

MITSCHRIEB

# Numerik

*Emma Bach*

Basierend auf:

Vorlesungen Numerik I + II von  
Prof. Dr. Patrick DONDL

2025-11-09

# Inhalt

<b>1</b>	<b>Aufgabenstellung</b>	<b>2</b>
<b>2</b>	<b>Numerische Lineare Algebra</b>	<b>4</b>
2.1	Matrixfaktorisierung .....	4
2.1.1	Dreiecksmatrizen .....	4
2.1.2	$LU$ -Zerlegung .....	4
2.1.3	Matrixnormen.....	9
2.1.4	Konditionszahl.....	11
<b>3</b>	<b>Eliminationsverfahren</b>	<b>13</b>
3.1	Gauss-Jordan-Elimination.....	13
3.2	Pivotsuche.....	14

# Chapter 1

## Aufgabenstellung

In der Numerik beschäftigt man sich mit der praktischen Berechnung von Lösungen mathematischer Probleme.

**Beispiel 1.0.1.** Berechne  $\int_0^1 e^{-x^2} dx$ !

**Beispiel 1.0.2.** Berechne  $\sin(20)$ !

**Beispiel 1.0.3.** Berechne  $\sqrt{753}$ !

**Beispiel 1.0.4.** Berechne  $\min_{x \in [0,1]} F(x)$ , für eine geeignete Funktion  $F$ !

**Beispiel 1.0.5.** Berechne  $x$ , sodass  $f(x) = 0$ !

**Beispiel 1.0.6.** Berechne  $x \in \mathbb{R}^n$ , sodass  $Ax = b$ !

**Definition 1.0.7.** Eine Mathematische Aufgabe in der Numerik besteht im Finden einer Lösung von

$$F(x, d) = 0$$

für gegebenes Datum  $d$  und gegebene Funktion  $F$ .

Typischerweise können in akzeptabler Zeit keine exakten Lösungen gefunden werden, sondern nur Approximationen. Insbesondere stehen statt den vollen Mengen  $\mathbb{Q}$ ,  $\mathbb{R}$ ,  $\mathbb{C}$  etc. auch nur endlich viele **Maschinenzahlen** zur Verfügung - arbiträre reelle Zahlen benötigen unendlich viel Speicher! Rechenoperationen sind dementsprechend fehlerbehaftet, es gibt Rundungsfehler. Außerdem gibt es in reellen Anwendungen oft **Modellfehler** und **Datenfehler**.

Eine Grundlegende Idee in der Numerik ist es deshalb, eine gute Balance zwischen Exaktheit und Aufwand der Berechnung zu finden.

**Beispiel 1.0.8.** Die Berechnung der Determinante einer Matrix mittels Laplaceschem Entwicklungssatz benötigt  $O(n!)$  Rechenoperationen. Die Determinante mit diesem Verfahren zu berechnen, dauert sehr viel länger, als das Universum alt ist.

Besser: Matrix (approximativ) auf Dreiecksgestalt bringen und die Diagonalelemente multiplizieren.

**Definition 1.0.9.** Eine Mathematische Aufgabe heißt **wohlgestellt**, wenn zu geeigneten Daten  $d$  eindeutige Lösungen  $x$  existieren, und diese stetig von  $d$  abhängt. Andernfalls ergibt die Suche nach einer numerischen Lösung wenig Sinn. Für wohlgestellte Probleme existiert eine Lösungsfunktion  $\varphi$ , sodass  $x = \varphi(d)$  das Problem löst, d.h.  $f(\varphi(d), d) = 0$ .

**Definition 1.0.10.** Ein numerischer Algorithmus zur näherungsweisen Lösung einer wohlgestellten Aufgabe  $\varphi$  ist eine Abbildung  $\tilde{\varphi}$ , die durch Hintereinanderausführung möglicherweise fehlerbehafteter elementarer Rechenoperationen definiert ist, also

$$\tilde{\varphi} = f_j \circ f_{j-1} \circ \dots \circ f_1$$

**Definition 1.0.11.** Der **Aufwand** eines Verfahrens  $\tilde{\varphi}$  ist die Anzahl der benötigten elementaren Rechenschritte. Typischerweise interessiert uns nicht die exakte Anzahl an Schritten, sondern nur die Größenordnung.

