Kapitel 7
Algorithmus und Stabilität

7.1 Gleitkommaarithmetik

7.1.1 Definition und algebraische Eigenschaften

Die Menge $\mathbb{R}$ der reellen Zahlen, versehen mit Addition und Multiplikation, ist ein *Körper*. Die reellen Zahlen genügen also den folgenden Körperaxiomen (siehe z.B. Bosch [4, Kap. 2.1]).

(K1) $x+(y+z)=(x+y)+z$ (Assoziativität der Addition)
(K2) $x+y=y+x$ (Kommutativität der Addition)
(K3) Es gibt ein Element $0\in\mathbb{R}$ mit $0+x=x$ für alle $x\in\mathbb{R}$.
. (neutrales Element der Addition)
(K4) Zu jedem $x\in\mathbb{R}$ existiert ein Element $-x\in\mathbb{R}$ mit $(-x)+x=0$.
. (Inverses Element der Addition)
(K5) $x(yz)=(xy)z$ (Assoziativität der Multiplikation)
(K6) $xy=yx$ (Kommutativität der Multiplikation)
(K7) Es gibt ein Element $1\in\mathbb{R}$ mit $1x=x$ für alle $x\in\mathbb{R}$.
. (Neutrales Element der Multiplikation)
(K8) Zu jedem $x\in\mathbb{R}\setminus\{0\}$ existiert ein Element $x^{-1}\in\mathbb{R}$ mit $x^{-1}x=1$.
. (Inverses Element der Multiplikation)
(K9) x(y+z)=xy +xz (Distributivität)
(K10) $1\neq 0$ (Nullteilerfreiheit)

Diese zehn Eigenschaften bilden die Grundlage für alle äquivalenten algebraischen Umformungen, die wir aus der Schule gewohnt sind. Beispielsweise gewinnt man die binomische Formel $(a+b)^2=a^2+2ab+b^2$ aus (K6) und dreimaliger Anwendung von (K9). Wir wollen untersuchen, in welchem Umfang sich die Körpereigenschaften (K1) – (K10) auf die Teilmenge $\mathbb{G}\subset\mathbb{R}$ der Gleitkommazahlen vererben.

Gleich am Anfang stoßen wir auf eine grundsätzliche Schwierigkeit: Die Grundrechenarten können aus $\mathbb{G}$ herausführen. So ist offenbar $\tilde{x}=1234\in\mathbb{G}(10,4)$ und $\tilde{y}=12,34\in\mathbb{G}(10,4)$, für die Summe gilt aber $\tilde{x}+\tilde{y}=1246,34\notin\mathbb{G}(10,4)$ und auch die Multiplikation liefert $15227,56\in\mathbb{G}(10,4)$. Im allgemeinen gilt also

$$\tilde{x},\tilde{y}\in\mathbb{G}\quad\not\Longrightarrow\quad\tilde{x}+\tilde{y},\ \tilde{x}\cdot\tilde{y}\in\mathbb{G}\,.$$

Um die Grundrechenarten auf $\mathbb{G}$ zu beschränken, müssen wir nach jeder Rechenoperation das Ergebnis auf $\mathbb{G}$ zurückprojezieren, also *runden*.

Definition 7.1 (Gleitkommaarithmetik). Auf den Gleitkommazahlen $\mathbb{G}$ definieren wir die *gerundeten Grundrechenarten* $\tilde{+}$, $\tilde{-}$, $\tilde{*}$ und $\tilde{:}$ für alle $\tilde{x},\tilde{y}\in\mathbb{G}$ durch

$$\tilde{x}\tilde{+}\tilde{y}=\mathrm{rd}(\tilde{x}+\tilde{y}),\quad \tilde{x}\tilde{-}\tilde{y}=\mathrm{rd}(\tilde{x}-\tilde{y}),\quad \tilde{x}\tilde{*}\tilde{y}=\mathrm{rd}(\tilde{x}\tilde{y}),\quad \tilde{x}\tilde{:}\tilde{y}=\mathrm{rd}(\tilde{x}:\tilde{y}),\ \tilde{y}\neq 0.$$

Wir wollen sehen, ob die Gleitkommazahlen, versehen mit gerundeter Addition und Multiplikation, die Axiome (K1)-(K10) erfüllen. Die meisten Axiome postulieren die *Gleichheit* gewisser Ausdrücke. Wie wir am Ende des vorigen Abschnitts gesehen haben, sind Gleichheitsabfragen von Gleitkommazahlen aber im allgemeinen sinnlos. Das lässt nichts Gutes erwarten.

Und schon mit (K1) gibt es Schwierigkeiten. In $\mathbb{G}(10,4)$ gilt nämlich

$$(1234\tilde{+}0,4)\tilde{+}0,4 = 1234\tilde{+}0,4 = 1234 \neq 1235 = 1234\tilde{+}0,8 = 1234\tilde{+}(0,4\tilde{+}0,4) ,$$

und dieses Beispiel lässt sich leicht auf beliebige Basis q und Mantissenlänge ℓ übertragen. Die gerundete Addition ist also *nicht assoziativ*. Unser Beispiel zeigt übrigens auch, daß bei Gleitkommazahlen aus $\tilde{a}+\tilde{x}=\tilde{a}$ nicht notwendig $\tilde{x}=0$ folgen muß.

Nun zur Abwechslung eine gute Nachricht: Die gerundete Addition ist *kommutativ*. Dies folgt aus

$$\tilde{x}\tilde{+}\tilde{y} = \mathrm{rd}(a_x q^{e_x} + a_y q^{e_y}) = \mathrm{rd}(a_x + a_y q^{e_y-e_x}) q^{e_x} = \mathrm{rd}(a_y + a_x q^{e_x-e_y}) q^{e_y} = \tilde{y}\tilde{+}\tilde{x} . \tag{7.1}$$

Wegen $0 \in \mathbb{G}$ und $\tilde{x} \in \mathbb{G} \Rightarrow -\tilde{x} \in \mathbb{G}$ sind (K3) und (K4) klar.

Wir kommen zur Multiplikation. In $\mathbb{G}(10,4)$ gilt offenbar

$$\begin{aligned}(1234\tilde{*}0,9996)\tilde{*}0,9999 &= 1234\tilde{*}0,9999 = 1234 \\ &\neq 1233 = 1234\tilde{*}0,9995 = 1234\tilde{*}(0,9996\tilde{*}0,9999) .\end{aligned}$$

Wieder lässt sich dieses Beispiel auf $\mathbb{G}$ verallgemeinern. Die Multiplikation ist also *nicht assoziativ*. Wieder sieht man auch, daß sich $\tilde{a}\tilde{x}=\tilde{a}$, $\tilde{a}\neq 0$, nicht äquivalent in $\tilde{x}=1$ umformen lässt.

Wie schon die Addition, so ist auch die gerundete Multiplikation *kommutativ*. Der Nachweis erfolgt analog zu (7.1).

Wegen $1 \in \mathbb{G}$ ist (K7) kein Problem, aber schon (K8) macht wieder Schwierigkeiten. In $\mathbb{G}(10,4)$ gilt nämlich beispielsweise

$$0,1111\tilde{*}9 = 0,9999 < 1 < 1,001 = 0,1112\tilde{*}9 .$$

Da die gerundete Multiplikation mit 9 streng monoton ist, also $9\tilde{y} < 9\tilde{z}$ aus $\tilde{y} < \tilde{z}$ folgt, und da es kein $\tilde{y} \in \mathbb{G}(10,4)$ mit $0,1111 < \tilde{y} < 0,1112$ gibt, folgt daraus, daß $\tilde{x}=9$ kein inverses Element $\tilde{x}^{-1} \in \mathbb{G}(10,4)$ hat. Die gerundete Multiplikation ist also im allgemeinen *nicht invertierbar*.

An dieser Stelle wird schon niemand mehr erwarten, daß das Distributivgesetz gilt. Tatsächlich findet man schnell Gegenbeispiele. Die gerundete Addition und Multiplikation sind *nicht distributiv*[1].

Wir haben uns also wohl oder übel darauf einzustellen, daß beim gerundeten Rechnen mit Gleitkommazahlen elementare Rechenregeln verletzt werden. Umformungen, die bei exaktem Rechnen in $\mathbb{R}$ zu äquivalenten Ausdrücken führen, liefern beim gerundeten Rechnen verschiedene Resultate. Beispielsweise gilt die binomische Formel für Gleitkommazahlen im allgemeinen nicht. Sind $\tilde{a},\tilde{b} \in \mathbb{G}$ Approximationen von $a,b \in \mathbb{R}$, so sind die Ausdrücke $(\tilde{a}+\tilde{b})^2$ und $\tilde{a}^2+2\tilde{a}\tilde{b}+\tilde{b}^2$ im allgemeinen *verschiedene* Approximationen von $(a+b)^2 = a^2+2ab+b^2$. Welche ist die bessere? Spielt der Unterschied überhaupt jemals eine Rolle? Zumindest die zweite Frage lässt sich leicht mit ja beantworten wie das folgende Beispiel zeigt.

7.1.2 Ein Polynom-Desaster

Gegeben seien das Polynom

$$f(x) = x^3 + 12a^2x - 6ax^2 - 8a^3 , \tag{7.2}$$

der Parameter $a = 4\,999\,999$ und das Argument

$$x_0 = 10\,000\,000 .$$

Gesucht ist der Funktionswert $f(x_0)$.

Um diese Aufgabe zu lösen, liegt es nahe, die Berechnungsvorschrift (7.2) direkt zu implementieren.

Algorithmus 7.2 (Auswertung von $f = x^3 + 12a^2x - 6ax^2 - 8a^3$).

[1] Vor dem Hintergrund all dieser Negativresultate stellt sich die Frage, warum zur Approximation von $\mathbb{R}$ nicht besser die rationalen Zahlen herangezogen werden. Schließlich ist $\mathbb{Q}$, versehen mit Addition und Multiplikation, nach wie vor ein Körper, in dem man, anders als in $\mathbb{G}$, die gewohnten algebraischen Umformungen vornehmen kann. Der entscheidende Grund liegt in dem hohen Zeitaufwand für das Rechnen mit Brüchen (vgl. Abschnitt 3.4). Angesichts exorbitant hoher Rechenzeiten wird die rationale Arithmetik derzeit nur in einzelnen Spezialanwendungen eingesetzt.

```
a =  4999999;
x = 10000000;
f = x^3 + 12*a^2*x - 6*a*x^2 - 8*a^3
```

Algorithmus 7.2 liefert:

```
f =
    393216
```

also das Ergebnis $f(x_0) = 393216$. Nun könnten wir uns zufrieden zurücklehnen.

Besser nicht. Unter Verwendung der binomischen Formel

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

lässt sich (7.2) äquivalent umschreiben in

$$f(x) = (x-2a)^3 \; . \tag{7.3}$$

Die andere Darstellung erzeugt einen anderen Algorithmus.

Algorithmus 7.3 (Auswertung von $f = (x-2a)^3$).

```
a =  4999999;
x = 10000000;
f = (x - 2*a)^3
```

Dieser Algorithmus liefert:

```
f =
    8
```

und wegen $x - 2a = 2$ ist das offenbar auch das richtige Ergebnis.

Wir wollen verstehen, weshalb der erste Algorithmus 7.2 derart versagt. An der Kondition kann es nicht liegen, denn Eingabefehler treten nicht auf.

7.1.3 Praktische Realisierung

Die `IEEE 754`-Norm fordert bei allen Operationen exaktes Runden, d.h. die Ergebnisse der Operationen müssen dieselben sein, die man bei zunächst exakter Ausführung der Rechenoperation und anschließendem Runden erhielte. In der Praxis wird dies erreicht, indem ein interner Zwischenspeicher mit erhöhter Genauigkeit für die Speicherung der Zwischenergebnisse verwendet wird. Aus diesen Ergebnissen mit erhöhter Genauigkeit wird am Ende der Rechnung auf die übliche (einfache bzw. doppelte) Genauigkeit gerundet.

7.2 Stabilität geschachtelter Funktionsauswertungen

Nachdem wir im vorigen Kapitel 6 das *Problem* der Auswertung einer Funktion $f : I \to \mathbb{R}$ an einer Stelle $x_0 \in I \subset \mathbb{R}$ untersucht haben, wollen wir uns nun mit Algorithmen zu dessen Lösung beschäftigen. Unter einem *Algorithmus* wollen wir dabei eine Zerlegung

$$f(x_0) = g_n \circ g_{n-1} \circ \cdots \circ g_1(x_0) \tag{7.4}$$

der Funktion f in elementare Funktionen g_i, $i = 1, \ldots, n$, verstehen. Dabei bedeutet $\circ$ die Hintereinanderschaltung, also $g_i \circ g_{i-1}(y) = g_i(g_{i-1}(y))$. Verschiedene Zerlegungen charakterisieren verschiedene Algorithmen. Beispielsweise ist

$$f(x) = ax + b = g_2 \circ g_1(x) \; , \qquad g_1(y) = ay \; , \quad g_2(y) = y + b \; . \tag{7.5}$$

Unter der Voraussetzung $a \neq 0$ kann man f aber auch in der folgenden Form schreiben

$$f(x) = ax + b = h_2 \circ h_1(x)\,, \qquad h_1(y) = y + \frac{b}{a}\,, \quad h_2(y) = ay\,. \tag{7.6}$$

Solange alle Elementarfunktionen exakt ausgewertet werden, liefern beide Zerlegungen dasselbe Ergebnis. Das ist für alle Algorithmen der Form (7.4) nicht anders. Oft können aber nur Approximationen $\tilde{g}_i(y)$ von g_i praktisch realisiert werden. Beispielsweise führt die Berücksichtigung von Rundungsfehlern auf die Approximation

$$\tilde{g}_i(y) = \mathrm{rd}\,(g_i(y)) = g_i(y)(1+\varepsilon_i)\,, \qquad |\varepsilon_i| \le eps\,. \tag{7.7}$$

Dann erhält man auch anstelle der Funktion f nur eine Approximation, nämlich

$$\tilde{f}(\varepsilon, x) = \tilde{g}_n \circ \tilde{g}_{n-1} \circ \cdots \circ \tilde{g}_1(x)\,. \tag{7.8}$$

Die Approximation $\tilde{f}(\varepsilon,x)$ hängt offenbar von den Störungen ε_i aller Elementarfunktionen g_i ab, die in dem Vektor $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$ aufgehoben sind.

Verschiedene Algorithmen führen auf *verschiedene Approximationen*. Beispielsweise führt Algorithmus (7.5) bei gerundeter Arithmetik auf die Approximation

$$\tilde{f}_g(\varepsilon, x) = (ax(1+\varepsilon_1) + b)\,(1+\varepsilon_2)\,,$$

während (7.6) eine andere Approximation, nämlich

$$\tilde{f}_h(\varepsilon, x) = (ax+b)(1+\varepsilon_1)(1+\varepsilon_2)\,,$$

ergibt. Welche ist besser? Dazu schauen wir uns den jeweiligen *Auswertungsfehler* $f(x_0) - \tilde{f}(\varepsilon, x_0)$ an. Zu dessen Quantifizierung verwenden wir das *relative Fehlerkonzept*[2]. Wir erhalten einerseits

$$\frac{|f(x_0) - \tilde{f}_g(\varepsilon, x_0)|}{|f(x_0)|} = \frac{|ax_0\varepsilon_1 + (ax_0+b)\varepsilon_2|}{|ax_0+b|} + o(\max\{|\varepsilon_1|, |\varepsilon_2|\})$$

und andererseits

$$\frac{|f(x_0) - \tilde{f}_h(\varepsilon, x_0)|}{|f(x_0)|} = |1 - (1+\varepsilon_1)(1+\varepsilon_2)| = |\varepsilon_1 + \varepsilon_2| + o(\max\{|\varepsilon_1|, |\varepsilon_2|\})\,.$$

Die Version (7.6) erlaubt also insbesondere im Falle $\frac{|ax_0|}{|ax_0+b|} \gg 1$ (Auslöschung!) deutlich bessere Fehlerschranken als (7.5).

Vor diesem Hintergrund wollen wir jetzt die Stabilität von Algorithmen zur Funktionsauswertung definieren. Dabei betrachten wir nur Störungen der Form (7.7), also Rundungsfehler. Zu deren Quantifizierung verwenden wir die Abkürzung

$$\|\varepsilon\| = \max_{i=1,\ldots,n} |\varepsilon_i|\,, \qquad \varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)\,.$$

Bekanntlich gilt $\|\varepsilon\| \le eps$ nach Satz 5.5 in Abschnitt 5.3.

Definition 7.4 (Relative Stabilität). Es sei $f(x_0) \neq 0$. Dann ist die *relative Stabilität* σ_{rel} von Algorithmus (7.4) gegenüber Rundungsfehlern (7.7) die kleinste Zahl σ_{rel} mit der Eigenschaft

$$\frac{|f(x_0) - \tilde{f}(\varepsilon, x_0)|}{|f(x_0)|} \le \sigma_{\mathrm{rel}}\|\varepsilon\| + o(\|\varepsilon\|) \tag{7.9}$$

für genügend kleine Störungen ε. Liegt (7.9) für keine reelle Zahl σ_{rel} vor, so wird $\sigma_{\mathrm{rel}} = \infty$ gesetzt.

Die Einschränkung auf „genügend kleine Störungen“ bedeutet, daß es ein $\rho > 0$ geben muß, so daß (7.9) für alle ε mit $\|\varepsilon\| \le \rho$ richtig ist. Es kann sein, daß es kein $\rho > 0$ gibt, so daß $\tilde{f}(\cdot, x_0)$ für alle ε mit $\|\varepsilon\| \le \rho$ auch nur auswertbar ist. Dann ist $\sigma_{\mathrm{rel}} = \infty$.

Die Stabilität $\sigma_{\mathrm{rel}} = \sigma_g(x_0)$ hängt nur von der Zerlegung (7.4) in Elementarfunktionen $g = (g_1, \ldots, g_n)$ und dem Argument x_0 ab.

[2] Die Betrachtung von Rundungsfehlern (7.7) ist in natürlicher Weise mit dem relativen Fehlerkonzept verbunden (vgl. Satz 5.5 in Abschnitt 5.3).

Die Stabilität ist eine Eigenschaft des Algorithmus.

Analog zu (6.10) ist Definition 7.4 gleichbedeutend mit

$$\sigma_{\text{rel}} = \frac{1}{|f(x_0)|} \limsup_{\|\varepsilon\|\to 0} \frac{|f(x_0) - \tilde{f}(\varepsilon, x_0)|}{\|\varepsilon\|} .$$

Allgemein gilt

$$\sigma_{\text{rel}} \geq 1 , \tag{7.10}$$

denn bei Wahl von $\varepsilon_1 = \cdots = \varepsilon_{n-1} = 0$ ist offenbar

$$\frac{|f(x_0) - \tilde{f}(\varepsilon, x_0)|}{|f(x_0)|} = \frac{|f(x_0) - (1+\varepsilon_n) f(x_0)|}{|f(x_0)|} = |\varepsilon_n| = \|\varepsilon\| .$$

Beispielsweise hat der *triviale Algorithmus*

$$f(x_0) = g_1(x_0)$$

wegen

$$\frac{|f(x_0) - \tilde{f}(\varepsilon, x_0)|}{|f(x_0)|} = \frac{|g_1(x_0) - (1+\varepsilon) g_1(x_0)|}{|g_1(x_0)|} = |\varepsilon|$$

die Stabilität $\sigma_{\text{rel}} = 1$.

