Höhere Mathematik

Jil Zerndt, Lucien Perret
May 2024

Rechnerarithmetik

Zahlendarstellung

Maschinenzahlen Eine maschinendarstellbare Zahl zur Basis B ist ein Element der Menge:

$$M = \{x \in \mathbb{R} \mid x = \pm 0.m_1 m_2 m_3 \dots m_n \cdot B^{\pm e_1 e_2 \dots e_l}\} \cup \{0\}$$

Mit:
- $m_1 \neq 0$ (Normalisierungsbedingung)
- $m_i, e_i \in \{0, 1, \dots, B-1\}$ für $i \neq 0$
- $B \in \mathbb{N}, B > 1$ (Basis)

Zahlenwert Der Wert $\hat{\omega}$ einer Maschinenzahl berechnet sich durch:

$$\hat{\omega} = \sum_{i=1}^{n} m_i B^{\hat{e}-i}, \quad \text{mit} \quad \hat{e} = \sum_{i=1}^{l} e_i B^{l-i}$$

Werteberechnung Berechnung einer vierstelligen Zahl zur Basis 4:

$$\underbrace{0.3211}_{n=4} \cdot \underbrace{4^{12}}_{l=2}$$

1. Exponent: $\hat{e} = 1 \cdot 4^1 + 2 \cdot 4^0 = 6$
2. Wert: $\hat{\omega} = 3 \cdot 4^5 + 2 \cdot 4^4 + 1 \cdot 4^3 + 1 \cdot 4^2 = 3664$

IEEE-754 Standard
Der IEEE-754 Standard definiert zwei wichtige Gleitpunktformate:

Single Precision (32 Bit)
- Vorzeichen (V): 1 Bit
- Exponent (E): 8 Bit (Bias 127)
- Mantisse (M): 23 Bit + 1 hidden bit

Double Precision (64 Bit)
- Vorzeichen (V): 1 Bit
- Exponent (E): 11 Bit (Bias 1023)
- Mantisse (M): 52 Bit + 1 hidden bit

Darstellungsbereich Für jedes Gleitpunktsystem existieren:
- Grösste darstellbare Zahl: $x_{\max} = (1 - B^{-n}) \cdot B^{e_{\max}}$
- Kleinste darstellbare positive Zahl: $x_{\min} = B^{e_{\min}-1}$

Approximations- und Rundungsfehler

Fehlerarten Sei $\tilde{x}$ eine Näherung des exakten Wertes x:

Absoluter Fehler: $|\tilde{x} - x|$

Relativer Fehler: $\left|\frac{\tilde{x} - x}{x}\right|$ bzw. $\frac{|\tilde{x} - x|}{|x|}$ für $x \neq 0$

Maschinengenauigkeit eps ist die kleinste positive Zahl, für die gilt:

Allgemein: $\text{eps} := \frac{B}{2} \cdot B^{-n}$

Dezimal: $\text{eps}_{10} := 5 \cdot 10^{-n}$

Sie begrenzt den maximalen relativen Rundungsfehler:

$$\left|\frac{rd(x) - x}{x}\right| \leq \text{eps}$$

Rundungseigenschaften Für alle $x \in \mathbb{R}$ mit $|x| \geq x_{\min}$ gilt:

Absoluter Fehler: $|rd(x) - x| \leq \frac{B}{2} \cdot B^{e-n-1}$

Relativer Fehler: $\left|\frac{rd(x) - x}{x}\right| \leq \text{eps}$

Fehlerfortpflanzung

Konditionierung
Die Konditionszahl K beschreibt die relative Fehlervergrösserung bei Funktionsauswertungen:

$$K := \frac{|f'(x)| \cdot |x|}{|f(x)|}$$

Interpretation
- $K \leq 1$: gut konditioniert
- $K > 1$: schlecht konditioniert
- $K \gg 1$: sehr schlecht konditioniert

Fehlerfortpflanzung
Für eine differenzierbare Funktion f gilt näherungsweise:

