

# Numerische Programmierung

## Konkrete Mathematik

Hinweise  
Für Algorithmen immer Animationen  
(Gauss-El, Givens-QR, Div.diff)  
Bilder für image (CT, Registrierung,  
Kalibrierung, Drehen)

### Literatur

Numerik für Informatiker (Huckle/Schneider)  
= Numerische Methoden

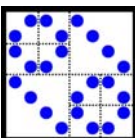
Folien voriger Semester

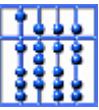
Herzberger: Wissenschaftliches Rechnen

Opfer: Numerik für Anfänger

Überhuber: Computer-Numerik

Kahaner/Moler/Nash: Numerical Methods and Software





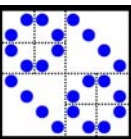
# I. Warum Numerik?

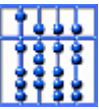
1. Zu Computer Science gehört auch die Numerik, genauso wie Datenbanken, Software Engineering.

*Ursprung der Informatik ist die Numerik!*

*Alles was mit Computer zu tun hat, gehört zu CS!*

- z.B. - Rechner-Arithmetik, Parallelrechner, usw.
- Computergraphik (Flächen, Kurven)
  - Bildverarbeitung (Komprimierung, Filtern, Analyse) : JPEG
  - Soundverarbeitung: MP3
  - Information Retrieval, Prozessverwaltung





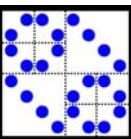
## 2. Wissenschaftliches Rechnen

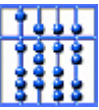
Verbund von Naturwissenschaft, Mathematik, Numerik, Informatik zur Lösung wiss. Probleme:

z.B. Wettervorhersage, Raketensteuerung, Strömungs-Simulation (NASA, BMW, Siemens,...)

## 3. Numerik zur Lösung von Informatik-Problemen

z.B. - Warteschlangen, Betriebsmittelzuteilung, und Stochastische Automaten,  
- Neuronale Netze und Fuzzy-Logik,  
- Räuber-Beute-Modelle zur Ressourcenverteilung





## Voraussetzungen aus der Informatik:

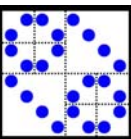
- Zahldarstellung
- Programmiersprache (JAVA)
- (Komplexitätstheorie)

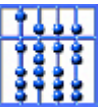
## Zur Mathematik:

Numerik als Fortführung der ‚Reinen Mathematik‘  
mit anderen Mitteln:

z.B. bestimme -  $\int_a^b e^{-t^2} dt$  ,

- Gleichungssysteme, Nullstellen
- Grenzwerte, Eigenwerte (Resonanzen)
- Interpolation

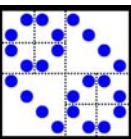


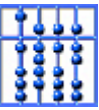


kontinuierlich versus diskret!

**Voraussetzungen aus der Mathematik:**

Taylor-Entwicklung und Mittelwertsatz, Ableitungen,  
Summen und Reihen, trigonometrische Funktionen,  
Lineare Gleichungssysteme, Vektoren und Matrizen,  
Normen, Komplexe Zahlen.





## II. Rechnerarithmetik und Rundungsfehler

Motto:

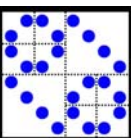
„Die natürlichen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk.“

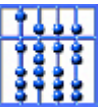
L. Kronecker

“Nobody is perfect”

Osgood zu Daphne (Jerry) in ‘Some like it hot’

*Beispiel: Digitalisierung von Audio – Audacity, JPEG*





## Problem: Endlichkeit!

Die Menge  $\mathbf{R}$  der reellen Zahlen besteht aus unendlich vielen Zahlen mit unendlich vielen Stellen!

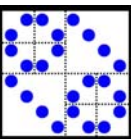
Jeder Computer ist endlich!

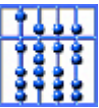
### 2.1. Definition:

Die endliche Menge  $\mathbf{M}$  der in einem Rechner darstellbaren Zahlen heißt

**Maschinenzahlen**

Wir unterscheiden ganzzahlige Maschinenzahlen und Maschinenzahlen für reelle Zahlen (=Gleitpunktzahlen).





