

# Numerische Mathematik I

Eine Zusammenfassung von Bernhard Kabelka  
zur Vorlesung von Prof. Carstensen im WS 2002/03

Version 1.03, 15. März 2004

Es sei ausdrücklich betont, dass

- (1) dieses Essay ohne das Wissen und die Mitarbeit von Prof. Carstensen entstanden ist,
- (2) trotz großer Anstrengungen seitens des Autors, eine möglichst fehlerfreie und vollständige Zusammenfassung zu liefern, sich Fehler eingeschlichen haben könnten (sollte jemand einen Fehler entdecken, so bittet der Autor um Benachrichtigung, vorzugsweise per eMail an [bernhard@kabelka.net](mailto:bernhard@kabelka.net)),
- (3) die Lektüre dieser Zusammenfassung keinesfalls den persönlichen Besuch der Vorlesung bzw. das Studium des Skriptums ersetzen, sondern bestenfalls ergänzen kann.

Die aktuelle Version dieser Datei ist erhältlich unter:

<http://fsmat.at/~bkabelka/math/numerik/download/num1.pdf>

<http://fsmat.at/~bkabelka/math/numerik/download/num1.ps.gz>

# Inhaltsverzeichnis

<b>I Fehlerbetrachtung</b>	<b>1</b>
1 Modellierungsfehler	1
2 Datenfehler	1
3 Verfahrensfehler	1
4 Rundungsfehler	2
5 Rundungsfehleranalyse	3
5.1 Unvermeidbarer Fehler und gutartige Algorithmen . . . . .	3
5.2 Fehlerfortpflanzung im linearen Modell . . . . .	4
<b>II Numerik der Linearen Algebra</b>	<b>5</b>
1 Modellbildungsprozesse	5
2 Fundamentals	6
2.1 Lineares Ausgleichsproblem und Pseudoinverse . . . . .	6
2.2 Singulärwertzerlegung . . . . .	6
2.3 QR-Zerlegung . . . . .	7
2.4 Cholesky-Zerlegung . . . . .	7
3 Traditionelle Lösung von $Ax = b$	7
3.1 „Naive“ Gauß-Elimination . . . . .	7
3.2 Pivotisierung . . . . .	8
4 Linearer Ausgleich	9
4.1 Householder-Matrizen . . . . .	9
4.2 Householder-Algorithmus . . . . .	9
4.3 Kondition von linearen Ausgleichsproblemen . . . . .	10

<b>5</b>	<b>Iterative Methoden</b>	<b>10</b>
5.1	Eigenwertalgorithmen . . . . .	10
5.2	Algorithmische Berechnung der Singulärwertzerlegung . . . . .	13
5.3	Iterative Lösung von $Ax = b$ . . . . .	14
<b>Anhang</b>		<b>16</b>
A	Orthogonalisierungsverfahren von Gram-Schmidt . . . . .	16
B	Matrix-Faktorisierungen . . . . .	17
C	Gauß-Algorithmus . . . . .	18
D	QR-Verfahren . . . . .	21
E	cg-Verfahren . . . . .	22

## Teil I

# Fehlerbetrachtung

In der Numerischen Mathematik sind Fehler unvermeidlich – allerdings nicht Fehler im Sinne von Irrtümern, sondern Modellierungsfehler, Datenfehler, Verfahrensfehler und Rundungsfehler.

## 1 Modellierungsfehler

Oft geht es darum, einen bestimmten realen Vorgang aufgrund von physikalischen, chemischen, biologischen, wirtschaftlichen, ... Prinzipien und Gesetzen zu modellieren. Dabei werden aber meistens nicht alle Einflüsse berücksichtigt, da das Modell sonst zu komplex werden würde. So entstehen Modellfehler, deren Einfluss abgeschätzt werden müsste.

## 2 Datenfehler

Eine weitere Fehlerquelle liegt in den Messfehlern, mit denen die verwendeten Daten üblicherweise behaftet sind. Man rechnet also nicht mit dem exakten Wert  $x$ , sondern mit einem verfälschten  $\tilde{x}$ , was natürlich auch das Ergebnis verfälscht.

Nun stellt sich die Frage, wie groß der Fehler im Ergebnis in Abhängigkeit von der Größe des Datenfehler ist. Dieses Verhältnis wird durch die **Konditionszahl** eines Problems beschrieben: Ist  $\mathcal{A}$  die Problemlösungsabbildung, die einem Datensatz  $D$  die Lösung  $\mathbf{y}$  zuordnet, so ist die Lipschitzkonstante von  $\mathcal{A}$  genau die gesuchte Konditionszahl  $k$ , für die dann gilt:

$$\|\mathbf{y} - \bar{\mathbf{y}}\|_{\text{Bildraum}} \leq k \cdot \|D - \bar{D}\|_{\text{Urbildraum}}$$

für alle Datensätze  $D$  und  $\bar{D}$  sowie  $\mathbf{y} = \mathcal{A}(D)$  und  $\bar{\mathbf{y}} = \mathcal{A}(\bar{D})$ .

## 3 Verfahrensfehler

Die numerischen Werte von irrationalen Zahlen wie  $\sqrt{2}$ ,  $\pi$  oder  $e$ , von Nullstellen von Polynomen höheren Grades, etc. lassen sich bekanntlich *nicht* durch Auswerten einer Formel mit endlich vielen Rechenschritten exakt berechnen. Man muss also approximative Verfahren heranziehen, die alle einen Fehler (eben den Verfahrensfehler) hinterlassen müssen.

Ist nun  $\mathcal{A}$  eine Problemlösungsabbildung und  $\mathcal{A}_n$  eine Realisierung von  $\mathcal{A}$ , kann der Verfahrensfehler  $\mathcal{A}(D) - \mathcal{A}_n(D)$  entweder

- **a priori**, d. h. nur aufgrund von  $\mathcal{A}$  und  $\mathcal{A}_n$ , oder
- **a posteriori**, d. h. aufgrund von  $\mathcal{A}(D)$ ,  $\mathcal{A}_n(D)$  und weiteren ausführbaren Operationen mit diesen Größen

abgeschätzt werden.

## 4 Rundungsfehler

Zu einer Basis  $b \in \{2, 3, \dots\}$  und Mantissenlänge  $t \in \mathbb{Z}^+$  sowie für Exponentialschranken  $e_{\min} < 0 < e_{\max}$  ist die Menge der **normalisierten Maschinenzahlen** gegeben durch

$$\begin{aligned} \mathbb{F} &= \mathbb{F}(b, t, e_{\min}, e_{\max}) = \\ &= \{0\} \cup \left\{ \pm \left( \sum_{k=1}^t a_k \cdot b^{-k} \right) \cdot b^e \mid \begin{array}{l} a_1, \dots, a_t \in \{0, 1, \dots, b-1\}, \\ a_1 \neq 0, \quad e \in \mathbb{Z}, \quad e_{\min} \leq e \leq e_{\max} \end{array} \right\} \end{aligned}$$

und die Menge der **denormalisierten Maschinenzahlen** gegeben durch

$$\begin{aligned} \widehat{\mathbb{F}} &= \widehat{\mathbb{F}}(b, t, e_{\min}, e_{\max}) = \\ &= \mathbb{F} \cup \left\{ \pm \left( \sum_{k=2}^t a_k \cdot b^{-k} \right) \cdot b^{e_{\min}} \mid a_2, \dots, a_t \in \{0, 1, \dots, b-1\} \right\} \end{aligned}$$

Es gilt dann

$$\begin{aligned} x_{\min} &:= \min_{f \in \mathbb{F}} |f| = b^{e_{\min}-1} & x_{\max} &:= \max_{f \in \mathbb{F}} |f| = b^{e_{\max}} \cdot (1 - b^{-t}) \\ \widehat{x}_{\min} &:= \min_{f \in \widehat{\mathbb{F}}} |f| = b^{e_{\min}-t} \end{aligned}$$

Die **Maschinengenauigkeit**  $\text{eps}$  ist dann definiert als  $\text{eps} := \frac{1}{2} \cdot b^{1-t}$ .

