# Kapitel 2

# Besonderheiten des Numerischen Rechnens

## 2.1 Zahlendarstellung

Jede reelle Zahl $x \neq 0$ lässt sich folgendermaßen darstellen:

$$x = \pm \left(\alpha_m 10^m + \alpha_{m-1} 10^{m-1} + \alpha_{m-2} 10^{m-2} + \ldots\right)$$

mit $m \in \mathbb{Z}$, $\alpha_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $\alpha_m \neq 0$.

Schreibweise: $x = \pm \alpha_m \ldots \alpha_1 \alpha_0, \alpha_{-1} \alpha_{-2} \ldots$ für $m \geq 0$, $x = \pm 0, 0 \ldots 0 \alpha_m \alpha_{m-1} \alpha_{m-2} \ldots$ für $m < 0$ mit $|m - 1|$ Nullen zwischen Komma und $\alpha_m$.

Diese Darstellung lässt sich nicht nur für die Zahl 10 sondern allgemein für eine beliebige positive ganze Zahl $B \neq 1$ durchführen:

$$x = \pm \left(\alpha_m B^m + \alpha_{m-1} B^{m-1} + \alpha_{m-2} B^{m-2} + \ldots\right)$$

mit $m \in \mathbb{Z}$, $\alpha_i \in \{0, 1, \ldots, B - 1\}$, $\alpha_m \neq 0$.

Man nennt $B$ die Basis des Zahlensystems, für die Zahlen $0, 1, \ldots, B-1$ werden spezielle Symbole, die Ziffern des Zahlensystems, verwendet.

Neben dem Dezimalsystem ($B = 10$, Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) ist im Zusammenhang mit Computern vor allem das Dualsystem ($B = 2$, Ziffern 0, 1) in Verwendung.

Am Computer wird für reelle Zahlen die normalisierte Gleitkommadarstellung verwendet:

$$x = \pm 0, \alpha_1 \alpha_2 \ldots \alpha_t \cdot B^e \tag{2.1}$$

mit $\alpha_i \in \{0, 1, \ldots, B - 1\}$ und $\alpha_1 \neq 0$. Dabei ist $B \in \mathbb{N} - \{1\}$ die Basis des verwendeten Zahlensystems. $m = 0, \alpha_1 \alpha_2 \ldots \alpha_t$ heißt die Mantisse, $t$ die Mantissenlänge. Der Exponent $e$ ist eine ganze Zahl zwischen vorgegebenen Schranken: $U \leq e \leq O$.

**Beispiel:** Für den Datentyp `float` (einfache Genauigkeit) in C wird meistens eine Zahlendarstellung mit $B = 2$, $t = 24$, $U = -125$ und $O = 128$ verwendet. Das Rechnen mit doppelter Genauigkeit (Datentyp `double`) basiert auf der Setzung $B = 2$, $t = 53$, $U = -1021$ und $O = 1024$. 24 (53) Dualstellen entsprechen etwa 7 (15) Dezimalstellen.

Das Überschreiten bzw. Unterschreiten des Exponentenbereiches wird als overflow bzw. underflow bezeichnet. Während manchmal ein underflow nicht angezeigt wird und der Computer 0 oder die kleinste positive oder negative darstellbare Zahl verwendet, führt ein overflow im Allgemeinen zum Programmabbruch. Wir werden im Weiteren das Auftreten von underflows oder overflows nicht berücksichtigen.

Die Zahl 0 und alle Zahlen der Form (2.1) bilden die Menge $\mathcal{M}$ aller Maschinenzahlen. Nachdem $\mathcal{M}$ nur endlich viele Zahlen enthält, kann natürlich nicht jede reelle Zahl am Rechner exakt dargestellt werden. Man muss runden, d.h., jede reelle Zahl $x$ wird durch eine geeignete Maschinenzahl $rd(x)$ approximiert.

