Rechnen mit beliebiger Genauigkeit trotz Gleitkommaarithmetik

Lukas Pöckl, Catherina Bauer, Dragana Jovanovic, Julia Gell

Überblick

1. Gleitkommazahlen - Lukas
    1.1. Definition / Schreibweise
    1.2. Normalisierung
    1.3. IEEE 754 Format
    1.4. Spezialfälle
    1.5. Beispiele zur Berechnung
    1.6. Vor- und Nachteile der Gleitkommadarstellung
2. Genauigkeiten - Catherina
    2.1. Fixe Genauigkeiten
    2.2. Beliebige Genauigkeit
    2.3. Rundungsfehler
3. Gleitkommaarithmetik - Dragana, Julia
    3.1. Geschichte
    3.2. Eigenschaften
    3.3. Operationen
    3.4. Ausnahmen
4. Bibliotheken- Julia

1. Gleitkommazahlen

1.1 Definition / Schreibweise

- Auch Fließkommazahl genannt
- Angenäherte Darstellung einer reellen Zahl
- Endliche Teilmenge der rationalen Zahlen

In Exponentialschreibweise darstellbar als: $\pm z = \pm 1 * m * b^e$

z := Zahl, die umgeformt werden soll
m := Mantisse
b := Basis $b \in \mathbb{N}$
e := Exponent $e \in \mathbb{Z}$

1. Gleitkommazahlen

1.2 Normalisierung

Beispiel: Die 452.956.000.000 in Gleitkommadarstellung:

±z	**= ±1*m*b^{e}**
452.956.000.000	= 1 * 452956 * 10^{6}
	= 1 * 452,956 * 10^{9}
	= 1 * 4,52956 * 10^{11}

Normalisierung: Mantissen Wert: $1 \leq m < 10$

1. Gleitkommazahlen

1.3 IEEE 754 Format

- 1 Bit für Vorzeichen: 0(+), 1(-)
- 8 Bit für Exponent
  Bias sorgt für Speicherung eines positiven Exponenten
- 23 Bit für Mantisse, normalisiert
- Ergibt 32Bit = einfache Genauigkeit

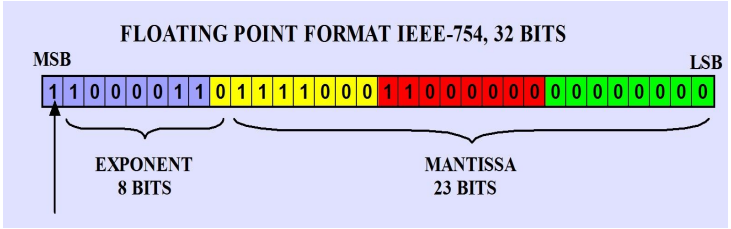


Abb.1: IEEE 754 Standard, 32 Bits

1. Gleitkommazahlen

1.4. Spezialfälle

Spezialfälle im IEEE 754 Format für:

- **0** bzw. **-0** := Sowohl e als auch m nur 0er -> 0(1) 00000000 00000000000000000000000
- **-∞** bzw. **+∞** := e nur 1er m nur 0er -> 0(1) 11111111 00000000000000000000000
- **NaN**: e nur 1er m #1er > 0 -> 0(1) 11111111 10000000000000000000000
- **Denormalisierte Zahl:** e nur 0er m#1er > 0 -> 0(1) 00000000 10000000000000000000000

Denormalisiert heißt: Mantissenwert wird mit 0. statt mit 1. gelesen

1. Gleitkommazahlen

1.5 Beispiel von IEEE Format in Dezimalzahl

Zahl im IEEE 754 Format:
1 10000111 11100000000000000000000

- Vorzeichen Bit = 1 -> negativ
- Exponent $10000111_2 = 135_{10} - 127_{10} = 8_{10}$
- Mantisse $11100000000000000000000_2 = 1 + 1* ½ + 1* ¼ + 1* ⅛ = 1{,}875_{10}$
- $-1 * 1{,}875 * 2^8 = -480.0_{10}$