Die Stabilität σ_g und σ_h der Algorithmen (7.5) und (7.6) ist gegeben durch

$$\sigma_g = 1 + \frac{|ax_0|}{|ax_0 + b|} , \qquad \sigma_h = 2 .$$

Im allgemeinen hat man es mit gestörten Eingabedaten $\tilde{x}_0$ zu tun. Die Stabilität von Algorithmus (7.4) ist dann $\sigma_{\text{rel}}(\tilde{x}_0)$. Eingabefehler und Auswertungsfehler bewirken zusammen den *Gesamtfehler*

$$\frac{|f(x_0) - \tilde{f}(\varepsilon, \tilde{x}_0)|}{|f(x_0)|} .$$

Den Gesamtfehler können wir mit Hilfe der Kondition κ_{rel} des Problems (6.8) und der Stabilität $\sigma_{\text{rel}}(\tilde{x}_0)$ kontrollieren.

Satz 7.5. *Es sei $x_0 \neq 0$ und $f(x_0) \neq 0$. Dann genügt der Gesamtfehler der Abschätzung*

$$\frac{|f(x_0) - \tilde{f}(\varepsilon, \tilde{x}_0)|}{|f(x_0)|} \leq \kappa_{\text{rel}} \frac{|x_0 - \tilde{x}_0|}{|x_0|} + \sigma_{\text{rel}}(\tilde{x}_0)\|\varepsilon\| + o(|x_0 - \tilde{x}_0| + \|\varepsilon\|) . \tag{7.11}$$

Beweis. Unter Verwendung der Definitionen von relativer Kondition und Stabilität berechnet man

$$\begin{aligned}
\frac{|f(x_0) - \tilde{f}(\varepsilon, \tilde{x}_0)|}{|f(x_0)|} &\leq \frac{|f(x_0) - f(\tilde{x}_0)|}{|f(x_0)|} + \frac{|f(\tilde{x}_0) - \tilde{f}(\varepsilon, \tilde{x}_0)|}{|f(\tilde{x}_0)|} \frac{|f(\tilde{x}_0)|}{|f(x_0)|} \\
&\leq \kappa_{\text{rel}} \frac{|x_0 - \tilde{x}_0|}{|x_0|} + \sigma_{\text{rel}}(\tilde{x}_0)\|\varepsilon\| \left(1 + \frac{|f(x_0) - f(\tilde{x}_0)|}{|f(x_0)|}\right) \\
&\quad + o(|x_0 - \tilde{x}_0|) + o(\|\varepsilon\|) \\
&\leq \kappa_{\text{rel}} \frac{|x_0 - \tilde{x}_0|}{|x_0|} + \sigma_{\text{rel}}(\tilde{x}_0)\|\varepsilon\| + o(|x_0 - \tilde{x}_0| + \|\varepsilon\|) .
\end{aligned}$$

□

Die Summe aus Eingabefehler, verstärkt durch die Kondition, und Auswertungsfehler, verstärkt durch die Stabilität ist also eine obere Schranke für den Gesamtfehler.

Wir kommen nun zu *Stabilitätsabschätzungen*.

Satz 7.6. *Es seien $h : I \mapsto I_g \subset \mathbb{R}$ und $g : I_g \mapsto \mathbb{R}$ zwei Funktionen und*

$$h(x_0) = h_n \circ \cdots \circ h_1(x_0) \tag{7.12}$$

ein Algorithmus zur Auswertung von $h(x_0)$. Bezeichnet κ_g die Kondition der Auswertung von g an der Stelle $y = h(x_0)$ und σ_h die Stabilität von Algorithmus (7.12)*, so gilt für die Stabilität σ_{rel} von*

$$f(x_0) = g \circ h_n \circ \cdots \circ h_1(x_0) \tag{7.13}$$

die Abschätzung

$$\sigma_{\text{rel}} \leq \kappa_g \sigma_h + 1 \,. \tag{7.14}$$

Beweis. Berücksichtigung der Auswertungsfehler ergibt

$$\tilde{f}(\varepsilon, x_0) = (1+\varepsilon_g) g(\tilde{h}(\varepsilon_h, x_0))$$

mit $\varepsilon = (\varepsilon_h, \varepsilon_g)$, $\varepsilon_h = (\varepsilon_1, \ldots, \varepsilon_n)$ und $\varepsilon_g \in \mathbb{R}$. Aus den Definitionen von Kondition und Stabilität erhält man mit der Dreiecksungleichung

$$\begin{aligned}
\frac{|f(x_0) - \tilde{f}(\varepsilon, x_0)|}{|f(x_0)|} &\leq \frac{|g(h(x_0)) - g(\tilde{h}(\varepsilon_h, x_0))|}{|g(h(x_0))|} + |\varepsilon_g| \frac{|g(\tilde{h}(\varepsilon_h, x_0)|}{|g(h(x_0))|} \\
&\leq \kappa_g \frac{|h(x_0) - \tilde{h}(\varepsilon_h, x_0)|}{|h(x_0)|} + o\left(\frac{|h(x_0) - \tilde{h}(\varepsilon_h, x_0)|}{|h(x_0)|} \right) + |\varepsilon_g| \\
&\qquad + |\varepsilon_g| \tfrac{|g(h(x_0)) - g(\tilde{h}(\varepsilon_h, x_0)|}{|g(h(x_0))|} \\
&\leq \kappa_g \sigma_h \|\varepsilon_h\| + |\varepsilon_g| + o(\|\varepsilon\|) \\
&\leq (\kappa_g \sigma_h + 1) \|\varepsilon\| + o(\|\varepsilon\|)
\end{aligned}$$

und damit die Behauptung. □

Durch induktive Anwendung von Satz 7.6 wollen wir nun die Stabilität der Approximation (7.8) durch die Kondition der elementaren Funktionen g_i abschätzen. Als Vorbereitung notieren wir folgendes Lemma.

Lemma 7.7. *Gegeben sei die inhomogene Differenzengleichungen erster Ordnung*

$$x_i = \alpha_i x_{i-1} + \beta_i \,, \quad i = 1, 2, \ldots \,, \qquad x_0 = 0 \,, \tag{7.15}$$

mit nicht-negativen Koeffizienten $\alpha_i, \beta_i \in \mathbb{R}$, $i = 1, 2 \ldots$. Dann genügen alle Folgen e_i mit der Eigenschaft

$$e_i \leq \alpha_i e_{i-1} + \beta_i \,, \quad i = 1, 2 \ldots \,, \qquad e_0 = 0 \,, \tag{7.16}$$

der Abschätzung[3]

$$e_i \leq x_i = \sum_{j=1}^{i} K_{ij} \beta_j \,, \quad K_{ij} = \prod_{k=j+1}^{i} \alpha_k \,, \qquad i = 1, 2, \ldots \,. \tag{7.17}$$

Beweis. Der Beweis erfolgt durch vollständige Induktion. Für $i = 1$ bestätigt man die Gültigkeit von (7.17) durch Einsetzen. Ist (7.17) für ein $i \geq 1$ richtig, so folgt

$$x_{i+1} = \alpha_{i+1} \sum_{j=1}^{i} K_{ij} \beta_j + \beta_{i+1} = \sum_{j=1}^{i+1} K_{i+1,j} \beta_j$$

durch Einsetzen in (7.15). Schließlich erhält man aus (7.16) und α_{i+1}, $\beta_{i+1} \geq 0$ sofort

$$e_{i+1} \leq \alpha_{i+1} e_i + \beta_{i+1} \leq \alpha_{i+1} x_i + \beta_{i+1} = x_{i+1}$$

und damit die Behauptung. □

[3] Man beachte $\prod_{k=2}^{1} \alpha_k = 1$.

Satz 7.8. *Es bezeichne κ_i die Kondition der Auswertung der elementaren, reellwertigen Funktion g_i an der Stelle y_{i-1}, und es sei $y_i = g_{i-1}(y_{i-1})$, $i = 2,\dots,n$, $y_1 = x_0$. Dann gilt*

$$\sigma_{\text{rel}} \leq \sum_{j=1}^{n} \prod_{i=j+1}^{n} \kappa_i = 1 + \kappa_n(1 + \kappa_{n-1}(1 + \dots \kappa_3(1 + \kappa_2)\dots)) \,. \tag{7.18}$$

Beweis. Setzt man

$$f_i(x_0) = g_i \circ \dots \circ g_1(x_0) \,, \qquad i = 1,\dots,n, \tag{7.19}$$

so ist offenbar $f_i(x_0) = g_i \circ f_{i-1}(x_0)$ und $f_n(x_0) = f(x_0)$. Anwendung von Satz 7.6 ergibt für die Stabilität σ_i von Algorithmus (7.19) die Ungleichungsrekursion

$$\sigma_i \leq \kappa_i \sigma_{i-1} + 1 \,, \qquad i = 2,\dots,n \,. \tag{7.20}$$

Der triviale Algorithmus $f_1(x_0) = g_1(x_0)$ hat die Stabilität $\sigma_1 = 1$. Setzt man $\sigma_0 = 0$, so ist die Rekursion (7.20) also auch für $i = 1$ richtig. Mit $\alpha_i = \kappa_i$ und $\beta_i = 1$, folgt die Abschätzung (7.18) nun unmittelbar aus Lemma 7.7. □

Sind alle Elementarfunktionen g_i *gut konditioniert*, also κ_i klein, so ist nach Satz 7.8 auch σ_{rel} klein, also der Algorithmus (7.4) *stabil*. Für die Entwicklung von Algorithmen bedeutet das:

Schlecht konditionierte Elementarfunktionen vermeiden!

Als Beispiel betrachten wir die Funktion

$$f(x) = \sqrt{1 - \cos^2(x)}$$

auf $I = (0, \pi)$. Einen naheliegenden Algorithmus zur Auswertung von f an der Stelle $x_0 \in I$ liefert die Zerlegung

$$f(x_0) = g_3 \circ g_2 \circ g_1(x_0) \,, \quad g_1(x_0) = \cos(x_0) \,, \quad g_2(y_1) = 1 - y_1^2 \,, \quad g_3(y_2) = \sqrt{y_2} \,. \tag{7.21}$$

Unter Verwendung von Satz 7.8 und

$$\kappa_2 = \frac{2\cos^2(x_0)}{1 - \cos^2(x_0)} \,, \qquad \kappa_3 = \frac{1}{2}$$

lässt sich die Stabilität σ_g von (7.21) wie folgt abschätzen

$$\sigma_g \leq 1 + \kappa_3(1 + \kappa_2) = 1 + \frac{1 + \cos^2(x_0)}{2(1 - \cos^2(x_0))} \,.$$

Offenbar wird diese obere Schranke für $x_0 \to \pi$ beliebig groß. Der Grund dafür ist die schlechte Kondition von g_2 (Auslöschung!). Zu deren Vermeidung schreiben wir f äquivalent um. Mit Hilfe der bekannten Formel $\sin^2(x) + \cos^2(x) = 1$ gilt nämlich

$$f(x) = \sin(x_0) \,.$$

Der resultierende triviale Algorithmus

$$f(x_0) = h_1(x_0) \,, \quad h_1(x_0) = \sin(x_0) \,,$$

hat bekanntlich die optimale Stabilität $\sigma_h = 1$.

Wir werfen noch einen Blick auf die Stabilitätsabschätzung in Satz 7.8. Offenbar taucht die Kondition κ_1 der Auswertung von $g_1(x_0)$ dort überhaupt nicht auf. Diese Beobachtung führt zu der zweiten Regel.

Unvermeidbare, schlecht konditionierte Elementarfunktionen an den Anfang!

Beispiele haben wir mit den Algorithmen (7.5) und (7.6) schon gleich zu Beginn dieses Abschnitts kennengelernt.

Als nächstes wollen wir Zerlegungen in Summen, Differenzen, Produkte oder Quotienten von Funktionen betrachten. Da diese Grundrechenarten jeweils zwei Argumente haben, sind die entsprechenden Elementarfunktionen g_i vektorwertig oder haben Vektoren als Argumente. Beispielsweise wird die Zerlegung

$$f(x) = ax^2 + bx = u(x) + v(x) \,, \qquad u(x) = ax^2 \,, \quad v(x) = bx \,,$$

in der Form (7.4) geschrieben, indem man

$$f(x) = g_2 \circ g_1(x) \,, \qquad g_1(x) = (u(x), v(x)) \,, \quad g_2(u,v) = u + v$$

setzt. Wir beginnen mit einem Analogon zu Satz 7.6.

Satz 7.9. *Es sei $f(x_0) \neq 0$ sowie $u(x_0)$, $v(x_0) \neq 0$ und*

$$u(x_0) = u_n \circ u_{n-1} \circ \cdots \circ u_1(x_0) \,, \quad v(x_0) = v_m \circ v_{m-1} \circ \cdots \circ v_1(x_0)$$

Algorithmen zur Auswertung von $u(x_0)$ und $v(x_0)$ mit der relativen Stabilität σ_u, σ_v. Dann genügt der Algorithmus

$$f(x_0) = u(x_0) * v(x_0) = (u_n \circ \cdots \circ u_1(x_0)) * (v_m \circ \cdots \circ v_1(x_0))$$

die Stabilitätsabschätzung

$$\sigma_{\text{rel}} \leq 1 + \kappa_* \max\{\sigma_u, \sigma_v\} \,,$$

wobei κ_* jeweils die Kondition der Grundrechenart $* = +, -, \cdot, /$ an der Stelle $u(x_0)$, $v(x_0)$ bedeutet. Im Falle $* = +, -$ ist dabei $u(x_0), v(x_0) > 0$ vorausgesetzt.

Beweis. Ist $\sigma_u = \infty$ oder $\sigma_v = \infty$, so bleibt nichts zu zeigen. Es sei also σ_u, $\sigma_v < \infty$ und

$$\tilde{f}(\varepsilon, x_0) = (1 + \varepsilon_*)\tilde{u}(\varepsilon_u, x_0) * \tilde{v}(\varepsilon_v, x_0)$$

mit $\varepsilon_u = (\varepsilon_{u,1}, \ldots, \varepsilon_{u,n})$, $\varepsilon_v = (\varepsilon_{v,1}, \ldots, \varepsilon_{v,m})$, $\varepsilon_* \in \mathbb{R}$ und $\varepsilon = (\varepsilon_u, \varepsilon_v, \varepsilon_*)$. Mit den Abkürzungen

$$f = f(x_0) \,, \quad \tilde{f} = \tilde{f}(\varepsilon, x_0) \,, \quad u = u(x_0) \,, \quad \tilde{u} = \tilde{u}(\varepsilon_u, x_0) \,, \quad v = v(x_0) \,, \quad \tilde{v} = \tilde{v}(\varepsilon_v, x_0) \,,$$

erhält man aus den Definitionen von Kondition und Stabilität mit der Dreiecksungleichung

$$\begin{aligned}
\tfrac{|f-\tilde{f}|}{|f|} &\leq \tfrac{|u*v-\tilde{u}*\tilde{v}|}{|u*v|} + |\varepsilon_*| \tfrac{|\tilde{u}*\tilde{v}|}{|u*v|} \\
&\leq \kappa_* \max\left\{ \tfrac{|u-\tilde{u}|}{|u|}, \tfrac{|v-\tilde{v}|}{|v|} \right\} + o\left(\max\left\{ \tfrac{|u-\tilde{u}|}{|u|}, \tfrac{|v-\tilde{v}|}{|v|} \right\} \right) + |\varepsilon_*| + |\varepsilon_*| \tfrac{|u*v-\tilde{u}*\tilde{v}|}{|u*v|} \\
&\leq \kappa_* \max\{\sigma_u, \sigma_v\} \max\{\|\varepsilon_u\|, \|\varepsilon_v\|\} + |\varepsilon_*| + o(\max\{\|\varepsilon_u\|, \|\varepsilon_v\|\}) \\
&\leq (1 + \kappa_* \max\{\sigma_u, \sigma_v\}) \, \|\varepsilon\| + o(\|\varepsilon\|)
\end{aligned}$$

und damit die Behauptung. □

Wir verzichten darauf, beliebig geschachtelte vektorwertige Elementarfunktionen zu betrachten und ein entsprechendes Gegenstück zu Satz 7.8 zu formulieren.

Stattdessen wollen wir Satz 7.9 nutzen, um das Polynomdesaster aus Abschnitt 7.1.2 zu erklären. Die in Algorithmus 7.2 implementierte Darstellung $f(x) = x^3 + 12a^2x - 6ax^2 - 8a^3$ basiert auf der Zerlegung[4]

$$f(x_0) = (u_1(x_0) + u_2(x_0)) - v(x_0) \,, \qquad u_1(x) = x^3 \,, \quad u_2(x) = 12a^2x \,, \quad v(x) = 6ax^2 + 8a^3 \,. \tag{7.22}$$

Wir ignorieren also die bei der Auswertung der Komponenten u_1, u_2 und v auftretenden Rundungsfehler. Bekanntlich ist

$$\sigma_{u_1} = \sigma_{u_2} = \sigma_v = 1$$

die Stabilität der trivialen Algorithmen zur Auswertung von $u_1(x_0)$, $u_2(x_0)$ und $v(x_0)$. Aus Satz 7.9 erhält man daher für die Stabilität σ_u des Algorithmus

[4] Die Klammerung wählt MATLAB automatisch.

$$u(x_0) = u_1(x_0) + u_2(x_0)$$

die Abschätzung

$$\sigma_u \leq 1 + \max\{\sigma_{u_1}, \sigma_{u_2}\} = 2 .$$

Eine weitere Anwendung von Satz 7.9 ergibt für die Stabilität von Algorithmus (7.22) die Abschätzung

$$\sigma_{\text{rel}} \leq 1 + \frac{|u(x_0)| + |v(x_0)|}{|u(x_0) - v(x_0)|} \max\{\sigma_u, \sigma_v\} \approx 10^{21} .$$

Dementsprechend müssen wir im Ergebnis mit einem relativen Fehler der Größenordnung

$$10^{21} eps = 10^{21} \cdot 10^{-16} = 10^5$$

rechnen. Genau das ist passiert.

Die äquivalente Umformulierung

$$f(x) = (x - 2a)^3$$

von (7.22) führt auf den Algorithmus

$$f(x_0) = h_2 \circ h_1(x_0) , \qquad h_1(x_0) = x_0 - 2a , \quad h_2(y_1) = y_1^3 . \tag{7.23}$$

Man beachte, daß die *unvermeidliche Auslöschung* nun in h_1 *am Anfang des Algorithmus* untergebracht ist. Direkte Anwendung von Satz 7.8 liefert für die Stabilität σ_h von (7.23) die Abschätzung

$$\sigma_h \leq 1 + \kappa_2 = 4 .$$

Das ist eine echte Verbesserung.

7.3 Rekursive Stabilitätsberechung mittels Auswertungsbäumen

Unsere bisherigen Berechnungen von Stabilitätsindikatoren hinterlassen nicht unbedingt den Eindruck, dass man solche Berechnungen auch für sehr viel komplexere Algorithmen durchführen kann. Genau diesen Eindruck wollen wir nun widerlegen und zeigen, dass man die Abschätzung des Rundungsfehlers sogar recht einfach automatisch parallel zu jedem Algorithmus durchführen kann. Dazu braucht man nicht mehr Einsicht in das Problem als wir uns bisher erarbeitet haben.

Unser Startpunkt ist die Einsicht, dass jede Auswertung einer beliebigen Funktion $F : \mathbb{R}^n \to R$ durch eine Sequenz von geschachtelten Elementaroperationen erfolgt. Verschiedene Auswertungen unterscheiden sich in der Reihenfolge, in der Elementarfunktionen rekursiv ineinander geschachtelt sind. Manche Elementarfunktionen greifen dabei direkt auf die Eingabewerte $x_1, \ldots, x_n$ zurück, während andere auf Ergebnisse vorher durchgeführter Berechungen als Eingabe verwenden.

