1 Fehleranalyse

1.1 Zahldarstellung und Rundungsfehler

Bei der Verarbeitung numerischer Algorithmen auf dem „Computer“ treten zwangsläufig Fehler auf, die durch die Endlichkeit des Bereiches der darstellbaren Zahlen bedingt sind. Zur Approximation von reellen (und auch komplexen) Zahlen und der elementaren arithmetischen Operationen zwischen ihnen werden sog. „Maschinenzahlen“ und „Maschinenoperationen“ verwendet, welche auf dem Computer realisierbar sind.

Eine „(normalisierte) Gleitkommazahl“ zur Basis $b \in \mathbb{N}$, $b \geq 2$, ist eine Zahl $x \in \mathbb{R}$ dargestellt in der Form

$$x = \pm\, m \cdot b^{\pm e} \tag{1.1.1}$$

mit der „Mantisse“ $m = m_1 b^{-1} + \ldots + m_r b^{-r} + \ldots \in \mathbb{R}$, und dem „Exponenten“ $e = e_{s-1} b^{s-1} + \ldots + e_0 b^0 \in \mathbb{N} \cup \{0\}$, wobei $m_i, e_i \in \{0, \ldots, b-1\}$. Für $x \neq 0$ ist diese Darstellung durch die Normierungsvorschrift $m_1 \neq 0$ eindeutig bestimmt. Für $x = 0$ setzt man $m = 0$.

Bemerkung 1.1: Die Verwendung der Gleitkommadarstellung im numerischen Rechnen ist wesentlich, um Zahlen sehr unterschiedlicher Größe verarbeiten zu können; z. B. Ruhemasse Elektron $M_0 = 9.11 \cdot 10^{-28}\,\mathrm{g}$, Lichtgeschwindigkeit $c = 2.998 \cdot 10^{10}\,\mathrm{cm/sec}$.

Auf dem Rechner stehen für die Darstellung von reellen Zahlen nur endlich viele Stellen zur Verfügung:

r Ziffern + 1 Vorzeichen für die Mantisse

s Ziffern + 1 Vorzeichen für den Exponenten.

Die Speicherung einer solchen Zahl

$$x = \pm\,[m_1 b^{-1} + \ldots + m_r b^{-r}] \cdot b^{\pm[e_{s-1} b^{s-1} + \ldots + e_0 b^0]}$$

erfolgt dann in der Form $x : (\pm)\,[m_1 \ldots m_r](\pm)[e_{s-1} \ldots e_0]$. Aus technischen Gründen verwenden moderne Rechner eine Zahldarstellung mit den Basen $b = 2$ (Dualsystem) oder $b = 16$ (Sedezimalsystem) oder Mischungen davon. Die in der obigen Form auf einem Rechner dargestellten (rationalen) Zahlen werden „Maschinenzahlen“ genannt; sie bilden das sog. „numerische Gleitkommagitter“ $A = A(b, r, s)$. Da A endlich ist, gibt es eine größte/kleinste darstellbare Zahl:

$$x_{\max\,/\,\min} = \pm\,(b-1)\{b^{-1} + \ldots + b^{-r}\} \cdot b^{(b-1)\{b^{s-1} + \ldots + b^0\}} = \pm\,(1 - b^{-r}) \cdot b^{b^s - 1}$$

sowie eine kleinste positive/größte negative darstellbare Zahl:

$$x_{\mathrm{posmin/negmax}} = \pm\, b^{-1} \cdot b^{-(b-1)\{b^{s-1} + \ldots + b^0\}} = \pm\, b^{-b^s}\,.$$

Beispiel 1.1: Beim sog. „IEEE-Format“ (üblich auf UNIX-Workstations) werden zur Darstellung von doppelt genauen Zahlen (REAL*8 in FORTRAN) 64 Bits (= 8 Bytes) verwendet:

$$x = \pm m \cdot 2^{c-1022} .$$

Dabei stehen 1 Bit für das Vorzeichen, 52 Bits für die Mantisse $m = 2^{-1} + m_2 2^{-2} + \ldots + m_{53} 2^{-53}$ (die erste Mantissenstelle ist aus Normierungsgründen stets 1) und 11 Bits für die sog. „Charakteristik“ $c = c_0 2^0 + \ldots + c_{10} 2^{10} \in [1, 2046]$ zur Verfügung, wobei $m_i\,,\, c_i \in \{0, 1\}$ Dualzahlen sind. Durch die vorzeichenfreie Darstellung des Exponenten in der Form $e = c - 1022$ wird der Zahlbereich um eine Zweierpotenz erweitert. Für REAL*8-Zahlen gilt somit:

$$\begin{aligned} x_{\max} &\sim 2^{1024} \sim 1.8 \cdot 10^{308} , & x_{\min} &\sim -2^{1024} \sim -1.8 \cdot 10^{308} , \\ x_{\text{posmin}} &= 2^{-1022} \sim 2.2 \cdot 10^{-308} , & x_{\text{negmax}} &= -2^{-1022} \sim -2.2 \cdot 10^{-308} . \end{aligned}$$

Die ausgenommenen Werte $c = 0$ und $c = 2047$ der Charakteristik werden zur Darstellung der Null ($m_2 = \ldots = m_{53} = 0$, $c_0 = \ldots = c_{10} = 0$) sowie einer Sondergröße „nan“ (not a number) verwendet.

Die Ausgangsdaten $x \in \mathbb{R}$ einer numerischen Aufgabe und die Zwischenergebnisse einer Rechnung müssen durch Maschinenzahlen dargestellt werden. Für Zahlen innerhalb des „zulässigen“ Bereiches

$$D := [x_{\min}, x_{\text{negmax}}] \cup \{0\} \cup [x_{\text{posmin}}, x_{\max}]$$

wird eine „Rundungsoperation“ $\mathrm{rd} : \mathrm{D} \to \mathrm{A}$ verwendet, an die man die natürliche Forderung stellt

$$|x - \mathrm{rd(x)}| = \min_{\mathrm{y} \in \mathrm{A}} |\mathrm{x} - \mathrm{y}| \qquad \forall \mathrm{x} \in \mathrm{D} . \tag{1.1.2}$$

Dies ist beim IEEE-Format z. B. realisiert durch „natürliche“ Rundung:

$$\mathrm{rd(x)} = \mathrm{sgn(x)} \cdot \begin{cases} 0.m_1 \cdots m_{53} \cdot 2^e , & \text{für } m_{54} = 0 \\ (0.m_1 \cdots m_{53} + 2^{-53}) \cdot 2^e , & \text{für } m_{54} = 1 . \end{cases}$$