**Proposition 1.0.12.** *Das Gaußverfahren hat Aufwand  $\mathcal{O}(n^3)$ .*

## Chapter 2

# Numerische Lineare Algebra

### 2.1 Matrixfaktorisierung

#### 2.1.1 Dreiecksmatrizen

**Definition 2.1.1.** Eine Matrix  $L \in \mathbb{R}^{n \times n}$  heißt **untere Dreiecksmatrix**, falls  $\forall i < j : l_{ij} = 0$ .

**Definition 2.1.2.** Eine Matrix  $U \in \mathbb{R}^{n \times n}$  heißt **obere Dreiecksmatrix**, falls  $U^\top$  eine untere Dreiecksmatrix ist.

**Definition 2.1.3.** Eine Dreiecksmatrix heißt **normalisiert**, falls alle ihre Diagonaleinträge 1 sind.

**Definition 2.1.4.** Eine Matrix heißt **regulär**, wenn sie invertierbar ist.

**Lemma 2.1.5.** *Die quadratischen oberen (bzw. unteren) Dreiecksmatrizen bilden unter Matrixmultiplikation eine Gruppe.*

Lineare Gleichungssysteme mit regulärer Dreiecksmatrix lassen sich leicht lösen. Sei  $U \in \mathbb{R}^{n \times n}$  eine reguläre obere Dreiecksmatrix und  $b \in \mathbb{R}^n$ . Wir berechnen  $x \in \mathbb{R}^n$  folgendermaßen:

1. for  $i = n : -1 : 1$ :

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

2. end.

Der Aufwand dieses Verfahrens ist  $\mathcal{O}(n^2)$ . Ein analoger Algorithmus existiert für untere Dreiecksmatrizen.

#### 2.1.2 LU-Zerlegung

Falls für eine reguläre Matrix  $A \in \mathbb{R}^{n \times n}$  eine Zerlegung  $A = LU$  in eine untere Dreiecksmatrix  $U$  und eine obere Dreiecksmatrix  $L$  gegeben ist, so lässt sich das lineare Gleichungssystem  $Ax = b$  in zwei Schritten lösen:

1. Löse  $Ly = b$ .

2. Löse  $Ux = y$ .

**Definition 2.1.6.** Eine Faktorisierung  $A = LU$  mit unterer Dreiecksmatrix  $L \in \mathbb{R}^{n \times n}$  und oberer Dreiecksmatrix  $U \in \mathbb{R}^{n \times n}$  heißt **LU-Zerlegung** von  $A$ . Die Zerlegung heißt **normalisiert**, falls  $L$  normalisiert ist.

**Satz 2.1.7.** Für jede reguläre Matrix  $A \in \mathbb{R}^{n \times n}$  sind äquivalent:

1. Es existiert eine eindeutige normalisierte LU-Zerlegung.
2. Alle Untermatrizen  $A_k = (a_{ij})_{(i,j) \in (1, \dots, k)^2}$  sind regulär.

*Beweis.*

→ Ist  $A$  regulär, so sind auch  $L$  und  $U$  regulär. Damit sind von  $L$  und  $U$  alle Diagonaleinträge nicht null. Somit sind auch die Untermatrizen  $L_k$  und  $U_k$  regulär, somit auch die Untermatrizen  $A_k = L_k U_k$ .

← Für  $n = 1$  ist die Aussage klar. Sei nun angenommen, die Aussage gelte für Matrizen der Größe  $(n-1) \times (n-1)$ . Damit existieren Matrizen  $L_{n-1}, U_{n-1}$ , sodass  $A_{n-1} = L_{n-1} U_{n-1}$  eine normalisierte LU-Zerlegung ist. Seien nun  $\begin{pmatrix} b \\ a_{nn} \end{pmatrix}$  die letzte Spalte und  $(c^\top, a_{nn})$  die letzte Zeile von  $A$ . Die Aussage ist bewiesen, wenn geeignete  $l, u \in \mathbb{R}^{n-1}$  und  $r \in \mathbb{R}$  existieren, sodass

$$\begin{aligned} \begin{pmatrix} A_{n-1} & b \\ c^\top & a_{nn} \end{pmatrix} &= \begin{pmatrix} L_{n-1} & 0 \\ l^\top & 1 \end{pmatrix} \begin{pmatrix} U_{n-1} & u \\ 0 & r \end{pmatrix} \\ &= \begin{pmatrix} L_{n-1} U_{n-1} & L_{n-1} u \\ (U_{n-1}^\top l)^\top & l^\top u + r \end{pmatrix} \\ &= \begin{pmatrix} A_{n-1} & L_{n-1} u \\ (U_{n-1}^\top l)^\top & l^\top u + r \end{pmatrix} \end{aligned}$$

Wir brauchen also  $L_{n-1} u = b$ ,  $U_{n-1}^\top l = c$ , und  $a_{nn} = l^\top u + r$ . Durch Regularität von  $L_{n-1}$  und  $U_{n-1}$  existieren eindeutige Lösungen  $u, l$ , der ersten beiden Gleichungen, somit ist auch  $r$  festgelegt.

