Vorlesungsmitschrift

Algorithmische Mathematik 1

Wintersemester 2021/2022 *
Universität Bonn

Fabian

30.01.2022

Online aufrufbar auf https://git-fabus.github.io.
Veränderungsvorschläge und Verbesserungen bitte an git-fabus@uni-bonn.de.

*Letzte Änderung: (None) Commit: (None)

Preface

Das folgende Skript sind meine persönlichen Notizen für die Vorlesung *Algorithmische Mathematik 1*, welche von Dr. Jürgen Dölz an der Universität Bonn gehalten wurden. Diese Mitschriften wurden intial als Klausurvorbereitung verfasst.
Seit dem wurde das Skript ständig überarbeitet und soll nun als Vorlesungskript für den jeweiligen Vortragenden der Alma 1 dienen.

Inhaltsverzeichnis

Einführung

Algorithmische Mathematik -Was ist das?

Gegenstand der Alma ist die Konstruktion und ANalyse effizienter Algorithmen zur Lösung mathematischer Problemstellugnen mit Hilfe des Computers.
Damit liegt sie im Bereich der Angewandten Mathematik. Konkrete Problemstellungen ergeben sich oft aus technischen Problemen, Naturwissenschaften, Medizin etc.
Man kann hierbei annehmen, dass verschiedene Problemstellungen aus der Anwendungen oft zu ähnlichen oder gleichen mathematischen Modellen zurückgeführt werden können.

0.1 Beispiel. Ein typisches Problem ist das lösen eines linearen Gleichungssystems.

Im Rahmen dieser Vorlesung konzentrieren wird uns auf die Teilbereiche

a) Numerik

b) Diskrete Mathematik

c) Statistik

der Angewandten Mathematik

1 Zahlendarstellung am Computer

1.1 Zahlensystem

Die Darstellung von Zahlen basiert auf sogenannten *Zahlensystemen*. Diese Zahlensysteme unterscheiden sich in der Wahl des zugrundeliegendes Alphabets. Unsere Zahl entspricht dann einem Wort, bestehend aus Elementen des Alphabets.

1.1 **Definition** (Alphabet). Es sei $\mathbb{N} = \{1, 2, 3 \ldots\}$ und $b \in \mathbb{N}$.
Wir bezeichnen mit $\sum_b$ das Alphabet des *b-adischen Zahlensystems*

1.2 **Beispiel.** Verschiedene Zahlensysteme

a) Dezimalsystem: $\sum_{10} = \{0, 1, \ldots, 9\}$
wie $(384)_1 0$

b) Dual bzw. Binärsystem $\sum_2 = \{0, 1\}$
wie $(42)_1 0 = (101010)_2$

c) ...

Um die Zahlendarstellung sinnvoll zu nutzen Ergibt sich folgendes:

1.3 **Satz.** Seien $b, n \in \mathbb{N}, b > 1$.
Dann ist jede ganze, nicht-negative Zahl z mit $0 \leq z \leq b^n - 1$ *eindeutig* als Wort der Länge n über $\sum_b$ darstellbar durch:

$$z = \sum_{i=0}^{n-1} z_i b^i$$

mit $z_i \in \sum_b$ für alle $i = 0, 1, \ldots, n-1$.
Wir schreiben vereinfachend:

$$z = (z_{n-1}, \ldots, z_0)_b$$

Beweis. • Induktion

◊ *Induktionsanfang*: $\underline{z < b}$ hat die eindeutige Darstellung $z_0 = z$ und $z_i = 0$ sonst.

◊ *Induktionsschritt*: $\underline{z - 1 \to z \geq b}$

Betrachte

$$z = \left\lfloor \frac{z}{b} \right\rfloor \cdot b + (z \mod b)$$

Da $\hat{z} < z$ besitzt die eindeutige Darstellung

$$\hat{z} = (\hat{z_{n-1}}, \ldots, \hat{z_0})_b$$

Bemerke $\hat{z_{n-1}} = 0$, da

$$\left(\hat{z_{n-1}} b^{n-1}\right) b \leq z \leq b^n - 1$$

Also ist z_b eine n-stellige Darstellung von z in b-adischen Zahlensystem

- Eindeutigkeit wird durch Widerspruch gezeigt.
  Angenommen: Es gibt zwei verschiedene Darstellungen
  $$z = \left(z_{n-1}^{(2)}, \ldots, z_0^{(2)}\right)_b = \left(z_{n-1}^{(1)}, \ldots, z_0^{(1)}\right)_b$$
  Sei $m \in \mathbb{N}$ der größte Index mit $z_m^{(1)} \neq z_m^{(2)}$,
  Ohne Beschränkung der Allgemeinheit kann gesagt werden $z_m^{(1) > z_m^{(2)}}$.
  Dann müssten die Stellen $0, 1, \ldots, m-1$ den niedrigeren Wert von $z_m^{(2)}$ kompensieren.
  Die größte mit diesen Stellen darstellbare Zahl ist aber:
  $$\sum_{i=0}^{m-1} (b-1)b^i = (b-1)\sum_{i=0}^{m-1} b^i = b^m - 1.$$
  Da b^m der kleinstmögliche Wert der fehlende Stelle m ist, kann diese aber nicht kompensiert werden. Dies ist ein Widerspruch.

