Prof. Dr. Torsten Linß

Modul 61511

Numerische Mathematik I

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Kapitel 1

Fehleranalyse und Computerzahlen

Sich zu Beginn eines Mathematikkurses lang und breit über Fehler auszulassen, mag etwas absurd erscheinen, wo doch gerade in der Mathematik auf logische Strenge und Exaktheit größter Wert gelegt wird. Es geht vielmehr darum, im Grunde außermathematische Einflüsse in ihrer Wirkung auf Resultate mathematischer Rechnungen zu analysieren und zu bewerten. Um ein konkretes Problem aus der Praxis zu lösen, beschreibt man es mit dem Formalismus der Mathematik, formt es so um, dass es algorithmisch zugänglich ist und bestimmt schließlich eine genäherte Lösung mit Hilfe verfügbarer Rechentechnik. In allen drei Stufen können Fehler auftreten, die einzugrenzen sind, um entscheiden zu können, ob das erhaltene Resultat einen akzeptablen Ersatz für das eigentlich gesuchte Ergebnis darstellt. Eine kleine Geschichte, die von Prof. Franz Locher überliefert wurde, mag die Problematik erläutern.

Der berühmte Gallier Obelix wollte in der Nähe seines Dorfes eine Hinkelstein-Allee errichten. Die Steinreihe sollte 60 Steine umfassen und genau geradlinig verlaufen. Je zwei Steine sollten einen Abstand von 30 Obelix-Längen haben. Obelix hatte sich einen Stock geschnitten, der genau so lang war wie er selbst und den er als Maßstab benutzte. Zum Peilen, ob die Steine in einer Geraden stehen, verließ er sich auf sein scharfes Auge.

Alles war gut gegangen, aber beim 57. Stein gab es Probleme. Denn gleich nach dem 56. Stein war das Ufer eines kleinen, ungefähr 15 Obelix-Längen breiten Teiches. Wo genau sollte der 57. Stein aufgestellt werden?

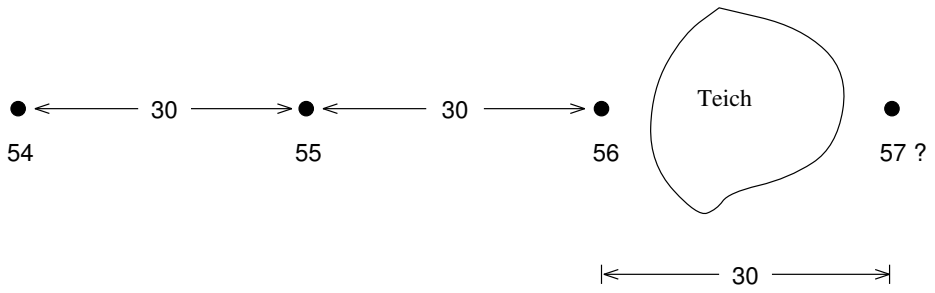

Abbildung 1.1: Lage des 57. Steines

Obelix überlegte hin und her. Auch ein leckerer Wildschweinbraten half ihm nicht auf die richtige Spur. Schließlich holte er sich Rat bei dem Druiden Numerix aus dem Nachbardorf Agen, einem etwas introvertierten Vetter seines Freundes Asterix, der bei so kniffligen Problemen oft ganz gute Ideen hatte.

Abbildung 1.2: Obelix und Numerix

Kein Problem für Numerix! Er malte eine Skizze in den Sand und erklärte Obelix, dass er zwei Hilfshinkelsteine *A* und *B* genau mit den angegebenen Abständen plazieren solle. Dann könne er von *A* und *B* aus auch den 57. Stein genau positionieren. Wichtig seien die Abstände von 30 und 42 Obelix-Längen, was immer die Zahl 42 bedeuten mochte.

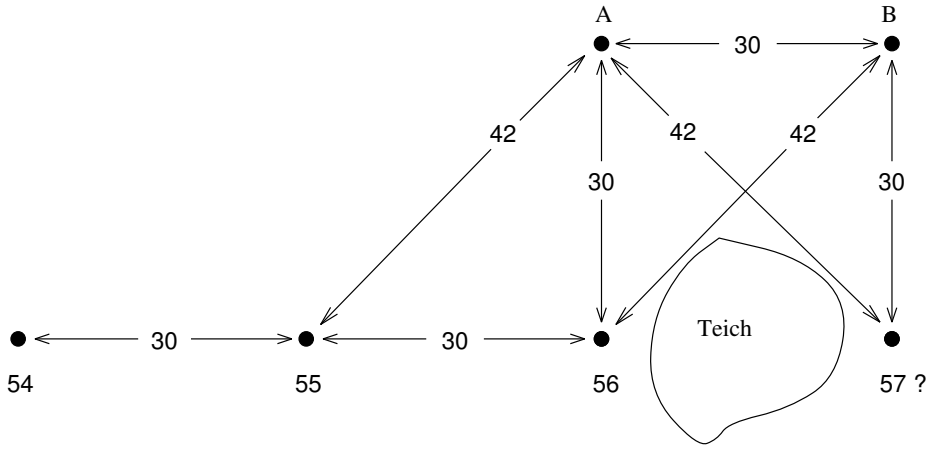

Abbildung 1.3: Konstruktion mit Hilfe von Hilfshinkelsteinen

Obelix ging ans Werk und plagte sich einen ganzen Tag (Numerix hatte vergessen, ihm zu raten, ein langes Seil zu verwenden!), bis er schließlich den 57. Stein aufgestellt hatte, der aber leider nicht in einer Fluchtlinie mit den anderen stand. Am nächsten Morgen zog er nochmals Numerix

zu Rate. Der meinte zwar, ein bißchen mogeln müsse man am Schluss vielleicht schon, aber nicht so viel, wie es hier anscheinend nötig sei. Irgend etwas war faul im Staate Gallien. Numerix maß alles nach und stellte dann auch fest, daß der Abstand von A nach B nur 28 Obelixlängen betrug. (Obelix waren beim Abstecken zwei Wildschweine über den Weg gelaufen; er hatte seine Arbeit kurz unterbrochen und in der Aufregung die Wildschweine mitgezählt.) Als Numerix dies berichtigt hatte, stand auch der 57. Stein ganz ordentlich in der Reihe. Obelix freute sich; aber Numerix dachte, man hätte halt doch 42 und eine halbe Länge statt nur 42 nehmen sollen!

Ihnen ist sicher schnell klar, warum auch beim zweiten Mal der 57. Stein nicht genau in der Fluchtlinie stand – offensichtlich eine Folge von Fehlern und von Fehlerakkumulation. Bei den Fehlern, mit denen Obelix zu kämpften hatte, können wir verschiedene Typen unterscheiden:

- menschlicher Irrtum,
- Modellfehler (der geometrischen Konstruktion liegt der Satz von Pythagoras[1] zu Grunde, der natürlich nur in vollkommen ebenem Gelände anwendbar ist),
- Messfehler (z.B. bei der Peilung auf Geradlinigkeit),
- Rundungsfehler (1.4 statt $\sqrt{2}$).

1.1 Absoluter und relativer Fehler

Diese beiden Begriffe sind wesentlich in der Fehlerrechnung und ihrem Wesen nach bekannt. Trotzdem möchten wir sie hier nochmals formal definieren.

Definition 1.1.1. *Seien $\alpha, \tilde{\alpha} \in \mathbb{C}$, wobei wir $\tilde{\alpha}$ als Näherung der Größe α interpretieren. Wir nennen*

$$\Delta_\alpha := \tilde{\alpha} - \alpha$$

*den **absoluten Fehler** von α und*

$$\varepsilon_\alpha := \frac{\Delta_\alpha}{|\alpha|} = \frac{\tilde{\alpha} - \alpha}{|\alpha|}, \quad \alpha \neq 0,$$

*den **relativen Fehler** von α.*

Bemerkung 1.1.2. Beachten Sie, dass die Fehler vorzeichenbehaftet sind. Oft wird man sich lediglich für eine Abschätzung des Betrags der Fehler interessieren.

Bemerkung 1.1.3. Absolute Fehler sind von geringer Aussagekraft. Wird, z.B., für einen Marathonlauf die Strecke um 1mm falsch vermessen, hat dies keinen Einfluss auf den Rennausgang. Bei einer 4mm-Schraube sieht dies anders aus. Abweichungen von 1mm machen sie unbrauchbar. Von größerer Beutung sind daher relative Fehler, da sie die Abweichung in Relation zur Größe setzen.

[1] Pythagoras von Samos, *≈569 v. Chr., †≈ 475 v. Chr.

Bei Rechnung mit fehlerbehafteten Größen ist ein Ergebnis zu erwarten, das ebenfalls mit einem Fehler behaftet ist. Wir wollen uns dies an einem einfachen Beispiel veranschaulichen, der Berechnung des Volumens einer Pyramide mit Grundfläche $F > 0$ und Höhe $h > 0$.

Da sowohl F als auch h fehlerbehaftet sein können, ist auch für das Volumen

$$V = \frac{1}{3} F \cdot h$$

ein Fehler zu erwarten. Seien also $\tilde{F} = F + \Delta_F$ und $\tilde{h} = \Delta_h + h$ die fehlerbehaftete Grundfläche bzw. Höhe. Das mit diesen Näherungen ermittelte Volumen der Pyramide sei

$$\tilde{V} = \frac{1}{3} \tilde{F} \cdot \tilde{h}.$$

Für den absoluten Fehler des Volumens gilt (Übungsaufgabe!)

$$\Delta_V = \tilde{V} - V = \frac{1}{3}\Big(F\Delta_h + h\Delta_F + \Delta_F\Delta_h\Big). \tag{1.1}$$

Sind die Fehler Δ_F und Δ_h klein, dann ist ihr Produkt $\Delta_F\Delta_h$ sehr klein und kann vernachlässigt werden:

$$\Delta_V \doteqdot \frac{1}{3}\Big(F\Delta_h + h\Delta_F\Big). \tag{1.2}$$

Im Rahmen der Fehlerrechnung sprechen wir von **Gleichheit in linearen Näherung** (oder **erster Näherung**), wenn Produkte und höhere Potenzen (größer als 1) der Fehler vernachlässigt werden. Statt des Gleichheitszeichens verwenden wir das Symbol „$\doteqdot$“.

Zur Bestimmung des relativen Fehlers des Volumens dividieren wir (1.2) durch $V = \frac{1}{3} F \cdot h > 0$ und erhalten für den relativen Fehler des Volumens in linearer Näherung

$$\varepsilon_V = \frac{\Delta_V}{V} \doteqdot \frac{\frac{1}{3}\left(F\Delta_h + h\Delta_F\right)}{\frac{1}{3} F \cdot h} = \frac{\Delta_h}{h} + \frac{\Delta_F}{F} = \varepsilon_F + \varepsilon_h.$$

Wir stellen fest, dass sich bei der Berechnung des Volumens der Pyramide die relativen Fehler (in linearer Näherung) addieren.

