispiel konvergiert die Iteration für $x^{(0)} = 0.5$ gegen die "falsche" Nullstelle $-0.778\dots$, während sie für $x^{(0)} = 2$ gegen die gewünschte Nullstelle $1.1347\dots$ konvergiert. \diamond

6.4.3 Sekantenverfahren

Ein grosser Nachteil der Newton-Iteration ist die Notwendigkeit f' zu kennen. In Anwendungen ist f oft sehr komplex. Häufig hat man gar keine explizite Darstellung für f zur Verfügung; z.B. wenn f nur implizit über eine MATLAB-Funktion gegeben ist. In diesen Fällen ist es ein nahezu hoffnungsloses Unterfangen, eine analytische Formel von f' zu finden. Eine Alternative bietet das automatische Differenzieren (siehe Wikipedia) von Programmcode; dies ist aber nicht ganz problemlos, üblicherweise bläht sich der Code dadurch stark auf. Aus numerischer Sicht wäre es naheliegend, die Ableitung $f'(x^{(k)})$ durch die **finite Differenz** zu ersetzen:

$$f'(x^{(k)}) \approx f[x^{(k-1)}, x^{(k)}] = \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.$$

Eingesetzt in die Newton-Iteration (6.8) ergibt sich das **Sekantenverfahren**:

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} \quad (6.10)$$

Das Sekanten-Verfahren ist das prominenteste Beispiel einer Iteration die nicht nur von der letzten Iterierten sondern auch von älteren (hier der vorletzten) Iterierten abhängt. Dies kompliziert die Konvergenzanalyse erheblich.

Satz 6.18 Sei f zwei Mal stetig differenzierbar, $f(x^*) = 0$ und $f'(x^*) \neq 0$. Dann konvergiert das Sekantenverfahren (6.10) lokal gegen x^* mit der Ordnung (des goldenen Schnitts)

$$p = (1 + \sqrt{5})/2 \approx 1.62\dots$$

Beweis. Wir skizzieren die Hauptidee des Beweises, ohne alle technischen Details auszuarbeiten: Es gilt, falls $x^{(k)} \approx x^*$ und $x^{(k-1)} \approx x^*$, und falls $f(x^{(k)}) \neq f(x^{(k-1)})$,

$$\begin{aligned} x^{(k+1)} - x^* &= x^{(k)} - x^* - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})} \\ &= (x^{(k)} - x^*) \frac{f[x^{(k-1)}, x^{(k)}] - f[x^{(k)}, x^*]}{f[x^{(k-1)}, x^{(k)}]} \\ &= (x^{(k)} - x^*)(x^{(k-1)} - x^*) \frac{f[x^{(k-1)}, x^{(k)}, x^*]}{f[x^{(k-1)}, x^{(k)}]}, \end{aligned}$$

wobei die Ausdrücke $f[a, b]$ und $f[a, b, x^*]$ definiert sind als

$$f[a, b] = \frac{f(b) - f(a)}{b - a}, \quad f[a, b, x^*] = \frac{f[a, b] - f[b, x^*]}{a - x^*}$$

Da f zwei Mal stetig differenzierbar, gibt es Punkte

$$\begin{aligned} \xi^{(k)} &\in [\min(\{x^{(k-1)}, x^{(k)}\}), \max(\{x^{(k-1)}, x^{(k)}\})], \\ \eta^{(k)} &\in [\min(\{x^{(k-1)}, x^{(k)}, x^*\}), \max(\{x^{(k-1)}, x^{(k)}, x^*\})], \end{aligned}$$

so dass gilt

$$f[x^{(k-1)}, x^{(k)}] = f'(\xi^{(k)}) \quad f[x^{(k-1)}, x^{(k)}, x^*] = \frac{1}{2} f''(\eta^{(k)}).$$

Daraus folgt, dass

$$x^{(k+1)} - x^* = (x^{(k)} - x^*)(x^{(k-1)} - x^*) \frac{f''(\eta^{(k)})}{2 f'(\xi^{(k)})}, \quad k \geq 1.$$

Nehmen wir an, dass alle Werte $x^{(k)}$ in einem gewissen Intervall liegen und dass es ein $M \geq 0$ gibt, so dass in diesem Intervall gilt

$$\left| \frac{f''(\eta)}{2 f'(\xi)} \right| \leq M.$$

Durch setzen von $e^{(k)} := M|x^{(k)} - x^*|$ erhalten wir die Ungleichungen

$$e^{(k+1)} \leq e^{(k)} e^{(k-1)} \quad k = 1, 2, \dots$$

Definieren wir nun $\delta = \max(e^{(0)}, e^{(1)})$, so folgt direkt

$$\begin{aligned} e^{(2)} &\leq \delta^2 \\ e^{(3)} &\leq \delta^3 \\ e^{(4)} &\leq \delta^5 \\ &\vdots \\ e^{(k)} &\leq \delta^{m_k}, \end{aligned}$$

wobei die Zahlen m_k die Fibonacci-Folge bilden. Man kann zeigen, dass

$$m_k = \frac{1}{\sqrt{5}}(r_+^{k+1} - r_-^{k+1}), \quad \text{mit } r_{\pm} = \frac{1 \pm \sqrt{5}}{2}.$$

Für grosse k gilt also (da $|r_-| < 1$), $m_k \approx \frac{1}{\sqrt{5}}(r_+)^{k+1} \approx 0.447(1.618)^{k+1}$. \square

Auflösung der Fibonacci-Folge*

Die **Fibonacci-Folge** im Beweis von Satz 6.18 hat die Form

$$m_{k+2} = m_{k+1} + m_k, \quad m_0 = 1, \quad m_1 = 1. \tag{6.11}$$

Rekursionen dieser Form können wie folgt aufgelöst werden. (Der im Folgenden beschriebene Weg ist nicht der schnellste, kommt aber ohne "Überraschungsmomente" aus.) Zunächst schreibt man (6.11) in eine äquivalente zweidimensionale Rekursion um, die nur von der letzten Iterierten abhängt:

$$\begin{pmatrix} m_{k+1} \\ m_{k+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} m_k \\ m_{k+1} \end{pmatrix}.$$

Setzen wir $\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ und $\underline{y}_k = \begin{pmatrix} m_k \\ m_{k+1} \end{pmatrix}$, so erhalten wir die Beziehung

$$\underline{y}_{k+1} = \mathbf{A}\underline{y}_k, \quad \underline{y}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{6.12}$$

Insbesondere folgt also $\underline{y}_k = \mathbf{A}^k \underline{y}_0$. Wir geben nun das Vorgehen für allgemeine $\mathbf{A} \in \mathbb{R}^{n \times n}$ an. Die Rekursionsformel vereinfacht sich wesentlich, wenn wir \mathbf{A} diagonalisieren können, d.h., es existiert eine invertierbare Matrix \mathbf{P} , so dass

$$\mathbf{P}^{-1} \mathbf{A} \mathbf{P} = \text{diag}(\lambda_1, \dots, \lambda_n),$$

mit den Eigenwerten $\lambda_1, \dots, \lambda_n$ von \mathbf{A} . (Bemerkung: Für nichtdiagonalisierbare Matrizen verwendet man die Jordan'sche Normalform.) Durch Einfügen "produktiver Identitäten" folgt

$$\begin{aligned} \underline{y}_k &= \mathbf{P} \mathbf{P}^{-1} (\mathbf{A})^k \mathbf{P} \mathbf{P}^{-1} \underline{y}_0 = \mathbf{P} (\mathbf{P}^{-1} \mathbf{A} \mathbf{P})^k \mathbf{P}^{-1} \underline{y}_0 \\ &= \mathbf{P} \text{diag}(\lambda_1^k, \dots, \lambda_n^k) \mathbf{P}^{-1} \underline{y}_0. \end{aligned}$$

Für die Fibonacci-Folge ist \mathbf{A} symmetrisch, damit können wir \mathbf{P} sogar orthogonal wählen! Durch Bestimmung der Eigenwerte und Eigenvektoren von \mathbf{A} erhält man

$$\lambda_1 = \frac{1 - \sqrt{5}}{2}, \quad \lambda_2 = \frac{1 + \sqrt{5}}{2}, \quad \mathbf{P} = \frac{1}{\sqrt{1 + \lambda_2^2}} \begin{pmatrix} \lambda_2 & 1 \\ -1 & \lambda_2 \end{pmatrix}.$$

Also

$$\underline{y}_k = \frac{1}{1 + \lambda_2^2} \begin{pmatrix} \lambda_2 & 1 \\ -1 & \lambda_2 \end{pmatrix} \begin{pmatrix} \lambda_2^k & 0 \\ 0 & \lambda_2^k \end{pmatrix} \begin{pmatrix} \lambda_2 & -1 \\ 1 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Für den ersten Eintrag von \underline{y}_k ergibt sich

$$m_k = \frac{-\lambda_2 \lambda_1^{k+1} + \lambda_2^k (1 + \lambda_2)}{1 + \lambda_2^2} = \frac{-\lambda_2 \lambda_1^{k+1} + \lambda_2^k (1 + \lambda_2)}{\sqrt{5} \lambda_2} = \frac{\lambda_2^{k+1} - \lambda_1^{k+1}}{\sqrt{5}}.$$

6.4.4 Kondition von Nullstellen

Bisektions-, Newton- und Sekanten-Verfahren sind *nicht* anwendbar, wenn $f'(x^*) = 0$ gilt. Dies ist kein gravierender Nachteil; das folgende Resultat zeigt, dass die sinnvolle *numerische* Berechnung einer mehrfachen Nullstelle (also unter dem Einfluss von Fehlern) sowieso höchst zweifelhaft ist. Dazu betrachten wir die Kondition einer Nullstelle x^* von $f : \mathbb{R} \rightarrow \mathbb{R}$ bezüglich Störungen von f . Dabei ist es sinnvoll anzunehmen, dass die Störung in der Nähe von x^* durch ein (kleines) ϵ beschränkt ist, d.h.,

$$|\tilde{f}(x) - f(x)| \leq \epsilon. \tag{6.13}$$

Ist ϵ genügend klein, so gibt es eine Nullstelle $\tilde{x}^* \approx x^*$ von \tilde{f} , also $\tilde{f}(\tilde{x}^*) = 0$. Aus (6.13) ergibt sich

$$|f(\tilde{x}^*)| \leq \epsilon. \tag{6.14}$$

Sei m die **Vielfachheit** der Nullstelle x^* , d.h.:

$$f(x^*) = 0, \quad f'(x^*) = 0, \quad f''(x^*) = 0, \quad \dots, \quad f^{(m-1)}(x^*) = 0, \quad f^{(m)}(x^*) \neq 0.$$

Taylor-Entwicklung ergibt

$$\begin{aligned} f(\tilde{x}^*) &= f(x^*) + (\tilde{x}^* - x^*)f'(x^*) + \dots + \frac{(\tilde{x}^* - x^*)^m}{m!}f^{(m)}(x^*) + O((\tilde{x}^* - x^*)^{m+1}) \\ &= \frac{(\tilde{x}^* - x^*)^m}{m!}f^{(m)}(x^*) + O((\tilde{x}^* - x^*)^{m+1}). \end{aligned}$$

- Für $m = 1$ ergibt sich als **absolute Konditionszahl** einer *einfachen* Nullstelle x^*

$$\lim_{\epsilon \rightarrow 0} \frac{|\tilde{x}^* - x^*|}{\epsilon} = \frac{1}{|f'(x^*)|}.$$

- Für $m > 1$ folgt aber $\lim_{\epsilon \rightarrow 0} |\tilde{x}^* - x^*|/\epsilon \rightarrow \infty$. Also hat eine mehrfache Nullstelle x^* formal eine absolute Konditionszahl von ∞ ! Stattdessen gilt lediglich

$$\lim_{\epsilon \rightarrow 0} \frac{|\tilde{x}^* - x^*|}{\epsilon^{1/m}} = \left| \frac{m!}{f^{(m)}(x^*)} \right|^{1/m}.$$

6.5 Newton-Verfahren im Mehrdimensionalen

Wir betrachten jetzt die Lösung von $f(\underline{x}) = 0$, wobei $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ stetig differenzierbar ist.

6.5.1 Grundlagen

Sei $\underline{x}^{(0)} \in \mathbb{R}^n$ der Startwert. Die Taylor-Entwicklung von f um $\underline{x}^{(0)}$ ergibt für die i -te Komponente f_i von f die Beziehung:

$$f_i(\underline{x}) = f_i(\underline{x}^{(0)}) + \sum_{j=1}^n \frac{\partial f_i(\underline{x}^{(0)})}{\partial x_j} (x_j - x_j^{(0)}) + O(\|\underline{x} - \underline{x}^{(0)}\|_2^2).$$

Kompaktere Schreibweise mit dem Differential f' von f :

$$f(\underline{x}) = f(\underline{x}^{(0)}) + f'(\underline{x}^{(0)})(\underline{x} - \underline{x}^{(0)}) + O(\|\underline{x} - \underline{x}^{(0)}\|_2^2).$$

Die nächste Iterierte $\underline{x}^{(1)}$ ist die Nullstelle der linearen Funktion, die man durch Abschneiden des quadratischen Terms erhält:

$$0 = f(\underline{x}^{(0)}) + f'(\underline{x}^{(0)})(\underline{x}^{(1)} - \underline{x}^{(0)}).$$

Ist $f'(\underline{x}^{(0)})$ invertierbar, so folgt

$$\underline{x}^{(1)} = \underline{x}^{(0)} - [f'(\underline{x}^{(0)})]^{-1} f(\underline{x}^{(0)}).$$

Wiederholtes Anwenden dieser Prozedur ergibt schliesslich die **Newton-Iteration**

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - [f'(\underline{x}^{(k)})]^{-1} f(\underline{x}^{(k)}), \quad k = 0, 1, 2, \dots \quad (6.15)$$

Diese kann man wieder als Fixpunktiteration für die Fixpunktgleichung $\Phi(\underline{x}) = \underline{x} - [f'(\underline{x})]^{-1} f(\underline{x})$ auffassen und aus Bemerkung 6.13 folgt lokal quadratische Konvergenz (siehe auch Satz 6.20 unten).

Beispiel 6.19 (Dahmen/Reusken) Es sei die folgende Integralgleichung zu lösen. Gesucht ist eine Funktion $u: [0, 1] \rightarrow \mathbb{R}$, so dass

$$u(\xi) + \int_0^1 \cos(\xi t) u(t)^3 dt = 2, \quad \xi \in [0, 1]. \quad (6.16)$$

Das Integral wird nun mit der summierten Mittelpunktsregel auf dem Gitter

$$t_1 = (1 - \frac{1}{2})h, \quad t_2 = (2 - \frac{1}{2})h, \quad \dots, \quad t_n = (n - \frac{1}{2})h$$

mit $h = 1/n$ approximiert:

$$\int_0^1 \cos(\xi t) u(t)^3 dt \approx \sum_{j=1}^n \cos(\xi t_j) u(t_j)^3.$$

Betrachten wir (6.16) nur an den Stützstellen $\xi = t_i$ für $i = 1, \dots, n$ und setzen diese Approximation ein, so ergibt sich das nichtlineare Gleichungssystem

$$u_i + h \sum_{j=1}^n \cos(t_i t_j) u_j^3 - 2 = 0, \quad i = 1, \dots, n, \quad (6.17)$$

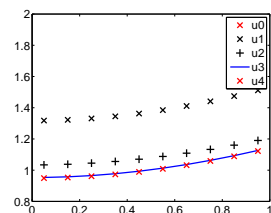
mit den Unbekannten $u_i \approx u(t_i)$. Mit

$$f(\underline{u}) := \begin{pmatrix} u_1 + h \sum_{j=1}^n \cos(t_1 t_j) u_j^3 - 2 \\ \vdots \\ u_n + h \sum_{j=1}^n \cos(t_n t_j) u_j^3 - 2 \end{pmatrix}$$

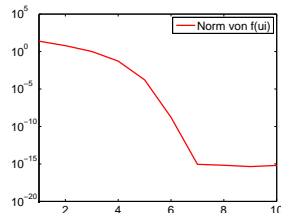
erhalten wir also die Gleichung $f(\underline{u}) = 0$. Die Einträge des Differentials $f'(\underline{u}) \in \mathbb{R}^{n \times n}$ sind gegeben durch

$$[f'(\underline{u})]_{i,j} = \frac{\partial f_i(\underline{u})}{\partial u_j} = \begin{cases} 1 + 3h \cos(t_i^2) u_i^2, & \text{für } i = j, \\ 3h \cos(t_i t_j) u_j^2, & \text{für } i \neq j. \end{cases}$$

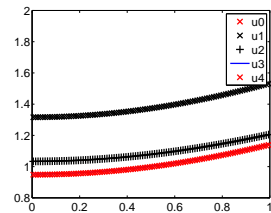
Wir wenden die Newton-Iteration mit den Startvektor $\underline{u}^{(0)} = [2, \dots, 2]^T$ an, um (6.17) zu lösen.



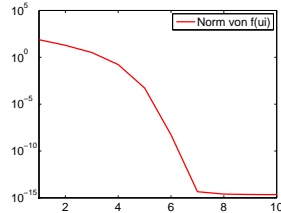
$n = 10$, Iterierte $\underline{u}^{(0)}, \dots, \underline{u}^{(4)}$



$n = 10$, $\|f(\underline{u}^{(i)})\|_2$



$n = 100$, Iterierte $\underline{u}^{(0)}, \dots, \underline{u}^{(4)}$



$n = 100$, $\|f(\underline{u}^{(i)})\|_2$

Es reichen also wenige Iterierte, um eine sehr genaue Approximation zu erreichen. (Vorsicht! Der Integrationsfehler der summierten Mittelpunktsregel wurde hier ausser Acht gelassen.) \diamond

6.5.2 Konvergenz und Abbruchkriterien

In Beispiel 6.19 ist deutlich die quadratische Konvergenz des Newton-Verfahrens auch im mehrdimensionalen Fall sichtbar. Der folgende Satz bestätigt diese Beobachtung.

Satz 6.20 Sei $f : D \rightarrow \mathbb{R}^n$ stetig differenzierbar, wobei $D \subseteq \mathbb{R}^n$ eine offene Menge ist, die ein \underline{x}^* mit $f(\underline{x}^*) = \underline{0}$ enthält. Angenommen $f'(\underline{x}^*)$ ist invertierbar und es gibt Konstanten L, R derart, dass

$$\|f'(\underline{x}) - f'(\underline{y})\| \leq L\|\underline{x} - \underline{y}\|, \quad \forall \underline{x}, \underline{y} \in B_R \quad (6.18)$$

für irgendeine Vektornorm $\|\cdot\|$ (mit der zugehörigen induzierten Matrixnorm) und $B_R = \{\underline{x} \in \mathbb{R}^n : \|\underline{x} - \underline{x}^*\| \leq R\}$. Setze $r := \min\{R, 1/(2CL)\}$ mit $C := \|[f'(\underline{x}^*)]^{-1}\|$. Dann ist für jeden Startwert $\underline{x}^{(0)} \in B_r$ die Newton-Iteration (6.15) wohlbestimmt und konvergiert gegen \underline{x}^* , wobei

$$\|\underline{x}^{(k+1)} - \underline{x}^*\| \leq CL\|\underline{x}^{(k)} - \underline{x}^*\|^2. \quad (6.19)$$

Beweis. Zunächst zeigen wir die Wohlbestimmtheit der ersten Iteration für jedes $\underline{x}^{(0)} \in B_r$, also dass $f'(\underline{x}_0)$ invertierbar ist. Es gilt

$$\|[f'(\underline{x}^*)]^{-1}\| \|f'(\underline{x}^{(0)}) - f'(\underline{x}^*)\| \leq CL\|\underline{x}^{(0)} - \underline{x}^*\| \leq CLr \leq \frac{1}{2}.$$

Proposition 2.20 (Inverse einer gestörten Matrix) angewandt auf

$$f'(\underline{x}^{(0)}) = \underbrace{f'(\underline{x}^*)}_{=: \mathbf{A}} + \underbrace{f'(\underline{x}^{(0)}) - f'(\underline{x}^*)}_{=: \Delta \mathbf{A}},$$

impliziert, dass $f'(\underline{x}^{(0)})$ invertierbar und damit $\underline{x}^{(1)}$ wohlbestimmt ist. Ausserdem gilt

$$\|[f'(\underline{x}^{(0)})]^{-1}\| \leq \frac{\|[f'(\underline{x}^*)]^{-1}\|}{1 - \|[f'(\underline{x}^*)]^{-1}\| \|f'(\underline{x}^{(0)}) - f'(\underline{x}^*)\|} \leq 2\|[f'(\underline{x}^*)]^{-1}\| \leq 2C.$$

Wegen

$$\underline{x}^{(1)} - \underline{x}^* = \underline{x}^{(0)} - \underline{x}^* - [f'(\underline{x}^{(0)})]^{-1}(f(\underline{x}^{(0)}) - f(\underline{x}^*))$$

erhalten wir

$$\|\underline{x}^{(1)} - \underline{x}^*\| \leq \|[f'(\underline{x}^{(0)})]^{-1}\| \|f(\underline{x}^{(0)}) - f(\underline{x}^*) - f'(\underline{x}^{(0)})(\underline{x}^{(0)} - \underline{x}^*)\|. \quad (6.20)$$

Um den zweiten Faktor auf der rechten Seite abzuschätzen, nutzen wir die Beziehung

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

Aus dieser folgt zusammen mit (6.18), dass

$$\begin{aligned} & \|f(\underline{x}^{(0)}) - f(\underline{x}^*) - f'(\underline{x}^{(0)})(\underline{x}^{(0)} - \underline{x}^*)\| \\ & \leq \int_0^1 \|f'(\underline{x}^{(0)} + t(\underline{x}^* - \underline{x}^{(0)})) - f'(\underline{x}^{(0)})\| \|\underline{x}^{(0)} - \underline{x}^*\| dt \\ & \leq \int_0^1 Lt \|\underline{x}^{(0)} - \underline{x}^*\|^2 dt = \frac{L}{2} \|\underline{x}^{(0)} - \underline{x}^*\|^2. \end{aligned}$$

Also ergibt sich aus (6.20) die Abschätzung

$$\|\underline{x}^{(1)} - \underline{x}^*\| \leq \frac{L}{2} \|[f'(\underline{x}^{(0)})]^{-1}\| \|\underline{x}^{(0)} - \underline{x}^*\|^2 \leq CL\|\underline{x}^{(0)} - \underline{x}^*\|^2 \quad (6.21)$$

und damit ist (6.19) gezeigt. Da $\|\underline{x}^{(0)} - \underline{x}^*\| \leq r$ und $r \leq 1/(CL)$, impliziert (6.21) die Beziehung $\|\underline{x}^{(1)} - \underline{x}^*\| \leq \frac{1}{2}\|\underline{x}^{(0)} - \underline{x}^*\|$ und damit ist $\underline{x}^{(1)} \in B_r$. Also lässt sich die obige Argumentation wiederholen und zeigen, dass $\underline{x}^{(2)}$ wohldefiniert und $\underline{x}^{(2)} \in B_r$, usw. Letztendlich ist jedes $\underline{x}^{(k)}$ wohldefiniert und es gilt $\underline{x}^{(k)} \in B_r$. Ausserdem verringert sich der Abstand zwischen \underline{x}^* und $\underline{x}^{(k)}$ in jeder Iteration um mindestens den Faktor 1/2 und damit konvergiert $\underline{x}^{(k)}$ gegen \underline{x}^* für $k \rightarrow \infty$. \square

Satz 6.20 ist einer der einfachsten Vertreter einer ganzen Klasse von Konvergenzresultaten für das Newton-Verfahren. Weitergehende Resultate stellen schwächere Forderungen an f und resultieren in bessere Abschätzungen für r , sind aber auch wesentlich technischer. Allen Resultaten ist gemein, dass die Konstante $C = \|[f'(\underline{x}^*)]^{-1}\|$, die der Kondition der nichtlinearen Gleichung $f(\underline{x}) = \underline{0}$ entspricht, eine entscheidende Rolle spielt. Je grösser C desto kleiner ist der nachweisbare Konvergenzbereich und desto langsamer ist die Konvergenz im prä-asymptotischen Bereich.

Die theoretischen Annahmen von Satz 6.20 lassen sich in Unkenntnis der Lösung \underline{x}^* nicht überprüfen. Es stellt sich die Frage, wie möglichst früh festgestellt werden kann, dass die Newton-Iteration die gewünschte Genauigkeit erreicht hat. Wenn die Newton-Iteration nicht konvergiert, sollte dies auch rechtzeitig feststellbar sein, um gegebenenfalls den Startwert anzupassen. Wir geben im folgenden zwei in der Praxis übliche Kriterien an.

Residuenschätzer Das **Residuum** in der k -ten Iterierten ist $f(\underline{x}^{(k)})$. Der Test $\|f(\underline{x}^{(k)})\| \leq \text{tol}$ zu einer vorgegebenen Benutzertoleranz $\text{tol} > 0$ ist eine einfache Möglichkeit das Verfahren auf Konvergenz zu testen. Um in der $(k+1)$ -ten Iteration festzustellen, ob das Verfahren überhaupt Fortschritte macht, bietet sich der folgende **Monotonietest** an:

$$\|f(\underline{x}^{(k+1)})\| \leq \theta \|f(\underline{x}^{(k)})\| \quad (6.22)$$

für ein $\theta < 1$. Dieser Test hat den subtil scheinenden aber in der Praxis überaus bedeutenden Nachteil, das er nicht affin invariant ist. **Affine Invarianz** ist die folgende Eigenschaft des Newton-Verfahrens: Die Iterierten $\underline{x}^{(k)}$ des Newton-Verfahrens ändern sich *nicht*, wenn die nichtlineare Gleichung mit einer invertierbaren Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ vormultipliziert wird: $\mathbf{A}f(\underline{x}) = \underline{0}$. Der Test (6.22) ändert sich allerdings unter einer solchen Transformation. Deswegen bevorzugt man den sogenannten **natürlichen Monotonietest** [Deuffhard]:

$$\|[f'(\underline{x}^{(k)})]^{-1}f(\underline{x}^{(k+1)})\| \leq \theta \|[f'(\underline{x}^{(k)})]^{-1}f(\underline{x}^{(k)})\| \quad (6.23)$$

für ein $\theta < 1$. Dieser ist affin invariant und wird meist mit $\theta = 1/2$ verwendet. Gilt

$$\|[f'(\underline{x}^{(k)})]^{-1}f(\underline{x}^{(k+1)})\| > \frac{1}{2} \|[f'(\underline{x}^{(k)})]^{-1}f(\underline{x}^{(k)})\|,$$

so wird die Iteration abgebrochen und ein Konvergenzfehler signalisiert.

MATLAB

```
function x = newtonnatuerlich(fun,jac,x0,tol)
% Newton-Verfahren mit natuerlichem Konvergenztest
x = x0; f = feval(fun,x);
while (1==1),
    J = feval(jac,x); [L,U] = lu(J);
    x = x - U \ ( L \ f );
    fold = f; f = feval(fun,x);
    resold = norm( U \ ( L \ fold ) );
    resnew = norm( U \ ( L \ f ) );
    if resnew <= tol, return; end
    if resnew / resold > 1/2,
        warning('Newton-Verfahren konvergierte nicht ...
                zur vorgegebenen Genauigkeit. '); return
    end
end
```

Vorwärtsfehlerschätzer In vielen Implementierungen findet sich statt (6.22) bzw. (6.23) die folgende Heuristik: *Ändern sich die Iterierten $\underline{x}^{(k)}$ nur noch geringfügig,*

so wird die Newton-Iteration abgebrochen und Konvergenz deklariert. Die mathematische Begründung für dieses Vorgehen leitet sich aus der Konvergenzschranke (6.19) ab. Für $k \rightarrow \infty$ gelten

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k)} - x^*\|} \leq \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} + 1 \leq CL\|x^{(k)} - x^*\| + 1 \rightarrow 1$$

und

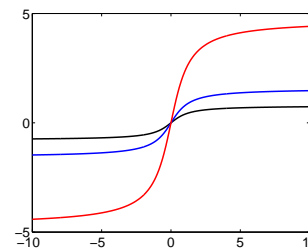
$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k)} - x^*\|} \geq 1 - \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \geq 1 - CL\|x^{(k)} - x^*\| \rightarrow 1.$$

Für grosse k folgt also $\|x^{(k+1)} - x^{(k)}\| \approx \|x^{(k)} - x^*\|$.

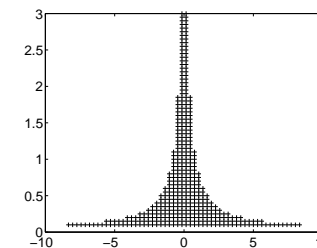
6.5.3 Globalisierungstechniken

Der mit Abstand grösste Nachteil des Newton-Verfahrens ist das Versagen der Konvergenz, wenn $\underline{x}^{(0)}$ nicht genügend nahe bei \underline{x}^* ist.

Beispiel 6.21 Sei $f(x) = \arctan(ax)$ für ein $a > 0$. Die einzige Nullstelle ist $x^* = 0$. Je grösser a desto steiler geht die Funktion durch a und desto kleiner wird der Konvergenzbereich.



$\arctan(ax)$ für $a = 0.1, 1, 3$



x -Achse: Bereich der $x^{(0)}$, für die Newton konvergiert. y -Achse: a .

◇

Das Problem in Beispiel 6.21 ist ein "Überschessen" des Korrekturterms

$$-(f'(\underline{x}^{(k)}))^{-1}f(\underline{x}^{(k)}).$$

Es liegt nahe dies durch eine Dämpfung der Korrektur zu vermeiden. Für einen Dämpfungsfaktor $0 < \lambda_k \leq 1$ hat das **gedämpfte Newton-Verfahren** die Form

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \lambda_k (f'(\underline{x}^{(k)}))^{-1} f(\underline{x}^{(k)}). \quad (6.24)$$

Für die Wahl von λ_k gibt es verschiedene Möglichkeiten. Eine recht praktikable Strategie besteht darin λ_k so zu wählen, dass der natürliche Monotonietest (6.23) für $\theta = 1 - \lambda_k/2$ erfüllt ist, d.h., es gelte

$$\|[f'(\underline{x}^{(k)})]^{-1}f(\underline{x}^{(k+1)})\| \leq (1 - \lambda_k/2) \|[f'(\underline{x}^{(k)})]^{-1}f(\underline{x}^{(k)})\|.$$

Es ist zu beachten, dass nun $\underline{x}^{(k+1)}$ von λ_k abhängig ist! Üblicherweise wählt man λ_k aus einer Folge

$$\left\{ 1, \frac{1}{2}, \frac{1}{4}, \dots, \lambda_{\min} \right\}.$$

Falls $\lambda_k < \lambda_{\min}$, wird die Iteration abgebrochen. Ist λ_k erfolgreich, so versuchen wir im nächsten Iterationsschritt $\lambda_{k+1} = \min\{1, 2\lambda_k\}$. Ist der natürliche Monotonietest verletzt, so versuchen wir es mit $\lambda_k \leftarrow \lambda_k/2$.