Beispiel 7.10. Für $F(x_1, x_2) = (x_1 - x_2)^2$ betrachten wir die zwei Auswertungen

$$F_1(x_1, x_2) = f_2(f_1(x_1, x_2)), \quad f_1(x, y) = x - y, \quad f_2(x) = x^2$$

$$F_2(x_1, x_2) = f_5\Big(f_1\Big(f_2(x_1), f_4(f_3(x_1, x_2))\Big), f_2(x_2)\Big),$$
$$f_3(x, y) = xy, \quad f_4(x) = 2x, \quad f_5(x, y) = x + y$$

Wir wollen die Verschachtelungen der sukzessiven Auswertung von Elementarfunktionen, wie in Abbildung 7.1 für das obige Beispiel illustriert, in Baumform darstellen. Ein solcher *Auswertungsbaum* besteht aus Knoten und Kanten, wobei die Knoten jeweils entweder eine Elementaroperation oder einen Eingabewert repräsentieren und die in einen Knoten eingehenden Kanten angeben, welche Eingabewerte oder Ergebnisse vorher durchgeführter Elementaroperationen in dem Knoten verwendet werden. Als Blätter bezeichnen wir jene Knoten, die Eingabewerte repräsentieren und daher keine eingehenden Kanten haben.

Ausserdem hat der Baum eine Wurzel, d.h., einen Knoten, der nur eingehende Kanten und keine ausgehenden hat; jedes Blatt ist mit der Wurzel verbunden und es gibt nur eine Wurzel (was die eindeutige

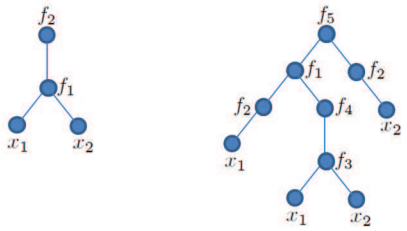

Abb. 7.1 Darstellung der Auswertungen F_1 und F_2 als Auswertungsbäume.

Charakterisierung eines Baumes ist). Nimmt man alle Kanten weg, die in die Wurzel einmünden, so zerfällt der Baum in (Teil-)Bäume, deren jeweilige Wurzeln die vorher mit der ursprünglichen Wurzel verbundenen Knoten sind.

Wir werden jeden solchen Auswertungsbaum mit β bezeichnen und mit $\#\beta$ die Anzahl der Knoten in dem Baum β. Von seiner Wurzel aus gesehen, ist der Baum eindeutig dadurch gekennzeichnet, dass man die (Teil-)Bäume angibt, in die er nach Wegnahme der zur Wurzel führenden Kanten zerfällt. Also kann β wie folgt dargestellt werden:

$$\beta = [\beta_1, \ldots, \beta_m],$$

wobei $\beta_1, \ldots, \beta_m$ die Teilbäume bezeichnen. Die Gesamtzahl von Knoten in einem solchen Baum ergibt sich dann als

$$\#\beta = 1 + \#\beta_1 + \ldots + \#\beta_m.$$

Der einfachste Baum hat nur einen Knoten (seine Wurzel) und die Form

$$\beta = [\,], \quad \text{mit} \quad \#\beta = 1,$$

da nach Entfernen der Wurzel nichts übrig bleibt.

Beispiel 7.11. Für F_1 und F_2 aus dem letzten Beispiel ergeben sich die in Abbildung 7.1 gezeigten Auswertungsbäume, deren formale Darstellung nun so aussieht:

$$\beta_{F_1} = [\beta_1], \quad \beta_1 = [\beta_{0,1}, \beta_{0,2}]$$

wobei die beiden Bäume $\beta_{0,1} = \beta_{0,2} = []$ die Blätter darstellen, mit den Eingabewerten x_k in $\beta_{0,k}$, $k = 1, 2$. Der Auswertungsbaum zu F_2 ist vergleichsweise komplizierten (man beachte Abbildung 7.2):

$$\begin{aligned} \beta_{F_2} &= [\beta_e, \beta_f] \\ \beta_e &= [\beta_a, \beta_b], \quad \beta_f = [\beta_d] \\ \beta_a &= [\beta_{0,i}], \quad \beta_b = [\beta_c], \quad \beta_c = [\beta_{0,ii}, \beta_{0,iii}], \quad \beta_d = [\beta_{0,iv}] \end{aligned}$$

wobei die vier Bäume $\beta_{0,k} = []$, $k = i, ii, iii, iv$ die Blätter darstellen, mit den Eingabewerten x_1 in $\beta_{0,k}$, $k = i, ii$ und x_2 in $\beta_{0,k}$, $k = iii, iv$.

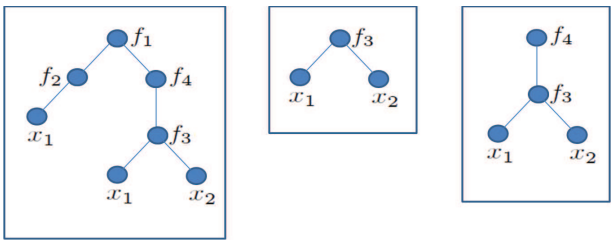

Abb. 7.2 Teilbäume β_e, β_c und β_b (von rechts nach links) des Auswertungsbaums β_{F_2}.

Für die mit einem Auswertungsbaum assoziierte Auswertung ordnen wir jedem Knoten, der kein Blatt ist, eine Elementarfunktion zu. Jeder solcher Knoten ist die Wurzel eines Teilbaums mit mehr als einem

Knoten; wir bezeichnen die mit dem Teilbaum β assoziierte Elementarfunktion mit f^β. Durch die Auswertung ergeben sich in den Knoten (Zwischen-)Ergebnisse. Das Zwischenergebnis in dem Knoten, der die Wurzel des Teilbaumes β ist, bezeichnen wir mit z^β. z^β berechnet sich mittels der Elementarfunktion f^β aus den (vorher zu berechnenden) Ergebnissen $z^{\beta_1},\ldots,z^{\beta_m}$ der Berechnungen in den Teilbäumen $\beta_1,\ldots,\beta_m$ die durch Kanten mit der Wurzel von β verbunden sind. Stellt β ein Blatt dar (gilt also $\#\beta = 1$, so dass es keinen dem Baum β untergeordneten Teilbaum gibt), so ist z^β ein Eingabewert. Daraus ergibt sich die folgenden rekursive Darstellung der Auswertung:

$$z^\beta = \begin{cases} z^\beta & \text{if} \quad \#\beta = 1 \\ f^\beta(z^{\beta_1},\ldots,z^{\beta_m}) & \text{if} \quad \#\beta = m+1 > 1, \quad \beta = [\beta_1,\ldots,\beta_m] \end{cases} \tag{7.24}$$

wobei eine zusätzliche Zuordnung angibt, welche Eingabewerte den Größen z^β mit $\#\beta = 1$ zu geordnet sind.

Beispiel 7.12. Für die Auswertungsbäume zu F_1 und F_2 ergibt sich folgendes

$$f^{\beta_{F_1}} = f_2, \quad f^{\beta_1} = f_1, \quad z^{\beta_{0,k}} = x_k, k = 1,2 \tag{7.25}$$

und

$$\begin{aligned} f^{\beta_{F_2}} = f_5,\ f^{\beta_e} = f_1, \quad f^{\beta_f} = f_2, \\ f^{\beta_a} = f_2, \quad f^{\beta_b} = f_4, \quad f^{\beta_c} = f_3, \quad f^{\beta_d} = f_2, \end{aligned}$$

mit den Eingabewerten $z^{\beta_{0,k}} = x_1$, $k = i, ii$ und $z^{\beta_{0,k}} = x_2$, $k = iii, iv$.

Aus der rekursive Darstellung (7.24) der Auswertung folgt mittels unserer bisherigen Ergebnissen zu Stabilitätsindikatoren (Sätze 7.8 und 7.9 und offensichtliche Verallgemeinerung auf Elementarfunktionen mit mehr als zwei Argumenten/Eingabewerten) sofort die assoziierte *rekursive Abschätzungsvorschrift* für die Stabilitätsindikatoren der zu den Teilbäumen gehörenden Auswertungen:

Satz 7.13. *Die Stabilitätsindikatoren der durch (7.24) definierten rekursiven Auswertungsvorschrift lassen sich mittels*

$$\sigma^\beta \leq \begin{cases} 1 & \text{if} \quad \#\beta = 1 \\ 1 + \kappa(f^\beta)\max(\sigma^{\beta_1},\ldots,\sigma^{\beta_m}) & \text{if} \quad \#\beta = m+1 > 1, \\ & \text{and} \quad \beta = [\beta_1,\ldots,\beta_m] \end{cases} \tag{7.26}$$

rekursiv abschätzen, wobei $\kappa(f^\beta)$ die relative Kondition der Funktion f^β in den Eingabewerten $z^{\beta_1},\ldots,z^{\beta_m}$ bezeichnet.

Offensichtlich erlaubt (7.26) die völlig *automatische* Abschätzung der Stabilitätsindikatoren und damit des in der Berechung angesammelten Rundungsfehlers. Wenn wir annehmen, dass wir alle Teilkonditionen $\kappa(f^\beta)$ stabil berechnen können, dann können wir folgenden Merksatz formulieren:

Automatische Abschätzung des während der Ausführung eines Algorithmus angesammelten Rundungsfehlers ist möglich!

Doch bevor wir diese Einsicht auf wesentlich komplexere Algorithmen anwenden, wollen wir noch einige einfachere Beispiele studieren, um uns an die technische Durchführung der rekursiven Abschätzung zu gewöhnen.

Beispiel 7.14. Wir wollen diese rekursive Abschätzungsvorschrift zuerst in Anwendung auf den Auswertungsbaum zu F_1 illustrieren. Wir wählen konkrete Eingabewerte, um die Rechnung übersichtlicher zu gestalten: $x_1 = 10^{11} - 1 = 99999999999$, $x_2 = 10^{11}$. Aus $\sigma_{0,k} \leq 1$ für die Eingabebäume mit $k = 1,2$ und

$$\kappa(f^{\beta_1}) = \kappa(f_1) = \frac{|x_1| + |x_2|}{|x_1 - x_2|} = 2 \cdot 10^{11} - 1,$$

erhalten wir sofort $\sigma^{\beta_1} \leq 2 \cdot 10^{11}$. Dann mit $\kappa(f^{\beta_2}) = \kappa(f_2) \leq 2$ weiterhin sofort

$$\sigma^{F_1} = \sigma^{\beta_2} \leq 1 + 4 \cdot 10^{11}.$$

Wir erwarten bei der Auswertung F_1 mit den Eingabewerten $x_1 = 10^{11} - 1$ und $x_2 = 10^{11}$ auf einer Maschine mit Maschinengenauigkeit $\text{eps} = 10^{-16}$ also einen maximalen (relativen) Rundungsfehler von

$$\frac{|F(x_1,x_2) - F_1(x_1,x_2)|}{|F(x_1,x_2)|} \leq \sigma^{F_1}\text{eps} \leq (1 + 4 \cdot 10^{11}) \cdot \text{eps} \approx 4 \cdot 10^{-5}.$$

Wegen $F(x_1,x_2) = 1$ erwarten wir also auch den maximalen absoluten Rundungsfehler $4 \cdot 10^{-5}$. Die Durchführung dieser Berechnung (zum Beispiel auf einem handelsüblichen Taschenrechner) liefert in der Tat $F_1(x_1,x_2) = 1$, also einen Fehler (relativ und absolut) von 0. Der Fehler ist erheblich kleiner als abgeschätzt, da die Eingabewerte nicht gerundet werden und somit schon $\sigma_{0,k} = 1$ eine Überschätzung darstellt. Wenn wir allerdings zu $x_1 = 99999999999.0001$ und $x_2 = 10^{11}$ übergehen, dann ist das nicht mehr der Fall. Die Auswertung auf demselben Taschenrechner ergibt dann $F_1(x_1,x_2) = 0.9998$ und damit einen (relativen und absoluten) Fehler von $2 \cdot 10^{-4}$.

Beispiel 7.15. Zum Vergleich betrachten wir nun die rekursive Abschätzungsvorschrift in Anwendung auf den Auswertungsbaum zu F_2. Wieder sei $x_1 = 10^{11} - 1$, $x_2 = 10^{11}$. Aus $\sigma_{0,k} \leq 1$ für die Eingabebäume mit $k = i, ii, iii, iv$ und

$$\kappa(f^{\beta_c}) = \kappa(f_3) \leq 2, \quad \kappa(f^{\beta_a}) = \kappa(f^{\beta_d}) = \kappa(f_2) \leq 2, \quad \kappa(f^{\beta_b}) = \kappa(f_4) \leq 2,$$

erhalten wir sofort $\sigma^{\beta_c} \leq 3$, $\sigma^{\beta_d} \leq 3$, $\sigma^{\beta_a} \leq 3$ und $\sigma^{\beta_b} \leq 7$. Dann mit

$$\kappa(f^{\beta_e}) \leq \frac{|x_1^2| + |2x_1x_2|}{|x_1^2 - 2x_2x_1|} = \frac{3 \cdot 10^{22} - 4 \cdot 10^{11} + 1}{10^{22} - 1} \approx 3,$$

und

$$\kappa(f^{\beta_{F_2}}) \leq \frac{|x_2^2| + |x_1^2 - 2x_1x_2|}{|x_1^2 + x_1^2 - 2x_2x_1|} = 2 \cdot 10^{22} - 1,$$

und in Folge sofort

$$\begin{aligned} \sigma^{\beta_e} &\leq 1 + 7\kappa(f^{\beta_e}) \approx 22 \\ \sigma^{F_2} = \sigma^{\beta_{F_2}} &\leq 1 + \sigma^{\beta_e}\kappa(f^{\beta_{F_2}}) \approx 44 \cdot 10^{22} \end{aligned}$$

Wir erwarten bei der Auswertung F_2 mit den Eingabewerten $x_1 = 10^{11} - 1$ und $x_2 = 10^{11}$ auf einer Maschine mit Maschinengenauigkeit $\text{eps} = 10^{-16}$ also einen maximalen (relativen) Rundungsfehler von

$$\frac{|F(x_1,x_2) - F_2(x_1,x_2)|}{|F(x_1,x_2)|} \leq \sigma^{F_2}\text{eps} \approx 4.4 \cdot 10^7.$$

Wegen $F(x_1,x_2) = 1$ erwarten wir also auch den maximalen absoluten Rundungsfehler $4.4 \cdot 10^7$. Die Durchführung dieser Berechnung (zum Beispiel auf einem handelsüblichen Taschenrechner) liefert in der Tat $F_2(x_1,x_2) = 2097152$, also einen Fehler (relativ und absolut) von ca. $2.1 \cdot 10^6$.

7.4 Summationsalgorithmen

Unsere Aufgabe besteht in der Summation von n positiven reellen Zahlen, also der Auswertung von

$$s_n = \sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n \,.$$

Genauer gesagt wollen wir einen Algorithmus finden, der s_n durch *sukzessive gerundete Addition* möglichst genau approximiert. Diese Problematik hat praktische Bedeutung und ist daher recht genau untersucht. Ueberhuber [36, Kapitel 5.8] widmet dem Thema eine Fallstudie. Einen umfassenden Überblick über Summationsalgorithmen gibt Highham [18, Kapitel 4].

7.4.1 Rekursive Summation

Eine naheliegende Möglichkeit s_n auszuwerten besteht darin, die Summanden x_1 bis x_n einfach nacheinander zu addieren. Man beachte, daß dadurch eine bestimmte Klammerung festgelegt wird, nämlich beispielsweise

$$s_5 = (((x_1 + x_2) + x_3) + x_4) + x_5 \, .$$

Die entsprechende Rekursionsformel

$$s_1 = x_1 \, , \quad s_i = s_{i-1} + x_i \, , \qquad i = 2, \ldots, n \, ,$$

lässt sich direkt in einen Algorithmus umsetzen.

Algorithmus 7.16 (Rekursive Summation).

```
s=x(1);
for i = 2:n
  s=s+x(i);
end
```

Gleichbedeutend mit der Anwendung von Algorithmus 7.16 ist die Auswertung der *gestörten Rekursion*

$$\tilde{s}_1 = x_1 \, , \qquad \tilde{s}_i = \tilde{s}_{i-1} \tilde{+} x_i \, , \quad i = 2, \ldots, n \, , \tag{7.27}$$

bei der wir die exakte Addition durch die Gleitkommaaddition ersetzt haben. Wir wollen feststellen, wie *stabil* sich die Berechnungsvorschrift (7.27) gegenüber dieser Störung verhält. Die Frage ist also, wie stark sich die Lawine von Rundungsfehlern, die bei den vielen Gleitkommaadditionen entsteht, auf das Endergebnis auswirkt. Vereinfachend setzen wir dabei voraus, daß

$$x_i = \tilde{x}_i > 0 \in \mathbb{G}, \quad i = 1, \ldots, n \, . \tag{7.28}$$

Es gibt also *keine Eingabefehler*, und es kann *keine Auslöschung* auftreten.

Satz 7.17. *Unter der Voraussetzung* (7.28) *gilt für die rekursive Summation* (7.27) *die Fehlerabschätzung*

$$|s_n - \tilde{s}_n| \leq \sum_{i=1}^{n} x_i (2^{n+1-i} - 1) eps \, . \tag{7.29}$$

Dabei bedeutet $eps = eps(q, \ell)$ die Maschinengenauigkeit.