Aufgabe 1.1.4. Verifizieren Sie (1.1). ◊

1.2 Maschinenarithmetik

Die Beschränktheit der Resourcen eines Computers, insbesondere des Speicherplatzes, bedingt, dass nicht alle reellen Zahlen auf einem Computer zur Verfügung stehen, sondern lediglich eine endliche Teilmenge. Dieser Abschnitt führt ein in das Rechnen mit Computerzahlen.

1.2.1 Gleitkommazahlen

Für technisch-wissenschaftliche Anwendungen – so auch in der numerischen Mathematik – haben sich **Gleitkommazahlen** (engl.: floating-point numbers) durchgesetzt. Sie werden durch

- *Vorzeichen*, *Mantisse*, *Basis* und *Exponent*

beschrieben. Z.B. entspricht dem Dezimalbruch -5432.1 die Gleitkomma-Darstellung

$$\begin{array}{ccccccc} - & & 0.54321 & \times & 10^4 & & \\ \uparrow & & \uparrow & & \uparrow & \nwarrow & \text{Exponent} \\ \text{Vorzeichen} & & \text{Mantisse} & & \text{Basis} & & e = 4 \\ \sigma = -1 & & & & b = 10 & & \end{array}$$

Die Eingabe und Ausgabe von Zahlen erfolgt in der Regel im Dezimalsystem, da es uns am geläufigsten ist. Zur eigentlichen Rechnung müssen diese Zahlen jedoch in das interne Zahlensystem des Computers umgewandelt werden, das i.Allg. auf dem Dualsystem basiert.

Im Dualsystem mit Basis $b = 2$ wird dabei jede Zahl $x \in \mathbb{R}$ nach Zweierpotenzen entwickelt:

$$x = \pm\left(\alpha_k 2^k + \alpha_{k-1} 2^{k-1} + \cdots\right), \quad \alpha_k \neq 0, \quad \alpha_j \in \{0, 1\}, \quad j = k, k-1, \ldots$$

Dem entspricht die Darstellung als Dualbruch

$$x = \pm[\alpha_k \alpha_{k-1} \alpha_{k-2} \cdots \alpha_0 \;.\; \alpha_{-1} \alpha_{-2} \cdots]_2,$$
$$\uparrow$$
$$\text{Komma, Dualpunkt}$$

wobei ein tiefgestelltes b in der Darstellung $[\ldots]_b$ die zugrundeliegende Basis angibt.

Definition 1.2.1. *Seien $b \in \mathbb{N} \setminus \{1\}$, $m \in \mathbb{N}$, $e_*, e^* \in \mathbb{Z}$, $e_* < e^*$. Die Elemente der Menge*

$$\mathbb{M}_b(m, [e_*, e^*]) := \{0\} \cup \left\{ x \in \mathbb{R} : x = \sigma b^e \sum_{i=1}^{m} z_i b^{-i}, \quad \sigma \in \{-1, 1\}, \right.$$
$$\left. z_i \in \{0, 1, \ldots, b-1\}, \quad z_1 \neq 0, \quad e \in \mathbb{Z}, \quad e_* \leq e \leq e^* \right\}$$

*heißen die **(normalisierten) Gleitkommazahlen** mit **Mantissenlänge** $m \in \mathbb{N}$, **Basis** $b \in \mathbb{N} \setminus \{1\}$ und **Exponentenbereich** $[e_*, e^*]$. In dieser halbexponentiellen Darstellung wird das **Vorzeichen** der Gleitkommazahl durch σ repräsentiert, die z_i sind die **Ziffern** und e der **Exponent**.*

Die endliche Menge $\mathbb{M}_b(m, [e_*, e^*])$ normalisierter Gleitkommazahlen ist durch die Parameter b, m sowie das Intervall $[e_*, e^*]$ festgelegt. Man kann sich überlegen, dass $\mathbb{M}_b(m, [e_*, e^*])$ exakt

$$2(b-1)b^{m-1}(e^* - e_* + 1) + 1$$

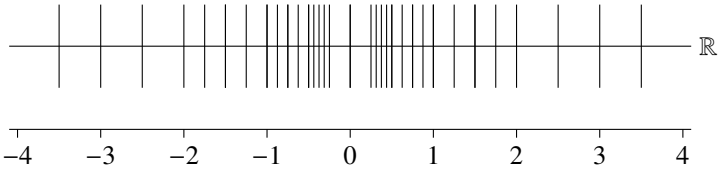

Abbildung 1.4: Normalisierte Gleitkommazahlen; vertikale Linien repräsentieren die Elemente von $\mathbb{M}_2(3,[-1,2])$.

Zahlen umfasst, die auf der reellen Zahlenachse aber nicht gleichmäßig verteilt sind, was Abb. 1.4 an einem einfachen Beispiel verdeutlicht.

Die kleinste positive Zahl in $\mathbb{M}_b(m,[e_*,e^*])$ ist

$$z_{\min} := \min\Big\{y : y \in \mathbb{M}_b(m,[e_*,e^*]),\ \ y>0\Big\} = [0.10\cdots0]_b \cdot b^{e_*} = b^{e_*-1},$$

die größte ist

$$z_{\max} := \max\Big\{y : y \in \mathbb{M}_b(m,[e_*,e^*])\Big\} = [0.\underbrace{(b-1)(b-1)\cdots(b-1)}_{m-\text{mal Ziffer } b-1}]_b \cdot b^{e^*} \approx b^{e^*}.$$

Bei Betrachtung von Abb. 1.4 fallen die Lücken zwischen Null und $\pm z_{\min}$ auf – die sog. „Lücken um die Null“. Diese können durch Übergang zu den **denormalisierten Gleitkommazahlen** $\mathring{\mathbb{M}}_b(m,[e_*,e^*])$ aufgefüllt werden, bei denen als erste Ziffer der Mantisse auch die Null zugelassen wird, siehe Abb. 1.5.

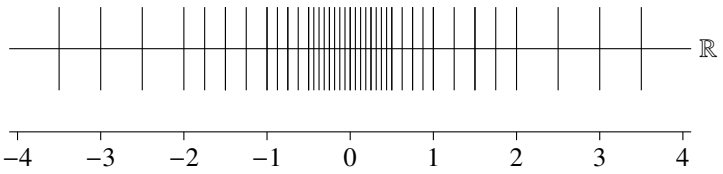

Abbildung 1.5: Denormalisierte Gleitkommazahlen $\mathring{\mathbb{M}}_2(3,[-1,2])$.

Aufgabe 1.2.2. Bestimmen Sie alle Elemente von $\mathbb{M}_2\,(3,[-1,2])$! ◊

1.2.2 Rundung

Eine gegebene reelle Zahl z lässt sich i.Allg. nicht durch eine Gleitkommazahl darstellen. Zahlen, die betragsmäßig größer als $z_{\max}$ sind, lassen sich nicht sinnvoll durch eine Gleitkommazahl repräsentieren und werden computerintern als „NaN“ (engl.: not-a-number) gekennzeichnet. Man spricht auch davon, dass ein **Gleitkommaüberlauf** (engl.: overflow) stattgefunden hat.

Ist $|z| \leq z_{\max}$, so muss $z \in \mathbb{R} \setminus \mathring{\mathbb{M}}_b(m, [e_*, e^*])$ auf eine Maschinenzahl **gerundet** werden. Formal wird z durch eine nächstgelegene Gleitkommazahl $\mathrm{rd}(z)$ approximiert. Die Abbildung

$$\mathrm{rd} : [-z_{\max}, z_{\max}] \to \mathring{\mathbb{M}}_b(m, [e_*, e^*]) : z \mapsto \mathrm{rd}(z)$$

mit

$$|\mathrm{rd}(z) - z| = \min_{y \in \mathring{\mathbb{M}}_b(m,[e_*,e^*])} |y - z| \,. \tag{1.3}$$

wird als **Rundung** bezeichnet. In der Regel ist $\mathrm{rd}(z)$ eindeutig bestimmt. Liegt allerdings z exakt in der Mitte zweier benachbarter Gleitkommazahlen, dann gibt es zwei potentielle Kandidaten für $\mathrm{rd}(z)$. Für diesen Fall sind in den entsprechenden Standards Festlegungen getroffen, die Eindeutigkeit schaffen. Ist $\mathrm{rd}(z) = 0$ für ein $z \neq 0$, so spricht man von **Gleitkommaunterlauf**.

Realisiert wird die Rundung auf zwei Wegen:

- prozessorintern, falls z das Ergebnis einer Rechenoperation mit Maschinenzahlen ist,
- durch Programmbibliotheken, falls z eine Programmeingabe ist, z.B. durch Eingabe als Zeichenkette.

Für ein gegebenes $z \in \mathbb{R}$ ist eine obere Schranke für den Betrag des **absoluten Rundungsfehlers** durch (1.3) gegeben:

$$|\mathrm{rd}(z) - z| = \min_{y \in \mathring{\mathbb{M}}_b(m,[e_*,e^*])} |y - z| \,.$$

Wie eingangs diskutiert (Bem. 1.1.3) sind absolute Fehler von geringer Aussagekraft. Von größerer Bedeutung sind relative Fehler. Für eine reelle Zahl $z \neq 0$ ist der **relative Rundungsfehler** bei Repräsentation durch Maschinenzahlen gegeben durch

$$\frac{\mathrm{rd}(z) - z}{|z|},$$

vgl. Def. 1.1.1. Ihn wollen wir im Folgenden genauer untersuchen.

Sei $z \in \mathbb{R}$ mit $|z| \in [z_{\min}, z_{\max}]$. Dann existiert ein eindeutig bestimmtes $e \in [e_*, e^*]$ mit $|z| \in [b^{e-1}, b^e)$. Die normalisierten Gleitkommazahlen im Intervall $[b^{e-1}, b^e]$ sind

$$[0.10\cdots00]_b \cdot b^e, [0.10\cdots01]_b \cdot b^e, \ldots, [0.(b-1)\cdots(b-1)]_b \cdot b^e, [1.0\cdots0]_b \cdot b^e.$$

Man überlegt sich, dass auf dem Intervall $[b^{e-1}, b^e]$ der Abstand zwischen zwei benachbarten Gleitkommazahlen b^{e-m} ist. Für den Betrag des **absoluten Rundungsfehler** folgt somit zunächst

$$|\mathrm{rd}(z) - z| \leq \frac{1}{2} b^{e-m}$$

und für den **relativen Rundungsfehler**

$$\frac{|\mathrm{rd}(z) - z|}{|z|} \leq \frac{1}{2}\frac{b^{e-m}}{b^{e-1}} = \frac{1}{2}b^{1-m} =: \texttt{eps}, \tag{1.4}$$

da $|z| \geq b^{e-1}$.

Die Größe `eps`, welche bei gewählter Basis nur von der Länge m der Mantisse abhängt, wird als **Maschinengenauigkeit** bezeichnet. Sie ist die wichtigste Größe, die Gleitkommaarithmetiken charakterisiert.

