Vorlesung Informatik 1 (Wintersemester 2020/2021)

Kapitel 5: Codierung reeller Zahlen

Martin Frieb
Johannes Metzger

Universität Augsburg
Fakultät für Angewandte Informatik

25. November 2020

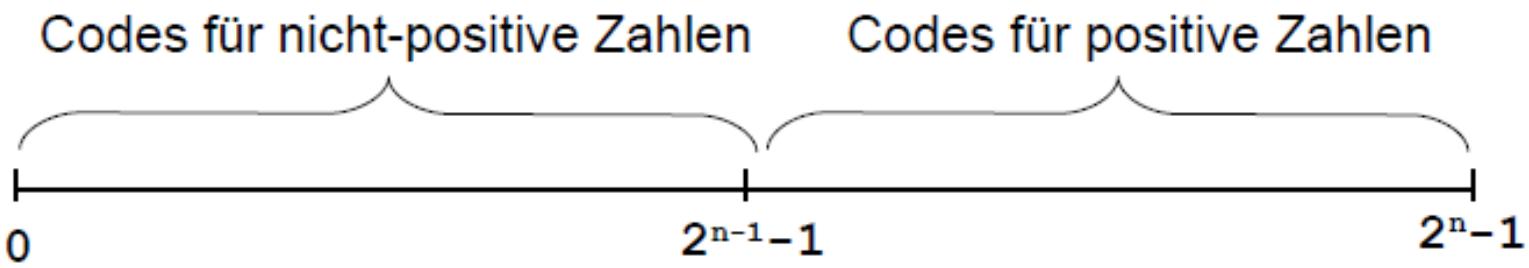
Was ist die Exzeß-q-Codierung?

Definition 5.1 (Exzeß-q-Codierung)

Die **Exzeß-q-Codierung** (**EX-q-Codierung**)
$c_{EX-q,n} : \{-q, ..., 0, 1, ..., 2^n - 1 - q\} \to \mathbb{B}^n$
zu einer Zahl $q \in \mathbb{N}_0$ ist definiert durch

$$c_{EX-q,n}(x) := c_{2,n}(x + q)$$

Die Exzeß-q-Codierung wird in der Praxis mit $q = 2^{n-1} - 1$ eingesetzt zur **Codierung des Exponenten einer normierten Gleitkommazahl**:



Decodierung und Beispiele

Decodierung

Es gilt (ohne Beweis, da einfache Rechnung)

$$(b_{n-1}...b_1 b_0)_{EX-q,n} = (b_{n-1}...b_1 b_0)_{2,n} - q = (\sum_{i=0}^{n-1} b_i \cdot 2^i) - q$$

Berechnungsschema für $c_{EX-127,8}$

q=127	128	64	32	16	8	4	2	1	2er-Potenzen / q
-1	0	0	1	0	0	1	1	1	Ziffern

Beispiel 5.2

$c_{EX-127,8}(7) = c_{2,8}(7+127) = c_{2,8}(134) = 10000110$
$(0001)_{EX-7,4} = (0001)_{2,4} - 7 = 1 - 7 = -6$

Addition / Subtraktion von EX-q-Darstellungen

Definition 5.3 (Addition)

$c_{EX-q,n}(x) \oplus_{EX-q,n} c_{EX-q,n}(y) := (c_{EX-q,n}(x) + c_{EX-q,n}(y)) - c_{2,n}(q)$

- $\oplus_{EX-q,n}$: Addition auf EX-q-Darstellungen in n Bit
- $+$: Addition auf Binärzahlen
- $-$: Subtraktion auf Binärzahlen

Satz 5.4

Es gilt für $-q \leq x, y < 2^n - 1 - q$ (Definitionsbereich) und $-q \leq x + y < 2^n - 1 - q$ (kein Bereichsüberlauf):

$$c_{EX-q,n}(x) \oplus_{EX-q,n} c_{EX-q,n}(y) = c_{EX-q,n}(x + y).$$

(ohne Beweis, da einfache Rechnung)

Beispiel 5.5

$c_{EX-127,8}(7) \oplus_{EX-127,8} c_{EX-127,8}(-7) = 10000110 + 01111000 - 01111111 = 11111110 - 01111111 = 01111111 = c_{EX-127,8}(0) = c_{EX-127,8}(7-7)$