Möchte man nun eine reelle Zahl  $x$  durch eine Maschinenzahl darstellen, so gibt es folgende Möglichkeiten:

- (1) **Runden**, d. h. man bestimmt jene Zahl  $\text{rd}(x) \in \mathbb{F}$  mit

$$|\text{rd}(x) - x| = \min_{y \in \mathbb{F}} |y - x|$$

- (2) **Abschneiden**, d. h. man ersetzt  $x = \sigma \cdot \left( \sum_{k=1}^{\infty} a_k \cdot b^{-k} \right) \cdot b^e \in \mathbb{R}$  durch

$$\text{tc}(x) = \sigma \cdot \left( \sum_{k=1}^t a_k \cdot b^{-k} \right) \cdot b^e$$

Es gilt:

$$\left| \frac{\text{rd}(x) - x}{x} \right| \leq \text{eps} \quad \text{bzw.} \quad \left| \frac{\text{tr}(x) - x}{x} \right| \leq 2\text{eps}$$

Für das Rechnen am Computer gibt es ebenfalls zwei Möglichkeiten:

- (1) Bei der **Gleitpunktarithmetik** rechnet man mit Maschinenzahlen, wobei man beachten muss, dass  $\mathbb{F}$  selbst bezüglich der elementaren Operationen  $+$ ,  $-$ ,  $\cdot$  und  $\div$  *nicht* abgeschlossen ist. Das Ergebnis  $\text{op}(a, b)$  einer solchen Operation  $\text{op} \in \{+, -, \cdot, \div\}$  muss also wieder in eine Maschinenzahl umgewandelt werden, wodurch sich Rundungsfehler akkumulieren. Möglicherweise ist sogar  $|\text{op}(a, b)| < x_{\min}$  bzw.  $|\text{op}(a, b)| > x_{\max}$  (dann spricht man von **Unter-** bzw. **Überlauf**), und die Rechnung muss abgebrochen werden.
- (2) Bei der **Intervallarithmetik** rechnet man mit Intervallen, in denen die exakte Zahl enthalten ist. Ungenauigkeiten äußern sich dann in langen Intervallen.

Ein großes Problem (das es zu vermeiden gilt) stellt die **Auslöschung** dar, die bei der Subtraktion von fast gleichen Zahlen auftritt: In diesem Fall heben sich die führenden Stellen weg, und das Ergebnis ist daher deutlich ungenauer als die beiden Ausgangswerte.

## 5 Rundungsfehleranalyse

### 5.1 Unvermeidbarer Fehler und gutartige Algorithmen

Ein **Algorithmus** ist eine Komposition von endlich vielen elementaren Operationen (d. h.  $+$ ,  $-$ ,  $\times$ ,  $\div$ ,  $\sqrt{\quad}$ ,  $\sin$ ,  $\cos$ ,  $\exp$ ,  $\dots$ ).

Hat man einen Algorithmus  $\varphi : D \rightarrow \mathbb{R}^m$  ( $D \subseteq \mathbb{R}^n$  offen) mit

$$\varphi = \varphi^{(J)} \circ \varphi^{(J-1)} \circ \dots \circ \varphi^{(0)}$$

gegeben, wobei für  $j = 1, 2, \dots, J$  gelte

$$\varphi^{(j)} : D_j \rightarrow D_{j+1} \quad \text{mit} \quad D_j \subseteq \mathbb{R}^{n_j} \text{ offen,} \quad n_j \in \mathbb{N}$$

sowie  $D_0 = D$ ,  $n_0 = n$  und  $n_{J+1} = m$ , so erfolgt die Auswertung dieses Algorithmus auf einer Rechenmaschine mit der Maschinengenauigkeit **eps** durch die dort implementierte Ersatzabbildung

$$\text{gl}(\varphi) = \text{gl}(\varphi^{(J)}) \circ \text{gl}(\varphi^{(J-1)}) \circ \dots \circ \text{gl}(\varphi^{(0)})$$

Ist nun  $\varphi$  bei  $x$  differenzierbar, so heißt (der Vektor)

$$\text{eps} \cdot (|D\varphi(x)| \cdot |x| + |y|) \quad | \cdot | \text{komponentenweise}$$

der **unvermeidbare Fehler** von  $y = \varphi(x)$ .

Bei der Auswertung von  $\varphi$  muss also mit einem relativen Fehler von

$$\frac{\|\varphi(x) - \varphi(\tilde{x})\|_2}{\|\varphi(x)\|_2} \leq \text{eps} \cdot \left(1 + \frac{\|D\varphi(x)\|_2 \cdot \|x\|_2}{\|\varphi(x)\|_2}\right)$$

gerechnet werden.  $\frac{\|D\varphi(x)\|_2 \cdot \|x\|_2}{\|\varphi(x)\|_2}$  wird daher gelegentlich Konditionszahl genannt.

Schließlich wird ein Algorithmus **gutartig** (zur Berechnung von  $y = \varphi(x)$ ) genannt, falls der Fehler  $\varphi(x) - \text{gl}(\varphi(x))$  mit dem unvermeidlichen vergleichbar ist, d. h. in derselben Größenordnung liegt.

## 5.2 Fehlerfortpflanzung im linearen Modell

Gegeben sei ein Algorithmus  $\varphi = \varphi^{(J)} \circ \varphi^{(J-1)} \circ \dots \circ \varphi^{(0)}$  und dessen Computerrealisierung  $\text{gl}(\varphi) = \text{gl}(\varphi^{(J)}) \circ \text{gl}(\varphi^{(J-1)}) \circ \dots \circ \text{gl}(\varphi^{(0)})$ , der bei  $x$  ausgewertet werden soll. Es sei  $\tilde{x} = \text{gl}(x)$  und damit der Anfangsfehler gegeben durch  $x - \tilde{x}$ . Weiters gelte

$$\varphi^{(j)} - \text{gl}(\varphi^{(j)}) = E_{j+1} \cdot \varphi^{(j)}(\tilde{x}_j)$$

wobei  $E_{j+1}$  eine Diagonalmatrix mit Einträgen in  $[-\text{eps}, \text{eps}]$  bezeichne. Ferner bezeichne  $\tilde{x}_0 := \tilde{x}$  und  $x_0 := x$ . Schließlich gelte für  $j = 0, 1, \dots, J$

$$\tilde{x}_{j+1} := \text{gl}(\varphi^{(j)}(\tilde{x}_j)) \quad x_{j+1} := \text{gl}(\varphi^{(j)}(x_j))$$

sowie  $x_{J+1} =: y$  und  $\tilde{x}_{J+1} =: \tilde{y}$ .

Schließlich bezeichne  $\psi^{(j)} := \varphi^{(J)} \circ \varphi^{(J-1)} \circ \dots \circ \varphi^{(j)}$  die Resteabbildung und  $\varphi^{(j)}$  sei bei  $x_j$  differenzierbar.

Dann gilt für den Fehler  $y - \tilde{y} = \varphi(x) - \text{gl}(\varphi(\tilde{x}))$  die Abschätzung

$$\begin{aligned} y - \tilde{y} &= E_{J+1} \cdot y + D\psi^{(J)}(x_J) \cdot E_J \cdot x_J + D\psi^{(J-1)}(x_{J-1}) \cdot E_{J-1} \cdot x_{J-1} + \\ &\quad + \dots + D\psi^{(1)}(x_1) \cdot E_1 \cdot x_1 + D\varphi(x) \cdot (x - \tilde{x}) + o(\text{eps}) \end{aligned}$$

$D\psi^{(j)}(x_j)$  ist dann der **Verstärkungsfaktor** zu einem Fehler im Rechenschritt zur Berechnung von  $x_j$  und  $D\varphi(x)$  ist der Verstärkungsfaktor zum Anfangsfehler.