Die bestmögliche Approximation $\mathrm{rd}(x)$ einer reellen Zahl $x$ erfüllt die Bedingung

$$|x - \mathrm{rd}(x)| \leq |x - y| \quad \text{für alle Maschinenzahlen } y. \tag{2.2}$$

Diese Forderung lässt sich leicht realisieren: Für

$$x = \pm 0, \alpha_1 \alpha_2 \ldots \alpha_t \alpha_{t+1} \ldots \cdot B^e$$

mit $\alpha_1 \neq 0$ rundet man auf, falls $\alpha_{t+1} \geq B/2$, bzw. rundet man ab, falls $\alpha_{t+1} < B/2$.

**Bezeichnung:** Sei $\bar{x}$ eine Approximation einer reellen Zahl $x$. Dann heißt

$$\Delta x = \bar{x} - x$$

absoluter Fehler und, falls $x \neq 0$,

$$\varepsilon_x = \frac{\bar{x} - x}{x}$$

relativer Fehler. Natürlich gilt:

$$\Delta x = \varepsilon_x x \quad \text{und} \quad \bar{x} = x(1 + \varepsilon_x).$$

Wir werden auch $|\bar{x} - x|$ als absoluten Fehler bzw. $|(\bar{x} - x)/x|$ als relativen Fehler bezeichnen.

Falls für den absoluten Fehler gilt:

$$|\bar{x} - x| \leq \frac{1}{2} B^s,$$

so heißt die Ziffer mit dem Stellenwert $B^s$ gültig. Die gültigen Ziffern ohne die führenden Nullen heißen signifikante Ziffern. Die gültigen Ziffern beschreiben den absoluten Fehler, die Anzahl der signifikanten Ziffern den relativen Fehler.

Der relative Fehler ist im Allgemeinen aussagekräftiger, weil er den Fehler relativ zum exakten Wert misst. Allerdings gibt es Schwierigkeiten in der Gegend von 0.

Es lässt sich leicht zeigen, dass für das Runden nach (2.2) gilt:

$$\frac{|\,\mathrm{rd}(x) - x|}{|x|} \leq \mathrm{eps} \tag{2.3}$$

mit $\mathrm{eps} = \frac{1}{2} B^{1-t}$, der so genannten **Maschinengenauigkeit**, oder anders ausgedrückt:

$$\mathrm{rd}(x) = x(1 + \varepsilon) \quad \text{mit } |\varepsilon| \leq \mathrm{eps}.$$

*Beweis.* Der Einfachheit halber gelte $x > 0$. Sei $x = mB^e$ eine reelle Zahl mit normalisierter Mantisse $m$, also $B^{-1} \leq |m| < 1$, und sei $\mathrm{rd}(x)$ die gerundete normalisierte Gleitkommazahl mit Mantissenlänge $t$. Man sieht sofort, dass gilt:

$$|\,\mathrm{rd}(x) - x| \leq \frac{1}{2} B^{-t} B^e \quad \text{und} \quad |x| \geq B^{-1} B^e.$$

Also folgt durch Division:

$$\frac{|\,\mathrm{rd}(x) - x|}{|x|} \leq \frac{1}{2} B^{-t} B^e B^1 B^{-e} = \frac{1}{2} B^{1-t} = \mathrm{eps}.$$

□

Der relative Fehler, der beim Runden entsteht, ist also durch die Maschinengenauigkeit eps nach oben beschränkt.

**Beispiel:** Für das Rechnen mit einfacher Genauigkeit und der obigen Form des Rundens erhält man $\mathrm{eps} = 2^{-24} \approx 10^{-7}$, mit doppelter Genauigkeit $\mathrm{eps} = 2^{-53} \approx 10^{-16}$.

Eine andere Möglichkeit des Rundens ist das Abschneiden der Mantisse nach $t$ Ziffern, d.h., man rundet immer ab. In diesem Fall gilt die Abschätzung (2.3) für die Maschinengenauigkeit $\mathrm{eps} = B^{1-t}$.