Addition / Subtraktion von EX-q-Darstellungen

Definition 5.6 (Subtraktion)

$c_{EX-q,n}(x) \ominus_{EX-q,n} c_{EX-q,n}(y) := (c_{EX-q,n}(x) - c_{EX-q,n}(y)) + c_{2,n}(q)$

- $\ominus_{EX-q,n}$: Subtraktion auf EX-q-Darstellungen in n Bit
- $+$: Addition auf Binärzahlen
- $-$: Subtraktion auf Binärzahlen

Satz 5.7

Es gilt für $-q \leq x, y < 2^n - 1 - q$ (Definitionsbereich) und $-q \leq x - y < 2^n - 1 - q$ (kein Bereichsüberlauf):

$$c_{EX-q,n}(x) \ominus_{EX-q,n} c_{EX-q,n}(y) = c_{EX-q,n}(x - y).$$

(ohne Beweis, da einfache Rechnung)

Bei **Bereichsüberlauf** ist das Ergebnis von Addition und Subtraktion **undefiniert**

Was ist eine Festkommacodierung in n Bit?

Definition 5.8 (Festkommacodierung)

Die **Festkommacodierung** $c_{FK,k,n} : [0, 2^{n-k}[\to \mathbb{B}^n$ mit $n-k$ **Vorkommastellen** und k **Nachkommastellen** ist definiert durch

$$c_{FK,k,n}(x) := c_{2,n}(rd(x \cdot 2^k)),$$

wobei

$$rd : [0, \infty[\to \mathbb{N}_0,\ rd(y) := \begin{cases} \lfloor y \rfloor & , \text{falls } y - \lfloor y \rfloor < 0.5 \text{ (abrunden)} \\ \lceil y \rceil & , \text{falls } \lceil y \rceil - y \leq 0.5 \text{ (aufrunden)} \end{cases}$$

die **Rundung zur nächstgelegenen ganzen Zahl** ist.

Decodierung und Beispiel

Decodierung

Es gilt (ohne Beweis, da einfache Rechnung)

$$\begin{aligned}(b_{n-1}\dots b_1 b_0)_{FK,k,n} &= \frac{(b_{n-1}\dots b_1 b_0)_2}{2^k}\\ &= (b_{n-1}\dots b_k \,.\, b_{k-1}\dots b_0)_2\\ &= \sum_{i=0}^{n-1} b_i \cdot 2^{i-k}\end{aligned}$$

Beispiel 5.9 (Berechne $c_{FK,3,8}(1.2)$)

1. Multiplizieren mit 2^3: $1.2 \cdot 2^3 = 9.6$
2. Runden: $rd(9.6) = 10$
3. Binärcodierung mit 8 Bit: $c_{FK,3,8}(1.2) = c_{2,8}(10) = 00001010$
4. Decodierung: $(00001010)_{FK,3,8} = (1.010)_2 = 1.25$
5. (Absoluter) Rundungsfehler: $|1.2 - 1.25| = 0.05$

Exakt darstellbare Zahlen

Definition 5.10 (Exakt darstellbare Zahlen)

Eine reelle Zahl x heißt **exakt darstellbar bzgl. einer Codierung** c, falls

$$c^{-1}(c(x)) = x$$

(es tritt kein Rundungsfehler bei der Codierung auf)

Beispiel 5.11 (Berechne $c_{FK,3,8}(1.75)$)

1. Multiplizieren mit 2^3: $1.75 \cdot 2^3 = 14$
2. Runden: $rd(14) = 14$
3. Binärcodierung mit 8 Bit: $c_{FK,3,8}(1.75) = c_{2,8}(14) = 00001110$
4. Decodierung: $(00001110)_{FK,3,8} = (1.110)_2 = 1.75$
5. (Absoluter) Rundungsfehler: $|1.75 - 1.75| = 0$