Andere manchmal vorkommende Rundungsarten, welche (1.1.2) nicht erfüllen, werden im Folgenden nicht betrachtet. Für Zahlen außerhalb des zulässigen Bereiches D (z. B. als Resultat einer Division durch Null) wird von einigen Maschinen Exponentenüberlauf („overflow“ oder „underflow“) registriert und die Verarbeitung abgebrochen, während im IEEE-Format in diesem Fall mit der unbestimmten Variable „nan“ weitergearbeitet wird. Der mit der Rundung verbundene sog. „absolute Rundungsfehler“

$$|x - \mathrm{rd(x)}| \leq \tfrac{1}{2}\, \mathrm{b}^{-\mathrm{r}} \mathrm{b}^{\mathrm{e}} \tag{1.1.3}$$

hängt jeweils noch vom Exponenten e von x ab. Dagegen ist der sog. „relative Rundungsfehler“

$$\left|\frac{x - \mathrm{rd(x)}}{x}\right| \leq \tfrac{1}{2}\,\frac{b^{-r}b^{e}}{|m|b^{e}} \leq \tfrac{1}{2}\,b^{-r+1} \tag{1.1.4}$$

für $x \in D$, $x \neq 0$, beschränkt durch die sog. „Maschinengenauigkeit“ $\mathrm{eps} := \frac{1}{2}\,b^{-r+1}$. Für $x \in D$ ist dann offenbar

$$\mathrm{rd(x)} = \mathrm{x}(1+\varepsilon) \quad \text{mit} \quad |\varepsilon| \leq \mathrm{eps}\,. \tag{1.1.5}$$

Bei Anwendung des IEEE-Formats ist der maximale relative Rundungsfehler

$$\mathrm{eps}_{\mathrm{REAL*8}} \leq \tfrac{1}{2}2^{-52} \sim 10^{-16}\,.$$

Die arithmetischen Grundoperationen $* \in \{+, -, \cdot, /\}$ werden auf der Rechenanlage durch entsprechende „Maschinenoperationen“ $\circledast \in \{\oplus, \ominus, \odot, \oslash\}$ ersetzt, welche Maschinenzahlen wieder in Maschinenzahlen überführen. Dies ist meist für $x, y \in A$ im Falle $x * y \in D$ gemäß

$$x \circledast y = \mathrm{rd(x * y)} = (\mathrm{x * y})(1+\varepsilon)\,, \quad |\varepsilon| \leq \mathrm{eps}\,, \tag{1.1.6}$$

realisiert. Dazu werden die Operationen maschinenintern (meist unter Verwendung einer erhöhten Stellenzahl für die Mantisse) ausgeführt, in normalisierte Form gebracht und dann gerundet. Im Fall $x * y \notin D$ erscheint meist eine Fehlermeldung. Bei dem Gebrauch von „IF-Abfragen“ in Programmen ist zu berücksichtigen, dass die Maschinenoperationen $\oplus$ und $\odot$ dem Assoziativgesetz und dem Distributivgesetz nur näherungsweise genügen; i. Allg. ist für $x, y, z \in A$:

$$(x \oplus y) \oplus z \;\neq\; x \oplus (y \oplus z)\,, \quad (x \oplus y) \odot z \;\neq\; (x \odot z) \oplus (y \odot z)\,.$$

Insbesondere gilt i. Allg. für Zahlen $x, y \in A$:

$$x \oplus y = x, \quad \text{für} \quad |y| \leq \frac{|x|}{b}\mathrm{eps}. \tag{1.1.7}$$

Hieraus lässt sich für einen konkreten Rechner die Größe der Maschinengenauigkeit eps experimentell ermitteln.

1.2 Konditionierung numerischer Aufgaben

Eine numerische Aufgabe (z. B. Bestimmung einer Nullstelle, Lösung eines linearen Gleichungssystems u.s.w.) wird als „gut konditioniert“ bezeichnet, wenn eine kleine Störung der Eingangsdaten auch nur eine kleine Änderung der Ergebnisse zur Folge hat.

Beispiel 1.2: Als Beispiel betrachten wir das lineare Gleichungssystem

$$\begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.8642 \\ 0.1440 \end{bmatrix}$$

mit der (eindeutig bestimmten) Lösung $(x, y)^T = (2, -2)^T$. Störung der rechten Seite zu $(0.86419999, 0.14400001)^T$ erzeugt die „Näherungslösung“ $(\tilde{x}, \tilde{y})^T = (0.9911, -0.4870)^T$ Diese numerische Aufgabe ist offenbar sehr schlecht konditioniert.

Zur Präzisierung des Begriffes „Konditionierung“ müssen wir zunächst den der „numerischen Aufgabe“ definieren. Wir wollen hier unter einer „numerischen Aufgabe“ die Berechnung endlich vieler Größen $y_i\,(i = 1, \ldots, n)$ aus gewissen Größen $x_j\,(j = 1, \ldots, m)$ mittels einer funktionalen Vorschrift $y_i = f_i(x_1, \ldots, x_m)$ verstehen. Der Einfachheit halber betrachten wir hier nur den Fall, dass die y_i, x_j reelle (oder komplexe) Zahlen sind, und verwenden zur Abkürzung die vektorielle Schreibweise $y = f(x)$ mit

$$x = (x_1, \ldots, x_m)^T\,, \quad y = (y_1, \ldots, y_n)^T\,, \quad f = (f_1, \ldots, f_n)^T\,.$$

Als Beispiel kann die Berechnung eines Vektors $x \in \mathbb{R}^n$ als Lösung eines linearen Gleichungssystems $Ax = b$ dienen, wobei $x = f(A, b) := A^{-1}b$.

Definition 1.1: *Bei Verwendung fehlerhafter Eingangsdaten $x_j + \Delta x_j$ (z. B. aufgrund des Rundungsfehlers) ergeben sich fehlerhafte Resultate $y_i + \Delta y_i$. Wir bezeichnen $|\Delta y_i|$ als den „absoluten“ Fehler und $|\Delta y_i / y_i|$ (für $y_i \neq 0$) als den „relativen“ Fehler.*

Große absolute Fehler können offenbar, „relativ“ gesehen, klein sein und umgekehrt; z. B. mag ein Fehler von $\pm\,100$ km beim Messen der Entfernung Erde-Mond als „klein“ angesehen werden, während derselbe Fehler bezogen auf die Entfernung Heidelberg-Paris sicherlich als „groß“ anzusehen ist.

Wir haben gesehen, dass der relative Rundungsfehler durch die Maschinengenauigkeit eps beschränkt ist. Hier wird uns auch hauptsächlich nur dieser interessieren. Im Folgenden betreiben wir eine sog. „differentielle“ Fehleranalyse, die sich auf die Betrachtung des Einflusses relativ kleiner Datenfehler $|\Delta x_j| \ll |x_j|$ beschränkt. Sind die Funktionen $f_i = f_i(x_1, \ldots, x_m)$ stetig partiell differenzierbar nach den Argumenten x_j, so gilt nach dem Taylorschen Satz

$$\Delta y_i \;=\; f_i(x + \Delta x) - f_i(x) \;=\; \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}\,(x)\,\Delta x_j + R_i^f(x; \Delta x), \quad i = 1, \ldots, m, \qquad (1.2.8)$$

mit einem Restglied $R_i^f(x; \Delta x)$, welches schneller als $|\Delta x| = \max_{j=1,\ldots,m} |\Delta x_j|$ gegen Null geht; wir schreiben dies abgekürzt als $R_i^f(x; \Delta x) = o(|\Delta x|)$. Der Einfachheit halber nehmen wir an, dass sogar $R_i^f(x; \Delta x) = O(|\Delta x|^2)$ gilt, was im Falle der zweimaligen Differenzierbarkeit der Funktion f gesichert ist.