1. Gleitkommazahlen

1.5 Beispiel von Dezimalzahl in IEEE Format

IEEE 754 Darstellung für die Zahl $18{,}4_{10}$:

$18_{10} = 10010_2$ $0{,}4_{10} = 0{,}11001..._2$ $18{,}4_{10} = 10010{,}011001..._2$

- Exponent der größten Zweierpotenz + Bias gerechnet: also $4 + 127 = 131_{10}$
  $131_{10} = 10000011_2$
- Anschließend erfolgt eine Kommaverschiebung und Normalisierung für Mantisse:
  Aus $10010{,}011001..._2$ wird $1{,}0010011001..._2$

Es ergibt sich somit:
Vorzeichen = 0 Exponent = 10000011 Mantisse = 00100110011001100110011 also:
0 10000011 00100110011001100110011

1. Gleitkommazahlen

1.6 Vor- und Nachteile der Gleitkommadarstellung bzw. IEEE Format

Vorteile:

- Hoher Dynamikbereich
- Spezialfälle sind definiert, Vorteil bei Rechenoperationen

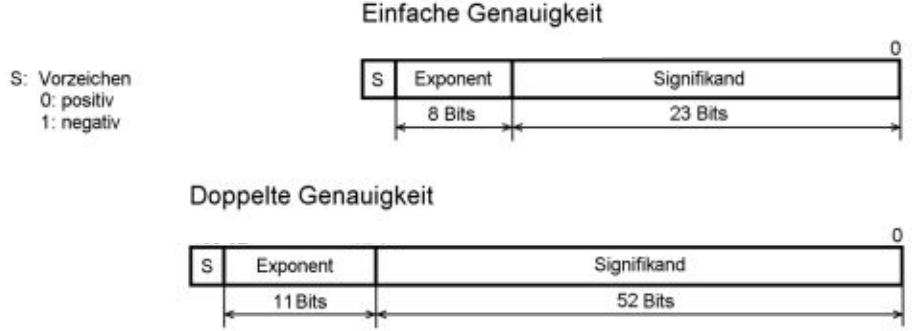
Nachteile:

- Rundungsfehler
- Unterläufe von sehr kleinen Zahlen auf 0
- Prüfen auf Gleichheit
- Bei Subtraktion von fast gleichen Zahlen, falsches Ergebnis

2. Genauigkeiten

2.1 Fixe Genauigkeiten

Abb.2: Genauigkeiten



- Genauigkeiten sind nicht nur den Gleitkommazahlen vorbehalten
- Einfache und doppelte Genauigkeit in fast allen Programmiersprachen integriert.
- Einfache Genauigkeit: Als Dezimalzahl gerundet 7 Nachkommastellen
- Doppelte Genauigkeit: Als Dezimalzahl gerundet 16 Nachkommastellen

2. Genauigkeiten

2.2 Beliebige Genauigkeit

Repräsentiert eine Gleitkommazahl mit beliebiger Genauigkeit.

- Allein limitiert durch den Speicher des Rechners
- Basis erhöhen, um größeren Wertebereich zu erlangen
- Sehr große Zahlen werden in kleine Einheiten geteilt, und mit diesen weitergerechnet.

Beispiel: 6.854.112 =

$13*256^2 + 42*256^1 + 209*256^0$ (Aufteilung in drei 8-Bit Einheiten)

2. Genauigkeiten

2.3 Rundungsfehler

Rundungsmöglichkeiten:

Beispiel	**1,40**	**1,60**	**1,50**	**2,50**	**-1,50**
Round-toward-zero	1	1	1	2	-1
Round-down(-∞)	1	1	1	2	-2
Round-up(+∞)	2	2	2	3	-1
Round-to-nearest	1	2	??	??	??
Round-to-even	1	2	2	2	-2

Rundungsfehler treten auf dann,

- wenn nicht alle Nachkommastellen berücksichtigt werden.
- wenn durch eine Rechnung mehr Stellen entstehen als darstellbar sind.

Kann durch Rechnen mit beliebiger Genauigkeit vorgebeugt werden.