□

**Korollar 2.1.8.**

- Jede positiv definite Matrix besitzt eine eindeutige LU-Zerlegung.
- Jede strikt diagonaldominante Matrix, also jede Matrix  $A$  mit  $\sum_{j \in 1, \dots, n, i \neq j} |a_{ij}| < |a_{ii}|$  besitzt eine eindeutige LU-Zerlegung.
- Die Matrix  $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$  besitzt keine LU-Zerlegung.

- Die Nullmatrix besitzt zwar LU-Zerlegungen, diese sind aber nicht eindeutig.

**Lemma 2.1.9.** Falls  $A = LU$  eine normalisierte LU-Zerlegung von  $A$  ist, so gilt

$$a_{ik} = u_{ik} + \sum_{j=1}^{i-1} l_{ij} u_{jk}$$

und

$$a_{ki} = l_{ki} u_{ii} + \sum_{j=1}^{i-1} l_{kj} u_{ji}$$

*Beweis.* Es gilt  $l_{ij} = 0$  für  $j > i$  und  $l_{ii} = 1$ . Es gilt außerdem  $u_{ij} = 0$  für  $j < i$ . Die Formeln folgen direkt aus der Definition des Matrixprodukts.  $\square$

Diese Formeln lassen sich für  $i \leq k$  nach  $u_{ik}$  auflösen und für  $k > i$  nach  $l_{ki}$  auflösen. Wir erhalten folgenden Algorithmus:

---

```

1: for  $i = 1, i \leq n, i++$  do
2:   for  $k = i, k \leq n, k++$  do
3:      $u_{ik} \leftarrow a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk}$ 
4:   end for
5:   for  $k = i + 1, k \leq n, k++$  do
6:      $l_{ki} \leftarrow \frac{1}{u_{ii}} \cdot \left( a_{ki} - \sum_{j=1}^{i-1} l_{kj} u_{ji} \right)$ 
7:   end for
8: end for

```

---

**Proposition 2.1.10.** Der Rechenauftrag dieses Algorithmus beträgt  $O(n^3)$ .

**Proposition 2.1.11.** Es ist nicht mehr Speicher nötig, als sowieso für  $A$  benötigt wird. Die Einträge von  $A$  können im Speicher einfach sukzessiv durch die jeweiligen Einträge von  $L$  bzw.  $U$  ersetzt werden.

**Definition 2.1.12.** Ein numerisches Problem  $\varphi$  heißt **schlecht Konditioniert**, wenn kleine Unterschiede in der Eingabe zu großen Unterschieden in der korrekten Lösung führen, also wenn

$$\frac{|\varphi(\tilde{x}) - \varphi(x)|}{|\varphi(x)|} \gg \frac{|\tilde{x} - x|}{|x|}$$

Ansonsten heißt die Aufgabe **gut konditioniert**.

**Definition 2.1.13.** Ein Verfahren  $\tilde{\varphi}$  heißt **numerisch instabil**, wenn eine Störung  $\tilde{x}$  existiert, sodass der durch Rundungsfehler verursachte relative Fehler erheblich größer ist als der rein durch die Störung verursachte Fehler.

**Beispiel 2.1.14.** Sei

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 + \varepsilon \end{pmatrix}$$

Für  $\varepsilon \in \mathbb{R}^+$ . Es gilt

$$A^{-1} = \begin{pmatrix} 1 + \frac{1}{\varepsilon} & -\frac{1}{\varepsilon} \\ -\frac{1}{\varepsilon} & \frac{1}{\varepsilon} \end{pmatrix}$$

$$A^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad A^{-1} \begin{pmatrix} 1 \\ 1 + \varepsilon \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Eine kleine Störung in den Daten  $b$  des linearen Gleichungssystems  $Ax = b$  führt zu Problemen der Größenordnung  $1/!!!$ . So können wir keine Numerik machen!!! Dieses Problem ist **schlecht konditioniert**.