Damit obige Abschätzung gültig ist, muss gelten:

- (1)  $J$  bzw.  $n_j$  genügend klein sein (d. h.  $J \cdot \text{eps} = O(\text{eps})$ )
- (2)  $|D\varphi^{(j)}(x_j)|$  nur moderat groß
- (3)  $|x - \tilde{x}|$ , etc. klein gegenüber  $\|D\varphi^{(j)}(x_j)\|_2$

Im Allgemeinen erweist sich das Modell (in der Praxis) – trotz obiger einschränkender Voraussetzungen – oft als sehr leistungsfähig.

## Teil II

# Numerik der Linearen Algebra

## 1 Modellbildungsprozesse

Bei vielen Anwendung treten in der Praxis Problemstellungen aus der linearen Algebra zu Tage, die (oft mit Computerhilfe) gelöst werden müssen. Zu diesem Zweck müssen Algorithmen entwickelt werden, die einerseits schnell und andererseits numerisch stabil sind.

Wenn beispielsweise die Daten  $(x_1, f_1), \dots, (x_n, f_n)$  in  $\mathbb{R}^2$  durch ein quadratisches Polynom  $p(x) = a_2x^2 + a_1x + a_0$  approximiert werden sollen, so lässt sich das durch **Linearen Ausgleich** bewerkstelligen:

$$g(a_0, a_1, a_2) := \sum_{j=1}^n (f_j - p(x_j))^2 \rightarrow \text{MIN}. \quad (1)$$

Definiert man für  $P_j(z) := z^j$  eine **Vandermonde-Matrix** als

$$V \begin{pmatrix} P_0 & \dots & P_{n-1} \\ t_1 & \dots & t_m \end{pmatrix} = \begin{pmatrix} P_k(x_j) \\ j=1, \dots, m \end{pmatrix}_{k=0, \dots, n-1} = \begin{pmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \dots & t_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_m & t_m^2 & \dots & t_m^{n-1} \end{pmatrix}$$

so kann man  $g(a_0, a_1, a_2)$  schreiben als

$$g(a_0, a_1, a_2) = \left\| \begin{pmatrix} f_1 & \dots & f_n \end{pmatrix}^T - V \cdot \begin{pmatrix} a_0 & a_1 & a_2 \end{pmatrix}^T \right\|_2^2$$

wobei  $V = V \begin{pmatrix} P_0 & P_1 & P_2 \\ x_1 & \dots & x_m \end{pmatrix}$ .

Als notwendige Bedingung für (1) erhält man durch Differenzieren (und einigen einfachen Umformungen) schließlich die **Gauß'sche Normalengleichung**

$$V^T V \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = V^T \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}$$

Auch die Diskretisierung von gewöhnlichen und partiellen (linearen) Differentialgleichungen bzw. Randwertproblemen führt beispielsweise auf lineare Gleichungssysteme mit strukturierten Koeffizientenmatrizen.

## 2 Fundamentals

### 2.1 Lineares Ausgleichsproblem und Pseudoinverse

Ist  $A \in \mathbb{R}^{m \times n}$  und  $b \in \mathbb{R}^m$  ( $m, n \in \mathbb{N}$ ) gegeben, so heißt  $x \in \mathbb{R}^n$  eine Lösung des **linearen Ausgleichsproblems** zu  $(A, b)$ , wenn gilt

$$\|Ax - b\|_2 = \min_{y \in \mathbb{R}^n} \|Ay - b\|_2$$

Sofern  $x \in \mathbb{R}^n$  unter all diesen Lösungen die Norm  $\|x\|_2$  minimiert, so spricht man von einer **Minimum-Norm-Lösung** des linearen Ausgleichsproblems zu  $(A, b)$ .

Es zeigt sich, dass jedes lineare Ausgleichsproblem genau eine Minimum-Norm-Lösung  $x$  besitzt und dass alle anderen Lösungen durch  $x + \ker A$  gegeben sind.

Die Abbildung, die  $b$  (für fixiertes  $A$ ) auf die Minimum-Norm-Lösung abbildet, ist linear und wird durch die sogenannte **Pseudoinverse**  $A^+$  von  $A$  repräsentiert. Diese wird auch durch folgende vier Bedingungen eindeutig charakterisiert:

- (1)  $A \cdot A^+ \cdot A = A$
- (2)  $A^+ \cdot A \cdot A^+ = A^+$
- (3)  $A \cdot A^+$  ist symmetrisch
- (4)  $A^+ \cdot A$  ist symmetrisch

### 2.2 Singulärwertzerlegung

Ist  $A \in \mathbb{R}^{m \times n}$  gegeben, so existieren orthogonale Matrizen  $U \in \mathbb{R}^{m \times m}$  und  $V \in \mathbb{R}^{n \times n}$ , sodass  $A = U\Sigma V^T$  mit einer Diagonalmatrix

$$\Sigma = U^T A V = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}$$

wobei  $r = \text{rg } A$  und  $\sigma_1, \sigma_2, \dots, \sigma_r > 0$ . Setzt man  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$  fest, so ist  $\Sigma$  sogar eindeutig. Die so erhaltende Zerlegung von  $A = U\Sigma V^T$  heißt **Singulärwertzerlegung** von  $A$ , und  $\sigma_1, \sigma_2, \dots, \sigma_r$  nennt man die **Singulärwerte** von  $A$  (das sind genau die Wurzeln aus den positiven Eigenwerten von  $A^T A$ ).

Ist  $A = U\Sigma V^T$  die Singulärwertzerlegung von  $A$ , so gilt:  $A^+ = V\Sigma^+ U^T$ .

## 2.3 QR-Zerlegung

Ist  $A \in \mathbb{R}^{m \times n}$  spaltenregulär (d. h.  $\text{rg } A = n \leq m$ ), so existiert genau eine orthogonale Matrix  $Q \in \mathbb{R}^{m \times m}$  (d. h. eine Matrix mit  $Q^T Q = 1$ ) und eine obere Dreiecksmatrix  $R \in \mathbb{R}^{m \times n}$  mit positiven Diagonaleinträgen  $R_{11}, R_{22}, \dots, R_{nn} > 0$ , sodass gilt:  $A = QR$ . Diese Zerlegung nennt man **QR-Zerlegung**.

Eine theoretische Möglichkeit zur Berechnung der QR-Zerlegung einer Matrix  $A$  liefert das **Orthogonalisierungsverfahren von Gram-Schmidt** (siehe Anhang A.1):

$$Q := (q_1 \ \cdots \ q_m) \in \mathbb{R}^{m \times m}$$
$$R_{kj} := \begin{cases} 0 & \text{für } k > j \\ \|\tilde{q}_j\|_2 & \text{für } k = j \\ a_j \cdot q_k & \text{für } k < j \end{cases}$$

Das Verfahren ist in der Praxis (für Computerrechnungen) allerdings ungeeignet, da die Orthogonalität der konstruierten Vektoren sehr stark in die Berechnung eingeht, durch Rundungsfehler aber verloren gehen kann.

Man sollte daher bestenfalls das **modifizierte Orthogonalisierungsverfahren von Gram-Schmidt** (siehe Anhang A.2) verwenden, das in jedem Teilschritt eine numerisch günstigere orthogonale Projektion durchführt.

## 2.4 Cholesky-Zerlegung

Eine Matrix  $A \in \mathbb{R}^{n \times n}$  heißt **SPD-Matrix**, wenn sie symmetrisch und positiv definit ist, d. h. es gilt  $A^T = A$  und für alle  $x \in \mathbb{R}^n \setminus \{0\}$  ist  $x^T A x > 0$ .

Ist  $A \in \mathbb{R}^{n \times n}$  eine SPD-Matrix, so existiert genau eine untere Dreiecksmatrix  $U \in \mathbb{R}^{n \times n}$  mit positiven Diagonaleinträgen  $U_{11}, U_{22}, \dots, U_{nn} > 0$ , sodass gilt:  $A = UU^T$ . Diese Zerlegung nennt man **Cholesky-Zerlegung**.

