1 Fehleranalyse

1.0 Einleitung

Eine der wichtigsten Aufgaben der numerischen Mathematik ist es, die Genauigkeit eines Rechenresultats zu beurteilen. Es gibt verschiedene Arten von Fehlern, die diese Genauigkeit begrenzen, man unterscheidet:

a) Fehler in den Eingabedaten der Rechnung,
b) Rundungsfehler,
c) Approximationsfehler.

Fehler in den Eingangsdaten lassen sich nicht vermeiden, wenn z.B. die Eingangsdaten Messwerte von nur beschränkter Genauigkeit sind. Rundungsfehler entstehen, wenn man, wie es in aller Regel geschieht, nur mit Zahlen einer endlichen aber festen Stellenzahl rechnet.

Approximationsfehler hängen mit den Rechenmethoden zusammen: Viele Methoden liefern selbst bei rundungsfehlerfreier Rechnung nicht die eigentlich gesuchte Lösung eines gegebenen Problems P, sondern nur die Lösung eines einfacheren Problems $\tilde{P}$, das P approximiert.

Beispiel: Das Problem P der Berechnung der Zahl e mit Hilfe der unendlichen Reihe

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots$$

approximiert man durch ein einfacheres Problem $\tilde{P}$, wenn man nur endlich viele Glieder dieser Reihe summiert.

Den dabei entstehenden Approximationsfehler nennt man oft auch Abbrechfehler (truncation error, manchmal werden damit aber auch nur die Rundungsfehler bezeichnet, die man beim Abschneiden einer Zahl nach einer bestimmten Stelle begeht).

Häufig erhält man das approximierende Problem $\tilde{P}$ durch „Diskretisierung“ des ursprünglichen Problems P: z.B. approximiert man Integrale durch endliche Summen, Differentialquotienten durch Differenzenquotienten usw. In diesem Zusammenhang bezeichnet man den Approximationsfehler auch als Diskretisierungsfehler.

In diesem Kapitel werden wir die Auswirkung der Eingangs- und Rundungsfehler einer Rechnung auf das Endresultat untersuchen. Approximationsfehler werden wir bei der Behandlung der einzelnen Methoden diskutieren. Eine systematische Behandlung von Rundungsfehlern findet man bei Sterbenz (1974).

1.1 Zahldarstellung

Aufgrund ihrer verschiedenen Zahldarstellung kann man zwei Arten von Rechengeräten unterscheiden

a) Analogrechner,
b) Digitalrechner.

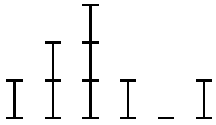
Beispiele für Analogrechner sind der Rechenschieber, das Planimeter sowie die elektronischen Analogrechner. Bei diesen Geräten werden Zahlen direkt durch physikalische Grössen (wie die Länge eines Stabes oder die Grösse einer Spannung) ersetzt und das mathematische Problem durch ein physikalisches simuliert, dessen Lösung in einem Experiment gemessen wird und so indirekt die Lösung des mathematischen Problems ergibt.

Beispiel: Auf den Skalen eines Rechenschiebers werden Zahlen durch Strecken der Länge $k \cdot \ln(x)$ dargestellt. Ihre Multiplikation wird daher durch Aneinanderlegen entsprechend langer Stäbe simuliert und das Resultat der Multiplikation als Ergebnis einer Längenmessung abgelesen.

Dementsprechend ist die Genauigkeit von Analogrechnern durch die physikalische Messgenauigkeit begrenzt.

Bei Digitalrechnern wird eine Zahl ähnlich der normalen Dezimaldarstellung durch eine endliche Folge diskreter physikalischer Grössen repräsentiert. Typische Vertreter sind die gewöhnlichen Tischrechenmaschinen und die elektronischen Digitalrechner.

Beispiel:

123101 $\longleftrightarrow$ 

Jeder Ziffer entspricht eine physikalische Grösse, etwa der Ziffer 8 die Spannung 8 Volt. Da z.B. im Dezimalsystem höchstens 10 verschiedene Ziffern darzustellen sind, sind an die Genauigkeit der Darstellung der Ziffern durch physikalische Grössen keine so hohen Ansprüche zu stellen wie bei einem Analogrechner (beispielsweise könnte man für die Darstellung der Ziffer

8 Spannungen zwischen 7.8 und 8.2 V tolerieren): Die Genauigkeit der Digitalrechner ist in diesem Sinne *nicht* durch die physikalische Messgenauigkeit beschränkt.

Aus technischen Gründen stützen sich die meisten modernen elektronischen Digitalrechner nicht auf die übliche Darstellung der Zahlen im Dezimalsystem, sondern auf das Dualsystem, in dem die Koeffizienten α_i der Dualzerlegung

$$x = \pm(\alpha_n 2^n + \alpha_{n-1}2^{n-1} + \cdots + \alpha_0 2^0 + \alpha_{-1}2^{-1} + \alpha_{-2}2^{-2} + \cdots),$$
$$\alpha_i = 0 \quad \text{oder} \quad \alpha_i = 1,$$

von x zur Darstellung benutzt werden. Um Verwechslungen mit der Dezimaldarstellung von Zahlen zu vermeiden, bezeichnet man im Dualsystem die Ziffern 0 und 1 durch die Ziffern **0** bzw. **L**.

Beispiel: Die Zahl $x = 18.5$ besitzt entsprechend der Zerlegung

$$18.5 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1}$$

die Dualdarstellung

L00L0.L

Der gewohnteren Zahldarstellung wegen verwenden wir vorwiegend das Dezimalsystem, weisen jedoch an den entsprechenden Stellen auf Unterschiede zum Dualsystem hin.

Wie das Beispiel $3.999\ldots = 4$ zeigt, ist die Dezimaldarstellung einer reellen Zahl x nicht notwendig eindeutig. Um solche Mehrdeutigkeiten auszuschliessen, soll im folgenden unter der Dezimaldarstellung einer Zahl im Zweifelsfall stets die endliche Darstellung gemeint sein. Ähnliches gilt für Dualdarstellungen.

Bei Digitalrechnern steht für die interne Darstellung einer Zahl nur eine feste endliche Anzahl n (= *Wortlänge*) von Dezimalstellen (Dualstellen) zur Verfügung, die durch die Konstruktion der Maschine festgelegt ist und, wenn überhaupt, nur auf ganze Vielfache $2n, 3n, \ldots$ (doppelte, dreifache ... Wortlänge) von n erweitert werden kann. Die Wortlänge von n Stellen kann auf verschiedene Weise zur Darstellung einer Zahl benutzt werden.

Bei der *Festpunktdarstellung* sind zusätzlich zur Zahl n auch die Zahlen n_1 und n_2 der Stellen vor bzw. nach dem Dezimalpunkt fixiert, $n = n_1 + n_2$ (in der Regel ist $n_1 = 0$ oder $n_1 = n$).

Beispiel: Für $n = 10$, $n_1 = 4$, $n_2 = 6$ hat man die Darstellungen

$$30.421 \rightarrow \boxed{0030}\boxed{421000}$$
$$0.0437 \rightarrow \underbrace{\boxed{0000}}_{n_1}\underbrace{\boxed{043700}}_{n_2}$$

Bei dieser Darstellung ist die Lage des (Dezimal-, Dual-) Punktes festgelegt. Nur sehr einfache Rechner, insbesondere für kaufmännische Rechnungen, beschränken sich heute noch auf die Festpunktdarstellung von Zahlen. Viel häufiger und bedeutsamer, insbesondere für wissenschaftliche Rechnungen, sind Rechner, in denen die *Gleitpunktdarstellung* von Zahlen realisiert ist. Hier liegt der Punkt nicht für alle Zahlen fest, und man muss dementsprechend bei jeder Zahl angeben, an der wievielten Stelle nach der ersten Ziffer der Darstellung der Punkt liegt. Dazu dient der sog. *Exponent*: Man nutzt aus, dass sich jede reelle Zahl x in der Form

$$x = a \cdot 10^b \quad (\text{bzw.} \quad x = a \cdot 2^b), \quad \text{wobei} \quad |a| < 1 \quad \text{und} \quad b \in \mathbb{Z}, \tag{1.1.1}$$

schreiben lässt, etwa $30.421 = 0.30421 \cdot 10^2$. Durch den *Exponenten* b wird die Lage des Dezimalpunktes in der *Mantisse* a angegeben. Mit Rutishauser wird häufig die Basis des Exponenten tiefgestellt („halblogarithmische Schreibweise“)

$$0.30421_{10}2$$

und analog im Dualsystem

$$\mathbf{0.L00L0L_2L0L}$$

für die Zahl 18.5. Natürlich stehen in jedem Rechner für die Gleitpunktdarstellung von Zahlen nur eine endliche feste Anzahl t bzw. e von (Dezimal-, Dual-) Stellen für die Darstellung der Mantisse bzw. des Exponenten zur Verfügung, $n = t + e$.

Beispiel: Für $t = 4, e = 2$ besitzt die Zahl 5420 im Dezimalsystem die Gleitpunktdarstellung

$$0.\boxed{5420}_{10}\boxed{04} \quad \text{oder kurz} \quad \boxed{5420}\boxed{04}.$$

Die Gleitpunktdarstellung einer Zahl ist i.a. nicht eindeutig. Z.B. hat man im letzten Beispiel wegen $5420 = 0.542_{10}4 = 0.0542_{10}5$ auch die Gleitpunktdarstellung

$$0.\boxed{0542}_{10}\boxed{05} \quad \text{oder} \quad \boxed{0542}\boxed{05}.$$

Normalisiert heisst diejenige Gleitpunktdarstellung einer Zahl, für die die erste Ziffer der Mantisse von 0 (bzw. **0**) verschieden ist. In (1.1.1) gilt dann $|a| \geq 10^{-1}$ bzw. im Dualsystem $|a| \geq 2^{-1}$. Als *wesentliche Stellen* einer Zahl werden alle Ziffern der Mantisse ohne die führenden Nullen bezeichnet.

Wir wollen uns im folgenden nur noch mit der normalisierten Gleitpunktdarstellung und der zugehörigen Gleitpunktrechnung befassen. Die Zahlen t und e bestimmen (zusammen mit der Basis $B = 10$ oder $B = 2$) der Zahldarstellung die Menge $A \subseteq \mathbb{R}$ von reellen Zahlen, die in der Maschine exakt dargestellt werden können, ihre Elemente heissen *Maschinenzahlen*.

Bei den heutigen Digitalrechnern ist die normalisierte Gleitpunktdarstellung die Regel. Seit 1985 gibt es den *IEEE-Standard für Gleitpunktrechnung*, der auf fast allen Rechnern realisiert ist. Dieser IEEE-Standard

gewährleistet, dass die Gleitpunktdarstellung und die Resultate der Gleitpunktrechnung unabhängig von der Rechnerarchitektur sind. Für eingehende Darstellungen des IEEE-Standards für die Gleitpunktrechnung verweisen wir auf Goldberg (1991) und Overton (2001).

1.2 Rundungsfehler und Gleitpunktrechnung

Die Menge A der in einer Maschine darstellbaren Zahlen ist endlich. Damit erhebt sich die Frage, wie man eine Zahl $x \notin A$, die keine Maschinenzahl ist, durch eine Maschinenzahl $g \in A$ approximieren kann. Dieses Problem stellt sich nicht nur bei der Eingabe von Daten in einen Rechner, sondern auch innerhalb des Rechners während des Ablaufs einer Rechnung. Wie einfachste Beispiele zeigen, kann nämlich das Resultat $x \pm y$, $x \cdot y$, x/y selbst der einfachen arithmetischen Operationen nicht zu A gehören, obwohl beide Operanden $x, y \in A$ Maschinenzahlen sind. Von einer vernünftigen Approximation einer Zahl $x \notin A$ durch eine Maschinenzahl $\mathrm{rd}(x) \in A$ wird man verlangen, dass

$$|x - \mathrm{rd}(x)| \leq |x - g| \quad \text{für alle} \quad g \in A \tag{1.2.1}$$

gilt. Man erhält ein solches $\mathrm{rd}(x)$ gewöhnlich durch *Rundung*.

Beispiel 1: $(t = 4)$

$$\begin{aligned} \mathrm{rd}(0.14285_{10}0) &= 0.1429_{10}0 \\ \mathrm{rd}(3.14159_{10}0) &= 0.3142_{10}1 \\ \mathrm{rd}(0.142842_{10}2) &= 0.1428_{10}2 \end{aligned}$$

Allgemein kann man bei t-stelliger Dezimalrechnung $\mathrm{rd}(x)$ so finden: Man stellt zunächst $x \notin A$ in normalisierter Form $x = a \cdot 10^b$ (1.1.1), $|a| \geq 10^{-1}$, dar. Die Dezimaldarstellung von $|a|$ sei

$$|a| = 0.\alpha_1\alpha_2 \ldots \alpha_t\alpha_{t+1} \ldots, \quad 0 \leq \alpha_i \leq 9, \quad \alpha_1 \neq 0.$$

Man bildet damit

$$a' := \begin{cases} 0.\alpha_1\alpha_2 \ldots \alpha_t & \text{falls} \quad 0 \leq \alpha_{t+1} \leq 4, \\ 0.\alpha_1\alpha_2 \ldots \alpha_t + 10^{-t} & \text{falls} \quad \alpha_{t+1} \geq 5, \end{cases}$$

d.h. man erhöht a_t um 1, falls die $(t+1)$-te Ziffer $a_{t+1} \geq 5$ ist, und schneidet nach der t-ten Ziffer ab. Schliesslich setzt man

$$\widetilde{\mathrm{rd}}(x) := \mathrm{sign}(x) \cdot a' \cdot 10^b.$$

Offensichtlich gilt für den relativen Fehler von $\widetilde{\mathrm{rd}}(x)$

$$\left| \frac{\widetilde{\mathrm{rd}}(x) - x}{x} \right| \leq \frac{5 \cdot 10^{-(t+1)}}{|a|} \leq 5 \cdot 10^{-t}$$

wegen $|a| \geq 10^{-1}$, oder mit der Abkürzung $\text{eps} := 5 \cdot 10^{-t}$,

$$\widetilde{\text{rd}}(x) = x(1+\varepsilon) \quad \text{mit} \quad |\varepsilon| \leq \text{eps}. \tag{1.2.2}$$

Die Zahl $\text{eps} = 5 \cdot 10^{-t}$ heisst *Maschinengenauigkeit*. Für das Dualsystem kann man $\widetilde{\text{rd}}(x)$ analog definieren: Ausgehend von der Zerlegung $x = a \cdot 2^b$ mit $2^{-1} \leq |a| < 1$ und der Dualdarstellung

$$|a| = \mathbf{0}.\alpha_1 \ldots \alpha_t \alpha_{t+1} \ldots, \quad \alpha_i = \mathbf{0} \quad \text{oder} \quad \alpha_i = \mathbf{L}$$

von $|a|$ bildet man

$$a' := \begin{cases} \mathbf{0}.\alpha_1 \ldots \alpha_t & \text{falls} \quad \alpha_{t+1} = \mathbf{0} \\ \mathbf{0}.\alpha_1 \ldots \alpha_t + 2^{-t} & \text{falls} \quad \alpha_{t+1} = \mathbf{L} \end{cases}$$

$$\widetilde{\text{rd}}(x) := \text{sign}(x) \cdot a' \cdot 2^b.$$

Auch hier gilt (1.2.2), wenn man die Maschinengenauigkeit durch $\text{eps} := 2^{-t}$ definiert.