**Beispiel 2.1.15.** Sei

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

also

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

So haben wir kein Problem bei der Berechnung von  $A^{-1}b$  - die Aufgabe ist gut konditioniert. Sagen wir nun, wir versuchen, das Gleichungssystem effizient durch  $LU$ -Zerlegung zu lösen. Wir sehen,  $LU$ -Zerlegung von  $A$  ist jedoch gegeben durch

$$A = \begin{pmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{pmatrix} \begin{pmatrix} \varepsilon & 1 \\ 0 & \frac{1}{\varepsilon} \end{pmatrix},$$

und die Berechnung von  $L^{-1}b$  und  $U^{-1}b$  führt nun wieder zu großen Rundungsfehlern. Aus unserer Idee entsteht also ein **instabiler Algorithmus**.

**Satz 2.1.16. Cholesky-Zerlegung:** Sei  $A \in \mathbb{R}^{n \times n}$  symmetrisch und positiv definit. So existiert eine eindeutige untere Dreiecksmatrix  $L$ , sodass

$$A = LL^\top.$$

und  $l_{ii} > 0$

*Beweis.* Für  $n = 1$  ist die Suche durch  $l_{11} = \sqrt{a_{11}}$  erledigt.

Die Untermatrix  $A_{n-1}$  ist immer ebenfalls positiv definit und symmetrisch. Sei also  $A_{n-1} = L_{n-1}L_{n-1}^\top$ . Wir setzen  $\begin{pmatrix} b \\ a_{nn} \end{pmatrix}^\top$  als die letzte Zeile von  $A$ . Dann



müssen wir zum Beweis des Satzes einen Vektor  $c \in \mathbb{R}^{n-1}$  und ein  $\alpha \geq 0$  finden, sodass

$$\begin{pmatrix} A_{n-1} & b \\ b^\top & a_{nn} \end{pmatrix} = \begin{pmatrix} L_{n-1} & 0 \\ c^\top & \alpha \end{pmatrix} \begin{pmatrix} L_{n-1}^\top & c \\ 0 & \alpha \end{pmatrix} = \begin{pmatrix} A_{n-1} & L_{n-1}c \\ (L_{n-1}c)^\top & \alpha^2 + c^\top c \end{pmatrix}$$

Dies ist nach Annahme äquivalent zu  $L_{n-1}c = b$  und  $c^\top c + \alpha^2 = a_{nn}$

Da  $L$  regulär ist existiert ein eindeutiges  $c$ , welches die erste Gleichung erfüllt. Es gilt:

$$\det A = \det \begin{pmatrix} L_{n-1} & 0 \\ c^\top & \alpha \end{pmatrix} \cdot \det \begin{pmatrix} L_{n-1}^\top & c \\ 0 & \alpha \end{pmatrix} = \alpha^2 (\det L_{n-1})^2$$

Da  $\det A > 0$  und  $\det L_{n-1} \geq 0$  bekommen wir  $\alpha > 0$ , sodass  $c^\top c + \alpha^2 = a_{nn}$  ebenfalls eine eindeutige positive Lösung hat.  $\square$

**Lemma 2.1.17.** Für  $A = LL^\top$  gilt:

$$a_{ik} = \begin{cases} l_{ik}l_{kk} + \sum_{j=i}^{k-1} l_{ij}l_{kj} & i > k \\ l_{kk}^2 + \sum_{j=1}^{k-1} l_{kj}^2 & i = k \end{cases}$$

*Beweis.* Matrixmultiplikation ohne triviale Summanden.  $\square$

---

### Cholesky-Zerlegung:

```

1: for  $k = 1, i \leq n, i++$  do
2:    $l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$ 
3:   for  $i = k+1, i \leq n, i++$  do
4:      $l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij}l_{kj}) \frac{1}{l_{kk}}$ 
5:   end for
6: end for
```