Definition 1.2: *Wir verwenden hier und im Folgenden die sog. „Landauschen*[1]*Symbole“ $O(\cdot)$ und $o(\cdot)$ zur quantitativen Beschreibung von Grenzprozessen. Für Funktionen $g(t)$ und $h(t)$ der Variablen $t \in \mathbb{R}_+$ bedeutet die Schreibweise*

$$g(t) = O(h(t)) \quad (t \to 0)\,,$$

dass für kleine $t \in (0, t_0]$ mit einer Konstanten $c \geq 0$ gilt

$$|\, g(t)\,| \leq c\,|\, h(t)\,|\,.$$

Entsprechend bedeutet $g(t) = o(h(t))$ für $t \to 0$, dass für kleine $t \in (0, t_0]$ mit einer Funktion $c\,(t) \to 0\,(t \to 0)$ gilt

$$|\, g(t)\,| \leq c\,(t)\,|\, h(t)\,|\,.$$

Analoge Schreibweisen verwendet man für Grenzübergänge $t \to \infty$.

Beispiel 1.3: Für eine zweimal stetige differenzierbare Funktion $g\,(t)$ folgt aus

$$g\,(t + \Delta t) = g\,(t) + \Delta t\, g'(t) + \tfrac{1}{2}\Delta t^2\, g''(\tau)\,, \quad \tau \in (t, t + \Delta t)\,,$$

für den sog. „vorwärts genommenen Differenzenquotienten“ die Beziehung

$$\Delta t^{-1}\left\{g(t + \Delta t) - g(t)\right\} = g'(t) + O(\Delta t)\,.$$

Die obige Formel (1.2.8) besagt, dass der Fehler Δy_i „in erster Näherung“, d. h. bis auf eine Größe der Ordnung $O(|\,\Delta x|^2)$, gleich dem ersten Summanden auf der rechten Seite ist; in Symbolen:

$$\Delta y_i \doteq \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}\,(x)\,\Delta x_j\,. \tag{1.2.9}$$

Für den komponentenweisen relativen Fehler gilt dann

$$\frac{\Delta y_i}{y_i} \doteq \sum_{j=1}^{m} \frac{\partial f_i}{\partial x_j}\,(x)\,\frac{\Delta x_j}{y_i} = \sum_{j=1}^{m} \underbrace{\frac{\partial f_i}{\partial x_j}\,(x)\,\frac{x_j}{f_i(x)}}_{=:\,k_{ij}(x)}\,\frac{\Delta x_j}{x_j}\,. \tag{1.2.10}$$

[1]Edmund Georg Hermann Landau (1877–1938): Deutscher Mathematiker; seit 1909 Professor in Göttingen (Nachfolger von Minkowski); 1934 wegen seiner jüdischen Abstammung zwangsweise pensioniert; fundamentale Beiträge zur analytischen Zahlentheorie, insbesondere zur Primzahlverteilung, und zur (komplexen) Funktionentheorie.

Dabei verhält sich der vernachlässigte Term wie

$$\Big|\frac{R_i^f(x;\Delta x)}{y_i}\Big| = O\Big(\frac{|\Delta x|^2}{|y_i|}\Big).$$

Unter der Voraussetzung, dass $|\Delta x| = o(|y_i|)$ kann er gegen den führenden $O(|\Delta x|)$-Term vernachlässigt werden. Wir werden im Folgenden stets annehmen, dass sich die auszuwertende Größe und die betrachteten Datenstörungen verhalten wie

$$|\Delta x| = o(|y_i|), \quad i = 1, \ldots, n.$$

Andernfalls darf das Restglied nicht vernachlässigt werden.

Definition 1.3: *Die Größen $k_{ij}(x)$ heißen „(relative) Konditionszahlen" der Funktion f im Punkt x. Sie sind ein Maß dafür, wie sich kleine relative Fehler in den Ausgangsdaten im Ergebnis auswirken. Man nennt die Aufgabe, $y = f(x)$ aus x zu berechnen, „schlecht konditioniert", wenn ein $|k_{ij}(x)| \gg 1$ ist; andernfalls „gut konditioniert" oder auch „gutartig". Im Fall $|k_{ij}(x)| < 1$ spricht man von „Fehlerdämpfung" und im Fall $|k_{ij}(x)| > 1$ von „Fehlerverstärkung".*

Die Konditionierung einer numerischen Aufgabe ist eng verknüpft mit der Frage nach berechenbaren Fehlerschranken für zugehörige Näherungslösungen. Beim Problem der direkten Funktionsauswertung $y = f(x)$ ist die relative Fehlerempfindlichkeit beschrieben durch die Relation (1.2.10). Interessanter ist das umgekehrte Problem der Gleichungslösung $x = f^{-1}(y)$, bei dem zu gegebenem y die Lösung der Gleichung $f(x) = y$ gesucht ist. Hierbei wird o.B.d.A. Gleichheit der Dimensionen $n = m$ angenommen. Als Beispiele können wieder die Lösung eines linearen Gleichungssystems oder die Bestimmung der Wurzeln einer quadratischen Gleichung dienen. In diesem Fall liegt es nahe, eine Fehlerabschätzung für irgendeine Näherungslösung $\tilde{x}$ auf dem Weg der „Probe" zu erzielen. Zu diesem Zweck bildet man den sog. „Defekt" $d(\tilde{x}) = f(\tilde{x}) - y$ (Die Größe $r(\tilde{x}) = y - f(\tilde{x}) = -d(\tilde{x})$ wird das „Residuum" von $\tilde{x}$ genannt.). Die Frage ist nun, ob für einen kleinen Defekt auch der tatsächliche Fehler $\Delta x = \tilde{x} - x$ klein ist. Die differentielle Fehleranalyse liefert hierzu

$$\frac{\Delta x_i}{x_i} \doteq \sum_{j=1}^{n} k_{ij}^{(-1)} \frac{\Delta y_j}{y_j}, \quad k_{ij}^{(-1)} := \frac{\partial f_i^{-1}}{\partial y_j}(y)\frac{y_j}{x_i},$$

mit den Konditionszahlen $k_{ij}^{(-1)}$ der inversen Abbildung $f^{-1}(\cdot)$. Wir fassen die Konditionszahlen zu Matrizen $K = (k_{ij})_{i,j=1}^n$ und $K^{(-1)} = (k_{ij}^{(-1)})_{i,j=1}^n$ zusammen. Für deren Produkt gilt dann

$$(K^{(-1)}K)_{ij} = \sum_{k=1}^{n} k_{ik}^{(-1)} k_{kj} = \sum_{k=1}^{n} \frac{\partial f_i^{-1}}{\partial x_k}\frac{x_k}{y_i}\frac{\partial f_k}{\partial x_j}\frac{y_j}{x_k} = \frac{y_j}{y_i}\sum_{k=1}^{n}\frac{\partial f_i^{-1}}{\partial x_k}\frac{\partial f_k}{\partial x_j} = \delta_{ij},$$

wobei das sog. „Kronecker[2]-Symbol“ δ_{ij} für die Alternative $\delta_{ij} = 1$, für $i = j$, bzw. $\delta_{ij} = 0$, für $i \neq j$, steht. Die Matrix $K^{(-1)}$ ist also gerade die Inverse von K.