Beispiel 6.22 Für $f(x) = \arctan(x)$ wählen wir als Startwert $x^{(0)} = 20$. Das Standard-Newton-Verfahren konvergiert nicht (siehe Beispiel 6.21). Der Verlauf des gedämpften Newton-Verfahrens (mit $\lambda_{\min} = 0.001$):

k	λ_k	$x^{(k)}$	$f(x^{(k)})$
1	0.03125	0.94199967624205	0.75554074974604
2	0.06250	0.85287592931991	0.70616132170387
3	0.12500	0.70039827977515	0.61099321623952
4	0.25000	0.47271811131169	0.44158487422833
5	0.50000	0.20258686348037	0.19988168667351
6	1.00000	-0.00549825489514	-0.00549819949059
7	1.00000	0.00000011081045	0.00000011081045
8	1.00000	-0.00000000000001	-0.00000000000001

Nach anfänglicher Dämpfung und augenscheinlich linearer Konvergenz erreicht das Verfahren den lokalen (quadratischen) Konvergenzbereich. \diamond

6.5.4 Quasi-Newton-Verfahren*

Die Aufstellung und Inversion des Differentials $f'(\underline{x}^{(k)})$ kann bei einigen Anwendungen so teuer werden, dass die Verwendung des Newton-Verfahrens unpraktikabel wird. Es gibt verschiedene Strategien $[f'(\underline{x}^{(k)})]^{-1}$ weitestgehend zu vermeiden, typischerweise aber auf Kosten der Konvergenz. Wir wollen exemplarisch nur eine Möglichkeit behandeln: das **Quasi-Newton-Verfahren von Broyden**.

Die Idee ist, eine sinnvolle Verallgemeinerung des Sekantenverfahrens für $n > 1$ zu finden. Analog zur finiten Differenz im Sekantenverfahren suchen wir eine $n \times n$ Matrix \mathbf{J}_k , so dass

$$\mathbf{J}_k(\underline{x}^{(k)} - \underline{x}^{(k-1)}) = f(\underline{x}^{(k)}) - f(\underline{x}^{(k-1)}) \quad (6.25)$$

gilt. Dies sind nur n Gleichungen für die n^2 Einträge von \mathbf{J}_k ; es gibt also viele solche \mathbf{J}_k . Als weitere Nebenbedingung stellen wir, dass \mathbf{J}_k sich nicht sehr von der vorhergehenden Matrix \mathbf{J}_{k-1} unterscheiden soll, wir fordern

$$\mathbf{J}_k \underline{z} = \mathbf{J}_{k-1} \underline{z}, \quad \forall \underline{z} \in \mathbb{R}^n : \underline{z} \perp (\underline{x}^{(k)} - \underline{x}^{(k-1)}). \quad (6.26)$$

Lemma 6.23 Die Matrix

$$\mathbf{J}_k = \mathbf{J}_{k-1} + \frac{1}{\|\underline{x}^{(k)} - \underline{x}^{(k-1)}\|_2^2} f(x^{(k)}) (\underline{x}^{(k)} - \underline{x}^{(k-1)})^\top$$

erfüllt (6.25) und (6.26).

Damit ergibt sich das Verfahren

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \Delta_k, \quad \Delta_k = -\mathbf{J}_k^{-1} f(\underline{x}^{(k)}), \quad \mathbf{J}_{k+1} = \mathbf{J}_k + \frac{1}{\|\Delta_k\|_2^2} f(x^{(k+1)}) \Delta_k^\top.$$

Da \mathbf{J}_k sich von \mathbf{J}_{k-1} nur um eine Differenz vom Rang 1 unterscheidet, können wir die Sherman-Morrison-Formel (3.44) verwenden, um \mathbf{J}_k effizient zu invertieren. Als Startwert für \mathbf{J}_0 wählt man zum Beispiel $\mathbf{J}_0 = f'(\underline{x}^{(0)})$.

Kapitel 7

Ausgleichsrechnung

Bisher untersuchten wir Verfahren für die Lösung linearer Gleichungssysteme $\mathbf{Ax} = \mathbf{b}$ mit quadratischer Matrix \mathbf{A} . In vielen Anwendungen tritt das Problem auf, lineare Gleichungssysteme

$$\mathbf{Ax} = \mathbf{b} \quad (7.1)$$

zu “lösen”, wobei

$$\mathbf{A} \in \mathbb{C}^{m \times n}, \quad \mathbf{x} \in \mathbb{C}^n, \quad \mathbf{b} \in \mathbb{C}^m, \quad (7.2)$$

d.h., \mathbf{A} ist eine rechteckige Matrix. Einfache Beispiele zeigen, dass (7.1) entweder genau eine, keine oder unendliche viele Lösungen besitzen kann. Es ist daher zuerst nach einem Lösungsbegriff für (7.1) zu fragen oder: Was bedeutet es, (7.1) zu “lösen”?

7.1 Motivation: Gauss'sche Methode der kleinsten Quadrate

Bei Ausgleichsproblemen handelt es sich um das Problem, m Messdaten (t_i, y_i) , $i = 1, \dots, m$, mit einem mathematischen Gesetz eines vorgegebenen Typs in Einklang zu bringen. Dabei ist die Form des mathematischen Gesetzes bis auf einige Parameter x_j , $j = 1, \dots, n$, festgelegt, welche durch die Messungen bestimmt werden sollen:

$$y(t) = f(t) := \sum_{j=1}^n x_j \varphi_j(t) \quad (7.3)$$

mit vorgegebenen Funktionen $\varphi_j(t)$. Typischerweise tritt dann das “Problem” auf, dass mehr Messungen gemacht wurden als unbekannte Parameter vorliegen, d.h.

$$m > n,$$

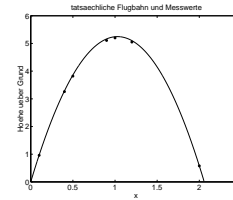
und die Messungen wegen *Messfehlern* nicht exakt dem vorgegebenen physikalischen Gesetz entsprechen. Man erhält damit ein *überbestimmtes* Gleichungssystem, das nicht exakt gelöst werden kann.

Im Folgenden wird die Gauss'sche Methode der kleinsten Fehlerquadrate vorgestellt. Diese ist anwendbar bei Problemen, bei denen die zu bestimmenden Parameter x_j linear in das mathematische Gesetz $f(t)$, wie in (7.3), eingehen.

Beispiel 7.1 Unter Einfluss der Schwerkraft fliegen geworfene Körper auf Parabeln. Hat der Körper die Anfangsgeschwindigkeit $v = (v_x, v_y)$ zum Zeitpunkt $t = 0$ am Punkt $(0, 0)$ und fliegt er anschliessend nur unter Einfluss der Schwerkraft, so ist er zum Zeitpunkt $t > 0$ am Ort

$$x = v_x t, \quad y = v_y t - \frac{1}{2} g t^2, \quad \text{wobei } g \text{ die Erdbeschleunigung ist.}$$

Es sei die Anfangsgeschwindigkeit v_y und die Erdbeschleunigung g unbekannt; diese sollen aus Messungen bestimmt werden. Hierzu wurde die Höhe über Grund des Körpers zu folgenden Zeiten gemessen: Es ergeben sich damit $m = 7$ Gleichungen



i	1	2	3	4	5	6	7
t_i [s]	0.1	0.4	0.5	0.9	1.0	1.2	2.0
y_i [m]	0.96	3.26	3.82	5.11	5.2	5.05	0.58

Abbildung 7.1. Flugbahn eines Körpers und gemessene Positionen.

für die $n = 2$ unbekannt Parameter v_y und g :

$$y_i = t_i v_y - \frac{1}{2} t_i^2 g, \quad i = 1, \dots, 7.$$

Führt man die Matrix $\mathbf{A} \in \mathbb{R}^{7 \times 2}$ mit

$$a_{i1} = \varphi_1(t_i) = t_i, \quad a_{i2} = \varphi_2(t_i) = -\frac{1}{2} t_i^2, \quad i = 1, \dots, 7,$$

ein, so ergibt sich

$$\mathbf{A} \begin{pmatrix} v_y \\ g \end{pmatrix} = \mathbf{y},$$

d.h.

$$\begin{pmatrix} 0.1 & -0.005 \\ 0.4 & -0.08 \\ 0.5 & -0.125 \\ 0.9 & -0.405 \\ 1.0 & -0.5 \\ 1.2 & -0.72 \\ 2.0 & -2.0 \end{pmatrix} \cdot \begin{pmatrix} v_y \\ g \end{pmatrix} = \begin{pmatrix} 0.96 \\ 3.26 \\ 3.82 \\ 5.11 \\ 5.2 \\ 5.05 \\ 0.58 \end{pmatrix} \quad (7.4)$$

◇

In Beispiel 7.1 erhielten wir ein überbestimmtes Gleichungssystem, das keine klassische Lösung hat. Die Ausgleichsrechnung liefert nun eine Methode, sinnvolle Lösungen von überbestimmten Gleichungssystemen zu definieren.

Für $m \geq n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ werde eine “Lösung” $\mathbf{x} \in \mathbb{R}^n$ des folgenden überbestimmten Gleichungssystems gesucht:

$$\sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1, \dots, m. \quad (7.5)$$

Da wir für $m > n$ diese m Gleichungen i.a. nicht alle exakt erfüllen können, führen wir die sogenannten *Residuen* r_i ein, die messen, inwieweit die i -te Gleichung verletzt ist:

$$r_i := b_i - \sum_{j=1}^n a_{ij}x_j, \quad i = 1, \dots, m.$$

In Matrixschreibweise ist dies

$$\underline{r} = \underline{b} - \mathbf{A}\underline{x}, \quad (7.6)$$

wobei der Residuumsvektor $\underline{r} \in \mathbb{R}^m$ die Komponenten r_i hat. Wir suchen nach Vektoren $\underline{x} \in \mathbb{R}^n$, für die das Residuum \underline{r} möglichst "klein" (in einer zu wählenden Vektornorm!) ist.

Ein $\underline{x} \in \mathbb{R}^n$, das das Residuum minimiert, wird **Ausgleichslösung** von (7.1) genannt. Gibt es mehrere solche \underline{x} so suchen wir dasjenige mit der kleinsten Norm (die Minimallösung).

Definition 7.2 (Lineares Ausgleichsproblem) Sei $\|\cdot\|$ eine Norm auf \mathbb{R}^m . Für eine Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ und ein $\underline{b} \in \mathbb{R}^m$ heisst das Problem

$$\text{Finde } \underline{x} \in \mathbb{R}^n \text{ mit } \|\underline{x}\| = \min! \text{ so, dass } \|\mathbf{A}\underline{x} - \underline{b}\| \leq \|\mathbf{A}\underline{y} - \underline{b}\| \quad \forall \underline{y} \in \mathbb{R}^n, \quad (7.7)$$

Ausgleichsproblem.

Die Wahl der Norm in Definition 7.2 ist entscheidend für (a) die praktische Relevanz von \underline{x} ; und (b) die numerische Berechenbarkeit von \underline{x} . Insbesondere der letztere Aspekt führt uns dazu im folgenden immer die Euklidische Norm $\|\cdot\|_2$ zu wählen.

7.2 Normalgleichungen

In diesem und dem folgenden Abschnitt wollen wir annehmen, dass $\mathbf{A} \in \mathbb{R}^{m \times n}$ vollen Spaltenrang hat, d.h.

$$m \geq n, \quad \text{rang}(\mathbf{A}) = n.$$

Wir wenden uns nun der Frage zu, wie eine Lösung $\underline{x} \in \mathbb{R}^n$ des Problems (7.7) bestimmt werden kann.

Sei $\underline{x} \in \mathbb{R}^n$ eine Lösung von (7.7). Dann gilt nach Definition

$$\Phi(\underline{x}) := \|\underline{b} - \mathbf{A}\underline{x}\|_2^2 \leq \|\underline{b} - \mathbf{A}\underline{y}\|_2^2 \quad \forall \underline{y} \in \mathbb{R}^n. \quad (7.8)$$

Wir betrachten nun die *Richtungsableitung* des Funktionals Φ an einem Minimum $\underline{x} \in \mathbb{R}^n$ in Richtung $\underline{z} \in \mathbb{R}^n$ mit $\|\underline{z}\|_2 = 1$. Die Beziehung (7.8) ist äquivalent dazu, dass die Funktion

$$\begin{aligned} \Phi_{\underline{z}} : \mathbb{R} &\rightarrow \mathbb{R}_+^+ \\ t &\mapsto \Phi_{\underline{z}}(t) := \|\underline{b} - \mathbf{A}(\underline{x} + t\underline{z})\|_2^2 \end{aligned}$$

für jede Richtung $\underline{z} \in \mathbb{R}^n$ ein Minimum bei $t = 0$ hat. Die Funktion $\Phi_{\underline{z}}$ ist ein quadratisches Polynom in t :

$$\begin{aligned} \Phi_{\underline{z}}(t) &= \|\underline{b} - \mathbf{A}(\underline{x} + t\underline{z})\|_2^2 = (\underline{b} - \mathbf{A}(\underline{x} + t\underline{z}))^T (\underline{b} - \mathbf{A}(\underline{x} + t\underline{z})) \\ &= (\underline{x} + t\underline{z})^T \mathbf{A}^T \mathbf{A} (\underline{x} + t\underline{z}) - 2(\underline{x} + t\underline{z})^T \mathbf{A}^T \underline{b} + \underline{b}^T \underline{b} \\ &= t^2 (\underline{z}^T \mathbf{A}^T \mathbf{A} \underline{z}) + 2t (\underline{z}^T \mathbf{A}^T \mathbf{A} \underline{x} - \underline{z}^T \mathbf{A}^T \underline{b}) + (\underline{x}^T \mathbf{A}^T \mathbf{A} \underline{x} + \underline{b}^T \underline{b} - 2\underline{x}^T \mathbf{A}^T \underline{b}). \end{aligned}$$

Also hat $\Phi_{\underline{z}}$ genau dann ein Minimum bei $t = 0$, wenn gilt

$$0 = \Phi'_{\underline{z}}(0) = 2 (\underline{z}^T \mathbf{A}^T \mathbf{A} \underline{x} - \underline{z}^T \mathbf{A}^T \underline{b}) = 2 \underline{z}^T (\mathbf{A}^T \mathbf{A} \underline{x} - \mathbf{A}^T \underline{b}). \quad (7.9)$$

Da (7.9) für *jedes* $\underline{z} \in \mathbb{R}^n$ gilt, erhalten wir als notwendige und hinreichende Bedingung für die gesuchte Lösung $\underline{x} \in \mathbb{R}^n$:

$$\mathbf{A}^T \mathbf{A} \underline{x} = \mathbf{A}^T \underline{b}. \quad (7.10)$$

Die Gleichung (7.10) wird als **Normalgleichung** bezeichnet. Das lineare Gleichungssystem (7.10) ist eindeutig lösbar, da $\mathbf{A}^T \mathbf{A}$ wegen dem vollen Spaltenrang von \mathbf{A} symmetrisch positiv definit ist (Beweis Übung).

Wir halten diese Ergebnisse in dem folgenden Satz fest.

Satz 7.3 Sei $m \geq n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\underline{b} \in \mathbb{R}^m$ und $n = \text{rang}(\mathbf{A}) \leq m$. Ein Vektor $\underline{x} \in \mathbb{R}^n$ ist genau dann Lösung des Ausgleichsproblems (7.7), wenn \underline{x} die Normalgleichung (7.10) erfüllt.

Beweis. Obige Herleitung der Normalgleichung zeigt die Äquivalenz von (7.10) mit $\|\mathbf{A}\underline{x} - \underline{b}\|_2 \leq \|\mathbf{A}\underline{y} - \underline{b}\|_2$ für alle $\underline{y} \in \mathbb{R}^n$. Da es nur ein \underline{x} gibt, das (7.10) erfüllt, ist \underline{x} auch Minimallösung und löst demnach (7.7). \square

Bemerkung 7.4 (Ausgleichsproblem bei vollem Spaltenrang) Für den Fall $n = \text{rang}(\mathbf{A})$ reduziert sich das Ausgleichsproblem (7.7) auf

$$\text{Finde } \underline{x} \in \mathbb{R}^n, \text{ so dass } \|\mathbf{A}\underline{x} - \underline{b}\| \leq \|\mathbf{A}\underline{y} - \underline{b}\| \quad \forall \underline{y} \in \mathbb{R}^n. \quad (7.11)$$

Das Gleichungssystem (7.10) kann z.B. mithilfe der Cholesky-Zerlegung der SPD-Matrix $\mathbf{A}^T \mathbf{A}$ gelöst werden. Es ergibt sich folgender Algorithmus.

Algorithmus 7.5 (Methode der Normalgleichungen)

1. $\mathbf{C} := \mathbf{A}^T \mathbf{A}$,
2. Bestimme Cholesky-Faktor \mathbf{R} von \mathbf{C} mithilfe von Alg. 2.36 (d.h. $\mathbf{R}^T \mathbf{R} = \mathbf{C}$).
3. $\underline{b}' := \mathbf{A}^T \underline{b}$.
4. Löse $\mathbf{R}^T \underline{y} = \underline{b}'$ mithilfe der Vorwärtssubstitution (Alg. 2.1).
5. Löse $\mathbf{R} \underline{x} = \underline{y}$ mithilfe der Rückwärtssubstitution (Alg. 2.2).

Beispiel 7.6 Wir kommen nun auf Beispiel 7.1 zurück. Für das überbestimmte

Aufstellen von $\mathbf{A}^T \mathbf{A}$	$n(n+1)m$
Cholesky-Zerlegung von $\mathbf{A}^T \mathbf{A}$	$\frac{1}{3}n^3$
$\mathbf{A}^T \underline{b}$	$2mn$
Vorwärtssubstitution	n^2
Rückwärtssubstitution	n^2

Tabelle 7.1. Flops bei Lösung eines Ausgleichsproblems mithilfe der Normalgleichung.

System (7.4) ist die gesuchte Normalgleichung $\mathbf{A}^T \mathbf{A} \underline{x} = \mathbf{A}^T \underline{y}$

$$\mathbf{A}^T \mathbf{A} = \begin{pmatrix} 0.1 & -0.005 & 0.4 & 0.5 & 0.9 & 1.0 & 1.2 & 2.0 \\ -0.005 & -0.08 & -0.125 & -0.405 & -0.5 & -0.72 & -2.0 & 2.0 \\ 0.4 & -0.08 & 0.5 & -0.125 & 0.9 & 1.0 & 1.2 & 2.0 \\ 0.5 & -0.125 & -0.405 & 0.9 & 1.0 & 1.2 & 2.0 & 2.0 \\ 0.9 & -0.405 & -0.5 & -0.72 & 1.0 & 1.2 & 2.0 & 2.0 \\ 1.0 & -0.5 & -0.72 & 1.2 & 2.0 & 2.0 & 2.0 & 2.0 \\ 1.2 & -0.72 & 2.0 & 2.0 & 2.0 & 2.0 & 2.0 & 2.0 \\ 2.0 & -2.0 & 2.0 & 2.0 & 2.0 & 2.0 & 2.0 & 2.0 \end{pmatrix} \begin{pmatrix} 0.1 & -0.005 \\ 0.4 & -0.08 \\ 0.5 & -0.125 \\ 0.9 & -0.405 \\ 1.0 & -0.5 \\ 1.2 & -0.72 \\ 2.0 & -2.0 \end{pmatrix}$$

$$= \begin{pmatrix} 7.6700 & -5.8235 \\ -5.8235 & 4.954475 \end{pmatrix}$$

$$\mathbf{A}^T \underline{y} = \begin{pmatrix} 0.1 & 0.4 & 0.5 & 0.9 & 1.0 & 1.2 & 2.0 \\ -0.005 & -0.08 & -0.125 & -0.405 & -0.5 & -0.72 & -2.0 \end{pmatrix} \begin{pmatrix} 0.96 \\ 3.26 \\ 3.82 \\ 5.11 \\ 5.2 \\ 5.05 \\ 0.58 \end{pmatrix} = \begin{pmatrix} 20.3290 \\ -10.20865 \end{pmatrix}$$

Die gesuchte Ausgleichslösung (v_y, g) des überbestimmten Systems (7.4) erfüllt nach (7.10)

$$\begin{pmatrix} 7.6700 & -5.8235 \\ -5.8235 & 4.954475 \end{pmatrix} \begin{pmatrix} v_y \\ g \end{pmatrix} = \begin{pmatrix} 20.3290 \\ -10.20865 \end{pmatrix}.$$

Die Matrix $\mathbf{A}^T \mathbf{A}$ ist in der Tat symmetrisch positiv definit, und die damit eindeutige Lösung (v_y, g) ist auf 5 Stellen

$$\begin{pmatrix} v_y \\ g \end{pmatrix} = \begin{pmatrix} 10.096 \\ 9.8065 \end{pmatrix}.$$

◇

Die **Komplexität von Algorithmus 7.5** wird in Tabelle 7.1 zusammengefasst. Dabei wurde bei der Berechnung von $\mathbf{A}^T \mathbf{A}$ ausgenutzt, dass $\mathbf{A}^T \mathbf{A}$ symmetrisch ist. Man sieht, dass für den Fall $m \gg n$ das Aufstellen der Normalgleichungen (7.10) (d.h. die Berechnung von $\mathbf{A}^T \mathbf{A}$ und $\mathbf{A}^T \underline{b}$) die Gesamtkosten dominiert. Dieser Fall tritt oft ein, weil in vielen Anwendungen die Anzahl n der zu bestimmenden Parameter relativ klein und die Anzahl m der Messungen gross ist, wie wir schon in Beispiel 7.6 beobachten.

Bemerkung 7.7 Die Normalgleichung (7.10) hat eine **geometrische Interpretation**: Aus

$$\mathbf{A}^T (\underline{b} - \mathbf{A} \underline{x}) = 0$$

folgt, dass das Residuum $\underline{r} = \underline{b} - \mathbf{A} \underline{x}$ senkrecht auf den Spalten von \mathbf{A} steht, d.h., das Residuum \underline{r} ist eine Normale zum von den Spalten der Matrix \mathbf{A} aufgespannten Raum. Daher erklärt sich auch die Bezeichnung Normalgleichung.

7.3 Methode der Orthogonalisierung

Die Normalgleichungen haben einen entscheidenden Nachteil: sie sind numerisch instabil, wenn $\kappa(\mathbf{A}) \gg 1$.²⁴

Beispiel 7.8 Sei $m = n$. Betrachte die Vandermonde-Matrix

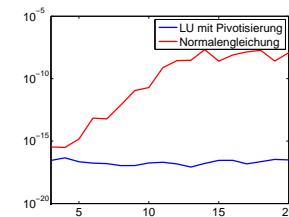
$$\mathbf{A} = \begin{pmatrix} t_0^0 & t_0^1 & \dots & t_0^{n-1} \\ t_1^0 & t_1^1 & \dots & t_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n-1}^0 & t_{n-1}^1 & \dots & t_{n-1}^{n-1} \end{pmatrix}, \quad t_i = i/(n-1).$$

Das Gleichungssystem $\mathbf{A} \underline{x} = \underline{b}$ mit zufälliger rechter Seite \underline{b} wird (a) mittels LU mit Spaltenpivotisierung und (b) mittels Cholesky angewandt auf die Normalgleichung gelöst. Für die berechnete Lösung $\hat{\underline{x}}$ wird jeweils die relative Norm des Residuums berechnet:

$$\frac{\|\underline{r}\|_2}{\|\mathbf{A}\|_2 \|\hat{\underline{x}}\|_2 + \|\underline{b}\|_2},$$

mit

$$\underline{r} = \underline{b} - \mathbf{A} \hat{\underline{x}}.$$



Es zeigt sich, dass die Normalgleichung numerisch instabil ist; der Rückwärtsfehler ist in doppelter Genauigkeit wesentlich höher als 10^{-16} . ◇

Der in Beispiel 7.8 beobachtete Effekt ist im wesentlichen darauf zurückzuführen, dass $\kappa(\mathbf{A}^T \mathbf{A}) = \kappa(\mathbf{A})^2$ gilt. Wir müssen also Alternativen zur Normalgleichung suchen. Die oben benützte **LR-Zerlegung** für \mathbf{A} scheidet bereits aus; sie ist nicht sinnvoll auf den Fall $m > n$ erweiterbar. Stattdessen werden wir Orthogonalisierungsverfahren verwenden, um \mathbf{A} auf einfachere Form zu bringen. Wir betrachten zuerst den Spezialfall, dass die Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ bereits **obere Dreiecksstruktur** hat in dem Sinn, dass

$$\mathbf{A} = \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \quad (7.12)$$

für eine obere Dreiecksmatrix $\mathbf{R} \in \mathbb{R}^{n \times n}$. Der Vektor $\underline{b} \in \mathbb{R}^m$ sei analog zerlegt:

$$\underline{b} = \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \end{pmatrix}, \quad \underline{b}_1 \in \mathbb{R}^n, \quad \underline{b}_2 \in \mathbb{R}^{m-n}.$$

Damit berechnen wir

$$\Phi(\underline{x}) = \|\underline{b} - \mathbf{A} \underline{x}\|_2^2 = \left\| \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \end{pmatrix} - \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \underline{x} \right\|_2^2 = \left\| \begin{pmatrix} \underline{b}_1 - \mathbf{R} \underline{x} \\ \underline{b}_2 - \mathbf{0} \underline{x} \end{pmatrix} \right\|_2^2 = \|\underline{b}_1 - \mathbf{R} \underline{x}\|_2^2 + \|\underline{b}_2\|_2^2.$$

Ist die Rechtsdreiecksmatrix $\mathbf{R} \in \mathbb{R}^{n \times n}$ invertierbar, so ist die Lösung des Minimierungsproblems (7.7) offensichtlich gegeben durch die Lösung $\underline{x} \in \mathbb{R}^n$ von $\mathbf{R} \underline{x} = \underline{b}_1$. D.h.,

²⁴Die Konditionszahl einer rechteckigen Matrix ist als der Quotient zwischen grösstem und kleinstem Singulärwert definiert.

die gesuchte Lösung \underline{x} kann durch Rückwärtssubstitution (Algorithmus 2.2) des Gleichungssystems $\mathbf{R}\underline{x} = \underline{b}_1$ bestimmt werden.

Bei Orthogonalisierungsverfahren wird die Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ so mithilfe orthogonaler Matrizen transformiert, dass sie die Gestalt (7.12) hat. Dabei nutzen wir wesentlich aus, dass orthogonale Matrizen die $\|\cdot\|_2$ -Norm nach Proposition 0.49 invariant lassen:

Satz 7.9 Sei $m \geq n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ mit $\text{rang}(\mathbf{A}) = n$, $\underline{b} \in \mathbb{R}^m$. Sei $\mathbf{Q} \in \mathbb{R}^{m \times m}$ orthogonale Matrix und $\mathbf{R} \in \mathbb{R}^{n \times n}$ obere Dreiecksmatrix, so dass gilt:

$$\mathbf{A} = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}.$$

Sei ferner der Vektor $\tilde{\underline{b}} = \mathbf{Q}^T \underline{b} \in \mathbb{R}^m$ wie folgt partitioniert:

$$\mathbf{Q}^T \underline{b} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}, \quad \tilde{b}_1 \in \mathbb{R}^n, \quad \tilde{b}_2 \in \mathbb{R}^{m-n}.$$

Dann ist die Lösung $\underline{x} \in \mathbb{R}^n$ des Ausgleichsproblems (7.11) die eindeutige Lösung des Dreieckssystems

$$\mathbf{R}\underline{x} = \tilde{b}_1.$$

Beweis. Unter Ausnutzung der Tatsache, dass die Anwendung einer orthogonalen Matrix \mathbf{Q}^T die Euklidische Länge eines Vektors invariant lässt berechnen wir

$$\|\underline{b} - \mathbf{A}\underline{x}\|_2 = \|\mathbf{Q}^T(\underline{b} - \mathbf{A}\underline{x})\|_2 = \|\mathbf{Q}^T \underline{b} - \mathbf{Q}^T \mathbf{A}\underline{x}\|_2 = \left\| \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} - \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \underline{x} \right\|_2.$$

Also ist das ursprüngliche Ausgleichsproblem (7.7) äquivalent zu einem Ausgleichsproblem von der oben betrachteten Form. Da die Spalten von \mathbf{A} linear unabhängig sind, ist die Rechtsdreiecksmatrix \mathbf{R} invertierbar (Übung: Man überzeuge sich davon!). Die Aussage des Satzes folgt. \square

Es bleibt \mathbf{Q} aus Satz 7.9 zu berechnen. Ein Verfahren kennen wir dabei schon: Gram-Schmidt liefert (fast) eine solche \mathbf{QR} -Zerlegung von \mathbf{A} . Es wird sich aber herausstellen, dass der klassische Gram-Schmidt-Algorithmus massive numerische Probleme hat (die zwar teilweise behoben werden können); dies wird uns zu Alternativen wie der Householder- \mathbf{QR} -Zerlegung führen.

7.4 Gram-Schmidt und modifizierter Gram-Schmidt

Der Vollständigkeit halber wiederholen wir den Gram-Schmidt-Algorithmus aus Abschnitt 0.8.2, jetzt im \mathbb{R}^m statt \mathbb{C}^n . Der folgende Algorithmus findet für gegebene linear unabhängige Vektoren $\underline{x}_1, \dots, \underline{x}_n \in \mathbb{R}^m$ eine Orthonormalbasis $\underline{q}_1, \dots, \underline{q}_n \in \mathbb{R}^m$, so dass

$$\text{span}\{\underline{x}_1, \dots, \underline{x}_\ell\} = \text{span}\{\underline{q}_1, \dots, \underline{q}_\ell\}, \quad \forall 1 \leq \ell \leq n. \tag{7.13}$$

Algorithmus 7.10 (Gram-Schmidt)

Input: Linear unabhängige Vektoren $\underline{a}_1, \dots, \underline{a}_n \in \mathbb{R}^m$.
Output: Orthonormalbasis $\underline{q}_1, \dots, \underline{q}_n$, so dass (7.13) erfüllt ist.

for $\ell = 1, \dots, m$ **do**
 $\hat{\underline{q}}_\ell := \underline{a}_\ell - \sum_{j=1}^{\ell-1} (q_j^T \underline{a}_\ell) \underline{q}_j$
 $\underline{q}_\ell := \frac{\hat{\underline{q}}_\ell}{\|\hat{\underline{q}}_\ell\|_2}$
end for

```

MATLAB
% Die Spalten von A enthalten
% a_1, ..., a_n und werden mit
% q_1, ..., q_n ueberschrieben.
for l = 1:n,
    sl = A(:,1:l-1)'*A(:,l);
    qlhut = A(:,l) - A(:,1:l-1)*sl;
    rll = norm(qlhut);
    A(:,l) = qlhut / rll;
end
    
```

Der Gram-Schmidt-Algorithmus erzeugt eine \mathbf{QR} -Zerlegung der $m \times n$ Matrix $\mathbf{A} = (\underline{a}_1, \dots, \underline{a}_n)$. Dies sieht man wie folgt. Setzen wir $r_{\ell\ell} := \|\hat{\underline{q}}_\ell\|_2$ und $r_{j\ell} = (\underline{q}_j, \underline{a}_\ell)$, so folgt aus Algorithmus 7.10 die Beziehung

$$\underline{a}_\ell = \hat{\underline{q}}_\ell + \sum_{j=1}^{\ell-1} (q_j, \underline{a}_\ell) \underline{q}_j = \sum_{j=1}^{\ell} r_{j\ell} \underline{q}_j.$$

In Matrix-Schreibweise:

$$\mathbf{A} = \mathbf{Q}_1 \mathbf{R}, \quad \text{mit} \quad \mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{pmatrix}. \tag{7.14}$$

Es ist zu beachten, dass \mathbf{Q}_1 nicht quadratisch ist! Eine Zerlegung der Form (7.14) nennt man deshalb auch **ökonomische \mathbf{QR} -Zerlegung** (siehe `help qr` in MATLAB). Eigentlich reicht diese aus, um das Ausgleichsproblem zu lösen, da wir in Satz 7.9 lediglich $\tilde{\underline{b}}_1 = \mathbf{Q}_1^T \underline{b}$ benötigen. Zur Vollständigkeit geben wir aber noch an, wie aus dem Gram-Schmidt-Algorithmus die **vollständige \mathbf{QR} -Zerlegung** von \mathbf{A} gewonnen werden kann. Dabei wird auch ein interessanter Zusammenhang zur Cholesky-Zerlegung von $\mathbf{A}^T \mathbf{A}$ aufgezeigt.