Wir formalisieren dieses letzte Ergebnis und schaffen uns zugleich ein Modell zur mathematischen Fassung der Gleitkommazahlen.

Satz 1.2.3. *Die Abbildung* $\mathrm{rd} : [-z_{max}, z_{max}] \to \mathring{\mathbb{M}}_b(m, [e_*, e^*])$ *erfülle* (1.3). *Dann existiert für jedes* $z \in \mathbb{R}$, $z_{min} \leq |z| \leq z_{max}$, *ein* $\varepsilon \in \mathbb{R}$ *mit*

$$\mathrm{rd}(z) = (1 + \varepsilon)\, z \quad \textit{und} \quad |\varepsilon| \leq \texttt{eps} = \frac{1}{2}b^{1-m}.$$

Bemerkung 1.2.4. Ist $z \in (-z_{\min}, z_{\min})$ – also wenn denormalisierte Gleitkommazahlen zum Einsatz kommen – dann gilt die Abschätzung (1.4) i.Allg. nicht. Für $z \to 0$ konvergiert der relative Rundungsfehler gegen Eins.

In der Praxis haben sich die IEEE[2]-Standards mit unterschiedlichen Genauigkeiten durchgesetzt:

- einfache Genauigkeit: $\mathring{\mathbb{M}}_2(24, [-126, 127])$,
- doppelte Genauigkeit: $\mathring{\mathbb{M}}_2(53, [-1022, 1023])$ und
- erweiterte doppelte Genauigkeit: $\mathring{\mathbb{M}}_2(64, [-16382, 16383])$.

Die für sie charakteristischen Parameter sind in der folgenden Tabelle aufgelistet. Sie enthält außerdem die Namen der entsprechenden C-Datentypen sowie die Anzahl der zur Speicherung einer Gleitkommazahl des jeweiligen Typs benötigten Bytes. Die Wahl der Exponentenbereiche bedingt, dass $z_{\max} \approx 1/z_{\min}$, weshalb $z_{\max}$ nicht angegeben ist.

b	m	e_*	e^*	`eps`	$z_{\min}$	C, C++	Bytes
2	24	−126	127	$5.960 \cdot 10^{-8}$	$5.878 \cdot 10^{-39}$	`float`	4
2	53	−1022	1023	$1.110 \cdot 10^{-16}$	$1.112 \cdot 10^{-308}$	`double`	8
2	64	−16382	16383	$5.421 \cdot 10^{-20}$	$1.681 \cdot 10^{-4932}$	`long double`	12

Tabelle 1.1: Parameter der Maschinenzahlen des IEEE-Standards

[2]Institute of Electrical and Electronics Engineers, `https://www.ieee.org/`

Bemerkung 1.2.5. Für höhere Genauigkeiten sind Programmbibliotheken entwickelt worden, z.B. die *GNU Multiple Precision Arithmetic Library*[3] oder *ARPREC*[4]. Allerdings hat höhere Genauigkeit ihren Preis, nämlich längere Rechenzeiten und höheren Speicherbedarf.

Bemerkung 1.2.6. Für die Kodierung des Vorzeichens wird ein Bit benötigt. Bei Verwendung der Basis 2 genügen für die Mantisse normalisierter Gleitkommazahlen $m-1$ Bits, da die erste Mantissenstelle stets Eins ist. Für den Exponenten werden 8, 11 bzw. 16 Bits (bei einfacher, doppelter bzw. erweiterter doppelter Genauigkeit) benötigt. Bei genauerer Betrachtung fällt auf, dass statt der möglichen 256, 2048 bzw. 32768 Exponenten lediglich 254, 2046 bzw. 32766 verwendet werden. Die verbleibenden zwei möglichen Exponenten werden genutzt, um anzuzeigen, dass

- ein Überlauf stattgefunden hat, also keine sinnvolle Gleitkommazahl vorliegt, oder
- die kodierte Zahl eine denormalisierte Gleitkommazahl ist. In diesem Fall ist der Exponent $e = e_*$ und die erste Mantissenstelle ist Null.

Bemerkung 1.2.7. Die vier Grundrechenarten mit IEEE-konformen *normalisierten* Gleitkommazahlen werden hardwareseitig durch alle aktuellen Prozessortypen unterstützt. Rechenoperationen mit *denormalisierten* Zahlen werden mittels Programmbibliotheken bewerkstelligt und sind i.Allg. wesentlich langsamer. Die Nutzung denormalisierter Zahlen kann durch Compileroptionen beeinflusst werden.

1.2.3 Grundoperationen und elementare Funktionen

Wir wenden uns jetzt den **arithmetischen Grundoperationen** zu. Diese liefern als Resultate i.a. selbst dann keine Gleitkommazahlen, wenn die Operanden Gleitkommazahlen sind. Daher sind maschinen-interne Rundungen erforderlich. Als Resultat erhält man Näherungen

$$x \oplus y, \quad x \ominus y, \quad x \otimes y \quad \text{und} \quad x \oslash y$$

für die arithmetischen Grundoperationen

$$x + y, \quad x - y, \quad x \times y \quad \text{und} \quad x/y.$$

Die Näherungen werden in zwei Schritten berechnet:

S1. Aus den zwei Operanden wird zunächst prozessorintern ein Zwischenresultat mit höherer Genauigkeit durch Rechnung mit längerer Mantisse gebildet.

S2. Anschließend wird dieses Ergebnis auf die normale Stellenzahl gerundet und auf den entsprechenden Speicherplatz geschrieben.

[3] `http://gmplib.org/`
[4] `http://crd-legacy.lbl.gov/~dhbailey/mpdist/`

Oder kurz

$$x \odot y := \mathrm{rd}(x \cdot y),$$

wobei „$\cdot$“ und „$\odot$“ für eine beliebige der vier Grundoperationen bzw. ihre maschinenseitige Realisierung stehen.

Die so implementierten arithmetischen Grundoperationen besitzen folgende Eigenschaft: Für alle $x, y \in \mathbb{M}_b(m, [e_*, e^*])$ mit $|x \cdot y| \in [z_{\min}, z_{\max}]$ existiert ein $\varepsilon \in \mathbb{R}$ derart, dass

$$x \odot y = (1 + \varepsilon)\,(x \cdot y) \quad \text{mit} \quad |\varepsilon| \leq \texttt{eps}.$$

Elementare Funktionen

$$f \in \left\{\sin, \cos, \tan, \sqrt{\ }, \exp, \ln, \ldots\right\}$$

werden mittels Programmbibliotheken ausgewertet. Ist $f_\circ$ die programmtechnische Implementierung von f, dann gilt

$$f_\circ(x) = (1 + \varepsilon)\, f(x) \quad \text{für alle} \quad x \in \Omega \subset D_f, \quad \text{mit} \quad |\varepsilon| \leq K_f \texttt{eps}, \tag{1.5}$$

mit einer moderaten Konstante K_f, z.B. $K_f \in [1, 10]$. Für welchen Teilbereich Ω der Gleitkommazahlen Abschätzungen dieses Typs tatsächlich gelten, und wie groß K_f ist, sollte dem Compiler-Handbuch zu entnehmen sein.

Aufgabe 1.2.8. Betrachtet werde die Menge der Gleitkommazahlen $\mathbb{M}_2(4, [-7, 7])$. Bestimmen Sie für jede der folgenden Operationen mit den Computerzahlen $x = [0.1011]_2 \cdot 2^0$ und $y = [0.1100]_2 \cdot 2^0$, ob sie exakt ausgeführt werden oder ob es zu Rundungen, Überlauf oder Unterlauf kommt:

$$z = x \ominus y, \quad z = (y \ominus x)^{10}, \quad z = x \oplus y, \quad z = y \oplus (x \oslash 4) \quad \text{und} \quad z = x \oplus (y \oslash 4).$$

Bemerkung: Zur Vereinfachung erfolge die Rundung nach jedem Rechenschritt durch einfaches Abschneiden nach der 4. Mantissenstelle. ◊

Aufgabe 1.2.9. Das Distributivgesetz

$$(a + b)c = ac + bc$$

gilt in der Menge der Gleitkommazahlen i.Allg. nicht. Diskutieren Sie, inwieweit das Distributivgesetz verletzt wird!

Gegeben seien dazu eine Gleitkommaarithmetik mit Maschinengenauigkeit eps sowie Gleitkommazahlen a, b und c. Wir setzen

$$y_1 = (a \oplus b) \otimes c \quad \text{und} \quad y_2 = (a \otimes c) \oplus (b \otimes c).$$

Die relativen Fehler ε_k, $k = 1, 2$, der berechneten Ergebnisse sind durch $y_k = (a + b)c(1 + \varepsilon_k)$, $k = 1, 2$, gegeben. Bestimmen Sie die relativen Fehler beider Berechnungsvorschriften in linearer Näherung! Wann ist eine wesentlich genauer als die andere? ◊

1.3 Rundungsfehlereinfluss und Fehlerfortpflanzung

Wir beginnen diesen Abschnitt mit einem Beispiel.

1.3.1 Numerische Differentiation

Eine der grundlegenden Aufgaben der numerischen Mathematik ist die näherungweise Berechnung der Ableitung einer Funktion, die zwar nicht explizit bekannt, aber z.B. mittels eines Algorithmus auswertbar ist. Eine ähnliche Aufgabenstellung ist die Bestimmung der Beschleunigung eines Körpers aus einer gegebenen Messreihe seiner Geschwindigkeit.

Definitionsgemäß gilt für eine Funktion $f \in C^1(a,b)$

$$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}, \quad x \in (a,b).$$

Eine naheliegende Idee ist nun, für ein hinreichend kleines $h > 0$ den Quotienten

$$(\delta_h f)(x) := \frac{f(x+h)-f(x)}{h} \tag{1.6}$$

als eine Approximation für $f'(x)$ zu verwenden. Die Sinnhaftigkeit dieses Zugangs wird durch folgenden Satz abgesichert, der eine Aussage über den auftretenden **Approximationsfehler** trifft.

Satz 1.3.1. *Seien $f \in C^2[a,b]$, $a < b$, und $M := \max_{\xi\in[a,b]} |f''(\xi)|$. Dann gilt*

$$\left|(\delta_h f)(x) - f'(x)\right| \le \frac{M}{2}\,|h|$$

für alle $x \in [a,b]$ und $h \neq 0$ mit $x + h \in [a,b]$.

Beweis. Seien x und $x+h$ wie in den Voraussetzungen des Satzes. Dem Satz von Taylor[5] zufolge existiert ein ξ zwischen x und $x + h$ mit

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\xi).$$

Wir dividieren durch $h \neq 0$ und bilden die Beträge:

$$\left|(\delta_h f)(x) - f'(x)\right| = \frac{|h|}{2}\,|f''(\xi)|\,.$$

Es folgt die Behauptung des Satzes, da $|f''(\xi)| \le M$. □

[5] Brook Taylor *18.8.1685 (Edmonton, Middlesex, England), †29.12.1731 (Somerset House, London, England)

Der Fehler bei der näherungsweisen Berechnung der ersten Ableitung (einer zweimal stetig differenzierbaren Funktion) gemäß (1.6) ist (im schlechtesten Fall) proportional zu h. Wir wollen dies an einem konkreten Beispiel illustrieren. Unser Vorgehen dabei ist typisch für die numerische Mathematik: ein Verfahren/Algorithmus wird an einer Testaufgabe erprobt, dessen Lösung bekannt ist.