□

Durch den Beweis ergibt sich sofort ein Algorithmus zur Umwandlung einer Zahl in ein anderes Zahlensystem:

1.4 Beispiel. Umwandlung von $(1364)_{10}$ in das Oktalsystem.

- $1364 = 170 \cdot 8 + 4$
- $170 = 21 \cdot 8 + 2$
- ...
- $0 \cdot 8 + 2$

$\Longrightarrow (2524)_8$

Daten: Dezimalzahl $z \in \mathbb{N}_0$, Basis $b \in \mathbb{N}$
Ergebnis: b-adische Darstellung $(z_{n-1}, \ldots, z_0)_b$
Initialisiere $i = 0$
solange $z > 0$ **tue**
  $z_i = z \mod b$
  $z = \left\lfloor \frac{z}{b} \right\rfloor$
  $i = i + 1$
Ende

Algorithmus 1: Bestimmung der b-adischen Darstellung

Beobachtung Wir sehen, dass das Honor-Schmea nur eine Schleife, Additionen und Multiplikationen benötigt. Diese Operationen können wir am Computer durchführen. Im Gegensatz dazu steht die Potenz b^i in modernen Programmiersprachen zwar zur Verfügung, wird aber im Hintergrund oft auf Multiplikationen zurückgeführt. Man überprüft leicht, dass das Honor-Schema weniger Multiplikationen benötigt und somit schnellt ist.

1.2 Vorzeichen-Betrag-Darstellung

Um auch Zahlen mit Vorzeichen am Computer darstellen zu können, betrachten wir im Folgenden die Vorzeichen-Betrag-Darstellung für Binärzahlen. Das Binäralphabet besteht nur aus 0 und 1, welche wir auch als *Bits* bezeichnen. Bei einer Wortlänge von n Bits wir das erste Bit als Vorzeichen verwendet, die restlichen $n-1$-Bits für den Betrag der Zahl. Da die 0 die Darstellung $+0$ und $-O$ besitzt, können wir insgesamt $2^n - 1$ Zahlen darstellen.

1.5 Beispiel. Für $n = 3$

Bitmuster	Dezimaldarstellung
000	$+0$
001	$+1$
...	...
100	-0
...	...
111	-3

Aber: Diese Darstellung am Computer ist unpraktisch, da die vier Grundrechenarten auf Hardwareebene typischerweise mit Hilfe von Addition und Zusatz-Logik umgesetzt werden.

Lösung: Komplementdarstellung

1.3 Komplementdarstellung

1.6 Definition ((b-1)-Komplement). Sei $z = (z_{n-1} \dots z_1 z_0)_b$ eine n-stellige b-adische Zahl. Das *(b-1)-Komplement* $K_{b-1}(z)$ ist definiert als:

$$K_{b-1} = (b-1-z_{n-q}, \dots, b-1-z_0)_b$$

Geben wir hierzu direkt ein paar Beispiele an

1.7 Beispiel. Komplemente

- $K_9((325)_{10}) = (674)_{10}$ (9er-Komplement im 10-er System)
- $K_1((10110)_2) = (01001)_2$ (1er-Komplement im 2er System)

1.8 Definition (b-Komplement). Das b-Komplement einer b-adischen Zahl $z \neq 0$ ist definiert als

$$K_b(z) = K_{b-1}(z) + (1)_b$$

1.9 Beispiel. • $K_{10}((325)_{10}) = (674)_{10} + (1)_{10} = (675)_{10}$

1.10 Lemma. Für jede n-stellige b-adische Zahl z gilt:

- i) $z + K_{b-1}(z) = (b-1, \dots, b-1)_b = b^n - 1$
- ii) $K_{b-1}\left(K_{b-1}(z)\right) = z$

Ist außerdem $z \neq 0$ so gilt:

- iii) $z + K_b(z) = b^n$
- iv) $K_b(K_b(z)) = z$

Beweis. Hilfssatz

(i) Durch nachrechnen:

$$\begin{aligned} z + K_{b-1}(z) &= (z_{n-1} \dots z_0)_b + (b-1-z_{n-1}, \dots, b-1-z_0)_b \\ &= \sum_{i=0}^{n-1} z_i b^i + \sum_{i=0}^{n-1} (b-1-z_i) b^i \\ &= \sum_{i=0}^{n-1} (b-1) b^i = (b-1, \dots, b-1)_b \\ &= (b-1) \sum_{i=0}^{n-1} b^i \\ &= (b-1) \left(\frac{b^n - 1}{b-1} \right) \\ &= b^n - 1 \end{aligned}$$

- (ii) per Definition
- (iii) Nachrechnen:

$$z + K_b(z) = z + K_{b-1} + 1 = b^n - 1 + 1 = b^n$$

- Definiere $\hat{z} = K_b\left(z\right) = K_{b-1}(z) + (1)_b > 0$ und rechne

$$z + K_b(z) = b^n = \hat{z} + K_b \hat{z} + K_b\left(K_b(z)\right) \implies \text{Behauptung.}$$

□

1.11 Bemerkung. Modifikation

- Die 3. Aussage gilt auch für $z = 0$, falls man dann bei der Addition von 1 die Anzahl der Stellen erweitert.
- die 4. Aussage gilt für $z = 0$, falls überall im Beweis modulo b^n gerechnet wird.

Außerdem impliziert die 3. Aussage des Lemmas, dass

$$K_b(z) = b^n - z. \tag{1.1}$$

Dies können wir geschickt zum Darstellen der b^n verschiedenen ganzen Zahlen z mit

$$-\left\lfloor \frac{b^n}{2} \right\rfloor \leq z \leq \left\lceil \frac{b^n}{2} \right\rceil - 1$$

nutzen. Diesen Bereich nennt man darstellbaren Bereich.