## 2.2 Gleitkommaarithmetik

Die Addition zweier Maschinenzahlen $x$, $y$ muss nicht wieder eine Maschinenzahl ergeben. Man fordert nun, dass die auf der Maschine verfügbare „Addition“, die so genannte Gleitkommaaddition, die im Weiteren mit dem Symbol $+^*$ bezeichnet wird, so genau ist, dass gilt:

$$x +^* y = \mathrm{rd}(x + y).$$

Diese Forderung lässt sich leicht konstruktiv realisieren: Gegeben seien zwei Maschinenzahlen $x = m_1 \cdot 10^{e_1}$ und $y = m_2 \cdot 10^{e_2}$. Der Einfachheit halber sei angenommen, dass $x \geq y > 0$. (Die anderen Fälle lassen sich analog behandeln.) Die Addition lässt sich in 4 Teilschritten durchführen:

1. Exponentenangleich: $d = e_1 - e_2$
2. Mantissenverschiebung: $m_2' = m_2 \cdot 10^{-d}$, falls $d \leq t$ bzw. $m_2' = 0$ sonst.
3. Mantissenaddition: $m = m_1 + m_2'$
4. Runden

Die Genauigkeit dieser Gleitkommaaddition lässt sich folgendermaßen abschätzen (für $x + y \neq 0$):

$$\frac{|(x +^* y) - (x + y)|}{|x + y|} = \frac{|\operatorname{rd}(x + y) - (x + y)|}{|x + y|} \leq \text{eps},$$

oder anders ausgedrückt:

$$x +^* y = (x + y)(1 + \varepsilon) \quad \text{mit } |\varepsilon| \leq \text{eps}.$$

Der relative Fehler der Gleitkommaaddition ist also durch eps beschränkt.

Die Gleitkommaaddition erfüllt nicht alle Rechengesetze der „normalen“ Addition. So ist z.B. das Assoziativgesetz im Allgemeinen nicht mehr erfüllt:

$$x +^* (y +^* z) \neq (x +^* y) +^* z,$$

d.h., die Reihenfolge der Summanden beeinflusst das Resultat.

Auch alle anderen arithmetischen Operationen (Subtraktion, Multiplikation und Division) und weitere einfache binäre Operationen sind am Computer durch entsprechende Gleitkommaoperationen realisiert. Sei $\circ$ eine dieser Operationen und bezeichne $\circ^*$ die dazugehörige Gleitkommaoperation. Dann fordern wir:

$$x \circ^* y = \operatorname{rd}(x \circ y),$$

woraus wie oben für die Genauigkeit folgt:

$$\frac{|(x \circ^* y) - (x \circ y)|}{|x \circ y|} \leq \text{eps}. \tag{2.4}$$

Am Computer stehen auch einfache Funktionen $f(x)$ wie $\sin x$, $\sqrt{x}$, $\log x$, ..., zur Verfügung. Es wird im Weiteren davon ausgegangen, dass die Realisierungen $f^*(x)$ dieser Funktionen die Abschätzung

$$\frac{|f^*(x) - f(x)|}{|f(x)|} \leq \text{eps} \tag{2.5}$$

erfüllen.

Man nennt die am Rechner realisierten Operationen mit den Abschätzungen (2.4), (2.5) **elementare Operationen**.

**Bemerkung:** Die in diesem und in dem vorigen Abschnitt gemachten Annahmen über die Genauigkeit der Zahlendarstellung und der am Rechner implementierten Operationen sind eine Idealisierung der realen Situation, die allerdings für die Zwecke einer Fehleranalyse zumindest größenordnungsmäßig zutreffende Aussagen erlauben.

## 2.3 Rechengeschwindigkeit

Eine Gleitkommaoperation (z.B.: $z = x + y$, aber auch eine zusammengesetzte Operation, z.B. $z = a * x + y$) wird als ein FLOP (floating point operation) bezeichnet. Für numerische Zwecke wird die Rechengeschwindigkeit durch die Anzahl der FLOPs, die eine Rechenanlage pro Sekunde durchführt, angegeben (kurz: FLOPS, floating point operations per second). Gebräuchlichere Einheiten sind 1 MFLOPS (MegaFLOPS) = $10^6$ FLOPS und 1 GFLOPS (GigaFLOPS) = 1000 MFLOPS.

Die Angabe der Anzahl der MFLOPS charakterisiert die Rechengeschwindigkeit eines skalaren Rechners sehr gut. PCs erreichen FLOP-Raten der Größenordnung $10^2$ MFLOPS bis 1 GFLOPS.

