Einleitung

Numerische Mathematik:

Zahlenmäßige Lösung eines mathem. Problems mit Hilfe des Computers

Historische Bemerkungen:

a) Mechanische Rechner:

Blaise Pascal (1623 - 1662)

Gottfried W. Leibniz (1646 - 1716)

b) Elektronische Rechner:

Alan M. Turing (1912 - 1954)

Johann v. Neumann (1903 - 1957)

c) Rasante Entwicklung 1950 – heute

- floating-point representation
- Fehleranalyse numerischer Verfahren

Lanczos, Givens, Wilkinson, ...

Problem (vereinfacht)

Eine vorgegebene Funktion $f : \mathbb{R} \to \mathbb{R}$ ist an einer vorgegebenen Stelle $x \in \mathbb{R}$ auszuwerten : $y = f(x)$.

x : **Eingangsgröße, Eingabe**

y : **Ergebnis, Resultat**

Prinzipieller Ablauf :

- Eingabe in den Rechner:

  $x \longmapsto \tilde{x}$ **Maschinenzahl**

- Ersetzung von f durch einen **Algorithmus** (= endliche Zahl elementarer Rechen=Operationen $+, -, *, /$ mit festgelegter Reihenfolge)

  $f \longmapsto \tilde{f}$

- **Numerisches Resultat :** $\tilde{y} = \tilde{f}(\tilde{x})$

- Weitere Verfälschung : **Rundungsfehler**

Aufgaben der Numerischen Mathem.:

- Konstruktion von Algorithmen
- Analyse von Algorithmen bezüglich
  - Genauigkeit
  - Rundungsfehlereinfluss
  - Effizienz (Aufwand)
  - Zuverlässigkeit

1. Fehleranalyse

1.1 Maschinenzahlen $\mathbb{M}$

Auf einem Computer sind nur endlich viele reelle Zahlen darstellbar, diese heißen **Maschinenzahlen**.

Darstellung:

$$\begin{aligned} x &= \pm\left(x_1 g^{-1} + x_2 g^{-2} + \ldots + x_L g^{-L}\right) * g^e \\ &= \pm\left(0.x_1 x_2 \ldots x_L\right)_g * g^e \end{aligned} \qquad (1.1)$$

g: **Basis**, $g \in \mathbb{N}$, $g > 1$
(zumeist $g = 2, 10, 16$)

e: **Exponent**, $e \in \mathbb{Z}$, $e_{min} \leq e \leq e_{max}$

x_i: **Ziffern**, $x_i \in \{0, 1, \ldots, g-1\}$

Die Ziffernfolge $x_1 x_2 \ldots x_L$ heißt **Mantisse**

Zumeist wird $x_1 \neq 0$ verlangt (für $x \neq 0$)

(Normalisierte Gleitpunkt-Darstellung)

L: **Mantissenlänge**

Beispiel: (Intel, IEC/IEEE-Norm)

einfach genau: $g=2$, $L=24$,
$e_{min} = -125$, $e_{max} = 128$

doppelt genau: $g=2$, $L=53$,
$e_{min} = -1021$, $e_{max} = 1024$

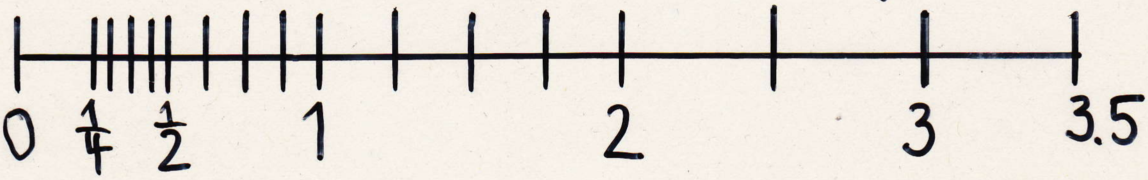
Folgerung:

a) Es gibt nur endlich viele Maschinenzahlen, nämlich $2(g-1)\,g^{L-1}(e_{max} - e_{min} + 1) + 1$

b) Es gibt keine beliebig großen und keine beliebig kleinen Maschinenzahlen,
$x_{max} = g^{e_{max}}(1 - g^{-L})$ ist die größte,
$x_{min} = g^{e_{min}-1}$ die kleinste pos. Masch.zahl

c) $x, y \in \mathbb{M} \Rightarrow$ I. Allg. $x \pm y,\ x * y,\ x/y \notin \mathbb{M}$

d) Die Maschinenzahlen sind nicht gleichverteilt:



($g=2$, $L=3$, $e_{min}=-1$, $e_{max}=2$)

1.2 Absolute u. relative Fehler

Ist $\tilde{x}$ Näherung für $x \in \mathbb{R}$, so heißt

$$\Delta x := \tilde{x} - x \quad \text{absoluter Fehler}$$

$$\varepsilon_x := \frac{\tilde{x} - x}{x} \quad \text{relativer Fehler} \qquad (1.2)$$

(für $x \neq 0$)

Beispiel:

a) $\left.\begin{aligned} x &= 0.1237 \cdot 10^8 \\ \tilde{x} &= 0.1238 \cdot 10^8 \end{aligned}\right\} \Rightarrow \begin{aligned} \Delta x &= 10^4 \\ \varepsilon_x &\doteq 0.8 \cdot 10^{-3} \end{aligned}$

b) $\left.\begin{aligned} x &= 0.7321 \cdot 10^{-5} \\ \tilde{x} &= 0.7921 \cdot 10^{-5} \end{aligned}\right\} \Rightarrow \begin{aligned} \Delta x &= 6 \cdot 10^{-7} \\ \varepsilon_x &\doteq 0.8 \cdot 10^{-1} \end{aligned}$

Merkregel: (1.3)

Gilt für den relativen Fehler $\varepsilon_x = \alpha \cdot 10^{-k}$ mit $0.1 \leq |\alpha| < 1$, $k \in \mathbb{N}$, so besitzt $\tilde{x}$ etwa k korrekte Dezimalziffern.

Erläuterung:

$$x = 0.x_1 \ldots x_k\, x_{k+1} \ldots * 10^e \,, \quad x_1 \neq 0$$

$$\tilde{x} = 0.x_1 \ldots x_k\, \tilde{x}_{k+1} \ldots * 10^e \,, \quad \tilde{x}_{k+1} \neq x_{k+1}$$

$\Longrightarrow$

$$\varepsilon_x = \frac{\tilde{x} - x}{x} = \underbrace{\frac{0.\tilde{x}_{k+1} \ldots - 0.x_{k+1} \ldots}{0.x_1 x_2 \ldots}}_{=:\, \alpha} * 10^{-k}$$

wobei $0.1 \leq |\alpha| \leq 10$.

1.3 Rundungsfehler

Neben den **Datenfehlern** $(x \mapsto \tilde{x})$ und den **Verfahrensfehlern** $(f \mapsto \tilde{f})$ spielen Rundungsfehler eine wichtige Rolle für die Beurteilung eines Algorithmus.

Rundungsfehler: Approximation einer reellen Zahl x durch eine Maschinenzahl $\tilde{x} \in \mathbb{M}$. Dies tritt i. Allg. bei jeder element. Rechenoperation auf.

Gegeben: $x = \pm (0.x_1 x_2 .. x_L x_{L+1} \dots)_g * g^e$

mit $x_1 \neq 0$, $e_{min} \leq e \leq e_{max}$

Gesucht: Eine zu x benachbarte Maschinenzahl $\tilde{x} \in \mathbb{M}$.