Nützlich ist die Cholesky-Zerlegung beispielsweise bei der Gauß'schen Normalgleichung  $V^T V x = V^T b$ . Bestimmt man nämlich eine Cholesky-Zerlegung von  $V^T V$ , was entweder über die QR-Zerlegung von  $V$  – sogar ohne explizite Berechnung von  $V^T V$  – oder mittels eines Algorithmus (siehe Anhang B.3) möglich ist, so kann man durch zweimaliges Lösen eines gestaffelten Gleichungssystems  $x$  bestimmen.

# 3 Traditionelle Lösung von $Ax = b$

## 3.1 „Naive“ Gauß-Elimination

Gleichungssysteme mit einer regulären unteren bzw. oberen Dreiecksmatrix als Koeffizientenmatrix sind durch Vorwärts- bzw. Rückwärtssubstitution direkt (mit je  $n^2$  Operationen) zu lösen.

Die Idee des **Gauß-Algorithmus'** (siehe Anhang C.1) ist daher, die (reguläre) Koeffizientenmatrix  $A$  auf die Form  $A = LR$  zu bringen, wobei  $L$  bzw.  $R$  eine untere bzw. obere Dreiecksmatrix darstellt. Ist der Algorithmus durchführbar (d. h.  $A_{kk}^{(k)} \neq 0$  für  $k = 1, 2, \dots, n-1$ ), so kann man ein  $n \times n$ -Gleichungssystem mit Hilfe von  $\frac{2}{3}n^3 + O(n^2)$  Operationen lösen.

Gleichzeitig liefert der Gauß-Algorithmus (sofern er durchführbar ist) eine **LR-Zerlegung** von  $A$  (d. h. eine Darstellung der Form  $A = LR$ , wobei  $L$  bzw.  $R$  eine untere bzw. obere Dreiecksmatrix ist) vermöge

$$L_{jk} := \begin{cases} \frac{A_{jk}^{(k)}}{A_{kk}^{(k)}} & \text{für } 1 \leq k < j \leq n \\ \delta_{jk} & \text{für } 1 \leq j \leq k \leq n \end{cases}$$

$$R := \begin{pmatrix} A_{11}^{(1)} & A_{12}^{(1)} & \cdots & A_{1n}^{(1)} \\ 0 & A_{22}^{(2)} & \cdots & A_{2n}^{(2)} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{nn}^{(n)} \end{pmatrix}$$

Ist  $A \in \mathbb{R}^{n \times n}$  **strikt diagonaldominant**, d. h. es gilt

$$\sum_{\substack{j=1 \\ j \neq k}}^n |A_{kj}| < |A_{kk}| \quad \forall k \in \{1, 2, \dots, n\}$$

so ist der Gauß-Algorithmus durchführbar.

Oft hat man mit nur schwach besetzten Matrizen zu tun, d. h. nur „wenige“ Elemente sind ungleich Null. Diese Matrizen werden im **Sparse-Format** gespeichert, d. h. abgespeichert wird  $(j, k, A_{jk})$  für alle NNE (Nicht-Null-Elemente). Führt man dann allerdings den Gauß-Algorithmus durch, so kommt es zum **Fill-In**, d. h. es entstehen zahlreiche neue NNE, die abgespeichert werden müssen. Die sogenannte **Skyline-Technik** berücksichtigt dieses Phänomen, indem nur die Diagonalelemente und solche  $A_{jk}$ , für die entweder ein Eintrag  $A_{jl} \neq 0$  (mit  $1 \leq l \leq k < j$ ) oder ein Eintrag  $A_{lk} \neq 0$  (mit  $1 \leq l \leq j < k$ ) ist, gespeichert werden.

### 3.2 Pivotisierung

Wie bereits erwähnt, ist der „naive“ Gauß-Algorithmus nicht durchführbar, wenn für ein  $k \in \{1, 2, \dots, n-1\}$  gilt:  $A_{kk}^{(k)} = 0$ .

Um diese Schwäche zu reparieren, bedient man sich der **Pivotisierung**, d. h. man führt vor dem  $k$ -ten Eliminationsschritt eine geeignete Zeilen- bzw. Spaltenvertauschung durch:

- (1) Bei der **Spaltenpivotsuche** (siehe Anhang C.2) beschränkt man sich auf Zeilenvertauschungen (die eine Umnummerierung der Gleichungen repräsentieren), indem man die  $k$ -te Zeile durch jene Zeile  $p_k$  ersetzt, für die gilt:

$$|A_{p_k k}^{(k)}| = \max_{j=k}^n |A_{jk}^{(k)}|$$

- (2) Bei der **Totalpivotsuche** (siehe Anhang C.3) führt man in jedem Schritt sowohl eine Zeilen- als auch eine Spaltenvertauschung durch, indem man die  $k$ -te Zeile mit der  $l$ -ten Zeile und die  $k$ -te Spalte mit der  $m$ -ten Spalte vertauscht, wobei gelten soll:

$$|A_{lm}^{(k)}| = \max_{\lambda, \mu=k}^n |A_{\lambda\mu}^{(k)}|$$

## 4 Linearer Ausgleich

### 4.1 Householder-Matrizen

Für einen Vektor  $w \in \mathbb{R}^n$  mit  $|w| = 1$  heißt

$$H := 1 - 2 \cdot w \otimes w \in \mathbb{R}^{n \times n}$$

**Householder-Matrix** von  $w$  und die zugehörige lineare Abbildung **Householder-Transformation**, wobei  $w \otimes w := ww^T$ .

Geometrisch gesehen ist  $H$  eine Spiegelung an der Hyperebene durch 0 mit Normalvektor  $w$ .

Man definiert die Householder-Elimination für  $x$  als

$$\text{HE4}(x) := 1 - 2 \cdot w \otimes w \quad \text{für} \quad w := \frac{x + \sigma \cdot e_1}{|x + \sigma \cdot e_1|} \quad \text{mit} \quad \sigma = \pm|x|$$

wobei das Vorzeichen von  $\sigma$  so zu wählen ist, dass der Nenner möglichst groß wird. Dann gilt  $\text{HE4}(x) \cdot x = -\sigma \cdot e_1$ .

### 4.2 Householder-Algorithmus

Mit Hilfe von  $\text{HE4}(x)$  kann man den **Householder-Algorithmus** (siehe Anhang B.1) zur Berechnung der QR-Zerlegung einer Matrix  $A \in \mathbb{R}^{m \times n}$  (mit vollem Spaltenrang) definieren, der immer dann durchführbar ist, wenn gilt:

$$\left( A_{j+1,j}^{(j)} \quad \cdots \quad A_{m,j}^{(j)} \right)^T \neq 0 \quad \forall j \in \{1, 2, \dots, \min(m, n)\}$$

Auch hier kann man eine Pivotisierung durchführen, um die Durchführbarkeit des Algorithmus in mehr Fällen zu sichern.

Für die praktische Implementierung „merkt“ man sich die benötigten Householder-Matrizen mit den erzeugenden Vektoren

$$w_j = (0, \dots, 0, \underbrace{w_{jj}, \dots, w_{jn}}_{u_j})$$

am besten, indem man die nicht verschwindenden  $u_j$  auf den frei werdenden Plätzen von  $A^{(j)}$  speichert, und die Diagonale von  $A^{(j)}$  noch extra abspeichert. Auch eventuelle Permutationen werden extra notiert.

### 4.3 Kondition von linearen Ausgleichsproblemen

Sei  $A \in \mathbb{R}^{m \times n}$  eine Matrix mit vollem Spaltenrang und  $b, \Delta b \in \mathbb{R}^m \setminus \{0\}$ , sowie  $x, \Delta x \in \mathbb{R}^n \setminus \{0\}$  mit  $A^+b = x$  und  $A^+\Delta b = \Delta x$ . Dann gilt

$$\frac{|\Delta x|}{|x|} \leq \kappa^+(A) \cdot \frac{|\Delta b|}{\min_{c \in \mathbb{R}^m, A^+c=x} |c|}$$

für  $\kappa^+(A) := \|A\|_2 \cdot \|A^+\|_2 = \frac{\sigma_1}{\sigma_n}$ .