Auf einem Vektorrechner werden gewisse Operationenfolgen (vektorisierbare Operationen) wesentlich schneller durchgeführt.

**Beispiel:** (Pipeline-Prinzip bei der Addition) Unter der vereinfachenden Annahme, dass die Addition in 4 Segmente aufgeteilt wird (siehe vorher) und der Rechner für die Durchführung jedes Segments die gleiche Zeiteinheit (einen Takt) benötigt, dauert eine Addition 4 Zeiteinheiten. Verlässt ein Operandenpaar das erste Segment, kann ein zweites Operandenpaar bereits in das erste Segment nachrücken, u.s.w. Für eine Schleife

```
for (i = 1; i <= n; i++)
  z(i) = x(i) + y(i);
```

gilt daher: Die erste Addition benötigt 4 Zeiteinheiten, dann wird in jeder weiteren Zeiteinheit eine weitere Addition fertig.

Die Beschleunigung wird allerdings nur dann voll wirksam, wenn es keine Datenabhängigkeiten der einzelnen Operationen gibt.

Man unterscheidet auf einem Vektorrechner zwischen der skalaren Leistung und der Spitzenleistung für vektorisierbare Operationen vom obigen Typ. Vektorrechner erreichen eine Spitzenleistung in der Größenordnung von GFLOPS. Je nach Anteil der vektorisierbaren Operationen (Vektorisierungsgrad) liegt die tatsächliche Rechengeschwindigkeit zwischen diesen Werten.

Ein Parallelrechner besteht aus mehreren Prozessoren. Zur Bewertung der Leistung eines Parallelrechners kommt es auf die Anzahl der Prozessoren, die Leistung der einzelnen Prozessoren und auf die Kommunikationsleistung zwischen den Prozessoren an. Die tatsächliche Rechengeschwindigkeit ergibt sich dann vor allem aus dem Anteil der parallel ausführbaren Programmteile (Parallelisierungsgrad), dem Kommunikationsaufwand und der Synchronität des Rechenablaufs.

## 2.4 Fehleranalyse

Man unterscheidet grob drei Fehlerarten:

- Verfahrensfehler,
- Datenfehler und
- Rundungsfehler.

Eine Behandlung von Verfahrensfehlern ist nur für jedes Verfahren individuell möglich und erfolgt daher in den einzelnen Kapiteln. Zur Illustration wird hier nur ein einfaches Beispiel eines Verfahrensfehlers diskutiert:

**Beispiel:** Wir betrachten das Problem der Berechnung der ersten Abnleitung einer Funktion $f$ an einer Stelle $x$, also das Problem

$$y = f'(x).$$

Mit Hilfe eines zentralen Differenzenquotienten lässt sich diese Ableitung näherungsweise berechnen:

$$y_h = \frac{1}{2h}[f(x+h) - f(x-h)].$$

Damit ergibt sich ein Verfahrensfehler

$$\Delta y^V = y_h - y = \frac{1}{2h}[f(x+h) - f(x-h)] - f'(x),$$

der sich mit Hilfe einer Taylor-Reihe für kleine Schrittweiten $h$ leicht abschätzen lässt:

$$\begin{aligned}
\Delta y^V = y_h - y &= \frac{1}{2h}[f(x+h) - f(x-h)] - f'(x) \\
&= \frac{1}{2h}\left[f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f'''(x)h^3 + O(h^4)\right. \\
&\qquad \left. -f(x) + f'(x)h - \frac{1}{2}f''(x)h^2 + \frac{1}{6}f'''(x)h^3 + O(h^4)\right] - f'(x) \\
&= \frac{1}{6}f'''(x)h^2 + O(h^3).
\end{aligned}$$

Im Folgenden werden die Auswirkungen von Daten- und Rundungsfehlern (Fehlerfortpflanzung) näher diskutiert.