a) Abschneiden:

$$\tilde{x} := rd_1(x) := \pm (0.x_1 \dots x_L)_g * g^e$$

$$\Longrightarrow \quad \tilde{x} = x(1+\varepsilon) \quad , \quad |\varepsilon| \leq g^{1-L} \qquad \underline{(1.4)}$$

denn:

$$\varepsilon_x = \frac{\tilde{x}-x}{x} = - \frac{(0.0..0\,x_{L+1}\dots)\,g^e}{(0.x_1 x_2 \dots\dots)\,g^e}$$

$$= - \underbrace{\frac{(0.x_{L+1}x_{L+2}\dots)_g}{(0.x_1 x_2 \dots)_g}}_{|\dots| \leq g} \cdot g^{-L} \qquad \blacktriangle$$

b) Runden:

$$\tilde{x} := rd_2(x) :=$$

$$:= \begin{cases} (0.x_1 \dots x_L)\,g^e\,, & \text{falls } x_{L+1} < g/2 \\ (0.x_1 \dots x_L)\,g^e + g^{e-L}\,, & \text{sonst} \end{cases}$$

$$\Longrightarrow \quad \boxed{\tilde{x} = x(1+\varepsilon), \quad |\varepsilon| \le \tfrac{1}{2} g^{1-L}}$$

Zusatz: Falls $x_j = g-1$ ($\forall j = 1, \ldots, L$) und $x_{L+1} \geq g/2$, so setze $rd_2(x) := (0.10\ldots0)\, g^{e+1}$ für $e < e_{max}$ und "overflow" für $e = e_{max}$.

Definition (1.5) Die Zahl $eps := \frac{1}{2} g^{1-L}$ heißt **relative Maschinengenauigkeit**.

Beispiel (IEC/IEEE Norm)

single precision : $g=2, L=24, eps \doteq 0.6 \cdot 10^{-7}$
double precision : $g=2, L=53, eps \doteq 1.1 \cdot 10^{-16}$
extended : $g=2, L=64, eps \doteq 0.5 \cdot 10^{-19}$

Gleitpunkt-Rechnung:

Die Grundoperationen $+, -, *, /$ werden in höherer Genauigkeit ausgeführt, normalisiert und auf L Stellen gerundet.