Satz 7.11 Sei $\mathbf{A} \in \mathbb{R}^{m \times n}$ mit $\text{rang}(\mathbf{A}) = n$. Dann gibt es eine orthogonale Matrix $\mathbf{Q} = (\mathbf{Q}_1, \mathbf{Q}_2) \in \mathbb{R}^{m \times m}$, so dass

$$\mathbf{A} = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}, \tag{7.15}$$

wobei $\mathbf{R} \in \mathbb{R}^{n \times n}$ obere Dreiecksmatrix mit positiven Diagonaleinträgen ist. Die Matrizen $\mathbf{Q}_1 \in \mathbb{R}^{m \times n}$ und $\mathbf{R} \in \mathbb{R}^{n \times n}$ sind dabei eindeutig bestimmt.

Beweis. Nach dem Basisergänzungssatz gibt es Vektoren $\underline{a}_{n+1}, \dots, \underline{a}_m \in \mathbb{R}^m$, so dass

$$\tilde{\mathbf{A}} = (\underline{a}_1, \dots, \underline{a}_n, \underline{a}_{n+1}, \dots, \underline{a}_m) \in \mathbb{R}^{m \times m}$$

invertierbar ist. Anwendung von Gram-Schmidt auf diese m Vektoren ergibt eine

QR -Zerlegung der Form (7.14):

$$\tilde{A} = Q\tilde{R} = (Q_1, Q_2) \begin{pmatrix} R & * \\ 0 & R_2 \end{pmatrix}, \tag{7.16}$$

wobei nach Konstruktion R eine obere Dreiecksmatrix mit positiven Diagonaleinträgen ist. Betrachten wir nur die ersten n Spalten in (7.16), so ergibt sich $A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$.

Für den zweiten Teil bemerken wir zunächst, dass aus der Zerlegung (7.15) folgt

$$A^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}^T \underbrace{Q^T Q}_{=I} \begin{pmatrix} R \\ 0 \end{pmatrix} = R^T R.$$

Also ist R ein Cholesky-Faktor von $A^T A$. Dieser ist eindeutig.²⁵ Aus der Eindeutigkeit von R folgt auch die Eindeutigkeit von Q_1 , da $Q_1 = AR^{-1}$ gelten muss. \square

Wir werden in Beispiel 7.12 sehen, dass die durch den Gram-Schmidt-Algorithmus produzierte Basis unter Rundungsfehlern weit von Orthonormalität entfernt sein kann. Dies tritt insbesondere dann auf, wenn A fast linear abhängige Spalten hat. Eine robustere Variante ist die folgende Modifikation, bei der alle verbliebenen Vektoren auf $\text{span}\{q_l\}^\perp$ projiziert werden, sobald q_l verfügbar ist.

```

MATLAB
% Modifizierter Gram-Schmidt-Algorithmus
% Die Spalten von A enthalten
% a_1, ..., a_n und werden mit
% q_1, ..., q_n ueberschrieben.
for l = 1:n,
    rll = norm( A(:,l) );
    ql = A(:,l) / rll;
    for j = l+1:m,
        rlj = ql'*A(:,j);
        A(:,j) = A(:,j) - rlj*ql;
    end
    A(:,l) = ql;
end
    
```

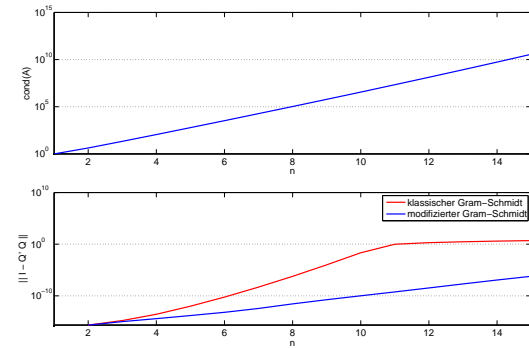
Die QR -Zerlegung lässt sich analog wie beim Gram-Schmidt-Algorithmus gewinnen.

Beispiel 7.12 Wir betrachten die ersten $n \leq m = 25$ Spalten der Vandermonde-Matrix aus Beispiel 7.8:

$$A = \begin{pmatrix} t_0^0 & t_0^1 & \dots & t_0^{n-1} \\ t_1^0 & t_1^1 & \dots & t_1^{n-1} \\ \vdots & \vdots & & \vdots \\ t_{24}^0 & t_{24}^1 & \dots & t_{24}^{n-1} \end{pmatrix}, \quad t_i = i/24.$$

²⁵Dies haben wir zwar formal nicht gezeigt; folgt aber sofort aus der Eindeutigkeit der LR-Zerlegung.

Auf A wird für $n = 2, \dots, 25$ (klassischer) Gram-Schmidt und modifizierter Gram-Schmidt angewendet. Die Orthonormalität der im berechneten \hat{Q}_1 enthaltenen Basis wird mittels $\|I_n - \hat{Q}_1^T \hat{Q}_1\|_2$ gemessen.



Es zeigt sich, dass die Orthonormalität beim klassischen Gram-Schmidt bereits für $n = 11$ verloren geht, während sie beim modifizieren Gram-Schmidt besser – wenngleich auch nicht perfekt – erhalten bleibt. \diamond

Der in Beispiel 7.12 beobachtete Verlust der Orthonormalität ist auf numerische Auslöschung zurückzuführen. Ist A nicht extrem schlecht konditioniert, so bietet sich als einfacher Ausweg an, einfach die Orthogonalisierung zu wiederholen. Im nächsten Abschnitt lernen wir aber eine elegantere robuste Alternative zu Gram-Schmidt kennen.

7.5 Householder-basierte QR -Zerlegung

In der Householder-basierten QR -Zerlegung baut man die Matrix Q – wie die Matrix L in der Gauss-Elimination – durch sukzessives Aufmultiplizieren *elementarer* orthogonaler Matrizen auf. Für die elementaren Matrizen gibt es zwei Konstruktionsmöglichkeiten: *Householder-Reflexionen* sowie *Givens-Rotationen*. Wir diskutieren im Rahmen dieser Vorlesung nur die Householder-Reflexionen. Givens-Rotationen sind dann von Interesse, wenn A bestimmte Eigenschaften hat, wie zum Beispiel obere Hessenberg-Gestalt, siehe [Golub/Van Loan'1996].

7.5.1 Konstruktion

Beim Orthogonalisieren mit Householder-Reflexionen wird eine QR -Zerlegung einer Matrix erzeugt, bei der die orthogonale Matrix Q das Produkt von sogenannten Householder-Reflexionen ist.

Satz 7.13 (Eigenschaften von Householder-Reflexionen) Sei $u \neq v \in \mathbb{R}^m$. Dann hat die Householder-Reflexion

$$Q := I_m - \frac{2}{v^T v} v v^T$$

die folgenden Eigenschaften:

1. Q ist symmetrisch, d.h. $Q^T = Q$.
2. Q ist orthogonal, d.h. $Q^T = Q^{-1}$.
3. Q ist involutorisch, d.h. $Q^2 = I_m$.

Beweis. Übung. \square

Bemerkung 7.14 Die Householder-Reflexionen haben eine geometrische Interpretation: Als lineare Abbildung gesehen, stellen sie eine Spiegelung ("Reflexion") an der Hyperebene $H := \{\underline{x} \in \mathbb{R}^m \mid \underline{x}^T \underline{v} = 0\}$ dar. In der Tat rechnet man nach: Für $\underline{x} \in H$ gilt $Q\underline{x} = \underline{x}$ und für \underline{x} parallel zu \underline{v} gilt $Q\underline{x} = -\underline{x}$.

Eine QR -Zerlegung einer Matrix $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ wird mithilfe von Householder-Reflexionen erzeugt, indem man schrittweise die Matrix A durch Multiplikation mit Householder-Reflexionen $Q^{(1)}, \dots, Q^{(n)}$ auf obere Dreiecksgestalt bringt. Entscheidend ist dabei das folgende Resultat.

Lemma 7.15 Sei $\underline{0} \neq \underline{a} \in \mathbb{R}^m$ und $\underline{e}_1 \in \mathbb{R}^m$ der 1. Einheitsvektor. Sei

$$\alpha = \|\underline{a}\|_2 \quad \text{oder} \quad \alpha = -\|\underline{a}\|_2.$$

(Im Falle $\underline{a} = \beta \underline{e}_1$ sei $\alpha = -\|\underline{a}\|_2 = -|\beta|$ gewählt.) Dann gilt für $\underline{v} := \underline{a} - \alpha \underline{e}_1$:

$$Q = I_m - \frac{2}{\underline{v}^T \underline{v}} \underline{v} \underline{v}^T \quad \Rightarrow \quad Q \underline{a} = \alpha \underline{e}_1. \quad (7.17)$$

Beweis. Die Wahl von α impliziert, dass $\underline{v} \neq \underline{0}$. Die Behauptung $Q \underline{a} = \alpha \underline{e}_1$ kann dann direkt nachgerechnet werden (Übung: man rechne dies tatsächlich nach!), und das Lemma ist damit bewiesen. \square

Bemerkung 7.16 Bei der Bestimmung von \underline{v} mithilfe von Lemma 7.15 wählt man in der Praxis $\alpha = -\text{sign}(a_1) \sqrt{\underline{a}^T \underline{a}}$, um Auslöschung bei der Auswertung von $\underline{v} = (a_1 - \alpha, a_2, \dots, a_k)^T$ zu vermeiden²⁶.

Bezeichne \underline{a}_1 die erste Spalte von $A \in \mathbb{R}^{m \times n}$. Im ersten Schritt der Householder-basierten QR -Zerlegung von A konstruieren wir mittels Lemma 7.15 eine Householder-Reflexion $Q^{(1)}$, so dass

$$Q^{(1)} \underline{a}_1 = \begin{pmatrix} r_{11} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

für ein $r_{11} \in \mathbb{R}$. Angewandt auf die Gesamtmatrix ergibt sich folgende Struktur:

$$Q^{(1)} A = \left(\begin{array}{c|ccc} r_{11} & r_{12} & \cdots & r_{1n} \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \begin{array}{c} \\ \\ \\ A^{(1)} \end{array} \right), \quad (7.18)$$

²⁶Wir definieren $\text{sign}(x) = 1$ für $x \geq 0$ und $\text{sign}(x) = -1$ für $x < 0$

mit $A^{(1)} \in \mathbb{R}^{(m-1) \times (n-1)}$.

Um weiter in Richtung Dreiecksgestalt zu transformieren, wiederholt man den Prozess für die Untermatrix $A^{(1)}$. Dazu beobachten wir, dass für jede Matrix $\tilde{Q}^{(2)}$ die Matrix

$$Q^{(2)} := \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \begin{array}{c} \\ \tilde{Q}^{(2)} \\ \\ \end{array} \right),$$

orthogonal ist und zudem

$$\left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \begin{array}{c} \\ \tilde{Q}^{(2)} \\ \\ \end{array} \right) \left(\begin{array}{c|ccc} r_{11} & r_{12} & \cdots & r_{1n} \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \begin{array}{c} \\ \\ \\ A^{(1)} \end{array} \right) = \left(\begin{array}{c|ccc} r_{11} & r_{12} & \cdots & r_{1n} \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \begin{array}{c} \\ \tilde{Q}^{(2)} A^{(1)} \\ \\ \end{array} \right)$$

gilt. Bezeichne \underline{a}_2 die erste Spalte von $A^{(1)}$, so wählen wir $\tilde{Q}^{(2)}$ als Householder-Reflexion die \underline{a}_2 auf ein skalares Vielfaches von $\underline{e}_1 \in \mathbb{R}^{m-1}$ transformiert. Dann hat das Produkt $Q^{(2)} Q^{(1)} A$ die Form

$$Q^{(2)} Q^{(1)} A = \left(\begin{array}{cc|ccc} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & & \\ 0 & 0 & & & \end{array} \begin{array}{c} \\ \\ A^{(2)} \\ \\ \end{array} \right),$$

Man setzt diesen Prozess so lange fort, bis die Matrix $Q^{(n)} Q^{(n-1)} \dots Q^{(1)} A$ obere Dreiecksgestalt hat (im Fall $m = n$ kann man sich $Q^{(n)}$ sparen).

Somit ist die gesuchte QR -Zerlegung von $A \in \mathbb{R}^{n \times n}$:

$$A = (Q^{(1)})^T \dots (Q^{(n)})^T R = \underbrace{Q^{(1)} \dots Q^{(n)}}_{=: Q} \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix},$$

wobei $R \in \mathbb{R}^{n \times n}$ eine obere Dreiecksmatrix ist und die Matrizen $Q^{(k)} \in \mathbb{R}^{m \times m}$ die folgende Form haben:

$$Q^{(k)} = \left(\begin{array}{c|ccc} I_{k-1} & & & 0 \cdots 0 \\ \hline 0 & & & \\ \vdots & & & \\ 0 & & & \end{array} \begin{array}{c} \\ I_{m-k+1} - \frac{2}{\underline{v}_k^T \underline{v}_k} \underline{v}_k \underline{v}_k^T \\ \\ \end{array} \right), \quad \underline{v}_k \in \mathbb{R}^{m-k+1}. \quad (7.19)$$

Formal wird die Prozedur im folgenden Algorithmus zusammengefasst.

Algorithmus 7.17 (QR -Zerlegung)

Input: Matrix $A \in \mathbb{R}^{m \times n}$.

Output: QR -Zerlegung $A = Q \begin{pmatrix} R \\ \mathbf{0} \end{pmatrix}$ mit Q orthogonal und R in oberer Dreiecksform.

$Q = I_m$
for $k = 1, \dots, \min\{n, m-1\}$ **do**
 Bestimme Householder-Reflexion $Q^{(k)}$ (vgl. (7.19)), so dass die letzten $m-k$ Einträge in der k -ten Spalte von $Q^{(k)}A$ Null werden.
 Überschreibe $A \leftarrow Q^{(k)}A$.
 Überschreibe $Q \leftarrow Q^{(k)}Q$.
end for
 Setze R als die ersten n Zeilen von A .

Beispiel 7.18 Sei

$$A = \begin{pmatrix} -4 & -2-2\sqrt{6} & -6-3\sqrt{2}-\sqrt{6} \\ 0 & -2\sqrt{3} & 9-\sqrt{3} \\ -4\sqrt{2} & -2\sqrt{2}+2\sqrt{3} & 3-6\sqrt{2}+\sqrt{3} \end{pmatrix}.$$

Gesucht ist eine QR -Zerlegung von A . Unsere Handrechnung stimmt im wesentlichen mit Algorithmus 7.17 überein. Die erste Spalte von A ist

$$\underline{a}_1 = \begin{pmatrix} -4 \\ 0 \\ -4\sqrt{2} \end{pmatrix}, \quad \|\underline{a}_1\|_2 = \sqrt{16+16 \cdot 2} = \sqrt{48} = 4\sqrt{3}.$$

Also ist $\alpha = -\text{sign}(a_1)\|\underline{a}_1\|_2 = -\text{sign}(-4)4\sqrt{3} = 4\sqrt{3}$ und damit

$$\underline{v}_1 = \underline{a}_1 - \alpha \underline{e}_1 = \begin{pmatrix} -4 \\ 0 \\ -4\sqrt{2} \end{pmatrix} - 4\sqrt{3} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -4-4\sqrt{3} \\ 0 \\ -4\sqrt{2} \end{pmatrix}, \quad \|\underline{v}_1\|_2^2 = 16(6+2\sqrt{3}).$$

Die erste Householder-Reflexion ist somit

$$\begin{aligned} Q^{(1)} &= I_3 - \frac{2}{\|\underline{v}_1\|_2^2} \underline{v}_1 \underline{v}_1^T = I_3 - \frac{2}{16(6+2\sqrt{3})} \begin{pmatrix} -4-4\sqrt{3} \\ 0 \\ -4\sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} -4-4\sqrt{3} & 0 & -4\sqrt{2} \end{pmatrix} \\ &= I_3 - \frac{1}{3+2\sqrt{3}} \begin{pmatrix} -1-\sqrt{3} \\ 0 \\ -\sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} -1-\sqrt{3} & 0 & -\sqrt{2} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{\sqrt{3}(1+\sqrt{3})} \begin{pmatrix} 4+2\sqrt{3} & 0 & \sqrt{2}(1+\sqrt{3}) \\ 0 & 0 & 0 \\ \sqrt{2}(1+\sqrt{3}) & 0 & 2 \end{pmatrix} \\ &= \frac{1}{\sqrt{3}} \begin{pmatrix} -1 & 0 & -\sqrt{2} \\ 0 & \sqrt{3} & 0 \\ -\sqrt{2} & 0 & 1 \end{pmatrix}. \end{aligned}$$

Man sieht, dass – wie erwartet – $Q^{(1)}$ eine orthogonale Matrix ist). Wir erhalten damit für $Q^{(1)}A$:

$$\begin{aligned} Q^{(1)}A &= \frac{1}{\sqrt{3}} \begin{pmatrix} -1 & 0 & -\sqrt{2} \\ 0 & \sqrt{3} & 0 \\ -\sqrt{2} & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -4 & -2-2\sqrt{6} & -6-3\sqrt{2}-\sqrt{6} \\ 0 & -2\sqrt{3} & 9-\sqrt{3} \\ -4\sqrt{2} & -2\sqrt{2}+2\sqrt{3} & 3-6\sqrt{2}+\sqrt{3} \end{pmatrix} \\ &= \frac{1}{\sqrt{3}} \begin{pmatrix} 12 & 6 & 18 \\ 0 & -6 & -3+9\sqrt{3} \\ 0 & 6\sqrt{3} & 9+3\sqrt{3} \end{pmatrix} = \sqrt{3} \begin{pmatrix} 4 & 2 & 6 \\ 0 & -2 & -1+3\sqrt{3} \\ 0 & 2\sqrt{3} & 3+\sqrt{3} \end{pmatrix}. \end{aligned}$$

Die erste Spalte von $Q^{(1)}A$ stimmt mit $\alpha \underline{e}_1$ überein—so wurde die Householder-Reflexion $Q^{(1)}$ schliesslich konstruiert!

Wir wiederholen nun obiges Vorgehen für die Untermatrix

$$A^{(1)} = \sqrt{3} \begin{pmatrix} -2 & -1+3\sqrt{3} \\ 2\sqrt{3} & 3+\sqrt{3} \end{pmatrix}.$$

Die erste Spalte von $A^{(1)}$ ist

$$\underline{a}_2 = \begin{pmatrix} -2\sqrt{3} \\ 6 \end{pmatrix}, \quad \|\underline{a}_2\|_2 = \sqrt{12+36} = 4\sqrt{3}.$$

Also ist nun $\alpha = -\text{sign}(a_1)\|\underline{a}_2\|_2 = -\text{sign}(-2\sqrt{3}) \cdot 4\sqrt{3} = 4\sqrt{3}$ und für $\underline{v}_2 = \underline{a}_2 - \alpha \underline{e}_1$ erhalten wir

$$\underline{v}_2 = \begin{pmatrix} -2\sqrt{3} \\ 6 \end{pmatrix} - 4\sqrt{3} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -6\sqrt{3} \\ 6 \end{pmatrix}, \quad \|\underline{v}_2\|_2^2 = 144.$$

Damit ist $\tilde{Q}^{(2)}$ gegeben durch

$$\begin{aligned} \tilde{Q}^{(2)} &= I_2 - \frac{2}{\|\underline{v}_2\|_2^2} \underline{v}_2 \cdot \underline{v}_2^T = I_2 - \frac{2}{144} \cdot \begin{pmatrix} -6\sqrt{3} \\ 6 \end{pmatrix} \cdot \begin{pmatrix} -6\sqrt{3} & 6 \end{pmatrix} \\ &= I_2 - \frac{1}{2} \cdot \begin{pmatrix} -\sqrt{3} \\ 1 \end{pmatrix} \cdot \begin{pmatrix} -\sqrt{3} & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 3 & -\sqrt{3} \\ -\sqrt{3} & 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} -1 & \sqrt{3} \\ \sqrt{3} & 1 \end{pmatrix}. \end{aligned}$$

Es gilt

$$\tilde{Q}^{(2)} A^{(1)} = \frac{1}{2} \begin{pmatrix} -1 & \sqrt{3} \\ \sqrt{3} & 1 \end{pmatrix} \cdot \sqrt{3} \begin{pmatrix} -2 & -1+3\sqrt{3} \\ 2\sqrt{3} & 3+\sqrt{3} \end{pmatrix} = \frac{\sqrt{3}}{2} \begin{pmatrix} 8 & 4 \\ 0 & 12 \end{pmatrix}.$$

und die gesuchte obere Dreiecksmatrix R ist

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

Die orthogonale Matrix Q^T mit $Q^T A = R$ ist somit

$$\begin{aligned} Q^T &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \tilde{Q}^{(2)} & 0 \\ 0 & 0 & 1 \end{pmatrix} Q^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{2} & \frac{1}{2}\sqrt{3} \\ 0 & \frac{1}{2}\sqrt{3} & \frac{1}{2} \end{pmatrix} \cdot \frac{1}{\sqrt{3}} \begin{pmatrix} -1 & 0 & -\sqrt{2} \\ 0 & \sqrt{3} & 0 \\ -\sqrt{2} & 0 & 1 \end{pmatrix} \\ &= \frac{1}{2\sqrt{3}} \begin{pmatrix} -2 & 0 & -2\sqrt{2} \\ -\sqrt{6} & -\sqrt{3} & \sqrt{3} \\ -\sqrt{2} & 3 & 1 \end{pmatrix}. \end{aligned}$$

Zur Probe berechnen wir noch QR :

$$\begin{aligned} QR &= \frac{1}{2\sqrt{3}} \begin{pmatrix} -2 & -\sqrt{6} & -\sqrt{2} \\ 0 & -\sqrt{3} & 3 \\ -2\sqrt{2} & \sqrt{3} & 1 \end{pmatrix} \cdot \sqrt{3} \begin{pmatrix} 4 & 2 & 6 \\ 0 & 4 & 2 \\ 0 & 0 & 6 \end{pmatrix} \\ &= \begin{pmatrix} -4 & -2-2\sqrt{6} & -6-3\sqrt{2}-\sqrt{6} \\ 0 & -2\sqrt{3} & 9-\sqrt{3} \\ -4\sqrt{2} & -2\sqrt{2}+2\sqrt{3} & 3-6\sqrt{2}+\sqrt{3} \end{pmatrix} = A \end{aligned}$$

◊

Es wäre natürlich extrem ineffizient, Algorithmus 7.17 wortwörtlich in der gegebenen Form zu implementieren. Insbesondere wird die Matrix $Q^{(k)}$ nie explizit aufgestellt, sondern immer nur indirekt mittels v_k angewandt. Um beispielsweise $Q^{(k)}$ auf eine Matrix $\begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$ mit $B_1 \in \mathbb{R}^{(k-1) \times \ell}$ und $B_2 \in \mathbb{R}^{(m-k+1) \times \ell}$ anzuwenden, schreibt man

$$Q^{(k)} \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 - \frac{2}{v_k^T v_k} v_k v_k^T B_2 \end{pmatrix}.$$

Hierbei berechnet man zuerst den Vektor $w = \frac{2}{v_k^T v_k} B_2^T v_k$ ($\approx 2(m-k+1)\ell$ flops), um danach die Rang-1-Aufdatierung $B_2 - v_k w^T$ ($\approx 2(m-k+1)\ell$ flops) durchzuführen. Unter Ausnutzung der bereits erzeugten Nullstruktur in der Matrix A kostet also die Operation $A \leftarrow Q^{(k)} A$ in Algorithmus 7.17 lediglich $4(m-k+1)(n-k+1)$ flops. Alle Aufdatierungen von A kosten zusammen also

$$4 \sum_{k=1}^n (m-k+1)(n-k+1) \approx 2mn^2 - \frac{2}{3}n^3 \quad (7.20)$$

flops. Analog lassen sich die Aufdatierungen von Q inert

$$4 \sum_{k=1}^n m(m-k+1) \approx 4m^2n - 2mn^2 \quad (7.21)$$

flops durchführen. Weitere Einsparungen kann man erzielen, führt man die Multiplikation von $Q^{(k)}$ in einer anderen Reihenfolge durch, nämlich

$$Q = Q^{(1)} (\dots (Q^{(n-1)} (Q^{(n)} I_n))).$$

Bemerkung 7.19 Bei der Wahl von v_k hat man noch gewisse Freiheiten, insbesondere kann statt v_k auch jedes skalare Vielfache benutzt werden. Es hat sich die folgende Konvention ergeben: v_k wird so gewählt, dass dessen erster Eintrag 1 ist. Dieser erste Eintrag wird natürlich nicht gespeichert. Die restlichen Einträgen von v_k passen gerade in die in A erzeugten Nullen und werden dort abgespeichert.²⁷ Die Koeffizienten $\beta_k = 2/(v_k^T v_k)$ werden getrennt abgespeichert.

7.5.2 Anwendung auf Ausgleichsprobleme

Im Prinzip haben wir mit Algorithmus 7.17 und Satz 7.9 alles zusammen, um das Ausgleichsproblem für den Fall $\text{rang}(A) = n$ zu lösen. Allerdings lässt sich noch die teure Berechnung des orthogonalen Faktors (siehe 7.21) vermeiden, indem wir nicht Q sondern nur $Q^T \underline{b}$ berechnen. Dies ergibt den folgenden Algorithmus.

Algorithmus 7.20 (Ausgleichsproblem mittels QR -Zerlegung)

Input: Matrix $A \in \mathbb{R}^{m \times n}$ mit $\text{rang}(A) = n$, $\underline{b} \in \mathbb{R}^m$

Output: Lösung $\underline{x} \in \mathbb{R}^n$ des Ausgleichsproblems $\min \|A\underline{x} - \underline{b}\|_2$.

for $k = 1, \dots, \min\{n, m-1\}$ **do**

²⁷Dieses Speicherformat wird von `qr` zurückgegeben, wenn nur ein Ausgabeargument angegeben wird.

Bestimme Householder-Reflexion $Q^{(k)}$ (vgl. (7.19)), so dass die letzten $m-k$ Einträge in der k -ten Spalte von $Q^{(k)} A$ Null werden.
 Überschreibe $A \leftarrow Q^{(k)} A$.
 Überschreibe $\underline{b} \leftarrow Q^{(k)} \underline{b}$.
end for
 Partitioniere $A = \begin{pmatrix} R \\ 0 \end{pmatrix}$ sowie $\underline{b} = \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \end{pmatrix}$.
 Berechne $\underline{x} = R^{-1} \underline{b}_1$ mittels Algorithmus 2.2.

Da alle anderen Kosten vernachlässigbar sind, ergeben sich als Gesamtkosten lediglich die Kosten der Aufdatierung von A (siehe (7.20)), also $2mn^2 - \frac{2}{3}n^3$ flops. Verglichen mit den Normalgleichungen, siehe Tabelle 7.1, ist der Aufwand also maximal doppelt so hoch.

7.6 Rechteckmatrizen mit Rangdefekt. Singulärwertzerlegung.

Gilt $\text{rang}(A) < \min\{m, n\}$, so lässt sich keines der bisher besprochenen Verfahren zur Lösung des allgemeinen Ausgleichsproblems einsetzen. So lässt sich zum Beispiel in Algorithmus 7.20 die QR -Zerlegung zwar noch problemlos durchführen, allein R wird singular und damit lässt sich der letzte Schritt zur Bestimmung von \underline{x} nicht durchführen. Statt der QR -Zerlegung benötigen wir die Singulärwertzerlegung von A , siehe Satz 0.61: Es gibt orthogonale Matrizen $U \in \mathbb{R}^{m \times m}$ und $V \in \mathbb{C}^{n \times n}$, so dass

$$A = U \Sigma V^T, \quad (7.22)$$

wobei

$$\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ 0 & & & 0 \end{pmatrix} \in \mathbb{R}^{m \times n}.$$

und

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0.$$

Mit der Zerlegung 7.22 lässt sich das Ausgleichsproblem

$$\text{Finde } \underline{x} \in \mathbb{R}^n \text{ mit } \|\underline{x}\|_2 = \min! \text{ so, dass } \|A\underline{x} - \underline{b}\|_2 \leq \|A\underline{y} - \underline{b}\|_2 \quad \forall \underline{y} \in \mathbb{R}^n, \quad (7.23)$$

wie folgt umschreiben:

$$\text{Finde } \tilde{\underline{x}} \in \mathbb{R}^n \text{ mit } \|\tilde{\underline{x}}\|_2 = \min! \text{ so, dass } \|\Sigma \tilde{\underline{x}} - \tilde{\underline{b}}\|_2 \leq \|\Sigma \underline{y} - \tilde{\underline{b}}\|_2 \quad \forall \underline{y} \in \mathbb{R}^n, \quad (7.24)$$

wobei $\tilde{\underline{b}} = U^T \underline{b}$. Man sieht leicht: $\tilde{\underline{x}}$ ist genau dann Lösung von (7.24), wenn $\underline{x} = V \tilde{\underline{x}}$ Lösung von (7.23) ist.

