

MITSCHRIEB

# Numerik

*Emma Bach*

Basierend auf:

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

October 22, 2025

# 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

# Chapter 1

## Aufgabenstellung

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

**Beispiel 1.1.** Berechne  $\int_0^1 e^{-x^2} dx$ !

**Beispiel 1.2.** Berechne  $\sin(20)$ !

**Beispiel 1.3.** Berechne  $\sqrt{753}$ !

**Beispiel 1.4.** Berechne  $\min_{x \in [0,1]} F(x)$ , für eine geeignete Funktion  $F$ !

**Beispiel 1.5.** Berechne  $x$ , sodass  $f(x) = 0$ !

**Beispiel 1.6.** Berechne  $x \in \mathbb{R}^n$ , sodass  $Ax = b$ !

**Definition 1.7.** 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 1.8.** 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 1.9.** 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  $\phi$ , sodass  $x = \phi(d)$  das Problem löst, d.h.  $f(\phi(d), d) = 0$ .

**Definition 1.10.** Ein numerischer Algorithmus zur näherungsweisen Lösung einer wohlgestellten Aufgabe  $\phi$  ist eine Abbildung  $\tilde{\phi}$ , die durch Hintereinanderausführung möglicherweise fehlerbehafteter elementarer Rechenoperationen definiert ist, also

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

**Definition 1.11.** Der **Aufwand** eines Verfahrens  $\tilde{\phi}$  ist die Anzahl der benötigten elementaren Rechenschritte. Typischerweise interessiert uns nicht die exakte Anzahl an Schritten, sondern nur die Größenordnung.

**Proposition 1.12.** *Das Gaußverfahren hat Aufwand  $\mathcal{O}(n^3)$ .*

## Chapter 2

# Numerische Lineare Algebra

## 2.1 Matrixfaktorisierung

### 2.1.1 Dreiecksmatrizen

**Definition 2.1.** Eine Matrix  $L \in \mathbb{R}^{n \times n}$  heißt **untere Dreiecksmatrix**, falls  $\forall i < j : l_{ij} = 0$ .

**Definition 2.2.** Eine Matrix  $U \in \mathbb{R}^{n \times n}$  heißt **obere Dreiecksmatrix**, falls  $U^\top$  eine untere Dreiecksmatrix ist.

**Definition 2.3.** Eine Dreiecksmatrix heißt **normalisiert**, falls alle ihre Diagonaleinträge 1 sind.

**Definition 2.4.** Eine Matrix heißt **regulär**, wenn sie invertierbar ist.

**Lemma 2.5.** *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 2.6.** 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 2.7.** 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 2.8.**

- 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 2.9.** 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 2.10.** Der Rechenauftrag dieses Algorithmus beträgt  $O(n^3)$ .

**Proposition 2.11.** 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 2.12.** Ein numerisches Problem  $\phi$  heißt **schlecht Konditioniert**, wenn kleine Unterschiede in der Eingabe zu großen Unterschieden in der korrekten Lösung führen, also wenn

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

Ansonsten heißt die Aufgabe **gut konditioniert**.

**Definition 2.13.** Ein Verfahren  $\tilde{\phi}$  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 2.14.** 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  $1/\varepsilon$ !!! So können wir keine Numerik machen!!! Dieses Problem ist **schlecht konditioniert**.

**Beispiel 2.15.** 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. Die  $LU$ -Zerlegung von  $A$  ist jedoch gegeben durch

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

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 2.16. 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  $l_{ii} > 0$*