Sofern $\widetilde{\text{rd}}(x) \in A$ eine Maschinenzahl ist, besitzt $\widetilde{\text{rd}}$ die Eigenschaft (1.2.1) einer korrekten Rundung rd, so dass man $\text{rd}(x) := \widetilde{\text{rd}}(x)$ für alle x mit $\widetilde{\text{rd}}(x) \in A$ definieren kann. Leider gibt es aber wegen der endlichen Anzahl e der für die Exponentendarstellung vorgesehenen Stellen immer Zahlen $x \notin A$ mit $\widetilde{\text{rd}}(x) \notin A$.

Beispiel 2: ($t = 4$, $e = 2$)

a)	$\widetilde{\text{rd}}(0.31794_{10}110)$	$= 0.3179_{10}110$	$\notin A$
b)	$\widetilde{\text{rd}}(0.99997_{10}99)$	$= 0.1000_{10}100$	$\notin A$
c)	$\widetilde{\text{rd}}(0.012345_{10} \pm 99)$	$= 0.1235_{10} \pm 100$	$\notin A$
d)	$\widetilde{\text{rd}}(0.54321_{10} \pm 110)$	$= 0.5432_{10} \pm 110$	$\notin A$

In den Fällen a) und b) ist der Exponent zu gross: *Exponentenüberlauf*. Fall b) ist besonders pathologisch: Hier tritt der Exponentenüberlauf erst nach dem Aufrunden ein. Die Fälle c) und d) sind Beispiele für *Exponentenunterlauf*, der Exponent der darzustellenden Zahl ist zu klein. In den Fällen c) und d) kann man sich durch die Definitionen

$$\begin{aligned} &\text{rd}(0.012345_{10} - 99) := 0.0123_{10} - 99 \in A \\ &\text{rd}(0.54321_{10} - 110) := 0 \in A \end{aligned} \tag{1.2.3}$$

vor einem Exponentunterlauf retten, doch dann gilt für rd statt $\widetilde{\text{rd}}$ nicht mehr (1.2.2), der relative Fehler von $\text{rd}(x)$ kann grösser als eps sein! Alle Rechenanlagen melden einen Exponentenüberlauf als Fehler. Ein Exponentenunterlauf wird (in der Regel) nicht als Fehler gemeldet, sondern es wird $\text{rd}(x)$ ähnlich wie in Beispiel (1.2.3) gebildet. In den übrigen Fällen, $\widetilde{\text{rd}}(x) \in A$, wird gewöhnlich (nicht bei allen Anlagen!) $\text{rd}(x) := \widetilde{\text{rd}}(x)$ als Rundung genommen.

Da Exponentenüber- und -unterläufe verhältnismässig seltene Ereignisse sind (bei den heutigen Anlagen ist e ausreichend gross), die man überdies durch geeignete Umskalierung der Daten vermeiden kann, wollen wir für die folgende Diskussion die idealisierte Annahme $e = \infty$ machen, so dass für die Rundungsabbildung $\mathrm{rd} := \widetilde{\mathrm{rd}}$ gilt

$$\begin{aligned} &\mathrm{rd} : \mathbb{R} \to A, \\ &\mathrm{rd}(x) = x(1+\varepsilon) \quad \text{mit } |\varepsilon| \le \mathrm{eps} \quad \text{für alle} \quad x \in \mathbb{R}. \end{aligned} \tag{1.2.4}$$

Dementsprechend wird bei den künftigen Beispielen nur noch die Mantissenlänge t angegeben. Man beachte, dass für $e \neq \infty$ einige der folgenden Aussagen über Rundungsfehler im Falle von Exponentenüber- oder -unterlauf nicht gelten.

Da das Resultat einer arithmetischen Operation $x \pm y$, $x \times y$, x/y nicht eine Maschinenzahl sein muss, selbst wenn es die Operanden x und y sind, kann man nicht erwarten, dass diese Operationen auf einem Rechner exakt realisiert sind. Statt der exakten Operationen $+$, $-$, $\times$, $/$ werden Ersatzoperationen $+^*$, $-^*$, $\times^*$, $/^*$, sog. *Gleitpunktoperationen*, angeboten, die die arithmetischen Operationen möglichst gut approximieren (v. Neumann and Goldstein (1947)). Solche Ersatzoperationen kann man etwa mit Hilfe der Rundung so definieren:

$$\begin{aligned} x +^* y &:= \mathrm{rd}(x+y) \\ x -^* y &:= \mathrm{rd}(x-y) \\ x \times^* y &:= \mathrm{rd}(x \times y) \\ x /^* y &:= \mathrm{rd}(x/y) \end{aligned} \quad \text{für} \quad x, y \in A. \tag{1.2.5}$$

Es gilt dann wegen (1.2.4)

$$\begin{aligned} x +^* y &= (x+y)(1+\varepsilon_1) \\ x -^* y &= (x-y)(1+\varepsilon_2) \\ x \times^* y &= (x \times y)(1+\varepsilon_3) \\ x /^* y &= (x/y)(1+\varepsilon_4) \end{aligned} \quad |\varepsilon_i| \le \mathrm{eps}. \tag{1.2.6}$$

Bei den heutigen Anlagen sind die Gleitpunktoperationen $\pm^*, \ldots$ häufig nicht genau durch (1.2.5) definiert, doch in aller Regel so, dass trotzdem (1.2.6) mit eventuell etwas schlechteren Schranken $|\varepsilon_i| \le k \cdot \mathrm{eps}$, $k \ge 1$ eine kleine Zahl, gilt. Da dies für das folgende unwesentlich ist, nehmen wir der Einfachheit halber an, dass die Gleitpunktoperationen durch (1.2.5) definiert sind und deshalb die Eigenschaft (1.2.6) haben.

Es sei darauf hingewiesen, dass die Gleitpunktoperationen nicht den üblichen Gesetzen der arithmetischen Operationen genügen. So ist z.B.

$$x +^* y = x, \quad \text{falls} \quad |y| < \frac{\mathrm{eps}}{B}|x|, \quad x, y \in A,$$

wobei B die Basis des Zahlsystems ist. Die Maschinengenauigkeit eps kann man z.B. als die kleinste positive Maschinenzahl g mit $1 +^* g > 1$ definieren,

$$\text{eps} = \min\{\, g \in A \mid 1 +^* g > 1 \text{ und } g > 0 \,\}.$$

Ferner sind für die Gleitpunktoperationen die Assoziativ- und Distributivgesetze falsch.

Beispiel 3: ($t = 8$) Für

$$\begin{aligned} a &:= \quad 0.23371258_{10} - 4 \\ b &:= \quad 0.33678429_{10}2 \\ c &:= -0.33677811_{10}2 \end{aligned}$$

gilt

$$\begin{aligned} a +^* (b +^* c) &= 0.23371258_{10} - 4 +^* 0.61800000_{10} - 3 \\ &= 0.64137126_{10} - 3, \\ (a +^* b) +^* c &= 0.33678452_{10}2 -^* 0.33677811_{10}2 \\ &= 0.64100000_{10} - 3. \end{aligned}$$

Das exakte Resultat ist

$$a + b + c = 0.641371258_{10} - 3.$$

Man achte auf den Fall der *Auslöschung* bei der Subtraktion von Zahlen $x, y \in A$, die das gleiche Vorzeichen haben: Dieser Fall liegt vor, wenn die Zahlen x, y, wie in dem Beispiel

$$\begin{aligned} x &= 0.315876_{10}1 \\ y &= 0.314289_{10}1 \end{aligned}$$

Darstellungen (1.1.1) mit dem gleichen Exponenten besitzen und eine oder mehrere führende Ziffern in den Mantissen übereinstimmen. Bei der Subtraktion $x-y$ tritt „Auslöschung“ der gemeinsamen führenden Ziffern der Mantisse ein, das exakte Resultat $x - y$ lässt sich als Maschinenzahl darstellen, sofern x und y Maschinenzahlen waren, so dass bei der Ausführung der Subtraktion kein *neuer* Rundungsfehler anfällt: $x -^* y = x - y$. In diesem Sinne ist die Subtraktion im Falle der Auslöschung eine sehr harmlose Operation. Wir werden jedoch im nächsten Abschnitt sehen, dass im Falle von Auslöschung eine ausserordentlich gefährliche Situation vorliegt, was die Fortpflanzung der *alten* Fehler angeht, die man bei der Berechnung von x und y bereits *vor* Ausführung der Subtraktion $x - y$ begangen hat.

Für das Ergebnis von Gleitpunktrechnungen hat sich eine bequeme aber etwas unpräzise Schreibweise eingebürgert, die wir im folgenden öfters benutzen wollen: Steht für einen arithmetischen Ausdruck E fest, wie er berechnet werden soll (dies wird evtl. durch geeignete Klammerung vorgeschrieben), so wird durch $\text{gl}(E)$ der Wert des Ausdrucks bezeichnet, den man bei Gleitpunktrechnung erhält.

Beispiel 4:

$$\begin{aligned} \mathrm{gl}(x \times y) &:= x \times^* y \\ \mathrm{gl}(a + (b + c)) &:= a +^* (b +^* c) \\ \mathrm{gl}((a + b) + c) &:= (a +^* b) +^* c \end{aligned}$$

Man benutzt diese Schreibweise auch in Fällen wie $\mathrm{gl}(\sqrt{x})$, $\mathrm{gl}(\cos(x))$ etc., wenn auf einer Rechenanlage die betreffenden Funktionen $\sqrt{}, \cos, \ldots$ durch Ersatzfunktionen $\sqrt{}^*, \cos^*, \ldots$ realisiert sind, d.h. $\mathrm{gl}(\sqrt{x}) := \sqrt{x}^*$ usw.

Die arithmetischen Operationen $+$, $-$, $\times$, $/$ sowie alle einfachen Funktionen, wie etwa $\sqrt{}$, cos, für die Gleitpunkt-Ersatzfunktionen vorliegen, heissen im folgenden *elementare Operationen*.

1.3 Fehlerfortpflanzung

Im letzten Abschnitt sahen wir bereits (s. Beispiel 3), dass zwei verschiedene, mathematisch äquivalente Methoden, $(a+b)+c$, $a+(b+c)$, zur Auswertung des Ausdruckes $a + b + c$ bei Gleitpunktrechnung zu unterschiedlichen Resultaten führen können. Es ist deshalb aus numerischen Gründen wichtig, genau zwischen verschiedenen mathematisch äquivalenten Formulierungen einer Rechenmethode zu unterscheiden: Wir bezeichnen als *Algorithmus* eine der Reihenfolge nach eindeutig festgelegte Sequenz von endlich vielen „elementaren“ Operationen (wie sie etwa in einem Rechner-Programm gegeben ist), mit denen man aus gewissen Eingabedaten die Lösung eines Problems berechnen kann.

Wir wollen den Begriff eines Algorithmus etwas formalisieren und dazu annehmen, dass das Problem darin besteht, aus endlich vielen Eingabedaten, gewissen reellen Zahlen $x_1, x_2, \ldots, x_n$, nur endlich viele Resultatdaten, nämlich weitere reelle Zahlen $y_1, y_2, \ldots, y_m$, zu berechnen. Ein Problem dieser Art zu lösen heisst, den Wert $y = \varphi(x)$ einer gewissen Funktion $\varphi : D \mapsto \mathbb{R}^m$, $D \subseteq \mathbb{R}^n$ zu bestimmen, wobei $y \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ und φ durch m reelle Funktionen φ_i

$$y_i = \varphi_i(x_1, x_2, \ldots x_n), \quad i = 1, 2, \ldots, m,$$

gegeben ist. Ein Algorithmus ist eine eindeutige Rechenvorschrift zur Berechnung von $\varphi(x)$. In jedem Stadium des Algorithmus sind Zwischenergebnisse gegeben, die wir, etwa im i-ten Stadium, uns durch einen reellen Vektor

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_{n_i}^{(i)} \end{bmatrix} \in \mathbb{R}^{n_i}$$

der Länge n_i repräsentiert denken. Der Übergang zum nächsten Stadium $i+1$ wird durch eine *elementare Abbildung*

$$\varphi^{(i)} : D_i \mapsto D_{i+1}, \quad D_k \subseteq \mathbb{R}^{n_k},$$

vermittelt, $x^{(i+1)} = \varphi^{(i)}(x^{(i)})$. Die elementaren Abbildungen $\varphi^{(i)}$ sind eindeutig durch den Algorithmus bestimmt, wenn man von trivialen Mehrdeutigkeiten absieht, die von Permutationen der Komponenten der Vektoren $x^{(k)}$ herrühren, d.h. von der im Prinzip willkürlichen Anordnung der Zwischenresultate eines Rechenstadiums in einem Vektor.