Für  $\|A^+\|_2 \cdot \|\Delta A\|_2 < 1$  ist auch eine Abschätzung des Koeffizientenfehlers möglich.

Beim linearen Ausgleichsproblem kann man von **Backward-Stability** sprechen: Die rundungsfehlerbehaftete Lösung  $\tilde{x}$  zum linearen Ausgleichsproblem zu  $(A, b)$  (wobei  $A$  vollen Spaltenrang besitzt) ist exakte Lösung des linearen Ausgleichsproblems zu  $(A + \Delta A, b)$  mit  $\|\Delta A\| \leq \|A\| \cdot O(\text{eps})$ .

## 5 Iterative Methoden

### 5.1 Eigenwertalgorithmen

Eigenwerte können (bei Dimension größer als vier) nur iterativ berechnet werden. Dabei wird typischerweise wie folgt vorgegangen:

- (1) Ähnlichkeitstransformation der Matrix  $A$  auf eine Matrix einfacher Struktur (z. B. Hessenberg-Matrix, Tridiagonalmatrix)
- (2) Iterationsverfahren zur Transformation auf obere Dreiecksmatrix (im Limes) bzw. Transformation von Vektoren, die gegen Eigenvektoren konvergieren.

#### 5.1.A Transformation auf Hessenberg-Form

Eine Matrix  $A \in \mathbb{R}^{m \times n}$  mit  $A_{jk} = 0$  für alle  $j > k + 1$  heißt **Hessenberg-Matrix**. Eine beliebige Matrix  $A$  kann mit Hilfe der **Householder-Transformation**, bei der  $A$  mit passenden Householder-Matrizen von links multipliziert wird, auf Hessenberg-Form gebracht werden.

### 5.1.B Störungstheorie für Eigenwertprobleme

Im Folgenden bezeichne  $\sigma(A)$  die Menge der Eigenwerte von  $A$ .

Eine Matrix  $A \in \mathbb{R}^{n \times n}$  heißt **diagonalisierbar**, wenn eine reguläre Matrix  $T \in \mathbb{C}^{n \times n}$  existiert, so dass gilt:  $T^{-1}AT = \text{diag}(\lambda_1, \dots, \lambda_n) \in \mathbb{C}^{n \times n}$ .

Nach dem **Satz von Bauer und Fike** gilt für eine diagonalisierbare Matrix  $A \in \mathbb{R}^{n \times n}$  mit  $T$  wie oben und beliebiges  $\Delta A \in \mathbb{R}^{n \times n}$

$$\max_{\mu \in \sigma(A + \Delta A)} \min_{\lambda \in \sigma(A)} |\mu - \lambda| \leq \text{cond}_p(T) \cdot \|\Delta A\|_p$$

wobei  $p \in [1, \infty]$  beliebig und  $\text{cond}_p(T) := \|T\|_p \cdot \|T^{-1}\|_p$ , wobei  $\|\cdot\|_p$  die Matrixnorm zur Vektornorm in  $l^p$  bezeichne.

Für allgemeine reelle Matrizen  $A$  gilt:

$$\max_{\mu \in \sigma(A + \Delta A)} \min_{\lambda \in \sigma(A)} |\mu - \lambda| \leq C \cdot \min \left( \|\Delta A\|_2, \|\Delta A\|_2^{\frac{1}{n}} \right)$$

wobei die Konstante  $C$  von der Schur-Zerlegung von  $A$  abhängt.

Die Eigenwerte einer Matrix hängen stetig von ihren Koeffizienten ab: Hat nämlich  $A \in \mathbb{R}^{n \times n}$  die Eigenwerte  $\lambda_1, \dots, \lambda_n$ , so existiert zu jedem  $\varepsilon > 0$  ein  $\delta > 0$ , sodass für jede Matrix  $\Delta A \in \mathbb{R}^{n \times n}$  mit  $\|\Delta A\| < \delta$  eine Nummerierung der Eigenwerte  $\mu_1, \dots, \mu_n$  von  $A + \Delta A$  existiert mit

$$\max_{j=1}^n |\lambda_j - \mu_j| < \varepsilon$$

Der **Satz von Gerschgorin** liefert eine Aussage über die Lage der Eigenwerte: Sei  $A \in \mathbb{R}^{n \times n}$  gegeben und  $G_j$  für  $j = 1, \dots, n$  definiert als

$$G_j := \left\{ z \in \mathbb{C} \mid |z - A_{jj}| \leq \sum_{\substack{k=1 \\ k \neq j}}^n |A_{jk}| \right\}$$

Dann gilt:  $\sigma(A) \subseteq G_1 \cup G_2 \cup \dots \cup G_n$ .

### 5.1.C Variationsformulierung für symmetrische Matrizen

Ist  $A \in \mathbb{R}^{n \times n}$  symmetrisch und  $x \in \mathbb{R}^n$ , so bezeichnet man den Ausdruck

$$R_A(x) := \frac{x^T A x}{x^T x}$$

als **Rayleigh-Quotienten**.

Diesen kann man zur Berechnung der Eigenwerte einer symmetrischen Matrix  $A \in \mathbb{R}^{n \times n}$  verwenden, wie der **Satz von Courant-Fischer** besagt: Sind  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$  die Eigenwerte von  $A$ , so gilt für  $j = 1, 2, \dots, n-1$ :

$$\begin{aligned}\lambda_{j+1} &= \min_{y_1, \dots, y_j \in \mathbb{R}^n} \max_{x \in \text{span}\{y_1, \dots, y_j\}^\perp \setminus \{0\}} R_A(x) \\ \lambda_{n-j} &= \max_{y_1, \dots, y_j \in \mathbb{R}^n} \min_{x \in \text{span}\{y_1, \dots, y_j\}^\perp \setminus \{0\}} R_A(x)\end{aligned}$$

Speziell gilt nach dem **Satz von Rayleigh-Ritz**:

$$\lambda_1 = \max_{x \neq 0} R_A(x) \quad \text{bzw.} \quad \lambda_n = \min_{x \neq 0} R_A(x)$$

Schließlich gilt nach dem **Störungssatz für symmetrische Matrizen** für symmetrisches  $A, \Delta A \in \mathbb{R}^{n \times n}$  und  $j = 1, \dots, n$ :

$$\lambda_j(A) + \lambda_n(\Delta A) \leq \lambda_j(A + \Delta A) \leq \lambda_j(A) + \lambda_1(\Delta A)$$

bzw.

$$|\lambda_j(A + \Delta A) - \lambda_j(A)| \leq \|\Delta A\|_2$$

### 5.1.D Vektoriterationen

Für  $B \in \mathbb{R}^{n \times n}$  und  $z^{(0)} \in \mathbb{R}^n \setminus \{0\}$  heißt die durch

$$z^{(j+1)} := Bz^{(j)} = B^{j+1}z^{(0)} \quad \text{für } j = 0, 1, 2, \dots$$

definierte Folge die **Folge der Iterierten** (mit Iterationsmatrix  $B$  und Startvektor  $z^{(0)}$ ).

Ist nun  $B$  eine diagonalisierbare Matrix mit den Eigenwerten  $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ , wobei  $\lambda_1 = \dots = \lambda_r$  und  $|\lambda_r| > |\lambda_{r+1}| \geq \dots \geq |\lambda_n|$  für ein  $r \in \{0, 1, \dots, n-1\}$ , und  $z^{(0)} \in \mathbb{R}^n \setminus \{0\}$  so gewählt, dass  $z^{(0)} \notin \ker(B - \lambda_{r+1}1) \oplus \dots \oplus \ker(B - \lambda_n1)$ , so gilt:

$$\frac{\|z^{(j+1)}\|}{\|z^{(j)}\|} = |\lambda_1| + O(q^j) \quad \text{für } j \rightarrow \infty$$

wobei  $q := \frac{|\lambda_{r+1}|}{|\lambda_1|} < 1$  und  $\|\cdot\|$  eine beliebige Vektornorm darstellt.