1.12 **Definition** (b-Komplement-Darstellung). Die b-Komplement-Darstellung $(z)_{K_b} = (z_{n-1} \dots z_0)_b$ einer Zahl $z \in \mathbb{Z}$ im darstellbaren Bereich ist definiert als:

$$(z)_{K_b} = \begin{cases} (z)_b & \text{falls } z \geq 0 \\ (K_b(|z|))_b & \text{falls } z < 0 \end{cases}$$

1.13 **Beispiel.** Der darstellbare Bereich.

- Sei $b = 10$, $n = 2$.
  Dann impliziert (1.1), dass

  $$K_{10}(50) = 10^2 - 50 = 50$$

  $$K_{10}(49) = 100 - 49 = 51.$$

  Der darstellbare Bereich ist nun

  $$-50 \leq z \leq 49$$

  und hat konkrete Darstellungen:

Darstellung	Zahl
0	+0
1	+1
...	...
49	+49
50	−50
...	...
99	−1

- Sei $b = 2$, $n = 3$ Der darstellbare Bereich ist $-4 \leq z \leq 3$.

Bitmuster	Dezimaldarstellung
000	0
001	1
...	...
100	−4
...	...
111	−1

Wir betrachten nun Addition und Subtraktion zweier Zahlen in b-Komplement-Darstellung. Hierzu bezeichne $(x)_{K_b} \oplus (y)_{K_b}$ die ziffernweise Addition der Darstellungen von x und y mit Übertrag (”schriftlich rechnen”), wobei ein eventueller Überlauf auf die $(n+1)$-te Stelle vernachlässigt wird (wir rechen also immer mit modulo b^n.

1.14 **Satz** (Addition in b-Komplement-Darstelllung). Seien x und y zwei n-stellige, b-adische Zahlen und x,y und $x+y$ im darstellbaren Bereich. Dann gilt:

$$(x+y)_{K_b} = (x)_{K_b} \oplus (y)_{K_b}$$

Beweis. Wir betrachten dazu mehrere Fälle.

- Fall $x, y \geq 0$:

$$\begin{aligned}(x)_{K_b} \oplus (y)_{K_b} &\overset{\text{Def.}}{=} ((x)_b + (y)_b) \mod b^n \\ &\overset{\text{Def.}}{=} (x+y) \mod b^n \\ &\overset{\text{Def.}}{=} (x+y)_{K_b}\end{aligned}$$

  Der letzte Schritt ist möglich, da $(x+y)_{K_b}$ im darstellbaren Bereich sind.

- Fall $x, y < 0$

$$\begin{aligned}(x)_{K_b} \oplus (y)_{K_b} &\overset{Def.}{=} ((K_b(|x|))_b + (K_b(|y|))_b) \mod b^n \\ &= ((K_b(|x|)) + (K_b\,(|x|))) \mod b^n \\ &= (b^n - |x| + b^n - |y|) \mod b^n \\ &= (x+y) \mod b^n \\ &= (x+y)_{K_b}\end{aligned}$$

- Fall $x \geq 0, y < 0$:

$$\begin{aligned}(x)_{K_b} \oplus (y)_{K_b} &\overset{Def.}{=} ((x)_b + (K_b(|y|))_b) \mod b^n \\ &= (x + K_b(|y|)) \mod b^n \\ &= (x + b^n - |y|) \mod b^n \\ &= (x+y) \mod b^n \\ &= (x+y)_{K_b}\end{aligned}$$

- Fall $x < 0, y \geq 0$: Analog

$\square$

1.15 **Satz** (Subtraktion in b-Komplement-Darstellung). Seien x und y n-stellige b-adische Zahlen und x,y und x-y im darstellbaren Bereich. Dann gilt:

$$(x-y)_{K_b} = (x)_{K_b} \oplus (K_b(y))_{K_b}$$

Beweis. • Fall $y = 0$ nichts zu zeigen.

- $y \neq 0$: (1.1) impliziert:
$$-y = K_b(y) - b^n$$
mit modulo b^n-rechnen folgt, dass
$$(-y)_{K_b} = (K_b(y))_{K_b}$$
ist und somit:
$$\begin{aligned}(x-y)_{K_b} &= (x+(-y))_{K_b}\\ &= (x)_{K_b} \oplus (-y)_{K_b}\\ &= (x)_{K_b} \oplus (K_b(y))_{K_b}\end{aligned}$$

□

1.16 **Beispiel.** Für $b = 10$ und $n = 2$ ist der darstellbare Bereich $-50 \leq z \leq 49$

- $(20+7)_{K_{10}} = (20)_{K_{10}} \oplus (7)_{K_{10}} = (27)_{K_{10}} = 27$
- $28 - 5 = 28 + (-5) = (28)_{K_{10}} \oplus (95)_{K_{10}} = (23)_{K_{10}} = 23$
- $-18 - 20 = (-18) + (-20) = \ldots = -28$

Die darstellbaren Zahlen kann man sich beim K_b-Komplement als Zahlenrad vorstellen.

Achtung Ein eventueller Überlauf bzw. Unterlauf wird im Allgemeinen nicht aufgefangen.