2. Genauigkeiten

2.3 Rundungsfehler

Beispiel und Auswirkungen:

- Die Geschichte von Edward Lorenz um 1960
- Rundung von x,506127 auf x,506
  (dabei stellt x eine beliebige Vorkommazahl dar)
- Der Begriff „Schmetterlingseffekt“

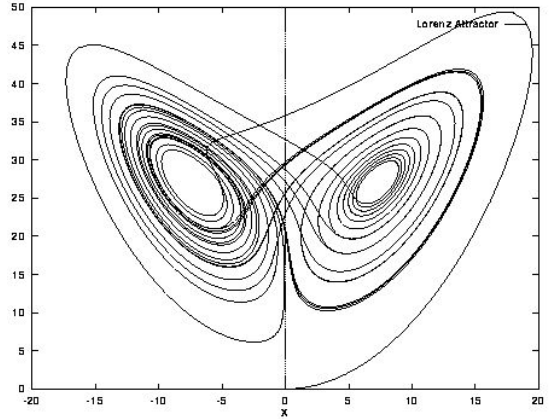


Abb.3: Schmetterlingseffekt

Kleine Abweichungen in der Berechnung führen zu drastischen Veränderungen im Endergebnis.

3. Gleitkommaarithmetik

3.1 Geschichte

- Erste dokumentierte Verwendung - 2700 Jahre
- Der Computer kann mehr!
- Eckdaten zu Konrad Zuse
- 1941 → Z3
- 1942 - 1945 → Z4

Abb.4: Konrad Zuse

3. Gleitkommaarithmetik

3.2 Eigenschaften

- Auslöschung
- Absorption
- Ungültigkeit der Assoziativ- und Distributivgesetze
- Unterlauf/Überlauf
- Usw.

3. Gleitkommaarithmetik

3.2.1 kurze Beispiele: Auslöschung und Absorption in Basis 10

Auslöschung:

Beispiel:

$1{,}2345 * 10^5 \rightarrow 1{,}235 * 10^5$

$\underline{+\ 1{,}2340 * 10^5} \rightarrow \underline{+\ 1{,}234 * 10^5}$

$0{,}0005 * 10^5 \qquad 0{,}001 * 10^5$

Absorption:

Beispiel:

$1{,}0 * 10^5 \rightarrow 1{,}000 * 10^5$

$1{,}5 * 10^2 \rightarrow \underline{+\ 0{,}001 * 10^5}$

$\rightarrow 0{,}0015 * 10^5 \qquad 1{,}001 * 10^5$

3. Gleitkommaarithmetik

3.2.3 Ungültigkeit der Assoziativ- und Distributivgesetze

Assoziativgesetz:

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

Distributivgesetz:

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

Diese beiden Gesetze gelten am Computer nicht mehr!

3. Gleitkommaarithmetik

3.2.2 Unterlauf/Überlauf

Beispiel: 8 Bit Computer, speichert unsigned integers im Wertebereich 0 - 255

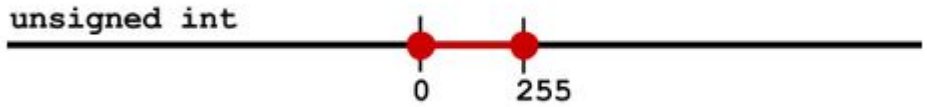


Abb.5: Zahlenstrahl

3. Gleitkommaarithmetik

3.2.2 Unterlauf/Überlauf

Beispiel nach wahrer Begebenheit:

Horizontale Geschwindigkeit der **Rakete** 32.768 [interne Einheit]
(64-bit, Gleitkommazahl)

⇓

Wertebereich einer **16-Bit Zahl** -32.768 bis +32.767
(16-bit, Festkommazahl)

Ein Überlauf brachte die Rakete jedoch zum Absturz.