Möchte man den Betrag des dominanten Eigenwertes von  $A$  berechnen, so bietet sich zum Beispiel folgende Wahl von  $B$  an:

- (1) **Von-Mises-Iteration:**  $B = A$
- (2) **inverse Iteration nach Wielandt:**  $B = (A - \mu \cdot 1)^{-1}$  für ein  $\mu \in \mathbb{R} \setminus \sigma(A)$

Ist unter obigen Voraussetzungen  $B$  zusätzlich symmetrisch, so gilt

$$r_j := \frac{z^{(j+1)} \bullet z^{(j)}}{z^{(j)} \bullet z^{(j)}} = R_B(z^{(j)}) = \lambda_1 + O(q^{2j})$$

### 5.1.E QR-Algorithmus

Unter gewissen Bedingungen liefert das **QR-Verfahren** (siehe Anhang D.1) eine Approximation für die Eigenwerte einer Matrix: Sei  $A \in GL(n)$  diagonalisierbar mit betragsmäßig einfachen, reellen Eigenwerten  $\lambda_1, \dots, \lambda_n$  mit  $|\lambda_1| > \dots > |\lambda_n|$ .  $T$  bezeichne die Matrix der Eigenvektoren zu  $\lambda_1, \dots, \lambda_n$ . Schließlich besitze  $T^{-1}$  eine LR-Zerlegung ohne Pivotisierung. Dann gilt:

$$A^{(j)} = S_j U S_j + O(q^j)$$

für  $q := \max_{j=1}^{n-1} \left| \frac{\lambda_{j+1}}{\lambda_j} \right| < 1$  und  $S_j := \text{diag}(\sigma_1^{(j)}, \dots, \sigma_n^{(j)})$  mit  $\sigma_k^{(j)} = \pm 1$  und rechter oberer Dreiecksmatrix  $U$  mit  $U_{jj} = \lambda_j$  für  $j = 1, \dots, n$ .

Durch Durchführung eines (geeigneten) **Shifts** (siehe Anhang D.2) in jedem Schritt kann die Konvergenz deutlich verbessert werden (in diesem Fall von linear auf kubisch).

Oft lässt sich ein Problem auch durch **Deflation** vereinfachen. Diese tritt dann auf, wenn eine Iterationsmatrix  $A^{(j)}$  die Gestalt  $A^{(j)} = \text{diag}(A_u^{(j)}, A_l^{(j)})$  hat. Dann können nämlich die beiden Matrizen  $A_u^{(j)}$  bzw.  $A_l^{(j)}$  (jeweils kleinerer Dimension) getrennt betrachtet werden.

Wird das QR-Verfahren mit nur  $m \ll n$  vielen Spaltenvektoren durchgeführt, so erhält man die **Subspace-Iteration**.

### 5.1.F Direkte Berechnung

Die direkte Berechnung der Eigenwerte (über das charakteristische Polynom) ist nur für Tridiagonalmatrizen zu empfehlen. In diesem Fall erhält man nämlich das charakteristische Polynom mit Hilfe von einfachen Rekursionsformeln, und dessen Nullstellen können anschließend mit Hilfe von Polynomnullstellenberechnungsverfahren (wie z. B. Newton-Verfahren oder Bisektionsverfahren) zumindest approximativ berechnet werden.

## 5.2 Algorithmische Berechnung der Singulärwertzerlegung

Sei  $A \in \mathbb{C}^{m \times n}$  gegeben. Durch Householder-Transformationen von links und rechts kann man  $A$  auf Bidiagonalform  $B = VAU$  (mit  $U, V \in O(n)$ ) bringen. Anschließend berechnet man sich die Eigenwerte von  $\overline{B}^T B$  mit einer Modifikation des QR-Verfahrens, um die Singulärwertzerlegung von  $B$  (und damit jene von  $A$ ) zu erhalten.

## 5.3 Iterative Lösung von $Ax = b$

### 5.3.A Banach'scher Fixpunktsatz

Mit Hilfe des Banach'schen Fixpunktsatzes zeigt man: Ist  $M \in \mathbb{R}^{n \times n}$  mit  $\|M\| < 1$  für eine natürliche Matrixnorm  $\|\cdot\|$  (d.h. eine Norm, die von einer Vektornorm induziert wird), so ist für jedes  $c \in \mathbb{R}^n$  und  $x^{(0)} \in \mathbb{R}^n$  das Iterationsverfahren

$$x^{(j+1)} = Mx^{(j)} + c \quad \text{für } j = 0, 1, 2, \dots$$

konvergent gegen den Fixpunkt von  $x = Mx + c$ .  $M$  heißt dann **Iterationsmatrix**.

Es gilt:  $\|M\| < 1$  für *eine* natürliche Matrixnorm  $\iff$  Spektralradius( $M$ )  $< 1$

### 5.3.B Klassische Iterationsverfahren

- **Jacobi-Iteration:**

Sei  $D := \text{diag}(A_{11}, \dots, A_{nn})$ .

$$Ax = b \iff Dx = Dx - Ax + b \iff x = \underbrace{D^{-1}(D - A)}_M x + \underbrace{D^{-1}b}_c$$

Man spricht von einem **Gesamtschrittverfahren**, da alle Komponenten von  $x^{(j)}$  (mehr oder weniger) gleichzeitig berechnet werden können.

- **Gauß-Seidel-Iteration:**

$$\text{Sei } S := \begin{pmatrix} A_{11} & 0 & \cdots & 0 \\ A_{21} & A_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix} \text{ der untere Anteil von } A.$$

$$Ax = b \iff Sx = Sx - Ax + b \iff x = \underbrace{S^{-1}(S - A)}_M x + \underbrace{S^{-1}b}_c$$

Man spricht von einem **Einzelschrittverfahren**, da die Komponenten von  $x^{(j)}$  nur der Reihe nach (durch Vorwärtssubstitution) berechnet werden können.

Diese beiden Iterationsverfahren sind zumindest dann konvergent, wenn  $A$  strikt diagonaldominant ist.

Bei der **Relaxation** wählt man für ein Iterationsverfahren  $x^{(j+1)} = Mx^{(j)} + c$  einen Relaxationsparameter  $\omega$  und betrachtet

$$x^{(j+1)} = \omega (Mx^{(j)} + c) + (1 - \omega)x^{(j)}$$

Durch geschickte Wahl von  $\omega$  können die Konvergenzeigenschaften u. U. verbessert werden.

Ist  $\omega < 1$ , so spricht man von **Unterrelaxation**; ist  $\omega > 1$ , so spricht man von **Überrelaxation**.

Bei der **Nachiteration** besitzt man näherungsweise eine LR-Zerlegung von  $A$  (d. h.  $A \approx LR$ ), und erhält

$$x = x - (LR)^{-1}(Ax - b) = \underbrace{(1 - (LR)^{-1}A)}_M x + \underbrace{(LR)^{-1}b}_c$$

### 5.3.C cg-Verfahren

Gegeben sei eine SPD-Matrix  $A \in \mathbb{R}^{n \times n}$ . Das **Energieskalarprodukt**  $\langle \cdot, \cdot \rangle_A$  bzw. die **Energienorm**  $\|\cdot\|_A$  sind dann wie folgt definiert:

$$\begin{aligned} \langle x, y \rangle_A &:= x^T A y \\ \|x\|_A &:= \sqrt{\langle x, x \rangle_A} = \sqrt{x^T A x} \end{aligned}$$

$x$  heißt dann **A-konjugiert** zu  $y$  (in Zeichen:  $x \perp_A y$ ), wenn gilt:  $\langle x, y \rangle_A = 0$

Sei  $A \in \mathbb{R}^{n \times n}$  eine SPD-Matrix,  $K_j \subseteq \mathbb{R}^n$  ein Unterraum,  $x, b \in \mathbb{R}^n$  mit  $Ax = b$ . Dann sind folgende Aussagen äquivalent und eindeutig lösbar:

- (a)  $x_j \in K_j$  und  $\|x - x_j\|_A = \min_{y \in K_j} \|x - y\|_A$
- (b)  $x_j \in K_j$  und  $b - Ax_j \perp K_j$

Die Abbildung  $x \mapsto x_j$  heißt dann **Galerkin-Projektion** und (b) nennt man **Galerkin-Orthogonalität**.