Lemma 7.21 Die Lösung von (7.24) ist eindeutig gegeben durch $\tilde{\underline{x}} = \Sigma^+ \tilde{\underline{b}}$, wobei

$$\Sigma^+ = \begin{pmatrix} 1/\sigma_1 & & & \\ & \ddots & & 0 \\ & & 1/\sigma_r & \\ 0 & & & 0 \end{pmatrix} \in \mathbb{R}^{n \times m}. \quad (7.25)$$

Beweis. Der Ausdruck

$$\|\Sigma\tilde{\underline{x}} - \tilde{\underline{b}}\|_2^2 = \sum_{j=1}^r |\sigma_j \tilde{x}_j - \tilde{b}_j|^2$$

wird genau dann minimiert, wenn $\tilde{x}_j = \tilde{b}_j/\sigma_j$ für $j = 1, \dots, r$ gilt. Die restlichen $\tilde{x}_{r+1}, \dots, \tilde{x}_n$ sind frei wählbar. Minimales $\|\tilde{\underline{x}}\|_2$ wird genau dann erreicht, wenn $\tilde{x}_{r+1} = \dots = \tilde{x}_n = 0$ gilt. Also folgt $\tilde{\underline{x}} = \Sigma^+ \tilde{\underline{b}}$. \square

Mit Lemma 7.21 erhalten wir für die Lösung des ursprünglichen Ausgleichsproblems (7.23) die Beziehung

$$\underline{x} = \mathbf{V}\tilde{\underline{x}} = \mathbf{V}\Sigma^+\tilde{\underline{b}} = \mathbf{V}\Sigma^+\mathbf{U}^T\underline{b}.$$

Definition 7.22 Sei (7.22) eine Singulärwertzerlegung von $\mathbf{A} \in \mathbb{R}^{m \times n}$. Dann heisst $\mathbf{A}^+ := \mathbf{V}\Sigma^+\mathbf{U}^T \in \mathbb{R}^{n \times m}$ mit Σ^+ wie in (7.25) **Pseudoinverse** von \mathbf{A} .

Mit der Pseudoinversen können wir kompakt $\underline{x} = \mathbf{A}^+\underline{b}$ schreiben; die Pseudoinverse von \mathbf{A} bildet also die rechte Seite \underline{b} auf die Lösung des dazugehörigen Ausgleichsproblems ab.

Beispiel 7.23 Für $\mathbf{A} \in \mathbb{R}^{m \times n}$ mit $\text{rang}(\mathbf{A}) = n$ gilt $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. \diamond

7.7 Niedrigrangapproximation*

Die Anwendungen der Singulärwertzerlegung gehen weit über Ausgleichsprobleme hinaus. Als weiteres Beispiel nennen wir hier die beste Approximation einer gegebenen Matrix durch eine Matrix niedrigeren Rangs.

Satz 7.24 Habe $\mathbf{A} \in \mathbb{R}^{m \times n}$ die Singulärwerte $\sigma_1 \geq \dots \geq \sigma_r > 0$. Dann gilt für $k < r$:

$$\min\{\|\mathbf{A} - \mathbf{B}\|_2 : \mathbf{B} \in \mathbb{R}^{m \times n}, \text{rang}(\mathbf{B}) \leq k\} = \sigma_{k+1}. \quad (7.26)$$

Beweis. Der Beweis hat zwei Teile; in Teil 1 konstruieren wir explizit eine Rang- k -Matrix \mathbf{A}_k , so dass $\|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}$ gilt, und in Teil 2 zeigen wir, dass es keine bessere Approximation geben kann.

i) Bezeichnen $\underline{u}_1, \dots, \underline{u}_m$ bzw. $\underline{v}_1, \dots, \underline{v}_n$ die Spalten der orthogonalen Faktoren \mathbf{U} bzw. \mathbf{V} der Singulärwertzerlegung (7.22) von \mathbf{A} . Setzen wir

$$\mathbf{U}_k = (\underline{u}_1, \dots, \underline{u}_k), \quad \Sigma_k = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_k & \\ & & & \end{pmatrix}, \quad \mathbf{V}_k = (\underline{v}_1, \dots, \underline{v}_k).$$

und

$$\mathbf{A}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T,$$

so gilt offenbar

$$\|\mathbf{A} - \mathbf{A}_k\|_2 = \left\| \mathbf{U}^T (\mathbf{A} - \mathbf{A}_k) \mathbf{V} \right\|_2 = \left\| \Sigma - \begin{pmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \right\|_2 = \sigma_{k+1}.$$

Da $\text{rang}(\mathbf{A}_k) = k$, haben wir also

$$\min\{\|\mathbf{A} - \mathbf{B}\|_2 : \mathbf{B} \in \mathbb{R}^{m \times n}, \text{rang}(\mathbf{B}) \leq k\} \leq \sigma_{k+1}. \quad (7.27)$$

ii) Damit Gleichheit in (7.27) gilt, muss aus $\text{rang}(\mathbf{B}) \leq k$ die Beziehung $\|\mathbf{A} - \mathbf{B}\|_2 \geq \sigma_{k+1}$ folgen. Sei \mathbf{B} eine solche Matrix, dann ist die Dimension von $\ker(\mathbf{B})$ mindestens $n - k$. Auf der anderen Seite ist die Dimension des Bildes von $\mathbf{V}_{k+1} = (\underline{v}_1, \dots, \underline{v}_k, \underline{v}_{k+1})$ gerade $k+1$. Da $n - k + k + 1 > n + 1$, gibt es einen Vektor $\underline{w} \in \mathbb{R}^n$ mit $\|\underline{w}\|_2 = 1$ und $\underline{w} \in \ker(\mathbf{B}) \cap \text{im}(\mathbf{V}_{k+1})$. Es gibt demnach ein $\underline{z} \in \mathbb{R}^{k+1}$ mit $\underline{w} = \mathbf{V}_{k+1}\underline{z}$ und es gilt

$$\begin{aligned} \|(\mathbf{A} - \mathbf{B})\mathbf{V}_{k+1}\underline{z}\|_2 &= \|\mathbf{A}\mathbf{V}_{k+1}\underline{z}\|_2 = \|\mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}_{k+1}\underline{z}\|_2 \\ &= \sqrt{\sigma_1^2 z_1^2 + \dots + \sigma_{k+1}^2 z_{k+1}^2} \\ &\geq \sqrt{\sigma_1^2 z_{k+1}^2 + \dots + \sigma_{k+1}^2 z_{k+1}^2} = \sigma_{k+1} \|\underline{z}\|_2 = \sigma_{k+1}. \end{aligned}$$

Dies zeigt die gewünschte Beziehung, da

$$\|\mathbf{A} - \mathbf{B}\|_2 = \max_{\|\underline{z}\|_2=1} \|(\mathbf{A} - \mathbf{B})\underline{z}\|_2 \geq \sigma_{k+1}.$$

\square

Man kann sich leicht überlegen, wie sich Satz 7.24 zur Komprimierung grosser Datenmengen einsetzen lässt: Ist σ_{k+1} genügend klein, so kann die Matrix \mathbf{A} – bei der mn Gleitpunktzahlen abgespeichert werden müssen – durch die im Beweis konstruierte Matrix \mathbf{A}_k ersetzt werden. Für \mathbf{A}_k müssen lediglich $k(m+n)$ Gleitpunktzahlen abgespeichert werden.

Bemerkung 7.25 Die Bestimmung des exakten Rangs einer Matrix ist numerisch fast nie möglich, da die durch Rundungsfehler gestörte Matrix $\hat{\mathbf{A}}$ fast immer vollen Rang haben wird! Als Ersatz betrachtet man den **numerischen Rang** einer Matrix \mathbf{A} . Zu gegebenem $\varepsilon > 0$ definiert man diesen als dasjenige k , so dass $\sigma_k \geq \varepsilon$ und $\sigma_{k+1} < \varepsilon$ erfüllt sind. Siehe auch **type rank** in MATLAB.

Kapitel 8

Iterative Lösung Linearer Gleichungssysteme

In der Praxis trifft man selten auf Probleme, die *unmittelbar* die Lösung eines linearen Gleichungssystems oder Ausgleichsproblems erfordern. Meist gelangt man erst mittels Diskretisierungen und allenfalls Linearisierungen zu diesen numerisch einfach behandelbaren Problemen. In Beispiel 6.19 haben wir diese Vorgehensweise bereits für die Lösung einer eindimensionalen, nichtlinearen Integralgleichung kennengelernt. Für zwei-, drei- und höherdimensionale Probleme werden die so erhaltenen Gleichungssysteme typischerweise sehr gross und verlieren darüber hinaus die in Beispiel 6.19 beobachtete Bandstruktur. Die in Kapitel 2 vorgestellten Verfahren stossen schnell an die Grenzen des verfügbaren Speicherplatzes. Für die Abspeicherung einer 100.000×100.000 -Matrix werden 150 Gigabyte benötigt! Hier müssen Verfahren eingesetzt werden, welche die Dünnbesetztheit der Matrix ausnutzen.

8.1 Speicherung dicht- und dünnbesetzter Matrizen

Bevor wir uns den Algorithmen zuwenden, diskutieren wir kurz wie eigentlich Matrizen im Rechner abgespeichert werden. Für Matrizen *ohne* nennenswerte Nullstruktur gibt es im wesentlichen nur die folgenden beiden Standardformate.

Column major: Matrix*spalten* werden hintereinander im Speicher abgelegt (Fortran, MATLAB, OpenGL).

Row major: Matrix*zeilen* werden hintereinander im Speicher abgelegt (C-arrays, bitmaps, Python).

Beispiel 8.1 (Row vs. column major)

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	Column major: A_arr 1 4 7 2 5 8 3 6 9	◇
	Row major: A_arr 1 2 3 4 5 6 7 8 9	

Da der Austausch von Daten zwischen den Speicherhierarchien moderner Rechner (Register, Cache, Hauptspeicher, Festplatte, Netzwerk) einen wesentlichen Anteil der Rechenzeit beansprucht, kann ein Algorithmus nur dann effizient sein, wenn Zugriffe auf weit entfernte Speicherplätze vermieden werden.²⁸

²⁸Siehe auch <http://en.wikipedia.org/wiki/Cache>.

Beispiel 8.2 Die folgenden beiden Algorithmen führen jeweils $n(n-1)$ flops durch. Der linke Algorithmus geht dabei spaltenweise und der rechte Algorithmus zeilenweise vor.

<pre style="margin: 0;">MATLAB A = randn(n); for j = 1:n-1, A(:,j+1) = A(:,j+1) - A(:,j); end</pre>	<pre style="margin: 0;">MATLAB A = randn(n); for i = 1:n-1, A(i+1,:) = A(i+1,:) - A(i,:); end</pre>
---	---

Die zeilenweise Vorgehensweise benötigt dabei auf einem Standard-PC ca. die 3-fache Rechenzeit. Ursache ist, dass in jeder inneren Schleife auf eine komplette Zeile zugegriffen wird, deren Einträge beim Format column major weit voneinander entfernt im Speicher liegen. ◇

Enthält eine Matrix $A \in \mathbb{R}^{n \times n}$ sehr viele Nullen, gilt also

$$\text{nnz}(A) := \#\{(i, j) : a_{ij} \neq 0\} \ll n^2,$$

so nennt man A **dünnbesetzt**. Für eine solche Matrix verschwenden die oben diskutierten Standardformate unnötig Speicher. Als Alternative bieten sich – je nach Anwendungshintergrund – verschiedene Speicherformate an, die lediglich die von Null verschiedenen Einträge von A speichern.

Ein verbreitetes Format ist dabei das **Compressed-Row-Storage-Format** (kurz: CRS-Format). Die Information zu den Einträgen von $A \in \mathbb{R}^{m \times n}$ wird in drei Arrays gespeichert:

```
real val      Grösse nnz(A)
int  col_ind  Grösse nnz(A)
int  row_ptr  Grösse n + 1   row_ptr[n + 1] = nnz(A) + 1
```

$\text{nnz}(A)$ = Anzahl der Nichtnullelemente (number of nonzeros)

Der Zugriff auf einen Matrixeintrag $a_{ij} \neq 0$, $1 \leq i, j \leq n$, erfolgt wie folgt:

$$\text{val}[k] = a_{ij} \Leftrightarrow \begin{cases} \text{col_ind}[k] = j, \\ \text{row_ptr}[i] \leq k < \text{row_ptr}[i + 1], \end{cases} \quad 1 \leq k \leq \text{nnz}(A).$$

Beispiel 8.3 (CRS-Matrixspeicherformat)

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix}$$

<i>val</i>	10	-2	3	9	3	7	8	7	3...9	13	4	2	-1
<i>col_ind</i>	1	5	1	2	6	2	3	4	1...5	6	2	5	6
<i>row_ptr</i>	1	3	6	9	13	17	20						

Grundbefehle zur Arbeit mit dünnbesetzten Matrizen in MATLAB:

◇

MATLAB

```
A = sparse(m,n); A = spalloc(m,n,nnz)
A = sparse(i,j,s,m,n);
A = spdiags(B,d,m,n); A = speye(n); A = spones(S);
```

Insbesondere bei der Initialisierung einer dünnbesetzten Matrix ist es wichtig, auf die Besonderheiten des Speicherformats Rücksicht zu nehmen. Eine ungünstige Implementierung kann hier leicht zur 100-fachen Laufzeit führen! Siehe <http://blogs.mathworks.com/loren/2007/03/01/creating-sparse-finite-element-matrices-in-matlab/>.

8.2 Allgemeines zu Splitting-Verfahren

Wir kommen jetzt wieder zurück auf ein lineares Gleichungssystem der Form

$$\mathbf{A}\underline{x} = \underline{b}; \quad (8.1)$$

mit $\mathbf{A} \in \mathbb{R}^{n \times n}$ invertierbar und $\underline{b} \in \mathbb{R}^n$.

Iterative Lösungsverfahren zur Lösung von (8.1) liefern, ausgehend von einem Startvektor $\underline{x}^{(0)} \in \mathbb{R}^n$ eine Folge $\{\underline{x}^{(k)}\}_{k=1}^{\infty}$ von Vektoren, die gegen die eindeutige Lösung \underline{x} von (2.1) konvergiert:

$$\underline{x} = \lim_{k \rightarrow \infty} \underline{x}^{(k)} \iff \|\underline{x} - \underline{x}^{(k)}\| \rightarrow 0 \quad \text{für } k \rightarrow \infty.$$

Diese Verfahren gehen von einer **Fixpunktform** des linearen Gleichungssystems (8.1) aus:

$$\underline{x} = \mathbf{B}\underline{x} + \underline{f} \quad (8.2)$$

Um (8.1) auf Fixpunktform zu bringen, spalten wir die Matrix \mathbf{A} auf:

$$\mathbf{A} = \mathbf{P} - \mathbf{N}, \quad \mathbf{P}, \mathbf{N} \in \mathbb{R}^{n \times n} \quad (8.3)$$

wobei die Matrix \mathbf{P} der sog. **Vorkonditionierer** (engl. *Preconditioner*) und die Matrix \mathbf{N} der restliche Anteil der Matrix \mathbf{A} ist. Mit der Aufspaltung (8.3) schreiben wir (8.1) um in

$$\mathbf{P}\underline{x} = \mathbf{N}\underline{x} + \underline{b}$$

und erhalten die Fixpunktform (8.2) mit der **Iterationsmatrix**

$$\mathbf{B} = \mathbf{P}^{-1}\mathbf{N}.$$

Insgesamt erhalten wir aus der Aufspaltung (8.3) die Fixpunktiteration

$$\mathbf{P}\underline{x}^{(k+1)} = \mathbf{N}\underline{x}^{(k)} + \underline{b} \iff \underline{x}^{(k+1)} = \mathbf{B}\underline{x}^{(k)} + \underline{f} \quad (8.4)$$

mit $\underline{f} = \mathbf{P}^{-1}\underline{b}$.

Die Aufspaltung (8.3) ist nicht eindeutig – jede Aufspaltung ergibt eine andere Iteration. Ausschlaggebend für die Wahl einer Aufspaltung sind folgende, sich oft widersprechende, Kriterien:

- einfache Invertierbarkeit der Vorkonditionierungsmatrix \mathbf{P} und
- schnelle Konvergenz der **Fixpunktiteration** (8.4).

Wir erinnern daran, dass die Fixpunktiteration (8.4) **konsistent** mit dem System (8.1) ist, falls die Fixpunktgleichung (8.2) genau eine Lösung hat, die (8.1) löst.

Wir untersuchen die Konvergenz der Fixpunktiteration (8.4). Hierfür subtrahieren wir (8.2) von (8.4); dies ergibt die *Fehlerrekurrenz*

$$\underline{e}^{(k+1)} = \mathbf{B}\underline{e}^{(k)}, \quad k = 0, 1, 2, \dots, \quad \text{wobei } \underline{e}^{(k)} := \underline{x}^{(k)} - \underline{x}. \quad (8.5)$$

Offensichtlich gilt $\underline{x}^{(k)} \rightarrow \underline{x}$ genau dann, wenn $\|\underline{e}^{(k)}\| = \|\underline{x}^{(k)} - \underline{x}\| \rightarrow 0$ gilt. Wegen

$$\underline{e}^{(k)} = \mathbf{B}\underline{e}^{(k-1)} = \mathbf{B}^2\underline{e}^{(k-2)} = \dots = \mathbf{B}^k\underline{e}^{(0)}$$

ist dies genau dann der Fall für beliebigen Startvektor $\underline{x}^{(0)} \in \mathbb{R}^n$, wenn $\mathbf{B}^k \rightarrow 0$ für $k \rightarrow \infty$. Nach Satz 2.22 ist dies genau dann der Fall, wenn der Spektralradius $\rho(\mathbf{B})$ kleiner als 1 ist.

Satz 8.4 Die Fixpunktiteration (8.4) konvergiert genau dann für alle Startvektoren $\underline{x}^{(0)} \in \mathbb{R}^n$, wenn $\rho(\mathbf{B}) < 1$ erfüllt ist.

Für die Iteration $\mathbf{P}\underline{x}^{(k+1)} = \mathbf{N}\underline{x}^{(k)} + \underline{f}$ gelten folgende allgemeine Konvergenzsätze, die wir ohne Beweis angeben. Zur Formulierung benötigen wir die folgende Klasse von Skalarprodukten.

Lemma 8.5 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Dann ist $(\underline{x}, \underline{y})_{\mathbf{A}} := \underline{y}^T \mathbf{A} \underline{x}$ genau dann Skalarprodukt auf \mathbb{R}^n , wenn \mathbf{A} symmetrisch positiv definit (SPD) ist.

Die durch $(\cdot, \cdot)_{\mathbf{A}}$ induzierte Vektornorm wird mit $\|\underline{x}\|_{\mathbf{A}} := \sqrt{(\underline{x}, \underline{x})_{\mathbf{A}}}$ bezeichnet.

Proposition 8.6 Sei $\mathbf{A} = \mathbf{P} - \mathbf{N}$ mit \mathbf{A}, \mathbf{P} SPD. Falls $2\mathbf{P} - \mathbf{A}$ positiv definit, konvergiert die Iteration für jedes $\underline{x}^{(0)}$ und für die Iterationsmatrix $\mathbf{B} = \mathbf{P}^{-1}\mathbf{N}$ gilt

$$\rho(\mathbf{B}) = \|\mathbf{B}\|_{\mathbf{A}} = \|\mathbf{B}\|_{\mathbf{P}} < 1.$$

Die Konvergenz ist monoton in der Norm $\|\cdot\|_{\mathbf{A}}$ und der Norm $\|\cdot\|_{\mathbf{P}}$:

$$\|\underline{e}^{(k+1)}\|_{\mathbf{P}} \leq \|\underline{e}^{(k)}\|_{\mathbf{P}}, \quad \|\underline{e}^{(k+1)}\|_{\mathbf{A}} \leq \|\underline{e}^{(k)}\|_{\mathbf{A}}.$$

8.3 Jacobi- und Gauss-Seidel Verfahren.

Wir geben die wichtigsten Beispiele für Splittings der Form $\mathbf{A} = \mathbf{P} - \mathbf{N}$ an.

Beispiel 8.7 (Jacobi-Iteration) Sei $n = 3$, und $\mathbf{A}\underline{x} = \underline{b}$ zu lösen. Wir schreiben

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}. \end{aligned}$$

Sei nun $\underline{x}^{(0)}$ gegeben, dann ist die Jacobi-Iteration:

$$x_i^{(k+1)} := \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right) / a_{ii}, \quad i = 1, \dots, n. \quad (8.6)$$

◇

Beispiel 8.8 (Gauss-Seidel Iteration) Beachte: (8.6) benutzt für $x_i^{(k+1)}$ nicht die neueste Information und der Speicher für die $\underline{x}^{(k)}$ ist $2n$. Besser ist es, die $x_j^{(k)}$, $j < i$, gleich zu ersetzen:

$$x_i^{(k+1)} := \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right) / a_{ii}. \quad (8.7)$$

◇

Wir schreiben (8.6) und (8.7) in Form von Matrix-Operationen. Sei dazu

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U} \quad (8.8)$$

wobei

$$\mathbf{D} = \begin{pmatrix} a_{11} & & 0 \\ & \ddots & \\ 0 & & a_{nn} \end{pmatrix}, \mathbf{L} = \begin{pmatrix} 0 & & & 0 \\ a_{21} & \ddots & & \\ & \ddots & \ddots & \\ a_{n1} & & a_{n,n-1} & 0 \end{pmatrix}, \mathbf{U} = \begin{pmatrix} 0 & a_{12} & & a_{1n} \\ & \ddots & & \vdots \\ & & \ddots & a_{n-1,n} \\ 0 & & & 0 \end{pmatrix}.$$

Jacobi in Fixpunktform: Aus (8.6) folgt

$$\begin{aligned} \mathbf{D}\underline{x}^{(k+1)} &= \underline{b} - \mathbf{L}\underline{x}^{(k)} - \mathbf{U}\underline{x}^{(k)} \iff \\ \underline{x}^{(k+1)} &= \mathbf{D}^{-1}(\underline{b} - \mathbf{L}\underline{x}^{(k)} - \mathbf{U}\underline{x}^{(k)}) \\ &= \underbrace{-\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})}_{\mathbf{B}^J} \underline{x}^{(k)} + \mathbf{D}^{-1}\underline{b} \\ &= \mathbf{B}^J \underline{x}^{(k)} + \underline{f}. \end{aligned} \quad (8.9)$$

Gauss-Seidel in Fixpunktform: Analog erhalten wir aus (8.7) das Verfahren

$$\begin{aligned} \underline{x}^{(k+1)} &= \mathbf{D}^{-1}(\underline{b} - \mathbf{L}\underline{x}^{(k+1)} - \mathbf{U}\underline{x}^{(k)}) \iff \\ (\mathbf{D} + \mathbf{L}) \underline{x}^{(k+1)} &= -\mathbf{U}\underline{x}^{(k)} + \underline{b} \iff \\ \underline{x}^{(k+1)} &= -(\mathbf{D} + \mathbf{L})^{-1} \mathbf{U}\underline{x}^{(k)} + (\mathbf{D} + \mathbf{L})^{-1}\underline{b} \iff \\ \underline{x}^{(k+1)} &= \mathbf{B}^{\text{GS}} \underline{x}^{(k)} + \underline{f} \end{aligned} \quad (8.10)$$

mit der Iterationsmatrix

$$\mathbf{B}^{\text{GS}} = -(\mathbf{D} + \mathbf{L})^{-1} \mathbf{U}, \quad \underline{f} = (\mathbf{D} + \mathbf{L})^{-1} \underline{b}.$$

Einen ersten Konvergenzsatz erhalten wir für diagonaldominante Matrizen \mathbf{A} . Wir erinnern an Definition 3.24.

Definition 8.9 $\mathbf{A} \in \mathbb{R}^{n \times n}$ heisst **strikt zeilendiagonaldominant**, falls

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1, \dots, n,$$

und **strikt spaltendiagonaldominant**, falls

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ji}|, \quad i = 1, \dots, n.$$

Satz 8.10 Sei \mathbf{A} strikt zeilen- bzw. spaltendiagonaldominant. Dann konvergieren sowohl das Jacobi- als auch das GS-Verfahren.

Beweis. Eine hinreichende Bedingung für $\rho(\mathbf{B}^J) < 1$ ist die Existenz einer induzierten Matrixnorm $\|\cdot\|_M$ mit $\|\mathbf{B}^J\|_M < 1$. Wir zeigen $\|\mathbf{B}^J\|_\infty < 1$. Sei dazu \mathbf{A} strikt zeilendiagonaldominant. Dann ist $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$, $i = 1 : n$, und

$$\|\mathbf{B}^J\|_\infty = \max_{i=1, \dots, n} \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| / |a_{ii}| < 1.$$

Für die Konvergenz des GS-Verfahrens bei Diagonaldominanz, siehe Iterative Solution Methods, O. Axelsson, CUP, 1994. □

Der folgende Satz zeigt, dass GS grundsätzlich immer für SPD-Matrizen einsetzbar ist.

Satz 8.11 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetrisch positiv definit. Dann konvergiert GS (8.10) für alle $\underline{x}^{(0)} \in \mathbb{R}^n$.

Beweis. $\mathbf{A} = \mathbf{A}^\top \implies \mathbf{U} = \mathbf{L}^\top$, also

$$\underline{x}^{(k+1)} = \mathbf{B}^{\text{GS}} \underline{x}^{(k)} + \underline{f}, \quad \mathbf{B}^{\text{GS}} = -(\mathbf{D} + \mathbf{L})^{-1} \mathbf{L}^\top.$$

Wir lassen im Rest des Beweises den Superskript "GS" weg und zeigen $\rho(\mathbf{B}) < 1$ in zwei Schritten.

- i) \mathbf{A} SPD $\implies \forall \underline{0} \neq \underline{x} \in \mathbb{R}^n: \underline{x}^\top \mathbf{A} \underline{x} > 0$.
 $\implies a_{ii} = \underline{e}_i^\top \mathbf{A} \underline{e}_i > 0 \implies \mathbf{D} > 0 \implies \mathbf{D}^{\pm 1/2} = \text{diag}(a_{ii}^{\pm 1/2})$ existiert.
- ii) Wegen $\mathbf{B}\underline{x} = \lambda \underline{x} \implies \mathbf{D}^{\frac{1}{2}} \mathbf{B} \mathbf{D}^{-\frac{1}{2}} \mathbf{D}^{\frac{1}{2}} \underline{x} = \lambda \mathbf{D}^{\frac{1}{2}} \underline{x}$ hat $\mathbf{B}_1 := \mathbf{D}^{\frac{1}{2}} \mathbf{B} \mathbf{D}^{-\frac{1}{2}}$ gleiche Eigenwerte wie \mathbf{B} , und es gilt

$$\rho(\mathbf{B}) < 1 \iff \rho(\mathbf{B}_1) < 1.$$

Es gilt weiter

$$\mathbf{B}_1 = -(\mathbf{I} + \mathbf{L}_1)^{-1} \mathbf{L}_1^\top \quad \text{mit} \quad \mathbf{L}_1 = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}.$$

Sei $\underline{x} \in \mathbb{C}^n$ Eigenvektor von \mathbf{B}_1 :

$$\mathbf{B}_1 \underline{x} = \lambda \underline{x}, \quad \underline{x}^H \underline{x} = 1.$$

Dann gilt:

$$-\mathbf{L}_1^\top \underline{x} = \lambda(\mathbf{I} + \mathbf{L}_1)\underline{x} \implies -\underline{x}^H \mathbf{L}_1^\top \underline{x} = \lambda(1 + \underline{x}^H \mathbf{L}_1 \underline{x}).$$

Sei $\underline{x}^H \mathbf{L}_1 \underline{x} = a + ib$, mit $a, b \in \mathbb{R}$.

$$\implies |\lambda|^2 = \left| \frac{-a + ib}{1 + a + ib} \right|^2 = \frac{a^2 + b^2}{1 + 2a + a^2 + b^2}.$$

$$\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} = \mathbf{I} + \mathbf{L}_1 + \mathbf{L}_1^\top \text{ pos. def. } \implies$$

$$0 < 1 + \underline{x}^H \mathbf{L}_1 \underline{x} + \underline{x}^H \mathbf{L}_1^\top \underline{x} = 1 + 2a \implies$$

$$|\lambda| < 1.$$

□

8.4 Relaxationsverfahren. JOR und SOR-Verfahren

Eine Verallgemeinerung des Jacobi-Verfahrens ist die sogenannte **Überrelaxationsmethode** (oder JOR von Jacobi Overrelaxation), bei der man zuerst die Näherung (8.9), die mit

$$\tilde{\underline{x}}_j^{(k+1)} = \mathbf{D}^{-1}(\underline{b} - \mathbf{L}\underline{x}^{(k)} - \mathbf{U}\underline{x}^{(k)})$$

bezeichnet sei, berechnet, und dann mit einem **Relaxationsparameter** $\omega > 0$ die neue Näherung $\underline{x}^{(k+1)}$ berechnet nach

$$\underline{x}^{(k+1)} = \omega \tilde{\underline{x}}_j^{(k+1)} + (1 - \omega) \underline{x}^{(k)} = \underline{x}^{(k)} + \omega (\tilde{\underline{x}}_j^{(k+1)} - \underline{x}^{(k)}).$$

Für $\omega = 0$ erhält man $\underline{x}^{(k+1)} = \underline{x}^{(k)}$, $\omega = 1$ ergibt offensichtlich das Jacobi-Verfahren; für $\omega > 1$ geht man weiter "in Richtung des Jacobischritts $\tilde{\underline{x}}_j^{(k+1)}$ ", und man spricht von **Überrelaxation**, für $\omega < 1$ resultiert die sogenannte **Unterrelaxation**. Es gilt

$$\underline{x}^{(k+1)} = \mathbf{B}_\omega^J \underline{x}^{(k)} + \underline{f}$$

mit

$$\mathbf{B}_\omega^J = \omega \mathbf{B}^J + (1 - \omega) \mathbf{I} \quad \text{und} \quad \mathbf{B}^J = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A},$$

Das JOR Verfahren lässt sich auch als iterative Verfeinerung

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \omega \mathbf{D}^{-1} \underline{r}^{(k)}, \quad \underline{r}^{(k)} = \underline{b} - \mathbf{A} \underline{x}^{(k)}$$

schreiben.

Satz 8.12 Für $\mathbf{A} \in \mathbb{R}^{n \times n}$ SPD konvergiert JOR für $0 < \omega < 2/\rho(\mathbf{D}^{-1} \mathbf{A})$.