Die endliche Sequenz der elementaren Operationen eines Algorithmus entspricht einer Zerlegung von φ in elementare Abbildungen

$$\begin{aligned} &\varphi^{(i)} : D_i \mapsto D_{i+1}, \quad i = 0, 1, \ldots, r, \quad D_j \subseteq \mathbb{R}^{n_j} \quad \text{mit} \\ &\varphi = \varphi^{(r)} \circ \varphi^{(r-1)} \circ \cdots \circ \varphi^{(0)}, \quad D_0 = D, \quad D_{r+1} \subseteq \mathbb{R}^{n_{r+1}} = \mathbb{R}^m. \end{aligned} \tag{1.3.1}$$

Beispiel 1: Für die Berechnung von $y = \varphi(a, b, c) := a + b + c$ hat man die beiden Algorithmen

$$\eta := a + b, \quad y := c + \eta, \quad bzw. \quad \eta := b + c, \quad y := a + \eta.$$

Die Zerlegungen (1.3.1) sind hier gegeben durch

$$\varphi^{(0)}(a, b, c) := \begin{bmatrix} a + b \\ c \end{bmatrix} \in \mathbb{R}^2, \quad \varphi^{(1)}(u, v) := u + v \in \mathbb{R},$$

bzw.

$$\varphi^{(0)}(a, b, c) := \begin{bmatrix} a \\ b + c \end{bmatrix} \in \mathbb{R}^2, \quad \varphi^{(1)}(u, v) := u + v \in \mathbb{R}.$$

Beispiel 2: Wegen $a^2 - b^2 = (a + b)(a - b)$ hat man für die Berechnung von $\varphi(a, b) := a^2 - b^2$ die beiden Algorithmen:

Algorithmus 1:		Algorithmus 2:	
	$\eta_1 := a \times a,$		$\eta_1 := a + b,$
	$\eta_2 := b \times b,$		$\eta_2 := a - b,$
	$y := \eta_1 - \eta_2,$		$y := \eta_1 \times \eta_2.$

Die zugehörigen Zerlegungen (1.3.1) sind gegeben durch

Algorithmus 1:

$$\varphi^{(0)}(a, b) := \begin{bmatrix} a^2 \\ b \end{bmatrix}, \quad \varphi^{(1)}(u, v) := \begin{bmatrix} u \\ v^2 \end{bmatrix}, \quad \varphi^{(2)}(u, v) := u - v.$$

Algorithmus 2:

$$\varphi^{(0)}(a, b) := \begin{bmatrix} a \\ b \\ a + b \end{bmatrix}, \quad \varphi^{(1)}(a, b, v) := \begin{bmatrix} v \\ a - b \end{bmatrix}, \quad \varphi^{(2)}(u, v) := u \cdot v.$$

Wir wollen nun die Gründe dafür untersuchen, weshalb verschiedene Algorithmen zur Lösung eines Problems i.a. unterschiedliche Resultate liefern, um Kriterien für die Beurteilung der Güte von Algorithmen zu gewinnen. Dabei spielt die Fortpflanzung der Rundungsfehler eine wichtige Rolle, wie wir zunächst an dem Beispiel der mehrfachen Summe $y = a + b + c$ sehen werden (s. Beispiel 3 von 1.2). Bei Gleitpunktrechnung erhält man statt y einen Näherungswert $\tilde{y} = \mathrm{gl}\big((a+b)+c\big)$, für den wegen (1.2.6) gilt

$$\begin{aligned}\eta :&= \mathrm{gl}(a+b) = (a+b)(1+\varepsilon_1)\\ \tilde{y} :&= \mathrm{gl}(\eta + c) = (\eta + c)(1+\varepsilon_2)\\ &= \big((a+b)(1+\varepsilon_1)+c\big)(1+\varepsilon_2)\\ &= (a+b+c)\Big(1 + \frac{a+b}{a+b+c}\varepsilon_1(1+\varepsilon_2) + \varepsilon_2\Big).\end{aligned}$$

Für den relativen Fehler $\varepsilon_y := (\tilde{y} - y)/y$ von $\tilde{y}$ gilt daher

$$\varepsilon_y = \frac{a+b}{a+b+c}\varepsilon_1(1+\varepsilon_2) + \varepsilon_2$$

oder in erster Näherung bei Vernachlässigung von Termen höherer Ordnung wie $\varepsilon_1\varepsilon_2$

$$\varepsilon_y \doteq \frac{a+b}{a+b+c}\varepsilon_1 + 1 \cdot \varepsilon_2.$$

Die Verstärkungsfaktoren $(a+b)/(a+b+c)$ bzw. 1 geben an, wie stark sich die Rundungsfehler $\varepsilon_1, \varepsilon_2$ im relativen Fehler ε_y des Resultats auswirken. Der kritische Faktor ist $(a+b)/(a+b+c)$: Je nachdem, ob $|a+b|$ oder $|b+c|$ kleiner ist, ist es günstiger („numerisch stabiler“) die Summe $a+b+c$ nach der Formel $(a+b)+c$ bzw. $a+(b+c)$ zu bilden.

Im Beispiel des letzten Abschnittes ist

$$\begin{aligned}\frac{a+b}{a+b+c} &= \frac{0.33\ldots_{10}2}{0.64\ldots_{10}-3} \approx \frac{1}{2}10^5,\\ \frac{b+c}{a+b+c} &= \frac{0.618\ldots_{10}-3}{0.64\ldots_{10}-3} \approx 0.97,\end{aligned}$$

was die höhere Genauigkeit von $\mathrm{gl}\big(a+(b+c)\big)$ erklärt.

Diese Methode, die Fortpflanzung spezieller Fehler durch Vernachlässigung von Grössen höherer Ordnung zu studieren, lässt sich systematisch zu einer *differentiellen Fehleranalyse* des Algorithmus (1.3.1)

$$\varphi = \varphi^{(r)} \circ \varphi^{(r-1)} \circ \cdots \circ \varphi^{(0)}$$

zur Berechnung von $\varphi(x)$ ausbauen. Dazu müssen wir untersuchen, wie sich die Eingangsfehler Δx von x und die im Laufe des Algorithmus begangenen Rundungsfehler auf das Endresultat $y = \varphi(x)$ auswirken. Wir wollen dies zunächst nur für die Eingangsfehler Δx ausführen und die dabei gewonnenen

Ergebnisse später auf die Fortpflanzung der Rundungsfehler anwenden. Wir setzen dazu voraus, dass die Funktion

$$\varphi : D \to \mathbb{R}^m, \quad \varphi(x) = \begin{bmatrix} \varphi_1(x_1, \ldots, x_n) \\ \vdots \\ \varphi_m(x_1, \ldots, x_n) \end{bmatrix}$$

auf einer offenen Teilmenge D des $\mathbb{R}^n$ definiert ist und die Komponentenfunktion φ_i, $i = 1, \ldots, n$, von φ auf D stetig differenzierbar sind. Sei $\tilde{x}$ ein Näherungswert für x. Mit

$$\Delta x_j := \tilde{x}_j - x_j, \quad \Delta x := \tilde{x} - x,$$

bezeichnen wir dann den *absoluten Fehler* von $\tilde{x}_i$ bzw. $\tilde{x}$ und als *relativen Fehler* von $\tilde{x}_i$ die Grössen

$$\varepsilon_{x_i} := \frac{\tilde{x}_i - x_i}{x_i}, \quad \text{falls} \quad x_i \neq 0.$$

Ersetzt man die Eingabedaten x durch $\tilde{x}$, so erhält man als Resultat $\tilde{y} := \varphi(\tilde{x})$ statt $y = \varphi(x)$. Durch Taylor-Entwicklung ergibt sich unter Vernachlässigung von Grössen höherer Ordnung

$$\begin{aligned} \Delta y_i :&= \tilde{y}_i - y_i = \varphi_i(\tilde{x}) - \varphi_i(x) \doteq \sum_{j=1}^{n} \frac{\partial \varphi_i(x)}{\partial x_j} (\tilde{x}_j - x_j) \\ &= \sum_{j=1}^{n} \frac{\partial \varphi_i(x)}{\partial x_j} \Delta x_j, \quad i = 1, \ldots, m, \end{aligned} \tag{1.3.2}$$

oder in Matrixschreibweise

$$\Delta y = \begin{bmatrix} \Delta y_1 \\ \vdots \\ \Delta y_m \end{bmatrix} \doteq \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_1} & \cdots & \frac{\partial \varphi_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial \varphi_m}{\partial x_1} & \cdots & \frac{\partial \varphi_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \end{bmatrix} = D\varphi(x) \Delta x \tag{1.3.3}$$

mit der Funktionalmatrix $D\varphi(x)$.

Dabei soll „$\doteq$“ statt „$=$“ andeuten, dass die betr. Gleichungen nur in erster Näherung richtig sind. Der Proportionalitätsfaktor $\partial \varphi_i(x)/\partial x_j$ in (1.3.2) misst die Empfindlichkeit, mit der y_i auf absolute Änderungen Δx_j von x_j reagiert.

Ist $y_i \neq 0$ und $x_j \neq 0$ für alle i und j, so folgt aus (1.3.2) eine Fehlerfortpflanzungsformel für die relativen Fehler:

$$\varepsilon_{y_i} \doteq \sum_{j=1}^{n} \frac{x_j}{\varphi_i(x)} \cdot \frac{\partial \varphi_i(x)}{\partial x_j} \cdot \varepsilon_{x_j}. \tag{1.3.4}$$

Wiederum gibt der Faktor $(x_j/\varphi_i)\partial\varphi_i/\partial x_j$ an, wie stark sich ein relativer Fehler in x_j auf den relativen Fehler von y_i auswirkt. Die Verstärkungsfaktoren $(x_j/\varphi_i)\partial\varphi_i/\partial x_j$ für die relativen Fehler haben den Vorteil, dass sie von der Skalierung der y_i und x_j unabhängig sind. Sind sie gross, spricht man von einem *schlecht konditionierten*, andernfalls von einem *gut konditionierten* Problem. Bei schlecht konditionierten Problemen bewirken kleine relative Fehler in den Eingangsdaten x grosse relative Fehler in den Resultaten $y = \varphi(x)$. Die hier gegebene Definition der Konditionszahlen hat den Nachteil, dass sie nur für nichtverschwindende y_i, x_j sinnvoll ist. Ausserdem ist sie für viele Zwecke zu unpraktisch (die Kondition von φ wird durch $m \cdot n$ Zahlen beschrieben). Es werden deshalb auch andere einfachere Definitionen für die Kondition eines Problems gegeben. So ist es z.B. in der linearen Algebra vielfach üblich, Zahlen c, für die bzgl. einer geeigneten Norm $\|\cdot\|$ gilt

$$\frac{\|\varphi(\tilde{x}) - \varphi(x)\|}{\|\varphi(x)\|} \leq c \cdot \frac{\|\tilde{x} - x\|}{\|x\|}$$

als Konditionszahlen zu bezeichnen (s. Abschnitt 4.4).

Beispiel 3: Für $y = \varphi(a, b, c) := a + b + c$ hat man nach (1.3.4):

$$\varepsilon_y \doteq \frac{a}{a+b+c}\varepsilon_a + \frac{b}{a+b+c}\varepsilon_b + \frac{c}{a+b+c}\varepsilon_c.$$

Das Problem ist gut konditioniert, falls jeder Summand a, b, c klein gegenüber $a + b + c$ ist.

Beispiel 4: Sei $y = \varphi(p, q) := -p + \sqrt{p^2 + q}$. Es ist

$$\frac{\partial\varphi}{\partial p} = -1 + \frac{p}{\sqrt{p^2+q}} = \frac{-y}{\sqrt{p^2+q}}, \quad \frac{\partial\varphi}{\partial q} = \frac{1}{2\sqrt{p^2+q}},$$

so dass

$$\varepsilon_y \doteq \frac{-p}{\sqrt{p^2+q}}\varepsilon_p + \frac{q}{2y\sqrt{p^2+q}}\varepsilon_q = -\frac{p}{\sqrt{p^2+q}}\varepsilon_p + \frac{p+\sqrt{p^2+q}}{2\sqrt{p^2+q}}\varepsilon_q.$$

Wegen

$$\left|\frac{p}{\sqrt{p^2+q}}\right| \leq 1, \quad \left|\frac{p+\sqrt{p^2+q}}{2\sqrt{p^2+q}}\right| \leq 1 \quad \text{für} \quad q > 0,$$

ist φ gut konditioniert, falls $q > 0$, und schlecht konditioniert, falls etwa $q \approx -p^2$.

Für die arithmetischen Operationen erhält man aus (1.3.4) die Fehlerfortpflanzungsformeln (für $x \neq 0$, $y \neq 0$)

(1.3.5)

1. $\varphi(x,y) := x \cdot y$: $\quad \varepsilon_{xy} \doteq \varepsilon_x + \varepsilon_y,$
2. $\varphi(x,y) := x/y$: $\quad \varepsilon_{x/y} \doteq \varepsilon_x - \varepsilon_y,$
3. $\varphi(x,y) := x \pm y$: $\quad \varepsilon_{x\pm y} = \dfrac{x}{x\pm y}\varepsilon_x \pm \dfrac{y}{x\pm y}\varepsilon_y, \quad$ falls $\quad x \pm y \neq 0,$
4. $\varphi(x) := \sqrt{x}$: $\quad \varepsilon_{\sqrt{x}} \doteq \dfrac{1}{2}\varepsilon_x.$

Es folgt, dass das Multiplizieren, Dividieren und Wurzelziehen keine gefährlichen Operationen sind: Die relativen Fehler der Eingabedaten pflanzen sich nicht stark in das Resultat fort. Dieselbe Situation liegt bei der Addition vor, sofern die Summanden x und y gleiches Vorzeichen haben: Die Konditionszahlen $x/(x+y)$, $y/(x+y)$ liegen zwischen 0 und 1 und ihre Summe ist 1, somit gilt

$$\left|\varepsilon_{x+y}\right| \leq \max\{\, |\varepsilon_x|, |\varepsilon_y| \,\}.$$

Ist ein Summand klein gegenüber dem anderen und ist er mit einem grossen relativen Fehler behaftet, so hat trotzdem das Resultat $x+\ y$ nach (1.3.5) nur einen kleinen relativen Fehler, wenn der grössere Summand einen kleinen relativen Fehler hat. Man spricht dann von *Fehlerdämpfung*. Wenn dagegen bei der Addition die Summanden x und y verschiedenes Vorzeichen haben, ist mindestens einer der Faktoren $|x/x+y)|$, $|y/(x+y)|$ grösser als 1, und es wird mindestens einer der relativen Fehler ε_x, ε_y verstärkt. Diese Verstärkung ist dann besonders gross, wenn $x \approx -y$ und damit Auslöschung bei Bildung von $x+y$ auftritt.