Absoluter Fehler: $|f(\tilde{x}) - f(x)| \approx |f'(x)| \cdot |\tilde{x} - x|$

Relativer Fehler: $\frac{|f(\tilde{x}) - f(x)|}{|f(x)|} \approx K \cdot \frac{|\tilde{x} - x|}{|x|}$

Fehleranalyse einer Funktion
So analysieren Sie die Fehlerfortpflanzung einer Funktion:
1. Berechnen Sie $f'(x)$
2. Bestimmen Sie die Konditionszahl K
3. Schätzen Sie den absoluten Fehler ab
4. Schätzen Sie den relativen Fehler ab
5. Beurteilen Sie die Konditionierung anhand von K

$$\underbrace{|f(\tilde{x}) - f(x)|}_{\text{absoluter Fehler von } f(x)} \approx \left|f'(x)\right| \cdot \underbrace{|\tilde{x} - x|}_{\text{absoluter Fehler von } x}$$

$$\underbrace{\frac{|f(\tilde{x}) - f(x)|}{|f(x)|}}_{\text{relativer Fehler von } f(x)} \approx \underbrace{\frac{\left|f'(x)\right| \cdot |x|}{|f(x)|}}_{\text{Konditionszahl } K} \cdot \underbrace{\frac{|\tilde{x} - x|}{|x|}}_{\text{relativer Fehler von } x}$$

Fehleranalyse Beispiel: Fehleranalyse von $f(x) = \sin(x)$
1. $f'(x) = \cos(x)$
2. $K = \frac{|x \cos(x)|}{|\sin(x)|}$
3. Für $x \to 0$: $K \to 1$ (gut konditioniert)
4. Für $x \to \pi$: $K \to \infty$ (schlecht konditioniert)
5. Der absolute Fehler wird nicht vergrössert, da $|\cos(x)| \leq 1$

Auslöschung Besonders problematisch: Auslöschung
Bei der Subtraktion fast gleich großer Zahlen können signifikante Stellen verloren gehen. Beispiel:
- $1.234567 - 1.234566 = 0.000001$
- Aus 7 signifikanten Stellen wird 1 signifikante Stelle

Praktische Fehlerquellen

Kritische Operationen
Die häufigsten Quellen für numerische Fehler sind:
- Auslöschung bei Subtraktion ähnlich großer Zahlen
- Überlauf (overflow) bei zu großen Zahlen
- Unterlauf (underflow) bei zu kleinen Zahlen
- Verlust signifikanter Stellen durch Rundung

Auslöschung bei der Berechnung von $\sqrt{x^2+1} - 1$:
Für kleine x führt die direkte Berechnung zu Auslöschung:
- Für $x = 10^{-8}$:
- $\sqrt{10^{-16}+1} - 1 \approx 1.000000000 - 1 = 0$
- Korrekte Lösung durch Umformung:
- $\sqrt{x^2+1} - 1 = \frac{x^2}{\sqrt{x^2+1}+1}$

Vermeidung von Auslöschung
So vermeiden Sie Auslöschungseffekte:
1. Identifizieren Sie Subtraktionen ähnlich großer Zahlen
2. Suchen Sie nach algebraischen Umformungen
3. Prüfen Sie alternative Berechnungswege
4. Verwenden Sie Taylorentwicklungen für kleine Werte

Analyse von Algorithmen

Fehlerakkumulation
Bei n aufeinanderfolgenden Operationen mit relativen Fehlern $\leq \varepsilon$ gilt für den Gesamtfehler:
- Best case: $\mathcal{O}(n\varepsilon)$ bei gleichverteilten Fehlern
- Worst case: $\mathcal{O}(2^n\varepsilon)$ bei systematischen Fehlern

Numerische Stabilität
Ein Algorithmus heißt numerisch stabil, wenn:
- Kleine Eingabefehler zu kleinen Ausgabefehlern führen
- Rundungsfehler sich nicht übermäßig akkumulieren
- Die Konditionszahl des Problems nicht künstlich verschlechtert wird