Beweis. Zur Vorbereitung erinnern wir an die binomische Formel

$$(a+b)^k = \sum_{l=0}^{k} \binom{k}{l} a^l b^{k-l} \, , \qquad \binom{k}{l} = \frac{k!}{l!(k-l)!} \, . \tag{7.30}$$

Setzt man $a = b = 1$, so folgt

$$\sum_{l=0}^{k} \binom{k}{l} = 2^k \, . \tag{7.31}$$

Doch nun zum eigentlichen Beweis. Wegen Satz 5.5 ist

$$\tilde{s}_1 = x_1 (1 + \varepsilon_1) \, , \qquad \tilde{s}_i = \mathrm{rd}(\tilde{s}_{i-1} + x_i) = (\tilde{s}_{i-1} + x_i)(1 + \varepsilon_i) \, , \quad i = 2, \ldots, n \, ,$$

mit $\varepsilon_1 = 0$ und $|\varepsilon_i| \leq eps$, $i = 2, \ldots, n$. Daraus folgt mit vollständiger Induktion die Darstellung

$$\tilde{s}_n = \sum_{i=1}^{n} x_i \prod_{j=i}^{n} (1 + \varepsilon_j) \, .$$

Mit Hilfe der Dreiecksungleichung erhält man daraus

$$|s_n - \tilde{s}_n| \leq \sum_{i=1}^{n} x_i K_i \, , \qquad K_i = \left| \prod_{j=i}^{n} (1 + \varepsilon_j) - 1 \right| \, . \tag{7.32}$$

Wir haben nun die Faktoren K_i abzuschätzen. Nach Ausmultiplizieren des Produkts heben sich die Summanden 1 und -1 weg. Dann wendet man die Dreiecksungleichung an, fügt die Summanden 1 und -1 wieder hinzu und macht die Ausmultiplikation rückgängig. Auf diese Weise ergibt sich

$$K_i = \left| \prod_{j=i}^{n} (1+\varepsilon_j) - 1 \right| \leq \prod_{j=i}^{n} (1+|\varepsilon_j|) - 1 \leq (1+eps)^{n+1-i} - 1 \,. \tag{7.33}$$

Nun liefert die binomische Formel wegen $eps < 1$ und (7.31) für $k = n+1-i$ die Abschätzung

$$(1+eps)^k - 1 = \sum_{l=1}^{k} \binom{k}{l} eps^l < eps \sum_{l=1}^{k} \binom{k}{l} = (2^k - 1)eps \,. \tag{7.34}$$

Insgesamt gilt also

$$K_i \leq (2^{n-i+1} - 1)eps \,, \qquad i = 1, \ldots, n \,.$$

Durch Einsetzen dieser Abschätzung in (7.32) erhalten wir schließlich die Behauptung. □

In Satz 7.17 haben wir eine *absolute Fehlerabschätzung* bewiesen. Offenbar liefert (7.29) aber unmittelbar auch die folgende Abschätzung des *relativen Fehlers*

$$\frac{|s_n - \tilde{s}_n|}{|s_n|} \leq (2^n - 1)eps \,. \tag{7.35}$$

Wir stellen erschreckt fest, daß sich die auftretenden Rundungsfehler von der Größenordnung *eps* im Laufe der Summation derart aufschaukeln können, daß sie in ihrer Auswirkung auf das Ergebnis um einen exponentiell wachsenden Faktor verstärkt werden. Nun, ganz so schlimm muß es nicht kommen, denn (7.29) und (7.35) stellen nur obere Schranken dar, welche den tatsächlichen Fehler im allgemeinen erheblich überschätzen.[5]

Um zu sehen, wie man die Abschätzung wesentlich realistischer machen kann, kehren wir kurz zu unseren Überlegungen zu Auswertungsbäumen zurück. Die rekursive Summation lässt sich in unserer Schreibweise aus Abschnitt 7.3 wie in Abbildung 7.3 darstellen:

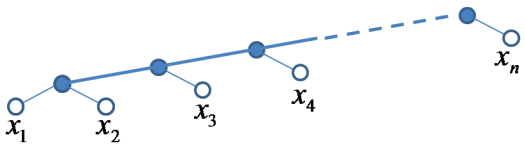

Abb. 7.3 Auswertungsbaum zur rekursiven Summation.

Zuerst haben wir all die Auswertungs(teil)bäume, die nur aus Blättern bestehen und die Eingabewerte zur Verfügung stellen (die nicht-gefüllten Knoten in Abbildung 7.3):

$$\beta_{0,i} = [], \quad \text{with } z^{\beta_{0,i}} = x_i,$$

für $i = 1 \ldots, n$. Die nächste Schicht von Knoten (die ausgefüllten in Abbildung 7.3) sind die Wurzeln der Teilbäume, die die Summen $s_{i-1} + x_i$ berechnen, angefangen mit $\beta_2 = [\beta_{0,1}, \beta_{0,2}]$ und $z_2 = x_1 + x_2$ und weiter für $i = 3, \ldots, n$,

$$\beta_i = [\beta_{i-1}, \beta_{0,i}], \quad f^{\beta_i}(x,y) = x+y, \quad z_i = z^{\beta_i} = z_{i-1} + x_i.$$

Da $\kappa(f^{\beta_i}) = 1$ (wir betrachten hier lediglich die Addition positiver Zahlen) erhalten wir unmittelbar

$$\sigma^{\beta_i} \leq 1 + \max\{\sigma^{\beta_{i-1}}, 1\} = 1 + \sigma^{\beta_{i-1}}, \quad i = 3, \ldots, n.$$

[5] Das liegt unter anderem an der sehr pessimistischen Abschätzung (7.34). In dieser Abschätzung haben wir auch höhere Ordnungen der Störungen in eps berücksichtigen. Das hatten wir eigentlich bei der Definition der Stabilität ausgeschlossen. Wenn wir die Abschätzung nur in führenden Ordnung ausführen, das heisst $(1+\text{eps})^k - 1$ durch $k$eps ersetzen, dann ergibt sich sofort die viel bessere Abschätzung $||s_n - \tilde{s}_n|/|s_n| \leq n\text{eps}$.

Daraus resultiert mit $\sigma^{\beta_2} \leq 2$ sofort $\sigma^{\beta_i} \leq i$. Daher haben wir gerade den folgenden Satz bewiesen:

Satz 7.18. *Für die rekursive Summation positiver Zahlen gilt die verbesserte Fehlerabschätzung*

$$\frac{|s_n - \tilde{s}_n|}{|s_n|} \leq n \,\mathrm{eps}\,. \tag{7.36}$$

Der Vergleich von (7.35) und (7.36) zeigt, welchen riesigen Gewinn wir soeben erzielt haben. Der Rundungsfehler kann nicht exponentiell mit der Anzahl der zu summierenden Zahlen wachsen, sondern lediglich linear und damit recht schwach! Also haben wir zum wiederholten Male gesehen, dass eine obere Abschätzung des Rundungsfehlers immer in Gefahr steht, den Fehler substantiell zu überschätzen. Ob das auch noch für die lineare Fehlerabschätzung (7.36) zutrifft, können wir jetzt noch nicht wissen; wir werden weiter unten numerische Experimente dazu zeigen. Die Abschätzung sagt uns allerdings, dass die rekursive Summation nur bei sehr grossen Summandenanzahlen signifikante Rundungsfehleranteile zeigen wird. Kann man dieses Problem auch noch vermeiden?

7.4.2 Hierarchische Summation

Die obige Abschätzung scheint zu zeigen, dass eine Verbesserung möglich ist, wenn man den Auswertungsbaum so flach wie möglich macht. Das führt uns sofort auf die Grundidee der sogenannten hierarchischen Summation. Dazu nehmen wir der Einfachheit halber an, daß $n = 2^J$ eine Zweierpotenz ist[6]. Setzt man nun die Klammern, beispielsweise für $n = 8 = 2^3$, wie folgt,

$$s_8 = ((x_1 + x_2) + (x_3 + x_4)) + ((x_5 + x_6) + (x_7 + x_8))\,,$$

so ist jeder Summand genau an $J = 3$ Additionen beteiligt. Das Prinzip besteht darin, in jedem Schritt die benachbarten Summanden zu addieren und dadurch eine neue Summe mit halbierter Anzahl von Summanden zu erzeugen. Die *hierarchische Struktur* dieser Vorgehensweise wird in Abbildung 7.4 deutlich.

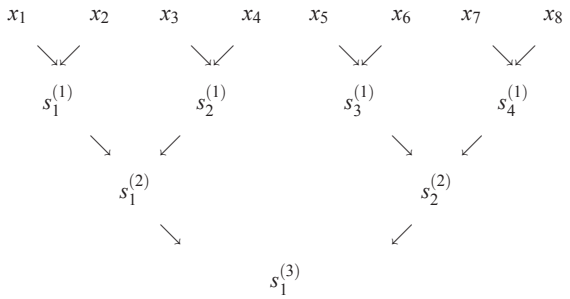

Abb. 7.4 Hierarchische Summation

Aus Abbildung 7.4 lässt sich auch die folgende Rekursion ablesen,

$$s_l^{(0)} = x_l\,, \quad l = 1,\ldots,2^J\,, \qquad s_l^{(k)} = s_{2l-1}^{(k-1)} + s_{2l}^{(k-1)}\,, \quad l = 1,\ldots 2^{J-k},\ k = 1,\ldots,J\,. \tag{7.37}$$

Das gesuchte Ergebnis ist dann $s_n = s_1^{(J)}$. Die Berechnungsvorschrift (7.37) lässt sich direkt algorithmisch umsetzen:

Algorithmus 7.19 (Hierarchische Summation).

```
s=x;

for k = 1:J
```

[6] Theoretisch können wir das natürlich immer erreichen, indem wir gegebenenfalls zusätzliche Nullen addieren. Praktisch wird man effizientere Modifikationen finden.

```
  for l=1:2**(J-k)
     s(l)=s(2*l-1)+s(2*l);
  end
end
```

Eleganter wäre allerdings eine rekursive Implementierung.

Abbildung 7.4 sollte ausreichen, um gleichzeitig deutlich zu machen, wie der zugehörige Auswertungsbaum aussieht. Die Auswertung von Algorithmus 7.19 in Gleitkommaarithmetik ist gleichbedeutend mit der gestörten Berechnungsvorschrift

$$\tilde{s}_l^{(0)} = x_l\,, \quad l = 1,\ldots,2^J\,, \qquad \tilde{s}_l^{(k)} = \tilde{s}_{2l-1}^{(k-1)} \tilde{+} \tilde{s}_{2l}^{(k-1)}\,, \quad l = 1,\ldots 2^{J-k},\ k = 1,\ldots,J\,, \tag{7.38}$$

und dem Endergebnis $\tilde{s}_n = \tilde{s}_1^J$. Im Vergleich mit Algorithmus 7.16 haben wir nur die Klammerung geändert. Wir wollen sehen, ob es uns gelungen ist, auf diese Weise die Stabilität zu verbessern.

Satz 7.20. *Unter der Voraussetzung* (7.28) *gilt für die hierarchische Summation* (7.38) *die Fehlerabschätzung*

$$\frac{|s_n - \tilde{s}_n|}{|s_n|} \le (1 + \log n)\,\mathrm{eps}\,. \tag{7.39}$$

Beweis. Die Blätter des zur Berechnung (7.37) gehörenden Auswertungsbaums seien mit

$$\beta_{0,i} = [], \quad \text{with } z^{\beta_{0,i}} = x_i,$$

bezeichnet, das entspricht der Ebene 0 im hierarchischen Baum und dem Superindex 0 in (7.37). Auf Ebene $j > 0$ (bzw. für Index j) werden die Ergebnisse je zweier Bäume der darunterliegenden Ebene addiert:

$$\beta_{j,i} = [\beta_{j-1,2i-1}\beta_{j-1,2i}], \quad z^{\beta_{j,i}} = z^{\beta_{j-1,2i}} + z^{\beta_{j-1,2i-1}} \quad i = 1,\ldots,n/(2^j).$$

Daher haben wir

$$\sigma^{\beta_{j,i}} \le 1 + \max\{\sigma^{\beta_{j-1,2i-1}}, \sigma^{\beta_{j-1,2i}}\} = 1 + \sigma^{\beta_{j-1,2k}} \le j + 1,$$

und damit auf der letzten Ebene $J = \log n$ in der Wurzel des Gesamtbaumes:

$$\sigma^{\beta_{J,1}} \le J + 1 = 1 + \log n.$$

Daraus folgt sofort für die behauptete Aussage für den relativen Gesamtfehler.

Abschließend wollen wir anhand numerischer Experimente feststellen, inwieweit sich unsere theoretischen Resultate auch im praktischen Verhalten der Algorithmen widerspiegeln. Da sich Rundungsfehler zufällig verstärken oder auslöschen können, sollen unsere Summationsverfahren anhand mehrerer Tests verglichen werden. Für jeden Test wählen wir jeweils $n_J = 2^J$ Zufallszahlen $x_i \in (0,1)$ mit $\ell = 7$ Stellen und $J = 1,\ldots,15$. Deren Summe berechnen wir dann approximativ mittels rekursiver und hierarchischer Summation bei 7-stelliger Gleitkommaaddition. Zum Vergleich wird noch die „exakte“ Summe s_n durch hierarchische Summation mit doppelt genauer Arithmetik (15 Stellen) ausgerechnet. Abbildung 7.5 zeigt die Mittelwerte der relativen Fehler von 100 solcher Tests in Abhängigkeit von der Anzahl $2^1,\ldots,2^{15}$ der Summanden. Zunächst fällt auf, daß bei allen Verfahren der Fehler deutlich geringer ausfällt als nach unserer Stabilitätsanalyse zu befürchten war. Qualitativ werden unsere theoretischen Untersuchungen aber nachhaltig bestätigt: Die Verbesserung der rekursiven Summation 7.16 durch aufsteigende Anordnung der Summanden ist sichtbar, aber nicht gravierend. Dagegen entspricht die Überlegenheit der hierarchischen Summation 7.19 dem dramatisch besseren Stabilitätsresultat in Satz 7.20. Während bei rekursiver und aufsteigender Summation der relative Fehler um zwei Größenordnungen anwächst, ist bei der hierarchischen Summation auch für $n_{15} = 32\,768$ Summanden kein wesentlicher Einfluß der Rundungfehler sichtbar.

Die Summation positiver Zahlen ist stabil.

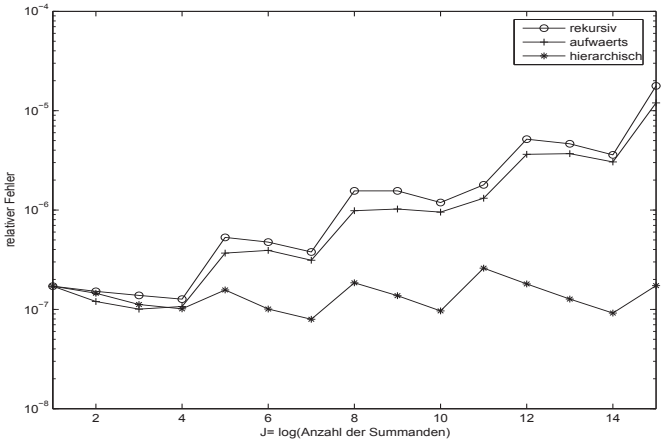

Abb. 7.5 Vergleich von rekursiver und hierarchischer Summation

7.5 Varianzberechnung

Wir wollen als weiteres etwas anspruchsvolleres Beispiel die Berechnung der Varianz einer Datenfolge $x_1,\ldots,x_n$ betrachten,

$$V(x_1,\ldots,x_n,\bar{x}) = \frac{1}{n}\sum_{k=1}^{n}(x_k-\bar{x})^2, \quad \bar{x} = \frac{1}{n}\sum_{k=1}^{n}x_k,$$

wobei wir davon ausgehen, dass wir den Mittelwert $\bar{x}$ bereits als vorab berechneten Wert vorliegen haben. Wir können dabei davon ausgehen, dass die Datenfolge durchweg positive Zahlen enthält, so dass die Berechnung des Mittelwerts mittels hierarchischer Summation kaum Rundungsfehler erzeugt und daher die durch die Berechnung ins Spiel kommenden Fehler im Weiteren ignorieren werden können. Wenn wir den Rundungsfehler dieser Vorabberechnung trotzdem in unserer Analyse berücksichtigen möchten, so könnte er als Eingabefehler betrachtet werden und dann mittels Satz 7.5 behandelt werden.

In der Literatur findet man zwei verschiedene Auswertungsvorschriften für die Funktion V:

$$V_1(x_1,\ldots,x_n,\bar{x}) = \frac{1}{n}\sum_{k=1}^{n}(x_k-\bar{x})^2 \quad \text{(Summe der Abweichungsquadrate)}$$

$$V_2(x_1,\ldots,x_n,\bar{x}) = \frac{1}{n}\sum_{k=1}^{n}x_k^2-\bar{x}^2 \qquad \text{(Summe der Datenquadrate)}$$

wobei V_2 voraussetzt, dass $\bar{x}$ tatsächlich der Mittelwert ist.

Wir werden nun das Rundungsfehlerverhalten von auf diesen beiden Formeln beruhenden Algorithmen studieren. Bei einer Auswertung von V_1, z.B. wie in Algorithmus 7.21, werden viele potentiell zu Auslöschung führende Subtraktionen ausgeführt, während bei einer Auswertung von V_2 nur eine Subtraktion nötig ist. Gilt die oben formulierte allgemeine Regel, dass potentiell schlecht konditionierte Operationen (also hier die Subtraktionen) zuerst auszuführen sind, dann immer noch? Anders gefragt: Ist eine Auswertung von V_1 im Normalfall stabiler als eine von V_2? Zur Beantwortung dieser Frage müssen wir natürlich wiederum zuerst die Rundungsfehler abschätzen.

Ein Algorithmus, der die Berechnung von V_1 realisiert ist der folgende:

Algorithmus 7.21 (Rekursive Berechnung der Varianz V_1).

```
Var=0;
for i = 1:n
  Var=Var+(x(i)-bx)^2;
end
Var=Var/n;
```

Ein möglicher Auswertungsbaum zu V_1 bzw. Algorithmus 7.21 auf Basis rekursiver Summation der Differenzen ist in Abbildung 7.6 skizziert. Natürlich könnte man auch hier zur hierarchischen Summation

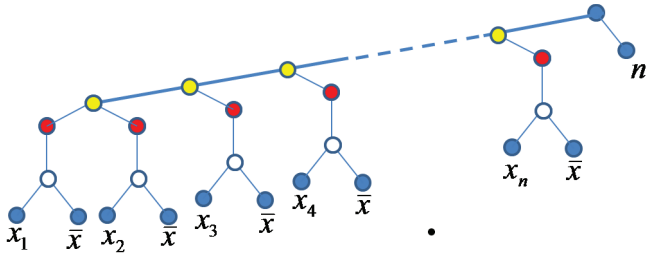

Abb. 7.6 Auswertungsbaum zur Varianzberechnung V_1.

greifen und dabei vermuten, dass der sich ergebende Rundungsfehler geringer ausfallen würde. Das werden wir aber wegen der etwas aufwendigeren Notation beim hierarchischen Vorgehen vermeiden; es wird sich letztlich herausstellen, dass die wesentlich wichtigeren Beiträge zum Gesamtrundungsfehler aus den (eventuell schlecht konditionierten) Subtraktionen kommen.