1. Problem: Abbildung  $\mathbf{R} \rightarrow \mathbf{M}$  liefert Fehler

2. Problem: Arithmetische Operationen  $+$   $-$   $*$   $/$  :  
 $\mathbf{M} \times \mathbf{M} \rightarrow \mathbf{M}$  gilt nicht!

Zum ersten Problem:

2.2. Definition:

Eine Abbildung  $rd: \mathbf{R} \rightarrow \mathbf{M}$  bezeichnen wir als Rundung, wenn gilt:

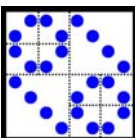
$$|x - rd(x)| = \min_{m \in M} |x - m|$$

Dies ist keine eindeutige Definition von  $rd$  !

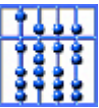
Andere Möglichkeiten statt  $rd$ : ceil, floor, to zero

Minimalforderung:  $rd(m) = m$  für  $m \in M$

Rundungsfehler (absoluter):  $f_{rd}(x) := x - rd(x)$







Zum zweiten Problem (arithmetische Operationen) :

$$+_M : M \times M \mapsto M$$

$$x +_M y \mapsto rd(x + y)$$

Genauso definieren wir für  
die Näherungsoperatoren

$$- , * , /$$

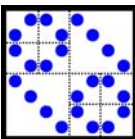
$$-_M , *_M , /_M$$

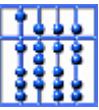
Beispiele von Maschinenzahlen:

Quantisierung von Musik- oder Bildwerten: 8 Bit, 16 Bit

Geldbeträge, Wechsel/Aktienkurse: DAX 3577,72

**Alternativen zu Maschinenzahlen und Maschinenarithmetik?**





- Rechnen mit beliebig vielen Stellen  
(symbolisches Rechnen, MATHEMATICA, MAPLE)
- Rechnen mit Intervallen (Intervall-Arithmetik)

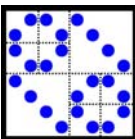
Folien-Beispiel:  $0.1114 + 0.001116$

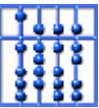
Modell: Jede Zahl repräsentiert durch drei Dezimalstellen und  
Position des Dezimalpunkts:

$$0.\boxed{111}4 \rightarrow 0.111 \quad \text{und} \quad 0.00\boxed{111}6 \rightarrow 0.00112$$

$$\text{Daher } 0.111 + 0.00112 = 0.11212 \rightarrow 0.112$$

Also Resultat:  $0.112$





Exaktes Ergebnis bei voller Stellenzahl:

$$0.1114 + 0.001116 = 0.112516$$

*Volle Stellenzahl* führt zu aufwendigen Rechnungen,  
Zahlen würden zu lang! Speicherplatz! Rechenzeit!

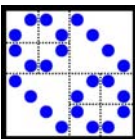
*Intervallarithmetik* auf 3 Stellen (Folien-Beispiel):  
Repräsentiere Zahlen durch Einschließungsintervalle:

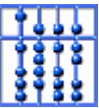
$$0.1114 \in [0.111, 0.112]$$

$$0.001116 \in [0.00111, 0.00112]$$

Untere Intervallgrenze der Lösung:

$$\lfloor 0.111 + 0.00111 \rfloor = \lfloor 0.11211 \rfloor = 0.112$$





Obere Grenze:

$$\lceil 0.112 + 0.00112 \rceil = \lceil 0.11312 \rceil = 0.114$$

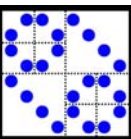
Resultat: Lösung von  $0.1114 + 0.001116$  liegt  
im Intervall  $[0.112, 0.114]$

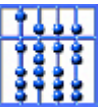
Vorteil: Exakte Information über Lage und Qualität der Lösung  
Nachteil: Rechenzeit! Ev. Ergebnisintervall sehr groß!

### 2.3. Definition: Festkommazahlen als Maschinenzahlen

$n_1 n_2 \cdots n_k, m_1 \cdots m_j$  mit

$$n_1, n_2, \dots, n_k, m_1, \dots, m_j \in \{0, 1, \dots, 9\}, \quad k, j \in \mathbb{N}$$





## Beispiele:

Geldbeträge wie Euro.Cent ,  $j=2$

Wechselkursangaben, wie 1.1591 (\$ zu €),  $j=4$ ,  $k=1$

Börsenindizes: DAX 3577.72,  $j=2$ ,  $k=4$ .