Instabilität Instabiles Verhalten bei rekursiver Berechnung: Berechnung der Fibonacci-Zahlen:

```python
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```

Problem: Exponentielles Wachstum der Operationen und Fehlerfortpflanzung.

Stabilitätsanalyse

Schritte zur Analyse der numerischen Stabilität:

1. Bestimmen Sie kritische Operationen
2. Schätzen Sie Rundungsfehler pro Operation ab
3. Analysieren Sie die Fehlerfortpflanzung
4. Berechnen Sie die worst-case Fehlerschranke
5. Vergleichen Sie alternative Implementierungen

Praktische Implementierungen

Implementierungsgenauigkeit

Die Implementierungsgenauigkeit eines Algorithmus beschreibt:

- Relative Genauigkeit der Ausgabe
- Maximale Anzahl korrekter Dezimalstellen
- Stabilität gegenüber Eingabefehlern

Robuste Implementierung

So implementieren Sie numerisch robuste Algorithmen:

1. Verwenden Sie stabile Grundoperationen
2. Vermeiden Sie Differenzen ähnlich großer Zahlen
3. Prüfen Sie auf Über- und Unterlauf
4. Implementieren Sie Fehlerkontrollen
5. Dokumentieren Sie numerische Einschränkungen

Robuste Implementation Beispiel: Quadratische Gleichung

```python
def quadratic_stable(a, b, c):
    # ax^2 + bx + c = 0
    if a == 0:
        return [-c/b] if b != 0 else []

    # Calculate discriminant
    disc = b*b - 4*a*c
    if disc < 0:
        return []

    # Choose numerically stable formula
    if b >= 0:
        q = -0.5*(b + sqrt(disc))
    else:
        q = -0.5*(b - sqrt(disc))

    x1 = q/a
    x2 = c/(q)
    return sorted([x1, x2])
```

Numerische Bibliotheken Verwendung spezialisierter Bibliotheken
Für kritische numerische Berechnungen:

- NumPy: Optimierte Array-Operationen
- SciPy: Wissenschaftliches Rechnen
- Mpmath: Beliebige Präzision
- Decimal: Dezimalarithmetik

Bibliotheksverwendung Beispiel: Präzise Berechnung mit Decimal