Satz 7.22. *Unter den oben gemachten Voraussetzungen gilt für die Algorithmus 7.21 die Abschätzung des Rundungsfehlers*

$$\left|\frac{V-\tilde{V}_1(x_1,\ldots,x_n,\bar{x})}{V}\right| \leq \left(C(n)+2\max_{j=1,\ldots,n}\frac{|x_j|+|\bar{x}|}{|x_j-\bar{x}|}\right)\text{eps}, \tag{7.40}$$

mit $V=V(x_1,\ldots,x_n,\bar{x})$ und einer Konstanten $C=C(n)$, die nicht von den jeweiligen Werten der betrachteten Datenfolge, sondern lediglich linear von deren Länge abhängt:

$$C(n)=5+2n=\mathcal{O}(n).$$

Dabei bedeutet eps $=eps(q,\ell)$ *die Maschinengenauigkeit.*

Beweis. Wir beziehen uns im Folgenden auf den in Abbildung 7.6 skizzierten Auswertungsbaum. Die Blätter dieses Baums sind $n+1$ Knoten mit den Eingabewerten (blau in Abb. 7.6):

$$\beta_0=[],\quad \text{with } z^{\beta_0}=\bar{x},\qquad \beta_i=[],\quad \text{with } z^{\beta_i}=x_i.$$

Die nächste Schicht von Knoten (weiss in Abb. 7.6) sind die Wurzeln der Teilbäume, die die Differenzen $x_i-\bar{x}$ berechnen:

$$\beta_{i+n}=[\beta_i,\beta_0],\quad f^{\beta_{i+n}}(x,y)=x-y,\quad z^{\beta_{i+n}}=x_i-\bar{x}.$$

Die bekannte Formel für die relative Kondition der Subtraktion führt uns sofort zu

$$\kappa(f^{\beta_{i+n}})=\frac{|x_i|+|\bar{x}|}{|x_i-\bar{x}|},\quad \sigma^{\beta_{i+n}}\leq 1+\frac{|x_i|+|\bar{x}|}{|x_i-\bar{x}|}.$$

Nun kommen wir der Schicht von Knoten (rot in Abb. 7.6), in der die Differenzen jeweils quadriert werden:

$$\beta_{i+2n}=[\beta_{i+n}],\quad f^{\beta_{i+2n}}(x)=x^2,\quad z^{\beta_{i+2n}}=(x_i-\bar{x})^2,$$

mit Kondition und Stabilität gemäss

$$\kappa(f^{\beta_{i+2n}})=2,\quad \sigma^{\beta_{i+2n}}\leq 3+2\frac{|x_i|+|\bar{x}|}{|x_i-\bar{x}|}.$$

Die letzte Schicht von Knoten sind die Wurzeln (gelb) der Bäume, die die quadrierten Differenzen sukzessive aufsummieren:

$$\beta_{1+3n}=[\beta_{1+2n},\beta_{2+2n}],\quad \beta_{i+3n}=[\beta_{i-1+3n},\beta_{i+1+2n}],i=2,\ldots,n-1,$$

mit

$$f^{\beta_{i+3n}}(x,y)=x+y,\quad \kappa(f^{\beta_{i+3n}})=1,$$

und daraus folgenden Stabilitäten

$$\sigma^{\beta_{1+3n}} \le 1+\max\left(\sigma^{\beta_{1+2n}},\sigma^{\beta_{2+2n}}\right) = 4+2\max_{i=1,2}\frac{|x_i|+|\bar{x}|}{|x_i-\bar{x}|},$$
$$\sigma^{\beta_{i+3n}} \le 1+\max\left(\sigma^{\beta_{i-1+3n}},\sigma^{\beta_{i+1+2n}}\right),$$

die wir nochmals abschätzen zu

$$\sigma^{\beta_{i+3n}} \le i+3+2\max_{j=1,\ldots,i+1}\frac{|x_j|+|\bar{x}|}{|x_j-\bar{x}|}, \quad i=1,\ldots,n-1.$$

Letztendlich wird das Ergebnis durch n geteilt:

$$\beta_{4n}=[\beta_{4n-1}], \quad f^{\beta_{4n}}(x)=x/n, \quad \kappa(f^{\beta_{4n}})=2,$$

so dass sich für die Gesamtauswertung ergibt:

$$\sigma^{V_1}=\sigma^{\beta_{4n}} \le 1+2\sigma^{\beta_{4n-1}} \le 5+2n+2\max_{j=1,\ldots,n}\frac{|x_j|+|\bar{x}|}{|x_j-\bar{x}|}.$$

Wir sehen, dass diese Auswertung instabil werden kann, wenn es Werte x_i in der Datenfolge gibt, die sehr nahe an dem Mittelwert $\bar{x}$ liegen.

Nun wenden wir uns einem Algorithmus zu, der die Berechnung von V_2 realisiert:

Algorithmus 7.23 (Rekursive Berechnung der Varianz V_2).

```
Var=0;
for i = 1:n
  Var=Var+x(i)^2;
end
Var=Var/n-bx^2;
```

Satz 7.24. *Unter den oben gemachten Voraussetzungen gilt für die Algorithmus 7.23 die Abschätzung des Rundungsfehlers*

$$\left|\frac{V-\tilde{V}_2(x_1,\ldots,x_n,\bar{x})}{V}\right| \le \left(C_1(n)+C_2(n)\frac{|\bar{x^2}|}{|V|}\right)\text{eps}. \tag{7.41}$$

mit $V=V(x_1,\ldots,x_n,\bar{x})$ und Konstanten C_1 und C_2, die nicht von den jeweiligen Werten der betrachteten Datenfolge, sondern nur von ihrer Länge abhängen:

$$C_1(n)=6+2n=\mathcal{O}(n), \quad C_2(n)=10+4n=\mathcal{O}(n).$$

Dabei bedeutet $\text{eps}=eps(q,\ell)$ *die Maschinengenauigkeit.*

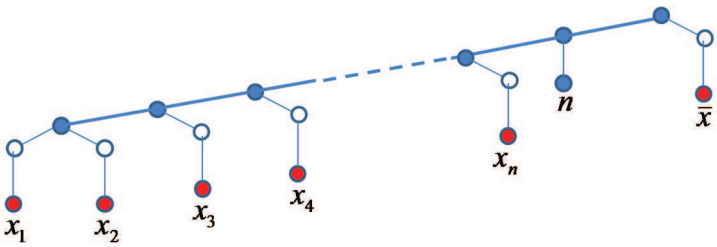

Abb. 7.7 Auswertungsbaum zur Varianzberechnung V_2.

Beweis. Der wiederum auf rekursiver Summation beruhende Auswertungsbaum zu V_2 sieht einfacher aus, siehe Abbildung 7.7. Zuerst haben wir wieder die Schicht der Eingabeknoten (rot in Abb. 7.7):

$$\beta_0=[], \quad \text{with } z^{\beta_0}=\bar{x}, \qquad \beta_i=[], \quad \text{with } z^{\beta_i}=x_i$$

und den Teilbaum, der zuerst das Quadrat des Mittelwertes berechnet:

$$\beta_{00} = [\beta_0], \quad f^{\beta_{00}}(x) = x^2, \quad \kappa(f^{\beta_{00}}) = 2, \quad \sigma^{\beta_{00}} \leq 3.$$

Die Folgeschicht von Knoten (weiss) berechnet die Quadrate der Daten

$$\beta_{i+n} = [\beta_i], \quad f^{\beta_{i+n}}(x) = x^2, \quad z^{\beta_{i+n}} = x_i^2,$$

mit

$$\kappa(f^{\beta_{i+n}}) = 2, \quad \sigma^{\beta_{i+n}} \leq 3.$$

Nun werden die Datenquadrate aufsummiert,

$$\beta_{1+2n} = [\beta_{1+n}, \beta_{2+n}], \quad \beta_{i+3n} = [\beta_{i-1+2n}, \beta_{i+1+n}], i = 2, \ldots, n-1,$$

mit Kondition

$$f^{\beta_{i+2n}}(x,y) = x+y, \quad \kappa(f^{\beta_{i+2n}}) = 1,$$

und Stabilität

$$\sigma^{\beta_{i+2n}} \leq 1 + \max\left(\sigma^{\beta_{i-1+2n}}, 3\right) = i+3$$

dann durch n geteilt,

$$\beta_{3n} = [\beta_{3n-1}], \quad f^{\beta_{3n}}(x) = x/n, \quad \kappa(f^{\beta_{3n}}) = 2, \quad \sigma^{\beta_{3n}} \leq 5+2n.$$

Dann wird erst die Differenz berechnet

$$\beta_{3n+1} = [\beta_{3n-1}, \beta_{00}], \quad f^{\beta_{3n+1}}(x,y) = x-y,$$

wobei die Kondition sich zu

$$\kappa(f^{\beta_{3n+1}}) = \frac{|\sum_k x_k^2/n| + |\overline{x^2}|}{|\sum_k x_k^2/n - \overline{x^2}|}$$

ergibt und daraus die Stabilität

$$\begin{aligned}\sigma^{V_2} &= \sigma^{\beta_{3n+1}} \leq 1 + \kappa(f^{\beta_{3n+1}}) \max\left(\sigma^{\beta_{3n}}, \sigma^{\beta_{00}}\right) \\ &\leq 1 + (5+2n)\frac{|\sum_k x_k^2/n| + |\overline{x^2}|}{|\sum_k x_k^2/n - \overline{x^2}|} = 1 + (5+2n)\left(1 + 2\frac{|\overline{x^2}|}{|V(x_1, \ldots, x_n, \overline{x})|}\right)\end{aligned}$$

folgt.

Wenn wir nun zu einer hierarchischen Summation der Datenquadrate übergehen, dann bleibt der Grossteil des Beweises gleich und nur der Teil, in dem die Summation abgeschätzt wird, muss angepasst werden. Daraus resultiert sofort:

Satz 7.25. *Unter den oben gemachten Voraussetzungen und $n = 2^J$ für ein ganzzahliges J gilt für die Berechnung von V_2 mittels hierarchischer Summation der Datenquadrate*

$$\left|\frac{V - \tilde{V}_2(x_1, \ldots, x_n, \overline{x})}{V}\right| \leq \left(c_1(n) + c_2(n)\frac{|\overline{x^2}|}{|V|}\right)\text{eps}. \tag{7.42}$$

mit $V = V(x_1, \ldots, x_n, \overline{x})$ und positiven Konstanten c_1 und c_2, die nicht von den jeweiligen Werten der betrachteten Datenfolge, sondern nur von ihrer Länge abhängen:

$$c_1(n) = \mathcal{O}(\log n), \quad c_2(n) = \mathcal{O}(\log n).$$

Dabei bedeutet $\text{eps} = eps(q, \ell)$ *die Maschinengenauigkeit.*

Zur genaueren Analyse unserer Ergebnisse wollen wir einige numerische Experimente durchführen und uns dazu spezifische Daten x_i anschauen. Dafür generieren wir eine Folge von n Zahlen y_i, $i = 1, \ldots, n$ gemäß der Standard-Normalverteilung. Dann ist die Folge $y_1, \ldots, y_n, -y_1, \ldots, -y_n$ ebenfalls normalverteilt und hat den exakten Mittelwert 0 und die Varianz

$$v = \frac{1}{n}\sum_{i=1}^{n} y_i^2.$$

Die Berechnung der Varianz v mittels V_2 ist stabil, da v wegen der Erzeugung gemäß der Standard-Normalverteilung für ausreichend grosse n nahe an 1 liegen wird. Dann berechnen wir zur späteren Nutzung den maximalen und minimalen Betrag $y_{\max} = \max_i |y_i|$ und $y_{\min} = \min_i |y_i|$ der Folge und nehmen der Einfachheit halber für das Weitere an, dass $y_{\min} > 0$. Für beliebig gegebene $\delta \in (0,1)$ und $\bar{x} > y_{\max}$ erhalten wir daraus eine Folge positiver Zahlen

$$x_i = \bar{x} + \delta y_i, \quad x_{i+n} = \bar{x} - \delta y_i, \quad i = 1, \ldots, n,$$

mit Mittelwert $\bar{x}$ und Varianz

$$V(x_1, \ldots, x_{2n}, \bar{x}) = \delta^2 v.$$

In Algorithmus 7.26 ist angegeben, wie diese Erzeugung in MATLAB realisiert werden kann.

Algorithmus 7.26 (Normalverteilte Zahlenfolge mit gegebenem Mittelwert).

```
y=randn(1,n);
bx = 1.5*max(y);
x(1:n)=bx+delta*y;
x(n+1:2n)=bx-delta*y;
```

Beispiel 7.27. Wir erzeugen uns gemäß der gerade erarbeiteten Idee eine Datenfolge positiver, normalverteilter Zahlen $x_1, \ldots, x_{2n}$ mit $n = 10.000$, Mittelwert $\bar{x}$ und Konstanten $y_{\min} > 0$ und v wie oben eingeführt. Wir wollen studieren, wie sich die Rundungsfehler der Algorithmen 7.21 und 7.23 für verschiedene Werte von δ verhalten. Mit Hilfe unserer obigen generellen Ergebnisse ergibt sich folgende obere Schranke für den relativen Fehler der Auswertung V_1:

$$\sigma^{V_1} \leq 5 + 4n + 2\Big(\frac{\bar{x}}{y_{\min}\delta} + 1\Big)$$

$$\Big|\frac{V(x_1, \ldots, x_{2n}, \bar{x}) - \tilde{V}_1(x_1, \ldots, x_{2n}, \bar{x})}{V(x_1, \ldots, x_{2n}, \bar{x})}\Big| \leq \Big(7 + 4n + 2\frac{\bar{x}}{y_{\min}\delta}\Big)\text{eps}.$$

Ebenso für die alternative Auswertung V_2:

$$\sigma^{V_2} \leq 1 + (5+4n)(1 + 2\frac{|\bar{x}|^2}{v\delta^2})$$

$$\Big|\frac{V(x_1, \ldots, x_{2n}, \bar{x}) - \tilde{V}_2(x_1, \ldots, x_{2n}, \bar{x})}{V(x_1, \ldots, x_{2n}, \bar{x})}\Big| \leq \Big(1 + (5+4n)(1 + 2\frac{|\bar{x}|^2}{v\delta^2})\Big)\text{eps}.$$

Abbildung 7.8 zeigt die jeweiligen relativen Fehler und die zugehörigen oberen Schranken in ihrer Abhängigkeit von δ. Dabei wurde v in höherer Genauigkeit vorab berechnet. Wir erkennen, dass die oberen Schranken den Verlauf der jeweiligen Fehler im gesamten Wertebereich prinzipiell gut wiedergeben, auch wenn sie natürlich Überschätzungen darstellen. Es wird deutlich, dass die Auswertung V_2 deutlich höhere Fehler erzeugt als V_1. Ausserdem wird sie für sehr kleine δ erwartungsgemäß instabil.

Wenn wir bei der Berechnung von V_2 hierarchische Summation statt rekursiver Rekursion verwenden, dann ergibt sich ein ähnliches Bild, wie man Abb. 7.9 entnehmen kann. Nur findet man eine wesentlich bessere obere Schranke, wie Satz 7.25 vielleicht schon vermuten liess. Das zeigt, dass der Hauptanteil des Rundungsfehlers nicht aus der Summation resultiert, seine oben beobachtete Überschätzung aber schon.

Bei der Berechnung der Varianz einer Datenfolge sind die Varianten, die die Subtraktionen zuerst ausführen (Berechnung mittels Summe der Abweichungsquadrate), im Normalfall stabiler als die, die eine später Subtraktion ausführen (Berechnung mittels Summe der Datenquadrate).

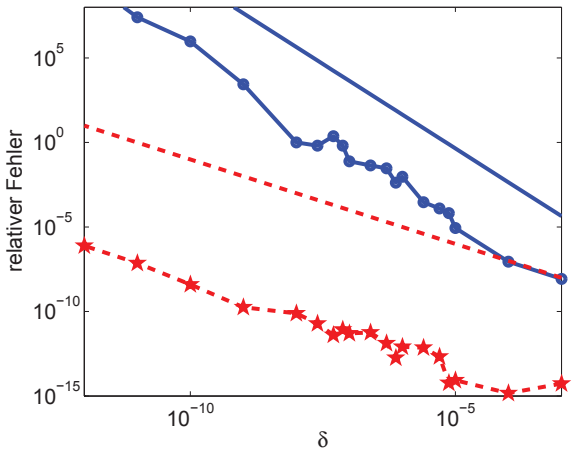

Abb. 7.8 Relativer Fehler der Varianz-Auswertungen V_1 und V_2 für die Daten aus Beispiel 7.27 in Abhängigkeit vom Parameter δ für $n = 10000$ auf einer Maschine mit Maschinengenauigkeit eps $= 2.22 \cdot 10^{-16}$. Rot, Sterne, gestrichelt: Relativer Fehler in V_1; rot, gestrichelt ohne Sterne: zugehörige obere Schranke. Blaue, durchgezogene Linie mit Kreisen: Relativer Fehler in V_2; blau-durchgezogen ohen Kreise: assoziierte obere Schranke, wie in Beispiel 7.27 berechnet.

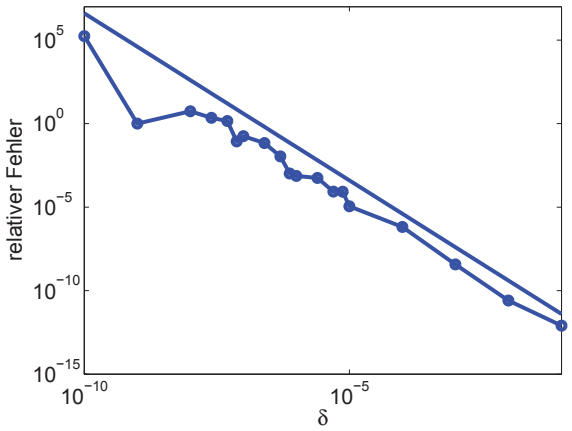

Abb. 7.9 Relativer Fehler der Varianz-Auswertungen und V_2 für die Daten aus Beispiel 7.27 in Abhängigkeit vom Parameter δ für $n = 1000$ auf einer Maschine mit Maschinengenauigkeit eps $= 2.22 \cdot 10^{-16}$. Blaue, durchgezogene Linie mit Kreisen: Relativer Fehler in V_2; blau-durchgezogen ohen Kreise: assoziierte obere Schranke. Die Berechnung von V_2 erfolgte hier mittels hierarchischer Summation, was im Vergleich zu obigem zu einer wesentlich schärferen Abschätzung des Rundungsfehlers führt.

7.6 Allgemeine Einsichten zur Rundungsfehlerabschätzung

Was können wir aus den vorangegangenen Betrachtungen lernen? Wir haben gesehen, dass unsere Abschätzungen der Stabilitätsindikatoren mittels Auswertungsbäumen in manchen Fällen eine gute Vorstellung des realen relativen Fehlers vermitteln, in anderen Fällen aber zu starken Überschätzungen führen. Was also ist der Gewinn dieser Vorgehensweise?

Der entscheidende Vorteil. Unsere *rekursive Abschätzungsvorschrift* 7.26 erlaubt uns, die Stabilitätsindikatoren der gerade durchgeführten Auswertung *parallel zur Auswertung selbst* abzuschätzen und damit zu jedem Zeitpunkt der Auswertung eine obere Schranken für den angesammelten relativen Fehler der bisher durchgeführten Berechnungen zu berechnen. Damit ergibt sich ein *Warnmechanismus* für das potentiell kritische Ausschaukeln von Rundungsfehlern in jedem Schritt der Berechnungen. Dabei setzen wir natürlich voraus, dass wir die in Zuge der Abschätzung vorkommenden Konditionsberechnung stabil ausführen können.

Der entscheidende Nachteil. Unsere Abschätzungen liefern nur eine obere Schranke und können das gegenseitige Sich-Aufheben von Rundungsfehlern nicht berücksichtigen: jeder Rundunsgfehler wird mit

dem maximal möglichen abgeschätzt und immer ihre schlimmstmögliche Kombination berücksichtigt. Im Allgemeinen werden wir auch keine genauere Schätzung des Rundungsfehlers angeben können. Genauere Betrachtungen einzelner Auswertungsklassen sind natürlich möglich.

Eine generelle Regel. Entscheidende Erhöhungen der Rundungsfehlerabschätzung erfolgen nur durch extrem schlecht konditionierte Elementaroperationen, d.h. i.A. nur durch Subtraktionen ähnlich grosser Zahlen. Also sollte man, wenn immer möglich, solche Subtraktionen vermeiden. Wir haben in Beispiel 7.27 gesehen, dass solche Subtraktionen manchmal nicht vermeidbar sind. In Beispiel 7.27 zeigt sich, dass die Auswertung, dort V_1, die die kritischen Subtraktionen möglichst frühzeitig durchführt, stabiler ist, als die Auswertung, die die kritische Subtraktion zuletzt ausführt. Damit kommen wir wieder zu unserer schon oben ausgeführten generellen Empfehlung zurück: Wähle Auswertungen, die potentiell kritische Elementaroperationen möglichst frühzeitig durchführen.

7.7 Numerisches Differenzieren

Im Hinblick auf die Definition A.4 ist es naheliegend, die Ableitung einer differenzierbaren Funktion $f : (a,b) \to \mathbb{R}$ an der Stelle $x \in (a,b)$ durch den Differenzenquotienten

$$D(h) = \frac{f(x+h) - f(x)}{h} \tag{7.43}$$

zur Schrittweite $h > 0$ zu approximieren. Dabei muss $h < H = b - x$ sein, denn sonst ist $f(x+h)$ nicht definiert. Wir interessieren uns für die Auswirkung von Störungen der Schrittweite h auf den Wert des Differenzenquotienten $D(h)$. Dazu haben wir die Kondition der Auswertung der Funktion $D : (0,H) \to \mathbb{R}$ an der Stelle $h_0 \in (0,H)$ zu untersuchen.