1.2.1 Arithmetische Grundoperationen

Im Folgenden diskutieren wir die Konditionierung, d. h. die Anfälligkeit gegenüber kleiner Störungen der Eingangsdaten, einiger einfacher Grundaufgaben.

1) Die Addition $y = f(x_1, x_2) = x_1 + x_2$ zweier Zahlen $x_1, x_2 \in \mathbb{R}$, $x_1, x_2 \neq 0$, mit

$$k_1 = \frac{\partial f}{\partial x_1}\frac{x_1}{f} = 1 \cdot \frac{x_1}{x_1 + x_2} = \frac{1}{1 + x_2/x_1}$$
$$k_2 = \frac{\partial f}{\partial x_2}\frac{x_2}{f} = 1 \cdot \frac{x_2}{x_1 + x_2} = \frac{1}{1 + x_1/x_2}$$

ist „schlecht“ konditioniert für $x_1/x_2 \sim -1$. Bei der Addition ähnlich großer Zahlen mit unterschiedlichem Vorzeichen kann bei der Fortpflanzung von kleinen Störungen in den Eingangsgrößen sog. „Auslöschung“ wesentlicher (dezimaler) Stellen auftreten.

Definition 1.4 (Auslöschung): *Unter „Auslöschung“ versteht man den Verlust an wesentlichen Dezimalstellen bei der Subtraktion von Zahlen gleichen Vorzeichens. Dies ist gefährlich im Fall, dass eine oder beide der Zahlen keine Maschinenzahlen sind und vor Ausführung der Operation gerundet werden. Bei der Subtraktion von Maschinenzahlen ist Auslöschung natürlich unschädlich.*

Beispiel 1.4: Dezimale Gleitpunktrechnung mit $r = 4$ und $s = 1$

$$\begin{array}{rcl@{\quad}c@{\quad}rcl} x_1 & = & 0.11258762 \cdot 10^2 & \rightarrow & \mathrm{rd}(x_1) & = & 0.1126 \cdot 10^2 \\ x_2 & = & 0.11244891 \cdot 10^2 & \rightarrow & \mathrm{rd}(x_2) & = & 0.1124 \cdot 10^2 \\ x_1 + x_2 & = & 0.\underline{2250}3653 \cdot 10^2 & , & \mathrm{rd}(x_1) + \mathrm{rd}(x_2) & = & 0.\underline{2250} \cdot 10^2 \\ x_1 - x_2 & = & 0.13871 \cdot 10^{-1} & , & \mathrm{rd}(x_1) - \mathrm{rd}(x_2) & = & 0.2000 \cdot 10^{-1} \end{array}$$

Im zweiten Fall gilt $k_1 \sim k_2 \sim 810$, d. h. fast 1000-fache Fehlerverstärkung.

2) Die Multiplikation $y = f(x_1, x_2) = x_1 \cdot x_2$ mit

$$k_1 = \frac{\partial f}{\partial x_1}\frac{x_1}{f} = x_2 \frac{x_1}{x_1 \cdot x_2} = 1, \qquad k_2 = \ldots = 1,$$

ist generell „gut“ konditioniert. Dasselbe gilt auch für die Division (Übungsaufgabe).

[2]Leopold Kronecker (1823–1891): Deutscher Mathematiker; wirkte in Berlin als „Privatgelehrter“; betrieb die Arithmetisierung der Mathematik; wichtiger Vertreter des „Konstruktivismus“, welcher die generelle Verwendung des Widerspruchsbeweises und des „aktual Unendlichen“ in Form z. B. der allgemeinen reellen Zahlen ablehnt.

1.2.2 Lösung quadratischer Gleichungen

Für Zahlen $p, q \in \mathbb{R}$ wird die folgende quadratische Gleichung betrachtet:

$$y^2 - py + q = 0 \quad \text{(o.b.d.A. } q \neq 0\text{)}.$$

Für die Wurzeln y_1 und y_2 gilt $y_{1,2} = y_{1,2}(p,q) = p/2 \pm \sqrt{p^2/4 - q}$ sowie $p = y_1 + y_2\,, q = y_1 \cdot y_2$ (Vietascher[3] Wurzelsatz). Damit erhält man:

$$\left.\begin{aligned} \frac{\partial y_1}{\partial p} + \frac{\partial y_2}{\partial p} &= 1 \\ \frac{\partial y_1}{\partial p}\, y_2 + y_1 \frac{\partial y_2}{\partial p} &= 0 \end{aligned}\right\} \Rightarrow \qquad \frac{\partial y_2}{\partial p} = \frac{y_2}{y_2 - y_1}\,, \quad \frac{\partial y_1}{\partial p} = \frac{y_1}{y_2 - y_1}$$

$$\left.\begin{aligned} \frac{\partial y_1}{\partial q} + \frac{\partial y_2}{\partial q} &= 0 \\ \frac{\partial y_1}{\partial q}\, y_2 + y_1 \frac{\partial y_2}{\partial q} &= 1 \end{aligned}\right\} \Rightarrow \qquad \frac{\partial y_1}{\partial q} = \frac{1}{y_1 - y_2} = -\frac{\partial y_2}{\partial q}$$

$$k_{11} = \frac{\partial y_1}{\partial p}\frac{p}{y_1} = \frac{y_1}{y_2 - y_1}\frac{p}{y_1} = \frac{y_1 + y_2}{y_1 - y_2} = \frac{1 + y_2/y_1}{1 - y_2/y_1}$$
$$k_{12} = \frac{\partial y_1}{\partial q}\frac{q}{y_1} = \frac{1}{y_1 - y_2}\frac{q}{y_1} = \frac{y_2}{y_1 - y_2} = \frac{1}{1 - y_2/y_1}$$

Für k_{21} und k_{22} gilt Analoges. Die Berechnung von y_1, y_2 ist schlecht konditioniert für $y_1/y_2 \sim 1$, d. h. wenn die Wurzeln relativ dicht beeinander liegen.