Bezeichnung: $fl(a \pm b)$, $fl(a * b)$, $fl(a/b)$

10

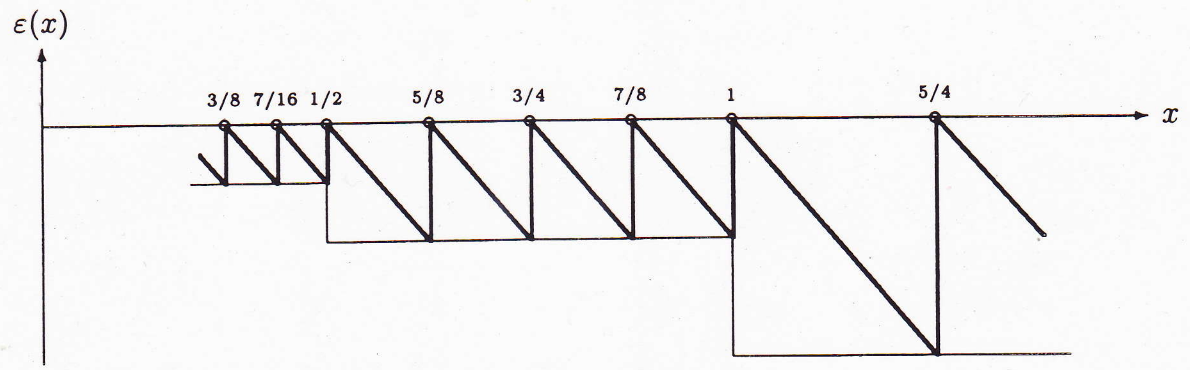


Abb. 4.6: Absoluter Rundungsfehler bei Rundung durch Abschneiden

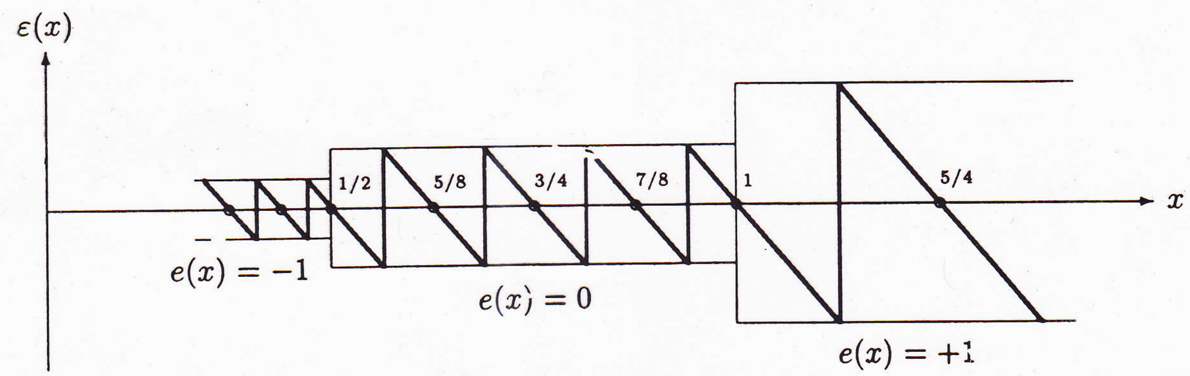


Abb. 4.7: Absoluter Rundungsfehler bei optimaler Rundung

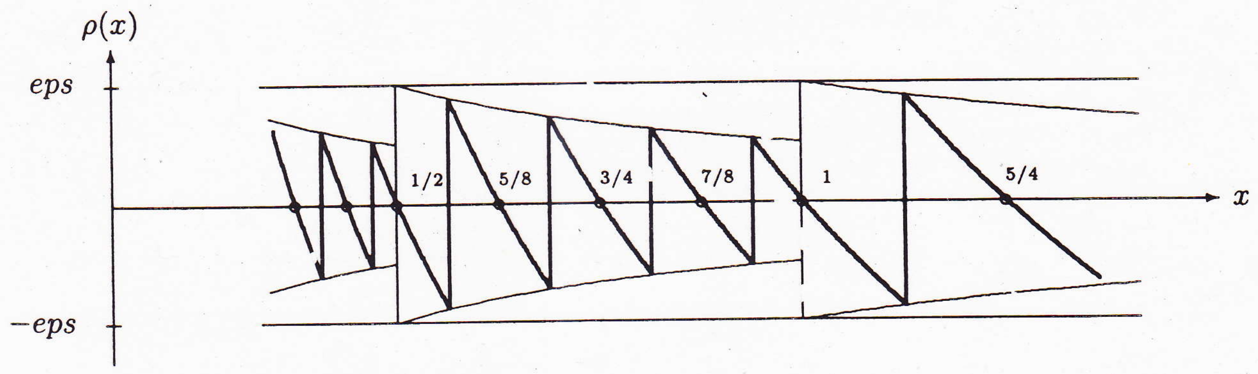


Abb. 4.8: Relativer Rundungsfehler bei optimaler Rundung

Genauigkeit:

$$fl(a \pm b) = (a \pm b)(1+\sigma)$$
$$fl(a * b) = (a * b)(1+\mu) \qquad \underline{(1.6)}$$
$$fl(a/b) = (a/b)(1+\delta)$$

mit $|\sigma|, |\mu|, |\delta| \leq eps$

1.4 Kondition

Ein mathem. Problem – z.B. $y = f(x)$ – heißt **gut konditioniert**, falls kleine Änderungen der Eingangsdaten nur zu kleinen Änderungen im Resultat führt (bei exakter Rechnung!)

Andernfalls:

schlecht konditioniert ... chaotisch

Quantitativ: $(x, f(x) \neq 0)$

$$\Delta y = f(\tilde{x}) - f(x) \approx f'(x)\, \Delta x$$

$$\varepsilon_y = \frac{\Delta y}{y} \approx \frac{f'(x)}{f(x)} \Delta x = \frac{x \cdot f'(x)}{f(x)} \cdot \varepsilon_x$$

Definition (1.7) Die Größen

$$\kappa_{abs} := f'(x), \quad \kappa_{rel} := \frac{x\, f'(x)}{f(x)}$$

heißen **absolute u. relative Konditionszahlen.**

Bei mehreren Eingangsdaten $y = f(x_1, \dots, x_n)$ addieren sich die Fehler, die durch die Verfälschung der einzelnen x_i hervorgerufen werden (→ Analysis II)