Exakt darstellbare Zahlen

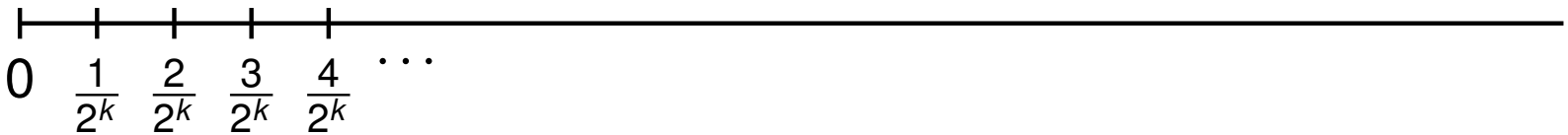
Satz 5.12

Eine reelle Zahl $x \in [0, 2^{n-k}[$ *ist genau dann exakt darstellbar bzgl.* $c_{FK,k,n}$*, falls sie in der Binärdarstellung* ***höchstens*** k ***Nachkommastellen*** *hat. Dies ist gleichbedeutend mit* $x \cdot 2^k \in \mathbb{N}_0$*.*

Beweis.

Nach der Definition wird ab der $(k+1)$-ten Nachkommastelle gerundet. □

Exakt darstellbare Zahlen auf der Zahlengeraden:



Rundungsfehler

Definition 5.13 (Absoluter und relativer Rundungsfehler)

Für eine reelle Zahl x und eine Codierung c heißt:

- $|x - c^{-1}(c(x))|$ **absoluter Rundungsfehler**.
- $\frac{|x-c^{-1}(c(x))|}{|x|}$ **relativer Rundungsfehler**.

Beispiel 5.14 ($c_{FK,3,8}(1.2)$)

1. $c_{FK,3,8}(1.2) = 00001010$
2. Decodierung: $(00001010)_{FK,3,8} = (1.010)_2 = 1.25$
3. Absoluter Rundungsfehler: $|1.2 - 1.25| = 0.05$
4. Relativer Rundungsfehler: $\frac{|1.2-1.25|}{1.2} = 0.041\overline{6}$

Rundungsfehler

Abstände zwischen exakt darstellbaren Zahlen sind gleichbleibend:

Satz 5.15 (Maximaler absoluter Rundungsfehler der Festkommacodierung)

Es gilt:

$$|x - (c_{FK,k,n}(x))_{FK,k,n}| \leq \frac{1}{2^{k+1}}.$$

Beweis.

$$\begin{aligned}
|x - (c_{FK,k,n}(x))_{FK,k,n}| &= |x - \frac{(c_{2,n}(rd(x \cdot 2^k)))_{2,n}}{2^k}| \\
&= |x - \frac{rd(x \cdot 2^k)}{2^k}| \\
&= \frac{|x \cdot 2^k - (rd(x \cdot 2^k))|}{2^k} \\
&\leq \frac{0.5}{2^k} = \frac{1}{2^{k+1}}
\end{aligned}$$

□

Rundungsfehler

Der relative Rundungsfehler kann für kleine x beliebig groß werden:

Satz 5.16 (Maximaler relativer Rundungsfehler der Festkommacodierung)

Es gilt:

$$\frac{|x - (c_{FK,k,n}(x))_{FK,k,n}|}{|x|} \leq \frac{1}{(|x| \cdot 2^{k+1})}.$$

Beispiel 5.17 ($c_{FK,3,8}(0.1)$)

1. $c_{FK,3,8}(0.1) = 00000001$
2. Decodierung: $(00000001)_{FK,3,8} = (0.001)_2 = 0.125$
3. Relativer Rundungsfehler: $\frac{|0.1-0.125|}{0.1} = 0.25$

Beispiel 5.18 ($c_{FK,3,8}(10.1)$)

1. $c_{FK,3,8}(10.1) = 001010001$
2. Decodierung: $(001010001)_{FK,3,8} = (1010.001)_2 = 10.125$
3. Relativer Rundungsfehler: $\frac{|10.1-10.125|}{10.1} = 0.00247..$

Bewertung

Verwendung

- Gleichbleibender Abstand $\frac{1}{2^k}$ zwischen exakt darstellbaren Zahlen: Geeignet für Zahlen ähnlicher Größenordnung (Anzahl der Nachkommastellen k an Größenordnung anpassen, so dass Rundungsfehler tolerierbar)
- Problematisch für Rechnungen mit Zahlen unterschiedlicher Größenordnung (Wertebereich muss an größere Zahl angepasst werden; führt zu ggf. zu wenig Nachkommastellen für die kleinere Zahl)