Beweis. Übung. □

Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ SPD zerlegt wie in (8.8):

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{L}^\top. \tag{8.11}$$

Dann ist das Gauss-Seidel Verfahren für $\mathbf{A}\underline{x} = \underline{b}$ nach (8.10):

$$\underline{x}_{\text{GS}}^{(k+1)} = \mathbf{D}^{-1}(\underline{b} - \mathbf{L}\underline{x}^{(k+1)} - \mathbf{L}^\top \underline{x}^{(k)}).$$

Das **SOR(ω)-Verfahren** (Successive OverRelaxation) erhält man durch "dämpfen" ($\omega < 1$) bzw. "verlängern" ($\omega > 1$) eines GS Schrittes. Sei dazu wieder $\omega > 0$ ein sog. "Relaxations-Parameter". Dann gilt

$$\begin{aligned} \underline{x}_{\text{SOR}}^{(k+1)} &:= (1 - \omega) \underline{x}^{(k)} + \omega \underline{x}_{\text{GS}}^{(k+1)} = \underline{x}^{(k)} + \omega (\underline{x}_{\text{GS}}^{(k+1)} - \underline{x}^{(k)}) \\ &= (1 - \omega) \underline{x}^{(k)} + \omega \mathbf{D}^{-1}(\underline{b} - \mathbf{L}\underline{x}^{(k+1)} - \mathbf{L}^\top \underline{x}^{(k)}). \end{aligned}$$

Fixpunktform:

$$\underline{x}^{(k+1)} = \mathbf{B}^{\text{SOR}}(\omega) \underline{x}^{(k)} + \underline{f}(\omega), \tag{8.12}$$

wobei

$$\mathbf{B}^{\text{SOR}}(\omega) := (\mathbf{D} + \omega \mathbf{L})^{-1} [(1 - \omega) \mathbf{D} - \omega \mathbf{L}^\top], \tag{8.13}$$

$$\underline{f}(\omega) := \omega (\mathbf{D} + \omega \mathbf{L})^{-1} \underline{b}.$$

Satz 8.13 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ SPD. SOR(ω) konvergiert genau dann für alle $\underline{x}^{(0)} \in \mathbb{R}^n$, wenn

$$0 < \omega < 2.$$

Beweis. Sei λ Eigenwert von $\mathbf{B}(\omega)$ in (8.12). Dann existiert ein Eigenvektor $\underline{q} \neq \underline{z} \in \mathbb{C}^n$ mit $\mathbf{B}(\omega) \underline{z} = \lambda \underline{z}$, woraus mit (8.13) und $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{L}^\top = \mathbf{D} + \mathbf{L} + \mathbf{U}$ folgt

$$((1 - \omega) \mathbf{D} - \omega \mathbf{U}) \underline{z} = \lambda (\mathbf{D} + \omega \mathbf{L}) \underline{z}. \tag{8.14}$$

Es gilt

$$2(1 - \omega) \mathbf{D} - 2\omega \mathbf{U} = (2 - \omega) \mathbf{D} - \omega \mathbf{A} + \omega (\mathbf{L} - \mathbf{U}),$$

$$2(\mathbf{D} + \omega \mathbf{L}) = (2 - \omega) \mathbf{D} + \omega \mathbf{A} + \omega (\mathbf{L} - \mathbf{U}).$$

Einsetzen in (8.14) und multiplizieren von (8.14) von links mit $\underline{z}^H \implies$

$$\begin{aligned} (2 - \omega) \underline{z}^H \mathbf{D} \underline{z} - \omega \underline{z}^H \mathbf{A} \underline{z} - \omega \underline{z}^H (\mathbf{U}^\top - \mathbf{U}) \underline{z} \\ = \lambda [(2 - \omega) \underline{z}^H \mathbf{D} \underline{z} + \omega \underline{z}^H \mathbf{A} \underline{z} - \omega \underline{z}^H (\mathbf{U}^\top - \mathbf{U}) \underline{z}]. \end{aligned} \tag{8.15}$$

\mathbf{A} SPD, $\underline{z} \neq \underline{0}$ und $a_{ii} > 0 \implies d := \underline{z}^H \mathbf{D} \underline{z} > 0$, $a := \underline{z}^H \mathbf{A} \underline{z} > 0$. Weiter ist $(\mathbf{U}^\top - \mathbf{U})$ schiefsymmetrisch

$$\implies \underline{z}^H (\mathbf{U}^\top - \mathbf{U}) \underline{z} = -ir, \quad r \in \mathbb{R}.$$

Damit ist (8.15):

$$(2 - \omega)d - \omega a + i\omega r = \lambda [(2 - \omega)d + \omega a + i\omega r]$$

und es folgt

$$\lambda = \frac{(2 - \omega)d - \omega a + i\omega r}{(2 - \omega)d + \omega a + i\omega r}. \tag{8.16}$$

Nun ist $|(2 - \omega)d - \omega a| < |(2 - \omega)d + \omega a| \iff 0 < \omega < 2$, während die Imaginärteile von Zähler und Nenner in (8.16) gleich sind. Daher $|\lambda(\omega)| < 1$ für $0 < \omega < 2$ und

$$\rho(\mathbf{B}(\omega)) < 1,$$

da λ beliebiger Eigenwert von $\mathbf{B}(\omega)$ war. □

8.5 Richardson-Verfahren*

Alle bisher betrachteten Methoden liessen sich als Fixpunktiteration mit fester Iterationsmatrix \mathbf{B} schreiben. Dies sind sogenannte stationäre Verfahren. Alternativ dazu können wir auch instationäre Verfahren betrachten, bei denen $\mathbf{B} = \mathbf{B}_k$ ist. Das einfachste instationäre Verfahren ist das sogenannte **Richardson-Verfahren**. Wir leiten es wie folgt her:

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \alpha \mathbf{P}^{-1} \underline{r}^{(k)}, \quad (8.17)$$

wobei $\alpha > 0$ ein Relaxationsparameter und

$$\underline{r}^{(k)} = \underline{b} - \mathbf{A} \underline{x}^{(k)} \quad (8.18)$$

das Residuum von $\underline{x}^{(k)}$ ist. Die Iterationsmatrix lautet hier $\mathbf{B} = \mathbf{I} - \alpha \mathbf{P}^{-1} \mathbf{A}$, also in (8.17) stationär. Ist α von Schritt k abhängig, also $\alpha = \alpha_k$, dann erhalten wir die Iterationsmatrizen

$$\mathbf{B}_k = \mathbf{I} - \alpha_k \mathbf{P}^{-1} \mathbf{A}.$$

Beachte, dass das Jacobi- sowie das GS-Verfahren Spezialfälle des stationären Richardson-Verfahrens sind mit $\alpha = 1$, $\mathbf{P} = \mathbf{D}$ bzw. $\mathbf{P} = \mathbf{D} + \mathbf{L}$.

Satz 8.14 Seien \mathbf{P} nichtsingulär und $\lambda_i \in \mathbb{C}$ die Eigenwerte von $\mathbf{P}^{-1} \mathbf{A}$: $\lambda_i \in \sigma(\mathbf{P}^{-1} \mathbf{A})$. Dann konvergiert das stationäre Richardson-Verfahren (8.17) genau dann, wenn

$$\frac{2 \Re \lambda_i}{\alpha |\lambda_i|^2} > 1, \quad i = 1 : n, \quad (8.19)$$

gilt.

Beweis. Es gilt $\mathbf{B}_\alpha = \mathbf{I} - \alpha \mathbf{P}^{-1} \mathbf{A}$ und daher gilt

$$\rho(\mathbf{B}_\alpha) = \max\{|\lambda_i(\mathbf{B}_\alpha)| : i = 1, \dots, n\} = \max\{|1 - \alpha \lambda_i(\mathbf{P}^{-1} \mathbf{A})| : i = 1, \dots, n\} < 1$$

genau dann, wenn $|1 - \alpha \lambda_i| < 1$. Daraus folgt

$$(1 - \alpha \Re \lambda_i)^2 + (\alpha \Im \lambda_i)^2 < 1,$$

was (8.19) beweist. \square

Satz 8.15 Sei \mathbf{P} invertierbar und $\mathbf{P}^{-1} \mathbf{A}$ habe positive reelle Eigenwerte

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n > 0.$$

Dann konvergiert das stationäre Richardson-Verfahren (8.17) genau dann, wenn $0 < \alpha < 2/\lambda_1$.

Für $\alpha = \alpha_{\text{opt}} := 2/(\lambda_1 + \lambda_n)$, ist der Spektralradius von \mathbf{B} minimal, d.h.

$$\rho_{\text{opt}} = \min_{\alpha > 0} |\rho(\mathbf{B}_\alpha)| = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}. \quad (8.20)$$

Beweis. Die Eigenwerte der Matrix \mathbf{B}_α sind durch $\lambda_i(\mathbf{B}_\alpha) = 1 - \alpha \lambda_i$ gegeben, so dass (8.17) genau dann konvergiert, wenn $|\lambda_i(\mathbf{B}_\alpha)| < 1$ für $i = 1, \dots, n$ gilt, d.h. wenn $0 < \alpha < 2/\lambda_1$. Es folgt (siehe Fig. 4.1), dass $\rho(\mathbf{B}_\alpha)$ minimal für $1 - \alpha \lambda_n = \alpha \lambda_1 - 1$ wird, d.h. für $\alpha = 2/(\lambda_1 + \lambda_n)$, was den gewünschten Wert für α_{opt} liefert. Durch Substitution erhält man den gewünschten Wert von ρ_{opt} . \square

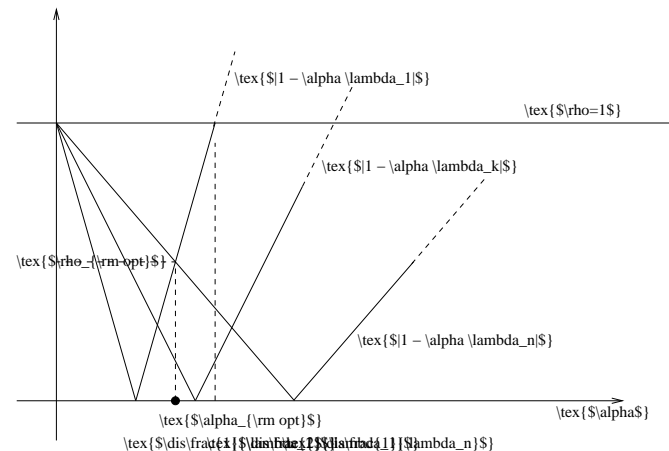


Abbildung 8.1. Fig. 3.1: Spektralradius von R_α als Funktion der Eigenwerte von $\mathbf{P}^{-1} \mathbf{A}$

Bemerkung 8.16 Für $\mathbf{P}^{-1} \mathbf{A}$ SPD gilt $\kappa_2(\mathbf{P}^{-1} \mathbf{A}) = \lambda_1/\lambda_n$. Damit folgt

$$\rho_{\text{opt}} = \frac{\lambda_1/\lambda_n - 1}{\lambda_1/\lambda_n + 1} = \frac{\kappa_2(\mathbf{P}^{-1} \mathbf{A}) - 1}{\kappa_2(\mathbf{P}^{-1} \mathbf{A}) + 1} \quad (8.21)$$

Somit hängt die Konvergenzgeschwindigkeit des Richardson Verfahrens **ausschliesslich** von $\text{cond}_2(\mathbf{P}^{-1} \mathbf{A})$ ab.

Die Konstruktion eines billigen Vorkonditionierers kann man z.B. mit der unvollständigen LU -Zerlegung (sogenannte “ILU-Belegung”) von \mathbf{A} erhalten (vgl. [QSS1], Kap. 4.3.2 für Details).

8.6 Das CG-Verfahren

Im folgenden werden wir das mit Abstand verbreitetste Verfahren zur Lösung eines linearen Gleichungssystems $\mathbf{A} \underline{x} = \underline{b}$ mit **symmetrisch positiv definit**er Matrix \mathbf{A} behandeln. Die Idee ist fundamental anders als bei den stationären Iterationsverfahren:

Wähle die beste Approximation $\tilde{\underline{x}}$ zu \underline{x} aus einem k -dimensionalen affinen Unterraum.

Diese Idee muss natürlich noch mit Leben gefüllt werden; unklar sind hier insbesondere in welchen Sinn “beste” gemeint ist und wie der affine Unterraum gewählt werden soll. Die Konstruktion des CG-Verfahrens ist anspruchsvoll; wir wollen sie deswegen in verschiedene Teilschritte zerlegen.

8.6.1 Minimierung in affinen Unterräumen

Definition 8.17 Eine Menge $V \subset \mathbb{R}^n$ heisst **affiner Unterraum**, wenn es einen Vektor $\underline{x}_0 \in \mathbb{R}^n$ und einen Unterraum $U \subset \mathbb{R}^n$ gibt, so dass $V = \underline{x}_0 + U = \{\underline{x}_0 + \underline{u} : \underline{u} \in U\}$.

Wir betrachten jetzt für einen gegebenen $(k+1)$ -dimensionalen Unterraum U eine ONB $\underline{q}_0, \underline{q}_1, \dots, \underline{q}_k$. Dann lässt sich die Lösung des Approximationsproblems

$$\min_{\underline{y} \in \underline{x}_0 + U} \|\underline{y} - \underline{x}\|_2$$

umschreiben als

$$\min_{\underline{w} \in \mathbb{R}^{k+1}} \|\underline{x}_0 + (\underline{q}_0, \underline{q}_1, \dots, \underline{q}_k) \underline{w} - \underline{x}\|_2.$$

Dies ist ein lineares Ausgleichsproblems; bezeichnen wir $\mathbf{Q}_k = (\underline{q}_0, \underline{q}_1, \dots, \underline{q}_k)$, so ergibt sich das eindeutige Minimum

$$\underline{w} = \mathbf{Q}_k^T (\underline{x}_0 - \underline{x}).$$

Zur Bestimmung von \underline{w} müssten wir also die exakte Lösung \underline{x} kennen!

Dieses Dilemma wird behoben, indem wir zu dem \mathbf{A} -Skalarprodukt übergehen:

$$(\underline{v}, \underline{w})_{\mathbf{A}} := \underline{w}^T \mathbf{A} \underline{v} \quad \implies \quad \|\underline{v}\|_{\mathbf{A}} := \sqrt{(\underline{v}, \underline{v})_{\mathbf{A}}}.$$

\mathbf{A} -Orthogonalität wird entsprechend als Orthogonalität bezüglich $(\cdot, \cdot)_{\mathbf{A}}$ verstanden.

Lemma 8.18 Sei \mathbf{A} SPD und \underline{x} Lösung von $\mathbf{A}\underline{x} = \underline{b}$. Sei weiterhin $\{\underline{p}_1, \dots, \underline{p}_k\}$ eine \mathbf{A} -orthogonale Basis eines Unterraums U und $\underline{r}_0 = \underline{b} - \mathbf{A}\underline{x}_0$, so ist

$$\underline{x}_k = \underline{x}_0 + \sum_{j=1}^k \frac{(\underline{p}_j, \underline{r}_0)}{(\underline{p}_j, \underline{p}_j)_{\mathbf{A}}} \underline{p}_j \quad (8.22)$$

die eindeutige Lösung des Minimierungsproblems

$$\|\underline{x}_k - \underline{x}\|_{\mathbf{A}} = \min_{\underline{y} \in \underline{x}_0 + U} \|\underline{y} - \underline{x}\|_{\mathbf{A}} \quad (8.23)$$

Beweis. Für \mathbf{A} gibt es eine eindeutige SPD-Matrix $\mathbf{A}^{1/2}$, so dass $\mathbf{A} = \mathbf{A}^{1/2} \mathbf{A}^{1/2}$ gilt (diese Wurzel kann zum Beispiel über die Spektralzerlegung von \mathbf{A} bestimmt werden). Damit lässt sich (8.23) auf ein Standard-Ausgleichsproblem zurückführen:

$$\min_{\underline{y} \in \underline{x}_0 + U} \|\underline{y} - \underline{x}\|_{\mathbf{A}} = \min_{\underline{y} \in \underline{x}_0 + U} \|\mathbf{A}^{1/2}(\underline{y} - \underline{x})\|_2$$

Setzen wir $\mathbf{P}_k = (\underline{p}_1, \dots, \underline{p}_k)$ so ergibt sich nämlich

$$\min_{\underline{y} \in \underline{x}_0 + U} \|\mathbf{A}^{1/2}(\underline{y} - \underline{x})\|_2 = \min_{\underline{w} \in \mathbb{R}^k} \|\mathbf{A}^{1/2}(\mathbf{P}_k \underline{w} + \underline{x}_0 - \underline{x})\|_2$$

Über die Normalengleichungen erhalten wir die Lösung

$$\underline{w} = (\mathbf{P}_k^T \mathbf{A} \mathbf{P}_k)^{-1} \mathbf{P}_k^T \mathbf{A} (\underline{x}_0 - \underline{x}), \quad (8.24)$$

Wegen der \mathbf{A} -Orthogonalität von \underline{p}_j ist $(\mathbf{P}_k^T \mathbf{A} \mathbf{P}_k)^{-1}$ eine Diagonalmatrix mit den Diagonaleinträgen $1/(\underline{p}_1, \underline{p}_1)_{\mathbf{A}}, \dots, 1/(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}$. Ausserdem haben wir²⁹

$$\mathbf{A}(\underline{x}_0 - \underline{x}) = \mathbf{A}\underline{x}_0 - \underline{b} = \underline{r}_0.$$

Somit folgt aus (8.24):

$$\underline{w} = \left(\frac{(\underline{p}_1, \underline{r}_0)}{(\underline{p}_1, \underline{p}_1)_{\mathbf{A}}}, \dots, \frac{(\underline{p}_k, \underline{r}_0)}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}} \right).$$

Wegen $\underline{x}_k = \underline{x}_0 + \mathbf{P}_k \underline{w}$ folgt (8.22). \square

Bemerkung 8.19 Für die in Lemma 8.18 gewonnene Lösung \underline{x}_k ist der Fehler $\underline{x}_k - \underline{x}$ \mathbf{A} -orthogonal zu U , d.h. $(\underline{x}_k - \underline{x}, \underline{u})_{\mathbf{A}} = 0$ für alle $\underline{u} \in U$.

Beweis. Für $\underline{u} \in U$ gibt es $\gamma_j \in \mathbb{R}$, so dass $\underline{u} = \gamma_1 \underline{p}_1 + \dots + \gamma_k \underline{p}_k$. Damit gilt für \underline{x}_k aus (8.22):

$$\begin{aligned} (\underline{x}_k - \underline{x}, \underline{u})_{\mathbf{A}} &= (\underline{x}_0 - \underline{x}, \underline{u})_{\mathbf{A}} - \sum_{j=1}^k \frac{(\underline{p}_j, \underline{r}_0)}{(\underline{p}_j, \underline{p}_j)_{\mathbf{A}}} (\underline{p}_j, \underline{u})_{\mathbf{A}} \\ &= (\underline{r}_0, \underline{u})_{\mathbf{A}} - \sum_{j=1}^k \frac{(\underline{p}_j, \underline{r}_0)}{(\underline{p}_j, \underline{p}_j)_{\mathbf{A}}} (\underline{p}_j, \underline{u})_{\mathbf{A}} \\ &= \sum_{j=1}^k \gamma_j (\underline{r}_0, \underline{p}_j)_{\mathbf{A}} - \sum_{j=1}^k \gamma_j (\underline{p}_j, \underline{r}_0)_{\mathbf{A}} = 0, \end{aligned}$$

wobei wir die Beziehung $(\underline{p}_j, \underline{u})_{\mathbf{A}} = \gamma_j (\underline{p}_j, \underline{p}_j)_{\mathbf{A}}$ benutzt haben. \square

Mit $\alpha_k = \frac{(\underline{p}_k, \underline{r}_0)}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}}$ ergeben sich aus (8.22) sofort die Rekursionen

$$\underline{x}_k = \underline{x}_{k-1} + \alpha_k \underline{p}_k, \quad \underline{r}_k = \underline{r}_{k-1} - \alpha_k \mathbf{A} \underline{p}_k, \quad (8.25)$$

wobei

$$\underline{r}_k := \underline{b} - \mathbf{A} \underline{x}_k = \mathbf{A}(\underline{x} - \underline{x}_k) = \mathbf{A}(\underline{x} - \underline{x}_{k-1} - \alpha_k \underline{p}_k) = \underline{r}_{k-1} - \alpha_k \mathbf{A} \underline{p}_k.$$

8.6.2 Konstruktion der \mathbf{A} -orthogonalen Basis

Wie wählen wir den Unterraum U günstig, dass (a) gute Approximation zu \underline{x} erzielt werden und (b) die Konstruktion einer \mathbf{A} -orthogonalen Basis möglichst preiswert ist? Den entscheidenden Hinweis liefert der Satz von Cayley-Hamilton.

Satz 8.20 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Dann gibt es ein Polynom p_l vom Grad höchstens $n-1$, so dass $\mathbf{A}^{-1} = p_l(\mathbf{A})$.

²⁹Dies ist der entscheidende Punkt, an dem wir uns dem unbekanntem \underline{x} entledigen.

Beweis. Hier nur eine Skizze für den Fall, dass \mathbf{A} symmetrisch positiv definit ist. Dann hat \mathbf{A} $l + 1 \leq n$ paarweise verschiedene reelle Eigenwerte $\lambda_1, \dots, \lambda_{l+1}$. Jetzt wählen wir das interpolierende Polynom p_l von $1/x$ an den Stützstellen $\lambda_1, \dots, \lambda_{l+1}$, es gilt also

$$p_l(\lambda_1) = 1/\lambda_1, \quad \dots, \quad p_l(\lambda_{l+1}) = 1/\lambda_{l+1}. \quad (8.26)$$

Dabei hat p_l Grad höchstens l (siehe Kapitel 3). Ausserdem folgt aus (8.26) die Beziehung $p_l(\mathbf{A}) = \mathbf{A}^{-1}$ (Nachweis z.B. über die Spektralzerlegung von \mathbf{A}). \square

Das Resultat impliziert

$$\underline{x} - \underline{x}_0 = \mathbf{A}^{-1} \underline{r}_0 = p_l(\mathbf{A}) \underline{r}_0 \in \text{span}\{\underline{r}_0, \mathbf{A} \underline{r}_0, \dots, \mathbf{A}^{n-1} \underline{r}_0\}.$$

Dies motiviert die Wahl des Unterraums

$$U_k = \text{span}\{\underline{r}_0, \mathbf{A} \underline{r}_0, \dots, \mathbf{A}^{k-1} \underline{r}_0\}.$$

Definition 8.21 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ und $\underline{r}_0 \in \mathbb{R}^n$. Dann heisst $\mathcal{K}_k(\mathbf{A}, \underline{r}_0) = \text{span}\{\underline{r}_0, \mathbf{A} \underline{r}_0, \dots, \mathbf{A}^{k-1} \underline{r}_0\}$ Krylov-Raum zu \mathbf{A} und \underline{b} .

Wie konstruieren wir nun eine \mathbf{A} -orthogonale Basis zu U_k ? Zunächst stellen wir fest, dass die Residuen $\underline{r}_j = \underline{b} - \mathbf{A} \underline{x}_j$ bereits eine orthogonale Basis für U_k bilden.

Lemma 8.22 Seien $\underline{x}_0, \underline{x}_1, \dots, \underline{x}_k$ wie in (8.25) definiert und $\underline{r}_k \neq 0$. Dann sind die Residuen $\underline{r}_0, \dots, \underline{r}_k$ paarweise orthogonal, d.h. $(\underline{r}_i, \underline{r}_j) = 0$ für $0 \leq i < j \leq k$. Ausserdem gilt $U_{k+1} = \text{span}\{\underline{r}_0, \dots, \underline{r}_k\}$.

Beweis. Der Beweis erfolgt durch vollständige Induktion über k . Der Fall $k = 0$ ist wegen $U_1 = \text{span}\{\underline{r}_0\}$ trivial. Sei die Behauptung für $k - 1$ richtig. Aus (8.25) folgt $\underline{r}_k \in U_{k+1}$. Aus Bemerkung 8.19 zu Lemma 8.18 folgt

$$0 = (\underline{x} - \underline{x}_k, \underline{u})_{\mathbf{A}} = (\mathbf{A}(\underline{x} - \underline{x}_k), \underline{u}) = (\underline{r}_k, \underline{u})$$

für alle $\underline{u} \in U_k$. Per Induktionsvoraussetzung formt $\{\underline{r}_0, \dots, \underline{r}_{k-1}\}$ eine Orthogonalbasis für U_k . Damit gilt also $(\underline{r}_i, \underline{r}_k) = 0$ für $0 \leq i < k$. Wegen $\underline{r}_k \neq 0$ formt dann $\{\underline{r}_0, \dots, \underline{r}_k\}$ eine Orthogonalbasis für U_{k+1} . \square

Wir konstruieren die \mathbf{A} -orthogonalen Basisvektoren \underline{p}_k wie folgt. Falls $\underline{r}_0 \neq 0$ (sonst ist \underline{x}_0 bereits Lösung) setzen wir $\underline{p}_1 = \underline{r}_0$. Lemma 8.22 sagt aus, dass entweder \underline{r}_k verschwindet oder aber die Vektoren $\underline{p}_1, \dots, \underline{p}_k$ (die als Induktionsvoraussetzung bereits eine \mathbf{A} -orthogonale Basis für U_k bilden sollen) und \underline{r}_k linear unabhängig sind und U_{k+1} aufspannen. Im unwahrscheinlichen ersten Fall ist $\underline{x} = \underline{x}_k$ und wir sind fertig. Im zweiten Fall wenden wir einen Schritt von Gram-Schmidt im \mathbf{A} -Skalarprodukt an, um aus \underline{r}_k einen zu U_k \mathbf{A} -orthogonalen Vektor \underline{p}_{k+1} zu gewinnen:

$$\underline{p}_{k+1} = \underline{r}_k - \sum_{j=1}^k \frac{(\underline{r}_k, \underline{p}_j)_{\mathbf{A}}}{(\underline{p}_j, \underline{p}_j)_{\mathbf{A}}} \underline{p}_j = \underline{r}_k - \frac{(\underline{r}_k, \underline{p}_k)_{\mathbf{A}}}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}} \underline{p}_k, \quad (8.27)$$

wobei wir in der zweiten Gleichung die folgende aus Rekursion (8.25) und Lemma 8.22 gewonnene Beziehung ausnutzen:

$$(\underline{r}_k, \underline{p}_j)_{\mathbf{A}} = (\underline{r}_k, \mathbf{A} \underline{p}_j) = \frac{1}{\alpha_j} (\underline{r}_k, \underline{r}_{j-1}) - \frac{1}{\alpha_j} (\underline{r}_k, \underline{r}_j) = 0 - 0 = 0, \quad \forall j < k.$$

Mit der Bezeichnung $\beta_{k+1} = -\frac{(\underline{r}_k, \underline{p}_k)_{\mathbf{A}}}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}}$ und (8.25) bzw. (8.27) erhalten wir also insgesamt die drei Rekursionen

$$\underline{x}_k = \underline{x}_{k-1} + \alpha_k \underline{p}_k, \quad \underline{r}_k = \underline{r}_{k-1} - \alpha_k \mathbf{A} \underline{p}_k, \quad \underline{p}_{k+1} = \underline{r}_k + \beta_{k+1} \underline{p}_k. \quad (8.28)$$

8.6.3 Der Algorithmus

Im Prinzip ist das CG-Verfahren (Verfahren der konjugierten Gradienten – Conjugate Gradients) durch die Rekursion (8.28) vollständig definiert. Der teuerste Anteil ist dabei in den allermeisten Fällen die Multiplikation mit der Matrix \mathbf{A} . Dabei stört uns noch, dass wir für die Berechnung der α_0 zusätzlich den Vektor \underline{r}_0 speichern müssen. Wir schreiben die Formeln im folgenden so um, dass wir nur die letzten iterierten \underline{r}_{k-1} , \underline{x}_{k-1} , sowie \underline{p}_k benötigen. Dazu nutzen wir die Beziehung³⁰

$$(\underline{p}_k, \underline{r}_0) = (\underline{p}_k, \underline{x} - \underline{x}_0)_{\mathbf{A}} = (\underline{p}_k, \underline{x} - \underline{x}_{k-1})_{\mathbf{A}} = (\underline{p}_k, \underline{r}_{k-1}) = (\underline{r}_{k-1}, \underline{r}_{k-1})$$

und erhalten

$$\alpha_k = \frac{(\underline{p}_k, \underline{r}_0)}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}} = \frac{(\underline{r}_{k-1}, \underline{r}_{k-1})}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}}. \quad (8.29)$$

Weiterhin gilt wegen

$$-\alpha_k (\underline{r}_k, \underline{p}_k)_{\mathbf{A}} = (\underline{r}_k, -\alpha_k \mathbf{A} \underline{p}_k) = (\underline{r}_k, \underline{r}_k - \underline{r}_{k-1}) = (\underline{r}_k, \underline{r}_k)$$

die Beziehung

$$\beta_{k+1} = -\frac{(\underline{r}_k, \underline{p}_k)_{\mathbf{A}}}{(\underline{p}_k, \underline{p}_k)_{\mathbf{A}}} = \frac{(\underline{r}_k, \underline{r}_k)}{\alpha_k (\underline{p}_k, \underline{p}_k)_{\mathbf{A}}} = \frac{(\underline{r}_k, \underline{r}_k)}{(\underline{r}_{k-1}, \underline{r}_{k-1})}.$$

Insgesamt ergibt sich der folgende Algorithmus nach Hestenes und Stiefel.

Algorithmus 8.23 CG-Verfahren

Input: SPD $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\underline{b} \in \mathbb{R}^n$. Startvektor $\underline{x}_0 \in \mathbb{R}^n$. $k_{\max} \in \mathbb{N}$.

Output: Approximation $\underline{x}_{k_{\max}}$ zur Lösung von $\mathbf{A} \underline{x} = \underline{b}$.

```


$\underline{p}_1 := \underline{r}_0 = \underline{b} - \mathbf{A} \underline{x}_0$   

for  $j = 1, \dots, k_{\max}$  do  

     $\alpha_k := (\underline{r}_{k-1}, \underline{r}_{k-1}) / (\underline{p}_k, \mathbf{A} \underline{p}_k)$   

     $\underline{x}_k := \underline{x}_{k-1} + \alpha_k \underline{p}_k$   

     $\underline{r}_k := \underline{r}_{k-1} - \alpha_k \mathbf{A} \underline{p}_k$   

     $\beta_{k+1} := (\underline{r}_k, \underline{r}_k) / (\underline{r}_{k-1}, \underline{r}_{k-1})$   

     $\underline{p}_{k+1} := \underline{r}_k + \beta_{k+1} \underline{p}_k$   

end for


```

Einige Anmerkungen zur Implementierung von Algorithmus 8.23:

1. Pro Schleife ist einmal die Matrix-Vektor-Multiplikation mit \mathbf{A} nötig. Um den Algorithmus so unabhängig wie möglich vom gewählten Format von \mathbf{A} zu halten, wird nicht \mathbf{A} selbst sondern eine Funktion, welche die Matrix-Vektor-Multiplikation zu einem gegebenen Vektor durchführt, übergeben.

³⁰Das Nachvollziehen dieser Argumentation eignet sich gut zum Überprüfen des Verständnis des CG-Algorithmus.

2. Algorithmus sollte so implementiert werden, dass nur 3 zusätzliche Vektoren der Länge n während der Ausführung gespeichert werden.
3. Algorithmus 8.23 führt zwei Gram-Schmidt-Prozesse durch; einmal im \mathbf{A} -Skalarprodukt zur Konstruktion von \underline{p}_k und einmal im Standardskalarprodukt zur Konstruktion von \underline{r}_k . Er teilt damit die in Abschnitt 0.8.2 besprochenen numerischen Probleme des Gram-Schmidt-Algorithmus, d.h. die Orthogonalität und \mathbf{A} -Orthogonalität kann wesentlich durch Rundungsfehler beeinträchtigt werden. Dieser Effekt tritt tatsächlich sehr häufig ein; sobald der Krylov-Unterraum gute Approximationen zu den Eigenvektoren von \mathbf{A} enthält, gehen die Orthogonalitäten numerisch *komplett* verloren. Die Reputation von CG wurde durch die Doktorarbeit von Paige gerettet – dort wurde gezeigt, dass die Konvergenz des CG-Verfahrens nur zu einem gewissen, begrenzten Masse durch den Verlust der Orthogonalitäten beeinträchtigt wird. Insbesondere bleibt die unten angegebene Konvergenzschranke erhalten.