1.17 **Beispiel.** In n-stelliger Binärarithmetik ist die größte darstellbare Zahl $x_{max} = (011\ldots1)_{K_2}$ gleich $2^{n-1} - 1$. Hingegen ist $x_{max} + 1 = (100\ldots0)_{K_2}$ und wir als -2^{n-1} interpretiert.

1.4 Fest-Komma-Darstellung

1.18 **Definition.** Bei der Festkommadarstellung einer n-stelligen Zahl werden k Vorkomma und n-k Nachkommastellen definiert:

$$z = +-(z_{k-1}\ldots z_0.z_{-1}\ldots z_{k-n})_b = +-\sum_{i=k-n}^{k-1} z_i b^i$$

1.19 **Beispiel.** Im Zehner System wie gehabt.
Im Binärsystem ergibt sich folgendes: $(101.01)_2 = 2^2 + 2^0 + 2^{-2} = 5.25$

Achtung Im Gegensatz zur Darstellung ganzer Zahlen können bereits bei der Konvertierung von Dezimalzahlen in das b-adische Zahlensystem Rundungsfehler auftreten.

1.20 **Beispiel.** $(0.8)_{10} = (0.110\overline{1100})_2$

Das größte Problem der Festkommadarstellung ist alllerdings, dass der darstellbare Bereich stark eingeschränkt ist und schlecht aufgelöst ist, da der Abstand zwischen zweier Zahlen immer gleich ist.

1.21 **Beispiel.** Die kleinste darstellbare Zahl in Fixkommadarstellung ist

$$z_1 = (0 \dots 0.0 \dots 01)_b$$

Die zweitkleinste Zahl ist $z_2 = 2z_1$. Wir wollen $x = \frac{z_1+z_2}{2}$ in Fixkommadarstellung darstellen, müssen wir entweder zu z_1 abrunden oder zu z_2 aufrunden.

1.22 Der relative Fehler dieses Runden ist:

$$\frac{|x - z_1|}{|x|} = \frac{1}{3}$$

Analog für z_2.
Solche Fehler machen jede Rechnung unbrauchbar.

1.5 Gleitkommadarstellung

1.23 **Definition** (Gleitkommadarstellung). Die Gleitkommastellung einer Zahl $z \in \mathbb{R}$ ist gegeben durch:

$$z = + - m \cdot b^e$$

mit einer Matisse m, dem Exponenten e und der Basis b.

Der Einfachheit halber nehmen wir an, dass die Basis für alle Zahlen gleich ist, obwohl sie prinzipiell verschieden sein könnte.

1.24 **Beispiel.** Die Gleitkommadarstellungen von:

- $(-384.753)_{10} = -3.84753 \cdot 10^2$
- $(0.00042)_{10} = 4.2 \cdot 10^{-4d}$
- $(1010.101)_2 = (1.010101)_2 \cdot 2^3$

Achtung: Die Gleitkommadarstellung ist nicht eindeutig!

1.25 **Definition.** Die Matisse m heißt normalisiert, falls $m = m_1.m_2m_3 \dots m_t$ mit $1 \leq m_1 \leq b$.

Um die eindeutige, normalisierte Gleitkommadarstellung im Computer zu speichern, müssen wir festlegen, wie viele Stellen für Matisse und Exponent zur Verfügung gestellt werden. Dies resultiert in der Menge

$$F = F(b, t, e_{min}, e_{max}) = \{z = + - m_1.m_2m_3 \dots m_t \cdot b^e | e_{min} \leq e \leq e_{max}\}$$

Da 0 nicht in dieser Form dargestellt werden kann, reserviert man dafür eine spezielle Ziffernfolge in Matisse und Exponent.

1.26 **Bemerkung.** Das Hidden Bit ist hilfreich

- Für $b = 2$ kann die erste Ziffer m_1 der Matisse weggelassen werden (Hidden Bit)
- Um den Exponenten besser vergleichen zu können, verwendet man oft die Exzess- oder Bias-Darstellung. Durch Addition der Exzesser $|e_{min}| + 1$ wird der Exponetn auf den Bereich $1, 2, \ldots, |e_{min}| + e_{max} + 1$ transformiert.

1.27 **Beispiel** (IEEE 754 Standard). Betrachte binäre Gleitkommazhalen mit 64 Bit auf dem Computer.

- 52 Bit für die Matisse in Hidden Bit Darstellung
- 11 Bit für den Exponenten mit $e_{min} = -1022$ und $e_{max} = 1023$ gespeichert in Exzessdarstellung.
- 1 Bit als Vorzeichen.

Weiter Fälle können Online nachgelesen werden.

Genauigkeit der Gleitkommadarstellung

Da die Menge aller Zahlen in $F = F(b, t, e_{min}, e_{max})$ endlich ist, müssen wir Zahlen in $\mathbb{R} \setminus F$ geeignet annähern.

1.28 **Definition.** Die Rundung ist eine Abbildung $rd\colon \mathbb{R} \to F$ mit

- $rd(a) = a$, $a \in F$
- $rd(z)$, $z \in \mathbb{R}$ ist gegeben so, dass $|z - rd(z)| = \min\limits_{a \in F}(z - a)$ für alle $z \in \mathbb{R}$.

1.29 **Definition** (Maschinengenauigkeit). Den maximalen relativen Rundungsfehler ε_{mach} für $z_{min} \leq |z| \leq z_{max}$ nennt man Maschinengenauigkeit. Die Stellen der Mantisse heißen signfikante Stellen. Wenn die Mantisse t-stellig ist, spricht man von t-stelliger Arithmetik.