### 2.4.1 Datenfehleranalyse

Eine mathematische Problemstellung lässt sich (zumindest formal) in folgende Form bringen: Die gesuchten Größen $y \in \mathbb{R}^m$ sollen aus gegebenen Größen (den Daten) $x \in \mathbb{R}^n$ über eine gegebene Vorschrift $\varphi\colon \mathbb{R}^n \to \mathbb{R}^m$ berechnet werden, kurz:

$$y = \varphi(x).$$

Wir untersuchen nun, wie sich eventuell vorhandene Störungen in den Daten auf die gesuchten Größen auswirken: Sei $x$ der Vektor der exakten Daten und $\bar{x} = x + \Delta x$ ein Vektor von verfälschten Daten. Entsprechend bezeichnet $y = \varphi(x)$ das exakte Ergebnis und $\bar{y} = y + \Delta y = \varphi(\bar{x})$ das Ergebnis der verfälschten Daten.

Wir nehmen im Weiteren an, dass $\varphi$ stetig differenzierbar ist. Dann gilt nach dem Satz von Taylor für den absoluten Fehler:

$$\Delta y_i = \varphi_i(x + \Delta x) - \varphi_i(x) \approx \sum_{j=1}^{n} \frac{\partial \varphi_i}{\partial x_j}(x)\, \Delta x_j \tag{2.6}$$

bzw. für den relativen Fehler, falls $y_i \neq 0$:

$$\varepsilon_{y_i} = \frac{\Delta y_i}{y_i} \approx \sum_{j=1}^{n} \frac{x_j}{\varphi_i(x)} \frac{\partial \varphi_i}{\partial x_j}(x)\, \varepsilon_{x_i}. \tag{2.7}$$

Die Verstärkungsfaktoren $k_{ij} = (x_j/\varphi_i(x))\partial\varphi_i/\partial x_j(x)$ heißen die Konditionszahlen des Problems.

Diese Formeln belegen zwei wichtige Erkenntnisse:

- Die Auswirkung der Störung einer einzelnen Komponente der Daten lässt sich (näherungsweise) durch Multiplikation mit der entsprechenden Ableitung bzw. der dazugehörigen Konditionszahl beschreiben.
- Der Effekt der Störung mehrerer Daten ist näherungsweise gleich der Summe der Effekte der einzelnen Störungen.

Ein Problem heißt **gut konditioniert** (oder auch: **datenstabil**), wenn kleine Fehler der Daten nur kleine Störungen der Lösung bewirken. Andernfalls heißt das Problem **schlecht konditioniert** (**dateninstabil**).

Nach der Fehlerformel (2.7) ist ein Problem dann gut (bzw. schlecht) konditioniert, wenn die Konditionszahlen des Problems klein (bzw. groß) im Vergleich mit 1 sind.

**Bezeichnung:** Die Fehlerformel (2.6) lässt sich in Matrix-Vektor-Schreibweise folgendermaßen kurz darstellen:

$$\Delta y \approx \varphi'(x)\, \Delta x$$

mit

$$\Delta y = \begin{pmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_m \end{pmatrix},\ \Delta x = \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{pmatrix},\ \varphi'(x) = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x_1}(x) & \frac{\partial \varphi_1}{\partial x_2}(x) & \cdots & \frac{\partial \varphi_1}{\partial x_n}(x) \\ \frac{\partial \varphi_2}{\partial x_1}(x) & \frac{\partial \varphi_2}{\partial x_2}(x) & \cdots & \frac{\partial \varphi_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varphi_m}{\partial x_1}(x) & \frac{\partial \varphi_m}{\partial x_2}(x) & \cdots & \frac{\partial \varphi_m}{\partial x_n}(x) \end{pmatrix}.$$

**Beispiel:** Für die Addition zweier Zahlen:

$$y = \varphi(x_1, x_2) = x_1 + x_2$$

folgt nach Formel (2.7):

$$\varepsilon_{x_1+x_2} = \frac{x_1}{x_1 + x_2}\,\varepsilon_{x_1} + \frac{x_2}{x_1 + x_2}\,\varepsilon_{x_2}.$$

Die Konditionszahlen sind besonders groß, wenn gilt: $x_1 \approx -x_2$. Diesen Effekt der großen Verstärkung von Datenfehlern bei der Addition zweier Zahlen mit $x_1 \approx -x_2$ nennt man Auslöschung.