Wir wollen nun die allgemeine Formel (1.3.3) benutzen, um die Fortpflanzung von Rundungsfehlern bei einem gegebenen Algorithmus zu studieren. Ein Algorithmus zur Berechnung der Funktion $\varphi : D \mapsto \mathbb{R}^m$, $D \subseteq \mathbb{R}^n$, für gegebenes $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \in D$ entspricht einer bestimmten Zerlegung der Abbildung φ in elementare Abbildungen $\varphi^{(i)}$ (1.3.1) und führt von $x^{(0)} := x$ über eine Kette von Zwischenergebnissen

$$x = x^{(0)} \to \varphi^{(0)}(x^{(0)}) = x^{(1)} \to \cdots \to \varphi^{(r)}(x^{(r)}) = x^{(r+1)} = y \tag{1.3.6}$$

zum Resultat y. Wir nehmen für die folgende Diskussion wieder an, dass jedes $\varphi^{(i)}$ stetig differenzierbar ist und bezeichnen mit $\psi^{(i)}$ die „Restabbildung“

$$\psi^{(i)} = \varphi^{(r)} \circ \varphi^{(r-1)} \circ \cdots \circ \varphi^{(i)} : D_i \mapsto \mathbb{R}^m, \quad i = 0, 1, 2, \ldots, r.$$

Es ist dann $\psi^{(0)} \equiv \varphi$. Mit $D\varphi^{(i)}$ bzw. $D\psi^{(i)}$ bezeichnen wir die Funktionalmatrizen der Abbildungen $\varphi^{(i)}$ bzw. $\psi^{(i)}$. Bekanntlich multiplizieren sich die Funktionalmatrizen, wenn man Abbildungen zusammensetzt

$$D(f \circ g)(x) = Df(g(x))Dg(x).$$

Also gilt für $i = 0, 1, 2, \ldots, r$

$$
\begin{aligned}
D\varphi(x) &= D\varphi^{(r)}(x^{(r)})D\varphi^{(r-1)}(x^{(r-1)})\dots D\varphi^{(0)}(x),\\
D\psi^{(i)}(x^{(i)}) &= D\varphi^{(r)}(x^{(r)})D\varphi^{(r-1)}(x^{(r-1)})\dots D\varphi^{(i)}(x^{(i)}).
\end{aligned}
\tag{1.3.7}
$$

In Gleitpunktarithmetik erhält man unter dem Einfluss der Eingangsfehler Δx und der Rundungsfehler statt der exakten Zwischenresultate $x^{(i)}$ Näherungswerte $\tilde{x}^{(i+1)} = \mathrm{gl}\big(\varphi^{(i)}(\tilde{x}^{(i)})\big)$. Für die Fehler $\Delta x^{(i)} = \tilde{x}^{(i)} - x^{(i)}$ gilt daher

$$
\Delta x^{(i+1)} = \Big(\mathrm{gl}\big(\varphi^{(i)}(\tilde{x}^{(i)})\big) - \varphi^{(i)}(\tilde{x}^{(i)})\Big) + \Big(\varphi^{(i)}(\tilde{x}^{(i)}) - \varphi^{(i)}(x^{(i)})\Big).
\tag{1.3.8}
$$

Nun ist wegen (1.3.3) in erster Näherung

$$
\varphi^{(i)}(\tilde{x}^{(i)}) - \varphi^{(i)}(x^{(i)}) \doteq D\varphi^{(i)}(x^{(i)}) \cdot \Delta x^{(i)}.
\tag{1.3.9}
$$

Wenn die elementaren Abbildungen $\varphi^{(i)}$ so beschaffen sind, dass man ähnlich wie bei den elementaren arithmetischen Operationen (s. (1.2.6)) bei der Gleitpunktauswertung von $\varphi^{(i)}$ das gerundete exakte Resultat erhält, so gilt

$$
\mathrm{gl}\big(\varphi^{(i)}(u)\big) = \mathrm{rd}\big(\varphi^{(i)}(u)\big).
\tag{1.3.10}
$$

Man beachte, dass die Abbildung $\varphi^{(i)} : D_i \mapsto D_{i+1} \subseteq \mathbb{R}^{n_{i+1}}$ durch einen Vektor

$$
\varphi^{(i)}(u) = \begin{bmatrix} \varphi_1^{(i)}(u) \\ \vdots \\ \varphi_{n_{i+1}}^{(i)}(u) \end{bmatrix}
$$

von reellen Funktionen $\varphi_j^{(i)} : D_i \mapsto \mathbb{R}$ gegeben und (1.3.10) komponentenweise zu lesen ist:

$$
\begin{aligned}
\mathrm{gl}\big(\varphi_j^{(i)}(u)\big) = \mathrm{rd}\big(\varphi_j^{(i)}(u)\big) &= (1+\varepsilon_j)\cdot\varphi_j^{(i)}(u),\\
|\varepsilon_j| \le \mathrm{eps}, \quad j &= 1,2,\dots,n_{i+1}.
\end{aligned}
\tag{1.3.11}
$$

Dabei ist ε_j der bei der Berechnung der j-ten Komponente von $\varphi^{(i)}$ in Gleitpunktarithmetik auftretende *neue* relative Rundungsfehler. Die Gleichung (1.3.10) lässt sich also in der Form

$$
\mathrm{gl}(\varphi^{(i)}(u)) = (I + E_{i+1}) \cdot \varphi^{(i)}(u)
$$

mit der Einheitsmatrix I und der diagonalen Fehlermatrix

$$
E_{i+1} := \begin{bmatrix} \varepsilon_1 & 0 & \cdots & 0 \\ 0 & \varepsilon_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \varepsilon_{n_{i+1}} \end{bmatrix}, \quad |\varepsilon_j| \le \mathrm{eps}
$$

schreiben. Damit lässt sich die erste Klammer in (1.3.8) umformen:

$$\mathrm{gl}\big(\varphi^{(i)}(\tilde{x}^{(i)})\big) - \varphi^{(i)}(\tilde{x}^{(i)}) = E_{i+1} \cdot \varphi^{(i)}(\tilde{x}^{(i)}).$$

Da $\tilde{x}^{(i)}$ in erster Näherung gleich $x^{(i)}$ ist, gilt ebenso in erster Näherung

$$\begin{aligned} \mathrm{gl}\big(\varphi^{(i)}(\tilde{x}^{(i)})\big) - \varphi^{(i)}(\tilde{x}^{(i)}) &\doteq E_{i+1}\varphi^{(i)}(x^{(i)}) \\ &= E_{i+1}x^{(i+1)} =: \alpha_{i+1}. \end{aligned} \tag{1.3.12}$$

α_{i+1} lässt sich als der bei der Auswertung von $\varphi^{(i)}$ in Gleitpunktarithmetik *neu entstehende* absolute Rundungsfehler, die Diagonalelemente von E_{i+1} als die entsprechenden relativen Rundungsfehler interpretieren. Für $\Delta x^{(i+1)}$ gilt daher wegen (1.3.8), (1.3.9) und (1.3.12) in erster Näherung

$$\Delta x^{(i+1)} \doteq \alpha_{i+1} + D\varphi^{(i)}(x^{(i)})\Delta x^{(i)} = E_{i+1} \cdot x^{(i+1)} + D\varphi^{(i)}(x^{(i)})\Delta x^{(i)},$$

wobei $\Delta x^{(0)} := \Delta x$. Man erhält daraus

$$\begin{aligned} \Delta x^{(1)} &\doteq D\varphi^{(0)}\Delta x + \alpha_1 \\ \Delta x^{(2)} &\doteq D\varphi^{(1)}[D\varphi^{(0)}\Delta x + \alpha_1] + \alpha_2 \\ \Delta y = \Delta x^{(r+1)} &\doteq D\varphi^{(r)} \cdots D\varphi^{(0)}\Delta x + D\varphi^{(r)} \cdots D\varphi^{(1)}\alpha_1 + \cdots + \alpha_{r+1}. \end{aligned}$$

Wegen (1.3.7) bekommt man so schliesslich für den Einfluss des Eingangsfehlers Δx und der Rundungsfehler α_i auf das Resultat $y = x^{(r+1)} = \varphi(x)$ die Formeln

$$\begin{aligned} \Delta y &\doteq D\varphi(x)\Delta x + D\psi^{(1)}(x^{(1)})\alpha_1 + \cdots + D\psi^{(r)}(x^{(r)})\alpha_r + \alpha_{r+1} \\ &= D\varphi(x)\Delta x + D\psi^{(1)}(x^{(1)})E_1x^{(1)} + \cdots + D\psi^{(r)}(x^{(r)})E_rx^{(r)} \\ &\quad + E_{r+1}y. \end{aligned} \tag{1.3.13}$$

Die Grösse der Funktionalmatrix $D\psi^{(i)}$ der Restabbildung $\psi^{(i)}$ ist also entscheidend für den Einfluss des bei der Berechnung von $x^{(i)}$ begangenen neuen Rundungsfehlers α_i bzw. E_i.

Beispiel 5: Für die in Beispiel 2 eingeführten Algorithmen zur Berechnung von $y = \varphi(a,b) = a^2 - b^2$ hat man

Algorithmus 1:

$$\begin{aligned} &x = x^{(0)} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad x^{(1)} = \begin{bmatrix} a^2 \\ b \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} a^2 \\ b^2 \end{bmatrix}, \quad x^{(3)} = y = a^2 - b^2, \\ &\psi^{(1)}(u,v) = u - v^2, \quad \psi^{(2)}(u,v) = u - v, \\ &D\varphi(x) = \begin{bmatrix} 2a & -2b \end{bmatrix}, \\ &D\psi^{(1)}(x^{(1)}) = \begin{bmatrix} 1 & -2b \end{bmatrix}, \quad D\psi^{(2)}(x^{(2)}) = \begin{bmatrix} 1 & -1 \end{bmatrix}, \\ &\alpha_1 = \begin{bmatrix} \varepsilon_1 a^2 \\ 0 \end{bmatrix}, \quad E_1 = \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \alpha_2 = \begin{bmatrix} 0 \\ \varepsilon_2 b^2 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}, \\ &\alpha_3 = \varepsilon_3(a^2 - b^2), \quad |\varepsilon_i| \le \mathrm{eps} \quad \text{für} \quad i = 1,2,3 \end{aligned}$$

wegen

$$\begin{bmatrix} a \times^* a \\ b \end{bmatrix} - \begin{bmatrix} a^2 \\ b \end{bmatrix} = \begin{bmatrix} \varepsilon_1 a^2 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} a^2 \\ b \times^* b \end{bmatrix} - \begin{bmatrix} a^2 \\ b^2 \end{bmatrix} = \begin{bmatrix} 0 \\ \varepsilon_2 b^2 \end{bmatrix}$$

und $(a^2 -^* b^2) - (a^2 - b^2) = \varepsilon_3(a^2 - b^2)$. Für (1.3.13) erhält man mit $\Delta x = \begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix}$

$$\Delta y \doteq 2a\Delta a - 2b\Delta b + a^2\varepsilon_1 - b^2\varepsilon_2 + (a^2 - b^2)\varepsilon_3. \tag{1.3.14}$$

Für *Algorithmus 2* erhält man analog

$$x = x^{(0)} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad x^{(1)} = \begin{bmatrix} a \\ b \\ a+b \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} a+b \\ a-b \end{bmatrix}, \quad x^{(3)} = y = a^2 - b^2,$$

$$\psi^{(1)}(a,b,u) := u(a-b), \quad \psi^{(2)}(u,v) = u \cdot v,$$

$$D\varphi(x) = \begin{bmatrix} 2a & -2b \end{bmatrix}, \quad D\psi^{(1)}(x^{(1)}) = \begin{bmatrix} a+b & -a-b & a-b \end{bmatrix},$$

$$D\psi^{(2)}(x^{(2)}) = \begin{bmatrix} a-b & a+b \end{bmatrix},$$

$$\alpha_1 = \begin{bmatrix} 0 \\ 0 \\ \varepsilon_1(a+b) \end{bmatrix}, \quad E_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \varepsilon_1 \end{bmatrix},$$

$$\alpha_2 = \begin{bmatrix} 0 \\ \varepsilon_1(a-b) \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & 0 \\ 0 & \varepsilon_2 \end{bmatrix},$$

$$\alpha_3 = \varepsilon_3(a^2 - b^2), \quad E_3 = \varepsilon_3, \quad |\varepsilon_i| \le \text{eps},$$

und damit aus (1.3.13)

$$\Delta y \doteq 2a\Delta a - 2b\Delta b + (a^2 - b^2)(\varepsilon_1 + \varepsilon_2 + \varepsilon_3). \tag{1.3.15}$$

Wählt man einen anderen Algorithmus, d.h. eine andere Zerlegung von φ in Elementarabbildungen $\varphi^{(1)}$, so ändert sich zwar $D\varphi$ nicht, wohl aber i.a. die Matrizen $D\psi^{(i)}$, die die Fortpflanzung der Rundungsfehler messen, und damit auch der gesamte Einfluss aller Rundungsfehler, der durch

$$D\psi^{(1)}\alpha_1 + \cdots + D\psi^{(r)}\alpha_r + \alpha_{r+1} \tag{1.3.16}$$

gegeben ist.