---

**Proposition 2.1.18.** Der Aufwand ist wieder  $\mathcal{O}(n^3)$ , allerdings mit kleineren Konstanten.

**Proposition 2.1.19.** Lösung von  $Ax = b$  für  $A = LL^\top$  wie gehabt durch  $Ly = b$  und  $L^\top x = y$ .

### 2.1.3 Matrixnormen

Bekannt sind die üblichen Vektornormen auf  $\mathbb{R}^n$ , insbesondere

$$\|\vec{v}\|_p = \left( \sum_{j=1}^n |v_j|^p \right)^{\frac{1}{p}}$$

und

$$\|\vec{v}\|_\infty = \max v_j$$

Für  $1 \leq p, q \leq \infty$  existiert eine Konstante  $c_{pqn}$ , sodass

$$\forall \vec{v} \in \mathbb{R}^n : \frac{1}{c_{pqn}} \|\vec{v}\|_p \leq \|\vec{v}\|_q \leq c_{pqn} \|\vec{v}\|_p$$

**Definition 2.1.20.** Für Normen  $\|\cdot\|_{\mathbb{R}^n}$  und  $\|\cdot\|_{\mathbb{R}^m}$  auf  $\mathbb{R}^n$  und  $\mathbb{R}^m$  definieren wir die Operatornorm auf  $\text{hom}(\mathbb{R}^n, \mathbb{R}^m) = \mathbb{R}^{m \times n}$  als

$$\|A\|_{op} = \sup_{x \in \mathbb{R}^n : \|x\|_{\mathbb{R}^n} = 1} \|Ax\|_{\mathbb{R}^m}$$

**Lemma 2.1.21.** Die Operatornorm ist eine Norm.

- Beweis.*
1. Skalare können aus der inneren Norm und dem Supremum wie nötig herausgezogen werden.
  2. Das Supremum ist über einer Menge positiver Zahlen, falls  $x \neq \vec{0}$  gibt es mindestens einen Vektor größer 0.
  3. Dreiecksungleichung folgt aus der Dreiecksungleichung für  $\|\cdot\|_{\mathbb{R}^m}$ .

□

**Lemma 2.1.22.**

$$\|A\|_{op} = \inf \{c > 0 : \forall x \in \mathbb{R}^n \|Ax\| \leq c\|x\|\}$$

**Lemma 2.1.23.** Für  $A \neq 0$  und  $x \in \mathbb{R}^n$  mit  $\|x\| \leq 1$  und  $\|Ax\| = \|A\|_{op}$  folgt  $\|x\| = 1$

**Korollar 2.1.24.** Für alle  $x \in \mathbb{R}^n$  gilt  $\|Ax\| \leq \|A\|_{op}\|x\|$

**Lemma 2.1.25.** Es gibt Vektoren, sodass die Matrixnorm ihr inf und ihren sup annimmt.

*Beweis.* Es handelt sich um eine stetige Funktion auf einem Kompaktum. □

**Beispiel 2.1.26.** 1. Die **Spaltensummennorm**  $\|\cdot\|_1$  ist eine Operatornorm:

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

2. Die **Zeilensummennorm**  $\|-\|_\infty$  ist eine Operatornorm:

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

3. Die **Spektralnorm**  $\|-\|_2$  ist eine Operatornorm:

$$\|A\|_2 = \rho(A^\top A) = (\max\{|\lambda| : \lambda \text{ ist Eigenwert von } A^\top A\})^{\frac{1}{2}}$$

**Lemma 2.1.27.** Für  $A \in \mathbb{R}^{l \times m}$ ,  $B \in \mathbb{R}^{m \times n}$  und eine beliebige Operatornorm  $\|-\|$  gilt  $\|AB\| \leq \|A\|\|B\|$

*Beweis.*

$$\begin{aligned} \|ABx\| &\leq \|A\|\|Bx\| \leq \|A\|\|B\|\|x\| \\ \Rightarrow \|AB\| &\leq \|A\|\|B\| \end{aligned}$$

□

**Lemma 2.1.28.** Falls die Normen in Bild und Urbild gleich sind, gilt

$$\|E_n\| = 1$$

**Lemma 2.1.29.** Falls die Normen in Bild und Urbild gleich sind, gilt für  $A$  symmetrisch mit Eigenwert  $\lambda$

$$\|A\| \geq |\lambda|$$

**Beispiel 2.1.30.** Die Frobeniusnorm  $\|-\|_{\mathcal{F}}$  einer Matrix  $A \in \mathbb{R}^{m \times n}$  ist gegeben durch

$$\|A\|_{\mathcal{F}} = \left( \sum_{j=1}^m \sum_{i=1}^n a_{ij}^2 \right)^{\frac{1}{2}}$$