```python
from decimal import Decimal, getcontext

# Set precision
getcontext().prec = 40

# Precise calculation
x = Decimal('1.0') / Decimal('7.0')
print(x)  # 0.1428571428571428571428571428571428571428
```

Numerische Lösung von Nullstellenproblemen

Fixpunktgleichung

Eine Gleichung der Form $F(x) = x$ heisst Fixpunktgleichung. Die Lösungen $\bar{x}$, für die $F(\bar{x}) = \bar{x}$ erfüllt ist, heissen Fixpunkte.

Fixpunktiteration

Grundprinzip der Fixpunktiteration

Gegeben sei $F : [a, b] \to \mathbb{R}$ mit $x_0 \in [a, b]$. Die rekursive Folge

$$x_{n+1} \equiv F(x_n), \quad n = 0, 1, 2, \ldots$$

heisst Fixpunktiteration von F zum Startwert x_0.

Konvergenzverhalten

Sei $F : [a, b] \to \mathbb{R}$ mit stetiger Ableitung F' und $\bar{x} \in [a, b]$ ein Fixpunkt von F. Dann gilt für die Fixpunktiteration $x_{n+1} = F(x_n)$:

Anziehender Fixpunkt:	**Abstossender Fixpunkt:**
$\lvert F'(\bar{x})\rvert < 1$	$\lvert F'(\bar{x})\rvert > 1$
x_n konvergiert gegen $\bar{x}$,	x_n konvergiert für keinen
falls x_0 nahe genug bei $\bar{x}$	Startwert $x_0 \neq \bar{x}$

Banachscher Fixpunktsatz

Sei $F : [a, b] \to [a, b]$ und es existiere eine Konstante α mit:

- $0 < \alpha < 1$ (Lipschitz-Konstante)
- $|F(x) - F(y)| \leq \alpha|x - y|$ für alle $x, y \in [a, b]$

Dann gilt:

- F hat genau einen Fixpunkt $\bar{x}$ in $[a, b]$
- Die Fixpunktiteration konvergiert gegen $\bar{x}$ für alle $x_0 \in [a, b]$
- Fehlerabschätzungen:
  - a-priori: $|x_n - \bar{x}| \leq \frac{\alpha^n}{1-\alpha} \cdot |x_1 - x_0|$
  - a-posteriori: $|x_n - \bar{x}| \leq \frac{\alpha}{1-\alpha} \cdot |x_n - x_{n-1}|$

Konvergenznachweis für Fixpunktiteration

So überprüfen Sie, ob eine Fixpunktiteration konvergiert:

1. Prüfen Sie, ob $F : [a, b] \to [a, b]$ gilt:
   $F(a) > a$ und $F(b) < b$
2. Bestimmen Sie $\alpha = \max_{x \in [a,b]} |F'(x)|$
3. Prüfen Sie, ob $\alpha < 1$
4. Berechnen Sie die nötigen Iterationen für Toleranz tol:
   $n \geq \frac{\ln(\frac{tol \cdot (1-\alpha)}{|x_1 - x_0|})}{\ln \alpha}$

Fixpunktiteration Nullstellen von $p(x) = x^3 - x + 0.3$
Fixpunktgleichung: $x_{n+1} = F(x_n) = x_n^3 + 0.3$

1. $F'(x) = 3x^2$ steigt monoton
2. Für $I = [0, 0.5]$: $F(0) = 0.3 > 0$, $F(0.5) = 0.425 < 0.5$
3. $\alpha = \max_{x \in [0,0.5]} |3x^2| = 0.75 < 1$
4. Konvergenz für Startwerte in $[0, 0.5]$ gesichert

Newton-Verfahren

Grundprinzip Newton-Verfahren

Approximation der Nullstelle durch sukzessive Tangentenberechnung:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Konvergiert, wenn für alle x im relevanten Intervall gilt:

$$\left| \frac{f(x) \cdot f''(x)}{[f'(x)]^2} \right| < 1$$

Newton-Verfahren anwenden

So finden Sie eine Nullstelle mit dem Newton-Verfahren:

1. Funktion $f(x)$ und Ableitung $f'(x)$ aufstellen
2. Geeigneten Startwert x_0 nahe der Nullstelle wählen
3. Iterieren bis zur gewünschten Genauigkeit:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

4. Konvergenz prüfen durch Vergleich aufeinanderfolgender Werte

Vereinfachtes Newton-Verfahren

Alternative Variante mit konstanter Ableitung:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}$$