Beispiel 1.5: $p = 4\,, \quad q = 3.999\,, \quad y_{1,2} = 2 \pm 0.01\,,$

$$k_{12} = \frac{1}{1 - y_1/y_2} = 99.5 \quad \Rightarrow \quad \text{fast 100-fache Fehlerverstärkung.}$$

1.3 Stabilität numerischer Algorithmen

Gegeben sei wieder eine numerische Aufgabe der Art $y = f(x)$ mit einer Abbildung $f : \mathbb{R}^m \to \mathbb{R}^n$. Unter einem „Verfahren“ (oder „Algorithmus“) zur gegebenenfalls näherungsweisen Berechnung von y aus x verstehen wir eine endliche (oder auch abzählbar unendliche) Folge von „elementaren“ Abbildungen $\varphi^{(k)}$, die durch sukzessive Anwendung einen Näherungwert $\tilde{y}$ zu y liefern:

$$x = x^{(0)} \to \varphi^{(1)}\,(x^{(0)}) = x^{(1)} \to \ldots \to \varphi^{(k+1)}\,(x^{(k)}) = x^{(k+1)} \to \ldots\, \tilde{y}\,.$$

[3] Francois Viete, lat. Franciscus Vieta (1540–1603): Französischer Mathematiker; Arbeiten über algebraische Gleichungen und sphärische Trigonometrie; gab trigonometrische Tafeln heraus und führte die systematische Buchstabenrechnung ein.

Im einfachsten Fall sind die $\varphi^{(k)}$ arithmetische Grundoperationen.

Definition 1.5: *Bei der Durchführung des Algorithmus auf einer Rechenanlage treten in jedem Schritt Fehler auf (z. B. Rundungsfehler, Auswertungsfehler von transzendenten Funktionen, u.s.w.), die sich bis zum Ende der Rechnung akkumulieren können. Der Algorithmus wird „stabil“ (oder auch „gutartig“) genannt, wenn die im Verlaufe der Ausführung akkumulierten Fehler den durch die Konditionierung der Aufgabe $y = f(x)$ bedingten unvermeidbaren Problemfehler nicht übersteigen.*

Eine der Hauptaufgaben der numerischen Mathematik ist es, für die in den Anwendungen auftretenden Aufgaben stabile Lösungsalgorithmen zu finden. Wir diskutieren im folgenden einige elementare Beispiele.

1.3.1 Lösung quadratischer Gleichungen

Wir betrachten die Auflösung einer quadratischen Gleichung der Form

$$y^2 - py + q = 0 \quad (0 \neq q < p^2/4)\,, \quad y_{1,2} = f(p,q) = p/2 \pm \sqrt{p^2/4 - q}\,.$$

Für $|y_1/y_2| \gg 1$, d. h. für $q \ll p^2/4$, ist die Aufgabe gut konditioniert. Der Algorithmus zur Berechnung der Wurzeln könnte wie folgt aussehen:

$$u = p^2/4\,, \quad v = u - q\,, \quad w = \sqrt{v} \quad (\geq 0)\,.$$

Im Fall $p < 0$ wird zur Vermeidung von Auslöschung zunächst $\tilde{y}_2 = p/2 - w$ berechnet mit der akzeptablen Fehlerfortpflanzung

$$\left|\frac{\Delta y_2}{y_2}\right| \dot{\leq} \underbrace{\left|\frac{1}{1 - 2w/p}\right|}_{\approx 1} \left|\frac{\Delta p}{p}\right| + \underbrace{\left|\frac{1}{1 - p/(2w)}\right|}_{\approx 1} \left|\frac{\Delta w}{w}\right|.$$

Die zweite Wurzel könnte dann auf folgenden Wegen bestimmt werden:

Variante A	Variante B
$\tilde{y}_1 = p/2 + w$	$\tilde{y}_1 = q/y_2 \quad (\text{wegen } q = y_1 y_2)$

Für $q \ll p^2/4$ ist $w \approx -p/2$, d. h. bei Variante A tritt zwangsläufig Auslöschung ein. Die Rundungsfehler in p und w übertragen sich auf y_1 wie

$$\left|\frac{\Delta y_1}{y_1}\right| \dot{\leq} \underbrace{\left|\frac{1}{1 + 2w/p}\right|}_{\gg 1} \underbrace{\left|\frac{\Delta p}{p}\right|}_{\leq \text{eps}} + \underbrace{\left|\frac{1}{1 + p/(2w)}\right|}_{\gg 1} \underbrace{\left|\frac{\Delta w}{w}\right|}_{\approx \text{eps}}.$$

Dieser Algorithmus ist offenbar im vorliegenden Fall $q \ll p^2/4$ sehr instabil. Bei Variante B gilt dagegen

$$\left|\frac{\Delta y_1}{y_1}\right| \dot{\leq} \underbrace{\left|\frac{\Delta q}{q}\right|}_{\leq \text{ eps}} + \underbrace{\left|\frac{\Delta y_2}{y_2}\right|}_{\approx \text{ eps}}.$$

d. h. dieser Algorithmus ist stabil.

Regel 1.1: *Bei der Lösung quadratischer Gleichungen sollten nicht beide Wurzeln aus der Lösungsformel berechnet werden.*

Beispiel 1.6: $p = -4\,, \quad q = 0.01$ (vierstellige Rechnung)

$$\left.\begin{array}{rcl} u & = & 4 \\ v & = & 3.99 \\ w & = & \underline{1.997}4984\ldots \\ \tilde{y}_2 & = & \underline{-3.997}4984\ldots \end{array}\right\} \quad \tilde{y}_1 = \left\{\begin{array}{rcl} \text{exakt} & : & -0.0025015\ldots \\ \text{A} & : & -0.0030 \quad \text{(rel. Fehler} \quad 0.2\text{)} \\ \text{B} & : & -0.0025 \end{array}\right.$$

1.3.2 Auswertung arithmetischer Ausdrücke

Im Folgenden bedienen wir uns einer sog. „Vorwärtsrundungsfehleranalyse“, bei welcher die Akkumulation des Rundungsfehlers ausgehend vom Startwert abgeschätzt wird. Wir beginnen mit der Auswertung eines einfachen arithmetischen Ausdrucks der Art

$$y = f(x_1, x_2) = x_1^2 - x_2^2 = (x_1 + x_2) \cdot (x_1 - x_2).$$

Der Problemfehler durch Rundung der Ausgangsdaten verhält sich wie

$$\left|\frac{\Delta y}{y}\right| \dot{\leq} \sum_{j=1}^{2} \left|\frac{\partial f}{\partial x_j}\frac{x_j}{f}\right| \left|\frac{\Delta x_j}{x_j}\right| \leq \left|2x_1\frac{x_1}{x_1^2 - x_2^2}\right| + \left|2x_2\frac{x_2}{x_1^2 - x_2^2}\right| = 2\left|\frac{(x_1/x_2)^2 + 1}{(x_1/x_2)^2 - 1}\right| \text{ eps}\,.$$

Für $|x_1/x_2| \approx 1$ liegt also schlechte Konditionierung vor. Zur algorithmischen Auswertung dieses Ausdrucks gibt es zwei Alternativen, wobei die Ausgangsdaten $x_1, x_2 \in A$ als *Maschinenzahlen* gegeben seien.