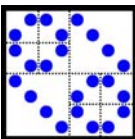
Alte Taschenrechner verwendeten Festpunktzahlen.

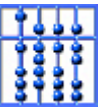
Für praktisches Rechnen ungenügend!

Zu wenige ganze, bzw. reelle Zahlen darstellbar!

Sehr große Zahlen? Sehr kleine Zahlen? Zahlen mit vielen Nachkommastellen?

Zusätzlich große Rundungsfehler.





Börsenindex Vancouver 1983.

Start des Indexes mit Wert 1000.

Bei jedem Verkaufereignis (ca. 3000 pro Tag) wurde der Index neu berechnet auf drei Stellen nach dem Komma:

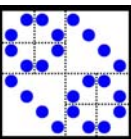
Rechne mit vier Stellen nach dem Komma und dann Abschneiden der vierten Stelle.

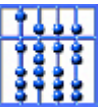
(Quasi rechnen wie mit ganzen Zahlen in C)

Nach 22 Monaten wurde 574.081 angegeben.

Der ‚wahre‘ Wert: 1098.892

Systematischer Fehler, der sehr oft auftritt!





Landtagswahl 1992 in Schleswig-Holstein:

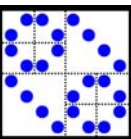
Grüne erhielten 4.97%.

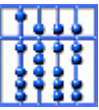
Zur Darstellung der Ergebnisse Festkomma mit einer Stelle nach dem Komma, ( $j=1$ ); also Anzeige des Ergebnisses als 5.0%.

Fehler wurde erst entdeckt, nachdem offizielle Ergebnisse bereits veröffentlicht waren.

**Rechnen mit Excel!!!**

Diese Zahlendarstellungen werden wir i.F. nicht verwenden!





## 2.4. Integer(Maschinen)zahlen

Endlicher Ausschnitt aus den ganzen Zahlen  $\mathbf{Z}$ ,  
symmetrisch um Null angeordnet.

Man verschiebt den Ausschnitt so, dass alle Zahlen positiv werden:

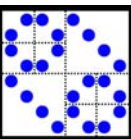
### Menge der Integer-Maschinenzahlen $\mathbf{M}$

$$z = \sum_{i=0}^{t-1} m_i \times 2^i - 2^{t-1}, \quad m_i \in \{0,1\} \quad i = 0, \dots, t-1$$

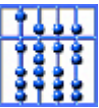
$t \in \mathbf{IN}$  gibt die **Stellenzahl** an, bzw. **Bits**.

**Stufenzahlen:**  $2^i$  mit  $0 \leq i \leq t-1$

Also  $\mathbf{M} = \langle -2^{t-1}, 2^{t-1} - 1 \rangle$







Bei 32 Bits (=4 Bytes) ergibt sich der Zahlenbereich der ganzzahligen Maschinenzahlen daher zwischen

$$-2^{31} = -2147483648$$

und

$$2^{31} - 1 = 2147483647$$

### Folien-Beispiel:

#### Darstellung der Dezimalzahl 11 mit $t = 5$

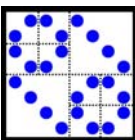
$$11 = 27 - 16 \quad (\text{da } 11+16=27)$$

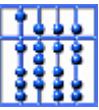
$$= (16 + 11) - 16 = 16 + (8+3) - 2^4$$

$$= 16 + 8 + (2+1) - 2^4$$

$$= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 - 2^4$$

$$\rightarrow (11011)_2, \quad \text{nicht } 11 = (01011)_2$$





oder

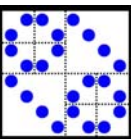
$$-11 = 5 - 16 \rightarrow (00101)_2$$

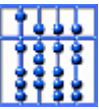
### *ZU BEACHTEN:*

- Fehler bei Bereichsüberschreitung (Overflow)  
ev. *Wrap-around*-Effekt.
- Integer-Division in Programmiersprachen:  $1 / 3 = 0$   
Division ohne Rest oder Rundung zur Null (Abschneiden).
- Division durch 0 (*Beispiel: USS Yorktown*)

Vorteil: Null hat nur eine Darstellung!