Konvergiert langsamer, aber benötigt weniger Rechenaufwand.

Sekantenverfahren

Alternative zum Newton-Verfahren ohne Ableitungsberechnung. Verwendet zwei Punkte $(x_{n-1}, f(x_{n-1}))$ und $(x_n, f(x_n))$:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \cdot f(x_n)$$

Benötigt zwei Startwerte x_0 und x_1.

Konvergenzverhalten

Konvergenzordnung

Sei (x_n) eine gegen $\bar{x}$ konvergierende Folge. Die Konvergenzordnung $q \geq 1$ ist definiert durch:

$$|x_{n+1} - \bar{x}| \leq c \cdot |x_n - \bar{x}|^q$$

wobei $c > 0$ eine Konstante ist. Für $q = 1$ muss zusätzlich $c < 1$ gelten.

Konvergenzordnungen der Verfahren

Die verschiedenen Verfahren zeigen folgende Konvergenzgeschwindigkeiten:

Newton-Verfahren: Quadratische Konvergenz $q = 2$

Vereinfachtes Newton: Lineare Konvergenz $q = 1$

Sekantenverfahren: Superlineare Konvergenz $q = \frac{1+\sqrt{5}}{2} \approx 1.618$

Konvergenzgeschwindigkeit Vergleich der Verfahren Startwert $x_0 = 1$, Funktion $f(x) = x^2 - 2$, Ziel: $\sqrt{2}$

n	Newton	Vereinfacht	Sekanten
1	1.5000000	1.5000000	1.5000000
2	1.4166667	1.4500000	1.4545455
3	1.4142157	1.4250000	1.4142857
4	1.4142136	1.4125000	1.4142136

Fehlerabschätzung

Nullstellensatz von Bolzano

Sei $f : [a, b] \to \mathbb{R}$ stetig. Falls $f(a) \cdot f(b) < 0$, dann existiert mindestens eine Nullstelle $\xi \in (a, b)$.

Fehlerabschätzung für Nullstellen

So schätzen Sie den Fehler einer Näherungslösung ab:

1. Sei x_n der aktuelle Näherungswert
2. Wähle Toleranz $\epsilon > 0$
3. Prüfe Vorzeichenwechsel: $f(x_n - \epsilon) \cdot f(x_n + \epsilon) < 0$
4. Falls ja: Nullstelle liegt in $(x_n - \epsilon, x_n + \epsilon)$
5. Damit gilt: $|x_n - \xi| < \epsilon$

Praktische Fehlerabschätzung Fehlerbestimmung bei $f(x) = x^2 - 2$

1. Näherungswert: $x_3 = 1.4142157$
2. Mit $\epsilon = 10^{-5}$:
3. $f(x_3 - \epsilon) = 1.4142057^2 - 2 < 0$
4. $f(x_3 + \epsilon) = 1.4142257^2 - 2 > 0$
5. Also: $|x_3 - \sqrt{2}| < 10^{-5}$

Abbruchkriterien Praktische Implementierung

In der Praxis verwendet man meist mehrere Abbruchkriterien:

- Absolute Änderung: $|x_n - x_{n-1}| < \epsilon_1$
- Funktionswert: $|f(x_n)| < \epsilon_2$
- Maximale Iterationszahl: $n < n_{max}$
- Kombination dieser Kriterien

Numerische Lösung linearer Gleichungssysteme

Grundlagen

Lineares Gleichungssystem

Ein lineares Gleichungssystem der Form $Ax = b$ besteht aus:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

Der Gauss-Algorithmus

Grundidee Gauss-Elimination

Transformation des Gleichungssystems $Ax = b$ in ein äquivalentes System $\tilde{A}x = \tilde{b}$, wobei $\tilde{A}$ eine obere Dreiecksmatrix ist.

Erlaubte Operationen:

- $z_j := z_j - \lambda z_i$ für $i < j$ und $\lambda \in \mathbb{R}$
- $z_i \to z_j$ (Vertauschen von Zeilen)

Gauss-Algorithmus

So lösen Sie ein lineares Gleichungssystem mit dem Gauss-Algorithmus:

1. Elimination (Vorwärts)

1. Für $i = 1, \ldots, n-1$:
2. Für $j = i+1, \ldots, n$:
3. Berechne $\lambda_{ji} = a_{ji}/a_{ii}$
4. $z_j := z_j - \lambda_{ji} z_i$