Zu berechnen sei die Ableitung der Exponentialfunktion $\exp : x \mapsto \mathrm{e}^x$ an der Stelle $x = 1$. Die Ableitung der Exponentialfunction ist natürlich bekannt:

$$\exp'(1) = \exp(1) \approx 2.718281828459045.$$

Wir approximieren $\exp'(1)$ durch $(\delta_h \exp)(1)$ mit $h = 10^{-1}, 10^{-2}, \ldots, 10^{-15}$. Zu diesem Zweck haben wir ein kleines Programm geschrieben, das uns auf einem handelsüblichen Rechner ausgeführt Tabelle 1.2 liefert.

h	$(\delta_h \exp)(1)$	Fehler
10^{-1}	2.858841954873883	$1.406 \cdot 10^{-1}$
10^{-2}	2.731918655787124	$1.364 \cdot 10^{-2}$
10^{-3}	2.719641422533225	$1.360 \cdot 10^{-3}$
10^{-4}	2.718417747082924	$1.359 \cdot 10^{-4}$
10^{-5}	2.718295419956717	$1.359 \cdot 10^{-5}$
10^{-6}	2.718283187430615	$1.359 \cdot 10^{-6}$
10^{-7}	2.718281968405734	$1.399 \cdot 10^{-7}$
10^{-8}	2.718281821856294	$6.603 \cdot 10^{-9}$
10^{-9}	2.718282043900899	$2.154 \cdot 10^{-7}$
10^{-10}	2.718283376168529	$1.548 \cdot 10^{-6}$
10^{-11}	2.718314462413218	$3.263 \cdot 10^{-5}$
10^{-12}	2.718714142702083	$4.323 \cdot 10^{-4}$
10^{-13}	2.717825964282383	$4.559 \cdot 10^{-4}$
10^{-14}	2.708944180085382	$9.338 \cdot 10^{-3}$
10^{-15}	3.108624468950438	$3.903 \cdot 10^{-1}$

Tabelle 1.2: Approximation der Ableitung von exp durch δ_h; korrekte Stellen unterstrichen

Zunächst beobachten wir, dass im Einklang mit Satz 1.3.1 für $h = 10^{-1}, \ldots, 10^{-8}$ der Fehler proportional zu h ist, und die Güte der Approximation mit kleiner werdendem h zunimmt. Bei weiterer Verkleinerung von h ist jedoch eine signifikante Verschlechterung zu beobachten. Auf den ersten Blick widerspricht dies Satz 1.3.1.

Dieser Widerspruch ist aber nur ein scheinbarer, denn der Beweis von Satz 1.3.1 geht davon aus, dass mit reellen Zahlen gerechnet wird. Unser Programm arbeitet aber mit Gleitkommazahlen! Dadurch ist die Berechnung von $(\delta_h f)(x)$ mit Rundungsfehlern behaftet:

$$f'(x) \approx (\delta_h f)(x) \approx \Big(f_\circ(\mathrm{rd}(x) \oplus \mathrm{rd}(h)) \ominus f_\circ(\mathrm{rd}(x))\Big) \oslash \mathrm{rd}(h).$$

Rundungsfehler treten auf

- bei der Repräsentation von x und h durch Gleitkommazahlen,
- bei der Addition von x und h,
- bei der Auswertung von f
- bei der Subtraktion im Zähler sowie
- bei der Division durch h.

Exemplarisch wollen wir lediglich den Einfluss der fehlerbehafteten Auswertung von f analysieren. Vereinfachend nehmen wir daher an, dass x, h und $x + h$ exakt durch Gleitkommazahlen repräsentiert werden, und auch die Division durch h exakt ausgeführt wird. (Die hierbei auftretenden Fehler können in der Tat vernachlässigt werden.) Mit diesen vereinbarten Annahmen gilt

$$f'(x) - \frac{f_\circ(x+h) - f_\circ(x)}{h} = f'(x) - \frac{(1+\varepsilon_1)f(x+h) - (1+\varepsilon_2)f(x)}{h}$$

mit relativen Fehlern bei der Auswertung von f entsprechend (1.5):

$$|\varepsilon_1|, |\varepsilon_2| \leq K_f \, \texttt{eps}\,. \tag{1.7}$$

Den Zähler des zweiten Bruches auf der rechten Seite multiplizieren wir aus und erhalten für den Gesamtfehler:

$$f'(x) - \frac{f_\circ(x+h) - f_\circ(x)}{h} = \underbrace{f'(x) - (\delta_h f)(x)}_{\text{Approximationsfehler}} + \underbrace{\frac{\varepsilon_1 f(x+h) - \varepsilon_2 f(x)}{h}}_{\text{Rundungsfehler}},$$

Den Approximationsfehler schätzen wir mit Hilfe von Satz 1.3.1 ab, für den Rundungsfehler verwenden wir (1.7):

$$\left| f'(x) - \frac{f_\circ(x+h) - f_\circ(x)}{h} \right| \leq \frac{M|h|}{2} + \frac{2\texttt{eps}}{|h|} K_f \max_{x\in[a,b]} |f(x)|\,. \tag{1.8}$$

Diese detailliertere Untersuchung zeigt den Einfluss der Rundungsfehler. Für kleiner werdendes h wird zwar der Approximationsfehler kleiner, aber die Rundungsfehler aus der Arbeit mit Gleitkommazahlen werden verstärkt. Abbildung 1.6 illustriert die Situation.

Eine gute Wahl von h zeichnet sich dadurch aus, dass Approximations- und Rundungsfehler auf der rechten Seite von (1.8) die gleiche Größenordnung besitzen. Das ist der Fall, wenn

$$h \approx \frac{\texttt{eps}}{h},$$

woraus sich $h^* \approx \sqrt{\texttt{eps}}$ ergibt. Durch Einsetzen von h^* in (1.8) erhalten wir für den Gesamtfehler

$$\left| f'(x) - \frac{f_\circ(x+h^*) - f_\circ(x)}{h^*} \right| \leq M_f \sqrt{\texttt{eps}}$$

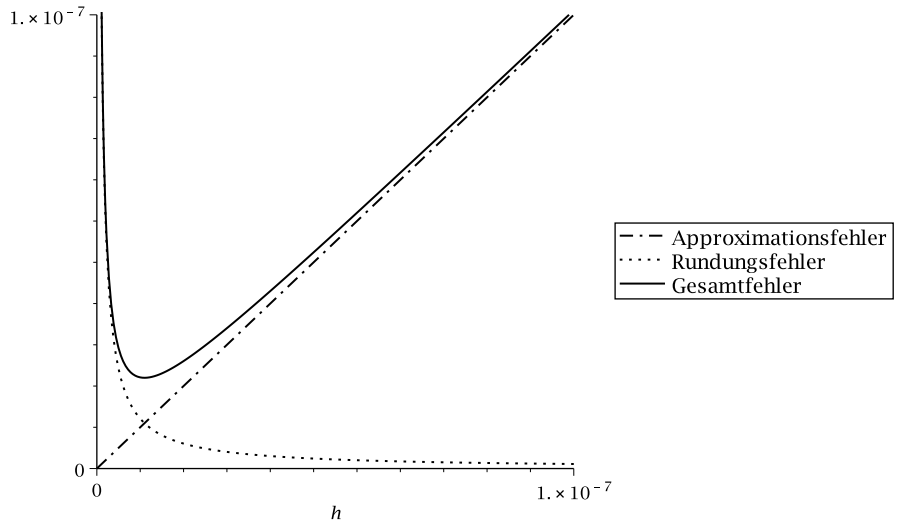

Abbildung 1.6: Fehlerkomponenten bei der numerischen Differentiation

mit einer (i. Allg. unbekannten) Konstante M_f, die von f, f'' und möglicherweise auch von verwendeten Programmbibliotheken abhängt.

In obigem Beispiel hat unser Computer mit doppelter Genauigkeit gerechnet. Für diese ist `eps` $\approx 1.110 \cdot 10^{-16}$. Eine gute Wahl ist dementsprechend $h^* \approx 10^{-8}$. Für diese Wert von h haben wir in Tabelle 1.2 auch die genaueste Approximation der Ableitung erhalten. Der Gesamtfehler liegt bei $\sqrt{\texttt{eps}} \approx 10^{-8}$.

Das Ergebnis ist etwas ernüchternd. Bei der näherungsweisen Berechnung der Ableitung mittels (1.6) sind lediglich die ersten 8 Dezimalstellen korrekt, obwohl unser Computer mit 16 Dezimalstellen arbeitet.

Aufgabe 1.3.2. Untersuchen Sie den symmetrischen Differenzenquotienten

$$f'(x) \approx (\delta_h^* f)(x) := \frac{f(x+h) - f(x-h)}{2h}, \quad h > 0.$$

Zeigen Sie: Ist $f \in C^3[a,b]$, $a < b$, und $x \pm h \in (a,b)$, dann gilt für den Approximationsfehler

$$\left|(\delta_h^* f)(x) - f'(x)\right| \leq Mh^2$$

mit einer Konstante $M \geq 0$. Bestimmen Sie M.

Analysieren Sie den Rundungsfehlereinfluss bei Arbeit mit Maschinengenauigkeit `eps`. Für welches h ist eine hohe Approximationsgüte zu erwarten? Wie groß ist der zu erwartende Fehler? ◊

1.3.2 Kondition von Aufgaben

Gegeben sei die **Aufgabe**, aus einem Satz von **Eingangsdaten** $x_1, \ldots, x_n \in \mathbb{R}$ ein **Resultat** $z \in \mathbb{R}$ entsprechend der Vorschrift

$$z = f(x_1, \ldots, x_n)$$

zu berechnen. Wir interessieren uns dafür, den Einfluss von Fehlern in den Eingangsdaten auf das Resultat abzuschätzen. Störungen der Eingangsdaten resultieren z.B. aus der Verwendung von Gleitkommazahlen oder von mit Messfehlern behafteten Daten.

Mit dieser Fragestellung hatten wir uns an einem einfachen Beispiel bereits im Abschnitt 1.1 beschäftigt, wo das Volumen einer Pyramide zu berechnen war. Wir wollen das Vorgehen jetzt verallgemeinern und formalisieren.

Bemerkung 1.3.3. An dieser Stellen müssen wir klar trennen zwischen der *Aufgabe* und einer konkreten *Berechnungsvorschrift* zu ihrer Lösung auf einem Computer. Durch die Arbeit mit Gleitkommazahlen werden im Laufe der Rechnung Rundungsfehler eingeführt, die bei einer schlechten Berechnungsvorschrift das Resultat ungünstig verfälschen können, wie wir es z.B. schon in Aufgabe 1.2.9 beobachtet haben.