**Lemma 2.1.31.** Für  $n > 1$  ist die Frobeniusnorm keine Operatornorm!

*Beweis.*

$$\|E_n\| = \sqrt{n}$$

Normieren wir die Norm, gilt

$$\frac{1}{\sqrt{2}} \left\| \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right\|_{\mathcal{F}} = \frac{1}{\sqrt{2}} < 1$$

Wobei 1 ein Eigenwert ist, was unseren vorherigen Lemmata widerspricht. □

### 2.1.4 Konditionszahl

**Satz 2.1.32.** Sei  $\|-\|$  eine Operatornorm auf  $\mathbb{R}^{n \times n}$ . Sei  $A \in \mathbb{R}^{n \times n}$  regulär und seien  $x, x', b, b' \in \mathbb{R}^n$ , sodass  $Ax = b$ ,  $Ax' = b'$ . Dann gilt:

$$\frac{\|x - x'\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|b - b'\|}{\|b\|}$$

**Satz 2.1.33.**

$$\|x - x'\| = \|A^{-1}(b - b')\| \leq \|A^{-1}\| \|b - b'\|$$

und

$$\|b\| = \|Ax\| \leq \|A\| \|x\|$$

Es folgt:

$$\frac{\|x - x'\|}{\|x\|} \leq \frac{\|A^{-1}\| \|b - b'\|}{\|x\|} \leq \frac{\|A^{-1}\| \|b - b'\|}{\|A^{-1}\| \|b\|}$$

**Definition 2.1.34.** Die **Konditionszahl** einer regulären Matrix  $A \in \mathbb{R}^{n \times n}$  bezüglich der durch  $\|-\|$  auf  $\mathbb{R}^n$  induzierten Operatornorm ist gegeben durch:

$$\text{cond}_{\|-\|}(A) = \|A\| \|A^{-1}\|$$

Wir schreiben oft  $\text{cond}_p$  statt  $\text{cond}_{\|-\|_p}$ .

**Lemma 2.1.35.**  $\text{cond}(A) \geq 1$

**Lemma 2.1.36.** Für  $A$  symmetrisch mit Eigenwerten  $\lambda_i$  gilt

$$\text{cond}_2(A) = \frac{\max |\lambda_j|}{\min |\lambda_j|}$$

**Beispiel 2.1.37.**

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 + \varepsilon \end{pmatrix}$$

Besitzt die Eigenwerte

$$\lambda_{1,2} = 1 + \frac{\varepsilon}{2} \pm \left(1 + \frac{\varepsilon^2}{4}\right)^{\frac{1}{2}}$$

Also  $\lambda_1 \approx 2 + \frac{\varepsilon}{2}$  und  $\lambda_2 \approx \frac{\varepsilon}{2}$ . Für  $\varepsilon \rightarrow 0$  geht also die Konditionszahl gegen unendlich.

**Satz 2.1.38.** Für  $A$  symmetrisch und positiv definit mit Cholesky-Zerlegung  $A = LL^\top$  gilt

$$\text{cond}_2(L) = \text{cond}_2(L^\top) = \sqrt{\text{cond}(A)}$$

Also kann das Problem, welches bei der  $LU$ -Zerlegung auftrat, bei der Cholesky-Zerlegung nicht vorkommen.

*Beweis.* Wir bemerken zunächst, dass  $L^\top L$  und  $LL^\top$  die selben Eigenwerte haben. Beide Matrizen sind symmetrisch und es gilt

$$\begin{aligned} \forall x \in \mathbb{R}^n, \lambda \in \mathbb{R} : \\ L^\top Lx = \lambda x \Leftrightarrow LL^\top Lx = \lambda(Lx) := \lambda y \end{aligned}$$

Somit folgt

$$\rho(LL^\top) = \rho(L^\top L)$$

und somit auch

$$\|L\|_2 = \|L^\top\|_2$$

analog gilt

$$\|L^{-1}\|_2 = \|L^{-\top}\|_2$$

Also  $\text{cond}_2(L) = \text{cond}_2(L^\top)$ . Da  $LL^\top = A$ , und  $A$  symmetrisch, folgt

$$\begin{aligned} \|L\|_2^2 &= \|L^\top\|_2^2 \\ &= \rho(LL^\top) \\ &= \rho(A) \\ &= \|A\|_2 \end{aligned}$$

und

$$\begin{aligned} \|L^{-1}\|_2^2 &= \rho(L^{-\top}L^{-1}) \\ &= \rho(A^{-1}) \\ &= \|A^{-1}\|_2, \end{aligned}$$

also insgesamt

$$\begin{aligned} \text{cond}_2(L) &= \|L\|_2 \|L^{-1}\|_2 \\ &= \|A\|_2^{1/2} \|A^{-1}\|_2^{1/2} \\ &= \sqrt{\text{cond}_2(A)} \end{aligned}$$