Die Konvergenz des CG-Verfahrens hängt wesentlich von der Konditionszahl von \mathbf{A} ab.

Satz 8.24 Für \underline{x}_k aus dem CG-Algorithmus 8.23 gilt

$$\|\underline{x} - \underline{x}_k\|_{\mathbf{A}} \leq \frac{2c^k}{1+c^{2k}} \|\underline{x} - \underline{x}_0\|_{\mathbf{A}}$$

mit

$$c = \frac{\sqrt{\kappa_2(\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{A})} + 1} < 1.$$

Beweis. Siehe Buch *Numerische Mathematik I* von Deuffhard und Hohmann. \square

Korollar 8.25 Um eine Reduktion des Fehlers in der \mathbf{A} -norm um einen Faktor $\epsilon > 0$ zu erreichen, also um $\|\underline{x} - \underline{x}_k\|_{\mathbf{A}} \leq \epsilon \|\underline{x} - \underline{x}_0\|_{\mathbf{A}}$ zu erfüllen, müssen maximal $k \geq \frac{1}{2} \sqrt{\kappa_2(\mathbf{A})} \log(2/\epsilon)$ Iterationen des CG-Algorithmus durchgeführt werden.

Beweis. Aus Satz 8.24 erhalten wir als hinreichende Bedingung für $\|\underline{x} - \underline{x}_k\|_{\mathbf{A}} \leq \epsilon \|\underline{x} - \underline{x}_0\|_{\mathbf{A}}$ die Ungleichungen

$$\epsilon \leq \frac{2c^k}{1+c^{2k}} \leq 2c^k \Leftrightarrow c^{-k} \leq \frac{2}{\epsilon} \Leftrightarrow k \geq \frac{\log(2/\epsilon)}{\log(1/c)}.$$

Die leicht zu beweisende Ungleichung

$$\log\left(\frac{a+1}{a-1}\right) > \frac{2}{a}, \quad \forall a > 1,$$

führt mit $a = \sqrt{\kappa_2(\mathbf{A})}$ auf die Ungleichung

$$k \geq \frac{\log(2/\epsilon)}{\log(1/c)} \geq \frac{\log(2/\epsilon)}{2/\sqrt{\kappa_2(\mathbf{A})}}.$$

\square

Alle Konvergenzaussagen beziehen sich auf die \mathbf{A} -Norm, die am natürlichsten im Zusammenhang mit dem CG-Algorithmus ist. Um zur Euklidischen Norm überzugehen, benutzen wir die Normäquivalenz

$$\frac{1}{\sqrt{\|\mathbf{A}^{-1}\|_2}} \|\underline{x} - \underline{x}_k\|_2 \leq \|\underline{x} - \underline{x}_k\|_{\mathbf{A}} \leq \sqrt{\|\mathbf{A}\|_2} \|\underline{x} - \underline{x}_k\|_2.$$

Man beachte, dass der Fehler zwar in der \mathbf{A} -Norm aber *nicht* in der Euklidischen Norm monoton fällt.

8.6.4 Geometrische Interpretation*

Es gibt eine alternative Herleitung des CG-Verfahrens, die insbesondere den Namen “konjugierte Gradienten” rechtfertigt. Wir wollen diese hier nur kurz andeuten, für eine ausführlichere Darstellung sei die Arbeit *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain* von Shewchuk empfohlen.

Die Lösung des LGS $\mathbf{A}\underline{x} = \underline{b}$ ist charakterisiert durch das Minimum der quadratischen Form

$$\Phi(\underline{y}) = \frac{1}{2} \underline{y}^T \mathbf{A} \underline{y} - \underline{y}^T \underline{b}. \quad (8.30)$$

Wegen \mathbf{A} SPD ist $\Phi(\cdot)$ strikt konvex und damit hat $\Phi(\cdot)$ nur ein lokales Minimum \underline{x} , für das gilt

$$(\nabla \Phi)(\underline{x}) = \mathbf{A}\underline{x} - \underline{b} \stackrel{!}{=} \underline{0}.$$

Das Gradientenverfahren erhält man, indem man ausgehend von der alten Iterierten \underline{x}_k in Richtung des steilsten Abstiegs

$$-(\nabla \Phi)(\underline{x}_k) = \underline{b} - \mathbf{A}\underline{x}_k = \underline{r}_k$$

geht. Man bildet also $\underline{x}_{k+1} = \underline{x}_k + \alpha_{k+1} \underline{r}_k$ mit der **Schrittlänge** α_{k+1} . Die optimale Schrittlänge erhält man durch die Lösung des eindimensionalen Minimierungsproblems

$$\Phi_k(\alpha_{k+1}) = \Phi(\underline{x}_k + \alpha_{k+1} \underline{r}_k) = \min!$$

Dessen Lösung ist gerade

$$\alpha_{k+1} = \frac{(\underline{r}_k, \underline{r}_k)}{(\underline{r}_k, \mathbf{A}\underline{r}_k)}.$$

Zusammen ergibt sich das **Verfahren des steilsten Abstiegs** oder Gradientenverfahren (engl: steepest descent method). Das oben besprochene Verfahren der konjugierten Gradienten erhält man, wenn die Richtung \underline{r}_k so gewählt wird, dass sie orthogonal zu allen vorhergehenden Richtungen ist.

8.6.5 Vorkonditionierung*

In Anwendungen ist die Konditionszahl von \mathbf{A} oft hoch³¹. Ein grosses Teilgebiet der Numerischen Mathematik beschäftigt sich mit der Konstruktion von sogenannten **Vorkonditionierern**; dies sind Matrizen \mathbf{P} die einerseits *leicht invertierbar* sind aber andererseits auch zu einer *niedrigen Konditionszahl* $\kappa_2(\mathbf{P}^{-1}\mathbf{A})$ führen.

³¹Ist \mathbf{A} die Diskretisierung eines unbeschränkten Operators, so geht zwangsläufig die Konditionszahl gegen ∞ je besser der Operator approximiert wird.

Wie kann der CG-Algorithmus von Vorkonditioniern profitieren? Wir könnten statt $\mathbf{Ax} = \mathbf{b}$ das äquivalente Gleichungssystem

$$\mathbf{P}^{-1}\mathbf{Ax} = \mathbf{P}^{-1}\mathbf{b}$$

lösen. Dann ist zwar $\kappa_2(\mathbf{P}^{-1}\mathbf{A})$ niedrig, aber $\mathbf{P}^{-1}\mathbf{A}$ ist *nicht* symmetrisch, selbst wenn \mathbf{P} und \mathbf{A} beide SPD sind. Als Ausweg benutzen wir deswegen (vorläufig) die Cholesky-Zerlegung $\mathbf{P} = \mathbf{LL}^T$ für SPD \mathbf{P} und betrachtet das zu $\mathbf{Ax} = \mathbf{b}$ äquivalente Gleichungssystem

$$\mathbf{L}^{-1}\mathbf{AL}^{-T}\hat{\mathbf{x}} = \mathbf{L}^{-1}\mathbf{b}, \quad \hat{\mathbf{x}} = \mathbf{L}^T\mathbf{x}. \quad (8.31)$$

Jetzt können wir CG auf (8.31) anwenden, da $\mathbf{L}^{-1}\mathbf{AL}^{-T}$ genau dann SPD ist, wenn \mathbf{A} SPD ist (Beweis Übung). Die Iterierten des CG angewendet auf (8.31) bezeichnen wir mit $\hat{\mathbf{x}}_k$. Die Vektoren $\hat{\mathbf{x}}_k$ approximieren die Lösung des vorkonditionierten Systems (8.31), dementsprechend approximieren die Vektoren $\mathbf{x}_k := \mathbf{L}^{-T}\hat{\mathbf{x}}_k$ die Lösung \mathbf{x} des originalen Systems. Die drei den CG-Algorithmus bestimmenden Rekursionen (8.28) ersetzen wir durch die äquivalenten Rekursionen (Nachrechnen!)

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \hat{\mathbf{p}}_k, \quad \hat{\mathbf{r}}_k = \hat{\mathbf{r}}_{k-1} - \alpha_k \mathbf{A}\hat{\mathbf{p}}_k, \quad \hat{\mathbf{p}}_{k+1} = \mathbf{P}^{-1}\hat{\mathbf{r}}_k + \beta_{k+1}\hat{\mathbf{p}}_k. \quad (8.32)$$

mit den Bezeichnungen

$$\hat{\mathbf{p}}_k := \mathbf{L}^{-T}\mathbf{p}_k, \quad \hat{\mathbf{r}}_k := \mathbf{L}\mathbf{r}_k.$$

Für die Koeffizienten α_k und β_{k+1} des CG angewendet auf (8.31) ergeben sich die Beziehungen

$$\begin{aligned} \alpha_k &= \frac{(\mathbf{r}_{k-1}, \mathbf{r}_{k-1})}{(\mathbf{p}_k, \mathbf{p}_k)_{\mathbf{L}^{-1}\mathbf{AL}^{-T}}} = \frac{(\mathbf{L}^{-1}\hat{\mathbf{r}}_{k-1}, \mathbf{L}^{-1}\hat{\mathbf{r}}_{k-1})}{(\mathbf{p}_k, \mathbf{L}^{-1}\mathbf{AL}^{-T}\mathbf{p}_k)} \\ &= \frac{(\hat{\mathbf{r}}_{k-1}, \mathbf{P}^{-1}\hat{\mathbf{r}}_{k-1})}{(\mathbf{L}^{-T}\mathbf{p}_k, \mathbf{AL}^{-T}\mathbf{p}_k)} = \frac{(\hat{\mathbf{r}}_{k-1}, \mathbf{P}^{-1}\hat{\mathbf{r}}_{k-1})}{(\hat{\mathbf{p}}_k, \hat{\mathbf{p}}_k)\mathbf{A}}, \\ \beta_{k+1} &= \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{r}_{k-1}, \mathbf{r}_{k-1})} = \frac{(\hat{\mathbf{r}}_k, \mathbf{P}^{-1}\hat{\mathbf{r}}_k)}{(\hat{\mathbf{r}}_{k-1}, \mathbf{P}^{-1}\hat{\mathbf{r}}_{k-1})}. \end{aligned}$$

Jetzt haben wir alle Größen so umformuliert, dass der Cholesky-Faktor \mathbf{L} nicht mehr vorkommt und wirklich nur \mathbf{P}^{-1} eine Rolle spielt. Insgesamt ergibt sich also der folgende Algorithmus (wobei wir den \mathbf{r}_k und \mathbf{p}_k die Hüte der Übersichtlichkeit halber wieder weggenommen haben).

Algorithmus 8.26 Vorkonditioniertes CG-Verfahren

Input: SPD \mathbf{A} , $\mathbf{P} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$. Startvektor $\mathbf{x}_0 \in \mathbb{R}^n$. $k_{\max} \in \mathbb{N}$.

Output: Approximation $\mathbf{x}_{k_{\max}}$ zur Lösung von $\mathbf{Ax} = \mathbf{b}$.

```

 $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$ ,  $\mathbf{p}_1 := \mathbf{P}^{-1}\mathbf{r}_0$ 
for  $j = 1, \dots, k_{\max}$  do
   $\alpha_k := (\mathbf{r}_{k-1}, \mathbf{P}^{-1}\mathbf{r}_{k-1}) / (\mathbf{p}_k, \mathbf{Ap}_k)$ 
   $\mathbf{x}_k := \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k$ 
   $\mathbf{r}_k := \mathbf{r}_{k-1} - \alpha_k \mathbf{Ap}_k$ 
   $\beta_{k+1} := (\mathbf{r}_k, \mathbf{P}^{-1}\mathbf{r}_k) / (\mathbf{r}_{k-1}, \mathbf{P}^{-1}\mathbf{r}_{k-1})$ 
   $\mathbf{p}_{k+1} := \mathbf{P}^{-1}\mathbf{r}_k + \beta_{k+1}\mathbf{p}_k$ 
end for

```

Es gibt zwei Extremfälle von Algorithmus 8.26: mit $\mathbf{P} = \mathbf{I}$ haben wir Standard-CG; und mit $\mathbf{P} = \mathbf{A}$ gilt $\mathbf{r}_1 = \mathbf{0}$ (vgl. mit iterativer Verbesserung) und Algorithmus 8.26 bricht nach einer Iteration (erfolgreich) ab. Der Nutzen von Algorithmus 8.26 liegt zwischen diesen beiden Extremen. Als Vorkonditionierer sollte man immer zuerst $\mathbf{P} = \text{diag}(\mathbf{A})$ ausprobieren. "Intelligenter" Vorkonditionierer erfordern mehr Wissen über das zugrundeliegende Problem und sind eine Kunst für sich, siehe auch Abschnitt 8.7 unten.

Korollar 8.27 Für \mathbf{x}_k aus dem vorkonditionierten CG-Algorithmus 8.26 gilt

$$\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}} \leq \frac{2c^k}{1 + c^{2k}} \|\mathbf{x} - \mathbf{x}_0\|_{\mathbf{A}}$$

mit

$$c = \frac{\sqrt{\kappa_2(\mathbf{P}^{-1}\mathbf{A})} - 1}{\sqrt{\kappa_2(\mathbf{P}^{-1}\mathbf{A})} + 1} < 1.$$

Beweis. Wie oben hergeleitet, löst das vorkonditionierte CG-Verfahren eigentlich das Gleichungssystem

$$\mathbf{L}^{-1}\mathbf{AL}^{-T}\hat{\mathbf{x}} = \mathbf{L}^{-1}\mathbf{b}$$

Aus Satz 8.24 folgt also

$$\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_k\|_{\mathbf{L}^{-1}\mathbf{AL}^{-T}} \leq \frac{2c^k}{1 + c^{2k}} \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_0\|_{\mathbf{L}^{-1}\mathbf{AL}^{-T}}$$

mit

$$c = \frac{\sqrt{\kappa_2(\mathbf{L}^{-1}\mathbf{AL}^{-T})} - 1}{\sqrt{\kappa_2(\mathbf{L}^{-1}\mathbf{AL}^{-T})} + 1}.$$

Mit $\kappa_2(\mathbf{L}^{-1}\mathbf{AL}^{-T}) = \kappa_2(\mathbf{P}^{-1}\mathbf{A})$ (Beweis Übung) und

$$\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_k\|_{\mathbf{L}^{-1}\mathbf{AL}^{-T}} = (\hat{\mathbf{x}} - \hat{\mathbf{x}}_k, \mathbf{L}^{-1}\mathbf{Ax} - \mathbf{L}^{-1}\mathbf{Ax}_k) = (\mathbf{x} - \mathbf{x}_k, \mathbf{Ax} - \mathbf{Ax}_k) = \|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{A}}$$

ergibt sich die Behauptung. \square

8.7 Numerisches Beispiel

Die vorgestellten Verfahren sollen an einem praktisch relevanten Beispiel illustriert werden; der "Mutter" aller partiellen Differentialgleichungen. Gesucht ist eine Funktion $u : \Omega \rightarrow \mathbb{R}$ auf dem Einheitsquadrat $\Omega = \{(x, y) \in \mathbb{R}^2 : x, y \in [0, 1]\}$, so dass

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f & \text{in } \Omega \\ u = 0 & \text{auf } \partial\Omega, \end{cases} \quad (8.33)$$

wobei $\partial\Omega = \{(x, y) \in \mathbb{R}^2 : x, y \in [0, 1]; x(x-1)y(y-1) = 0\}$ den Rand des Gebietes Ω bezeichnet. Die Gleichungen (8.33) werden als **Poisson-Problem** (mit Dirichlet-Randbedingungen) bezeichnet.

Zur numerischen Lösung von (8.33) verwenden wir die folgende finite Differenzen Approximation der zweiten Ableitung.

Lemma 8.28 Sei $v : \mathbb{R} \rightarrow \mathbb{R}$ zwei Mal stetig um x_0 differenzierbar. Dann gilt

$$v''(x_0) = \frac{v(x_0 - h) - 2v(x_0) + v(x_0 + h))}{h^2} + O(h^3)$$

Beweis. Wir verwenden Taylor-Entwicklung von v um x_0 :

$$v(x_0 \pm h) = v(x_0) \pm hv'(x_0) + \frac{h^2}{2}v''(x_0) \pm \frac{h^3}{6}v'''(x_0) + O(h^4).$$

Daraus ergibt sich

$$v(x_0 - h) + v(x_0 + h) = 2v(x_0) + h^2v''(x_0) + O(h^4),$$

und damit das gewünschte Ergebnis. \square

Wir diskretisieren das Gebiet Ω mit einem Gitter der Maschenweite $h = 1/N$:

$$\Omega_h = \{(x_i, y_j) : x_i = ih, y_j = jh; i, j = 0, \dots, N\},$$

siehe auch die Abbildung rechts. Jetzt betrachten wir das Poisson-Problem (8.33) nur auf den Gitterpunkten und ersetzen die zweiten Ableitungen durch die Approximation aus Lemma (8.28). Bezeichnen wir $u_{ij} = u(x_i, y_j)$ und $f_{ij} = f(x_i, y_j)$ so ergeben sich die Gleichungen

$$\frac{-u_{i-1,j} - u_{i+1,j} + 4u_{i,j} - u_{i,j-1} - u_{i,j+1}}{h^2} = f_{ij}$$

für $i, j = 1, \dots, N-1$. In der Nähe der Ränder nutzen wir, dass $u_{0,j} = u_{N,j} = u_{i,0} = u_{i,N} = 0$ gilt. Setzen wir

$$\underline{u}_h = (u_{11}, \dots, u_{1,N-1}, u_{21}, \dots, u_{2,N-1} \dots u_{N-1,1}, \dots, u_{N-1,N-1})^\top,$$

$$\underline{f}_h = (f_{11}, \dots, f_{1,N-1}, f_{21}, \dots, f_{2,N-1} \dots f_{N-1,1}, \dots, f_{N-1,N-1})^\top,$$

so ergibt sich das Gleichungssystem

$$\mathbf{A}_h \underline{u}_h = h^2 \underline{f}_h \quad (8.34)$$

mit der $(N-1)^2 \times (N-1)^2$ Matrix

$$\mathbf{A}_h = \begin{pmatrix} \mathbf{T}_h & -\mathbf{I}_{N-1} & & & \\ -\mathbf{I}_{N-1} & \mathbf{T}_h & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -\mathbf{I}_{N-1} & \mathbf{T}_h \\ & & & & & & -\mathbf{I}_{N-1} & \mathbf{T}_h \end{pmatrix}, \quad \mathbf{T}_h = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & & & \\ & & \ddots & & \\ & & & \ddots & -1 \\ & & & & -1 & 4 \end{pmatrix}. \quad (8.35)$$

Man kann zeigen, dass \mathbf{A}_h SPD ist, allerdings wächst die Konditionszahl je kleiner h (oder je grösser N) wird. Die Matrix \mathbf{A}_h kann in MATLAB mit dem Befehl `A = gallery('poisson',N-1)` erzeugt werden.

Zu Testzwecken betrachten wir das Poisson-Problem mit der rechten Seite

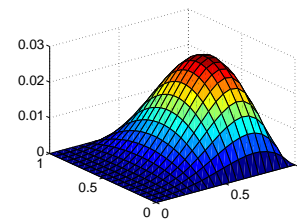
$$f(x, y) = -(12x^2 - 6x)y(y-1) - 2x^3(x-1). \quad (8.36)$$

Diese wurde so konstruiert, dass die Lösung von (8.33) gerade die Funktion

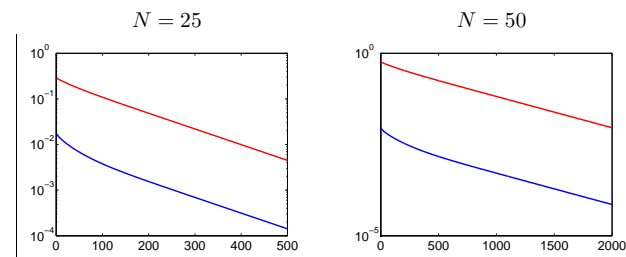
$$u(x, y) = x^3(x-1)y(y-1) \quad (8.37)$$

ist, siehe auch die Abbildung rechts.

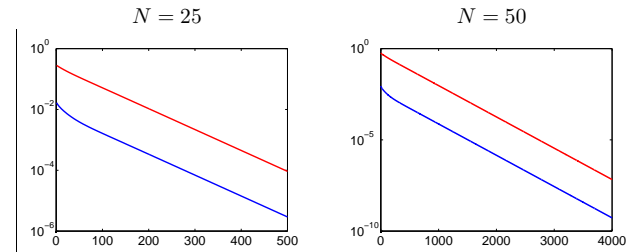
Wir wenden nun die in diesem Kapitel behandelten Verfahren auf das lineare Gleichungssystem (8.34) mit der aus (8.36) gewonnenen rechten Seite an.



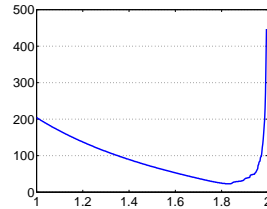
Jacobi-Iteration. Die in Abschnitt 8.3 beschriebene Jacobi-Iteration konvergiert sehr langsam; für $N = 50$ (2401×2401 Gleichungssystem) beträgt der Fehler $\|\underline{u}_k - \underline{x}_k\|_2$ nach 4000 (!) Iterationen ca. 1.8×10^{-4} . Die folgenden Abbildungen zeigen den Verlauf von $\|\underline{u}_k - \underline{x}_k\|_2$ (rote Kurve) und $\|h^2 \underline{f}_k - \mathbf{A}_h \underline{u}_k\|_2$ (blaue Kurve).



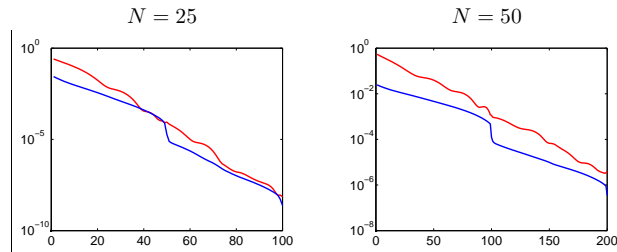
Gauss-Seidel-Iteration. Wiederholt man das Experiment mit der Gauss-Seidel-Iteration zeigt sich eine Verbesserung, wenn auch keine fundamentale. Nun beträgt der Fehler $\|\underline{u}_k - \underline{x}_k\|_2$ für $N = 50$ "bereits" nach 2000 Iterationen ca. 1.8×10^{-4} .



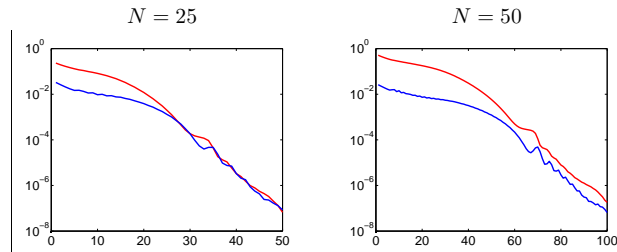
SOR-Relaxationsverfahren. Der kritischste Punkt bei dem in Abschnitt 8.4 behandelten SOR ist die Wahl des Relaxationsparameters ω . Die folgende Abbildung zeigt für $N = 25$ die nötige Anzahl von Iterationen, um $\|\underline{u}_k - \underline{x}_k\|_2 \leq 10^{-2}$ zu erreichen, im Verhältnis zu ω .



Für die fast optimalen Wahlen $\omega = 1.83$ ($N = 25$) und $\omega = 1.94$ ($N = 50$) ergeben sich die folgenden Konvergenzverläufe.



CG-Algorithmus. Das in Abschnitt (8.6) behandelte CG-Verfahren konvergiert sogar noch schneller als SOR (und hängt nicht von einer Parameterwahl ab).



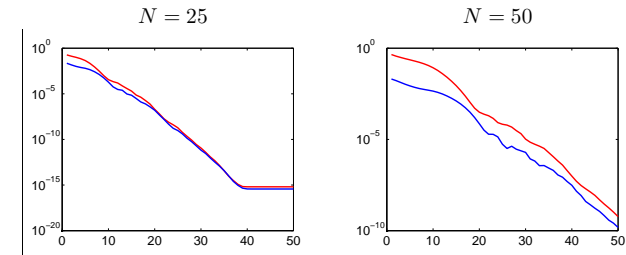
Wie in Tabelle 8.1 erkenntlich, verdoppelt sich aber auch hier die Anzahl der für einen gewissen Fehler notwendigen Iterationen, wenn h halbiert wird.

Vorkonditionierter CG-Algorithmus. Die Anzahl der benötigten Iterationen des CG-Algorithmus können mittels Vorkonditionierung noch weiter verringert werden. Wählen wir allerdings einfach $\mathbf{P} = \text{diag}(\mathbf{A})$ so verändert sich das Konvergenzverhalten der Iteration überhaupt nicht; mit $\mathbf{P} = \text{diag}(\mathbf{T}_h, \dots, \mathbf{T}_h)$ (in MATLAB: `P = spdiags(spdiags(A, [-1 0 -1]), [-1 0 1], n, n);`) sehen wir zumindest eine geringe Verbesserung (siehe Tabelle 8.1). Das Problem ist hier, dass die äusseren Diagonalen von \mathbf{A} in die Rechnung überhaupt nicht mit einbezogen werden. Die **unvollständige Cholesky-Zerlegung** (in MATLAB: `cholinc(A, '0')`) bietet einen kostengünstigen Ausweg. Dabei wird die Cholesky-Zerlegung von \mathbf{A} ganz normal durchgeführt, aber es werden keine Einträge des unvollständigen Cholesky-Faktors

	$N = 25$	$N = 50$
Jacobi	400	1952
Gauss-Seidel	204	985
SOR	23	77
CG	21	48
Vorkond. CG, $\mathbf{P} = \text{diag}(\mathbf{A})$	21	48
Vorkond. CG, $\mathbf{P} = \text{diag}(\mathbf{T}_h, \dots, \mathbf{T}_h)$	17	37
Vorkond. CG, $\mathbf{P} = \text{cholinc}$	7	15

Tabelle 8.1. Mindestanzahl von Iterationen um $\|\underline{x}_k - \underline{u}_k\|_2 \leq 10^{-2}$ zu erreichen.

$\tilde{\mathbf{R}}$ berechnet/gespeichert, die ausserhalb der Nichtnullenstruktur von \mathbf{A} liegen. Mit dem Vorkonditionierer $\mathbf{P} = \tilde{\mathbf{R}}^T \tilde{\mathbf{R}}$ ergeben sich die folgenden Konvergenzkurven.



Kapitel 9

Eigenwertprobleme

Wir erinnern an die Definition von Eigenwerten aus Abschnitt 0.5. Eine komplexe Zahl λ heisst **Eigenwert** der Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, wenn es einen von Null verschiedenen Vektor $\underline{v} \in \mathbb{C}^n$ gibt, so dass

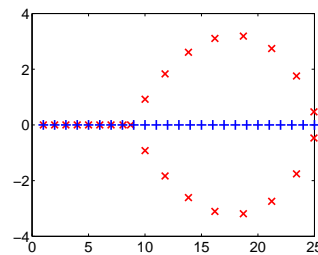
$$\mathbf{A}\underline{v} = \lambda\underline{v} \quad (9.1)$$

gilt. Der Vektor \underline{v} heisst der zu λ gehörige **Eigenvektor**. Die Menge aller Eigenwerte von \mathbf{A} heisst **Spektrum** und wird mit $\Lambda(\mathbf{A})$ bezeichnet.

Eine zu (9.1) äquivalente Charakterisierung ist $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$; die Eigenwerte sind also die Nullstellen des charakteristischen Polynoms $\det(\mathbf{A} - \lambda\mathbf{I})$. Diese Charakterisierung hat einige nützliche theoretische Konsequenzen – so sichert etwa der Fundamentalsatz der Algebra die Existenz von Eigenwerten. Numerisch ist sie aber *nicht* brauchbar.

Beispiel 9.1 (Wilkinson) Der folgende MATLAB-Code berechnet zunächst die Koeffizienten des charakteristische Polynoms der Matrix $\text{diag}(1, 2, \dots, 25)$ und *danach* die Nullstellen des Polynoms.

```
MATLAB
p = poly(diag(1:25));
plot(roots(p), 'rx');
```



Die berechneten Nullstellen weisen sehr stark von den exakten ab. Ursachen sind Rundungsfehler bei der Berechnung der Koeffizienten des Polynoms sowie die enormen Größenunterschiede der Koeffizienten (zwischen 1 und 10^{25}). \diamond

Die Entwicklung numerisch sinnvoller Verfahren für Eigenwertprobleme ist um einiges komplizierter als für lineare Gleichungssysteme. Im Rahmen dieser Einführungsvorlesung können wir nur auf die einfachsten Verfahren im Detail eingehen. Siehe [Golub/Van Loan'1996] für weitere Verfahren.

9.1 Eigenwertabschätzungen[★]

In vielen Anwendungen von Eigenwertproblemen benötigt man nicht die exakten Eigenwerte sondern lediglich den Nachweis, dass diese innerhalb eines gewissen Gebietes der komplexen Ebene liegen. Beispiel: Die Konvergenz einer linearen Fixpunktiteration ist nachgewiesen, wenn alle Eigenwerte der Iterationsmatrix innerhalb des offenen Einheitskreises liegen, siehe Satz 8.4. In solchen Fällen reicht unter Umständen eine hinreichend gute Schätzung der Eigenwerte aus.

Eine Eigenwertabschätzung kennen wir bereits: aus (9.1) ergibt sich $|\lambda| \leq \|\mathbf{A}\|$ für jede konsistente Matrixnorm $\|\cdot\|$. Ist \mathbf{A} invertierbar so folgt ebenfalls aus (9.1) die Beziehung

$$\mathbf{A}^{-1}\underline{v} = \frac{1}{\lambda}\underline{v} \quad (9.2)$$

und damit

$$\frac{1}{|\lambda|} \|\underline{v}\| = \|\mathbf{A}^{-1}\underline{v}\| \leq \|\mathbf{A}^{-1}\| \|\underline{v}\| \Rightarrow |\lambda| \geq \frac{1}{\|\mathbf{A}^{-1}\|}.$$

Die Eigenwerte von \mathbf{A} liegen also in einer Kreisscheibe mit innerem Radius $\|\mathbf{A}^{-1}\|$ und äusserem Radius $\|\mathbf{A}\|$.

Der folgende Satz von Gerschgorin erlaubt oftmals wesentlich bessere Abschätzungen.

Satz 9.2 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$. Jeder Eigenwert von \mathbf{A} befindet sich in einem der sogenannten **Gerschgorin-Kreise**

$$K_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\}, \quad i = 1, \dots, n.$$

Es gilt also $\Lambda(\mathbf{A}) \subset \bigcup_{i=1}^n K_i$.