Man nennt einen Algorithmus *numerisch stabiler* als einen zweiten Algorithmus zur Berechnung von $\varphi(x)$, falls der Gesamteinfluss der Rundungsfehler bei dem ersten Algorithmus kleiner ist als bei dem zweiten.

Beispiel: Der gesamte Rundungsfehlereinfluss von Algorithmus 1 in Beispiel 2 ist wegen (1.3.14)

$$\left|a^2\varepsilon_1 - b^2\varepsilon_2 + (a^2 - b^2)\varepsilon_3\right| \le \left(a^2 + b^2 + |a^2 - b^2|\right) \text{eps}, \tag{1.3.17}$$

der von Algorithmus 2 wegen (1.3.15)

$$\left|(a^2 - b^2)(\varepsilon_1 + \varepsilon_2 + \varepsilon_3)\right| \le 3\,\text{eps}\,|a^2 - b^2|. \tag{1.3.18}$$

Genau für $1/3 \le |a/b|^2 \le 3$ gilt $3|a^2 - b^2| \le a^2 + b^2 + |a^2 - b^2|$: Algorithmus 2 ist deshalb für $1/3 < |a/b|^2 < 3$ numerisch stabiler als Algorithmus 1, für die übrigen a, b ist Algorithmus 1 numerisch stabiler.

Zum Beispiel erhält man für $a := 0.3237$, $b := 0.3134$ bei 4-stelliger Rechnung ($t = 4$):

Algorithmus 1:	$a \times^* a = 0.1048,\ b \times^* b = 0.9882_{10} - 1,$
	$(a \times^* a) -^* (b \times^* b) = 0.6580_{10} - 2,$
Algorithmus 2:	$a +^* b = 0.6371,\ a -^* b = 0.1030_{10} - 1,$
	$(a \times^* b) \times^* (a -^* b) = 0.6562_{10} - 2,$
Exaktes Resultat:	$a^2 - b^2 = 0.656213_{10} - 2.$

In der Fehlerfortpflanzungsformel (1.3.13) gilt unabhängig von dem benutzten Algorithmus zur Berechnung von $y = \varphi(x)$ für den letzten Term die Abschätzung [1]

$$|E_{r+1} y| \le \text{eps}\, |y|.$$

Bei jedem Algorithmus muss man mindestens mit einem Fehler Δy dieser Grössenordnung eps$|y|$ rechnen. Weiter beachte man, dass bei Verwendung einer t-stelligen Maschine allein durch das Runden der Eingabedaten $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^T$ auf t Stellen ein Eingangsfehler $\Delta^{(0)} x$ mit

$$|\Delta^{(0)} x| \le \text{eps}\, |x|$$

entsteht, es sei denn, die Eingabedaten, etwa nicht zu grosse ganze Zahlen, sind bereits Maschinenzahlen und damit exakt darstellbar. Aus diesem Grunde muss man bei jedem Algorithmus zur Berechnung von $y = \varphi(x)$ ebenfalls mindestens mit einem weiteren Fehler der Grössenordnung $|D\varphi(x)|\,|x|$ eps rechnen, so dass insgesamt bei *jedem* Algorithmus mit einem Fehler der Grösse

$$\Delta^{(0)} y := \text{eps}\, \big(|D\varphi(x)|\,|x| + |y|\big) \tag{1.3.19}$$

zu rechnen ist. $\Delta^{(0)} y$ heisst der *unvermeidbare Fehler* von y. Da man ohnehin mit einem Fehler dieser Grössenordnung zu rechnen hat, wäre es unbillig, von einem Rundungsfehler α_i bzw. E_i eines Algorithmus zu verlangen, dass sein Beitrag zum Gesamtfehler wesentlich kleiner als $\Delta^{(0)}$ ist. Wir nennen deshalb einen Rundungsfehler α_i bzw. E_i eines Algorithmus *harmlos*, falls in (1.3.13) sein Beitrag zum Gesamtfehler Δy höchstens dieselbe Grössenordnung wie der unvermeidbare Fehler $\Delta^{(0)} y$ (1.3.19) besitzt:

$$|D\psi^{(i)}(x^{(i)})\alpha_i| = |D\psi^{(i)}(x^{(i)}) E_i x^{(i)}| \approx \Delta^{(0)} y.$$

Wenn alle Rundungsfehler eines Algorithmus harmlos sind, heisst der Algorithmus *gutartig* (vgl. Bauer (1965)). Eine der wichtigsten Aufgaben der numerischen Mathematik ist es, gutartige Algorithmen zu finden.

[1] Betragszeichen für Vektoren, Matrizen etc. sind komponentenweise zu verstehen, z.B. $|y| = \begin{bmatrix} |y_1| & \cdots & |y_m| \end{bmatrix}^T$.

Beispiel 6: Beide Algorithmen von Beispiel 2 sind für alle a und b gutartig. Für den unvermeidbaren Fehler $\Delta^{(0)}y$ hat man nämlich

$$\Delta^{(0)}y = \mathrm{eps}\Big(\begin{bmatrix} 2|a| & 2|b| \end{bmatrix} \begin{bmatrix} |a| \\ |b| \end{bmatrix} + |a^2 - b^2| \Big) = \mathrm{eps}\big(2(a^2+b^2) + |a^2 - b^2|\big).$$

Ein Vergleich mit (1.3.17), (1.3.18) zeigt sogar, dass der *gesamte* Rundungsfehlereinfluss beider Algorithmen dem Betrage nach höchstens gleich $\Delta^{(0)}y$ ist.

Einfache weitere Beispiele für die eingeführten Begriffe findet man im nächsten Abschnitt.

Der Begriff der Gutartigkeit von Algorithmen ist zentral; die Terminologie ist hier aber, insbesondere bei der Definition der numerischen Stabilität, fliessend und oft wenig präzise. So nennt man häufig einen Algorithmus schlechthin *numerisch stabil*, wenn er sich wie ein gutartiger Algorithmus verhält. In diesem Buch verwenden wir den Begriff „numerisch stabil" nur zum Vergleich zweier Algorithmen, unabhängig davon ob diese Algorithmen gutartig sind oder nicht.

Wir wollen noch auf eine typische Situation hinweisen, in der sich der Begriff der Gutartigkeit bewährt. Häufig verwendet man *zweistufige Algorithmen*

$$x \to z = \tilde{\varphi}(x) \to y = \tilde{\psi}(z) = \varphi(x)$$

zur Berechnung einer Funktion $y = \varphi(x)$, $\varphi = \tilde{\psi} \circ \tilde{\varphi}$, die man durch Hintereinanderschalten zweier anderer Algorithmen (Unterprogramme !) zur Berechnung von $z = \tilde{\varphi}(x)$ und von $y = \tilde{\psi}(z)$ erhält. Wir wollen zeigen, dass dies nicht notwendig zu einem gutartigen Algorithmus zur Berechnung von $y = \varphi(x)$ führt, selbst wenn man gutartige Algorithmen zur Berechnung von $z = \tilde{\varphi}(x)$ und $y = \tilde{\psi}(z)$ benutzt. In der Tat erhält man bei der Berechnung von z in Gleitpunktarithmetik bestenfalls einen Näherungswert $\tilde{z}$ mit einem absoluten Fehler $\alpha_z = \tilde{z} - z$ des Betrages

$$|\alpha_z| \approx |z| \,\mathrm{eps}.$$

Allein dieser Fehler steuert zum absoluten Fehler von y den Anteil $D\tilde{\psi}(z)\alpha_z$ bei, dessen Betrag die Grösse $|D\tilde{\psi}(z)||z|\mathrm{eps}$ erreichen kann. Der zweistufige Algorithmus wird sicher *nicht* gutartig sein, falls dieser Betrag sehr viel grösser als der unvermeidliche Fehler (1.3.19) $\Delta^{(0)}y$ von y ist,

$$|D\tilde{\psi}(z)|\,|z|\,\mathrm{eps} \gg \Delta^{(0)}y = \big(|D\varphi(x)||x| + |y|\big)\,\mathrm{eps}.$$

Es kommt hier nur auf die Grössenordnung von $|D\tilde{\psi}(z)|\,|z|$ im Vergleich zu $|D\varphi(x)|\,|x|$ an, d.h. (vgl. (1.3.4)) auf die *Kondition* der Abbildung $\tilde{\psi}$ im Vergleich mit der Kondition der Abbildung φ: Der Stufenalgorithmus ist nicht gutartig, falls die Kondition von $\tilde{\psi}$ sehr viel schlechter als die von φ ist. Da die Abschätzung nur die *Abbildungen* φ und $\tilde{\psi}$ involviert, gelten diese Aussagen unabhängig davon, welche Teilalgorithmen man zur Berechnung von

$z = \tilde{\varphi}(x)$ und von $y = \tilde{\psi}(z)$ verwendet. Man sollte deshalb unbedingt Stufenalgorithmen vermeiden, in denen Zwischenresultate z berechnet werden, von denen das Endresultat y empfindlicher abhängt als von den Eingangsdaten x.

Ein Mittel dazu, für einen Algorithmus die Gutartigkeit nachzuweisen, ist die sog. *backward-analysis*, die von Wilkinson zur Analyse der Algorithmen der linearen Algebra benutzt wurde. Bei dieser Technik wird zunächst versucht zu zeigen, dass das Resultat $\tilde{y} = y + \Delta y$ eines Algorithmus zur Berechnung von $y = \varphi(x)$, das man in Gleitpunktarithmetik bei der Auswertung von $\varphi(x)$ erhält, sich in der Form $\tilde{y} = \varphi(x + \Delta x)$ schreiben lässt und damit als Resultat einer exakten Rechnung mit abgeänderten Eingabedaten $x + \Delta x$ interpretiert werden kann. Kann man zusätzlich zeigen, dass Δx höchstens dieselbe Grössenordnung wie $|\Delta^{(0)} x| \leq \mathrm{eps}\,|x|$ hat, so ist damit die Gutartigkeit des Algorithmus gezeigt.

Zum Schluss soll noch kurz anhand eines Beispiels dargestellt werden, wie man das Fehlerverhalten eines Algorithmus statt durch (1.3.13) auch mit Hilfe eines *Graphen* veranschaulichen kann, wie von Bauer (1974) vorgeschlagen wurde. Die Algorithmen 1 und 2 von Beispiel 2 werden so durch die Graphen von Fig. 1 beschrieben.

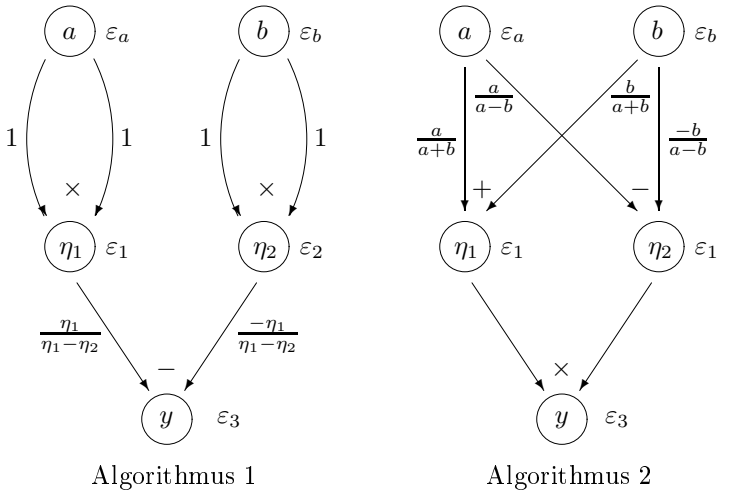


Fig. 1. Graphendarstellung der Fehlerfortpflanzung von Algorithmen

Die Knoten entsprechen den Zwischenresultaten. Knoten i wird mit Knoten j durch eine gerichtete Kante verbunden, falls das Zwischenresultat von Knoten i direkt in das Zwischenresultat von Knoten j eingeht. Dementsprechend entsteht an jedem Knoten ein neuer relativer Rundungsfehler, der neben den betreffenden Knoten geschrieben wird. Die Zahlen an den Kanten geben die Verstärkungsfaktoren für die relativen Fehler an. Z.B. kann man

vom Graphen des Algorithmus 1 folgende Beziehungen ablesen:

$$\varepsilon_{\eta_1} = 1 \cdot \varepsilon_a + 1 \cdot \varepsilon_a + \varepsilon_1, \quad \varepsilon_{\eta_2} = 1 \cdot \varepsilon_b + 1 \cdot \varepsilon_b + \varepsilon_2,$$
$$\varepsilon_y = \frac{\eta_1}{\eta_1 - \eta_2} \cdot \varepsilon_{\eta_1} - \frac{\eta_2}{\eta_1 - \eta_2} \cdot \varepsilon_{\eta_2} + \varepsilon_3.$$

Will man den Faktor wissen, mit dem multipliziert der Rundungsfehler von Knoten i zum relativen Fehler des Zwischenresultats von Knoten j beiträgt, hat man für jeden gerichteten Weg von i nach j die Kantenfaktoren zu multiplizieren und diese Produkte zu summieren. Z.B. besagt der Graph von Algorithmus 2, dass zum Fehler ε_y der Eingangsfehler ε_a den Beitrag

$$\left(\frac{a}{a+b} \cdot 1 + \frac{a}{a-b} \cdot 1\right) \cdot \varepsilon_a$$

liefert.

1.4 Beispiele

Beispiel 1: Dieses Beispiel schliesst sich an Beispiel 4 des letzten Abschnittes an. Es sei $p > 0$, $q > 0$, $p \gg q$ gegeben. Man bestimme die Wurzel

$$y = -p + \sqrt{p^2 + q}$$

kleinsten Betrages der quadratischen Gleichung

$$y^2 + 2py - q = 0.$$

Eingangsdaten: p, q. Resultat: $y = \varphi(p, q) = -p + \sqrt{p^2 + q}$.