Analog gilt für die Subtraktion:

$$\varepsilon_{x_1-x_2} = \frac{x_1}{x_1 - x_2}\,\varepsilon_{x_1} - \frac{x_2}{x_1 - x_2}\,\varepsilon_{x_2}.$$

Der kritische Fall der Auslöschung tritt hier für $x_1 \approx x_2$ ein.

In diesen Spezialfällen sind die Fehlerformeln sogar exakt. Es entsteht kein Fehler bei der Verwendung der Taylor-Formel, da höhere als erste Ableitungen von $\varphi$ verschwinden.

**Beispiel:** Für die Multiplikation zweier Zahlen

$$y = \varphi(x_1, x_2) = x_1 \cdot x_2$$

folgt nach Formel (2.7):

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

Die Konditionszahlen sind hier 1, das Problem ist gut konditioniert.

Analog gilt für die Division:

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

Die Division zweier Zahlen ist ebenfalls ein gut konditioniertes Problem.

**Beispiel:** Wir betrachten das Problem der Berechnung einer Näherung der ersten Ableitung $f'(x)$ einer Funktion $f$ an einer Stelle $x$ mit Hilfe des zentralen Differenzenquotienten:

$$y_h = \frac{1}{2h}[f(x + h) - f(x - h)]$$

Fehler in $h$ können vermieden werden und werden nicht weiter diskutiert.

- Zunächst werden nur die Auswirkungen von Fehlern in $x$ für eine fixe Funktion $f$ im interessanten Fall $h \ll x$ diskutiert:

$$y_h = \varphi(x) \equiv \frac{1}{2h}[f(x + h) - f(x - h)].$$

Somit folgt für den absoluten Datenfehler

$$\Delta y_h \approx \varphi'(x)\Delta x = \frac{1}{2h}[f'(x+h) - f'(x-h)]\Delta x \approx f''(x)\Delta x.$$

und für den relativen Datenfehler

$$\varepsilon_{y_h} \approx \frac{x\, f''(x)}{f'(x)}\, \varepsilon_x.$$

Das Problem ist also für den Fall $h \ll x$ bezüglich Störungen in $x$ sicherlich gut konditioniert, falls $f'(x)$ von Null verschieden ist und $|x\, f''(x)|$ nicht wesentlich größer als $|f'(x)|$ ist.

- Wenn davon ausgegangen werden muss, dass es auch zu Fehlern bei der Auswertung der Funktion $f$ kommt, ist auch die Funktion $f$ zu den Daten zu zählen und die Auswirkungen solcher Datenstörungen sind zu diskutieren.

  Angenommen, der relative Fehler bei der Berechnung von $f(z)$ überschreitet eine Schranke $\varepsilon_f$ nicht, d.h.:

  $$|\Delta f(z)| \leq |f(z)|\, \varepsilon_f.$$

  Will man nur die Auswirkungen von Störungen bei der Berechnung von $f_+ = f(x+h)$ und $f_- = f(x-h)$ diskutieren, lautet das Problem

  $$y_h = \varphi(f_+, f_-) = \frac{1}{2h}(f_+ - f_-)$$

  und man erhält für den dadurch verursachten absoluten Datenfehler:

  $$\Delta y_h = \frac{\partial \varphi(f_+, f_-)}{\partial f_+}\Delta f_+ + \frac{\partial \varphi(f_+, f_-)}{\partial f_-}\Delta f_- = \frac{1}{2h}[\Delta f_+ - \Delta f_-],$$

  also

  $$|\Delta y_h| \leq \frac{1}{2h}\,[|\Delta f_+| + |\Delta f_-|] \leq \frac{1}{2h}(|f(x+h)| + |f(x-h)|)\, \varepsilon_f \approx \frac{|f(x)|}{h}\, \varepsilon_f,$$

  und für den relativen Datenfehler:

  $$|\varepsilon_{y_h}| \leq \frac{|f(x+h)| + |f(x-h)|}{|f(x+h) - f(x-h)|}\, \varepsilon_f \approx \frac{|f(x)|}{|f'(x)|h}\, \varepsilon_f.$$

  Das Problem ist also für den Fall kleiner Schrittweiten $h$ bezüglich Störungen in $f$ im Allgemeinen schlecht konditioniert.