Dem Einfluss von Rundungsfehlern bei der Berechnung von f und der Analyse von Berechnungsvorschriften/Algorithmen werden wir uns in §1.3.3 widmen.

Um uns das Studium der Fehlerfortpflanzung zu erleichtern, nehmen wir an, dass die Funktion $f : \mathbb{R}^n \to \mathbb{R}$ in einer hinreichend großen Umgebung von $(x_1, \ldots, x_n)$ zweimal stetig differenzierbar ist. Dies gestattet uns eine Taylorentwicklung bis zum quadratischen Glied.

Die betrachteten Daten fassen wir zu dem Datenvektor

$$\boldsymbol{x} := (x_1, \ldots, x_n)^T,$$

zusammen. Die mit Fehlern behafteten Daten seien

$$\tilde{\boldsymbol{x}} := (\tilde{x}_1, \ldots, \tilde{x}_n)^T.$$

Definitionsgemäß ist der absolute Fehler

$$\boldsymbol{\Delta_x} := (\Delta_{x_1}, \ldots, \Delta_{x_n})^T = \tilde{\boldsymbol{x}} - \boldsymbol{x}.$$

Die relativen Fehler der einzelnen Komponenten sind

$$\varepsilon_{x_i} := \frac{\Delta_{x_i}}{|x_i|}, \quad i = 1, \ldots, n.$$

Uns interessiert der Einfluss von $\boldsymbol{\Delta_x}$ auf den Fehler $\Delta_z := \tilde{z} - z$ im Resultat. Dabei ist $z = f(\boldsymbol{x})$ der exakte Funktionswert, während sich $\tilde{z} = f(\tilde{\boldsymbol{x}})$ aus der Berechnung mit dem gestörten Argument ergibt.

Wir definieren die Hilfsfunktionen

$$\boldsymbol{\xi} : [0, 1] \to \mathbb{R}^n : t \mapsto \boldsymbol{\xi}(t) := \boldsymbol{x} + t\boldsymbol{\Delta_x} \tag{1.9a}$$

und

$$\varphi : [0, 1] \to \mathbb{R} : t \mapsto \varphi(t) := f(\boldsymbol{\xi}(t)). \tag{1.9b}$$

Diese ist so definiert, dass $\varphi(0) = f(\boldsymbol{x})$ und $\varphi(1) = f(\tilde{\boldsymbol{x}})$, was die Fehlerdarstellung

$$\Delta_z = \tilde{z} - z = f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x}) = \varphi(1) - \varphi(0)$$

gestattet. Dem Satz von Taylor zufolge existiert ein $\tau \in (0, 1)$ mit

$$\varphi(1) = \varphi(0) + \varphi'(0) \cdot (1 - 0) + \frac{(1-0)^2}{2}\varphi''(\tau).$$

Damit folgt

$$\Delta_z = \varphi'(0) + \frac{1}{2}\varphi''(\tau). \tag{1.10}$$

Wir müssen nun die Ableitungen von φ auf jene von f zurückführen. Dazu lösen wir die folgende Übungsaufgabe.

Aufgabe 1.3.4. Beweisen Sie: Für die gemäß (1.9) definierte Funktion φ gilt

$$\varphi'(t) = \nabla f(\boldsymbol{\xi}(t))^T \boldsymbol{\Delta_x} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\boldsymbol{\xi}(t))\Delta_{x_i}$$

und

$$\varphi''(t) = \boldsymbol{\Delta_x}^T \boldsymbol{H}_f(\boldsymbol{\xi}(t))\, \boldsymbol{\Delta_x} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\xi}(t))\Delta_{x_i}\Delta_{x_j}\,,$$

dabei sind ∇f und $\boldsymbol{H}_f$ der Gradient bzw. die Hesse[6]-Matrix von f. ◊

Wir wenden die Ergebnisse der Übungsaufgabe auf (1.10) an, und erhalten für den Fehler von z

$$\Delta_z = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\boldsymbol{x})\,\Delta_{x_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\xi}(\tau))\,\Delta_{x_i}\Delta_{x_j}\,, \quad \tau \in (0,1).$$

Wie im Abschnitt 1.1 nehmen wir an, dass die Fehler Δ_{x_i} klein sind und somit Produkte $\Delta_{x_i}\Delta_{x_j}$ vernachlässigt werden können. Für den Fehler von z erhalten wir also in linearer Näherung

$$\Delta_z \doteq \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\boldsymbol{x})\Delta_{x_i}. \tag{1.11}$$

Die Zahl $\sigma_i := \frac{\partial f}{\partial x_i}(\boldsymbol{x})$ gibt in erster Näherung an, um welchen Faktor verstärkt (oder gedämpft) der *absolute* Fehler Δ_{x_i} der i-ten Komponente des Datenvektors in den *absoluten* Fehler Δ_z des Resultates eingeht.

[6]Ludwig Otto Hesse, *22.4.1811 (Königsberg, Preussen) †4.8.1874 (München, Deutschland)

Von größerer Bedeutung für uns ist – insbesondere mit Blick auf die Gleitkommazahlen, die Repräsentanten reeller Zahlen mit gewissen *relativen* Genauigkeiten sind – der Zusammenhang zwischen *relativen* Fehlern in den Eingangsdaten und im Ergebnis.

Wir dividieren (1.11) durch $z = f(\boldsymbol{x})$:

$$\varepsilon_z = \frac{\Delta_z}{z} \doteq \sum_{i=1}^{n} \frac{x_i}{f(\boldsymbol{x})} \frac{\partial f}{\partial x_i}(\boldsymbol{x}) \frac{\Delta x_i}{x_i} = \sum_{i=1}^{n} \frac{x_i}{f(\boldsymbol{x})} \frac{\partial f}{\partial x_i}(\boldsymbol{x})\, \varepsilon_{x_i}.$$

Der relative Fehler in x_i geht also um den Faktor

$$\frac{x_i}{f(\boldsymbol{x})} \frac{\partial f}{\partial x_i}(\boldsymbol{x})$$

verstärkt in den relativen Fehler des Ergebnisses ein.

Definition 1.3.5. *Gegeben sei die Aufgabe $z = f(\boldsymbol{x})$ für ein $\boldsymbol{x} \in \mathbb{R}^n$ zu berechnen. Die Größe*

$$\kappa_i := \frac{x_i}{f(\boldsymbol{x})} \frac{\partial f}{\partial x_i}(\boldsymbol{x}), \quad i = 1, \ldots, n,$$

*wird als **Konditionszahl der Aufgabe bezüglich der i-ten Komponente des Datenvektors** (oder **bezüglich des i-ten Argumentes**) bezeichnet.*

Man beachte, dass im Vergleich zu den σ_i neben den partiellen Ableitungen von f auch die Faktoren $x_i / f(\boldsymbol{x})$ auftreten. Diese sind besonders ungünstig, wenn ein betragsmäßig kleines Resultat aus betragsmäßig großen Daten zu berechnen ist. Die Größen κ_i sind unabhängig vom Maßstab, in dem Daten und Resultat gemessen werden. Das gibt den Konditionszahlen eine hervorragende Bedeutung beim Arbeiten mit Gleitkommazahlen. Tritt eine betragsmäßig große Konditionszahl κ_i auf, so wirkt sich der relative Fehler ε_i in der i-ten Komponente des Datenvektors sehr ungünstig auf den relativen Fehler des Resultats aus.

Bemerkung 1.3.6. Eine Aufgabe mit (mindestens) einer betragsmäßig großen Konditionszahl wird als **schlecht konditioniert** bezeichnet. Man spricht auch von der **natürlicher Instabilität** der Aufgabe. Sind alle Konditionszahlen moderat, so spricht man hingegen von einer **gut konditionierten** Aufgabe.

Der Übergang von „gut konditioniert“ zu „schlecht konditioniert“ ist fließend. Entscheidend ist, ob in einer bestimmten Anwendung die Verstärkung der relativen Fehler um einen gewissen Faktor akzeptiert werden kann oder nicht.

Wir analysieren exemplarisch die Grundrechenarten.

Beispiel 1.3.7. Multiplikation: Gegeben ist die Aufgabe, das Produkt z zweier reeller Zahlen x_1 und x_2 zu berechnen. Der Zusammenhang zwischen Ergebnis und Eingangsdaten ist gegeben durch

$$z = f(x_1, x_2) = x_1 x_2.$$

Die Konditionszahl κ_1 bezüglich des ersten Argumentes ist

$$\kappa_1 = \frac{x_1}{f(x_1, x_2)} \frac{\partial f}{\partial x_1}(x_1, x_2) = \frac{x_1}{x_1 x_2} x_2 = 1.$$

Völlig analog erhalten wir $\kappa_2 = 1$. Die Multiplikation zweier Zahlen ist gut konditioniert. ♠

Beispiel 1.3.8. Division: Zu berechnen ist

$$z = f(x_1, x_2) = \frac{x_1}{x_2}, \quad x_2 \neq 0.$$

Für die Konditionszahlen erhalten wir

$$\kappa_1 = \frac{x_1}{f(x_1, x_2)} \frac{\partial f}{\partial x_1}(x_1, x_2) = \frac{x_1}{\frac{x_1}{x_2}} \frac{1}{x_2} = 1$$

sowie

$$\kappa_2 = \frac{x_2}{f(x_1, x_2)} \frac{\partial f}{\partial x_2}(x_1, x_2) = \frac{x_2}{\frac{x_1}{x_2}} \frac{-x_1}{x_2^2} = -1.$$

Die Division zweier Zahlen ist ebenfalls gut konditioniert. ♠

Beispiel 1.3.9. Addition, Subtraktion: Zu berechnen ist

$$z = f(x_1, x_2) = x_1 + x_2.$$

Die Konditionszahlen sind

$$\kappa_1 = \frac{x_1}{f(x_1, x_2)} \frac{\partial f}{\partial x_1}(x_1, x_2) = \frac{x_1}{x_1 + x_2} \quad \text{und} \quad \kappa_2 = \frac{x_2}{f(x_1, x_2)} \frac{\partial f}{\partial x_2}(x_1, x_2) = \frac{x_2}{x_1 + x_2}.$$

Diese werden groß, wenn $(x_1 + x_2)$ betragsmäßig wesentlich kleiner als x_1 bzw. x_2 ist. Mit anderen Worten, wird die Differenz zweier etwa gleichgroßer Zahlen berechnet (also $x_1 \approx -x_2$), ist eine signifikante Verstärkung von Rundungsfehlern zu erwarten. ♠

Der u.U. schlechten Kondition der Subtraktion sind wir bereits bei der numerischen Differentiation im Abschnitt 1.3.1 begegnet. Dort wurde gemäß

$$\exp'(1) \approx \frac{\mathrm{e}^{1+h} - \mathrm{e}^1}{h}$$

eine Approximation der ersten Ableitung berechnet. Wir wollen bei 16stelliger Genauigkeit die Berechnung der Differenz im Zähler für $h = 10^{-8}$ detailliert nachvollziehen.

$$\begin{array}{rl} \mathrm{e}^{1+h} = x_1 & 2.718\,281\,855\,641\,863 \\ \mathrm{e}^1 = x_2 & 2.718\,281\,828\,459\,045 \\ \hline \mathrm{e}^{1+h} - \mathrm{e}^1 = z & 0.000\,000\,027\,182\,818 = 2.718\,281\,800\,000\,000 \cdot 10^{-8} \end{array}$$

Wir beobachten bei der Berechnung der Differenz einen „Verlust“ der letzten 8 Mantissenstellen. Dieser Effekt, der bei der Subtraktion zweier annähernd gleicher Zahlen auftritt, wird als **Stellenauslöschung** bezeichnet. Er muss – falls möglich – vermieden werden.