Beweis. Sei $\lambda \in \Lambda(\mathbf{A})$ und \underline{v} der zugehörige Eigenvektor. Bezeichne v_i den Eintrag von \underline{v} , der Maximalbetrag hat: $|v_i| = \max_{j=1, \dots, n} |v_j|$. Umstellen der Beziehung $\mathbf{A}\underline{v} = \lambda\underline{v}$ ergibt

$$(\lambda - a_{ii})v_i = \sum_{j \neq i} a_{ij}v_j$$

und damit

$$|\lambda - a_{ii}| |v_i| = \left| \sum_{j \neq i} a_{ij}v_j \right| \leq \sum_{j \neq i} |a_{ij}| |v_j|.$$

Daraus folgt

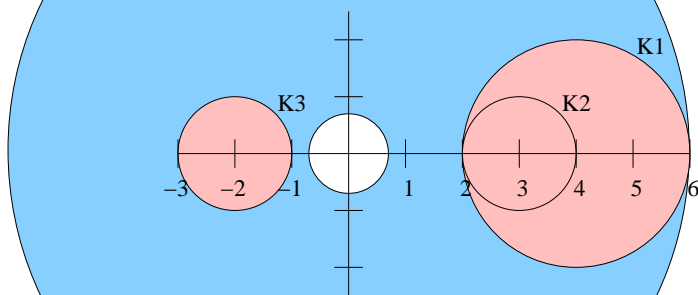
$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \frac{|v_j|}{|v_i|} \leq \sum_{j \neq i} |a_{ij}|$$

und schliesslich $\lambda \in K_i$. \square

Beispiel 9.3 Die Matrix

$$\mathbf{A} = \begin{pmatrix} 4 & 1 & -1 \\ 0 & 3 & -1 \\ 1 & 0 & -2 \end{pmatrix}.$$

hat das Spektrum $\Lambda(\mathbf{A}) = \{3.43 \pm 0.14i, -1.96\}$. Die folgende Abbildung zeigt die Gerschgorin-Kreise (pink) und einen Ausschnitt der Kreisscheibe mit den Radien $\|\mathbf{A}\|_\infty = 6$ und $\|\mathbf{A}^{-1}\|_\infty = 8/11$ (hellblau im Hintergrund).



Als Folge des Satzes von Gerschgorin erhalten wir auch Schranken für das Verhalten der Eigenwerte, wenn \mathbf{A} gestört wird, beispielsweise durch Rundungsfehler.

Satz 9.4 Sei $\mathbf{A} \in \mathbb{R}^{n \times n}$ diagonalisierbar:

$$\mathbf{V}^{-1}\mathbf{A}\mathbf{V} = \text{diag}(\lambda_1, \dots, \lambda_n).$$

Für jeden Eigenwert $\hat{\lambda}$ von $\mathbf{A} + \mathbf{E}$ gilt

$$\min_{1 \leq i \leq n} |\hat{\lambda} - \lambda_i| \leq \|\mathbf{V}\|_p \|\mathbf{V}^{-1}\|_p \|\mathbf{E}\|_p, \tag{9.3}$$

wobei $p \in \{1, 2, \infty\}$.

Beweis. Es wird nur der Fall $p = \infty$ (Zeilensummennorm) bewiesen. Um den Satz von Gerschgorin anwenden zu können, transformieren wir $\mathbf{A} + \mathbf{E}$:

$$\hat{\mathbf{A}} := \mathbf{V}^{-1}(\mathbf{A} + \mathbf{E})\mathbf{V} = \text{diag}(\lambda_1, \dots, \lambda_n) + \underbrace{\mathbf{V}^{-1}\mathbf{E}\mathbf{V}}_{=: \hat{\mathbf{E}}}.$$

Die Eigenwerte von $\mathbf{A} + \mathbf{E}$ und $\hat{\mathbf{A}}$ sind identisch. Wegen Satz 9.2 angewandt auf $\hat{\mathbf{A}}$ gibt es für jedes $\hat{\lambda}$ von $\mathbf{A} + \mathbf{E}$ ein $1 \leq i \leq n$, so dass

$$|\hat{\lambda} - \lambda_i - \hat{e}_{ii}| \leq \sum_{j \neq i} |\hat{e}_{ij}|, \tag{9.4}$$

wobei \hat{e}_{ij} die Einträge von $\hat{\mathbf{E}}$ bezeichnen. Mittels Dreiecksungleichung folgt aus (9.4)

$$|\hat{\lambda} - \lambda_i| = |\hat{\lambda} - \lambda_i - \hat{e}_{ii} + \hat{e}_{ii}| \leq |\mu - \lambda_i - \hat{e}_{ii}| + |\hat{e}_{ii}| \leq \sum_{j=1}^n |\hat{e}_{ij}| \leq \|\hat{\mathbf{E}}\|_\infty,$$

was zusammen mit $\|\hat{\mathbf{E}}\|_\infty = \|\mathbf{V}^{-1}\mathbf{E}\mathbf{V}\|_\infty \leq \|\mathbf{V}^{-1}\|_\infty \|\mathbf{E}\|_\infty \|\mathbf{V}\|_\infty$ die Behauptung nach sich zieht. \square

Als Korollar von Satz 9.4 erhalten wir für eine *symmetrische* (oder allgemeiner: normale) Matrix \mathbf{A}

$$\min_{1 \leq i \leq n} |\hat{\lambda} - \lambda_i| \leq \|\mathbf{E}\|_2.$$

Die Eigenwerte von normalen Matrizen sind also sehr gut konditioniert. Je weiter \mathbf{A} von einer normalen Matrix “entfernt” ist, desto schlechter sind ihre Eigenwerte konditioniert. Den Extremfall bildet der $n \times n$ Jordan-Block

$$\mathbf{J}_n = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

auf den Satz 9.4 *nicht* angewandt werden kann. Die Eigenwerte der gestörten Matrix

$$\mathbf{J}_n + \mathbf{E} = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 0 \end{pmatrix}$$

sind alle n -ten Wurzeln von ϵ . Ist zum Beispiel $n = 16$ und $\epsilon = 10^{-16}$ so haben alle Eigenwerte Betrag 10^{-1} obwohl die Eigenwerte der ungestörten Matrix 0 sind!

9.2 Die Potenzmethode

Wir behandeln jetzt das fundamentalste Verfahren zur Berechnung eines Eigenwertes und Eigenvektors, die **Potenzmethode**. Die Methode ist denkbar einfach; im Wesentlichen multiplizieren wir einen gegebenen Startvektor $\underline{x}_0 \neq 0$ wiederholt mit der Matrix \mathbf{A} :

$$\underline{x}_{k+1} \leftarrow \mathbf{A}\underline{x}_k,$$

oder $\underline{x}_k = \mathbf{A}^k \underline{x}_0$. Gilt nicht gerade $\rho(\mathbf{A}) = 1$, so wird diese Iteration i.a. divergieren ($\rho(\mathbf{A}) > 1$) oder gegen Null konvergieren ($\rho(\mathbf{A}) < 1$). Um diesen Effekt zu vermeiden, normalisieren wir die Iterierte in jedem Schritt:

$$\underline{y}_{k+1} \leftarrow \mathbf{A}\underline{x}_k, \quad \underline{x}_{k+1} \leftarrow \underline{y}_{k+1} / \|\underline{y}_{k+1}\|_2, \tag{9.5}$$

oder $\underline{x}_k = \mathbf{A}^k \underline{x}_0 / \|\mathbf{A}^k \underline{x}_0\|_2$. Wir werden im folgenden zeigen, dass \underline{x}_k gegen einen Eigenvektor von \mathbf{A} konvergiert.

Dazu nehmen wir an, dass die Eigenwerte $\lambda_1, \dots, \lambda_n$ von \mathbf{A} sich wie folgt sortieren lassen:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|;$$

es soll also nur einen betragsgrössten Eigenwert geben. Desweiteren nehmen wir die Diagonalisierbarkeit von \mathbf{A} an: es gibt also eine Basis von Eigenvektoren $\underline{v}_1, \dots, \underline{v}_n$, die jeweils zu $\lambda_1, \dots, \lambda_n$ gehören. Darstellung des Startvektors \underline{x}_0 in dieser Basis:

$$\underline{x}_0 = \gamma_1 \underline{v}_1 + \gamma_2 \underline{v}_2 + \dots + \gamma_n \underline{v}_n.$$

Dritte und letzte Annahme: $\gamma_1 \neq 0$. Zusammen mit der Beziehung $\mathbf{A}\underline{v}_i = \lambda_i \underline{v}_i$, $i = 1, \dots, n$, ergibt sich

$$\mathbf{A}\underline{x}_0 = \gamma_1 \mathbf{A}\underline{v}_1 + \gamma_2 \mathbf{A}\underline{v}_2 + \dots + \gamma_n \mathbf{A}\underline{v}_n = \gamma_1 \lambda_1 \underline{v}_1 + \gamma_2 \lambda_2 \underline{v}_2 + \dots + \gamma_n \lambda_n \underline{v}_n.$$

Wiederholtes Anwenden dieser Beziehung führt auf

$$\mathbf{A}^k \underline{x}_0 = \gamma_1 \lambda_1^k \underline{v}_1 + \gamma_2 \lambda_2^k \underline{v}_2 + \dots + \gamma_n \lambda_n^k \underline{v}_n.$$

Also gilt

$$\begin{aligned} \underline{x}_k &= \frac{\mathbf{A}^k \underline{x}_0}{\|\mathbf{A}^k \underline{x}_0\|_2} = \frac{\gamma_1 \lambda_1^k \underline{v}_1 + \gamma_2 \lambda_2^k \underline{v}_2 + \dots + \gamma_n \lambda_n^k \underline{v}_n}{\|\gamma_1 \lambda_1^k \underline{v}_1 + \gamma_2 \lambda_2^k \underline{v}_2 + \dots + \gamma_n \lambda_n^k \underline{v}_n\|_2} \\ &= \frac{\gamma_1 \underline{v}_1 + \gamma_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \underline{v}_2 + \dots + \gamma_n \left(\frac{\lambda_n}{\lambda_1}\right)^k \underline{v}_n}{\|\gamma_1 \underline{v}_1 + \gamma_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \underline{v}_2 + \dots + \gamma_n \left(\frac{\lambda_n}{\lambda_1}\right)^k \underline{v}_n\|_2} \\ &= \frac{\gamma_1 \underline{v}_1}{\|\gamma_1 \underline{v}_1\|_2} + O(|\lambda_2/\lambda_1|^k) \end{aligned} \quad (9.6)$$

wobei wir im letzten Schritt die Beziehungen $|\lambda_i|/|\lambda_1| < 1$ für $i \geq 2$ und $\gamma_1 \neq 0$ ausgenutzt haben. Der Vektor \underline{x}_k konvergiert also gegen einen zu λ_1 gehörigen Eigenvektor (zur Erinnerung: jedes skalare Vielfache von \underline{v}_1 ist ein Eigenvektor). Die Konvergenz ist linear mit Konvergenzrate $|\lambda_2/\lambda_1|$.

Wie erhalten wir aus \underline{x}_k Approximationen zu dem Eigenwert λ_1 ? Sei $\tilde{v}_1 = \gamma_1 \underline{v}_1 / \|\gamma_1 \underline{v}_1\|_2$. Dann folgt aus (9.6) die Beziehung

$$\mu_k = \underline{x}_k^H \mathbf{A} \underline{x}_k = \tilde{v}_1^H \mathbf{A} \tilde{v}_1 + O(|\lambda_2/\lambda_1|^k) = \lambda_1 + O(|\lambda_2/\lambda_1|^k). \quad (9.7)$$

Also ergibt der sogenannte **Rayleigh-Quotient** μ_k eine gute Approximation zu λ_1 , vorausgesetzt dass \underline{x}_k bereits eine gute Approximation zu einem Eigenvektor ist.

Bemerkung 9.5 Für eine symmetrische Matrix lässt sich die Konvergenzschranke (9.7) noch verbessern:

$$\mu_k = \lambda_1 + O(|\lambda_2/\lambda_1|^{2k}). \quad (9.8)$$

Als letztes müssen wir noch ein sinnvolles Stopkriterium für die Potenzmethode angeben. Eine Möglichkeit ist zu testen, ob die Norm des **Residuums** unter einer vorgegebenen Toleranz tol liegt:

$$\|\mathbf{A}\underline{x}_k - \mu_k \underline{x}_k\|_2 \leq \text{tol}. \quad (9.9)$$

Die für ein gewisses tol erzielte Genauigkeit des Eigenwertes bzw. Eigenvektors hängt – ähnlich wie bei linearen Gleichungssystemen – von der Kondition des Eigenwertes bzw. Eigenvektors ab. Insgesamt ergibt sich der folgende Algorithmus.

Algorithmus 9.6 (Potenzmethode)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, Startvektor \underline{x}_0 mit $\|\underline{x}_0\|_2 = 1$, Toleranz tol .

Output: Approximationen μ_k und \underline{x}_k zu dem betragsgrössten Eigenwert λ_1 von \mathbf{A} und dem zugehörigen Eigenvektor.

for $k = 0, 1, 2, \dots$ **do**

```

y_{k+1} ← A x_k
μ_k ← x_k^H y_{k+1}
Exit, wenn ||y_{k+1} - μ_k x_k||_2 ≤ tol
x_{k+1} ← y_{k+1} / ||y_{k+1}||_2
end for

```

Wir kommen auf die oben getroffenen Annahmen zurück, unter denen Algorithmus 9.6 konvergiert.

- $|\lambda_1| > |\lambda_2|$: Diese Annahme ist essentiell, wie man sich an dem Beispiel $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, $\underline{x}_0 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$ vergewissern kann.
- Diagonalisierbarkeit von \mathbf{A}** : Diese Annahme wurde lediglich aus Bequemlichkeit getroffen; die Potenzmethode konvergiert auch, wenn die Matrix \mathbf{A} Jordanblöcke zu von λ_1 verschiedenen Eigenwerten hat.
- $\gamma_1 \neq 0$: Im Prinzip ist diese Annahme ebenfalls essentiell; ansonsten konvergiert die Potenzmethode gegen den “falschen” Eigenwert, z.B. bei $\mathbf{A} = \text{diag}(3, 2, 1)$ und $\underline{x}_0 = (0, 1/\sqrt{2}, 1/\sqrt{2})^T$ gegen $\lambda_2 = 2$ anstatt $\lambda_1 = 3$. In endlicher Arithmetik ist eine solche Situation bei nichttrivialen Beispielen so gut wie ausgeschlossen; Rundungsfehler führen im Laufe der Iteration fast immer Komponenten von \underline{v}_1 in die Iteration ein und die Potenzmethode konvergiert letztendlich gegen λ_1 . Dies ist eine der seltenen Fälle bei denen Rundungsfehler vom Nutzen sind!

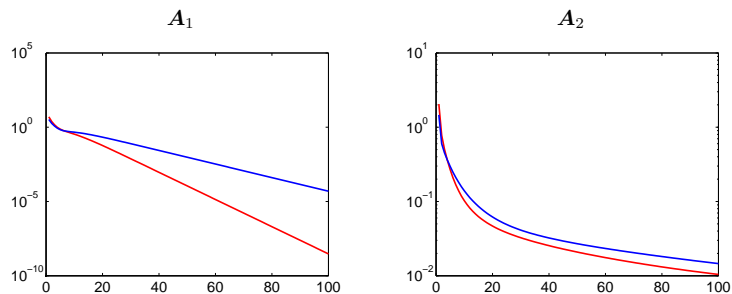
Beispiel 9.7 Wir wenden die Potenzmethode mit zufälligem Startvektor auf die folgenden beiden Matrizen an:

$$\mathbf{A}_1 = \text{diag}(1, 2, \dots, 10), \quad \mathbf{A}_2 = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Man kann zeigen, dass die Eigenwerte von \mathbf{A}_2 wie folgt gegeben sind:

$$\lambda_i = 4 \sin^2 \left(\frac{n-i+1}{n+1} \frac{\pi}{2} \right), \quad i = 1, \dots, n. \quad (9.10)$$

Für grosse n sind sowohl λ_1 als auch λ_2 sehr nahe bei 1; die Potenzmethode wird also nur sehr langsam konvergieren. Im folgenden verwenden wir $n = 100$ für das $|\lambda_2/\lambda_1| \approx 0.99981$ gilt. Die folgenden Abbildungen zeigen $|\mu_k - \lambda_1|$ (rote Kurve) und $\|\mathbf{A}\underline{x}_k - \mu_k \underline{x}_k\|_2$ (blaue Kurve) für $k = 1, \dots, 100$.



◇

9.3 Inverse Iteration

Mit der Potenzmethode kann nur der betragsgrösste Eigenwert berechnet werden. Soll der betragskleinste Eigenwert oder – allgemeiner – der Eigenwert, der am nächsten zu einem vorgegebenen Zielwert $\tau \in \mathbb{C}$ liegt, berechnet werden, so kann man den folgenden einfachen Trick anwenden. Die Beziehung $\mathbf{A}\underline{v} = \lambda\underline{v}$ ist äquivalent zu $(\mathbf{A} - \tau\mathbf{I}_n)\underline{v} = (\lambda - \tau)\underline{v}$. Ist τ kein Eigenwert von \mathbf{A} , dann ist $\mathbf{A} - \tau\mathbf{I}_n$ invertierbar und wir erhalten die äquivalente Beziehung

$$(\mathbf{A} - \tau\mathbf{I}_n)^{-1}\underline{v} = \frac{1}{\lambda - \tau}\underline{v}. \quad (9.11)$$

Also ist λ genau dann ein Eigenwert von \mathbf{A} , wenn $1/(\lambda - \tau)$ ein Eigenwert von $(\mathbf{A} - \tau\mathbf{I}_n)^{-1}$ ist. Die Eigenvektoren ändern sich durch die Transformation nicht. Der betragsgrösste Eigenwert von $(\mathbf{A} - \tau\mathbf{I}_n)^{-1}$ ist der zu τ nächsten gelegene Eigenwert von \mathbf{A} .³² Gilt $|\lambda_i - \tau| < |\lambda_j - \tau|$ für alle anderen Eigenwerte λ_j , $j \neq i$, so können wir also die Potenzmethode auf $(\mathbf{A} - \tau\mathbf{I}_n)^{-1}$ anwenden, um eine Vektorfolge $\{\underline{x}_k\}$ zu generieren die gegen den zu λ_i gehörigen Eigenvektor konvergiert. Um Eigenwertapproximationen μ_k zur Originalmatrix \mathbf{A} aus der generierten Vektorfolge \underline{x}_k zu gewinnen, muss die Transformation wieder rückgängig gemacht werden:

$$\mu_k = \frac{1}{\underline{x}_k^H (\mathbf{A} - \tau\mathbf{I})^{-1} \underline{x}_k} + \tau.$$

Algorithmus 9.8 (Inverse Iteration)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, Startvektor \underline{x}_0 mit $\|\underline{x}_0\|_2 = 1$, Zielwert $\tau \in \mathbb{C}$, Toleranz tol .

Output: Approximationen μ_k und \underline{x}_k zu dem zu τ nächsten Eigenwert von \mathbf{A} und dem zugehörigen Eigenvektor.

for $k = 0, 1, 2, \dots$ **do**

$$\underline{y}_{k+1} \leftarrow (\mathbf{A} - \tau\mathbf{I}_n)^{-1}\underline{x}_k$$

$$\mu_k \leftarrow \frac{1}{\underline{x}_k^H \underline{y}_{k+1}} + \tau$$

Exit, wenn $\|\mathbf{A}\underline{x}_k - \mu_k \underline{x}_k\|_2 \leq \text{tol}$

³²Für $\tau = 0$ ist dies gerade der betragskleinste Eigenwert von \mathbf{A} .

```

 $\underline{x}_{k+1} \leftarrow \underline{y}_{k+1} / \|\underline{y}_{k+1}\|_2$ 
end for

```

Man beachte, dass die inverse Iteration wesentlich teurer als die Potenzmethode ist: anstatt nur mit \mathbf{A} zu multiplizieren muss man in jedem Schritt ein lineares Gleichungssystem lösen. Algorithmus 9.8 konvergiert linear mit der Konvergenzrate

$$\frac{\max_{j \neq i} \frac{1}{|\lambda_j - \tau|}}{\frac{1}{|\lambda_i - \tau|}} = \frac{|\lambda_i - \tau|}{\min_{j \neq i} |\lambda_j - \tau|}.$$

9.4 Krylovraum-Verfahren

Bei der Potenzmethode nutzt man nur die Information aus der aktuellen Iterierten. Es liegt nahe, zur Verbesserung der Konvergenz die Information aus *allen* vorangegangenen Iterationen zu nützen. Die Iterierten der Potenzmethode spannen gerade den uns bereits bekannten Krylov-Raum auf, siehe Definition 8.21. Aus diesem werden wir Approximationen zu den Eigenvektoren extrahieren. Um dies tun zu können beschaffen wir uns zunächst eine orthonormale Basis (ONB) zu

$$\mathcal{K}_k(\mathbf{A}, \underline{x}_0) = \text{span}\{\underline{x}_0, \mathbf{A}\underline{x}_0, \dots, \mathbf{A}^{k-1}\underline{x}_0\}.$$

Dazu verwenden wir das sogenannte *Arnoldi-Verfahren*, das im wesentlichen eine Variante von Gram-Schmidt ist.

9.4.1 Arnoldi-Verfahren

Die ONB von $\mathcal{K}_k(\mathbf{A}, \underline{x}_0)$ wird schrittweise aufgebaut. Im ersten Schritt setzen wir $\underline{u}_1 = \underline{x}_0 / \|\underline{x}_0\|_2$. Dann bildet $\{\underline{u}_1\}$ eine ONB von $\mathcal{K}_1(\mathbf{A}, \underline{x}_0)$, natürlich nur unter der Voraussetzung, dass $\underline{x}_0 \neq 0$ gilt. Wir nehmen nun an, wir hätten bereits eine ONB $\{\underline{u}_1, \underline{u}_2, \dots, \underline{u}_j\}$ von $\mathcal{K}_j(\mathbf{A}, \underline{x}_0)$ konstruiert. Um eine ONB von $\mathcal{K}_{j+1}(\mathbf{A}, \underline{x}_0)$ zu konstruieren könnten wir $\mathbf{A}^j \underline{x}_0$ gegen die bereits berechneten Basisvektoren mittels Gram-Schmidt orthogonalisieren. Ein solches Vorgehen stellt sich aber als unpraktikabel heraus, insbesondere führt es zu numerischen Instabilitäten.³³ Abhilfe schafft hier die folgende Überlegung: Da $\mathbf{A}^{j-1} \underline{x}_0 \in \mathcal{K}_j(\mathbf{A}, \underline{x}_0) = \text{span}\{\underline{u}_1, \dots, \underline{u}_j\}$, gibt es β_1, \dots, β_j , so dass $\mathbf{A}^{j-1} \underline{x}_0 = \sum_{i=1}^j \beta_i \underline{u}_i$. Ausserdem muss $\beta_j \neq 0$ sein, denn sonst wäre $\mathbf{A}^{j-1} \underline{x}_0 \in \text{span}\{\underline{u}_1, \dots, \underline{u}_{j-1}\} = \mathcal{K}_{j-1}(\mathbf{A}, \underline{x}_0)$. Damit ist aber $\mathbf{A}^j \underline{x}_0 - \beta_j \mathbf{A} \underline{u}_j = \mathbf{A} \sum_{i=1}^{j-1} \beta_i \underline{u}_i \in \mathcal{K}_j(\mathbf{A}, \underline{x}_0)$. Das heisst, $\mathbf{A}^j \underline{x}_0$ und $\mathbf{A} \underline{u}_j$ unterscheiden sich voneinander nur durch einen Vektor aus $\mathcal{K}_j(\mathbf{A}, \underline{x}_0)$ und eine Skalierung. Daher können wir das Verfahren auch fortsetzen, indem wir $\underline{w} = \mathbf{A} \underline{u}_j$ anstatt $\mathbf{A}^j \underline{x}_0$ gegenüber der bestehenden ONB mittels Gram-Schmidt orthogonalisieren:

$$\tilde{\underline{u}}_{j+1} = \underline{w} - \sum_{i=1}^j (\underline{u}_i^T \underline{w}) \underline{u}_i, \quad \underline{u}_{j+1} = \tilde{\underline{u}}_{j+1} / \|\tilde{\underline{u}}_{j+1}\|_2.$$

Setzen wir $\mathbf{U}_j = (\underline{u}_1, \underline{u}_2, \dots, \underline{u}_j)$, so lässt sich dies auch kompakter schreiben als

$$\underline{h}_j = \mathbf{U}_j^T \underline{w}, \quad \tilde{\underline{u}}_{j+1} = \underline{w} - \mathbf{U}_j \underline{h}_j, \quad \underline{u}_{j+1} = \tilde{\underline{u}}_{j+1} / \|\tilde{\underline{u}}_{j+1}\|_2. \quad (9.12)$$

Insgesamt ergibt sich das folgende Verfahren von Arnoldi.

Algorithmus 9.9 (Arnoldi-Verfahren)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, Startvektor $\underline{x}_0 \neq 0$, $k \in \mathbb{N}$.

Output: ONB $\{\underline{u}_1, \dots, \underline{u}_k, \underline{u}_{k+1}\}$ des Krylov-Raums $\mathcal{K}_{k+1}(\mathbf{A}, \underline{x}_0)$.

$\underline{u}_1 \leftarrow \underline{x}_0 / \|\underline{x}_0\|_2$, $\mathbf{U}_1 = \underline{u}_1$.

³³Der Raum $\mathcal{K}_j(\mathbf{A}, \underline{x}_0)$ enthält die ersten $j-1$ Iterierten der Potenzmethode. Da alle Iterierten gegen den gleichen Vektor streben, nämlich den Eigenvektor zum betragsgrössten Eigenwert, ist davon auszugehen, dass die j -te Iterierte ($\mathbf{A}^j \underline{x}_0$) bereits nahezu vollständig im Raum $\mathcal{K}_j(\mathbf{A}, \underline{x}_0)$ enthalten ist. Dies ist genau die Situation, die im Gram-Schmidt-Algorithmus zu numerischen Instabilitäten führt.

```

for  $j = 1, 2, \dots, k$  do
   $\underline{w} \leftarrow \mathbf{A} \underline{u}_j$ 
   $\underline{h}_j \leftarrow \mathbf{U}_j^T \underline{w}$ 
   $\tilde{\underline{u}}_{j+1} \leftarrow \underline{w} - \mathbf{U}_j \underline{h}_j$ 
   $h_{j+1,j} \leftarrow \|\tilde{\underline{u}}_{j+1}\|_2$ 
   $\underline{u}_{j+1} \leftarrow \tilde{\underline{u}}_{j+1} / h_{j+1,j}$ 
   $\mathbf{U}_{j+1} \leftarrow (\mathbf{U}_j, \underline{u}_{j+1})$ 
end for

```

Es lohnt sich die im Arnoldi-Verfahren berechneten Koeffizienten \underline{h}_j und $h_{j+1,j}$ abzuspeichern. Bezeichnen wir $\underline{h}_j = (h_{1j}, h_{2j}, \dots, h_{jj})^T$, so ergibt sich aus diesen Koeffizienten die $(k+1) \times k$ -Matrix

$$\tilde{\mathbf{H}}_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \ddots & \vdots \\ & \ddots & \ddots & h_{k-1,k} \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix} \quad (9.13)$$

Lassen wir die letzte Zeile weg, so ergibt sich die $k \times k$ -Matrix

$$\mathbf{H}_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \ddots & \vdots \\ & \ddots & \ddots & h_{k-1,k} \\ & & h_{k,k-1} & h_{kk} \end{pmatrix}$$

Matrizen dieser Form, deren Einträge unterhalb der ersten Nebendiagonale alle Null sind, bezeichnet man auch als (obere) *Hessenberg-Matrizen*.

Die Beziehung (9.12), die zum Arnoldi-Verfahren führte, lässt sich umschreiben als

$$\begin{aligned} \mathbf{A} \underline{u}_j &= \mathbf{U}_j \underline{h}_j + \tilde{\underline{u}}_{j+1} = \mathbf{U}_j \underline{h}_j + h_{j+1,j} \underline{u}_{j+1} \\ &= (\mathbf{U}_j, \underline{u}_{j+1}) \begin{pmatrix} \underline{h}_j \\ h_{j+1,j} \end{pmatrix} = \mathbf{U}_{k+1} \begin{pmatrix} \underline{h}_j \\ h_{j+1,j} \\ 0 \end{pmatrix}. \end{aligned}$$

Fassen wir diese Beziehung für $j = 1, \dots, k$ zusammen, so ergibt sich die sogenannte *Arnoldi-Zerlegung*

$$\mathbf{A} \mathbf{U}_k = \mathbf{U}_{k+1} \tilde{\mathbf{H}}_k,$$

oder – äquivalent dazu –

$$\mathbf{A} \mathbf{U}_k = \mathbf{U}_k \mathbf{H}_k + h_{k+1,k} \underline{e}_k \underline{e}_k^T, \quad (9.14)$$

wobei \underline{e}_k den k -ten Einheitsvektor der Länge k bezeichnet. Daraus ergibt sich insbesondere

$$\mathbf{H}_k = \mathbf{U}_k^T \mathbf{A} \mathbf{U}_k. \quad (9.15)$$

Bemerkung 9.10 Trotz der oben besprochenen Umformulierung erbt auch das Arnoldi-Verfahren die numerische Instabilität der Gram-Schmidt-Orthogonalisierung Arnoldi-Verfahrens. Hier bietet sich der am Ende von Abschnitt 7.4 angesprochene verblüffend einfache Ausweg an: es wird in kritischen Fällen einfach noch einmal orthogonalisiert. Kritisch sind Auslöschungen in Zeile $\tilde{\mathbf{u}}_{j+1} \leftarrow \underline{\mathbf{w}} - \mathbf{U}_j \hat{\mathbf{h}}_j$ von Algorithmus 9.9, wenn also $\|\tilde{\mathbf{u}}_{j+1}\|_2$ sehr viel kleiner als $\|\underline{\mathbf{w}}\|_2$. (Ein in der Praxis übliches Kriterium ist $\|\tilde{\mathbf{u}}_{j+1}\|_2 < 0.7\|\underline{\mathbf{w}}\|_2$.) Man fügt nach dieser Zeile in Algorithmus 9.9 also noch folgendes ein:

if $\|\tilde{\mathbf{u}}_{j+1}\|_2 < 0.7\|\underline{\mathbf{w}}\|_2$ **then**
 $\hat{\mathbf{h}}_j = \mathbf{U}_j^T \tilde{\mathbf{u}}_{j+1}$, $\hat{\mathbf{h}}_j \leftarrow \hat{\mathbf{h}}_j + \hat{\mathbf{h}}_j$, $\tilde{\mathbf{u}}_{j+1} \leftarrow \tilde{\mathbf{u}}_{j+1} - \mathbf{U}_j \hat{\mathbf{h}}_j$
end if

Der zusätzliche Aufwand ist – im Vergleich zum Rest des Arnoldi-Verfahrens – oft vernachlässigbar.