1.30 **Satz** (Maschinengenauigkeit von F). Die Maschinengenauigkeit ε_{mach} für $F = F(b, t, e_{min}, e_{max})$ ist:

$$\varepsilon_{mach} = \frac{1}{2} b^{1-t}.$$

Beweis. Betrachte relativen Rundungsfehler ε, der vom Abscheiden nicht signifikanten Stellen herrüht. Sei $z_{min} < x < z_{max}$ und $\tilde{x}$ die durch Abschneiden entstehende Zahl. Dann gilt:

$$\begin{aligned}
\varepsilon &= \frac{|x - \tilde{x}|}{|x|} \\
&= \frac{|x_1.x_2x_3 \ldots x_t x_{t+1} \ldots \cdot b^e - x_1x_2x_3 \ldots x_t \cdot b^e}{|x|} \\
&= \frac{|0.x_{t+1} \ldots| \cdot |b^{e+1-t}|}{|x|}
\end{aligned}$$

Da $|0.x_{t+1}\dots| < 1$ und $b^e \leq |x|$ gilt:

$$\varepsilon < \frac{b^{e+1-t}}{b^e} = b^{1-t}$$

Da der Rundungsfehler ε_{mach} höchsten halb so groß ist wie der Abschneidefehler folgt die Behauptung. □

Die Maschinengenauigkeit ε_{mach} ist die wichtigste Größe zur Beurteilung der Genauigkeit von Gleitkommarechnungen am Computer. Sie gibt Aufschluss zur Anzahl signifikanter Stellen zur Basis b. Die zugehörige Anzahl s Stellen im Dezimalsystem erhält man durch das Auflösen von:

$$\varepsilon_{mach} = \frac{1}{2}b^{1-t} = \frac{1}{2}10^{1-s}$$

nach s. Es folgt daher:

$$s = \lfloor 1 + (t-1)\log_{10}(b) \rfloor$$

Für den IEEE 754 Standard mit $b = 2, t = 53$ erhält man $s = 16$ signifikante Stellen.

2 Fehleranalyse

2.1 Rechnerarithmetik

2.1 **Beispiel.** Betrachte $F = F(10, 5, -4, 5)$ und Maschinenzahlen

$$x = 2.5684 \cdot 10^0 = 2.56840000$$
$$y = 3.2791 \cdot 10^{-3} = 0.0032791$$

Es gilt:

$$\left.\begin{aligned} x + y &= 2.5716792 \\ x - y &= 2.5651209 \\ x \cdot y &= 0.00842204044 \\ \tfrac{x}{y} &= 783.2637004 \end{aligned}\right\} \notin F$$

2.2 **Bemerkung.** Die Menge $F(b, t, e_{min}, e_{max})$ ist nicht abgeschlossen bezüglich der Grundrechenarten und können somit im Allgemeinen nicht im Computer implementiert werden.

Lösung Wir runden das Ergebnis und implementieren so eine Pseudoarithmetik. Das bedeutet, wir ersetzen $\circ \in \{+, -, \cdot, \div\}$ durch $\boxcircle \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$ definiert durch:

$$x \boxcircle y := rd(x \circ y) \tag{2.2}$$

Auf Hardwareebene wird üblicherweise mit einer längeren Matisse gearbeitet und dann normalisiert und gerundet. Dies entspricht dem IEEE 754 Standard.

2.3 **Bemerkung.** Für $|x|, |y|, |x \circ y| \in [z_{min}, z_{max}]$ impliziert (2.2), dass

$$\frac{|x \boxcircle y - x \circ y|}{|x \circ y|} = \frac{rd(x \circ y - x \circ y)}{|x \circ y|} \leq \varepsilon_{mach}$$

Das bedeutet, dass $\boxcircle$ im Computer bestmöglich umgesetzt ist.

2.4 **Beispiel.** Betrachte $F = F(10, 5, -4, 5)$

- Setze:
  - $a = 0.98765$
  - $b = 0.012424$
  - $c = -0.0065432$

  Dann gilt:

  $$(a + b) + c = a + (b + c) = 0.9925208$$

Numerisch gilt:

$$(0.98765 \boxplus 0.012424) \boxminus 0.0065432 = rd(0.9935568) = 0.99356$$

und

$$0.98765 \boxplus (0.012424 \boxminus 0.0065432) = rd(0.9935308) = 0.99353$$

- Setze
  - $a = 4.2832$
  - $b = -4.2821$
  - 5.7632

  Dann gilt
  $$(a + b) \cdot c = a \cdot c + b \cdot c = 0.006339520000001$$
  Numerisch gilt:

2.5 **Bemerkung.** Mathematisch äquivalente Algorithmen auf Fließkommazahlen können je nach Implementierung zu wesentlich unterschiedlichen Ergebnissen führen, selbst wenn die Eingangszahlen exakt dargestellt werden.

2.2 Auslöschung

Unglücklicherweise pflanzen sich numerische Fehler, zum Beispiel durch Rundung im Verlauf eines Algorithmus fort.