Algorithmus A:	Algorithmus B:
$u = x_1 \odot x_1$	$u = x_1 \oplus x_2$
$v = x_2 \odot x_2$	$v = x_1 \ominus x_2$
$\tilde{y} = u \ominus v$	$\tilde{y} = u \odot v$

Zur Rundungsfehleranalyse beachten wir, dass für die Maschinenoperationen $\circledast$ auf der Menge A der Maschinenzahlen gilt:

$$a \circledast b = \text{rd}(\text{a} * \text{b}) = (\text{a} * \text{b})(1 + \varepsilon) \quad \text{mit} \quad |\varepsilon| \leq \text{eps}\,.$$

Unter Verwendung dieser Beziehung erhalten wir für den ersten Algorithmus:

$$\begin{aligned}(\mathrm{A}) \qquad u &= x_1^2\,(1+\varepsilon_1)\,, \quad v = x_2^2\,(1+\varepsilon_2)\\ \tilde{y} &= \left[x_1^2\,(1+\varepsilon_1) - x_2^2\,(1+\varepsilon_2)\right](1+\varepsilon_3)\\ &= \underbrace{x_1^2 - x_2^2}_{=\,y} + x_1^2\varepsilon_1 - x_2^2\varepsilon_2 + \underbrace{(x_1^2 - x_2^2)}_{=\,y}\,\varepsilon_3 + O(\mathrm{eps}^2)\end{aligned}$$

$$\left|\frac{\Delta y}{y}\right| \,\dot{\leq}\, \mathrm{eps}\,\frac{x_1^2 + x_2^2 + |x_1^2 - x_2^2|}{|x_1^2 - x_2^2|} \;=\; \mathrm{eps}\,\left\{1 + \left|\frac{(x_1/x_2)^2+1}{(x_1/x_2)^2-1}\right|\right\}\,.$$

Der Rundungsfehlereinfluss wird groß für $|x_1/x_2| \sim 1$, übersteigt aber nicht den Problemfehleranteil, d. h.: Der Algorithmus A ist nach unserer Definition durchaus stabil.

Für den zweiten Algorithmus gilt:

$$\begin{aligned}(\mathrm{B}) \qquad u &= (x_1+x_2)\,(1+\varepsilon_1)\,, \quad v = (x_1-x_2)\,(1+\varepsilon_2)\\ \tilde{y} &= (x_1+x_2)\,(1+\varepsilon_1)\,(x_1-x_2)\,(1+\varepsilon_2)\,(1+\varepsilon_3)\\ &= \underbrace{x_1^2 - x_2^2}_{=\,y} + \underbrace{(x_1^2 - x_2^2)}_{=\,y}\,(\varepsilon_1+\varepsilon_2+\varepsilon_3) + O(\mathrm{eps}^2)\end{aligned}$$

$$\left|\frac{\Delta y}{y}\right| \,\dot{\leq}\, |\varepsilon_1+\varepsilon_2+\varepsilon_3| \leq 3\,\mathrm{eps}\,.$$

Algorithmus B ist offenbar i. Allg. stabiler als Algorithmus A. Es sei nochmals betont, dass hierbei von bereits gerundeten Ausfgangsdaten in A ausgegangen wird. An diesem Beispiel kann man bereits eine einfache Regel ablesen, die allgemein gültigen Charakter hat. Die Auswirkung dieser Regel wird anhand des nächsten Beispiels noch klarer werden.

Regel 1.2: *Bei der Durchführung einer numerischen Rechnung sollte man die numerisch schlechter konditionierten Operationen möglichst frühzeitig ansetzen.*

1.3.3 Auswertung von Polynomen

Gegeben sei ein allgemeines Polynom in der üblichen Monomdarstellung:

$$y = p(x) = a_0 + a_1 x + \ldots + a_n x^n\,.$$

Als Modellfall betrachten wir zunächst das Polynom

$$p(x) = a_1 x + a_2 x^2 = x(a_1 + a_2 x)\,.$$

Zu seiner Auswertung in einem Punkt ξ bietet sich der Algorithmus

$$\mathrm{A)} \quad u = \xi \odot \xi\,, \quad v = a_2 \odot u\,, \quad w = a_1 \odot \xi\,, \quad \tilde{y} = v \oplus w\,,$$

und, bei Berücksichtigung der obigen Faustregel, der Algorithmus

$$\text{B)} \quad u = a_2 \odot \xi\,, \quad v = a_1 \oplus u\,, \quad \tilde{y} = \xi \odot v\,,$$

an. Bei Algorithmus B spart man offensichtlich eine arithmetische Operation. Die zugehörige Rundungsfehleranalyse sieht wie folgt aus:

$$\begin{aligned}
\text{(A)} \qquad & u = \xi^2(1+\varepsilon_1), \quad v = a_2\xi^2(1+\varepsilon_1)(1+\varepsilon_2), \quad w = a_1\xi(1+\varepsilon_3) \\
& \tilde{y} = [a_2\xi^2(1+\varepsilon_1)(1+\varepsilon_2) + a_1\xi(1+\varepsilon_3)](1+\varepsilon_4) \\
& \quad = a_2\xi^2 + a_1\xi + (a_2\xi^2 + a_1\xi)\varepsilon_4 + a_2\xi^2(\varepsilon_1+\varepsilon_2) + a_1\xi\varepsilon_3 + O(\mathrm{eps}^2) \\
& \quad = y + y\varepsilon_4 + a_2\xi^2(\varepsilon_1+\varepsilon_2) + a_1\xi\varepsilon_3 + O(\mathrm{eps}^2)
\end{aligned}$$

$$\left|\frac{\Delta y}{y}\right| \dot{\leq} \varepsilon_4 + \frac{a_1\xi\varepsilon_3 + a_2\xi^2(\varepsilon_1+\varepsilon_2)}{a_1\xi + a_2\xi^2} = \varepsilon_4 + \varepsilon_3 + \frac{\xi}{a_1/a_2 + \xi}\,(\varepsilon_1 + \varepsilon_2 - \varepsilon_3)$$

$$\begin{aligned}
\text{(B)} \qquad & u = a_2\xi(1+\varepsilon_1), \quad v = [a_1 + a_2\xi(1+\varepsilon_1)](1+\varepsilon_2) \\
& \tilde{y} = \xi[a_1 + a_2\xi(1+\varepsilon_1)](1+\varepsilon_2)(1+\varepsilon_3) \\
& \quad = y + a_1\xi(\varepsilon_2+\varepsilon_3) + a_2\xi^2(\varepsilon_1+\varepsilon_2+\varepsilon_3) + O(\mathrm{eps}^2)
\end{aligned}$$

$$\left|\frac{\Delta y}{y}\right| \doteq \varepsilon_2 + \varepsilon_3 + \frac{\xi}{a_1/a_2 + \xi}\,\varepsilon_1\,.$$

Für $\xi \sim -a_1/a_2$ (d. h. wenn ξ nahe bei einer Nullstelle von $p(x)$ liegt) ist Algorithmus B offensichtlich etwas stabiler als Algorithmus A.