9.4.2 Extraktion von Ritz-Paaren

Zur Approximation von Eigenvektoren und Eigenwerten der Matrix \mathbf{A} benutzen wir nur Vektoren $\underline{\mathbf{v}} \in \mathcal{K}_k(\mathbf{A}, \underline{\mathbf{x}}_0)$ und Skalare $\mu \in \mathbb{C}$, bei denen das entsprechende Residuum orthogonal zu $\mathcal{K}_k(\mathbf{A}, \underline{\mathbf{x}}_0)$ ist:

$$\mathbf{A}\underline{\mathbf{v}} - \mu\underline{\mathbf{v}} \perp \mathcal{K}_k(\mathbf{A}, \underline{\mathbf{x}}_0) \quad (9.16)$$

Diese Bedingung nennt man *Galerkin-Bedingung* und sie tritt in ähnlicher Form bei den linearen Ausgleichsproblemen auf, siehe Bemerkung 7.7. Mit der ONB \mathbf{U}_k lässt sich (9.16) auch schreiben als

$$0 = \mathbf{U}_k^T (\mathbf{A}\underline{\mathbf{v}} - \mu\underline{\mathbf{v}}) = \mathbf{U}_k^T (\mathbf{A}\mathbf{U}_k \underline{\mathbf{z}} - \mu\mathbf{U}_k \underline{\mathbf{z}}) = \mathbf{H}_k \underline{\mathbf{z}} - \mu\underline{\mathbf{z}},$$

für einen beliebigen Vektor $\underline{\mathbf{z}} \in \mathbb{C}^k$, wobei wir (9.15) ausgenutzt haben. In Worten: $\underline{\mathbf{v}} = \mathbf{U}_k \underline{\mathbf{z}} \neq 0$ und μ erfüllen genau dann (9.16) wenn μ ein Eigenwert von \mathbf{H}_k und $\underline{\mathbf{z}}$ der entsprechende Eigenvektor ist. Ein solches Paar $(\mu, \underline{\mathbf{v}})$ nennt man *Ritz-Paar*, und μ bzw. $\underline{\mathbf{v}}$ dementsprechend *Ritz-Wert* bzw. *Ritz-Vektor*.

Jeder der k Eigenwerte von \mathbf{H}_k liefert also ein Ritz-Paar. Um zu überprüfen ob ein Ritz-Paar eine gute Approximation zu einem Eigenwert/Eigenvektor von \mathbf{A} ergibt, berechnet man die 2-Norm des Residuums:

$$\begin{aligned} \|\mathbf{A}\underline{\mathbf{v}} - \mu\underline{\mathbf{v}}\|_2 &= \|\mathbf{A}\mathbf{U}_k \underline{\mathbf{z}} - \mu\mathbf{U}_k \underline{\mathbf{z}}\|_2 \\ &= \|\mathbf{U}_k \mathbf{H}_k \underline{\mathbf{z}} + h_{k+1,k} \underline{\mathbf{e}}_{k+1} \underline{\mathbf{e}}_k^T \underline{\mathbf{z}} - \mu\mathbf{U}_k \underline{\mathbf{z}}\|_2 \\ &= \|\mathbf{U}_k (\underbrace{\mathbf{H}_k \underline{\mathbf{z}} - \mu\underline{\mathbf{z}}}_{=0}) + h_{k+1,k} \underline{\mathbf{u}}_{k+1} \underline{\mathbf{e}}_k^T \underline{\mathbf{z}}\|_2 \\ &= |h_{k+1,k}| \underbrace{\|\underline{\mathbf{e}}_k^T \underline{\mathbf{z}}\|_2}_{=1} = |h_{k+1,k}| \|\underline{\mathbf{e}}_k^T \underline{\mathbf{z}}\|_2 \end{aligned}$$

wobei im zweiten Schritt die Arnoldi-Zerlegung (9.14) eingesetzt wurde. Die Norm des Residuums ergibt sich also aus dem Produkt von $h_{k+1,k}$ mit dem k -ten Eintrag des Eigenvektors $\underline{\mathbf{z}}$ von \mathbf{H}_k .

9.4.3 Lanczos-Verfahren

Ist \mathbf{A} symmetrisch, dann vereinfacht sich das Arnoldi-Verfahren. Insbesondere erbt die Hessenberg-Matrix $\mathbf{H}_k = \mathbf{U}_k^T \mathbf{A} \mathbf{U}_k$ die Symmetrie von \mathbf{A} . Eine symmetrische

Hessenberg-Matrix ist aber automatisch tridiagonal:

$$\mathbf{H}_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & & & \\ & & \ddots & & \\ & & & \ddots & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix}, \quad \alpha_j, \beta_j \in \mathbb{R}. \quad (9.17)$$

Die diese Nullstruktur ausnutzende Variante des Arnoldi-Verfahrens nennt man *Lanczos-Verfahren*.

Algorithmus 9.11 (Lanczos-Verfahren)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, Startvektor $\underline{\mathbf{x}}_0 \neq 0$, $k \in \mathbb{N}$.

Output: ONB $\{\underline{\mathbf{u}}_1, \dots, \underline{\mathbf{u}}_k, \underline{\mathbf{u}}_{k+1}\}$ des Krylov-Raums $\mathcal{K}_{k+1}(\mathbf{A}, \underline{\mathbf{x}}_0)$.

```

 $\underline{\mathbf{u}}_1 \leftarrow \underline{\mathbf{x}}_0 / \|\underline{\mathbf{x}}_0\|_2$ 
for  $j = 1, 2, \dots, k$  do
   $\underline{\mathbf{w}} = \mathbf{A}\underline{\mathbf{u}}_j$ 
   $\alpha_j = \underline{\mathbf{u}}_j^T \underline{\mathbf{w}}$ 
   $\tilde{\mathbf{u}}_{j+1} = \underline{\mathbf{w}} - \alpha_j \underline{\mathbf{u}}_j$ 
  if  $j > 1$  then
     $\tilde{\mathbf{u}}_{j+1} \leftarrow \tilde{\mathbf{u}}_{j+1} - \underline{\mathbf{u}}_{j-1} \beta_{j-1}$ 
  end if
   $\beta_j = \|\tilde{\mathbf{u}}_{j+1}\|_2$ 
   $\underline{\mathbf{u}}_{j+1} = \tilde{\mathbf{u}}_{j+1} / \beta_j$ 
end for

```

Die Extraktion von Eigenwert/-vektorapproximationen erfolgt wie in Abschnitt 9.4.2 beschrieben.

Ähnlich wie beim CG-Verfahren werden in jeder Schleife von Algorithmus 9.11 eigentlich nur die letzten beiden Vektoren der Orthonormalbasis benötigt. Allerdings werden sowohl bei der Extraktion von Ritzvektoren als auch bei der Reorthogonalisierung (siehe Bemerkung 9.10) *alle* in \mathbf{U}_k bzw. \mathbf{U}_j enthaltenen Vektoren benötigt. Bei der Extraktion lässt sich dies noch relativ einfach vermeiden, indem die Vektoren extern (z.B. auf Festplatte) gespeichert werden oder das Lanczos-Verfahren einfach zwei Mal hintereinander durchgeführt wird, ein Mal um die Eigenwerte und -vektoren von \mathbf{H}_k auszurechnen und ein zweites Mal um die Matrix-Vektor-Multiplikationen $\underline{\mathbf{v}} = \mathbf{U}_k \underline{\mathbf{z}}$ für die Konstruktion der entsprechenden Ritzvektoren durchzuführen. Die Reorthogonalisierung ist eine wesentlich diffizilere Problematik; wir verweisen dazu auf die Fachliteratur, z.B. [J.K. Cullum, R.A. Willoughby. Lanczos algorithms for large symmetric eigenvalue computations: Theory. SIAM, 2002].

9.4.4 Konvergenzschranken*

Wir führen hier den Konvergenzbeweis nur für den kleinsten Eigenwert λ_1 einer symmetrischen Matrix \mathbf{A} . Zunächst benötigen wir noch einige Vorkenntnisse zu Winkeln zwischen Vektoren und Unterräumen. Sei $\underline{\mathbf{x}} \in \mathbb{R}^n$ und $\mathcal{U} \subset \mathbb{R}^n$ Unterraum. Dann hat $\underline{\mathbf{x}}$ die eindeutige Zerlegung

$$\underline{\mathbf{x}} = \underline{\mathbf{x}}_{\mathcal{U}} + \underline{\mathbf{x}}_{\mathcal{U}^\perp}, \quad \underline{\mathbf{x}}_{\mathcal{U}} \in \mathcal{U}, \quad \underline{\mathbf{x}}_{\mathcal{U}^\perp} \in \mathcal{U}^\perp,$$

wobei \mathcal{U}^\perp das orthogonale Komplement von \mathcal{U} ist. Der Winkel zwischen \underline{x} und \underline{u} ist dann gegeben durch

$$\tan \angle(\underline{x}, \mathcal{U}) := \frac{\|\underline{x}_{\mathcal{U}^\perp}\|_2}{\|\underline{x}_{\mathcal{U}}\|_2} = \min_{\underline{u} \in \mathcal{U}} \tan \angle(\underline{x}, \underline{u}).$$

Diese Definition erlaubt es uns abzuschätzen wie gut der Eigenvektor zum kleinsten Eigenwert im Krylovraum enthalten ist.

Lemma 9.12 Sei \mathbf{A} symmetrisch und \underline{x}_1 , $\|\underline{x}_1\|_2 = 1$, Eigenvektor zum kleinsten Eigenwert λ_1 , der Vielfachheit 1 habe. Dann gilt mit $\mathcal{U} = \mathcal{K}_k(\mathbf{A}, \underline{x}_0)$ die Abschätzung

$$\tan \angle(\underline{x}_1, \mathcal{U}) \leq 2\rho^{-k+1} \tan \angle(\underline{x}_1, \underline{x}_0),$$

wobei $\rho = 1 + 2(\lambda_2 - \lambda_1)/(\lambda_n - \lambda_1)$.

Beweis. Jeder Vektor $\underline{u} \in \mathcal{U}$ hat die Form $\underline{u} = q(\mathbf{A})\underline{x}_0$ mit $q \in \mathbb{P}_{k-1}$. Da \underline{x}_1 Eigenvektor ist, kommutiert $q(\mathbf{A})$ mit der Matrix $\underline{x}_1 \underline{x}_1^T$. Für \underline{u} gilt also die orthogonale Zerlegung

$$\underline{u} = q(\mathbf{A})\underline{x}_1 \underline{x}_1^T \underline{x}_0 + q(\mathbf{A})(\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0,$$

und damit

$$\begin{aligned} \tan \angle(\underline{x}_1, \underline{u}) &= \frac{\|q(\mathbf{A})(\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0\|_2}{|q(\mathbf{A})\underline{x}_1 \underline{x}_1^T \underline{x}_0|} \\ &= \frac{\|q(\mathbf{A})(\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0\|_2}{|q(\lambda_1)| \|(\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0\|_2} \cdot \frac{\|(\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0\|_2}{\|\underline{x}_1 \underline{x}_1^T \underline{x}_0\|_2} \\ &= \frac{\|q(\mathbf{A})\underline{y}_0\|_2}{|q(\lambda_1)|} \cdot \tan \angle(\underline{x}_1, \underline{x}_0) = \|\pi(\mathbf{A})\underline{y}_0\|_2 \cdot \tan \angle(\underline{x}_1, \underline{x}_0), \end{aligned}$$

mit $\underline{y}_0 := (\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0 / \|(\mathbf{I} - \underline{x}_1 \underline{x}_1^T)\underline{x}_0\|_2$ und $\pi(\lambda) = q(\lambda)/q(\lambda_1)$. Der Vektor \underline{y}_0 wird jetzt in der orthonormalen Eigenbasis $\{\underline{x}_1, \underline{x}_2, \dots, \underline{x}_n\}$ von \mathbf{A} entwickelt:

$$\underline{y}_0 = \alpha_2 \underline{x}_2 + \dots + \alpha_n \underline{x}_n, \quad \alpha_2^2 + \dots + \alpha_n^2 = 1,$$

wobei hier ausgenutzt wurde, dass \underline{y}_0 keine Komponente in \underline{x}_1 hat. Damit erhalten wir

$$\|\pi(\mathbf{A})\underline{y}_0\|_2^2 = \sum_{i=2}^n |\pi(\lambda_i)\alpha_i|^2 \leq \max_{i=2, \dots, n} |\pi(\lambda_i)|^2 \leq \max_{\lambda \in [\lambda_2, \lambda_n]} |\pi(\lambda)|^2$$

wobei $\lambda_1 < \lambda_2 \leq \dots \leq \lambda_n$ die (sortierten) Eigenwerte von \mathbf{A} sind. Da diese Beziehung für jedes $\underline{u} \in \mathcal{U}$ (und damit für jedes $\pi \in \mathbb{P}_{k-1}$ mit $\pi(\lambda_1) = 1$) gilt, erhalten wir insgesamt die Beziehung

$$\tan \angle(\underline{x}_1, \mathcal{U}) \leq \min_{\substack{\pi \in \mathbb{P}_{k-1} \\ \pi(\lambda_1)=1}} \max_{\lambda \in [\lambda_2, \lambda_n]} |\pi(\lambda)| \cdot \tan \angle(\underline{x}_1, \underline{x}_0). \quad (9.18)$$

Für das Tschebyscheff-Polynom $q(\lambda) = T_{k-1}((2\lambda - \lambda_2 - \lambda_n)/(\lambda_n - \lambda_2))$ erhält man $|q(\lambda)| \leq 1$ für $\lambda \in [\lambda_2, \lambda_n]$ und $|q(\lambda_1)| \geq \frac{1}{2}\rho^{k-1}$. Einsetzen von $\pi(\lambda) = q(\lambda)/q(\lambda_1)$ in (9.18) beweist die Behauptung. \square

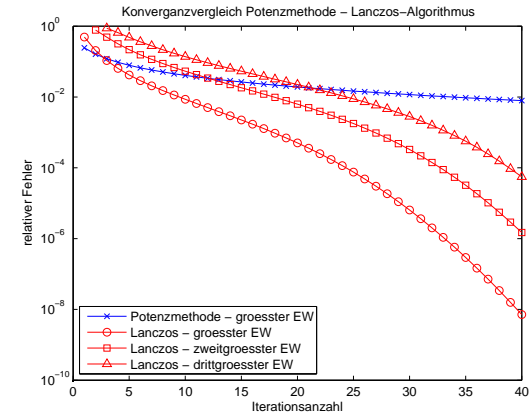
Als (nicht ganz unmittelbare) Folgerung von Lemma 9.12 erhalten wir, dass für den kleinsten zu $\mathcal{K}_k(\mathbf{A}, \underline{x}_0)$ gehörigen Ritzwert $\mu^{(k)}$ die folgende Abschätzung gilt:

$$0 \leq \mu^{(k)} - \lambda_1 \leq 4(\lambda_n - \lambda_1)\rho^{-2k+2} \tan^2 \angle(\underline{x}_1, \underline{x}_0).$$

Analoge Abschätzungen lassen sich für den grössten Eigenwert finden, und mit – entsprechenden Korrekturen bei den Konstanten – auch für den zweitkleinsten, zweitgrössten, usw. Für weitere Details verweisen wir auf die Fachliteratur, insbesondere das Buch [Y. Saad. Numerical Methods for Large Eigenvalue Problems. Halstead Press, 1992], das frei im Internet unter <http://www-users.cs.umn.edu/~saad/books.html> verfügbar ist.

9.4.5 Numerische Beispiele

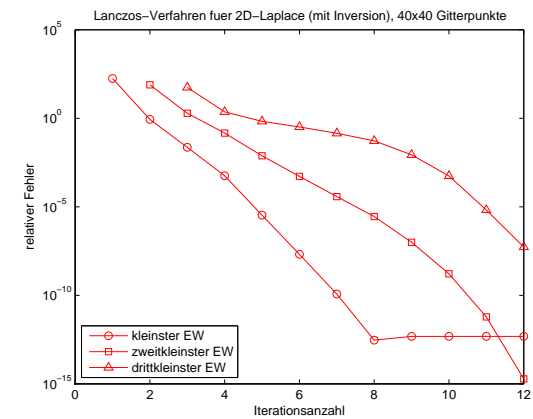
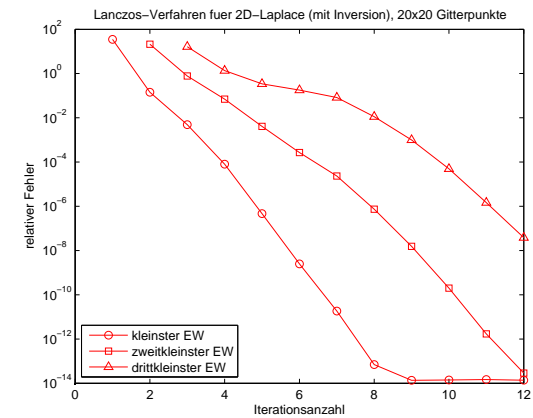
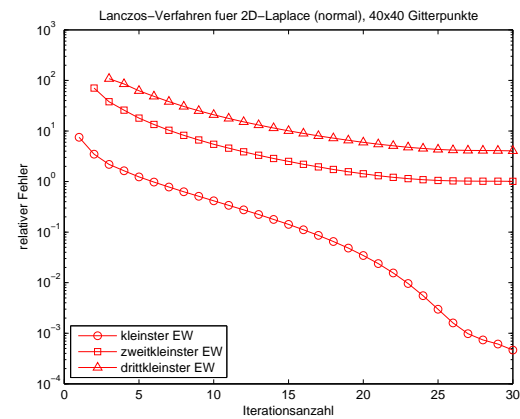
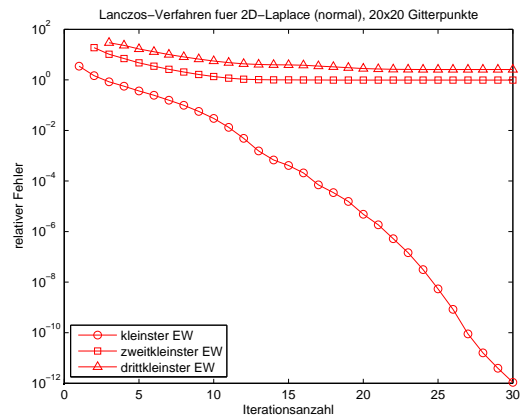
Zur Veranschaulichung der Konvergenzeigenschaften von Krylovraum-Verfahren, wollen wir im folgenden eine Reihe numerischer Experimente durchführen. Zunächst betrachten wir die Diagonalmatrix $\mathbf{A} = \text{diag}(1, 2, \dots, 100)$. Auf diese Matrix wenden wir zum einen die Potenzmethode und zum anderen den Lanczos-Algorithmus an, in beiden Fällen ausgehend vom Startvektor $(1, \dots, 1)^T$. Für den Lanczos-Algorithmus betrachten wir die drei grössten Ritz-Werte, im Falle der Potenzmethode verwenden wir den Rayleigh-Quotienten, um aus der aktuellen Iterierten eine Näherung an den grössten Eigenwert zu gewinnen. Die Resultate sind im folgenden Plot dargestellt.



In einem zweiten Experiment wird die drei kleinsten Eigenwerte der Matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & & -1 & 2 \end{bmatrix}$$

berechnen, welche eine Diskretisierung des 2D-Laplace-Operators mittels finiter Differenzen auf einem äquidistanten Gitter darstellt. Das Lanczos-Verfahren ausgehend von einem zufällig gewählten Startvektor ergab hier den folgenden Konvergenzverlauf:



Dargestellt sind jeweils die relativen Abweichungen der kleinsten Ritzwerte in jeder Iteration von der exakter kleinsten Eigenwerten. Die schlechte Konvergenz erklärt sich daraus, dass das Verfahren den zweit- und drittkleinsten Eigenwert nicht findet und stattdessen gegen grössere Eigenwerte konvergiert. Abhilfe schafft hier die Anwendung des Lanczos-Verfahrens auf die Inverse Matrix A^{-1} und die anschließende Invertierung der Ritzwerte. Der Plot zeigt hier ein deutlich besseres Verhalten.

Index

- (\cdot, \cdot) (Skalarprodukt), 13
- $\frac{3}{8}$ -Regel, 105
- \oplus (direkte Summe), 2
- \perp (orthogonal), 17
- \perp (orthogonales Komplement), 18
- A**-Norm, 162
- A**-Skalarprodukt, 162
- α_j (Quadraturgewichte), 104
- Ausgleichsproblem, 143
 - bei Rangdefekt, 156
 - bei vollem Spaltenrang, 144
 - im **A**-Skalarprodukt, 169
- Banach'scher Fixpunktsatz, 125
- Bandmatrix, 70
- baryzentrische Interpolationsformel, 77
- Basis, 2
- Bernstein-Polynome, 94
- Besselsche Quadraturformel, 107
- Bild, 7
- Bisektionsverfahren, 128
- BLAS, 5
- Blockmatrix, 4
- Broyden's Verfahren, 139
- Cayley-Hamilton, 170
- CG-Verfahren, 168
 - Algorithmus, 172
 - Konvergenz, 173
 - Verlust der Orthogonalität, 173
 - Vorkonditionierung, 174
- charakteristisches Polynom, 9
- Cholesky-Zerlegung, 67
 - Eindeutigkeit, 149
- column major format, 159
- cond(\cdot), 53
- C^p (p -mal diffbare Fkt), 1
- CRS-Format, 160
- dünnbesetzt, *siehe* sparse Matrix
- δ_{ij} (Kronecker-Symbol), 20
- det(\cdot) (Determinante), 7
- diag, 8
- Diagonaldominanz, 92, 163
 - und Regularität, 92
- Diagonale, 4
- Diagonalisierbarkeit, 21
- Diagonalmatrix, 8
- Dimension, 3
- direkte Summe, 2
- dividierte Differenz, 79
- Dreiecksmatrix, 8, 41
- Dreiecksungleichung, 11
- Eigenvektor, 9
- Eigenwert, 9
 - algebraische Vielfachheit, 21
 - geometrische Vielfachheit, 21
- Einheitsmatrix, 6
- Einheitsvektor, 3
- Einheitswurzel, 97
- e_j (Einheitsvektor), 3
- erzeugendes System, 2
- \mathbb{F} (Gleitpunktzahlen), 30
- $f[\cdot]$ (dividierte Differenz), 79
- FFT, 98
- Fibonacci-Folge, 132
- finite Differenz, 130
- Fixpunktgleichung, 122
 - eines linearen Gleichungssystems, 161
 - Konsistenz, 123
- Fixpunktiteration, 123
 - Konvergenz im linearen Fall, 162
 - und Konvergenz höherer Ordnung, 124, 127
 - und lineare Konvergenz, 124, 125
- flops, 42
- F_n (Fourier-Matrix), 98
- Fourier
 - Koeffizient, 96

-Transformation (diskret), 98
 Frobeniusnorm, 14

γ_n , 35
 Gauss'sche Quadraturformeln, 113
 Gauss-Seidel-Iteration, 163
 Gerschgorin-Kreise, 182
 Gleitpunktzahl, 30
 Darstellung im Rechner, 32
 denormalisiert, 31
 Gram-Schmidt-Algorithmus, 20, 147
 modifizierter, 149
 numerische Instabilität, 150
 und **QR**-Zerlegung, 148

H, 6
 Hermite'sche Quadraturformel, 108
 Hermite-Interpolation, 81
 hermitesch Transponierte, 6
 Hilbert-Matrix, 52
 Kondition, 59
 Horner-Schema, 74
 Householder-Reflexion, 150
 Hutfunktion, 3

IEEE 754(r), 31
 im(\cdot) (Bild), 7
 I_n (Einheitsmatrix), 6
 Inverse Iteration, 187
 Iterative Verbesserung, 65

Jacobi-Iteration, 162
 JOR, 165
 Jordan'sche Normalform, 21

$\mathcal{K}_k(\cdot, \cdot)$ (Krylov-Raum), 171
 $\kappa(\cdot)$ (Konditionszahl), 55
 ker(\cdot) (Kern), 7
 Kern, 7
 Kondition
 der Polynominterpolation, 84
 einer Funktion, 53
 einer Matrix, 55
 einer Nullstelle, 133
 Konditionszahl, 55
 Konvergenz, 121
 globale, 121
 höherer Ordnung, 122
 lineare, 122
 lokale, 121
 Kronecker-Symbol, 20

Krylov-Raum, 171

L_2 -Norm, 95
 L_2 -Skalarprodukt, 95
 Lagrange-Interpolation
 baryzentrisch, 77
 Basis, 75
 $\Lambda(\cdot)$ (Spektrum), 9
 Lambert-W-Funktion, 123
 Landau-Symbole, 38
 Lebesgue-Konstante, 85
 Legendre-Polynom, 113
 linear unabhängig, 2
LR-Zerlegung, 44
 Eindeutigkeit, 44
 Existenz, 48
 für Band-Matrizen, 71
 mit Pivotsuche, 61
 mit Vollpivotsuche, 65
 ohne Pivotsuche, 47

Matrix, 3
 -inverse, 6
 -multiplikation, 5
 ähnlich, 10
 hermitesch, 9
 invertierbar, 6
 normal, 9
 orthogonal, 9
 schief-symmetrisch, 9
 schwach besetzt, 69
 singulär, 6
 Speicherung im Rechner, 159
 symmetrisch, 9
 symmetrisch positiv definit, 67
 unitar, 9

Matrixnorm, 14
 induziert, 15
 konsistent, 14
 submultiplikativ, 16

Milne-Regel, 105
 Mittelpunktsregel, 103, 107
 summiert, 110
 und singuläre Integrale, 119

Monotonietest, 137
 Monte-Carlo-Methode, 120

Nachiteration, 65
 Neville's Schema, 77
 Newton-Cotes-Formeln

geschlossene, 104
 offene, 107
 Newton-Interpolation, 79–81
 Newton-Verfahren, 128
 affine Invarianz, 137
 gedämpftes, 138
 im Mehrdimensionalen, 133
 Konvergenz, 129, 135
 Niedrigrangapproximation, 157
 nnz(\cdot) (Anzahl Nichtnullen), 160
 Norm, 11, 14
 -äquivalenz, 13
 Euklidische, 11
 Normalengleichungen, 144
 Methode der, 144
 numerische Instabilität, 146

$O(\cdot)$ (Landau-Symbol), 38
 $o(\cdot)$ (Landau-Symbol), 38
 ω_n (Einheitswurzel), 97
 $\omega_{n+1}(\cdot)$ (Stützstellenfunktion), 76
 Operatornorm, 15
 Ordnung einer Quadraturformel, 112
 Orthogonalbasis, 17
 orthogonale Projektion, 18
 orthogonales Komplement, 18
 Orthogonalität, 17
 Orthonormalbasis, 17

Permutationsmatrix, 60
 \mathbb{P}_n (Raum der Polynome), 1
 p -Norm, 11, 15
 Polynominterpolation, *siehe auch* Lagrange-
 Interpolation
 Hermite-Interpolation, 81
 Interpolationsbedingungen, 73
 Interpolationsfehler, 76, 81
 Kondition, 84
 Lösbarkeit, 75
 Neville's Schema, 77
 Newton-Interpolation, 79–81
 Stützstellen, 73
 Tschebyscheff-Interpolation, 86
 und Quadratur, 103

Potenzmethode, 184
 P_π (Permutationsmatrix), 60
 Pseudoinverse, 157

QR-Zerlegung
 ökonomische, 148

Householder-basiert, 150
 Implementierungsaspekte, 155
 und Ausgleichsproblem, 147, 155
 und Gram-Schmidt-Algorithmus,
 148
 vollständige, 148
 Quasi-Newton-Verfahren, 139

Rückwärtsfehler
 bei **LR**-Zerlegung, 51
 bei Cholesky-Zerlegung, 69
 bei Matrix-Vektor-Multiplikation,
 36
 bei Vorwärtssubstitution, 43
 beim Skalarprodukt, 35
 Rückwärtssubstitution, 42
 Rang, 7
 numerischer, 158
 Rayleigh-Quotient, 185
 rd(\cdot) (Runden), 32
 Residuum
 bei überbestimmten Gleichungs-
 systemen, 143
 bei Eigenwertproblemen, 185
 bei linearen Gleichungssystemen,
 52
 bei nichtlinearen Gleichungssystemen,
 137
 $\rho(\cdot)$ (Spektralradius), 10
 Richardson-Verfahren, 167
 Romberg-Schema, 110
 row major format, 159
 Runden, 32
 Rundungseinheit, 33
 Rundungsfehler
 bei elementaren Operationen, 34
 beim Runden, 33
 beim Skalarprodukt, 34
 Runge-Phänomen, 84

Schur-Form, 22
 reell, 22
 Sekantenverfahren, 130
 Konvergenz, 130
 Sherman-Morrison-Formel, 92
 Simpson-Regel, 105
 summiert, 108
 Singulärvektor, 24
 Singulärwert, 24
 Singulärwertzerlegung, 23

- kompakt, 24
 - und Ausgleichsproblem, 156
- Skalarprodukt, 13
 - Euklidisches, 12
- SOR, 166
- Spaltenvektor, 4
- $\text{span}\{\cdot\}$, 2
- sparse grids, 120
- sparse Matrix, 69
 - Speicherung im Rechner, 160
- Spektralradius, 10
 - und Matrixnormen, 57
- Spektralzerlegung, 22
- Spektrum, 9
- Spline
 - Bézier-, 94
 - kubisch, 90
- Spline-Interpolation, 88–93
 - natürlich, 90
 - periodisch, 90
 - vollständig, 90
- Splitting-Verfahren, 161
- Spur, 6
- Störungsanalyse
 - bei linearen Gleichungssystemen, 56
 - bei Matrixinversion, 54
 - von Eigenwertproblemen, 183
- Summierte Quadraturformeln, 108
- SVD, *siehe* Singulärwertzerlegung

- T, 6
- Teilraum, 2
- T_n (trigonometrische Polynome), 97
- Transponierte, 6
- Trapezregel, 103, 105
 - summiert, 108
 - und periodische Funktionen, 117
- Trigonometrische Interpolation
 - FFT, 98
 - Interpolationsaufgabe, 97
 - und Quadratur, 118
- Tschebyscheff-Interpolation, 86
 - Lebesgue-Konstante, 88
- Tschebyscheff-Polynom, 86
 - Nullstellen, 87

- $u(\mathbb{F})$ (Rundungseinheit), 33
- Untermatrix, 4
- Unterraum
 - affiner, 169
- Vektornorm, 11
- Vektorraum, 1
 - der Polynome, 1
 - differenzierbarer Funktionen, 1
 - stückweiser linearer Funktionen, 2
- Vielfachheit einer Nullstelle, 133
- Vorkonditionierer
 - bei CG-Verfahren, 174
 - bei Splitting-Verfahren, 161
- Vorwärtsfehler
 - bei Matrix-Vektor-Multiplikation, 36
 - beim Skalarprodukt, 35
- Vorwärtssubstitution, 42

- Wilkinson-Beispiel, 181

- Zahl
 - Basis, 29
 - Exponent, 29
 - Mantisse, 30
 - wissenschaftliche Darstellung, 29
- Zeilenvektor, 4