2.6 **Lemma** (Fehlerfortpflanzung). Es seien $x, y \in \mathbb{R}$ mit Datenfehlern $\Delta x, \Delta y \in \mathbb{R}$ behaftet, welche

$$|\frac{\Delta x}{x}|, |\frac{\Delta y}{y}| \ll 1$$

erfüllen. Für $\circ \in \{+, -, \cdot, \div\}$ gilt für den fortgepflanzten Fehler

$$\Delta(x \circ y) \coloneqq (x + \Delta x) \circ (y + \Delta y) - x \circ y,$$

dass

$$\begin{aligned} \frac{\Delta(x \pm y)}{x \pm y} &= \frac{x}{x \pm y}\frac{\Delta x}{x} \pm \frac{y}{x \pm y}\frac{\Delta y}{y} \\ \frac{\Delta(x \cdot y)}{xy} &\approx \frac{\Delta x}{x} + \frac{\Delta y}{y} \\ \frac{\Delta(\frac{x}{y})}{\frac{x}{y}} &\approx \frac{\Delta x}{x} - \frac{\Delta y}{y} \end{aligned}$$

Hierbei bedeutet "$\approx$", dass Terme mit $(\Delta x)^2, (\Delta y)^2, \Delta x \Delta y$ vernachlässigt werden.

Beweis. Übung. $\square$

Für $\cdot, \div$ addieren bzw. subtrahieren sich die Fehler.

Achtung: Ist $|x \pm y|$ wesentlich kleiner als $|x|$ oder $|y|$, kann der relative Fehler massiv verstärkt werden. Dieses Phänomen heißt Auslöschung.
Bei der Konstruktion von Algorithmen sollte Auslöschung möglichst vermieden werden.

2.7 Beispiel. Betrachte

$$x^2 - 2px + q = 0 \tag{2.3}$$

mit Lösungen:

$$x_{1,2} = p \pm \sqrt[2]{p^2 - q}$$

Daten: $p, q \in \mathbb{R}$ so ,dass (2.3) lösbar ist
Ergebnis: Nullstellen x_1, x_2 von (2.3).
$d = \sqrt{p \cdot p - q}$
$x_1 = p + d$
$x_2 = p - d$

Algorithmus 2: naive Nullstellenberechung

Mit $p = 100$, $q = 1$ in dreistelliger, dezimaler Gleitkommaarithmetik ergibt sich:

$$\begin{aligned} d = \sqrt{10000 - 1} = \sqrt{rd(9999)} = \sqrt{10000} &= 100 \\ x_1 = 100 + 100 &= 200 \\ x_2 = 100 - 100 &= 0 \end{aligned}$$

Die exakten Werte sind $x_1 \approx 199.99, x_2 \approx 0.00500$, welche in dreistelliger, dezimaler Gleitkommaarithmetik als $x_1 = 200, x_2 = 0$ dargestellt werden.
Gemäß 1.30 ist die Maschinengenauigkeit:

$$\varepsilon_{mach} = \frac{1}{2} b^{1-t} = \frac{1}{2} 10^{1-3} = 0.005$$

Der relative Fehler $\frac{|0-0.005|}{|0.005|} = 1$ ist also zu 100 Prozent falsch.

Wir können die Genauigkeit von x_2 erhöhen, indem wir den Wurzelsatz von Vieta $x_1 x_2 = q$ verwenden.

Daten: $p, q \in \mathbb{R}$ so ,dass (2.3) lösbar ist.
Ergebnis: Nullstellen x_1, x_2 (2.3).
$d = \sqrt[2]{p \cdot p - q}$
wenn $q \geq 0$ **dann**
  $x_1 = p + d$ **sonst**
    $x_1 = p - d$
  Ende
Ende
$x_2 = \frac{q}{x_1}$

Algorithmus 3: verbesserte Nullstellenberechnung

Wir erhalten nun $d = 100, x_1 = 200, x_2 = \frac{1}{200} = 0.005$

2.3 Vorwärts- und Rückwertsanalyse

Abstrakt gesehen entspricht das Lösen eines Problems dem Auswerten einer Funktion f. Beim numerischen Auswerten von f können verschiedene Fehler passieren:

2.8 Definition (Fehlerarten). Sei $D \subset \mathbb{R}$ eine Menge von Eingangsdaten und $W \subset \mathbb{R}$ die Menge der möglichen Ergebnisse. Wir unterscheiden folgende Fehlerarten bei der Auswertung von $f: D \to W$.

- Datenfehler Typischerweise sind die Eingangsdaten $x \in D$ nicht exakt, sondern mit einem Datenfehler Δx behaftet. Die gestörten Eingangsdaten $\tilde{x} = x + \Delta x$ produzieren einen Fehler $f(x + \Delta x) - f(x)$.
- Verfahrensfehler Exakte Verfahren enden bei exakter Rechnung nach endlich vielen Operationen mit dem exakten Ergebnis. Näherungsverfahren enden in Abhängigkeit bestimmter Kriterien mit einer Näherung $\tilde{y}$ für die Lösung $y \in W$.
- Rundungsfehler Verursacht durch Maschinenzahlen und Rundungen während der Arithmetik.

Ein Teilgebiet der Numerik versucht diese Fehler durch eine Fehleranalyse zu quantifizieren. Hierzu verfolgt man die Auswirkungen von allen Fehlern, die in den einzelnen Schritten vorkommen können.

2.9 Definition (Analysearten). Bei der *Vorwärtsanalyse* wird der Fehler von Schritt zu Schritt verfolgt und der akkumulierte Fehler für jedes Teil-Ergebnis abgeschätzt.
Bei der *Rückwertsanalyse* geschieht die Verfolgung des Fehlers hingegen so, dass jedes Zwischenergebnis als exakt berechnetes Ergebnis zu gestörten Daten interpretiert wird, d.h , der akkumulierte Fehler wird als Datenfehler interpretiert.