Im letzten Abschnitt sahen wir, dass das Problem $\varphi(p, q)$ zu berechnen, für $p > 0$, $q > 0$ gut konditioniert ist, und die relativen Eingangsfehler ε_p, ε_q zum relativen Fehler des Resultates y folgenden Beitrag leisten:

$$\frac{-p}{\sqrt{p^2+q}}\varepsilon_p + \frac{q}{2y\sqrt{p^2+q}}\varepsilon_q = \frac{-p}{\sqrt{p^2+q}}\varepsilon_p + \frac{p+\sqrt{p^2+q}}{2\sqrt{p^2+q}}\varepsilon_q.$$

Wegen

$$\left|\frac{p}{\sqrt{p^2+q}}\right| \leq 1, \quad \left|\frac{p+\sqrt{p^2+q}}{2\sqrt{p^2+q}}\right| \leq 1$$

genügt der unvermeidbare Fehler $\Delta^{(0)}y$ des Resultates $y = \varphi(p, q)$ den Ungleichungen

$$\text{eps} \leq \varepsilon_y^{(0)} := \frac{\Delta^{(0)}y}{y} \leq 3\,\text{eps}.$$

Wir untersuchen zwei Algorithmen zur Berechnung von $y = \varphi(p, q)$.

Algorithmus 1:

$$s := p^2,$$
$$t := s + q,$$
$$u := \sqrt{t},$$
$$y := -p + u.$$

Wie man sieht, tritt wegen $p \gg q$ bei $y := -p + u$ Auslöschung auf, und es steht zu erwarten, dass der Rundungsfehler

$$\Delta u := \varepsilon \cdot \sqrt{t} = \varepsilon \cdot \sqrt{p^2 + q},$$

der bei der Gleitpunktberechnung der Quadratwurzel

$$\mathrm{gl}(\sqrt{t}) = \sqrt{t} \cdot (1 + \varepsilon), \quad |\varepsilon| \leq \mathrm{eps},$$

neu entsteht, verstärkt wird. In der Tat verursacht dieser Fehler folgenden relativen Fehler von y:

$$\varepsilon_y \doteq \frac{1}{y}\Delta u = \frac{\sqrt{p^2+q}}{-p+\sqrt{p^2+q}} \cdot \varepsilon = \frac{1}{q}(p\sqrt{p^2+q} + p^2 + q)\varepsilon = k \cdot \varepsilon.$$

Wegen $p, q > 0$ gilt für den Verstärkungsfaktor k die Abschätzung

$$k > \frac{2p^2}{q} > 0.$$

Er kann für $p \gg q$ sehr gross sein und zeigt, dass der vorgeschlagene Algorithmus nicht gutartig ist, weil allein der Einfluss des Rundungsfehlers ε, der bei der Berechnung von $\sqrt{p^2+q}$ anfällt, viel grösser als der relative unvermeidbare Fehler $\varepsilon_y^{(0)}$ ist.

Algorithmus 2:

$$\begin{aligned} s &:= p^2, \\ t &:= s + q, \\ u &:= \sqrt{t}, \\ v &:= p + u, \\ y &:= q/v. \end{aligned}$$

Bei diesem Algorithmus tritt bei der Berechnung von v keine Auslöschung auf. Der bei der Berechnung von $u = \sqrt{t}$ entstehende Rundungsfehler $\Delta u = \varepsilon\sqrt{p^2+q}$ steuert, durch die entsprechende Restabbildung $\psi(u)$

$$u \to p + u \to \frac{q}{p+u} =: \psi(u)$$

verstärkt, folgenden Betrag zum relativen Fehler ε_y von y bei:

$$\begin{aligned} \frac{1}{y}\frac{\partial \varphi}{\partial u}\Delta u &= \frac{-q}{y(p+u)^2} \cdot \Delta u \\ &= \frac{-q\sqrt{p^2+q}}{\left(-p+\sqrt{p^2+q}\right)\left(p+\sqrt{p^2+q}\right)^2} \cdot \varepsilon \\ &= -\frac{\sqrt{p^2+q}}{p+\sqrt{p^2+q}} \cdot \varepsilon = k \cdot \varepsilon. \end{aligned}$$

Der Verstärkungsfaktor k ist klein, $|k| < 1$, Algorithmus 2 ist gutartig.

Das folgende *Zahlenbeispiel* illustriert den Unterschied zwischen den Algorithmen 1 und 2 (wie bei weiteren numerischen Beispielen mit 40 Binär-Mantissenstellen

gerechnet, was etwa 12 Dezimalstellen entspricht; unrichtige Ziffern sind unterstrichen):

$p = 1000,\ q = 0.018\,000\,000\,081$

Resultat für y nach *Algorithmus 1:*	$0.900\,0\underline{30}\,\underline{136}\,\underline{108}_{10}-5$
Resultat für y nach *Algorithmus 2:*	$0.899\,999\,999\,9\underline{99}_{10}-5$
Exakter Wert von y:	$0.900\,000\,000\,000_{10}-5$

Beispiel 2: Mit Hilfe der Formel

$$\cos(m+1)x = 2\cos x\,\cos mx - \cos(m-1)x, \quad m = 1, 2, \ldots, k-1,$$

kann man $\cos kx$ für festes x und ganzzahliges k rekursiv berechnen. Man hat dazu nur einmal mit $c := \cos x$ eine trigonometrische Funktion auszuwerten. Sei nun $|x| \neq 0$ eine kleine Zahl. Bei der Berechnung von c entstehe ein kleiner Rundungsfehler,

$$\tilde{c} = (1+\varepsilon)\cos x, \quad |\varepsilon| \leq \text{eps}.$$

Wie wirkt sich dieser Rundungsfehler auf $\cos kx$ aus?
Antwort: $\cos kx$ hängt in folgender Weise von c ab:

$$\cos kx = \cos(k \arccos c) =: f(c).$$

Wegen

$$\frac{df}{dc} = \frac{k \sin kx}{\sin x}$$

bewirkt der absolute Fehler $\varepsilon \cos x$ von c in erster Näherung einen absoluten Fehler

$$\Delta \cos kx \doteq \varepsilon \frac{\cos x}{\sin x}\, k \sin kx = \varepsilon\, k\ \text{ctg}\, x\ \sin kx \tag{1.4.1}$$

in $\cos kx$.

Der unvermeidbare Fehler $\Delta^{(0)} c_k$ (1.3.19) des Resultats $c_k := \cos kx$ ist dagegen

$$\Delta^{(0)} c_k = \text{eps}\,\big(k|x \sin kx| + |\cos kx|\big).$$

Ein Vergleich mit (1.4.1) zeigt, dass $\Delta \cos kx$ für kleines $|x|$ wesentlich grösser als $\Delta^{(0)} c_k$ sein kann: Der Algorithmus ist für solche x nicht gutartig.

Beispiel 3: Für eine gegebene Zahl x und eine „grosse“ ganze Zahl k sollen $\cos kx$ und $\sin kx$ rekursiv mit Hilfe der Formeln

$$\begin{aligned} \cos mx &:= \cos x\,\cos(m-1)x - \sin x\,\sin(m-1)x, \\ \sin mx &:= \sin x\,\cos(m-1)x + \cos x\,\sin(m-1)x, \quad m = 1, 2, \ldots, k, \end{aligned}$$

berechnet werden.

Wie wirken sich kleine Fehler $\varepsilon_c \cos x$, $\varepsilon_s \sin x$ bei der Berechnung von $\cos x$, $\sin x$ auf die Endresultate $\cos kx$, $\sin kx$ aus? Setzt man zur Abkürzung $c_m := \cos mx$, $s_m := \sin mx$, $c := \cos x$, $s := \sin x$, so ist in Matrixschreibweise mit der unitären Matrix

$$U := \begin{bmatrix} c & -s \\ s & c \end{bmatrix},$$

die einer Drehung um den Winkel x entspricht,

$$\begin{bmatrix} c_m \\ s_m \end{bmatrix} = U \begin{bmatrix} c_{m-1} \\ s_{m-1} \end{bmatrix}, \quad m = 1, \ldots, k,$$

und daher

$$\begin{bmatrix} c_k \\ s_k \end{bmatrix} = U^k \begin{bmatrix} c_0 \\ s_0 \end{bmatrix} = U^k \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Nun ist

$$\frac{\partial U}{\partial c} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \frac{\partial U}{\partial s} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} =: A$$

und daher

$$\frac{\partial}{\partial c} U^k = kU^{k-1},$$
$$\frac{\partial}{\partial s} U^k = AU^{k-1} + UAU^{k-2} + \cdots + U^{k-1}A = kAU^{k-1},$$

weil A mit U vertauschbar ist. Da U einer Drehung im $\mathbb{R}^2$ um den Winkel x entspricht, ist schliesslich

$$\frac{\partial}{\partial c} U^k = k \begin{bmatrix} \cos(k-1)x & -\sin(k-1)x \\ \sin(k-1)x & \cos(k-1)x \end{bmatrix},$$
$$\frac{\partial}{\partial s} U^k = k \begin{bmatrix} -\sin(k-1)x & -\cos(k-1)x \\ \cos(k-1)x & -\sin(k-1)x \end{bmatrix}.$$

Die relativen Fehler ε_c, ε_s von $c = \cos x$ und $s = \sin x$ bewirken daher bei $\cos kx$, $\sin kx$ folgende absolute Fehler:

$$\begin{aligned} \begin{bmatrix} \Delta c_k \\ \Delta s_k \end{bmatrix} &\doteq \begin{bmatrix} \frac{\partial}{\partial c} U^k \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \varepsilon_c \cos x + \begin{bmatrix} \frac{\partial}{\partial s} U^k \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \varepsilon_s \sin x \\ &= \varepsilon_c k \cos x \begin{bmatrix} \cos(k-1)x \\ \sin(k-1)x \end{bmatrix} + \varepsilon_s k \sin x \begin{bmatrix} -\sin(k-1)x \\ \cos(k-1)x \end{bmatrix}. \end{aligned} \tag{1.4.2}$$

Die unvermeidbaren Fehler (1.3.19) $\Delta^{(0)} c_k$ und $\Delta^{(0)} s_k$ von $c_k = \cos kx$ und $s_k = \sin kx$ sind dagegen

$$\begin{aligned} \Delta^{(0)} c_k &= \big(k|x \sin kx| + |\cos kx|\big)\,\mathrm{eps}, \\ \Delta^{(0)} s_k &= \big(k|x \cos kx| + |\sin kx|\big)\,\mathrm{eps}. \end{aligned} \tag{1.4.3}$$

Ein Vergleich mit (1.4.2) zeigt, dass für grosses k und $|k \cdot x| \approx 1$ im Gegensatz zum Rundungsfehler ε_s der Einfluss von ε_c auf die Resultate wesentlich grösser als die unvermeidbaren Fehler sind. Der Algorithmus ist nicht gutartig, obwohl er als Algorithmus zur Berechnung von c_k allein numerisch stabiler als der Algorithmus von Beispiel 2 ist.

Beispiel 4: Die numerische Stabilität des Algorithmus in Beispiel 3 zur Berechnung von $c_m = \cos mx$, $s_m = \sin mx$, $m = 1, 2, \ldots$, für kleines $|x|$ lässt sich weiter verbessern: Es ist

$$\cos(m+1)x = \cos x \cos mx - \sin x \sin mx,$$
$$\sin(m+1)x = \sin x \cos mx + \cos x \sin mx,$$

und es gilt daher für die Differenzen dc_{m+1} und ds_{m+1} aufeinanderfolgender cos- und sin-Werte

$$\begin{aligned} dc_{m+1} &:= \cos(m+1)x - \cos mx \\ &= 2(\cos x - 1)\cos mx - \sin x \sin mx - \cos x \cos mx + \cos mx \\ &= -4\left(\sin^2 \frac{x}{2}\right)\cos mx + \big(\cos mx - \cos(m-1)x\big), \\ ds_{m+1} &:= \sin(m+1)x - \sin mx \\ &= 2(\cos x - 1)\sin mx + \sin x \cos mx - \cos x \sin mx + \sin mx \\ &= -4\left(\sin^2 \frac{x}{2}\right)\sin mx + \big(\sin mx - \sin(m-1)x\big). \end{aligned}$$

Dies führt zu folgendem Algorithmus zur Berechnung von c_k, s_k für $x > 0$:

$$\begin{aligned} dc_1 &:= -2\sin^2 \frac{x}{2}, \quad t := 2\,dc_1, \\ ds_1 &:= \sqrt{-dc_1(2 + dc_1)}, \\ s_0 &:= 0, \quad c_0 := 1, \end{aligned}$$

und für $m := 1, 2, \ldots, k$:

$$\begin{aligned} c_m &:= c_{m-1} + dc_m, \quad dc_{m+1} := t \cdot c_m + dc_m, \\ s_m &:= s_{m-1} + ds_m, \quad ds_{m+1} := t \cdot s_m + ds_m. \end{aligned}$$

Für die Fehleranalyse beachten wir, dass die Werte c_k und s_k Funktionen von $s = \sin(x/2)$ sind:

$$\begin{aligned} c_k &= \cos(2k \arcsin s) =: \varphi_1(s), \\ s_k &= \sin(2k \arcsin s) =: \varphi_2(s). \end{aligned}$$

Ein Fehler $\Delta_s = \varepsilon_s \sin(x/2)$ bei der Berechnung von s bewirkt daher in erster Näherung folgende Fehler in c_k, s_k:

$$\begin{aligned} \Delta c_k &\doteq \frac{\partial \varphi_1}{\partial_s}\varepsilon_s \sin\frac{x}{2} = \varepsilon_s \frac{-2k \sin kx}{\cos \frac{x}{2}} \sin\frac{x}{2} = -2k \operatorname{tg}\frac{x}{2} \sin kx \cdot \varepsilon_s, \\ \Delta s_k &\doteq \frac{\partial \varphi_2}{\partial_s}\varepsilon_s \sin\frac{x}{2} = 2k \operatorname{tg}\frac{x}{2} \cos kx \cdot \varepsilon_s. \end{aligned}$$

Ein Vergleich mit den unvermeidbaren Fehlern (1.4.3) zeigt, dass der Fehler ε_s für kleines $|x|$ harmlos ist.