Beispiel 1.3.10. Wurzelfunktion. Zu berechnen sei $z = f(a) = \sqrt{a}$, $a \geq 0$. Die Konditionszahl dieser Aufgabe ist

$$\kappa = \frac{a}{f(a)} f'(a) = \frac{a}{\sqrt{a}} \frac{1}{2\sqrt{a}} = \frac{1}{2}.$$

Die Aufgabe ist gut konditioniert. ♠

Beispiel 1.3.11. Lösung quadratischer Gleichungen. Vorgelegt sei die Aufgabe, bei gegebenen $p, q > 0$, $q \ll p^2$,

$$\lambda = \varphi(p, q) = p - \sqrt{p^2 - q}$$

zu berechnen. Es handelt sich dabei um eine der Lösungen von $\lambda^2 - 2p\lambda + q = 0$. Auf den ersten Blick scheint hier eine kritische Subtraktion aufzutreten, da $\sqrt{p^2 - q} \approx p$, weshalb man Stellenauslöschung erwarten könnte. Allerdings stehen Minuend und Subtrahend in einem Zusammenhang. Sie sind also nicht unabhängig voneinander mit Rundungsfehlern behaftet. Es bedarf daher einer genaueren Untersuchung.

Wir bestimmen zunächst die Konditionszahl der Aufgabe bezüglich p:

$$\kappa_1 = \frac{p}{\varphi(p, q)} \frac{\partial \varphi}{\partial p}(p, q) = \frac{p}{p - \sqrt{p^2 - q}} \left(1 - \frac{p}{\sqrt{p^2 - q}}\right) = \frac{-p}{\sqrt{p^2 - q}} \approx -1,$$

da $\sqrt{p^2 - q} \approx p$ für $q \ll p^2$.

Für die Konditionszahl bezüglich q gilt

$$\kappa_2 = \frac{q}{\varphi(p, q)} \frac{\partial \varphi}{\partial q}(p, q) = \frac{q}{p - \sqrt{p^2 - q}} \frac{1}{2\sqrt{p^2 - q}} = \frac{p + \sqrt{p^2 - q}}{2\sqrt{p^2 - q}},$$

wobei wir bei der letzten Umformung

$$\lambda = \varphi(p, q) = p - \sqrt{p^2 - q} = \frac{q}{p + \sqrt{p^2 - q}} \tag{1.12}$$

verwendet haben (Vieta[7]scher Wurzelsatz). Es folgt

$$\kappa_2 = \frac{p + \sqrt{p^2 - q}}{2\sqrt{p^2 - q}} \approx 1, \quad \text{für } q \ll p^2.$$

Beide Konditionszahlen sind in etwa 1. Die Aufgabe ist somit gut konditioniert. ♠

[7]François Viète (latinisiert Vieta), *1540 (Fontenay-le-Comte, Poitou (jetzt Vendée), Frankreich), †13.12.1603 (Paris, Frankreich)

1.3.3 Kondition von Algorithmen

Die natürliche Stabilität (oder Instabilität) ist eine Eigenschaft der gegebenen Aufgabe. Sie hängt nur von der vorgelegten Aufgabe ab, aber nicht von dem für die Berechnung gewählten Algorithmus.

In diesem Abschnitt werden wir sehen, dass das Ergebnis einer gut konditionierten Aufgabe durch einen ungeschickt gewählten **Algorithmus** (Berechnungsmethode) stark verfälscht werden kann, also ein Verlust an Genauigkeit eintritt. In diesem Fall spricht man von **numerischer Instabilität**. Sie ist eine Eigenschaft des **Algorithmus** und hängt vom gewählten Rechnungsverlauf ab, bei dem **Rundungsfehler** in einzelnen Berechnungsschritten unter Umständen erheblich verstärkt werden können.

Um unser weiteres Vorgehen zu motivieren kommen wir zurück auf Beispiel 1.3.11, in dem für gegebene $p, q > 0$, $q \ll p^2$,

$$\lambda = p - \sqrt{p^2 - q} \tag{1.13}$$

zu berechnen war. Wir setzen (1.13) direkt in einen Algorithmus um und berechnen nacheinander die Zwischenergebnisse

$$y_1 := p \cdot p\,, \quad y_2 := y_1 - q \quad \text{und} \quad y_3 := \sqrt{y_2}$$

sowie das Endergebnis

$$\lambda := p - y_3.$$

Wird hierbei mit Gleitkommazahlen gerechnet, werden alle drei Zwischenergebnisse y_i mit Fehlern behaftet sein. Wir müssen uns der Frage stellen, welchen Einfluss diese Fehler auf das Endergebnis haben. Dazu stellen wir λ in Abhängigkeit von den y_i dar und analysieren anschließend mit unseren Ergebnissen aus Abschnitt 1.3.2 die Fehlerfortpflanzung.

Wir rollen den Algorithmus von hinten auf. Der letzte Berechnungsschritt gibt uns unmittelbar die Abhängigkeit von y_3:

$$\lambda = p - y_3\,. \tag{1.14a}$$

Wir gehen einen Schritt zurück und ersetzen $y_3 = \sqrt{y_2}$:

$$\lambda = p - \sqrt{y_2}\,, \tag{1.14b}$$

was uns die Abhängigkeit von y_2 liefert. Als nächstes substituieren wir $y_2 = y_1 - q$:

$$\lambda = p - \sqrt{y_1 - q}\,. \tag{1.14c}$$

Wir haben jetzt explizite Darstellungen von λ in Abhängigkeit von den Zwischenergebnisse y_i und können nun mit Hilfe unserer Überlegungen aus §1.3.2 die Frage beantworten, wie sich relative Fehler in y_i auf das Resultat λ auswirken. Oder mit anderen Worten:

- Wie ist die Aufgabe, λ gemäß (1.14c) zu berechnen, bezüglich y_1 konditioniert?
- Wie ist die Aufgabe, λ gemäß (1.14b) zu berechnen, bezüglich y_2 konditioniert?
- Wie ist die Aufgabe, λ gemäß (1.14a) zu berechnen, bezüglich y_3 konditioniert?

Wir bestimmen die entsprechenden Konditionszahlen K_i gemäß Definition 1.3.5:

$$\begin{aligned} K_1 &:= \frac{y_1}{\lambda}\frac{\partial\lambda}{\partial y_1} = \frac{y_1}{\lambda}\frac{-1}{2\sqrt{y_1-q}}, \\ K_2 &:= \frac{y_2}{\lambda}\frac{\partial\lambda}{\partial y_2} = \frac{y_2}{\lambda}\frac{-1}{2\sqrt{y_2}}, \\ K_3 &:= \frac{y_3}{\lambda}\frac{\partial\lambda}{\partial y_3} = \frac{y_3}{\lambda}(-1). \end{aligned}$$

Man überlegt sich, dass $\lambda, y_1, y_2, y_3 > 0$. Ferner folgt aus $q \ll p^2$,

$$\lambda = p - \sqrt{p^2-q} = \frac{q}{p+\sqrt{p^2-q}} \approx \frac{q}{2p} \quad \text{sowie} \quad \sqrt{y_1-q} \approx p.$$

Somit ist

$$K_1 \approx -\frac{p^2}{q}, \quad \text{also} \quad |K_1| \approx \frac{p^2}{q} \gg 1.$$

Kleine Fehler in y_1 werden also signifikant verstärkt, was alles andere als wünschenswert ist!

Aber woran liegt das? Gehen wir die Schritte des Algorithmus im Einzelnen durch. Bei der Berechnung von y_1 werden zwei Zahlen multipliziert. Hier geht nichts schief, vgl. Bsp. 1.3.7. Im nächsten Schritt wird eine kleine Zahl von einer großen subtrahiert. Auch dies kann nicht die Ursache sein, siehe Bsp. 1.3.9. Zur Berechnung von y_3 wird eine Wurzel gezogen. Auch dies ist problemlos, wie wir in Bsp. 1.3.10 festgestellt hatten. Es ist der letzte Schritt, bei dem λ als Differenz von p und y_3 berechnet wird. Da $y_3 \approx p$ tritt hier Stellenauslöschung auf.

Formal können wir dies bestätigen, indem wir die Aufgabe, λ gemäß (1.14a) zu berechnen, betrachten und ihre Konditionszahl bzgl. y_3 untersuchen:

$$K_3 \approx -\frac{2p^2}{q}.$$

Sie ist betragsmäßig sehr groß. Also spätestens im letzten Schritt werden alle unsere Bemühungen, ein genaues Ergebnis zu berechnen, zunichte gemacht.

Die Aufgabe ist gut konditioniert, unser erster Algorithmus, den wir uns ausgedacht haben, aber unbrauchbar. Wir müssen also noch an einer Alternative arbeiten. Bevor wir uns dieser Aufgabe zuwenden, wollen wir unserer bisheriges Vorgehen verallgemeinern und formalisieren.

Ein Algorithmus zur Berechnung von $z = f(\boldsymbol{x})$ aus dem Datenvektor $\boldsymbol{x}$ bestehe aus folgenden **Berechnungsschritten**, bei denen Zwischenergebnisse y_i, $i = 1, \ldots, N-1$, berechnet werden.

$$\begin{aligned} y_1 &= r_1(\boldsymbol{x}) \\ y_2 &= r_2(\boldsymbol{x}, y_1) \\ y_3 &= r_3(\boldsymbol{x}, y_1, y_2) \\ &\vdots \\ y_{N-1} &= r_{N-1}(\boldsymbol{x}, y_1, y_2, \ldots, y_{N-2}) \\ z &= r_N(\boldsymbol{x}, y_1, y_2, \ldots, y_{N-1}) \end{aligned}$$

Uns interessiert die Frage, wie sich Rundungsfehler im Zwischenergebnis y_i auf das Resultat z auswirken.