Dieses Resultat legt zur Auswertung des allgemeinen Polynoms n-ter Ordnung, ausgehend von der Darstellung

$$p(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n)\ldots))\,,$$

den folgenden Algorithmus nahe:

Definition 1.6: *Das sog. „Horner[4]-Schema“*

$$b_n = a_n, \quad k = n-1, \ldots, 0: \quad b_k = a_k + \xi b_{k+1}, \tag{1.3.11}$$

liefert den Funktionswert $p(\xi) = b_0$ *des Polynoms* $p(x)$*.*

Zur Auswertung eines Polynoms in der allgemeineren Darstellung

$$p(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \ldots + a_n(x-x_0)\ldots(x-x_{n-1})$$

[4]William George Horner (1786–1837): Irischer Mathematiker; Betreiber verschiedener Schulen; bekannt durch das „Horner-Schema“ (1830) zur Auswertung algebraischer Gleichungen; dessen Prinzip war aber bereits vorher anderen Autoren bekannt (früheste Quelle ist Zhu Shijie im China des 13. Jahrh.).

mit gewissen Zahlen x_i, $i = 0, \ldots, x_{n-1}$, wird das Horner Schema wie folgt modifizert:

$$b_n = a_n, \quad k = n-1, \ldots, 0: \quad b_k = a_k + (\xi - x_k) b_{k+1}, \quad p(\xi) = b_0. \tag{1.3.12}$$

Regel 1.3: *Die Auswertung von (gegebenen) Polynomen sollte mit Hilfe des Horner-Schemas erfolgen.*

Bei der Auswertung von Polynomen mit Hilfe des Horner Schemas spielt auch die Einsparung von arithmetischen Operationen eine Rolle. Dies wird hier und in vielen ähnlich gelagerten Fällen besonders wichtig, wenn die algorithmische Komponente sehr häufig wiederholt werden soll. Den Rechenaufwand zählt man dabei üblicherweise in Form von sog. „arithmetischen Operationen“ (abgekürzt „a. Op.“). Eine a. Op. setzt sich zusammen aus jeweils einer Addition und einer Multiplikation. Die Unterschiedliche Betrachtung von Addition und Multiplikation geschah bisher aus technischen Gründen, da auf den älteren Prozessoren eine Multiplikation deutlich mehr Zeit in Anspruch nahm als eine Addition. Inzwischen hat sich dieser Unterschied aber nivelliert, so dass Addition und Multiplikation als praktisch gleich schnell im Verhältnis zur etwas langsameren Division angesetzt werden müssten.

Beispiel 1.7: Ausführungszeiten von jeweils 10^9 a. Op.:

1) („Antiker“) Prozessor 68040 von Motorola (Macintosh Quadra 700):

Addition 1220 Sek., Multiplikation 1320 Sek., Division 2540 Sek.

2. Alter 32 Bit-Prozessor Atlon 1.4 GHz (Standard PC):

Addition $4,5$ Sek., Multiplikation $4,5$ Sek., Division 6 Sek.

3. Späterer 64 Bit-Prozessor Atlon 3500+ (High-end PC):

Addition $2,5$ Sek., Multiplikation $2,5$ Sek., Division $4,5$ Sek.

Moderne Prozessoren sind aber noch wesentlich schneller. Bei solchen Leistungsmessungen spielt die verwendete Optimierungsstufe des Compilers eine nicht unwesentliche Rolle. Diese Werte sind erreichbar sowohl mit FORTRAN, C/C++ als auch mit MATLAB.

1.4 Übungsaufgaben

Übung 1.1: Man schreibe die folgenden Ausdrücke in der Form $f(h) = O(h^p)$, für $h \searrow 0$ mit möglichst großem $p \in \mathbb{N}$, bzw. $g(n) = O(n^q)$ für $n \nearrow \infty$ mit möglichst kleinem $q \in \mathbb{N}$:

$$a) \quad f(h) = 4(h^2+h)^2 - 4h^4, \qquad b) \quad g(n) = 4(n^2+n)^2 - 4n^4,$$

$$c) \quad f(h) = \frac{e^h - e^{-h}}{2h} - 1, \qquad d) \quad g(n) = \sup_{x>0} \frac{1 - e^{-nx}}{1 - e^{-x}}.$$

e) Wie lässt sich das asymptotische Verhalten von $f(h) = 1/\ln(h)$ beschreiben?

Übung 1.2: Man untersuche die Konditionierung der folgenden Rechenoperationen:

$$a) \quad f(x_1, x_2) = \frac{x_1}{x_2} \quad (x_2 \neq 0), \qquad b) \quad f(x_1, x_2) = x_1^{x_2} \quad (x_1 > 0).$$

Sind die einfachen Operationen $f(x) = 1/x$ und $f(x) = \sqrt{x}$ gut konditioniert?

Übung 1.3: Die Ausdrücke

$$a(x) = \frac{1-x}{1+2x} - \frac{1-2x}{1+x}, \qquad b(x) = \frac{3x^2}{(1+2x)(1+x)}$$

stellen für $x > 0$ dieselbe Funktion $f(x)$ dar.

a) Wie sieht es mit der Konditionierung der jeweiligen numerischen Aufgaben, $f(x)$ für $0 < |x| \ll 1$ aus diesen Darstellungen zu berechnen?

b) Wie würde man bei der praktischen Auswertung von $f(x)$ für $0 < |x| \ll 1$ zur Gewährleistung guter numerischer Stabilität vorgehen?

Übung 1.4: Man gebe einen Weg an zur experimentellen Bestimmung der Maschinengenauigkeit

$$\text{eps} := \max_{x \in D, x \neq 0} \left| \frac{\text{rd}(x) - x}{x} \right|.$$

Dabei kann verwendet werden, dass für die Maschinenoperationen $\circledast$ gilt:

$$x \circledast y = (x * y)(1 + \varepsilon), \quad x, y \in A, \ \ |\varepsilon| \leq \text{eps}.$$

Übung 1.5 (Praktische Aufgabe):
a) Man bestimme mit einem Testprogramm die Maschinengenauigkeit des benutzten Rechners (in der jeweils verwendeten Programmiersprache).

b) Man schreibe ein (MATLAB-)Programm zur Berechnung der Exponentialfunktion e^x mit Hilfe ihrer Taylor-Summen

$$T_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}.$$

Man plotte für $n \in [0, 20]$ den relativen Fehler für die Argumente $x \in \{10, 1, -1, -10\}$. Man erkläre die schlechten Ergebnisse für negative Argumente und gebe eine Modifikation an, mit deren Hilfe negative wie positive Argumente gleich gut behandelt werden können.

Übung 1.6: Man schreibe die folgenden Ausdrücke in der Form $f(h) = O(h^m)$ bzw.