Man erhält:

$$\boxed{\begin{aligned} \Delta y &\approx \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x)\, \Delta x_i \quad \text{(absoluter Fehler)} \\ \varepsilon_y &\approx \sum_{i=1}^{n} \frac{x_i}{f(x)} \frac{\partial f}{\partial x_i}(x) \cdot \varepsilon_{x_i} \quad \text{(rel. Fehler)} \end{aligned}} \tag{1.8}$$

Die $\frac{\partial f}{\partial x_i}(x)$ heißen partielle Ableitungen von f. Dabei wird jeweils nur x_i als Variable angesehen, die anderen x_j für $j \neq i$ sind fest!

Die Zahlen $K_{i,abs} := \frac{\partial f}{\partial x_i}(x)$ und

$K_{i,rel} := \frac{x_i}{f(x)} \frac{\partial f}{\partial x_i}(x)$, $i = 1, \ldots, n$ heißen

absolute u. relative Konditionzahlen

bzgl. der Eingangsgröße x_i.

Beispiel (1.9) (Elementare Operationen)

$$\varepsilon_{x_1 \pm x_2} \approx \frac{x_1}{x_1 \pm x_2} \varepsilon_{x_1} \pm \frac{x_2}{x_1 \pm x_2} \varepsilon_{x_2}$$

$$\varepsilon_{x_1 \cdot x_2} \approx \varepsilon_{x_1} + \varepsilon_{x_2}$$

$$\varepsilon_{x_1 / x_2} \approx \varepsilon_{x_1} - \varepsilon_{x_2}$$

$\Rightarrow$ Multiplikation u. Division sind gut konditionierte Operationen, ebenso die Addition von Zahlen mit gleichen Vorzeichen.

Aber: Subtraktion von annähernd gleichen Zahlen führt zu ev. erheblicher Fehler= verstärkung. **Auslöschung!**

(Cancellation!)

Beispiel (1.10)

$$y = x_1 - x_2$$

exakt:

$$x_1 = 0.1000548241\,0 * 10^5$$
$$x_2 = 0.0999734221\,3 * 10^5$$

$$\begin{aligned} y &= 0.1000548241\,0 * 10^5 \\ &- 0.0999734221\,3 * 10^5 \\ &0.0000814019\,7 * 10^5 \\ &= \underline{0.8140197 * 10^1} \end{aligned}$$

Rechner mit $g = 10$ und $L = 5$:

$$\tilde{x}_1 = 0.10005 * 10^5$$
$$\tilde{x}_2 = 0.99973 * 10^4$$

$$\begin{aligned} \tilde{y} &= 0.1000500 * 10^5 \\ &- 0.0999730 * 10^5 \\ &= 0.0000770 * 10^5 \\ &= \underline{0.77000 * 10^1} \end{aligned}$$

Beispiel (1.11) (Skalarprodukt)

$$x = (x_1, \dots, x_n)^T, \quad y = (y_1, \dots, y_n)^T \in \mathbb{R}^n$$

$$\boxed{\langle x, y \rangle := \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \dots + x_n y_n}$$

Nach Formel (1.8) erhält man:

$$\varepsilon_{\langle x,y \rangle} \approx \sum_{i=1}^{n} \frac{x_i}{\langle x,y \rangle} y_i \, \varepsilon_{x_i} + \sum_{i=1}^{n} \frac{y_i}{\langle x,y \rangle} x_i \, \varepsilon_{y_i}$$

Annahme: $|\varepsilon_{x_i}|, |\varepsilon_{y_i}| \lessapprox \text{eps}$,

Vorzeichen stimmt mit denen der Vorfaktoren überein

$\Longrightarrow$

$$|\varepsilon_{\langle x,y \rangle}| \lessapprox \underbrace{2 \frac{\sum |x_i y_i|}{|\sum x_i y_i|}}_{\text{Konditionszahl}} \cdot \text{eps}$$

Ergebnis: Die Skalarprodukt-Berechnung kann schlecht konditioniert sein, wenn nämlich $|\sum x_i y_i|$ klein, aber $\sum |x_i y_i|$ groß ist. Der Grund ist wiederum:

Auslöschung!