Zahlenbeispiel: $x = 0.001$, $k = 1000$

Algorithmus	Resultat für $\cos kx$	relativer Fehler
Beispiel 2	0.540 302 <u>121 124</u>	$-0.34_{10}-6$
Beispiel 3	0.540 302 305 <u>776</u>	$-0.17_{10}-9$
Beispiel 4	0.540 302 305 86<u>5</u>	$-0.58_{10}-11$
Exakter Wert:	0.540 302 305 868 140 ...	

Beispiel 5: Dieses Beispiel nimmt einige Resultate vorweg, die bei der Analyse der Algorithmen zur Gleichungsauflösung in Abschnitt 4.5 nützlich sind. Gegeben seien die Zahlen $c,\ a_1,\ldots,a_n,\ b_1,\ldots,b_{n-1}$ mit $a_n \neq 0$. Gesucht ist die Lösung β_n der linearen Gleichung

$$c - a_1b_1 - \cdots - a_{n-1}b_{n-1} - a_n\beta_n = 0. \tag{1.4.4}$$

Bei Gleitpunktrechnung erhält man statt der exakten Lösung β_n einen Näherungswert

$$b_n = \mathrm{gl}\Big(\frac{c - a_1b_1 - \cdots - a_{n-1}b_{n-1}}{a_n}\Big) \tag{1.4.5}$$

auf die folgende Weise:

$$\begin{aligned} &s_0 := c;\\ &\text{für}\quad j := 1,2,\ldots,n-1\\ &\qquad s_j := \mathrm{gl}(s_{j-1} - a_jb_j) = (s_{j-1} - a_jb_j(1+\mu_j))(1+\alpha_j),\\ &b_n := \mathrm{gl}(s_{n-1}/a_n) = (1+\delta)s_{n-1}/a_n, \end{aligned} \tag{1.4.6}$$

mit $|\mu_j|,\ |a_j|,\ |\delta| \leq \mathrm{eps}$. Häufig ist in den Anwendungen $a_n = 1$, in diesem Fall ist $\delta = 0$, wegen $b_n := s_{n-1}$. Wir wollen zwei verschiedene nützliche Abschätzungen für das *Residuum*

$$r := c - a_1b_1 - \cdots - a_nb_n$$

geben. Durch Summation der aus (1.4.6) folgenden Gleichungen

$$\begin{aligned} s_0 - c &= 0,\\ s_j - (s_{j-1} - a_jb_j) &= s_j - \Big(\frac{s_j}{1+\alpha_j} + a_jb_j\mu_j\Big)\\ &= s_j\frac{\alpha_j}{1+\alpha_j} - a_jb_j\mu_j, \quad j = 1,2,\ldots,n-1,\\ a_nb_n - s_{n-1} &= \delta s_{n-1}, \end{aligned}$$

erhält man

$$r = c - \sum_{i=1}^{n} a_ib_i = \sum_{j=1}^{n-1}\Big(-s_j\frac{\alpha_j}{1+\alpha_j} + a_jb_j\mu_j\Big) - \delta s_{n-1}$$

und damit die erste der gesuchten Abschätzungen

$$\begin{aligned} |r| &\leq \frac{\mathrm{eps}}{1-\mathrm{eps}}\Big(\delta'\cdot|s_{n-1}| + \sum_{j=1}^{n-1}\big(|s_j| + |a_jb_j|\big)\Big),\\ \delta' &:= \begin{cases} 0 & \text{falls } a_n = 1,\\ 1 & \text{sonst.}\end{cases} \end{aligned} \tag{1.4.7}$$

Die folgende Abschätzung ist gröber als (1.4.7). Aus (1.4.6) folgt

$$b_n = \Big(c\prod_{k=1}^{n-1}(1+\alpha_k) - \sum_{j=1}^{n-1}a_jb_j(1+\mu_j)\prod_{k=j}^{n-1}(1+\alpha_k)\Big)\frac{1+\delta}{a_n}. \tag{1.4.8}$$

Durch Auflösen nach c erhält man

$$c = \sum_{j=1}^{n-1} a_j b_j (1+\mu_j) \prod_{k=1}^{j-1} (1+\alpha_k)^{-1} + a_n b_n (1+\delta)^{-1} \prod_{k=1}^{n-1} (1+\alpha_k)^{-1}. \tag{1.4.9}$$

Durch vollständige Induktion nach m zeigt man nun leicht, dass aus

$$(1+\sigma) = \prod_{k=1}^{m} (1+\sigma_k)^{\pm 1}, \quad |\sigma_k| \le \text{eps}, \quad m \cdot \text{eps} < 1$$

folgt

$$|\sigma| \le \frac{m \cdot \text{eps}}{1 - m \cdot \text{eps}}.$$

Wegen (1.4.9) ergibt dies die Existenz von Zahlen ε_j mit

$$\begin{aligned} c &= \sum_{j=1}^{n-1} a_j b_j (1 + j \cdot \varepsilon_j) + a_n b_n \big(1 + (n-1+\delta')\varepsilon_n\big), \\ |\varepsilon_j| &\le \frac{\text{eps}}{1 - n \cdot \text{eps}}, \quad \delta' := \begin{cases} 0 & \text{falls } a_n = 1, \\ 1 & \text{sonst,} \end{cases} \end{aligned} \tag{1.4.10}$$

so dass für $r = c - a_1 b_1 - a_2 b_2 - \cdots - a_n b_n$ gilt

$$|r| \le \frac{\text{eps}}{1 - n \cdot \text{eps}} \left(\sum_{j=1}^{n-1} j |a_j b_j| + (n-1+\delta') |a_n b_n| \right). \tag{1.4.11}$$

Aus (1.4.8) folgt insbesondere die Gutartigkeit unseres Algorithmus zur Berechnung von β_n. In erster Näherung liefert nämlich z.B. der Rundungsfehler α_m folgenden Beitrag zum absoluten Fehler von β_n

$$\big((c - a_1 b_1 - a_2 b_2 - \cdots - a_m b_m)/a_n\big)\alpha_m.$$

Dieser Beitrag ist höchstens gleich dem Einfluss

$$\left| \frac{c \cdot \varepsilon_c - a_1 b_1 \varepsilon_{a_1} - \cdots - a_m b_m \varepsilon_{\alpha_m}}{a_n} \right| \le \text{eps} \frac{|c| + \sum_{i=1}^{m} |a_i b_i|}{|a_n|},$$

den allein Eingangsfehler ε_c, ε_{a_i} von c bzw. a_i, $i = 1, \ldots, m$, mit $|\varepsilon_c|$, $|\varepsilon_{a_i}| \le \text{eps}$ haben. Ähnlich kann man für die übrigen Rundungsfehler μ_k und δ argumentieren.

Gewöhnlich zeigt man die Gutartigkeit dadurch, dass man (1.4.10) im Sinne der „backward-analysis“ interpretiert: Das berechnete b_n ist exakte Lösung der Gleichung

$$c - \bar{a}_1 b_1 - \cdots - \bar{a}_n b_n = 0,$$

deren Koeffizienten

$$\begin{aligned} \bar{a}_j &:= a_j (1 + j \cdot \varepsilon_j), \quad 1 \le j \le n-1, \\ \bar{a}_j &:= a_j \big(1 + (n-1+\delta')\varepsilon_n\big) \end{aligned}$$

gegenüber den a_j nur leicht abgeändert sind.

Bei dieser Art von Analyse hat man die Schwierigkeit, erklären zu müssen, für wie grosse n Fehler der Form $n\varepsilon$, $|\varepsilon| \le \text{eps}$, noch als Fehler von der Grössenordnung der Maschinengenauigkeit eps gelten sollen.

1.5 Statistische Rundungsfehlerabschätzungen

Den Einfluss weniger Rundungsfehler kann man noch mit den Methoden von 1.3 abschätzen. Bei einem typischen Algorithmus ist jedoch die Zahl der arithmetischen Operationen und damit die Zahl der einzelnen Rundungsfehler zu gross, um auf diese Weise den Einfluss aller Rundungsfehler bestimmen zu können.

Grundlage von *statistischen* Rundungsfehlerabschätzungen (siehe Rademacher (1948)), ist die Annahme, dass die relativen Rundungsfehler ε (siehe (1.2.6)), die bei den elementaren arithmetischen Operationen entstehen als *Zufallsvariable* mit Werten im Intervall $[-\mathrm{eps}, \mathrm{eps}]$ aufgefasst werden können. Darüber hinaus nimmt man an, dass diese elementaren Zufallsvariablen ε unabhängig sind, wenn sie zu verschiedenen Elementaroperationen gehören. Mit μ_ε bezeichnen wir den Mittelwert, mit σ_ε die Streuung (σ_ε^2 die Varianz) der Zufallsvariablen ε. Für sie gelten mit dem Erwartungswert-Operator E die aus der mathematischen Statistik bekannten Beziehungen

$$\mu_\varepsilon = E(\varepsilon), \quad \sigma_\varepsilon^2 = E\big(\varepsilon - E(\varepsilon)\big)^2 = E(\varepsilon^2) - \big(E(\varepsilon)\big)^2 = \mu_{\varepsilon^2} - \mu_\varepsilon^2.$$

Unter der weiteren Annahme, dass die Zufallsvariable ε auf dem Intervall $[-\mathrm{eps}, \mathrm{eps}]$ *gleichverteilt* ist, erhält man die expliziten Formeln

$$\mu_\varepsilon = E(\varepsilon) = 0, \quad \sigma_\varepsilon^2 = E(\varepsilon^2) = \frac{1}{2\mathrm{eps}} \int_{-\mathrm{eps}}^{\mathrm{eps}} t^2 dt = \frac{1}{3}\mathrm{eps}^2 =: \bar{\varepsilon}^2. \tag{1.5.1}$$

Eine genauere Untersuchung zeigt jedoch (siehe Sterbenz (1974)), dass die Annahme der Gleichverteilung nicht ganz richtig ist. Man sollte deshalb nicht vergessen, dass man mit dieser Annahme ideale Rundungsfehler modelliert, die das Verhalten der tatsächlichen elementaren Rundungsfehler in Rechnern zwar sehr gut, aber nur näherungsweise beschreiben. Um bessere Ansätze als (1.5.1) für μ_ε und σ_ε^2 zu erhalten, wird man gegebenenfalls auf empirische Untersuchungen zurückgreifen müssen.

Die Resultate x eines Algorithmus werden unter dem Einfluss der Rundungsfehler selbst Zufallsvariable mit Erwartungswert μ_x und Varianz σ_x^2 sein, für die ebenfalls gilt

$$\sigma_x^2 = E\big(x - E(x)\big)^2 = E(x^2) - \big(E(x)\big)^2 = \mu_{x^2} - \mu_x^2.$$

Die Fortpflanzung früherer Rundungsfehler unter den Elementaroperationen werden für beliebige Zufallsvariable x, y und Zahlen $\alpha, \beta \in \mathbb{R}$ durch folgende Regeln beschrieben

$$\begin{aligned} \mu_{\alpha x \pm \beta y} &= E(\alpha x \pm \beta y) = \alpha E(x) \pm \beta E(y) = \alpha \mu_x \pm \beta \mu_y, \\ \sigma_{\alpha x \pm \beta y}^2 &= E\big((\alpha x \pm \beta y)^2\big) - \big(E(\alpha x \pm \beta y)\big)^2 \\ &= \alpha^2 E\big(x - E(x)\big)^2 + \beta^2 E\big(y - E(y)\big)^2 = \alpha^2 \sigma_x^2 + \beta^2 \sigma_y^2. \end{aligned} \tag{1.5.2}$$

Die erste dieser Formeln gilt für beliebige Zufallsvariable wegen der Linearität des Erwartungswert-Operators E. Die zweite gilt nur für *unabhängige* Zufallsvariable x, y, da hier die Beziehung $E(x \cdot y) = E(x) \cdot E(y)$ zum Beweis benötigt wird. Ebenso erhält man für unabhängige Zufallsvariable x, y

$$\begin{aligned} \mu_{x\times y} &= E(x \times y) = E(x)E(y) = \mu_x \mu_y, \\ \sigma^2_{x\times y} &= E\big((x \times y) - E(x)E(y)\big)^2 = \mu_{x^2}\mu_{y^2} - \mu_x^2\mu_y^2 \\ &= \sigma_x^2\sigma_y^2 + \mu_x^2\sigma_y^2 + \mu_y^2\sigma_x^2. \end{aligned} \tag{1.5.3}$$

Beispiel: Für die Berechnung von $y = a^2 - b^2$ (siehe Beispiel 2 in 1.3) findet man unter Verwendung von (1.5.1), $E(a) = a$, $\sigma_a^2 = 0$, $E(b) = b$, $\sigma_b^2 = 0$ und von (1.5.2), (1.5.3)

$$\begin{gathered} \eta_1 = a^2(1+\varepsilon_1), \quad E(\eta_1) = a^2, \quad \sigma^2_{\eta_1} = a^4\bar{\varepsilon}^2, \\ \eta_2 = b^2(1+\varepsilon_2), \quad E(\eta_2) = b^2, \quad \sigma^2_{\eta_2} = b^4\bar{\varepsilon}^2, \\ y = (\eta_1 - \eta_2)(1+\varepsilon_3), \quad E(y) = E(\eta_1 - \eta_2)E(1+\varepsilon_3) = a^2 - b^2, \end{gathered}$$

(η_1, η_2, ε_3 werden hier als unabhängige Zufallsvariable angenommen),

$$\begin{aligned} \sigma_y^2 &= \sigma^2_{\eta_1-\eta_2}\sigma^2_{1+\varepsilon_3} + \mu^2_{\eta_1-\eta_2}\sigma^2_{1+\varepsilon_3} + \mu^2_{1+\varepsilon_3}\sigma^2_{\eta_1-\eta_2} \\ &= (\sigma^2_{\eta_1} + \sigma^2_{\eta_2})\bar{\varepsilon}^2 + (a^2-b^2)^2\bar{\varepsilon}^2 + 1(\sigma^2_{\eta_1} + \sigma^2_{\eta_2}) \\ &= (a^4 + b^4)\bar{\varepsilon}^4 + \big((a^2-b^2)^2 + a^4 + b^4\big)\bar{\varepsilon}^2. \end{aligned}$$

Bei Vernachlässigung von $\bar{\varepsilon}^4$ gegenüber $\bar{\varepsilon}^2$ erhält man in erster Ordnung die Formel

$$\sigma_y^2 \doteq \big((a^2-b^2)^2 + a^4 + b^4\big)\bar{\varepsilon}^2.$$

Für $a := 0.3237$, $b = 0.3134$, $\mathrm{eps} = 5 \times 10^{-4}$ (siehe Beispiel 5 in 1.3) finden wir die Streuung

$$\sigma_y \doteq 0.144\bar{\varepsilon} = 0.000\ 0415,$$

die die gleiche Grössenordnung wie der wahre Fehler $\Delta y = 0.000\ 01787$ besitzt, den wir bei 4-stelliger Rechnung erhalten. Man vergleiche σ_y mit der Fehlerschranke 0.000 10478, die von (1.3.17) geliefert wird.