2.10 Beispiel. Betrachte $f(x, y) = x + y$

- Vorwärtsanalyse: $\boxed{f} = x \boxplus y = (x + y)(1 + \varepsilon)$
- Rückwärtsanalyse: $x \boxplus y = x(1 + \varepsilon) + y(1 + \varepsilon) = f(x(1 + \varepsilon), y(1 + \varepsilon)$

mit $|\varepsilon| \leq \varepsilon_{mach}$

In der Praxis ist die Vorwärtsanalyse kaum durchführbar. Für die meisten Algorithmen ist, wenn überhaupt, nur eine Rückwärtsanalyse bekannt.

2.11 Beispiel. Fortsetzung von Beispiel 2.7
Führe Rückwärtsanalyse durch zu Algorithmus 2 für die Betrags-kleinere Nullstelle x_2 durch: Dann ist

$$f(p, q) = p - \sqrt[2]{p^2 - q}$$

mit $p > 0$. Für $|\varepsilon_i| \leq \varepsilon_{mach}, i = 1, 2, 3, 4$ betrachten wir:

$$\begin{aligned}
\left(p - \sqrt[2]{(p^2(1+\varepsilon_1) - q)(1+\varepsilon_2)}(1+\varepsilon_3)\right)(1+\varepsilon_4) &= p(1+\varepsilon_4) - \sqrt[2]{(p^2(1+\varepsilon_1) - q)(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2} \\
&= p^2(1+\varepsilon_1)(1+\varepsilon_3)^2(1+\varepsilon_4)^2 - q(1+\varepsilon_2)(1+\varepsilon_3)^2(1+\varepsilon_4)^2 \\
&= \dots \\
&= p(1+\varepsilon_4) - \sqrt[2]{p^2(1+\varepsilon_4)^2 - q(1+\varepsilon_7)} \\
&= f(p(1+\varepsilon_4), q(1+\varepsilon_7))
\end{aligned}$$

Die Abschätzung für ε_7 explodiert, falls $0 < |q| \ll 1 < p$. Dies war in Beispiel 2.7 der Fall.

2.4 Kondition und Stabilität

Gegeben sei eine stetige und differenzierbare Funktion

$$f : \mathbb{R} \to \mathbb{R}, x \mapsto y = f(x)$$

Für fehlerhafte Daten $x + \Delta x$ mit kleinen Fehler Δx gilt:

$$\frac{f(x+\Delta x) - f(x)}{\Delta x} \approx f'(x).$$

Für den absoluten Datenfehler Δy gilt:

$$\Delta y = f(x + \Delta x) - f(x) \approx f'(x) \cdot \Delta x$$

und für den relativen Fehler:

$$\frac{\Delta y}{y} = \frac{f'(x) \cdot \Delta x}{f(x)} = \frac{f'(x)x}{f(x)} \cdot \frac{\Delta x}{x}$$

2.12 Definition (Konditionszahlen). Die Zahl

$$K_{abs} = |f'(x)|$$

heißt *absolute Konditionszahl* des Problems $x \mapsto f(x)$. Für $f(x) \cdot x \neq 0$ heißt

$$K_{rel} = |\frac{f'(x) \cdot x}{f(x)}|$$

die entsprechende *relative Konditionszahl.* Ein Problem heißt schlecht konditioniert,falls eine dieser beiden Konditionszahlen deutlich größer als 1 sind. Ansonsten heißt das Problem gut konditioniert.

2.13 Beispiel. Konditionierung der Addition und Multiplikation

- Für die Addition $f(x) = x + a$ gilt

$$K_{rel} = |\frac{f'(x)x}{f(x)}| = |\frac{x}{x+1}|$$

$\implies K_{rel}$ ist groß, falls $|x + a| \ll |x|$.

- Für die Multiplikation $f(x) = x \cdot a$ gilt

$$K_{rel} = \frac{|f'(x)x|}{|f(x)|} = \frac{|ax|}{|ax|} = 1$$

$\implies$ Die absolute Kondition ist schlecht, falls $1 \ll q$. Die relative Kondition ist immer gut.

2.14 Definition. Erfüllt die Implementierung eines Algorithmus $\boxed{f}$ zur Lösung eines Problems $x \mapsto f(x)$ die Abschätzung

$$|\frac{\boxed{f} - f(x)}{f(x)} \leq C_V K_{rel} \varepsilon_{mach}$$

mit einem mäßig großen $C_V > 0$, so wird der Algorithmus *vorwärtsstabil* genannt. Ergibt die Rückwärtsanalyse $\boxed{f}(x) = f(x + \Delta x)$ mit

$$|\frac{\Delta x}{x} \leq C_R \varepsilon_{mach}$$

mit $C_R > 0$ nicht zu groß, so heißt der Algorithmus $\boxed{f}$ *rückwärtsstabil*

2.15 Satz. Jeder rückwärtsstabile Algorithmus ist auch vorwärtsstabil

Beweis. Übung. □

Oft ist Rückwärtsstabilität einfacher nachzuweisen.

Faustregel zu konditionierten Probleme

- Gut konditioniertes Problem + stabiler Algorithmus
  $\implies$ Gute numerische Resultate.

- Schlecht konditioniertes Problem oder instabiler Algorithmus
  $\implies$ Fragwürdige Ergebnisse.