□

## Chapter 3

# Eliminationsverfahren

### 3.1 Gauss-Jordan-Elimination

**Definition 3.1.1. Gauss-Jordan-Elimination** Sei  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$ .

1. Setze  $A^{(1)} = A$ ,  $b^{(1)} = b$ ,  $k = 1$ .
2. Für  $A^{(k)}$  gelte für  $1 \leq j \leq k - 1$  und  $i \geq j + 1$   $a_{ij}^k = 0$ , d.h.  $A^{(k)}$  habe die Form

$$\begin{pmatrix} a_{11} & \dots & & \dots & a_{1n} \\ & a_{22} & & & \vdots \\ & & \ddots & & \\ & & & a_{kk} & \dots & a_{kn} \\ & & & \vdots & & \vdots \\ & & & a_{nk} & \dots & a_{nn} \end{pmatrix}$$

mit Nullen im unteren linken Teil.

3. Wir setzen  $l_{ik} = \frac{a_{ik}}{a_{kk}^{(k)}}$  und definieren  $L \in \mathbb{R}^{n \times n}$  als

$$L = \begin{pmatrix} 1 & \dots & & \dots & 0 \\ & 1 & & & \vdots \\ & & \ddots & & \\ & & & 1 & \dots & 0 \\ & & & -l_{k+1,k} & \dots & 0 \\ \vdots & & & \vdots & & \vdots \\ 0 & \dots & -l_{n,k} & \dots & 0 & \dots & 1 \end{pmatrix}$$

4. Setze  $A^{(k+1)} = L^{(k)} A^{(k)}$ ,  $b^{(k+1)} = L^{(k)} b^{(k)}$
5. Stoppe, falls  $k + 1 = n$ , sonst erhöhe  $k$  und gehe zu Schritt 2.

**Satz 3.1.2.** *Ist  $A \in \mathbb{R}^{n \times n}$  regulär, so ist Gauß-Jordan-Elimination genau dann durchführbar, wenn  $A$  eine LU-Zerlegung hat. Das Verfahren liefert dann die normierte LU-Zerlegung  $U = A^{(n)}$  und  $L = (L^{(n-1)} \cdot \dots \cdot L^{(1)})^{-1}$ . Die rechte Seite  $y = b^{(n)}$  löst dann  $y = L^{-1}b$  und die Lösung des Linearen Gleichungssystems  $Ax = b$  ist gegeben durch die Lösung von  $Ux = y$ .*

## 3.2 Pivotsuche

Das Gaußverfahren ist für

$$A = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix}$$

zwar durchführbar, aber instabil. Wir führen deswegen eine sogenannte Pivotsuche durch - im  $k$ -ten Schritt bestimmen wir  $p \in \{k, \dots, n\}$ , sodass

$$\left| a_{pk}^{(k)} \right| = \max a_{ik}^{(k)}$$

und vertauschen dann die Zeilen  $p$  und  $k$  in  $A^{(k)}$  und  $b^{(k)}$ . Wir müssen diese Vertauschung jedoch nicht im Speicher tatsächlich durchführen, es reicht, einen Permutationsvektor  $\pi \in \mathbb{N}^n$  vorzuschalten. Wir initialisieren  $\pi$  als  $(1, 2, \dots, n)^\top$ , sollen daraufhin  $k$  und  $p$  vertauscht werden, vertauschen wir die jeweiligen Komponenten in  $\pi$ . Wollen wir daraufhin im Programm auf  $a_{ij}$  zugreifen, müssen wir stattdessen auf  $a_{\pi(i)j}$  zugreifen.

**Satz 3.2.1.** *Ist  $A \in \mathbb{R}^{n \times n}$  regulär,  $b \in \mathbb{R}^n$ , so ist das Gaußverfahren mit Pivotsuche durchführbar und liefert die normalisierte LU-Zerlegung*

$$PA = LU$$

mit  $|l_{ij}| \leq 1$  für alle  $i, j$ , sowie die modifizierte rechte Seite  $b^{(n)} = L^{-1}Pb$ , wobei

$$P = P^{(n-1)} P^{(n-2)} \dots P^{(1)}$$