Konkretes Beispiel: (Kulisch, Miranker, '86)

$$x := \begin{bmatrix} 2.718281828 \\ -3.141592654 \\ 1.414213562 \\ 0.5772156649 \\ 0.3010299957 \end{bmatrix}, \quad y := \begin{bmatrix} 1486.2497 \\ 878366.9879 \\ -22.37492 \\ 4773714.647 \\ 0.000185049 \end{bmatrix}$$

Numerische Ergebnisse:

FORTRAN single prec.: $\langle x,y \rangle \doteq -0.49994\ldots$

FORTRAN double prec.: $\langle x,y \rangle \doteq 0.10251\ldots \cdot 10^{-9}$

MATLAB: $\langle x,y \rangle \doteq -0.52750\ldots \cdot 10^{-10}$

Exaktes Ergebnis:

$$\underline{\langle x,y \rangle = -0.100657107 \cdot 10^{-10}}$$

1.5 Fehleranalyse, Stabilität

In einfachen Fällen lässt sich der Einfluss von Rundungsfehler folgendermaßen untersuchen

- Simulation der Gleitpunkt-Rechnung
- Linearisierung der Fehler

Beispiel (1.12) $y = a^2 - b^2 = a \cdot a - b \cdot b$

Gleitpunkt-Rechnung:

$$\tilde{y} = fl(a^2 - b^2)$$
$$= \left[a^2 (1+\varepsilon_a)^2 (1+\mu_1) - b^2 (1+\varepsilon_b)^2 (1+\mu_2) \right] (1+\sigma)$$

$\varepsilon_a, \varepsilon_b$: Datenfehler

μ_1, μ_2 : Multiplikationsfehler

σ : Subtraktionsfehler

Linearisierung:

$$(1+\alpha)(1+\beta) \approx 1 + \alpha + \beta$$
$$(1+\alpha)^k \approx 1 + k\alpha$$
$$\sqrt{1+\alpha} \approx 1 + \alpha/2$$

$\Longrightarrow$

$$\tilde{y} \approx a^2(1+2\varepsilon_a+\mu_1+\sigma) - b^2(1+2\varepsilon_b+\mu_2+\sigma)$$

$$= (a^2-b^2)\left\{1 + \frac{a^2}{a^2-b^2}(2\varepsilon_a+\mu_1) - \frac{b^2}{a^2-b^2}(2\varepsilon_b+\mu_2) + \right.$$
$$\left. + \sigma \right\}$$

$$= y \cdot \{1 + \varepsilon_y\}$$

$\Longrightarrow$

$$\varepsilon_y \approx \left(\frac{2a^2}{a^2-b^2}\varepsilon_a - \frac{2b^2}{a^2-b^2}\varepsilon_b\right) +$$
$$+ \frac{a^2}{a^2-b^2}\mu_1 - \frac{b^2}{a^2-b^2}\mu_2 + \sigma$$

$(\ldots) \mathrel{\widehat{=}}$ <u>Kondition</u> bzw. <u>Eingangsfehler</u>

Rest $\mathrel{\widehat{=}}$ <u>Rundungsfehler</u>

Definition (1.13)

Ein Algorithmus heißt **stabil**, wenn der Einfluss der Rundungsfehler nicht (wesentlich) größer ist als der Einfluss der Eingabefehler.

In obigem Beispiel:

$$|\varepsilon_{\text{Eingabe}}| \lessapprox 2\left|\frac{a^2+b^2}{a^2-b^2}\right| \cdot \text{eps}$$

$$|\varepsilon_{\text{Rundung}}| \lessapprox \left(\left|\frac{a^2+b^2}{a^2-b^2}\right| + 1\right) \cdot \text{eps}$$

$\Longrightarrow$ Algorithmus ist stabil !

Für ein „vernünftiges" numerisches Verfahren hat man zwei Anforderungen zu stellen :

1.) Das Problem sollte <u>gut konditioniert</u> sein.

2.) Der numerische Algorithmus sollte <u>stabil</u> sein.