2.16 Beispiel. Fortsetzung von Beispiel 2.11
Die Rückwärtsanalyse hat gezeigt: Falls $0 < |q| \ll 1 < p$ wird der numerische Fehler untragbar. Unsere Abbildung ist:

$$f(q) = p - \sqrt[2]{p^2 - 1}$$

Als Konditionszahlen ergeben sich:

$$K_{abs} = |f'(q)| = |\frac{1}{2\sqrt[2]{p^2-q}}| < 1$$

$$\begin{aligned}
K_{rel} &= |\frac{f'(q)q}{f(q)}| \\
&= |\frac{q}{2\sqrt[2]{p^2-q}(p-\sqrt[2]{p^2-q})(p+\sqrt[2]{p^2-q})}| \\
&= \frac{1}{2}|\frac{p+\sqrt[2]{p^2-q}}{\sqrt[2]{p^2-q}} \\
&\approx \frac{1}{2}|\frac{p+p}{p}| \approx 1
\end{aligned}$$

$\Longrightarrow$ Nullstellenberechung ist ein gut konditioniertes Problem, aber Algorithmus 2 muss instabil sein

3 Dreitermrekursion

3.1 Theoretische Grundlagen

3.1 **Definition.** Für gegebene p_0 und p_1 heißt eine Rekursion der Form

$$p_k = a_k p_{k-1} + b_k p_{k-2} + c_k,\ k = 2, 3, \dots \tag{3.4}$$

mit $b_k \neq 0$ eine *Dreitermrekursion.* Die zugehörige *Rückwärtsrekursion* ist

$$p_{k-2} = -\frac{a_k}{b_k} p_{k-1} + \frac{1}{b_k} p_k - \frac{c_k}{b_k},\ k = n, n-1, \dots \tag{3.5}$$

Ist $b_k = 1$, das heißt (3.4) und (3.5) gehen durch vertauschen von p_{k-2} und p_k auseinander hervor, so heißt die Rekursion symmetrisch. Gilt $c_k = 0$ für alle k, so heißt die Rekursion homogen.

3.2 **Beispiel.** Die Fibonacci-Zahlen sind rekursiv definiert durch:

$$p_k = p_{k-1} + p_{k-2}$$

mit $p_0 = 0, p_1 = 1$. Es gilt $a_k = b_k = 1$.
Sei

$$p_k(x) = 2\cos(x) p_{k-1}(x) - 1 \cdot p_{k-2}(x)$$

mit $p_0 = 1$ und $p_1 = \cos(x)$. Man kann zeigen, dass

$$p_k(x) = 2\cos(kx)$$

ist.
Die Chebychev-Polynome T_k erfüllen

$$T_k(x) = 2xT_{k-1} - T_{k-2}(x)$$

mit $T_0(x) = 1$ und $T_1(x) = x$

Daten: Koeffizienten von $\{a_k\}_{k=2}^n$, $\{b_k\}_{k=2}^n$,$\{c_k\}_{k=2}^n$, Startwerte p_0, p_1
Ergebnis: Werte von $\{p_k\}_{k=2}^n$
für $k \leftarrow 2$ **bis** n **tue**
| $p_k = a_k p_{k-1} + b_k p_{k-2} + c_k$
Ende

Algorithmus 4: Dreitermrekursion

3.3 **Beispiel.** Betrachte: $p_0 = 1mp_1 = \sqrt[2]{2} - 1, p_k = -2p_{k-1} + p_{k-2},\ k = 2, 3, \dots$
Algorithmus 4 liefert:
Die Oszillationen für höhere k sollten uns misstrauische machen und uns motivieren das Problem genauer anzuschauen.

Wir können homogene Dreitermrekursion schreiben als:

$$\begin{bmatrix} p_k \\ p_{k-1} \end{bmatrix} = \begin{bmatrix} a_k p_{k-1} + b_k p_{k-2} \\ p_{k-1} \end{bmatrix} =: \begin{bmatrix} a_k & b_k \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{k-1} \\ p_{k-2} \end{bmatrix} = A_k \begin{bmatrix} p_{k-1} \\ p_{k-2} \end{bmatrix}$$

Rekursiv folgt:

$$\begin{bmatrix} p_k \\ p_{k-1} \end{bmatrix} = A_k \begin{bmatrix} p_{k-1} \\ p_{k-2} \end{bmatrix} = A_k A_{k-1} \begin{bmatrix} p_{k-2} \\ p_{k-3} \end{bmatrix} = \ldots = A_k \ldots A_2 \begin{bmatrix} p_1 \\ p_0 \end{bmatrix}$$

Offensichtlich gilt für alle $\alpha, \beta \in \mathbb{R}$ und Startwerte p_0, p_1, q_0, q_1 und $B_k := A_k \ldots A_2$, dass

$$B_k \begin{bmatrix} \alpha p_1 + \beta q_1 \\ \alpha p_0 + \beta q_0 \end{bmatrix} = \alpha B_k \begin{bmatrix} p_1 \\ p_0 \end{bmatrix} + \beta B_k \begin{bmatrix} q_1 \\ q_0 \end{bmatrix} = \alpha \begin{bmatrix} p_k \\ p_{k-1} \end{bmatrix} + \beta \begin{bmatrix} q_k \\ q_{k-1} \end{bmatrix}$$

Das heißt, die Lösungsfolge $\{p_k\}$ hängt *linear* von den Startwerten $\begin{bmatrix} p_1 \\ p_0 \end{bmatrix} \in \mathbb{R}^2$ ab.