Abb.6: Ariane 5

3. Gleitkommaarithmetik

3.3 Operationen

- Arithmetische Operationen
- Konvertierungen
- Veränderung des Vorzeichens
- Vergleichsoperationen
- Rundung
- Diverse Umwandlungen

3. Gleitkommaarithmetik

3.3.1 Addition

$$\mathbf{m_1 * b^{e1} + m_2 * b^{e2} = (m_1 + m_2) * b^{e}}$$

$1{,}001 * 2^{10} + 1{,}101 * 2^{11} = ?$

1. $0{,}100 * 2^{11} + 1{,}101 * 2^{11}$
2. $0{,}100 + 1{,}101 = 10{,}001$

   Somit: $10{,}001 * 2^{11}$
3. Normalisieren: $1{,}0001 * 2^{12}$
4. $-126 \leq 12 \leq 127 \rightarrow$ kein Über-/Unterlauf
5. Runden; $1{,}000 + 0{,}001 = 1{,}001$
6. Endergebnis: $1{,}001 * 2^{12}$

3. Gleitkommaarithmetik

3.3.1 Multiplikation

$$\mathbf{m_1 * b^{e1} * m_2 * b^{e2} = m_1 * m_2 * b^{e1+e2}}$$

$(1{,}101 * 2^{-1}) * (-1{,}100 * 2^{-2}) = ?$

1. $-1 + -2 = -3$
2. $1{,}101 * 1{,}100 = 10{,}011100$

   Somit: $10{,}011100 * 2^{-3}$
3. Normalisieren: $1{,}0011100 * 2^{-2}$
4. $-126 \leq -2 \leq 127 \rightarrow$ kein Über-/Unterlauf
5. Runden: $1{,}010 * 2^{-2}$
6. Endergebnis: $-1{,}010 * 2^{-2}$

3. Gleitkommaarithmetik

3.4 Ausnahmen

5 Ausnahmen nach IEEE754:

- Ungültige Operation
- Division durch 0
- Überlauf
- Unterlauf
- Ungenauigkeit

2 weitere Ausnahmen:

- Clamped
- Gerundet

4. Bibliotheken

Libraries für beliebige Genauigkeit bei Gleitkommazahlen:

Java: java.math.BigDecimal

Python: mpmath

C++: Boost

```
Input : double a=0.03;
        double b=0.04;
        double c=b-a;
        System.out.println(c);
Output :0.009999999999999998

Input : BigDecimal _a = new BigDecimal("0.03");
        BigDecimal _b = new BigDecimal("0.04");
        BigDecimal _c = _b.subtract(_a);
        System.out.println(_c);
Output :0.01
```

Abb.7: Code-Snippet Java

Quellenverzeichnis

Abbildung 1: https://www.puntoflotante.net/FLOATING-POINT-FORMAT-IEEE-754.htm

Abbildung 2: http://www.wikiqm.de/lib/exe/fetch.php?cache=&media=computertechnik:datenstruktur.png

Abbildung 3: http://www.qweep.net/~rocko/sufficiency/node10.html

Abbildung 4: https://de.wikipedia.org/wiki/Konrad_Zuse

Abbildung 5: https://medium.com/@jollyfish/integer-overflow-underflow-and-floating-point-imprecision-6ba869a99033

Abbildung 6: https://deacademic.com/dic.nsf/dewiki/18564

Abbildung 7: https://www.geeksforgeeks.org/bigdecimal-class-java/

Quellenverzeichnis

https://en.wikipedia.org/wiki/IEEE_754

http://www.informatik.uni-bremen.de/agra/doc/lehrmat/wise0304/ra/kap2.1b.pdf

https://docplayer.org/44751711-Algorithmen-programmierung-reelle-zahlen-in-c-2-rechnen-mit-gleitkommazahlen.html

http://openbook.rheinwerk-verlag.de/c_von_a_bis_z/005_c_basisdatentypen_008.htm

http://docwiki.embarcadero.com/RADStudio/Sydney/de/Gleitkommaarithmetik

https://deacademic.com/dic.nsf/dewiki/527576#Eigenschaften_einer_Gleitkommaarithmetik

https://www.extremeoptimization.com/Documentation/Mathematics/Arbitrary-Precision-Arithmetic/Arbitrary-Precision-Floating-Point-Numbers.aspx

Vielen Dank
für die Aufmerksamkeit