Arithmetik

Es wird die Arithmetik auf Binärzahlen verwendet:
$c_{FK,k,n}(x) \oplus_{FK,k,n} c_{FK,k,n}(y) := c_{FK,k,n}(x) + c_{FK,k,n}(y)$
$c_{FK,k,n}(x) \ominus_{FK,k,n} c_{FK,k,n}(y) := c_{FK,k,n}(x) - c_{FK,k,n}(y)$
Hierbei kommen weitere Rundungfehler hinzu:
$c_{FK,k,n}(x) \oplus_{FK,k,n} c_{FK,k,n}(y) = c_{2,n}(rd(x \cdot 2^k) + rd(y \cdot 2^k)) \approx$
$c_{2,n}(rd(x \cdot 2^k + y \cdot 2^k)) = c_{FK,k,n}(x + y)$

Was ist die Gleitkommacodierung?

Definition 5.19 (Gleitkommacodierung)

Die **Gleitkommacodierung** $c_{GK,k,n} :]-2^{2^{n-k-1}}, 2^{2^{n-k-1}}[\to \mathbb{B}^n$ einer reellen Zahl x mit **normierter Gleitkomma-Darstellung** $x = m \cdot 2^e$ ist definiert durch:

$$c_{GK,k,n}(x) := x_{n-1} \underbrace{x_{n-2} \quad \dots \quad x_{k-1}}_{\text{Charakteristik}} \underbrace{x_{k-2} \quad \dots \quad x_0}_{\text{Mantisse}},$$

wobei

- $x_{n-1} := 0$ für $m \geq 0$ und $x_{n-1} := 1$ für $m < 0$ (**Vorzeichenbit**).
- $x_{n-2} \dots x_{k-1} := c_{EX-(2^{n-k-1}-1),n-k}(e)$
  EX-q-Darstellung des **Exponenten** in $n-k$ Bit mit $q = 2^{n-k-1} - 1$.
- $x_{k-2} \dots x_0 := c_{FK,k-1,k-1}(|m| - 1)$
  Festkommacodierung der **Mantisse** mit $k-1$ Nachkommastellen und 0 Vorkommastellen (wegen der Normierung ist die Vorkommastelle immer 1)

Die **Definitionsbereichsgrenzen** werden später erklärt (siehe Satz 5.20)

Reservierte Bitmuster

Für bestimmte Werte und Rechenergebnisse gibt es reservierte Bitmuster:

Einige reservierte Bitmuster

- Spezielle Darstellung von 0:

$$0\underbrace{0\ \ldots\ 0}_{\text{Charakteristik}}\underbrace{0\ \ldots\ 0}_{\text{Mantisse}}$$

- Spezielle Darstellung von ∞ (z.B. als Ergebnis von $1/0$):

$$0\underbrace{1\ \ldots\ 1}_{\text{Charakteristik}}\underbrace{0\ \ldots\ 0}_{\text{Mantisse}}$$

- Spezielle Darstellung von $-\infty$ (z.B. als Ergebnis von $-1/0$):

$$1\underbrace{1\ \ldots\ 1}_{\text{Charakteristik}}\underbrace{0\ \ldots\ 0}_{\text{Mantisse}}$$

Reservierte Bitmuster

Für bestimmte Werte und Rechenergebnisse gibt es reservierte Bitmuster:

Weitere reservierte Bitmuster

- Darstellung von *NaN* (**Not a number**, z.B. als Ergebnis der Rechnung $0/0$ oder $\infty - \infty$):

$$x_{n-1}\underbrace{1 \quad \dots \quad 1}_{\text{Charakteristik}}\underbrace{x_{k-2} \quad \dots \quad x_0}_{\text{Mantisse}}$$

  ($x_{n-1}x_{k-2}\dots x_0$ ist ein plattformabhängiger Fehlercode)

- Darstellung von **denormalisierten Zahlen** x der Form $|x| = 0.x_{k-2}\dots x_0 \cdot 2^{min}$, wobei $min = -(2^{n-k-1} - 2)$ der kleinstmögliche Exponent ist:

$$x_{n-1}\underbrace{0 \quad \dots \quad 0}_{\text{Charakteristik}}\underbrace{x_{k-2} \quad \dots \quad x_0}_{\text{Mantisse}}$$

Zusammengefasst stehen die Bitmuster $0\dots0$ und $1\dots1$ **nicht** für die Darstellung des **Exponenten** zur Verfügung