Wir beginnen mit der absoluten Kondition $\kappa_{\text{abs}} = \kappa_{\text{abs}}(h_0)$. Zunächst stellen wir fest, daß mit f auch D differenzierbar ist. Aus Quotienten- und Kettenregel folgt nämlich

$$D'(h) = \frac{f'(x+h)h - (f(x+h) - f(x))}{h^2} = \frac{1}{h}\left(f'(x+h) - D(h)\right) .$$

Aus Satz 6.7 erhält man nun unmittelbar das folgende Ergebnis.

Satz 7.28. *Die absolute Kondition $\kappa_{\text{abs}}(h_0)$ der Auswertung des Differenzenquotienten* (7.43) *zur Schrittweite h_0 ist*

$$\kappa_{\text{abs}}(h_0) = \frac{1}{h_0}\left|f'(x+h_0) - D(h_0)\right| . \tag{7.44}$$

Als Beispiel betrachten wir die Funktion $f(x) = x + \frac{\alpha}{2}x^2$ mit positivem $\alpha \in \mathbb{R}$ und $x = 0$. Einsetzen in (7.44) liefert

$$\kappa_{\text{abs}}(h_0) = \frac{1}{h_0}\left|1 + \alpha h_0 - (1 + \tfrac{\alpha}{2}h_0)\right| = \frac{\alpha}{2} .$$

Mit wachsendem α kann also die absolute Kondition beliebig groß werden.

Wenn wir uns für die Anzahl der gültigen Stellen von $D(h)$ interessieren, müssen wir uns die *relative Kondition* $\kappa_{\text{rel}} = \kappa_{\text{rel}}(h_0)$ ansehen.

Satz 7.29. *Es sei $D(h_0) \neq 0$. Die relative Kondition $\kappa_{\text{rel}}(h_0)$ der Auswertung des Differenzenquotienten ist dann*

$$\kappa_{\text{rel}}(h_0) = \left|\frac{f'(x+h_0)}{D(h_0)} - 1\right| . \tag{7.45}$$

Beweis. Aus Satz 6.12 folgt unmittelbar

$$\kappa_{\text{rel}}(h_0) = \frac{h_0}{|D(h_0)|}\kappa_{\text{abs}}(h_0) .$$

Einsetzen der Darstellung (7.44) liefert

$$\kappa_{\text{rel}}(h_0) = \frac{h_0}{|D(h_0)|}\frac{1}{h_0}\left|f'(x+h_0) - D(h_0)\right| = \left|\frac{f'(x+h_0)}{D(h_0)} - 1\right|$$

und damit die Behauptung. □

Im Falle $f(x) = x + \frac{\alpha}{2}x^2$ und $x = 0$ erhält man für $\kappa_{\text{rel}}(h_0)$ die Abschätzung

$$\kappa_{\text{rel}}(h_0) = \left| \frac{1+\alpha h_0}{1+\frac{\alpha}{2}h_0} - 1 \right| = h_0 \left| \frac{\alpha}{2+\alpha h_0} \right| < 1 \ .$$

Aus $h_0 \to 0$ folgt sogar $\kappa_{\text{rel}}(h_0) \to 0$. Das gilt sehr viel allgemeiner.

Satz 7.30. *Es sei f' stetig und $f'(x) \neq 0$. Dann folgt $\kappa_{\text{rel}}(h_0) \to 0$ für $h_0 \to 0$.*

Beweis. Aus der Stetigkeit von f' folgt $f'(x+h) \to f'(x)$ und aus der Definition der Ableitung folgt $D(h) \to f'(x)$ für $h \to 0$. Wegen $f'(x) \neq 0$ muss für genügend kleine h auch $D(h) \neq 0$ gelten. Insgesamt erhält man für $h_0 \to 0$

$$\kappa_{\text{rel}}(h_0) = \left| \frac{f'(x+h)}{D(h)} - 1 \right| \to \left| \frac{f'(x)}{f'(x)} - 1 \right| = 0 \ .$$

□

Für immer kleinere Schrittweiten h_0 spielt nach Satz 7.30 die Anzahl der gültigen Stellen von h_0 kaum noch eine Rolle für die gültigen Stellen des Differenzenquotienten $D(h)$. In diesem Sinne ist also die Auswertung des Differenzenquotienten $D(h_0)$ für $h_0 \to 0$ beliebig gut konditioniert.

Wir wollen nun die Probe aufs Exempel machen und dazu die Approximation der Ableitung von $f(x) = \sin(x)$ an der Stelle $x = 0.6$ wirklich ausrechnen. Offenbar ist $f'(x) = \cos(x)$ stetig und $f'(x) = \cos(x) \neq 0$ für $x = 0.6$. Für die Schrittweiten $h_k = 10^{-k}$ und $k = 1, \ldots, 15$ wird nun der Differenzenquotient durch direkte Umsetzung von (7.43) in ein MATLAB–Programm ausgewertet. Die relativen Diskretisierungsfehler relative Abweichung der Ergebnisse $\tilde{D}(h_k)$, $k = 1, \ldots, 15$, von der Ableitung $\cos(0.6)$ ist in Figur 7.10 dargestellt. Offenbar verbessert sich zunächst durch Verkleinerung der Schrittweite die Ge-

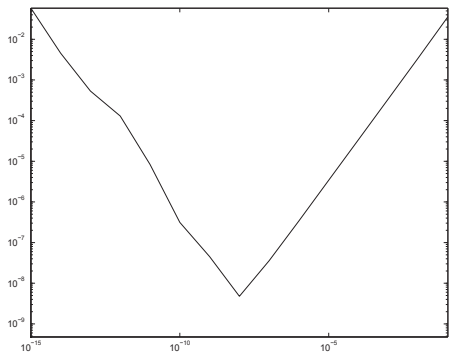

Abb. 7.10 Relativer Fehler der Differenzenapproximation in Abhängigkeit von der Schrittweite

nauigkeit, um dann aber jenseits eines Schwellwerts von ungefähr 10^{-8} wieder schlechter zu werden. Die berechnete Differenzenapproximation zur Schrittweite $h_{15} = 10^{-15}$ ist schließlich schlechter als für $h_1 = 10^{-1}$. Da nach Satz 7.30 Störungen von h keine Rolle spielen und definitionsgemäß $D(h) \to f'(x)$ gilt, könnten die großen Abweichungen für kleine Schrittweiten auf mangelnde Stabilität zurückzuführen sein. Das wollen wir genauer untersuchen.

Wir gehen der Einfachheit halber wieder zur Bezeichnung h für das Inkrement im Differenzenquotienten über und betrachten nun die direkte Umsetzung von (7.43) durch den Algorithmus

$$D(h) = g_3(g_2(f(g_1(x,h)), f(x)), h) \tag{7.46}$$

mit

$$g_1(x,h) = x+h, \quad g_2(a,b) = a-b, \quad g_3(c,h) = c/h \ .$$

Wir wollen nun die Stabilität von (7.46) für kleine Werte von h abschätzen. Dazu betrachten wir zuerst einmal x und h als gestörte Eingabewerte und den (7.46) gehörenden Auswertungsbaum

$$\begin{aligned}
\beta_0 &= [[],[]], \quad z_0 = f^{\beta_0}(x,h) = g_1(x,h) = x+h \\
\beta_1 &= [\beta_0], \quad z_1 = f^{\beta_1}(z_0) = f(z_0) \\
\beta_2 &= [[]], \quad z_2 = f^{\beta_2}(x) = f(x) \\
\beta_3 &= [\beta_1,\beta_2], \quad z_3 = f^{\beta_3}(z_1,z_2) = g_2(z_1,z_2) = z_1 - z_2 \\
\beta_4 &= [\beta_3,[]], \quad z_4 = f^{\beta_4}(z_3,h) = g_3(z_3,h) = z_3/h.
\end{aligned}$$

Die rekursive Berechnung der Stabilitätsindikatoren liefert dann

$$\sigma_4 \leq 1 + 2(1 + \kappa_-(1 + 2\kappa_f)) \tag{7.47}$$

mit den beiden relativen Konditionen

$$\kappa_- = \frac{|f(x)| + |f(x+h)|}{|f(x+h) - f(x)|}, \quad \kappa_f = \frac{|f'(x)| \cdot |x|}{|f(x)|}.$$

Damit haben wir folgendes Resultat:

Satz 7.31. *Sei f mindestens einmal stetig differenzierbar in x und $f(x) = 0$ sowie $f'(x) = 0$. Dann gilt für den Rundungsfehler bei der direkten Berechnung des Differenzenquotienten:*

$$\frac{|D(h) - \tilde{D}(h)|}{|D(h)|} \leq \Big(1 + 2(1 + \kappa_-(1 + 2\kappa_f))\Big)\,\text{eps}. \tag{7.48}$$

Wenn f zweimal differenzierbar in x ist, dann verhält sich dieser Fehler für $h \to 0$ wie

$$\frac{|D(h) - \tilde{D}(h)|}{|D(h)|} \leq C + 4\frac{\text{eps}}{h}\left(2|x| + \frac{|f(x)|}{|f'(x)|}\right) + \mathcal{O}(h), \tag{7.49}$$

wobei C eine von h unabhängige Konstante ist.

Beweis. Die erste Aussage ergibt sich sofort aus (7.47). Die zweite Behauptung resultiert aus Taylorentwicklung von $f(x+h)$ um $f(x)$ in h, was sofort

$$\kappa_- \leq C_1 + \frac{2}{h}\frac{|f(x)|}{|f'(x)|} + \mathcal{O}(h)$$

mit einer von h unabhängigen Konstante C_1 liefert. Weiteres Einsetzen in (7.47) liefert die Behauptung.

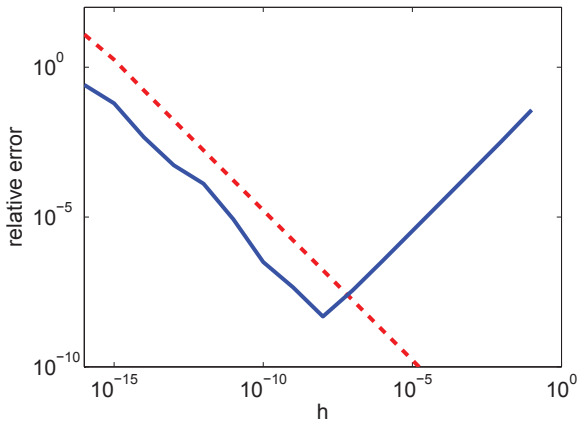

Abb. 7.11 Relativer Fehler der direkten Berechnung des Differenzenquotienten. Die durchgezogene Linie zeigt den Verlauf des real auftretenden relativen Fehlers $|\tilde{D}(h) - f'(x)|/|f'(x)|$ und die gestrichelte Linie die des sich aus der Stabilitätsabschätzung (7.47) ergebenden Rundungsfehler anteils $\sigma_4$eps.

In Abbildung 7.11 sind numerische Ergebnisse zum Studium dieses Fehlers gezeigt. In numerischen Rechnungen haben wir aber normalerweise keinen Zugang zu $D(h)$, sondern nur zu $\tilde{D}(h)$. Daher können

wir hier nur den real auftretenden relativen Fehler

$$\frac{|\tilde{D}(h)-f'(x)|}{|f'(x)|} \leq \Big(\frac{|D(h)-\tilde{D}(h)|}{|D(h)|}+\frac{|D(h)-f'(x)|}{|D(h)|}\Big)\cdot\frac{|D(h)|}{|f'(x)|}$$

numerisch berechnen. Der zweite Fehleranteil auf der rechten Seite der letzten Abschätzung ist allerdings genau die relative Kondition, von der wir wissen, dass sie für $h \to 0$ verschwindet, was auch $D(h) \to f'(x)$ und somit $D(h)/f'(x) \to 1$ impliziert. Also wird der real auftretende relative Fehler für ausreichend kleine h mit dem in (7.48) betrachteten in sehr guter Genauigkeit übereinstimmen. Genau das sehen wir in Abbildung 7.11: Für $h < 10^{-9}$ verhält sich der real auftretende relative Fehler wie unsere Abschätzung und steigt für kleiner werdendes h wie eps/h an, während für grössere Werte von h der Rundungsfehler nicht dominiert, sondern der Unterscheid zwischen $D(h)$ und $f'(x)$, also der zweite Fehleranteil der obigen Abschätzung, der sich im Wesentlichen wie $\mathscr{O}(h)$ verhält. Die beiden Fehleranteile führen dazu, dass sich ein Minimum des real auftretenden Fehlers bei $h \approx \sqrt{\mathrm{eps}}$ ergibt, das die generelle Grössenordnung $\sqrt{\mathrm{eps}}$ nicht unterschreitet.

Die numerische Approximation der ersten Ableitung durch direkte Berechnung des Differenzenquotienten in Gleitkomma-Arithmetik mit Maschinengenauigkeit eps ist im Allgemeinen auf eine Genauigkeit der Grössenordnung $\sqrt{\mathrm{eps}}$ beschränkt.

Wie wir schon in Abschnitt 7.2 gesehen haben, besteht ein möglicher Ausweg in einer *stabilen Umformung des Differenzenquotienten.* Dazu verwenden wir nacheinander das Additionstheorem $\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta)+\sin(\beta)\cos(\alpha)$, die binomischen Formel $(\alpha+\beta)(\alpha-\beta) = \alpha^2-\beta^2$ und $\sin^2(\alpha)+\cos^2(\alpha)=1$, $\alpha,\beta\in\mathbb{R}$. So erhält man

$$\begin{aligned}
\frac{\sin(x+h)-\sin(x)}{h} &= \frac{1}{h}\big(\sin(x)(\cos(h)-1)+\cos(x)\sin(h)\big)\\
&= \frac{1}{h}\left(\sin(x)\,\frac{\cos^2(h)-1}{\cos(h)+1}+\cos(x)\sin(h)\right)\\
&= \frac{1}{h}\left(-\sin(x)\frac{\sin^2(h)}{\cos(h)+1}+\cos(x)\sin(h)\right)\\
&= \frac{\sin(h)}{h}\left(\cos(x)-\sin(h)\,\frac{\sin(x)}{\cos(h)+1}\right).
\end{aligned}$$

Dieser Ausdruck ist zwar erheblich komplizierter, führt dafür aber auf einen stabilen Algorithmus der prinzipiellen Form

$$D_2(h) = g_1(h)\cdot g_2(x,h),\quad g_1(h)=\frac{\sin(h)}{h},\quad g_2(x,h)=\cos(x)-\sin(h)\,\frac{\sin(x)}{\cos(h)+1}.$$

Auswertung zuerst von g_1, dann g_2 und dann der Multiplikation ergibt sofort die Stabilitätsabschätzung

$$\sigma \leq 1+\kappa_{g_1}(1+\kappa_{sin})) = 1+\Big|\frac{h\cos h}{\sin h}-1\Big|\Big(1+\Big|\frac{\cos(h)}{\sin h}\Big|\cdot h^2\Big),$$

da bei der Auswertung von g_2 keine Auslöschung auftritt. Daraus erhalten wir für kleine h sofort:

$$\sigma \leq 1+\mathscr{O}(h^2).$$

Bei dieser Berechnung wird also kein signifikanter Rundungsfehleranteil auftreten. Genau das ergeben auch die in Abbildung 7.12 illustrierten numerischen Experimente, wobei dieselben Überlegungen betreffs der verschiedenen Fehleranteile anzustellen sind, wie weiter oben für die direkte Berechnung des Differenzenquotienten.

Leider hat die stabile Umformung des Differenzenquotienten $D_2(h)$ große Nachteile. Abgesehen davon, daß man über jeden Fall neu nachdenken muß lässt sich diese Technik nur anwenden, wenn die Funktion f, wie in unserem Beispiel, in geschlossener Form vorliegt. Dann wird man aber im allgemei-

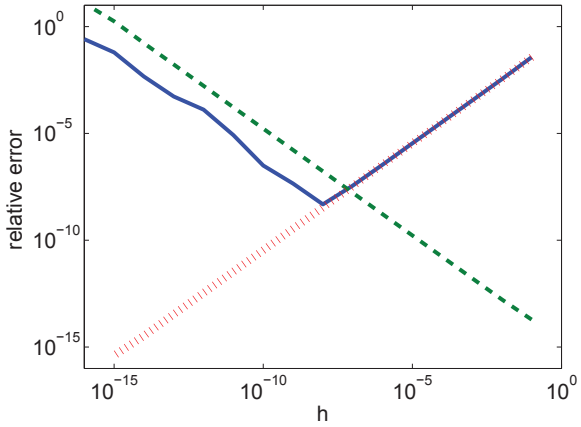

Abb. 7.12 Vergleich der relativer Fehler der direkten und stabilen Berechnung des Differenzenquotienten für $f(x) = \sin(x)$. Die durchgezogene und die dünne gestrichelte Linie zeigen wie in Abb. 7.11 den realen Fehler und den zugehörigen Rundungsfehleranteil der direkten Berechnung des Differenzenquotienten. Die dickere Strich-Punkt Linie zeigt den relativen Fehler $|\tilde{D}_2(h) - f'(x)|/|\tilde{D}_2(h)|$ der stabilen Berechnung des Differenzenquotienten.

nen die Ableitung f' lieber exakt berechnen, notfalls mit Unterstützung eines Symbolik-Programms und nicht umständlich durch Differenzenquotienten oder gar deren stabile Umformung approximieren. Differenzenapproximationen haben ihre Bedeutung, wenn exaktes Differenzieren nicht möglich ist, etwa weil nur Funktionswerte von f verfügbar sind, beispielsweise aus einem Computerprogramm. Gerade dann können wir aber auch keine stabile Umformungen vornehmen. Ein Ausweg aus diesem Dilemma besteht in einem völlig anderen Verfahren, dem sogenannten *automatisches Differenzieren*. Dieses Verfahren ist anwendbar, wenn f durch einen Programmcode erzeugt wird und beruht auf einer sukzessiven numerischen Umsetzung der elementaren Ableitungsregeln. Einzelheiten dazu finden sich in dem Standardwerk von Griewank und Walther [15].

7.8 Drei-Term-Rekursionen

Wir betrachten nun wiederum die Drei-Term-Rekursion (6.32) aus Abschnitt 6.5 mit vorgegebenen Anfangswerten x_{-1} und x_0 und Koeffizienten $a, b \in \mathbb{R}$, die so gewählt sind, dass das Polynom $\lambda^2 + a\lambda + b$ zwei reelle Nullstellen λ_1, λ_2 hat. Wir nehmen an, dass

$$q = \frac{\lambda_1}{\lambda_2}, \quad |q| < 1,$$

dass die zwei Nullstellen also nicht zusammenfallen. Die Lösung ergab sich dann zu

$$x_k = \alpha\lambda_1^{k+1} + \beta\lambda_2^{k+1}, \tag{7.50}$$

mit Konstanten α, β, die nur von den Startwerten und den Eigenwerten abhängen.

7.8.1 Rekursive Auswertung

Ausnutzung der rekursiven Struktur führt unmittelbar auf die folgende Berechnungsvorschrift zur Auswertung von $f_k(x_0) = x_k$.

Algorithmus 7.32 (Rekursive Auswertung einer Drei-Term-Rekursion).

```
function f = REKDT(x0,K)

  x(-1) = xM;
```

```
    x(0) = x0;

    for k = 0:K-1
      x(k+1) = -a*x(k)-b*x(k-1);
    end

    f=x(K);

return
```

Gegenüber der geschlossenen Darstellung aus Lemma 6.23 hat Algorithmus 7.32 den Vorteil, daß er auch auf *inhomogene Drei-Term-Rekursionen mit variablen Koeffizienten* der Form (6.30) angewandt werden kann.