Wir bezeichnen mit $M(x)$ die Menge aller Grössen, die bei einem gegebenen Algorithmus direkt oder indirekt in die Berechnung von x eingegangen sind. Falls $M(x) \cap M(y) \neq \emptyset$, werden die Zufallsvariablen x und y im allgemeinen abhängig sein. Die exakte statistische Rundungsfehlerabschätzung wird bei einem Algorithmus aber extrem schwierig, wenn man Abhängigkeiten berücksichtigen will. Sie motiviert die folgenden weiteren Voraussetzungen, die die Analyse erheblich vereinfachen:

(1.5.4)

a) *Die Operanden jeder arithmetischen Operation sind unabhängige Zufallsvariable.*

b) *Bei der Berechnung der Varianzen werden nur die Terme niedrigster Ordnung in* eps *berücksichtigt.*

c) *Alle Varianzen sind so klein, dass für jede arithmetische Operation* $*$ *in erster Näherung gilt*

$$E(x * y) \doteq E(x) * E(y) = \mu_x * \mu_y.$$

Wenn man zusätzlich die Erwartungswerte μ_x von x durch x ersetzt und *relative Varianzen* $\varepsilon_x^2 := \sigma_x^2/\mu_x^2 \approx \sigma_x^2/x^2$ einführt, erhält man aus (1.5.2), (1.5.3) (vergleiche (1.2.6), (1.3.5)) die Formeln

$$\begin{aligned} z = \mathrm{gl}(x \pm y): &\quad \varepsilon_z^2 \doteq \left(\frac{x}{z}\right)^2 \varepsilon_x^2 + \left(\frac{y}{z}\right)^2 \varepsilon_y^2 + \bar{\varepsilon}^2, \\ z = \mathrm{gl}(x \pm y): &\quad \varepsilon_z^2 \doteq \varepsilon_x^2 + \varepsilon_y^2 + \bar{\varepsilon}^2, \\ z = \mathrm{gl}(x/y): &\quad \varepsilon_z^2 \doteq \varepsilon_x^2 + \varepsilon_y^2 + \bar{\varepsilon}^2. \end{aligned} \tag{1.5.5}$$

Man beachte aber, dass diese Relationen nur unter der Annahme (1.5.4), insbesondere (1.5.4)a) gültig sind.

Es ist möglich, diese Formeln im Verlauf eines Algorithmus mitzuberechnen. Man erhält so Schätzwerte für die relativen Varianzen der Endresultate. Dies führt zu modifizierten Algorithmen, die mit Paaren (x, ε_x^2) von Grössen arbeiten, die über elementare Operationen entsprechend (1.5.5) oder analogen Formeln zusammenhängen. Fehlerschranken für die Endresultate r erhält man dann aus den relativen Varianzen ε_r^2 mit Hilfe der weiteren Annahme, dass die Endvariablen r *normalverteilt* sind. Diese Annahme ist in der Regel gerechtfertigt, weil die Verteilung von r und von Zwischenresultaten x umso besser durch eine Normalverteilung approximiert wird, je mehr unabhängige Rundungsfehler in ihre Berechnung eingeflossen sind. Unter dieser Normalitätsannahme folgt jedenfalls, dass der relative Fehler eines Resultats r mit der Wahrscheinlichkeit 0.9 dem Betrag nach höchstens gleich $2\varepsilon_r$ ist.

Übungsaufgaben zu Kapitel 1

1. Man zeige, dass bei t-stelliger dezimaler Gleitpunktrechnung analog zu (1.2.2) gilt
$$\mathrm{rd}(a) = \frac{a}{1+\varepsilon} \quad \text{mit} \quad |\varepsilon| \le 5 \cdot 10^{-t}.$$
(Daher gilt neben (1.2.6) auch $\mathrm{gl}(a * b) = (a * b)/(1+\varepsilon)$ mit $|\varepsilon| \le 5 \cdot 10^{-t}$ für alle arithmetischen Operationen $* = +, -, /$.)
2. a, b, c seien Festkommazahlen mit N Dezimalstellen hinter dem Komma und $0 < a, b, c < 1$. Das Produkt $a * b$ sei wie folgt erklärt: Zu $a \cdot b$ wird $10^{-N}/2$ addiert, danach werden $(n+1)$-te und die folgenden Dezimalstellen weggelassen.
   a) Man gebe eine Schranke für $\left|(a * b) * c - abc\right|$ an.
   b) Um wieviele Einheiten der N-ten Stelle können sich $(a * b) * c$ und $a * (b * c)$ unterscheiden?

3. Bei der Gleitpunktberechnung von $\sum_{i=1}^{n} a_j$ kann ein beliebig grosser relativer Fehler auftreten, der jedoch beschränkt bleibt, falls alle a_j das gleiche Vorzeichen haben. Man leite unter Vernachlässigung von Gliedern höherer Ordnung eine grobe Schranke für ihn her.
4. Die folgenden Ausdrücke sollen so umgeformt werden, dass ihre Auswertung gutartig wird:

$$\frac{1}{1+2x} - \frac{1-x}{1+x} \quad \text{für} \quad |x| \ll 1,$$
$$\sqrt{x+\frac{1}{x}} - \sqrt{x-\frac{1}{x}} \quad \text{für} \quad x \gg 1,$$
$$\frac{1-\cos x}{x} \quad \text{für} \quad x \neq 0, \quad |x| \ll 1.$$

5. Für die Auswertung der Funktion $\arcsin y$ in t-stelliger dezimaler Gleitpunktarithmetik stehe ein Programm zur Verfügung, das für $|y| \leq 1$ den Funktionswert mit einem relativen Fehler ε liefert, wobei $|\varepsilon| \leq 5 \cdot 10^{-t}$ ist. Entsprechend der Identität
$$\arctan x = \arcsin \frac{x}{\sqrt{1+x^2}}$$
könnte dieses Programm auch zur Auswertung von $\arctan x$ eingesetzt werden. Man untersuche durch eine Abschätzung des relativen Fehlers, für welche Werte x diese Methode arctan zu berechnen, gutartig ist.
6. Zu gegebenem z kann $\tan z/2$ mittels der Formel
$$\tan \frac{z}{2} = \pm \left(\frac{1-\cos z}{1+\cos z} \right)^{1/2}$$
berechnet werden. Ist die Auswertung dieser Formel für $z \approx 0$ und $z \approx \pi/2$ gutartig? Gegebenenfalls gebe man gutartige Methoden an.
7. Es soll ein Verfahren zur Berechnung der Funktion
$$f(\varphi, k_c) := \frac{1}{\sqrt{\cos^2 \varphi + k_c^2 \sin^2 \varphi}}$$
für $0 \leq \varphi \leq \pi/2$, $0 < k_c \leq 1$ angegeben werden. Die Methode
$$k^2 := 1 - k_c^2, \quad f(\varphi, k_c) := \frac{1}{\sqrt{1 - k^2 \sin^2 \varphi}},$$
vermeidet die Auswertung von $\cos \varphi$ und ist somit schneller. Man vergleiche diese Methode mit der direkten Auswertung des gegebenen Ausdrucks für $f(\varphi, k_c)$ im Hinblick auf die Gutartigkeit.
8. Für die lineare Funktion $f(x) := a + b \cdot x$ mit $a \neq 0$, $b \neq 0$ soll die erste Ableitung $f'(0) = b$ mit Hilfe der Differenzenformel
$$D_h f(0) = \frac{f(h) - f(-h)}{2h}$$
in dualer Gleitpunktarithmetik berechnet werden. Dabei seien a und b gegebene Gleitpunktzahlen, h sei eine Potenz von 2, so dass die Multiplikation mit h und die Division durch $2h$ exakt ausgeführt werden. Man gebe eine Schranke für den relativen Fehler von $D_h f(0)$ an. Wie verhält sich diese Schranke für $h \to 0$?

9. Die Quadratwurzel $\pm(u+iv)$ einer komplexen Zahl $x+iy$ mit $y \neq 0$ kann nach den Formeln

$$u = \pm\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}$$
$$v = \frac{y}{2u}$$

berechnet werden. Man vergleiche die Fälle $x \geq 0$ und $x < 0$ im Hinblick auf Gutartigkeit und ändere die Formeln nötigenfalls ab, um sie gutartig zu machen.

10. Zu den Messwerten $x_1, \ldots, x_n$ soll ein Schätzwert S^2 für die Varianz berechnet werden. Welche der Formeln

$$S^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right),$$
$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2 \quad mit \quad \bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i$$

ist numerisch stabiler?

11. Die Koeffizienten a_r, b_r $r = 0, \ldots, n$, seien für festes x wie folgt miteinander verknüpft:

$$(*) \qquad \begin{aligned} b_n &:= a_n, \\ b_r &:= x b_{r+1} + a_r, \quad r = n-1, n-2, \ldots, 0. \end{aligned}$$

a) Man zeige, dass für die Polynome

$$A(z) := \sum_{r=0}^{n} a_r z^r, \quad B(z) := \sum_{r=1}^{n} b_r z^{r-1}$$

gilt:

$$A(z) = (z-x)\cdot B(z) + b_0.$$

b) $A(x) = b_0$ soll nach der Rekursion $(*)$ für festes x in Gleitpunktarithmetik berechnet werden, das Resultat sei b_0'. Man zeige unter Verwendung der Formeln (vgl. Aufg. 1)

$$\mathrm{gl}(u+v) = \frac{u+v}{1+\sigma}, \quad |\sigma| \leq \mathrm{eps},$$
$$\mathrm{gl}(u\cdot v) = \frac{u\cdot v}{1+\pi}, \quad |\pi| \leq \mathrm{eps},$$

dass für b_0' die Abschätzung

$$|A(x) - b_0'| \leq \frac{\mathrm{eps}}{1-\mathrm{eps}}\left(2e_0 - |b_0'|\right)$$

gilt, wo e_0 durch folgende Rekursion gewonnen wird:

$$\begin{aligned} e_n &:= |a_n|/2, \\ e_r &:= |x|a_{r+1} + |b_r'|, \quad r = n-1, n-2, \ldots, 0. \end{aligned}$$

[*Hinweis:* Mit

$$b'_n := a_n,$$

und

$$p_r := \mathrm{gl}(xb'_{r+1}) = \frac{xb'_{r+1}}{1+\pi_{r+1}},$$
$$b_r := \mathrm{gl}(p_r + a_r) = \frac{p_r + a_r}{1+\sigma_r} = xb'_{r+1} + a_r + \delta_r$$

zeige man zunächst für $r = n-1, n-2, \ldots, 0$

$$\delta_r = -xb'_{r+1}\frac{\pi_{r+1}}{1+\pi_{r+1}} - \sigma_r b'_r,$$

zeige dann, $b'_0 = \sum_{k=0}^{n}(a_k + \delta_k)x^k$, $\delta_n := 0$, und schätze $\sum_0^n |\delta_k||x|^k$ ab.]

Literatur zu Kapitel 1

Bauer, F.L. (1974): Computational graphs and rounding error. SIAM J. Numer. Anal. **11**, 87–96.

Bauer, F.L., Heinhold, J., Samelson, K., Sauer, R. (1965): *Moderne Rechenanlagen.* Stuttgart: Teubner.

Edelman, A. (1997): The mathematics of the Pentium division bug. SIAM Rev. **39**, 54–67.

Goldberg, D. (1991): What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**, 5–48.

Goldberg, D. (2003): Computer arithmetic. Appendix H in: Hennessy, J.L., Patterson, D.A. *Computer Architecture: A Quantitative Approach*, 3rd Edition. San Mateo, CA: Morgan Kaufmann.

Henrici, P. (1963): *Error Propagation for Difference Methods.* New York: Wiley.

Higham, N.J. (2002): *Accuracy and Stability of Numerical Algorithms*, 2nd Edition. Philadelphia: SIAM.

Knuth, D.E. (1997): *The Art of Computer Programming. Vol. 2. Seminumerical Algorithms*, 3rd Edition. Reading, MA: Addison-Wesley.

Neumann, J. von, Goldstein, H.H. (1947): Numerical inverting of matrices. Bull. Amer. Math. Soc. **53**, 1021-1099.

Overton, M.L. (2001): *Numerical Computing with IEEE Floating Point Arithmetic.* Philadelphia: SIAM.

Rademacher, H.A. (1948): On the accumulation of errors in processes of integration on high-speed calculating machines. Proceedings of a symposium on large-scale digital calculating machinery. Ann. Comput. Labor. Harvard Univ. **16**, 176–185.

Scarborough, J.B. (1966): *Numerical Mathematical Analysis*, 6th Edition. Baltimore: The Johns Hopkins Press.

Sterbenz, P.H. (1974): *Floating Point Computation.* Englewood Cliffs, NJ: Prentice Hall.

Wilkinson, J.H. (1960): Error analysis of floating-point computation. Numer. Math. **2**, 219–340.

Wilkinson, J.H. (1969): *Rundungsfehler*. Berlin-Heidelberg: Springer.

Wilkinson, J.H. (1994): *Rounding Errors in Algebraic Processes*. Mineola, NY: Dover.

Wilkinson, J.H. (1988): *The Algebraic Eigenvalue Problem*, Paperback Edition. Oxford: Oxford University Press.