Exakt darstellbare Zahlen

Satz 5.20

Eine reelle Zahl $x \in]-2^{2^{n-k-1}}, 2^{2^{n-k-1}}[$ *in der normierten Gleitkomma-Darstellung* $x = m \cdot 2^e$ *ist genau dann exakt darstellbar bzgl.* $c_{GK,k,n}$*, falls*

- *die Mantisse* m *in der Binärdarstellung* ***höchstens*** $k-1$ ***Nachkommastellen*** *hat*
- *und für den Exponenten* $-(2^{n-k-1}-2) \leq e \leq 2^{n-k-1}-1$ *gilt.*

Beweis.

- Mantisse: Nach der Definition wird ab der k-ten Nachkommastelle gerundet.
- Exponent: Wegen der reservierten Bitmuster hat der
  - größte Exponent das Bitmuster $1\ldots10$: das entspricht $e = 2^{n-k-1}-1$
  - kleinste Exponent das Bitmuster $0\ldots01$: das entspricht $e = -(2^{n-k-1}-2)$

  (siehe Definitionsbereich der EX-q-Darstellung)

□

Rundungsfehler

Abstände zwischen exakt darstellbaren Zahlen werden für große Exponenten sehr groß:

Satz 5.21 (Maximaler absoluter Rundungsfehler der Gleitkommacodierung)

Es gilt:

$$|m \cdot 2^e - (c_{GK,k,n}(m \cdot 2^e))_{GK,k,n}| \leq \frac{2^e}{2^k}.$$

Beweis.

$$\begin{aligned}
&|m \cdot 2^e - (c_{GK,k,n}(m \cdot 2^e))_{GK,k,n}| \\
&= |m - (c_{FK,k-1,n}(m))_{FK,k-1,n}| \cdot 2^e \\
&\leq 2^e/2^k
\end{aligned}$$

□

Rundungsfehler

Der relative Rundungsfehler ist unabhängig von der Größe von x:

Satz 5.22 (Maximaler relativer Rundungsfehler der Gleitkommacodierung)

Es gilt:

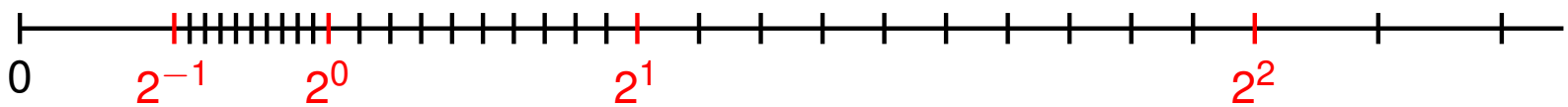
$$\frac{|m \cdot 2^e - (c_{GK,k,n}(m \cdot 2^e))_{GK,k,n}|}{|m \cdot 2^e|} \leq \frac{1}{2^k}.$$

Beweis.

$|m \cdot 2^e - (c_{GK,k,n}(m \cdot 2^e))_{GK,k,n}| / |m \cdot 2^e|$
$\leq 2^e / (2^k \cdot |m \cdot 2^e|)$
$\leq 1/2^k$

□

Exakt darstellbare Zahlen auf der Zahlengeraden:



Decodierung und Beispiel

Decodierung

Es gilt mit $q = 2^{n-k-1} - 1$ (ohne Beweis, da einfache Rechnung):

$$(x_{n-1} \dots x_1 x_0)_{GK,k,n} = \begin{cases} (1.x_{k-2} \dots x_0)_2 \cdot 2^{(x_{n-2} \dots x_{k-1})_{EX-q,n-k}} & \text{falls } x_{n-1} = 0 \\ -(1.x_{k-2} \dots x_0)_2 \cdot 2^{(x_{n-2} \dots x_{k-1})_{EX-q,n-k}} & \text{falls } x_{n-1} = 1 \end{cases}$$

Beispiel 5.23 (Berechne $c_{GK,5,8}(4.6)$)