Für eine SPD-Matrix  $A \in \mathbb{R}^{n \times n}$  und einen Vektor  $b \in \mathbb{R}^n$  nennt man

$$\mathcal{K}_j(A, b) := \text{span} \{b, Ab, A^2b, \dots, A^{j-1}b\}$$

den **Krylow-Raum** der Ordnung  $j$ .

Unter denselben Voraussetzungen heißt die durch (a) bezüglich der Krylow-Räume  $K_j := \mathcal{K}_j(A, b)$  definierte Folge  $(x_j)$  die **Folge der cg-Iterierten** (zur Berechnung siehe Anhang E). Für diese Folge gilt:

$$\|x - x_j\|_A \leq 2 \cdot \gamma^j \cdot \|x\|_A \quad \text{bzw.} \quad \|x - x_j\|_2 \leq 2\sqrt{\kappa} \cdot \gamma^j \cdot \|x\|_2$$

wobei  $x = A^{-1}b$ ,  $\gamma := \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$  und  $\kappa := \text{cond}_2(A)$ .

Da die Kondition  $\kappa$  sehr groß werden kann, ist das cg-Verfahren immer mit **Vorkonditionierung** zu implementieren. Dabei wird das System  $Ax = b$  (zum Beispiel) in ein neues System der Form  $P^{-1/2}AP^{-1/2}(P^{1/2}x) = P^{-1/2}b$  umgewandelt. Dafür muss (wenn man die Umformungen genauer betrachtet) nur die Wirkung von  $P^{-1}$  bekannt sein.

In der Konvergenzanalyse tritt dann  $\kappa = \text{cond}_2(P^{-1}A)$  auf. Klarerweise wird  $P^{-1}$  so gewählt, dass  $\text{cond}_2(P^{-1}A) \ll \text{cond}_2(A)$ .

## Anhang

### A Orthogonalisierungsverfahren von Gram-Schmidt

#### A.1 „Klassische“ Version

---

**Algorithmus GramSchmidt(A)**

---

**Input:** Matrix  $A$  mit l. u. Spaltenvektoren  $a_1, a_2, \dots, a_n \in \mathbb{R}^m$

**Output:** Orthonormalsystem  $(q_1, q_2, \dots, q_n)$

```
1  $q_1 := \frac{a_1}{\|a_1\|_2}$ 
2 für  $j = 2, 3, \dots, n$  {
3    $\tilde{q}_j := a_j - \sum_{k=1}^{j-1} (a_j \cdot q_k) \cdot q_k$ 
4    $q_j := \frac{\tilde{q}_j}{\|\tilde{q}_j\|_2}$ 
5 }
```

---

#### A.2 Modifizierte Version

---

**Algorithmus ModifizierterGramSchmidt(A)**

---

**Input:** Matrix  $A$  mit l. u. Spaltenvektoren  $a_1, a_2, \dots, a_n \in \mathbb{R}^m$

**Output:** Orthonormalsystem  $(q_1, q_2, \dots, q_n)$

```
1 für  $j = 1, 2, \dots, n$  {
2    $q_j^{(0)} := a_j$ 
3   für  $k = 1, 2, \dots, j - 1$  {
4      $q_j^{(k)} := q_j^{(k-1)} - (q_k \cdot q_j^{(k-1)}) \cdot q_k$ 
5   }
6    $q_j := \frac{\tilde{q}_j^{(j)}}{\|\tilde{q}_j^{(j)}\|_2}$ 
7 }
```

---

## B Matrix-Faktorisierungen

### B.1 Householder-Algorithmus

---

**Algorithmus** Householder(A)

---

**Input:** spaltenreguläre Matrix  $A \in \mathbb{R}^{m \times n}$

**Output:** QR-Zerlegung von  $A$

```
1   $A^{(1)} := A$ 
2   $k := \min(m, n)$ 
3  für  $j = 1, 2, \dots, k$  {
4       $H_j := \text{HE4} \left( \left( 0, \dots, 0, A_{jj}^{(j)}, \dots, A_{mj}^{(j)} \right)^T \right)$ 
5       $A^{(j+1)} := H_j \cdot A^{(j)}$ 
6  }
7   $Q := H_k \cdot H_{k-1} \cdot \dots \cdot H_1$ 
8   $R := A^{(k+1)}$ 
```

---

### B.2 LR-Zerlegung

---

**Algorithmus** LR(A)

---

**Input:** Matrix  $A \in GL(n)$

**Output:** LR-Zerlegung von  $A$

```
1   $L := 1$ 
2   $R := 0$ 
3  für  $j = 1, 2, \dots, n$  {
4      für  $k = j, j + 1, \dots, n$  {
5           $R_{jk} := A_{jk} - \sum_{m=1}^{j-1} L_{jm} R_{mk}$ 
6      }
7      für  $k = j + 1, j + 2, \dots, n$  {
8           $L_{kj} = \frac{1}{R_{jj}} \cdot \left( A_{kj} - \sum_{m=1}^{j-1} L_{km} R_{mj} \right)$ 
9      }
10 }
```

---

### B.3 Cholesky-Zerlegung

---

**Algorithmus Cholesky(A)**

---

**Input:** SPD-Matrix  $A$

**Output:** obere Dreiecksmatrix  $U$  mit  $A = UU^T$

```
1  für  $k = 1, 2, \dots, n$  {
2       $U_{kk} := \sqrt{A_{kk} - \sum_{j=1}^{k-1} U_{kj}^2}$ 
3      für  $j = k + 1, k + 2, \dots, n$  {
4           $U_{jk} := \frac{1}{U_{kk}} \cdot \left( A_{jk} - \sum_{l=1}^{k-1} U_{jl}U_{lk} \right)$ 
5      }
6  }
```

---

## C Gauß-Algorithmus

### C.1 „Naive“ Gauß-Elimination

---

**Algorithmus Gauß(A, b)**

---

**Input:** Matrix  $A \in GL(n)$ , rechte Seite  $b \in \mathbb{R}^n$

**Output:** Lösung  $x$  des linearen Gleichungssystems  $Ax = b$

```
1   $A^{(1)} := A$ 
2   $b^{(1)} := b$ 
3  für  $k = 1, 2, \dots, n - 1$  {
4      für  $j = k + 1, k + 2, \dots, n$  {
5           $L_{jk} := \frac{A_{jk}^{(k)}}{A_{kk}^{(k)}}$ 
6           $b_j^{(k+1)} := b_j^{(k)} - L_{jk} \cdot b_k^{(k)}$ 
7      }
8      für  $l = k + 1, k + 2, \dots, n$  {
9           $A_{jl}^{(k+1)} := A_{jl}^{(k)} - L_{jk} \cdot A_{kl}^{(k)}$ 
10     }
11 }
12  $x_n := \frac{b_n^{(n)}}{A_{nn}^{(n)}}$ 
13 für  $k = n - 1, n - 2, \dots, 1$  {
14      $x_k := \frac{1}{A_{kk}^{(n)}} \cdot \left( b_k^{(n)} - \sum_{j=k+1}^n A_{kj}^{(n)} \cdot x_j \right)$ 
15 }
```