*Sonst treten bei Integerzahlen keine Rundungsfehler auf!*





## 2.5. Gleitpunktzahlen

Stelle  $x \in \mathfrak{R}$  durch Vorzeichen, Mantisse  $m$  und Exponenten  $e$  dar, bzgl. Basis  $b > 1$ :

$$x = (-1)^v \cdot m \cdot b^e$$

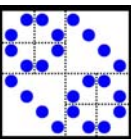
Wir betrachten nur  $b=2$  (b=8,10,16 kommen kaum vor)

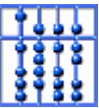
Beispiel:

Darstellungen der Zahl 16:

$$\dots = 0.25 \cdot 2^6 = 0.5 \cdot 2^5 = 1 \cdot 2^4 = 2 \cdot 2^3 = 4 \cdot 2^2 = \dots$$

mit  $v=0$ ,  $m = 2^{-i}$  und  $e = i+4$ .

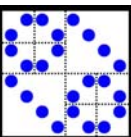


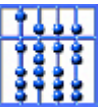


**Normierung ist notwendig, damit Darstellung eindeutig:  
Der Exponent ist stets so zu wählen, dass die Mantisse  
m genau eine von Null verschiedene Stelle vor dem  
Komma hat.**

Also in unserem Beispiel  $16 = + 1.0 \cdot 2^4$

Vorsicht: andere Bücher normieren so, dass erste Stelle  
hinter dem Komma von Null verschieden!  
(also  $16 = + 0.5 \cdot 2^5 = + (0.1)_2 \cdot 2^5$  )





## 2.6. Definition: Normierte Gleitpunktzahlen

Die Menge  $M$  der (reellen) Maschinenzahlen besteht aus Zahlen der Form

$$x = (-1)^v \cdot \sum_{i=0}^{t-1} x_i 2^{-i} \cdot 2^e$$

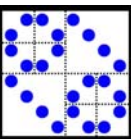
**Vorzeichen:**  $v \in \{0,1\}$

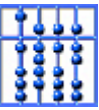
**Mantisse:**  $x_0 = 1$ ,  $x_i \in \{0,1\}$  sonst,

**Exponent:** Integerzahl  $e$ .

Also hat die Mantisse eigentlich  $t$  Stellen; aber die erste Stelle muss nicht gespeichert werden, da sie wegen der Normierung 1 ist.

Daher werden für die Mantisse nur  $(t-1)$ -Bits gebraucht!





*Bei Zahlen nahe 0 kann die Normierung aufgehoben werden (sog. subnormale Gleitpunktzahlen)!*

*(Ausnahmeregeln, falls Exponent minimal oder maximal ist!)*

Exponent wird gespeichert als ganzzahlige Integer-Maschinenzahl

### **Folien-Beispiel:**

$$13.6 = 8 + 5.6$$

$$= 8 + (4 + 1.6)$$

$$= 8 + 4 + (1 + 0.6)$$

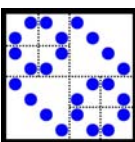
$$= 8 + 4 + 1 + (1/2 + 0.1)$$

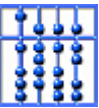
$$= 8 + 4 + 1 + 1/2 + (0.0625 + 0.0375)$$

$$= 8 + 4 + 1 + 1/2 + 1/16 + (0.03125 + 0.00625)$$

Also im Zweiersystem

$$13.6 = (1101.10011\dots)_2$$



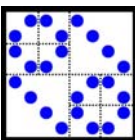


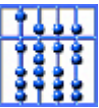
Als normierte Gleitpunktzahl:

$$13.6 = (-1)^0 \cdot (1.10110011\dots)_2 \cdot 2^3 \quad \text{mit } e=3.$$

## Die Zahl 13.6 in verschiedenen Genauigkeiten:

$t$	<i>Darstellung</i>	<i>Rundung</i>	<i>Fehler</i>
2	$(1.1)_2 * 2^3$	<i>ab</i>	$(1.6)_{10}$
3	$(1.11)_2 * 2^3$	<i>auf</i>	$(0.4)_{10}$
4	$(1.110)_2 * 2^3$	<i>auf</i>	$(0.4)_{10}$
5	$(1.1011)_2 * 2^3$	<i>ab</i>	$(0.1)_{10}$
6	$(1.10110)_2 * 2^3$	<i>ab</i>	$(0.1)_{10}$
7	$(1.101101)_2 * 2^3$	<i>auf</i>	$-(0.025)_{10}$