2. Rückwärtseinsetzen

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}, \quad i = n, n-1, \ldots, 1$$

Gauss-Elimination Lösung eines 3×3 Systems Gegebenes System:

$$A = \begin{pmatrix} 2 & 1 & -1 \\ 4 & -1 & 3 \\ -2 & 2 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 1 \\ 4 \end{pmatrix}$$

1. Elimination der ersten Spalte:

$$\left(\begin{array}{ccc|c} 2 & 1 & -1 & 3 \\ 0 & -3 & 5 & -5 \\ 0 & 3 & -1 & 7 \end{array}\right)$$

2. Elimination der zweiten Spalte:

$$\left(\begin{array}{ccc|c} 2 & 1 & -1 & 3 \\ 0 & -3 & 5 & -5 \\ 0 & 0 & 4 & 2 \end{array}\right)$$

3. Rückwärtseinsetzen:

$$x_3 = \frac{2}{4} = \frac{1}{2}$$

$$x_2 = \frac{-5 - 5(\frac{1}{2})}{-3} = -2$$

$$x_1 = \frac{3 - 1(-2) - (-1)(\frac{1}{2})}{2} = 1$$

Pivotisierung

Spaltenpivotisierung

Strategie zur numerischen Stabilisierung des Gauss-Algorithmus durch Auswahl des betragsmäßig größten Elements als Pivotelement.

Vor jedem Eliminationsschritt in Spalte i:

- Suche k mit $|a_{ki}| = \max\{|a_{ji}| \mid j = i, \ldots, n\}$
- Falls $a_{ki} \neq 0$: Vertausche Zeilen i und k
- Falls $a_{ki} = 0$: Matrix ist singulär

Gauss mit Pivotisierung

Erweiterter Gauss-Algorithmus mit Spaltenpivotisierung:

1. Für $i = 1, \ldots, n-1$:
2. Finde $k \geq i$ mit $|a_{ki}| = \max\{|a_{ji}| \mid j = i, \ldots, n\}$
3. Falls $a_{ki} = 0$: Stop (Matrix singulär)
4. Vertausche Zeilen i und k
5. Für $j = i+1, \ldots, n$:
6. $z_j := z_j - \frac{a_{ji}}{a_{ii}} z_i$

Pivotisierung Gauss-Elimination mit Pivotisierung System:

$$\begin{pmatrix} 1 & 2 & -1 \\ 4 & 8 & -3 \\ 9 & 18 & -8 \end{pmatrix} x = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix}$$

1. Erste Spalte: Pivot $|9| \rightarrow$ Tausche Zeilen 1 und 3
2. Nach Elimination der ersten Spalte:

$$\left(\begin{array}{ccc|c} 9 & 18 & -8 & 9 \\ 0 & 0 & 0.89 & 0 \\ 0 & 0 & 0.89 & 0 \end{array}\right)$$

3. System ist schlecht konditioniert (identische Zeilen)

Matrix-Zerlegungen

Dreieckszerlegung

Eine Matrix $A \in \mathbb{R}^{n \times n}$ kann zerlegt werden in:

Untere Dreiecksmatrix L:	**Obere Dreiecksmatrix R:**
$l_{ij} = 0$ für $j > i$	$r_{ij} = 0$ für $i > j$
Diagonale meist normiert ($l_{ii} = 1$)	Diagonalelemente $\neq 0$

LR-Zerlegung

Jede reguläre Matrix A, für die der Gauss-Algorithmus ohne Zeilenvertauschungen durchführbar ist, lässt sich zerlegen in:

$$A = LR$$

wobei L eine normierte untere und R eine obere Dreiecksmatrix ist.

Berechnung der LR-Zerlegung

So berechnen Sie die LR-Zerlegung:

1. Führen Sie Gauss-Elimination durch
2. R ist die resultierende obere Dreiecksmatrix
3. Die Eliminationsfaktoren $-\frac{a_{ji}}{a_{ii}}$ bilden L
4. Lösen Sie dann nacheinander:
   - $Ly = b$ (Vorwärtseinsetzen)
   - $Rx = y$ (Rückwärtseinsetzen)

LR-Zerlegung Berechnung einer LR-Zerlegung Matrix:

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & -1 & 0 \\ -2 & 3 & 1 \end{pmatrix}$$

Schrittweise Elimination führt zu:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 2 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 2 & 1 & 1 \\ 0 & -3 & -2 \\ 0 & 0 & -2 \end{pmatrix}$$