---

## C.2 Gauß-Elimination mit Spaltenpivotsuche

---

### Algorithmus Spaltenpivot(A, b)

---

**Input:** Matrix  $A \in GL(n)$ , rechte Seite  $b \in \mathbb{R}^n$

**Output:** Lösung  $x$  des linearen Gleichungssystems  $Ax = b$

```
1   $A^{(1)} := A$ 
2   $b^{(1)} := b$ 
3  für  $k = 1, 2, \dots, n - 1$  {
4      Wähle  $p_k$  so, dass  $|A_{p_k k}^{(k)}| = \max_{j=k, \dots, n} |A_{jk}^{(k)}|$ 
5      falls  $p_k \neq k$  {
6           $A^{(k)} \begin{pmatrix} k \dots n \\ p_k \end{pmatrix} \leftrightarrow A^{(k)} \begin{pmatrix} k \dots n \\ k \end{pmatrix}$ 
7           $b_{p_k}^{(k)} \leftrightarrow b_k^{(k)}$ 
8      }
9      für  $j = k + 1, k + 2, \dots, n$  {
10          $L_{jk} := \frac{A_{jk}^{(k)}}{A_{kk}^{(k)}}$ 
11          $b_j^{(k+1)} := b_j^{(k)} - L_{jk} \cdot b_k^{(k)}$ 
12         für  $l = k + 1, k + 2, \dots, n$  {
13              $A_{jl}^{(k+1)} := A_{jl}^{(k)} - L_{jk} \cdot A_{kl}^{(k)}$ 
14         }
15     }
16 }
17  $x_n := \frac{b_n^{(n)}}{A_{nn}^{(n)}}$ 
18 für  $k = n - 1, n - 2, \dots, 1$  {
19      $x_k := \frac{1}{A_{kk}^{(n)}} \cdot \left( b_k^{(n)} - \sum_{j=k+1}^n A_{kj}^{(n)} \cdot x_j \right)$ 
20 }
```

---

### C.3 Gauß-Elimination mit Totalpivotsuche

---

**Algorithmus Totalpivot(A, b)**

---

**Input:** Matrix  $A \in GL(n)$ , rechte Seite  $b \in \mathbb{R}^n$

**Output:** Lösung  $x$  des linearen Gleichungssystems  $Ax = b$

```

1   $A^{(1)} := A$ 
2   $b^{(1)} := b$ 
3   $\text{pivot} := (1, 2, \dots, n)^T$ 
4  für  $k = 1, 2, \dots, n - 1$  {
5      Wähle  $l$  und  $m$  so, dass  $|A_{lm}^{(k)}| = \max_{\lambda, \mu=k, \dots, n} |A_{\lambda\mu}^{(k)}|$ 
6      falls  $l \neq k$  {
7           $A^{(k)} \begin{pmatrix} k \dots n \\ l \end{pmatrix} \leftrightarrow A^{(k)} \begin{pmatrix} k \dots n \\ k \end{pmatrix}$ 
8           $b_l^{(k)} \leftrightarrow b_k^{(k)}$ 
9      }
10     falls  $m \neq k$  {
11          $A^{(k)} \begin{pmatrix} m \\ k \dots n \end{pmatrix} \leftrightarrow A^{(k)} \begin{pmatrix} k \\ k \dots n \end{pmatrix}$ 
12          $\text{pivot}_k \leftrightarrow \text{pivot}_m$ 
13     }
14     für  $j = k + 1, k + 2, \dots, n$  {
15          $L_{jk} := \frac{A_{jk}^{(k)}}{A_{kk}^{(k)}}$ 
16          $b_j^{(k+1)} := b_j^{(k)} - L_{jk} \cdot b_k^{(k)}$ 
17         für  $l = k + 1, k + 2, \dots, n$  {
18              $A_{jl}^{(k+1)} := A_{jl}^{(k)} - L_{jk} \cdot A_{kl}^{(k)}$ 
19         }
20     }
21 }
22  $y_n := \frac{b_n^{(n)}}{A_{nn}^{(n)}}$ 
23 für  $k = n - 1, n - 2, \dots, 1$  {
24      $y_k := \frac{1}{A_{kk}^{(n)}} \cdot \left( b_k^{(n)} - \sum_{j=k+1}^n A_{kj}^{(n)} \cdot y_j \right)$ 
25 }
26 für  $k = 1, 2, \dots, n$  {
27      $x_{\text{pivot}_k} := y_k$ 
28 }

```

---



## E cg-Verfahren

---

**Algorithmus** cgVerfahren( $A, b$ )

---

**Input:** SPD-Matrix  $A \in \mathbb{R}^{n \times n}$ , rechte Seite  $b \in \mathbb{R}^n$

**Output:** Folge der cg-Iterierten zur Matrix  $A$  und Vektor  $b$

```
1   $r_0 := -b$ 
2   $x_0 := 0$ 
3   $d_{-1} := 0$ 
4  für  $j = 0, 1, 2, \dots$  bis Abbruch  {
5      falls  $r_j = 0$   {
6           $J := j$ 
7          Ausgabe „ $x_j$  ist Lösung“
8      } sonst  {
9           $\beta_{j-1} := \begin{cases} 0 & \text{für } j = 0 \\ \frac{\|r_j\|^2}{\|r_{j-1}\|^2} & \text{für } j \geq 1 \end{cases}$ 
10          $d_j := -r_j + \beta_{j-1}d_{j-1}$ 
11          $\alpha_j := \frac{\|r_j\|^2}{\|d_j\|_A^2}$ 
12          $x_{j+1} := x_j + \alpha_j d_j$ 
13          $r_{j+1} := r_j + \alpha_j A d_j$ 
14     }
15 }
```

---

## Index

- Algorithmus, 3
  - gutartiger, 4
- Ausgleichsproblem, 6
- Auslöschung, 3
  
- Backward-Stability, 10
- Bauer-Fike, 11
  
- cg-Iterierte, 15
- cg-Verfahren, 15, 22
- Cholesky-Zerlegung, 7, 18
- Courant-Fischer, 12
  
- Datenfehler, 1
- Deflation, 13
  
- Einzelstufenverfahren, 14
- Energienorm, 15
- Energieskalarprodukt, 15
  
- Fehlerabschätzung
  - a posteriori, 2
  - a priori, 2
- Fill-In, 8
  
- Galerkin-Orthogonalität, 15
- Galerkin-Projektion, 15
- Gauß'sche Normalengleichung, 5
- Gauß-Algorithmus, 8
- Gauß-Algorithmus, 18–20
  - mit Spaltenpivotsuche, 19
  - mit Totalpivotsuche, 20
- Gauß-Seidel-Iteration, 14
- Gerschgorin, 11
- Gesamtschrittverfahren, 14
- Gleitpunktarithmetik, 3
- Gram-Schmidt, 7, 16
  - modifizierter, 7, 16
  
- Hessenberg-Matrix, 10
- Householder-Algorithmus, 9, 17
- Householder-Matrix, 9
- Householder-Transformation, 9, 10
  
- Intervallarithmetik, 3
- inverse Iteration nach Wielandt, 12
  
- Iterationsmatrix, 14
- Jacobi-Iteration, 14
  
- Konditionszahl, 1
- Krylow-Raum, 15
  
- lineares Ausgleichsproblem, 6
- LR-Zerlegung, 8, 17
  
- Maschinengenauigkeit, 2
- Maschinenzahlen
  - denormalisierte, 2
  - normalisierte, 2
- Matrix
  - diagonalisierbare, 11
  - strikt diagonaldominante, 8
- Minimum-Norm-Lösung, 6
- Modellierungsfehler, 1
  
- Nachiteration, 15
  
- Pivotisierung, 8
- Pseudoinverse, 6
  
- QR-Verfahren, 13, 21
- QR-Zerlegung, 7, 17
  
- Rayleigh-Quotienten, 11
- Rayleigh-Ritz, 12
- Relaxation, 14
- Rundungsfehler, 2
  
- Satz
  - von Bauer-Fike, 11
  - von Courant-Fischer, 12
  - von Gerschgorin, 11
  - von Rayleigh-Ritz, 12
- Shift, 13
- Singulärwert, 6
- Singulärwertzerlegung, 6, 13
- Skyline-Technik, 8
- Spaltenpivotsuche, 9, 19
- Sparse-Format, 8
- SPD-Matrix, 7
- Störungssatz für symmetrische Matrizen, 12

Subspace-Iteration, 13  
Totalpivotsuche, 9, 20  
Überlauf, 3  
Überrelaxation, 15  
Unterlauf, 3  
Unterrelaxation, 15  
unvermeidbarer Fehler, 3  
Vandermonde-Matrix, 5  
Verfahrensfehler, 1  
Von-Mises-Iteration, 12  
Vorkonditionierung, 15