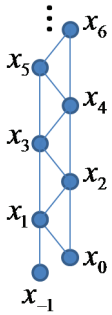

Abb. 7.13 Auswertungsgraph der 3-Term-Rekursion zum rekursiven Algorithmus 7.32. In jedem Knoten, der kein Blatt ist, führen wir die Funktion $x_{k+1} = f^{\beta_{k+1}}(x_k, x_{k-1}) = -(ax_k + bx_{k-1})$ aus.

Der in Abb. 7.13 dargestellte Auswertungsgraph zum Algorithmus 7.32 ergibt für die rekursive Abschätzung der Stabilitätsindikatoren

$$\sigma_{k+1} = \sigma^{f^{\beta_{k+1}}} \leq 1 + \kappa_{k+1} \max\{\sigma_k, \sigma_{k-1}\}, \quad \kappa_{k+1} = \frac{|a||x_k| + |b||x_{k-1}|}{|ax_k + bx_{k-1}|}$$

mit Startwerten $\sigma_0 = \sigma_{-1} = 1$. Da $\kappa_k \geq 1$ für alle $k \geq 1$ ergibt sich sofort $\sigma_{k+1} > \sigma_k$ für alle $k \geq 0$. Daher vereinfacht sich die rekursive Abschätzung zu

$$\sigma_{k+1} \leq 1 + \kappa_{k+1}\,\sigma_k, \quad \kappa_{k+1} = \frac{|a||x_k| + |b||x_{k-1}|}{|x_{k+1}|} \tag{7.51}$$

mit $\sigma_0 = 1$. Diese Abschätzung kann man trivial parallel zur eigentlichen Drei-Term Rekursion ausführen. Da diese Ausführung aber ebenfalls Rundungsfehlern unterliegen wird, hätten wir gerne grundsätzliche Einsicht in das Wachstumsverhalten der σ_k. Zu diesem Zwecke bemerken wir zuerst, dass

$$\sigma_k \leq \xi_k, \quad \text{mit} \quad \xi_{k+1} = 1 + \kappa_{k+1}\,\xi_k, \quad \xi_0 = 1.$$

Für die ξ_k findet man schnell eine explizite Formel (siehe Lemma 7.7):

$$\sigma_k \leq \xi_k = 1 + \sum_{i=1}^{k} \prod_{j=i}^{k} \kappa_j. \tag{7.52}$$

Wir müssen uns also mit dem Wachstumsverhalten der κ_j beschäftigen, um das Wachstumsverhalten der Stabilitätsindikatoren σ_k grundsätzlich studieren zu können.

Zum näheren Studium der κ_k bemerken wir zuerst, dass aus der Definition der λ_1, λ_2 sofort folgt, dass

$$a = -(\lambda_1 + \lambda_2), \quad b = \lambda_1 \lambda_2.$$

Dieses zusammen mit der Lösungsformel für die x_k ausnutzend, finden wir mit $q = \lambda_1/\lambda_2$:

$$\begin{aligned}\kappa_{k+1} &= \frac{|a||x_k| + |b||x_{k-1}|}{|x_{k+1}|} \\ &= \frac{|1+q| \cdot |\alpha q^{k+1} + \beta| + |q| \cdot |\alpha q^k + \beta|}{|\alpha q^{k+2} + \beta|}\end{aligned} \tag{7.53}$$

Um das Wachstumsverhalten zu verstehen, betrachten wir zuerst einmal zwei Spezialfälle:

(A) $\beta = 0$:
Es ergibt sich wegen $|q| < 1$ sofort, dass

$$\kappa_{k+1} = \frac{|1+q|+1}{|q|} = \begin{cases} 2|q|^{-1} - 1 & \text{falls } q \le 0 \\ 2|q|^{-1} + 1 & \text{falls } q > 0 \end{cases}.$$

Dann liefert unsere Formel für κ_k nach Einsetzen in (7.52) mit $\rho = (|1+q|+1)/|q| > 1$ sofort

$$\sigma_k \le 1 + \sum_{i=1}^{k} \rho^{k-i+1} = 1 + \sum_{i=1}^{k} \rho^i = \frac{\rho}{\rho - 1}(\rho^k - 1) \le C\rho^k, \tag{7.54}$$

mit einer von k unabhängigen Konstante C. Das ist eine exponentiell wachsende obere Schranke! Wir müssen also erwarten, dass der Rundungsfehler dieser Berechnung der 3-Term-Rekursion mit k *exponentiell wächst* und ab einem bestimmten Rekursionsindex k_0 das Ergebnis komplett dominiert und damit völlig unbrauchbar macht.

(B) $\alpha = 0$ und $q \le 0$:
Hier finden wir sofort $\kappa_k = |1+q| + |q| = 1$ und daher wegen (7.52) sofort $\sigma_k \le 1 + \sum_{i=1}^{k} 1 = k+1$. Das ist eine lediglich *linear* wachsende obere Schranke! Der Rundungsfehler der 3-Term-Rekursion wird also in diesem Fall bis zu sehr grossen k keine signifikante Rolle spielen.

Beispiel 7.33. Als erstes Beispiel betrachten wir die Rekursion

$$4x_{k+1} - 4x_k - 3x_{k-1} = 0, \quad x_{-1} = 1 \tag{7.55}$$

mit $\lambda_2 = 3/2$ und $\lambda_1 = -1/2$ und somit $q = -1/3 < 0$. Zuerst betrachten wir $\beta = 0$, indem wir $x_0 = \lambda_1$ setzen und dann $\alpha = 0$ mittels $x_0 = \lambda_2$. Die Abbildungen 7.14 und 7.15 zeigen numerische Experimente zu diesen beiden Fällen. In beiden bestätigt sich das oben analytisch Gefundene: Für $\beta = 0$ kann der Fehler exponentiell schnell wachsen und schon für recht kleine k die Grössenordnung 1 erreichen, während für $\alpha = 0$ kein signifikanter Rundungsfehler beobachtet wird.

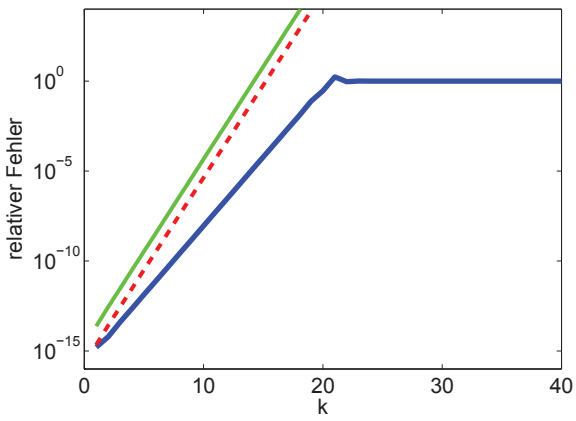

Abb. 7.14 Relativer Fehler bei der direkten Auswertung der Dreiterm-Rekursion (7.55) mit $\beta = 0$ in **logarithmischer** Skala. Die beiden (fast) parallelen Linien zeigen die Abschätzung des Rundungsfehlers laut (7.52) (gestrichelte Linie) und (7.54) (durchgezogene Linie). Die darunter verlaufende durchgezogene Linie zeigt den realen relativen Fehler $|\tilde{x}_k - x_k|/|\tilde{x}_k|$ mit x_k laut (7.50). Das exponentielle Fehlerwachstum sowohl für den realen Fehler als auch für die Fehlerschranken ist deutlich sichtbar.

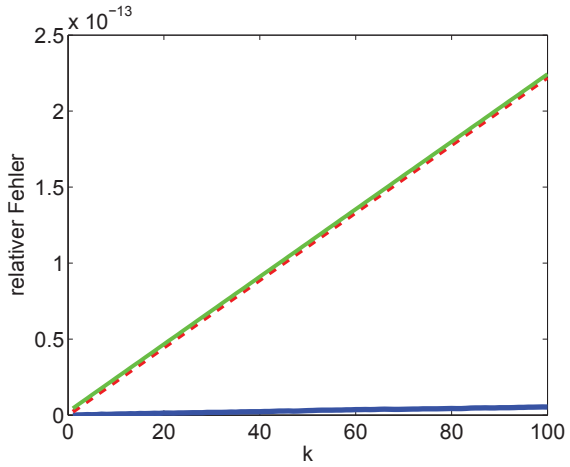

Abb. 7.15 Relativer Fehler bei der direkten Auswertung der Dreiterm-Rekursion (7.55) mit $\alpha = 0$ in **linearer** Skala. Die beiden (fast) parallelen Linien zeigen die Abschätzung des Rundungsfehlers laut (7.52) (gestrichelte Linie) und (7.54) (durchgezogene Linie). Die darunter verlaufende durchgezogene Linie zeigt den realen relativen Fehler $|\tilde{x}_k - x_k|/|\tilde{x}_k|$ mit x_k laut (7.50). Das sehr beschränkte Fehlerwachstum sowohl für den realen Fehler als auch für die Fehlerschranken ist deutlich sichtbar.

Wir wollen die Stabilitätseigenschaften von Algorithmus 7.32 nun genauer untersuchen. Wir hätten gerne eine analytische Bestätigung dafür, dass der Fall $\beta = 0$ zu exponentiellem Rundungsfehlerwachstum führt. Das werden wir dadurch erreichen, dass wir statt einer oberen Schranke für den Fehler eine exponentiell wachsende *untere* Schranke beweisen. Als Nebenresultat dieser Untersuchung werden wir sehen, dass unsere obige Beobachtung sehr schwachen Fehlerwachstums für $\alpha = 0$ in gewissem Sinne auf den gesamten Fall $\beta \neq 0$ übertragen werden kann.

Um unsere Untersuchung in einem übersichtlichen Rahmen zu halten, berücksichtigen wir der Einfachheit halber nur die Rundung

$$\tilde{x}_{k+1} = \mathrm{rd}(x_{k+1}) = (-a\tilde{x}_k - b\tilde{x}_{k-1})(1+\varepsilon_k)\,, \qquad |\varepsilon_k| \le eps\,. \tag{7.56}$$

Mögliche Rundungsfehler oder gar Auslöschung bei der Berechnung von $-a\tilde{x}_k - b\tilde{x}_{k-1}$ werden ignoriert. Die gestörte Drei-Term-Rekursion (7.56) hat somit die variablen Koeffizienten $a_k = a(1+\varepsilon_k)$, $b_k = b(1+\varepsilon_k)$. Um den technischen Aufwand im Rahmen zu halten, nehmen wir vereinfachend an, daß in jedem Schritt derselbe Rundungsfehler ε gemacht wird, also

$$\varepsilon = \varepsilon_k\,, \quad |\varepsilon| \le eps \qquad \forall k = 0, 1, \ldots\,.$$

Dann wird aus (7.56) die Drei-Term-Rekursion

$$\tilde{x}_{k+1} = -a(1+\varepsilon)\tilde{x}_k - b(1+\varepsilon)\tilde{x}_{k-1}\,, \qquad \tilde{x}_1 = x_{-1}\,,\ \tilde{x}_0 = x_0\,, \tag{7.57}$$

mit *konstanten Koeffizienten*. Das zugehörige *gestörte* charakteristische Polynom

$$\lambda^2 + \lambda a(1+\varepsilon) + b(1+\varepsilon)$$

hat die ε-abhängigen Nullstellen $\lambda_1(\varepsilon)$, $\lambda_2(\varepsilon)$. Offenbar sind dabei $\lambda_1(0)$, $\lambda_2(0)$ gerade die Nullstellen des exakten charakteristischen Polynoms (6.33). Unter der Voraussetzung $|\lambda_2(0)| > |\lambda_1(0)|$ aus Abschnitt 6.5 gilt ebenfalls $|\lambda_2(\varepsilon)| > |\lambda_1(\varepsilon)|$, falls $|\varepsilon|$ genügend klein ist. Wir nehmen an, daß

$$|\lambda_2(\varepsilon)| > |\lambda_1(\varepsilon)|\,, \qquad |\varepsilon| \le eps\,, \tag{7.58}$$

vorliegt. Dann ist nach Lemma 6.23 die gestörte Lösung $\tilde{x}_k$ gegeben durch

$$\begin{aligned} \tilde{x}_k &= x_k(\varepsilon) = \alpha(\varepsilon)(\lambda_1(\varepsilon))^{k+1} + \beta(\varepsilon)(\lambda_2(\varepsilon))^{k+1}\,, \\ \alpha(\varepsilon) &= \frac{\lambda_2(\varepsilon)x_{-1} - x_0}{\lambda_2(\varepsilon) - \lambda_1(\varepsilon)}\,, \quad \beta(\varepsilon) = x_{-1} - \alpha(\varepsilon)\,. \end{aligned} \tag{7.59}$$

Die so definierte Funktion

$$x_k : [-eps, eps] \ni \varepsilon \mapsto x_k(\varepsilon) \in \mathbb{R}$$

liefert für $\varepsilon = 0$ die exakte Lösung $x_k(0)$ der Drei-Term-Rekursion (6.32). In der Notation der vorausgegangenen Abschnitte 6.5 und 7.2 bedeutet das

$$f_k(x_0) = x_k(0) , \qquad \tilde{f}_k(\varepsilon, x_0) = x_k(\varepsilon) .$$

Offenbar ist $x_k(\varepsilon)$ differenzierbar in $\varepsilon = 0$. Damit können wir die Stabilität σ_k von Algorithmus 7.32 direkt angeben. Nach Definition der Ableitung gilt nämlich

$$\lim_{\varepsilon \to 0} \frac{x_k(\varepsilon) - x_k(0)}{\varepsilon} = x_k'(0)$$

und gleichbedeutend

$$x_k(\varepsilon) - x_k(0) = x_k'(0)\varepsilon + o(\varepsilon) .$$

Einsetzen ergibt

$$\frac{|f_k(x_0) - \tilde{f}_k(\varepsilon, x_0)|}{|f_k(x_0)|} = \frac{|x_k'(0)|}{|x_k(0)|}|\varepsilon| + o(\varepsilon) .$$

In direkter Analogie zu Satz 6.7 erhalten wir also

$$\sigma_k = \frac{|x_k'(0)|}{|x_k(0)|} .$$

Nun können wir das Hauptresultat dieses Abschnitts formulieren.

Satz 7.34. *Es sei $x_0 \neq 0$, $f_k(x_0) = x_k(0) \neq 0$ und es sei* (7.58) *erfüllt. Unter der Voraussetzung $x_0 \neq \lambda_1 x_{-1}$ gilt dann für genügend große k die Abschätzung*

$$\sigma_k \leq C\, k$$

mit einer Konstanten C, die nicht von k abhängt.
Im Falle $x_0 = \lambda_1 x_{-1}$ gilt für genügend große k

$$\sigma_k \geq c \left| \frac{\lambda_2}{\lambda_1} \right|^k$$

mit einer k-unabhängigen Konstanten c.

Beweis. Wir schreiben kurz $\lambda_i = \lambda_i(0)$ und $\lambda_i' = \lambda_i'(0)$, $i = 1, 2$, sowie $\alpha = \alpha(0)$, $\alpha' = \alpha'(0)$ und $\beta = \beta(0)$, $\beta' = \beta'(0)$. Außerdem setzen wir

$$q = \frac{\lambda_1}{\lambda_2} .$$

Wir beginnen mit dem Fall $x_0 \neq \lambda_1 x_{-1}$, nehmen also an, daß $\beta = (x_0 - \lambda_1 x_{-1})/(\lambda_2 - \lambda_1) \neq 0$ vorliegt. Mit Produkt- und Kettenregel folgt dann

$$\begin{aligned} x_k'(0) &= \alpha' \lambda_1^{k+1} + (k+1)\alpha \lambda_1^k \lambda_1' + \beta' \lambda_2^{k+1} + (k+1)\beta \lambda_2^k \lambda_2' \\ &= \lambda_2^k \left((\alpha' \lambda_1 + (k+1)\alpha \lambda_1') q^k + \beta' \lambda_2 + (k+1)\beta \lambda_2' \right) . \end{aligned}$$

Berücksichti¡gt man nun $|q| < 1$ und $k|q|^k \to 0$ für $k \to \infty$, so erhält man für genügend großes k die Abschätzung

$$|x_k'(0)| \leq c_1 k |\lambda_2|^k$$

mit einer geeigneten k-unabhängigen Konstanten c_1. Außerdem erhalten wir etwa aus (7.59)

$$x_k(0) = \alpha \lambda_1^{k+1} + \beta \lambda_2^{k+1} = \lambda_2^k (\alpha \lambda_1 q^k + \beta \lambda_2) .$$

Aus $\lambda_2 \neq 0$, $\beta \neq 0$ und $|q| < 1$ folgt damit für genügend großes k die Abschätzung

$$|x_k(0)| = |\lambda_2|^k \left| |\beta\lambda_2| - |\alpha\lambda_1||q|^k \right| \geq c_2|\lambda_2|^k > 0$$

mit einer geeigneten k-unabhängigen Konstanten c_2. Einsetzen ergibt

$$\frac{|x_k'(0)|}{|x_k(0)|} \leq \frac{c_1 k}{c_2} = Ck$$

und damit die Behauptung.

Nun betrachten wir den Fall $\beta = 0$. Aus (7.59) erhält man unmittelbar $\alpha = x_{-1}$ und wegen $x_0 \neq 0$ ergibt sich daraus $\alpha \neq 0$ und $\lambda_1 \neq 0$. Etwas langwieriger ist der Nachweis von $\lambda_1' \neq 0$. Insgesamt erhält man schließlich $\beta' \neq 0$. Damit können wir $|x_k'(0)|$ nach unten abschätzen. Für genügend große k gilt

$$\begin{aligned} |x_k'(0)| &= |\lambda_2|^k \left| (\alpha'\lambda_1 + (k+1)\alpha\lambda_1')q^k + \beta'\lambda_2 \right| \\ &\geq |\lambda_2|^k \left| |\beta'\lambda_2| - |\alpha'\lambda_1 + (k+1)\alpha\lambda_1'||q|^k \right| \geq c_3|\lambda_2|^k > 0 \end{aligned}$$

mit einer geeigneten k-unabhängigen Zahl $c_3 > 0$. Zur Abschätzung von $|x_k(0)|$ verwenden wir die Darstellung (6.36) aus dem Beweis von Satz 6.24. Mit elementaren Umformungen folgt

$$|x_k(0)| = |\lambda_2|^k \left| \frac{1-q^{k+1}}{1-q} \right| |x_0||q|^k \left| \frac{1-q}{1-q^{k+1}} \right| = |x_0||\lambda_2|^k|q|^k \,.$$

Zusammengenommen führen diese beiden Abschätzungen auf

$$\sigma_k = \frac{|x_k'(0)|}{|x_k(0)|} \geq \frac{c_3}{|x_0|}|q|^{-k} = c\left|\frac{\lambda_2}{\lambda_1}\right|^k \,.$$

Damit ist alles bewiesen. □

Im *generischen Fall* $x_0 \neq \lambda_1 x_{-1}$ oder, gleichbedeutend, $\beta \neq 0$ wächst also die Auswirkung der Störung in jedem Schritt proportional zur Anzahl k der Schritte. Auch wenn die Konstante C groß sein kann, so ist das ein zumindest qualitativ akzeptables Verhalten. Wir nennen Algorithmus 7.32 daher *stabil*. Diese Einschätzung wollen wir anhand eines weiteren numerischen Beispiels überprüfen. Dazu betrachten wir erneut die Drei-Term-Rekursion (6.38)

$$4x_{k+1} - 4x_k - 3x_{k-1} = 0 \,, \qquad x_{-1} = 1$$

mit dem Anfangswert $x_0 = 1$ aus Abschnitt 6.5, was zwar zu $\beta \neq 0$ aber nicht zu $\alpha = 0$ führt, also von den weiter oben betrachteten Beispielen zu unterscheiden ist.

k	10	100	1000	1500
rel. Fehler	0	0	$0.1 \cdot 10^{-16}$	$0.1 \cdot 10^{-16}$

Tabelle 7.1 Rekursive Auswertung der Drei-Term-Rekursion im generischen Fall

Die relativen Fehler bleiben durchweg im Bereich der Maschinengenauigkeit und bestätigen damit die Stabilität von Algorithmus 7.32 im Falle $\beta \neq 0$.[7]

Ist $\beta = 0$, so haben wir es mit der Berechnung einer schlecht konditionierten, rezessiven Lösung zu tun (vgl. Abschnitt 6.5). Nach Satz 7.34 wächst dann die Stabilität selbst im *besten* Fall exponentiell! Algorithmus 7.32 ist *instabil*. Was das bedeutet, haben wir in unserem ersten Beispiel schon eindrucksvoll gesehen, siehe Abbildung 7.14. Schon bei $k = 20$ setzt sich der dominante Lösungszweig durch und macht die Ergebnisse völlig unbrauchbar.