1. Normierte Gleitkommadarstellung: $4.6 = 2.3 \cdot 2^1 = 1.15 \cdot 2^2$.
2. Vorzeichenbit: 0 (positive Mantisse)
3. q bestimmen: $q = 2^{(8-5)-1} - 1 = 3$
4. Charakteristik bestimmen: $c_{EX-3,3}(2) = c_{2,3}(2+3) = 101$
5. Code der Mantisse bestimmen: $c_{FK,4,4}(0.15) = c_{2,4}(rd(0.15 \cdot 2^4)) = 0010$
6. Decodierung: $(01010010)_{GK,5,8} = (1.001)_2 \cdot 2^2 = 100.1 = 4.5$
7. Absoluter Rundungsfehler: $|4.6 - 4.5| = 0.1$

Weitere Beispiele

Beispiel 5.24 (Kleinste positive darstellbare normalisierte Zahl bzgl. $c_{GK,5,8}$)

- Kleinster Exponent (Bitmuster 0…0 schon vergeben):
  $(001)_{EX-3,3} = (001)_{2,3} - 3 = 1 - 3 = -2.$
- Nachkommastellen der kleinsten Mantisse:
  $(0000)_{FK,4,4} = (0.0000)_2$
- *Ergebnis*:
  $(00010000)_{GK,5,8} = (1.0000)_2 \cdot 2^{-2} = (0.01)_2 = 0.25$

Beispiel 5.25 (Größte positive darstellbare normalisierte Zahl bzgl. $c_{GK,5,8}$)

- Größter Exponent (Bitmuster 1…1 schon vergeben):
  $(110)_{EX-3,3} = (110)_{2,3} - 3 = 6 - 3 = 3.$
- Nachkommastellen der größten Mantisse:
  $(1111)_{FK,4,4} = (0.1111)_2$
- *Ergebnis*:
  $(01101111)_{GK,5,8} = (1.1111)_2 \cdot 2^3 = (1111.1)_2 = 15.5$

IEEE-Standard 754

IEEE: Institute of Electrical and Electronics Engineers

Einfache Genauigkeit (32 Bit, $k = 24$, Datentyp `float` in C)

- *8-Bit Charakteristik (q = 127)*:
  Exponent zwischen -126 (Bitmuster 0...01) und 127 (Bitmuster 1...10)
- *23-Bit Mantisse*:
  Werte zwischen 1 (Bitmuster 0...0) und $2 - 2^{-23}$ (Bitmuster 1...1)
- Maximaler absoluter Fehler: $2^{127}/2^{24}$
- Maximaler relativer Fehler: $1/2^{24}$

IEEE-Standard 754

Doppelte Genauigkeit (64 Bit, Datentyp `double` in C)

- *11-Bit Charakteristik (q = 1023)*:
  Exponent zwischen -1022 (Bitmuster 0...01) und 1023 (Bitmuster 1...10)
- *52-Bit Mantisse*:
  Werte zwischen 1 (Bitmuster 0...0) und $2 - 2^{-52}$ (Bitmuster 1...1)

Erweiterte Genauigkeit (80 Bit, Datentyp `long double` in C)

- *15-Bit Charakteristik (q = 16383)*:
  Exponent zwischen $-(2^{-14} - 2)$ (Bitmuster 0...01) und $(2^{14} - 1)$ (Bitmuster 1...10)
- *64 -Bit Mantisse*:
  Werte zwischen 1 (Bitmuster 0...0) und $2 - 2^{-64}$ (Bitmuster 1...1)

Arithmetik

Addition und Subtraktion zweier normierter Gleitkommazahlen $m_1 \cdot 2^{e_1}$ und $m_2 \cdot 2^{e_2}$ erfolgen auf den Bitmustern für Mantisse und Exponent:

- Vergleich der beiden Zahlen und ggf. Exponentenangleich der kleineren Zahl mit Rundung (der Exponentenangleich entspricht einer Verschiebung der Stellen nach rechts - das kann zu einer nicht exakt darstellbaren Zahl führen)
- Addition / Subtraktion der Mantissen und ggf. Normierung mit Rundung (die Normierung entspricht einer Verschiebung der Stellen - das kann zu einer nicht exakt darstellbaren Zahl führen)

Problem: Addition sehr unterschiedlich großer Zahlen und Subtraktion fast gleich großer Zahlen kann zu Stellenauslöschungen durch Rundung führen (siehe nächste Folie)