Zu diesem Zweck betrachten wir y_i als Eingabe und stellen uns die Frage, wie die Aufgabe, z aus $\boldsymbol{x}$ sowie $y_1, \ldots, y_i$ zu bestimmen, bezüglich y_i konditioniert ist. Diese Aufgabe wird durch die sog. **Restabbildung**

$$R_i : (\boldsymbol{x}, y_1, \ldots, y_i) \mapsto z$$

beschreiben. Die Restabbildungen werden sukzessive, durch Elimination der Zwischenergebnisse $y_{N-1}, y_{N-2}, \ldots, y_{i+1}$, aus den Berechnungsvorschriften des Algorithmus bestimmt:

$$\begin{aligned} z = R_{N-1}(\boldsymbol{x}, y_1, \ldots, y_{N-1}) &:= r_N(\boldsymbol{x}, y_1, \ldots, y_{N-1}) \\ = R_{N-2}(\boldsymbol{x}, y_1, \ldots, y_{N-2}) &:= R_{N-1}(\boldsymbol{x}, y_1, \ldots, y_{N-2}, r_{N-1}(\boldsymbol{x}, y_1, \ldots, y_{N-2})) \\ &\vdots \\ = R_{N-k}(\boldsymbol{x}, y_1, \ldots, y_{N-k}) &:= R_{N-k+1}(\boldsymbol{x}, y_1, \ldots, y_{N-k}, r_{N-k+1}(\boldsymbol{x}, y_1, \ldots, y_{N-k})), \quad k < N. \end{aligned}$$

Die Verstärkung oder Abschwächung der relativen Fehler im Zwischenergebnis y_i berechnen sich, unseren Ergebnissen aus Abschnitt 1.3.2 folgend, gemäß

$$\frac{y_i}{R_i(\boldsymbol{x}, y_1, \ldots, y_i)} \frac{\partial R_i}{\partial y_i}(\boldsymbol{x}, y_1, \ldots, y_i) = \frac{y_i}{z} \frac{\partial R_i}{\partial y_i}(\boldsymbol{x}, y_1, \ldots, y_i), \quad i = 1, \ldots, N-1.$$

Definition 1.3.12. *Die Größen*

$$K_i := \frac{y_i}{z} \frac{\partial R_i}{\partial y_i}(\boldsymbol{x}, y_1, \ldots, y_i), \quad i = 1, \ldots, N-1,$$

*heißen die **Konditionszahlen des Algorithmus** (bzgl. des Zwischenergebnisses y_i).*

Bemerkung 1.3.13. Der Algorithmus heißt **numerisch stabil**, wenn alle Konditionszahlen des Algorithmus betragsmäßig nicht wesentlich größer sind als die Konditionszahlen der Aufgabe. Ansonsten heißt er **instabil**.

Wie bei der Kondition von Aufgaben ist der Übergang von „stabil“ zu „instabil“, fließend. Entscheidend ist wiederum, in welchem Ausmaß Verstärkungen der relativen Fehler akzeptiert werden können.

Beispiel 1.3.14. Wir kommen zurück auf unser einführendes Beispiel und analysieren es mit dem soeben hergeleiteten Formalismus. Für gegebene $p, q > 0$, $q \ll p^2$, ist also $\lambda = p - \sqrt{p^2 - q}$ zu berechnen.

Der Algorithmus ist:

$$\begin{aligned} y_1 &:= r_1(p,q) &&= p \cdot p, \\ y_2 &:= r_2(p,q,y_1) &&= y_1 - q, \\ y_3 &:= r_3(p,q,y_1,y_2) &&= \sqrt{y_2}, \\ \lambda &:= r_4(p,q,y_1,y_2,y_3) &&= p - y_3. \end{aligned}$$

Offenbar sind $\lambda, y_1, y_2, y_3 > 0$.

Zur Untersuchung der Stabilität bestimmen wir zunächst die Restabbildungen:

$$\begin{aligned} \lambda = R_3(p,q,y_1,y_2,y_3) &= r_4(p,q,y_1,y_2,y_3) &&= p - y_3, \\ = R_2(p,q,y_1,y_2) &= R_3(p,q,y_1,y_2,r_3(p,q,y_1,y_2)) &&= p - \sqrt{y_2}, \\ = R_1(p,q,y_1) &= R_2(p,q,y_1,r_2(p,q,y_1)) &&= p - \sqrt{y_1 - q} \end{aligned}$$

und berechnen anschließend die Konditionszahlen des Algorithmus

$$\begin{aligned} K_1 &= \frac{y_1}{\lambda} \frac{\partial R_1}{\partial y_1}(p,q,y_1) &&= \frac{y_1}{\lambda} \frac{-1}{2\sqrt{y_1 - q}}, \\ K_2 &= \frac{y_2}{\lambda} \frac{\partial R_2}{\partial y_2}(p,q,y_1,y_2) &&= \frac{y_2}{\lambda} \frac{-1}{2\sqrt{y_2}}, \\ K_3 &= \frac{y_3}{\lambda} \frac{\partial R_3}{\partial y_3}(p,q,y_1,y_2,y_3) &&= \frac{y_3}{\lambda} (-1). \end{aligned}$$

In der gegebenen Situation, also $q \ll p^2$, gilt

$$y_2 \approx p^2, \quad y_3 \approx p \quad \text{und} \quad \lambda = p - \sqrt{p^2 - q} = \frac{q}{p + \sqrt{p^2 - q}} \approx \frac{q}{2p}$$

Woraus dann für die Beträge der Konditionszahlen des Algorithmus folgt

$$|K_1|, |K_2| \approx \frac{p^2}{q} \gg 1 \quad \text{und} \quad |K_3| \approx \frac{2p^2}{q} \gg 1.$$

Der Algorithmus ist numerisch instabil und somit unbrauchbar, da eine (hier sogar alle) Konditionszahlen des Algorithmus wesentlich größer als die Konditionszahlen der Aufgabe sind, vgl. Beispiel 1.3.11. Die Ursache liegt – wie bereits diskutiert – in der Subtraktion im letzten Schritt des Verfahrens, bei der Stellenauslöschung auftritt. ♠

Durch Anwendung des Vietasche Wurzelsatzes kann eine alternative Berechungsvorschrift angegeben werden, die numerisch stabil ist. Wir lösen dazu die folgende Übungsaufgabe.

Aufgabe 1.3.15. Ausgehend von (1.12) kann zum Berechnen von

$$\lambda = p - \sqrt{p^2 - q} = \frac{q}{p + \sqrt{p^2 - q}}, \quad p, q > 0, \;\; p^2 \gg q,$$

der folgende Algorithmus verwendet werden:

$$y_1 := p \cdot p, \quad y_2 := y_1 - q, \quad y_3 := \sqrt{y_2}, \quad y_4 := p + y_3, \quad \lambda := q/y_4.$$

Bestimmen Sie die Restabbildungen R_i, $i = 1, \ldots, 4$, und berechnen Sie die zugehörigen Konditionszahlen K_i, $i = 1, \ldots, 4$, des Algorithmus! Entscheiden Sie, ob der Algorithmus numerisch stabil ist! ◊

1.4 Lösungen der Aufgaben

Lösung 1.1.4.

$$\begin{aligned}\Delta_V = \tilde{V} - V &= \frac{1}{3}\tilde{F}\cdot\tilde{h} - \frac{1}{3}F\cdot h = \frac{1}{3}\Big\{(F+\Delta_F)(h+\Delta_h) - F\cdot h\Big\}\\ &= \frac{1}{3}\Big\{Fh + F\Delta_h + h\Delta_F + \Delta_F\Delta_h - Fh\Big\} = \frac{1}{3}\Big\{F\Delta_h + h\Delta_F + \Delta_F\Delta_h\Big\}.\end{aligned}$$

♣

Lösung 1.2.2. Wegen $b = 2$ sind die in der Darstellung möglichen Ziffern gerade 0 und 1. Bei den normalisierte Gleitkommazahlen muss die erste Nachkommastelle der Mantisse von 0 verschieden sein. Zudem hat die Mantisse jeweils 3 Nachkommastellen. In untenstehender Tabelle erhalten wir also die erste Spalte aller vorkommenden Mantissen (1. Spalte). Die erste Nachkommastelle hat dabei den Wert $\frac{1}{2}$, die zweite den Wert $\frac{1}{4}$ und die dritte den Wert $\frac{1}{8}$. Für eine Mantisse $[0.\alpha_1\alpha_2\alpha_3]_3$ errechnet sich so ein Wert von $\frac{\alpha_1}{2} + \frac{\alpha_2}{4} + \frac{\alpha_3}{8}$ (siehe 2. Spalte).

Für die vorkommenden Exponenten $e \in \{-1, 0, 1, 2\}$ müssen alle Mantissen noch mit 2^e multipliziert werden. Diese Ergebnisse stehen in der dritten bis sechsten Spalte.

m	$x = \frac{\alpha_1}{2} + \frac{\alpha_2}{4} + \frac{\alpha_3}{8}$	$x\cdot 2^{-1}$	$x\cdot 2^{0}$	$x\cdot 2^{1}$	$x\cdot 2^{2}$
0.100	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2
0.101	$\frac{1}{2}+\frac{1}{8}=\frac{5}{8}$	$\frac{5}{16}$	$\frac{5}{8}$	$\frac{5}{4}$	$\frac{5}{2}$
0.110	$\frac{1}{2}+\frac{1}{4}=\frac{3}{4}$	$\frac{3}{8}$	$\frac{3}{4}$	$\frac{3}{2}$	3
0.111	$\frac{1}{2}+\frac{1}{4}+\frac{1}{8}=\frac{7}{8}$	$\frac{7}{16}$	$\frac{7}{8}$	$\frac{7}{4}$	$\frac{7}{2}$

Sämtliche in der Tabelle aufgeführten Zahlen kommen zudem mit negativem Vorzeichen vor. Zusätzlich gehört noch die 0 zu $\mathbb{M}_2(3,[-1,2])$. ♣

Lösung 1.2.8. Wir arbeiten in $\mathbb{M}_2(4,[-7,7])$, d.h. mit Basis 2, der Mantissenlänge 4 und dem Exponentenbereich $-7,\ldots,7$. Die beiden Operanden sind

$$x = [0.1011]_2\cdot 2^0 = 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} + 1\cdot 2^{-4}$$

und

$$y = [0.1100]_2\cdot 2^0 = 1\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 0\cdot 2^{-4}$$

$\underline{z = x\ominus y:}$

$$x - y = -[0.1000]_2\cdot 2^{-3} \in \mathbb{M}_2(4,[-7,7]), \quad\Longrightarrow\quad x\ominus y = x - y = -[0.1000]_2\cdot 2^{-3}.$$

Die Rechnung wird also *exakt* ausgeführt.

$\underline{z = (y \ominus x)^{10} :}$

$$y \ominus x = [0.1000]_2 \cdot 2^{-3} \quad \Longrightarrow \quad (y \ominus x)^{10} = [0.1000]_2 \cdot 2^{-39}.$$

Exponent (-39) ist kleiner als -7, unterschreitet also den Exponentenbereich. Das Ergebnis wird auf 0 gerundet: $(y \ominus x)^{10} = 0$. Es findet ein *Unterlauf* statt

$\underline{z = x \oplus y :}$

$$x + y = [1.0111]_2 \cdot 2^0 \notin \mathbb{M}_2(4, [-7, 7]), \quad \Longrightarrow \quad x \oplus y = [0.1011]_2 \cdot 2^1.$$

Das Ergebnis ist durch Abschneiden der 5. Mantissenstelle *gerundet*.