$f(h) = o(h^m)$ für $h \in \mathbb{R}_+$, $h \to 0$, mit einem möglichst großen $m \in \mathbb{N}$:

$$a) \quad f(h) = \frac{\sin(1+h) - 2\sin(1) + \sin(1-h)}{h^2} + \sin(1);$$

$$b) \quad f(h) = \frac{h}{\ln(h)}.$$

Übung 1.7: Wie groß ist in erster Näherung der relative Fehler bei der Bestimmung der Molmenge m eines idealen Gases (mit Gaskonstante $\gamma = 0,082$) aus der Formel

$$m(P,V,T) = \frac{PV}{\gamma T},$$

wenn die Temperatur T mit $200 \pm 0,5$ Grad, der Druck P mit $2 \pm 0,01$ atm und das Volumen V mit $10 \pm 0,2$ l bestimmt wurden. Welche Messung muss verfeinert werden, um den Fehler unter 1% zu drücken?

Übung 1.8: In vielen Fällen kann die Konvergenzordnung eines Grenzprozesses

$$a(h) \to a \ \ (h \to 0), \quad a(h) - a = O(h^\alpha),$$

nur experimentell bestimmt werden. Dazu werden bei bekanntem Limes a für zwei Werte h und $h/2$ die Fehler $a(h)-a$ und $a(h/2)-a$ berechnet und dann die Ordnung α über den formalen Ansatz $a(h) - a = ch^\alpha$ aus der folgenden Formel ermittelt:

$$\alpha = \frac{1}{\log(2)} \log\left(\left| \frac{a(h) - a}{a(h/2) - a} \right| \right).$$

a) Man rekapituliere die Rechtfertigung dieser Formel und überlege, wie man vorgehen könnte, wenn kein exakter Limes a bekannt ist.

b) Man bestimme die inhärenten Konvergenzordnungen für die folgenden, von Funktionen $a(h)$ und $b(h)$ abgegriffenen Werte:

h	$a(h)$	$b(h)$
2^{-1}	7.188270827204928	8.89271737217539
2^{-2}	7.095485351135761	8.971800326329658
2^{-3}	7.047858597600531	8.992881146463981
2^{-4}	7.023726226390662	8.998220339291473
2^{-5}	7.011579000356371	8.999559782988968
2^{-5}	7.005485409034109	8.999895247704067
Limes	$a(0) = 7.0$	$b(0) =?$

Übung 1.9: Man betrachte die Funktion

$$f(x) = \frac{1 - \cos(x)}{x}.$$

a) Für welche x ist die Auswertung von $f(x)$ gut bzw. schlecht konditioniert?

b) Man gebe für $|x| \ll 1$ einen stabilen Algorithmus zur Berechnung von $f(x)$ an. Dabei sei angenommen, dass $\cos(x)$ mit Maschinengenauigkeit berechnet wird. (Hinweis: Die Darstellung von f kann mit Hilfe der Rechenregeln für trigonometrische Funktionen umgeformt werden.)

Übung 1.10 (Praktische Aufgabe): Man berechne Näherungswerte

$$\sum_{k=0}^{n} \frac{x^k}{k!} \approx e^x$$

für $x = -5,5$ mit $n = 1, 2, \ldots, 30$, auf die folgenden drei Arten:

1) mit der obigen Formel;

2) mit der Umformung $e^{-5,5} = 1/e^{5,5}$ und der obigen Formel;

3) mit der Umformung $e^{-5,5} = (e^{-0,5})^{11}$ und der obigen Formel.

Der exakte Wert ist $e^{-5,5} = 0,0040867714\ldots$. Wie sind die beobachteten Effekte zu interpretieren? Dies ist ein Beispiel dafür, dass scheinbar kleine Modifikationen in numerischen Algorithmen gravierende Konsequenzen für die Approximationsgenauigkeit haben können. Welche Ergebnisse ergeben sich, wenn die Auswertung der Taylor-Polynome mit Hilfe des Horner-Schemas erfolgt?

Übung 1.11: Sei $A \in \mathbb{R}^{n \times n}$ beliebig gegeben. Man gebe einen Algorithmus an zur Auswertung des Matrixpolynoms

$$p(A) = \sum_{i=0}^{m} a_i A^i$$

mit Koeffizienten $a_i \in \mathbb{R}$, der möglichst wenig Speicherplatz und arithmetische Operationen (1 a. Op. = 1 Mult. + 1 Add.) benötigt.

Übung 1.12: Es seien die Nullstellen eines Polynoms $p(x) = \sum_{i=0}^{m} a_i x^i$ zu bestimmen. Man zeige, dass für eine Näherung $\tilde{z}$ zu einer einfachen Nullstelle $z \neq 0$ in erster Näherung die folgende Abschätzung gilt:

$$\Big|\frac{\tilde{z} - z}{z}\Big| \dot{\leq} \Big|\frac{p(\tilde{z})}{p'(z)z}\Big|.$$

Dies motiviert die Genauigkeitskontrolle bei der Berechnung von Nullstellen von Polynomen in der übernächsten praktischen Aufgabe. (Hinweis: Die Aufgabe ist leichter als sie aussieht; Taylor-Entwicklung.)

Übung 1.13: Die Funktion $f(x) = x + 1$ stelle eine physikalische Größe dar, von der Werte $\tilde{f}(x_i) \approx f(x_i)$ an äquidistant verteilten Punkten

$$x_i = ih, \quad 0 \le i \le n := 10^3, \quad h = 10^{-3},$$

mit einem maximalen Fehler von $0,1\%$ gemessen werden. Man zeige, dass bei der Approximation der Ableitungswerte $f'(x_i)$ mit dem zentralen Differenzenquotienten

$$f'(x_i) \approx \frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h}, \quad i = 1, \ldots, n-1$$

aus diesen Werten ein relativer Fehler von 100% auftreten kann. Dies zeigt die Fragwürdigkeit der Approximation von Ableitungen durch Differenzenquotienten. (Hinweis: Man konstruiere spezielle Störungen.)

Übung 1.14 (Praktische Aufgabe): Man schreibe ein Programm zur Berechnung der reellen Lösungen der quadratischen Gleichung

$$p(x) = ax^2 + bx + c = 0,$$

zu gegebenen $a, b, c \in \mathbb{R}$. Es sollen alle möglichen Fälle der Degenerierung (z. B.: $a = 0$) berücksichtigt und der Einfluß des Rundungsfehlers minimiert werden. Man erprobe das Programm anhand der folgenden Fälle:

$a:$	0	0	0	0	2	2	2	4	1	1	1	-1	-1	-4	-1	-4	-1	-1	$2,5 \cdot 10^9$
$b:$	0	0	1	2	0	0	0	2	2	2	2	0	0	0	2	8	2	2	-10^5
$c:$	0	1	0	1	0	1	-4	0	-3	1	2	0	-1	2	0	12	-1	-5	1

Die berechneten Lösungen sollen akzeptiert werden, wenn das folgende heuristische Kriterium erfüllt ist (s. Aufgabe 2):

$$\left| \frac{p(\tilde{z})}{p'(\tilde{z})\tilde{z}} \right| < 10^{-12}.$$