Arithmetik

Rundung und Stellenauslöschung durch Exponentenangleich

Betrachte $m_1 \cdot 2^{e_1} + m_2 \cdot 2^{e_2}$ mit $e_2 < e_1$ in einer $c_{GK,k,n}$-Codierung. Durch den Exponentenangleich werden die Stellen der kleineren Zahl $m_2 \cdot 2^{e_2}$ um $e_2 - e_1$ Stellen rechts verschoben:

$$m_1 \cdot 2^{e_1} + m_2 \cdot 2^{e_2} = (m_1 + m_2 \cdot 2^{e_2 - e_1}) \cdot 2^{e_1}$$

Bzgl. der k-ten Nachkommastelle von $m_2 \cdot 2^{e_2 - e_1}$ wird gerundet, die Nachkommastellen ab der $k+1$-ten werden gelöscht (da diese nicht mehr darstellbar sind). Im Falle $e_2 - e_1 < -k$ gilt also sogar:

$$(m_1 + m_2 \cdot 2^{e_2 - e_1}) \cdot 2^{e_1} = m_1 \cdot 2^{e_1}$$

Ein ähnlicher Effekt tritt bei Subtraktion fast gleich großer Zahlen auf.

Arithmetik

In der Gleitkommarechnung hängt das Rechenergebnis wegen unterschiedlicher Rundungsfehler von der Auswertungsreihenfolge ab!

Beispiel 5.26 (Rechenergebnis hängt von der Auswertungsreihenfolge ab)

Betrachte folgende Rechnungen in einfacher Genauigkeit, d.h. in der $c_{GK,24,32}$-Codierung.

- $(1.0 \cdot 2^{-9} + 1.0 \cdot 2^{23}) - 2^{23} = (2^{-32} + 1.0) \cdot 2^{23} - 2^{23} = 0$
- $2^{-9} + (1.0 \cdot 2^{23} - 1.0 \cdot 2^{23}) = 2^{-9} + (1.0 - 1.0) \cdot 2^{23} = 2^{-9}$

Nachkommastellen ab der 24-ten Stelle werden abgeschnitten bzw. nicht berücksichtigt!

- $2^{23} \cdot ((2^{-9} + 2^{23}) - 2^{23}) = 0$
- $2^{23} \cdot (2^{-9} + (2^{23} - 2^{23})) = 2^{14}$

Arithmetik

Sonderrolle der 0

- Hat keine Gleitkomma-Darstellung
- Arithmetische Sonderbehandlung (siehe IEEE-Standard)
- Exaktes Ergebnis 0 wird in der Regel wegen Rundungsfehlern nicht angenommen

Verwendung der 0 in Abfragen / Bedingungen

- **`double`-Variablen nicht auf `0` testen**
  Verwende `-r < x && x < r` statt `x == 0` für eine kleine positive Fehlerschranke `r`
- **`double`-Variablen nicht auf Gleichheit testen**
  Verwende `-r < x-y && x-y < r` statt `x == y` für eine kleine positive Fehlerschranke `r`

Arithmetik

Fazit

- Ergebnisse von Gleitkommaberechnungen können u.U. erheblich von dem exakten Wert abweichen
- Übliche Rechengesetze gelten im Allgemeinen nicht (Assoziativ-/Distributiv-/Kommutativgesetz)

Auswege

- Exakte Arithmetik (z.B. Intervallarithmetik: nicht durch Hardware realisiert, aber auch Software dafür erhältlich)
- Gruppierung von Zahlen nach Größenbereichen

Allgemeines zu Speicherbedarf und Wertebereich

- Der Speicherbedarf und die Grenzen des Wertebereichs eines Dezimalzahl-Datentyps können über Bibliotheks-Konstanten in `float.h` abgefragt werden
- Wenn in einer Berechnung oder durch eine Benutzereingabe der Wertebereich verlassen wird, kommt es zu Programm- oder Rechenfehlern
- Der Speicherbedarf einer Variable `x` kann mit `sizeof(x)` abgefragt werden
- Der Speicherbedarf eines Datentyps `T` kann mit `sizeof(T)` abgefragt werden
- Der Compiler ordnet jeder Variablen in einem Programm einen festen Speicherbereich zu, und zwar durch Festlegung der **Adresse der ersten Speicherzelle** dieses Bereichs.

Lesestoff zur Vor- und Nachbereitung

[Hellmann] Rechnerarchitektur
Kapitel 14.1 + 14.2