Zur Vorbereitung des nächsten Schritts wollen wir nochmals den Fall $\beta = 0$ für eine andere als die in Beispiel 7.33 betrachteten Dreiterm-Rekursion studieren:

$$x_{k+1} + 2x_k - x_{k-1} = 0 \,, \qquad x_{-1} = 1 \,, \quad x_0 = \sqrt{2} - 1 \,,$$

[7] Angesichts von nur 15 verfügbaren Stellen wächst der absolute Fehler allerdings auf über 10^{160} an.

mit der rezessiven Lösung $x_k = \lambda_1^{k+1}$, $\lambda_1 = \sqrt{2}-1$. Wie in Abbildung 7.16 zu sehen ist, macht sich die Instabilität schon nach etwa 40 Schritten verheerend bemerkbar. Algorithmus 7.32 ist zur Approximation

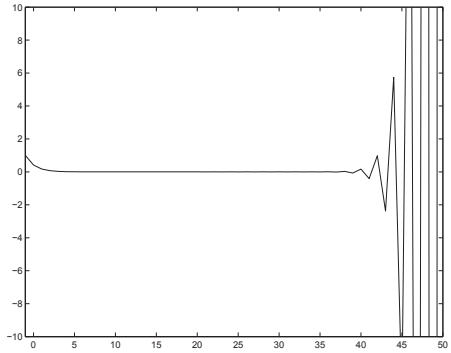

Abb. 7.16 Instabilitäten von Algorithmus 7.32 bei rezessiven Lösungen. Gezeigt sind die berechneten Folgeglieder $\tilde{x}_k$ aufgetragen gegen k. Die analytische Lösung $x_k = (\sqrt{2}-1)^{k+1}$ fällt exponentiell mit k.

rezessiver Lösungen nicht zu gebrauchen! Das zeigt unsere theoretische Analyse, und das bestätigen auch die numerischen Rechnungen.

Die Berechnung rezessiver Lösungen von Dreiterm-Rekursionen mittels Vorwärtsrekursion ist instabil, die der dominanten Lösungen stabil.

Was tun? Die geschlossene Darstellung aus Lemma 6.23 macht zwar keinerlei Stabilitätsprobleme, ist aber auf konstante Koeffizienten beschränkt. Im Falle variabler Koeffizienten müssen zur Berechnung rezessiver Lösungen spezielle, aufwendigere Algorithmen verwendet werden. Darum geht es im nächsten Abschnitt.

7.8.2 Der Algorithmus von Miller

Wir betrachten nach wie vor die Drei-Term-Rekursion (6.32) aus Abschnitt 6.5, konzentrieren uns dabei aber ausschließlich auf die Berechnung rezessiver Lösungen, also auf den Fall

$$x_0 = \lambda_1 x_{-1}\,, \qquad |\lambda_2| > |\lambda_1| > 0\,. \tag{7.60}$$

Dabei sind x_{-1}, x_0 die gegebenen Anfangswerte und λ_1, λ_2 die Nullstellen des zugehörigen charakteristischen Polynoms 3.6. Mit Hilfe von Lemma 6.23 können wir dann zwar zu jedem x_0 die exakte Lösung

$$x_k = x_0 \lambda_1^k\,, \qquad k = 1, \ldots, \tag{7.61}$$

direkt angeben, allerdings nur für konstante Koeffizienten. Auf der anderen Seite versagt der allgemein anwendbare rekursive Algorithmus 7.32 bei rezessiven Lösungen (6.39) wegen mangelnder Stabilität. Das haben wir im vorigen Abschnitt gesehen.

Einen Ausweg aus diesem Dilemma bietet die zu (6.32) gehörige *Rückwärtsrekursion*

$$y_{k-1} = -\frac{1}{b}\left(a y_k + y_{k+1}\right)\,, \quad k = n, \ldots, 0\,, \tag{7.62}$$

mit gegebenen Anfangswerten y_n und y_{n+1}. Man beachte, daß sich $b \neq 0$ aus $\lambda_1 \neq 0$ ergibt.[8] Die Nullstellen des zur Rückwärtsrekursion (7.62) gehörigen charakteristischen Polynoms

$$\mu^2 + \frac{a}{b}\mu + \frac{1}{b} = 0$$

[8] Selbstverständlich ließe sich (7.62) durch die Indextransformation $k' = n - k$ auch in der üblichen Form (6.32) schreiben.

sind gerade

$$\mu_1 = \frac{1}{\lambda_1}\,, \qquad \mu_2 = \frac{1}{\lambda_2}\,,$$

und aus (7.60) folgt

$$|\mu_1| > |\mu_2|\,.$$

Im Vergleich mit der Vorwärtsiteration dreht sich also das Wachstum der zu λ_1 bzw. λ_2 gehörenden Beiträge zur Lösung gerade um. Bei der Rückwärtsrekursion ist der zu $\mu_1 = \lambda_1^{-1}$ gehörige Lösungsanteil dominant! Wären x_n und x_{n+1} bekannt, so könnte man daher alle anderen Folgenglieder x_k, $k < n$, mit dem rekursiven Algorithmus 7.32 in stabiler Weise berechnen. Leider ist das gerade nicht der Fall.

Trotzdem können wir die stabile Rückwärtsrekursion (7.62) verwenden, um für jedes feste k_0 zumindest *Näherungen* $x_k^{(n)}$, $k = 1,\ldots,k_0$, zu konstruieren, welche mit wachsendem n die exakten Werte x_k immer genauer approximieren[9]. Dazu wählen wir die Anfangswerte

$$y_n = 1\,, \qquad y_{n+1} = 0$$

und berechnen aus der Rückwärtsrekursion (7.62) die zugehörige Lösung

$$y_k = \alpha\mu_1^{n-k+1} + \beta\mu_2^{n-k+1} = \alpha\lambda_1^{-(n-k+1)} + \beta\lambda_2^{-(n-k+1)}\,, \qquad k = n-1,\ldots,0\,. \tag{7.63}$$

Das ist nach Satz 7.34 mit Algorithmus 7.32 *in stabiler Weise* möglich. Wir brauchen dazu weder λ_1, λ_2 noch α, β explizit zu kennen. Für die gewählten Anfangswerte $y_n = 1$ und $y_{n+1} = 0$ folgt aus der Lösungsdarstellung in Lemma 6.23 unmittelbar $\alpha \neq 0$. Zusammen mit $|\mu_1| > |\mu_2|$ ergibt sich daraus $y_0 \neq 0$ für genügend großes n. Wir setzen

$$x_k^{(n)} = x_0\frac{y_k}{y_0}\,, \qquad k = 1,\ldots,k_0\,.$$

Die Rechtfertigung dafür liefert der folgende Satz.

Satz 7.35. *Unter der Voraussetzung $x_k \neq 0$, $k = 1,\ldots,k_0$, gilt*

$$\lim_{n\to\infty}\frac{|x_k - x_k^{(n)}|}{|x_k|} = 0\,, \qquad k = 1,\ldots,k_0\,.$$

Beweis. Unter Verwendung von (7.61) erhalten wir

$$x_k - x_k^{(n)} = x_k - x_0\frac{y_k}{y_0} = x_k\left(1 - \lambda_1^{-k}\frac{y_k}{y_0}\right)\,.$$

Einsetzen von (7.63) ergibt wegen $\alpha \neq 0$

$$\lambda_1^{-k}\frac{y_k}{y_0} = \lambda_1^{-k}\,\frac{\alpha\lambda_1^{-(n-k+1)} + \beta\lambda_2^{-(n-k+1)}}{\alpha\lambda_1^{-(n+1)} + \beta\lambda_2^{-(n+1)}} = \frac{1 + \frac{\beta}{\alpha}\left(\frac{\lambda_1}{\lambda_2}\right)^{n-k+1}}{1 + \frac{\beta}{\alpha}\left(\frac{\lambda_1}{\lambda_2}\right)^{n+1}} \to 1\,, \quad \text{für} \quad n\to\infty$$

und damit die Behauptung. □

Für genügend große n wird also x_k durch $x_k^{(n)}$ beliebig genau approximiert. Andererseits bedeutet ein großes n aber auch einen großen Rechenaufwand. Die Frage nach einem „genügend großen, aber nicht zu großen“ Genauigkeitsparameter n, führt auf die Frage nach möglichst genauen Schätzungen des Approximationsfehlers $|x_k - x_k^{(n)}|/|x_k|$ in Abhängigkeit von n. Wir wollen auf dieses Thema hier nicht eingehen und überlassen einfach dem Anwender die Wahl.

Der folgende Algorithmus berechnet eine Näherung des Folgenglieds x_k einer rezessiven Lösung der Drei-Term-Rekursion (6.32).

Algorithmus 7.36 (Miller, 1952).

[9] Wir wissen bereits, daß numerische Verfahren im allgemeinen nur (rundungs-) fehlerbehaftete Lösungen liefern. Daher ist es durchaus kein Nachteil, wenn ein Verfahren nicht die exakte Lösung, sondern beliebig genaue Approximationen liefert.

```
function f = MILLER(x0,K,n)

  y(n) = 1;  y(n+1) = 0;

  for k = n:-1:0
    y(k-1) = -(a*y(k)+y(k+1))/b;
  end

  f = x0*y(K)/y(0);

return
```

Wir wollen Algorithmus 7.36 ausprobieren und betrachten dazu das Beispiel

$$x_{k+1} + 2x_k - x_{k-1} = 0\,, \qquad x_{-1} = 1\,, \quad x_0 = \lambda_1\,, \quad \lambda_1 = \sqrt{2} - 1\,,$$

an dem der rekursive Algorithmus 7.32 im vorigen Abschnitt so kläglich gescheitert ist. Zum Vergleich verwenden wir jeweils die geschlossene Lösungsdarstellung $x_k = \lambda_1^k$. Bei Wahl von $n = 510$ liefert Algorithmus 7.36 die folgenden Ergebnisse.

k_0	10	50	100	500
rel. Fehler	$0.2 \cdot 10^{-16}$	$0.1 \cdot 10^{-15}$	$0.2 \cdot 10^{-15}$	$0.3 \cdot 10^{-10}$

Tabelle 7.2 Approximation rezessiver Lösungen mit dem Miller-Algorithmus

Für $k_0 = 10,\ 50,\ 100$ erhalten wir also bis auf Maschinengenauigkeit das richtige Ergebnis. Für $k_0 = 500$ macht sich die Wahl von $n = 510$ durch nachlassende Genauigkeit bemerkbar. Für größere k_0 hat man auch n weiter zu vergrößern. Dabei stößt man auf eine neue Schwierigkeit: Bei Wahl von $n > 805$ liegt der zugehörige Wert von $y(0)$ außerhalb des darstellbaren Bereichs von Gleitkommazahlen. Jede Implementierung von Algorithmus 7.36, die nicht nur Testzwecken dienen, sondern anderweitige Verwendung finden soll, muß dem Rechnung tragen!

Die Berechnung rezessiver Lösungen von Dreiterm-Rekursionen mittels Rückwärtsrekursion im Sinne des Miller-Algorithmus ist stabil.

Im Gegensatz zu der Lösungsdarstellung aus Lemma 6.23 lässt sich der Algorithmus von Miller auf Drei-Term-Rekursionen mit variablen Koeffizienten verallgemeinern, wird dann allerdings etwas aufwendiger. Für Einzelheiten verweisen wir auf Kapitel 6 im Lehrbuch von Deuflhard und Hohmann [10].

Nun wissen wir aber aus unseren Überlegungen zur Koniditon der Dreiterm-Rekursion, dass sie gerade im Falle rezessiver Lösungen ($\beta = 0$ oder $x_0 = \lambda_1 x_{-1}$) schlecht konitioniert ist, siehe Abschätzung 6.37. Unser Ergebnis bedeutet also, dass wir ein Beispiel für die folgende Aussage gefunden haben:

Man kann auch für schlecht konditionierte Probleme stabile Algorithmen finden!

Diese Einsicht ist in vielen Bereichen der angewandten Mathematik zentral. Zum Beispiel werden unter dem Schlagwort „Inverse Probleme“ eine Vielzahl von z.B. in der Medizin, Geophysik oder Bildbearbeitung wichtige Probleme behandelt, deren gemeinsames Merkmal jeweils ist, dass sie schlecht konditioniert und/oder schlecht gestellt sind und die Konstruktion stabiler Algorithmen ein bedeutsames Forschungsthema darstellt.

7.9 Aufgaben

Aufgabe 7.1 *Wir betrachten die durch $f(x) = x^2 \sin(1/x)$ für $x \neq 0$ und $f(0) = 0$ gegebene Funktion $f : \mathbb{R} \to \mathbb{R}$. a) Zeigen Sie, daß f differenzierbar, aber die Ableitung f' an der Stelle $x = 0$ nicht stetig ist. b) Berechnen Sie die relative Kondition $\kappa_{\text{rel}}(f,h)$ der Auswertung des Differenzenquotienten*

$$d(h) = \frac{f(x+h) - f(x)}{h}$$

und untersuchen Sie deren Verhalten für $h \to 0$.
c) Werten Sie $d(h)$ für $h_0 = 10^{-3}$ und $\tilde{h}_0 = h_0 + 10^{-6}$ aus, berechenen Sie den relativen Fehler und kommentieren Sie die Ergebnisse.

Aufgabe 7.2 *Wiederholen Sie die rekursive Abschätzung der Stabilität der Dreiterm-Rekursion aus Abschnitt 7.8 für den Fall $\alpha = 0$ und $q > 0$. Zeigen Sie, dass man eine exponentiell wachsende obere Schranke für den Rundungsfehler findet. Kann der Rundungsfehler im ungünstigsten Fall tatsächlich so stark anwachsen? Zeigen Sie, dass sich für $\lambda_1, \lambda_2 > 0$ in jedem Schritt der Rekursion eine milde Auslöschung ergibt.*

Aufgabe 7.3 *Berechnen sie die relative Kondition der Berechnung der Varianz. Überlegen Sie dazu zuerst, was die Eingabewerte sein sollen und ob der Mittelwert dazu gehört. Dann schätzen Sie die Kondition für die zwei Berechnungsformeln V_1 und V_2 ab und erklären Sie eventuelle Diskrepanzen.*

Aufgabe 7.4 *Konstruieren Sie einen Auswertungsbaum zur Berechnung der in Abschnitt 7.5 diskutieren Varianz V_2 mittel hierarchischer Summation und schätzen Sie das zugehörige Rundungsfehlerverhalten ab. Vergleichen Sie mit dem Ergebnis aus Satz 7.25.*

Aufgabe 7.5 *In dieser Aufgabe betrachten wir für $0 < q < 1$ die geometrische Reihe*

$$S_\infty := \lim_{n\to\infty} S_n = \sum_{k=0}^{\infty} q^k = \frac{1}{1-q}$$

mit den Partialsummen

$$S_n = \sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q}\,.$$

1. *Geben Sie eine Abschätzung für $|S_\infty - S_n|$ in Abhängigkeit von q und n an.*
2. *Es sei $q = 1/2$. Bestimmen Sie unter Verwendung der obigen Abschätzung eine Zahl N so, dass aus $n \geq N$ die Abschätzung*
$$|S_\infty - S_n| \leq 10^{-8} \tag{7.64}$$
folgt.
3. *Überprüfen Sie Ihr theoretisch erzieltes Resultat am Rechner. Verwenden Sie dabei zur Aufsummierung der Reihenglieder eine `while`-Schleife und wählen Sie (7.64) als Abbruchkriterium. Stimmt das theoretisch gewonnene N mit dem numerisch ermittelten überein? Wie erklären Sie sich etwaige Differenzen?*

Aufgabe 7.6 *Man betrachte die homogene Drei-Term-Rekursion*

$$x_{k+1} - 2{,}1x_k + 0{,}2x_{k-1} = 0, \qquad k = 1,2,3,\ldots$$

mit dem Startwert $x_0 = 1$.

1. *Geben Sie unter Verwendung von Lemma 6.32 eine geschlossene Darstellung für x_k als Funktion von x_1 an.*
2. *Implementieren Sie die Drei-Term-Rekursion in `MATLAB` zur rekursiven Berechnung der Folge x_k, $k = 1,\ldots,$ in Abhängigkeit von dem Startwert x_1.*
3. *Benutzen Sie dieses Programm, um für die Startwerte*

$$x_1 = 0{,}1 \qquad \textit{und} \qquad x_1 = 0{,}01$$

jeweils die Folgenglieder x_k, $k = 1,\ldots,100$, auszurechnen. Berechnen Sie außerdem die Folgenglieder $\tilde{x}_k$, $k = 1,\ldots,100$, für die gestörten Startwerte

$$\tilde{x}_1 = 0{,}10001 \qquad \textit{bzw.} \qquad \tilde{x}_1 = 0{,}010001\,.$$

Plotten Sie den relativen Fehler $|x_k - \tilde{x}_k|/|x_k|$ für $0 \le k \le 100$. Kommentieren Sie die Ergebnisse mit Bezug auf die relative Kondition $\kappa^k_{rel,1}$ aus Formel (6.41).

4. *Implementieren Sie die geschlossene Darstellung aus a) und stellen Sie die resultierenden Folgenglieder x_k, $0 \le k \le 100$, zu den Startwerten $x_1 = 0{,}1$ und $x_1 = 0{,}01$ jeeils graphisch dar. Vergleichen Sie das Ergebnis mit den Resultaten aus b)*

Aufgabe 7.7 *Wir betrachten die durch $f(x) = x^2 \sin(1/x)$ für $x \neq 0$ und $f(0) = 0$ gegebene Funktion $f : \mathbb{R} \to \mathbb{R}$.*
a) Zeigen Sie, daß f differenzierbar, aber die Ableitung f' an der Stelle $x = 0$ nicht stetig ist.
b) Berechnen Sie die relative Kondition $\kappa_{\mathrm{rel}}(f,h)$ der Auswertung des Differenzenquotienten

$$d(h) = \frac{f(x+h) - f(x)}{h}$$

und untersuchen Sie deren Verhalten für $h \to 0$.
c) Werten Sie $d(h)$ für $h_0 = 10^{-3}$ und $\tilde{h}_0 = h_0 + 10^{-6}$ aus, berechenen Sie den relativen Fehler und kommentieren Sie die Ergebnisse.