$\underline{z = y \oplus (x \oslash 4) :}$ Zunächst ist $x \oslash 4 = [.1011]_2 \cdot 2^{-2} = x/4$, also

$$y + x \oslash 4 = [.111011]_2 \cdot 2^0 \notin \mathbb{M}_2(4, [-7, 7]), \quad \Longrightarrow \quad y \oplus (x \oslash 4) = [0.1110]_2 \cdot 2^0.$$

Das Ergebnis ist durch Abschneiden der 5. und 6. Mantissenstelle *gerundet*.

$\underline{z = x \oplus (y \oslash 4) :}$ Es ist $(y \oslash 4) = [0.1100]_2 \cdot 2^{-2} = y/4$, also

$$x + y \oslash 4 = [0.1110]_2 \cdot 2^0 \in \mathbb{M}_2(4, [-7, 7]), \quad \Longrightarrow \quad x \oplus (y \oslash 4) = y + x/4 = [0.1110]_2 \cdot 2^0.$$

Das Ergebnis ist *exakt*. ♣

Lösung 1.2.9. Sowohl die Multiplikation als auch die Addition von Maschinenzahlen werden mit relativen Fehlern kleiner als `eps` ausgeführt. D.h. für die beiden Berechnungsvorschriften gilt

$$y_1 = (a \oplus b) \otimes c = (1 + e_1)((a \oplus b)c) = (1 + e_1)(1 + e_2)(a + b)c$$

bzw.

$$\begin{aligned} y_2 &= (a \otimes c) \oplus (b \otimes c) = (1 + e_3)(a \otimes c + b \otimes c) = (1 + e_3)((1 + e_4)ac + (1 + e_5)bc) \\ &= (1 + e_3)\left((1 + e_4)\frac{a}{a + b} + (1 + e_5)\frac{b}{a + b}\right)(a + b)c \end{aligned}$$

mit $|e_i| \leq \texttt{eps}$, $i = 1, \ldots, 5$.

Für die relativen Fehler ε_1 und ε_2 der beiden Ergebnisse erhalten wir

$$\varepsilon_1 = e_1 + e_2 + e_1 e_2 \doteq e_1 + e_2.$$

sowie

$$\varepsilon_2 = e_3 + e_4\frac{a}{a + b} + e_5\frac{b}{a + b} + e_3\left(e_4\frac{a}{a + b} + e_5\frac{b}{a + b}\right) \doteq e_3 + e_4\frac{a}{a + b} + e_5\frac{b}{a + b}\,.$$

Bei Verwendung der ersten Berechnungsvorschrift addieren sich die Rundungsfehler. Die zweite Berechnungsvorschrift ist wesentlich ungünstiger falls $a \approx -b$, da $|a + b|$ sehr klein ist. Dann sind $|a/(a + b)|$ und $|b/(a + b)|$ sehr groß und die Fehler e_4 und e_5 werden signifikant verstärkt.

Ein Beispiel bei Rechnung auf dem Computer mit doppelter Genauigkeit ($\texttt{eps} = 2^{-52} \approx 1.1 \cdot 10^{-16}$) möge dieses Ergebnis verdeutlichen. Wir wählen die drei Gleitkommazahlen $a = 1 - 2^{-52}$, $b = -1$ und $c = 3/2$. Die relativen Fehler sind $\varepsilon_1 = 0$ bzw. $\varepsilon_2 = 1/3 \gg \texttt{eps}$.

Bei der zweiten Berechnungsvorschrift ist der relative Fehler ca. 10^{15} mal größer als die Maschinengenauigkeit. Das Ergebnis wäre also völlig unbrauchbar. ♣

Lösung 1.3.2. Seien $x \in (a, b)$ beliebig und $h > 0$ hinreichend klein, so dass $x \pm h \in [a, b]$. Nach Taylor existieren ein $\xi^- \in [x - h, x]$ und ein $\xi^+ \in [x, x + h]$ mit

$$f(x - h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{6} f'''(\xi^-)$$

und

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(\xi^+).$$

Es folgt

$$(\delta_h^* f)(x) - f'(x) = \frac{h^2}{12}\Big(f'''(\xi^+) + f'''(\xi^-)\Big)$$

Setze $M := \frac{1}{6} \max\limits_{\xi \in [a,b]} |f'''(\xi)|$. Wir erhalten für den Approximationsfehler

$$\left|(\delta_h^* f)(x) - f'(x)\right| \le Mh^2.$$

Zur Analyse des Rundungsfehlereinflusses verfahren wir wie bei der Herleitung von (1.8) :

$$f'(x) - \frac{f_\circ(x + h) - f_\circ(x - h)}{2h} = f'(x) - (\delta_h^* f)(x) + \frac{\varepsilon_1 f(x + h) - \varepsilon_2 f(x - h)}{2h},$$

woraus die Abschätzung

$$\left| f'(x) - \frac{f_\circ(x + h) - f_\circ(x - h)}{2h} \right| \le Mh^2 + \frac{\texttt{eps}}{h} K_f \max_{x \in [a,b]} |f(x)|$$

folgt.

Eine gute Wahl von h ergibt sich wieder, wenn beide Fehlerterme die gleiche Größe besitzen, d.h.

$$h^2 \approx \frac{\texttt{eps}}{h} \quad \Longleftrightarrow \quad h^* = \texttt{eps}^{1/3}.$$

Das zugehörige Fehlerniveau bei der Approximation der ersten Ableitung ist $\texttt{eps}^{2/3}$.

Bei Rechnung mit doppelter Genauigkeit, also $\texttt{eps} \approx 1.1 \cdot 10^{-16}$, ist $h^* \approx 4.8 \cdot 10^{-6}$ mit Fehlerniveau von ungefähr $2.3 \cdot 10^{-11}$.

Insgesamt eine signifikante Verbesserung im Vergleich zur Verwendung der einseitigen Differenz δ_h im Abschnitt 1.3.1. Man beachte aber, dass eine höhere Regularität (Differenzierbarkeit) erforderlich ist. ♣

Lösung 1.3.4. $\boldsymbol{\xi}(t) = \boldsymbol{x} + t\boldsymbol{\Delta_x} \implies \boldsymbol{\xi}'(t) = \boldsymbol{\Delta_x}$, also $\xi_i'(t) = \Delta_{x_i}$, $i = 1, \ldots, n$.
Wir wenden die Kettenregel auf φ an:

$$\begin{aligned}
\varphi(t) &= f(\boldsymbol{\xi}(t)) \\
\varphi'(t) &= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\boldsymbol{\xi}(t))\xi_i'(t) \\
&= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\boldsymbol{\xi}(t))\,\Delta_{x_i} = \nabla f(\boldsymbol{\xi}(t))^T \boldsymbol{\Delta_x} \\
\varphi''(t) &= \sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial f}{\partial x_i}(\boldsymbol{\xi}(t))\,\Delta_{x_i}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_j}\left(\frac{\partial f}{\partial x_i}(\boldsymbol{\xi}(t))\,\Delta_{x_i}\right)\xi_j'(t) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\xi}(t))\,\Delta_{x_i}\Delta_{\xi_j} = \boldsymbol{\Delta_x}^T \boldsymbol{H}_f(\boldsymbol{\xi}(t))\,\boldsymbol{\Delta_x}
\end{aligned}$$

♣

Lösung 1.3.15. Algorithmus:

$$\begin{aligned}
y_1 &= r_1(p,q) = p \cdot p, \\
y_2 &= r_2(p,q,y_1) = y_1 - q, \\
y_3 &= r_3(p,q,y_1,y_2) = \sqrt{y_2}, \\
y_4 &= r_4(p,q,y_1,y_2,y_3) = p + y_3, \\
\lambda &= r_5(p,q,y_1,y_2,y_3,y_4) = q/y_4.
\end{aligned}$$

Offenbar sind $\lambda, y_1, y_2, y_3 > 0$.
Die Restabbildungen sind:

$$\begin{aligned}
\lambda &= R_4(p,q,y_1,y_2,y_3,y_4) = r_5(p,q,y_1,y_2,y_3,y_4) = q/y_4, \\
&= R_3(p,q,y_1,y_2,y_3) = R_4(p,q,y_1,y_2,y_3,r_4(p,q,y_2,y_3)) = q/(p+y_3), \\
&= R_2(p,q,y_1,y_2) = R_3(p,q,y_1,y_2,r_3(p,q,y_1,y_2)) = q/(p+\sqrt{y_2}), \\
&= R_1(p,q,y_1) = R_2(p,q,y_1,r_2(p,q,y_1)) = q/(p+\sqrt{y_1-q})
\end{aligned}$$

Die Konditionszahlen des Algorithmus sind

$$\begin{aligned}
K_1 &= \frac{y_1}{\lambda}\frac{\partial R_1}{\partial y_1}(p,q,y_1) = \frac{y_1}{\lambda}\,\frac{-q}{(p+\sqrt{y_1-q})^2}\,\frac{1}{2\sqrt{y_1-q}} = -\frac{y_1 q}{2\lambda y_3 y_4^2}, \\
K_2 &= \frac{y_2}{\lambda}\frac{\partial R_2}{\partial y_2}(p,q,y_1,y_2) = \frac{y_2}{\lambda}\,\frac{-q}{(p+\sqrt{y_2})^2}\,\frac{1}{2\sqrt{y_2}} = -\frac{y_3 q}{2\lambda y_4^2}, \\
K_3 &= \frac{y_3}{\lambda}\frac{\partial R_3}{\partial y_3}(p,q,y_1,y_2,y_3) = \frac{y_3}{\lambda}\,\frac{-q}{(p+y_3)^2} = -\frac{y_3 q}{\lambda y_4^2}, \\
K_4 &= \frac{y_4}{\lambda}\frac{\partial R_4}{\partial y_4}(p,q,y_1,y_2,y_3,y_4) = \frac{y_4}{\lambda}\,\frac{-q}{y_4^2} = -\frac{q}{\lambda y_4}
\end{aligned}$$

In der gegebenen Situation, also $q \ll p^2$, gilt

$$y_2 \approx p^2, \quad y_3 \approx p, \quad y_4 \approx 2p \quad \text{und} \quad \lambda \approx \frac{q}{2p}$$

Woraus für die Konditionszahlen des Algorithmus

$$|K_1|, |K_2| \approx \frac{1}{4}, \quad |K_3| \approx \frac{1}{2} \quad \text{sowie} \quad |K_4| \approx 1$$

folgt. Die Konditionszahlen des Algorithmus besitzen die gleiche Größenordnung wie die der Aufgabe. Der Algorithmus ist damit numerisch stabil. ♣

Inhaltsverzeichnis