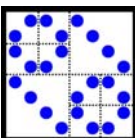
**Anzahl der für Mantisse und Exponent benutzten Bit und daraus sich ergebende Maschinengenauigkeit und Exponentenbereich bei IEEE-Datentyp *float* und *double*:**

<i>Typ</i>	<i>Mantisse t</i>	<i>Exponent</i>	$[e_{\min}, e_{\max}]$	$\varepsilon$
<i>float</i>	23(+1)	8	$[-126, 127]$	$2^{-24} \approx 6 * 10^{-8}$
<i>double</i>	52(+1)	11	$[-1022, 1023]$	$2^{-53} \approx 1 * 10^{-16}$

**IEEE-Standard (single precision, 32 Bit):**

Exponent e mit 8 Bit, gespeichert in der ‚positiven‘ Form

$$p = e + 127$$







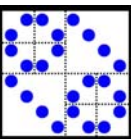
$p=(0000\ 0000)_2$  ist dann kleinstmöglicher Exponent.  
Ist auch noch in der Mantisse (bis auf Normierung)  
alles 0, so wird diese Zahl als Null interpretiert.

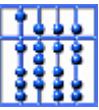
Entsprechend  $p=(1111\ 1111)_2 = 255$   
als unendlich (NaN oder *Not a number*)

$$-126 \leq e \leq 127 \quad \text{entspricht} \quad 1 \leq p \leq 254$$

Für die Mantisse bleiben 23 Bit  
(24 unter Berücksichtigung der Normierung).  
1 Bit für Vorzeichen  $\rightarrow$  Insgesamt  $1+23+8=32$  Bit .

*Bereits Konrad Zuse verwendete in seinen Z1-Z4  
Gleitpunktzahlen.*





## Beispiel Ariane 5

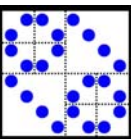
1996 endete die erste Ariane 5 durch Selbstzerstörung  
40 sec nach Start.

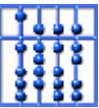
Ursache: 36.7 sec nach Start versuchte der Bordcomputer den  
Wert der horizontalen Geschwindigkeit von 64 Bit Gleitpunkt in  
16 Bit signed Integer umzurechnen.

Der sich ergebende Wert war zu groß → Overflow → Absturz  
des Computers, → Übergabe an Back-up-Rechner, der aber aus  
demselben Grund bereits abgestürzt war → Kein Lenksystem  
mehr → instabiler Flug → Selbstzerstörung.

Benutzte Software stammte von Ariane 4  
Ariane 5 war schneller.

Umwandlung war nicht abgesichert!





## Rundung

**2.7. Def.:** Für  $x = (-1)^v \cdot 2^e \cdot 1.x_1x_2\dots x_{t-1}x_t\dots$  definieren wir

$$rd(x) = (-1)^v \cdot 2^e \cdot 1.x_1x_2\dots x_{t-1} \text{ für } x_t = 0$$

$$rd(x) = (-1)^v \cdot 2^e \cdot 1.x_1x_2\dots x_{t-1} + 2^{-t+1} \text{ für } x_t = 1 \text{ und } x_tx_{t+1}x_{t+2}\dots \neq 1000\dots$$

$$rd(x) = (-1)^v \cdot 2^e \cdot 1.x_1x_2\dots x_{t-1} \text{ für } x_t = 1 \text{ und } x_{t-1}x_tx_{t+1}\dots = 0100\dots$$

$$rd(x) = (-1)^v \cdot 2^e \cdot 1.x_1x_2\dots x_{t-1} + 2^{-t+1} \text{ für } x_t = 1 \text{ und } x_{t-1}x_tx_{t+1}\dots = 1100\dots$$

1. Fall: Abschneiden, falls  $x_t = 0$ ;
2. Fall: Letztes Bit wird um eins erhöht, falls

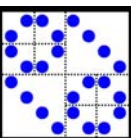
$$x_t = 1 \text{ und } x_tx_{t+1}x_{t+2}\dots \neq 1000\dots$$

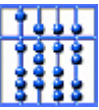
3. Fall:  $(x_t = 1)$ 

$$rd(1.x_1x_2\dots x_{t-2}0|1) = 1.x_1x_2\dots x_{t-2}0$$

oder

$$rd(1.x_1x_2\dots x_{t-2}1|1) = 1.x_1x_2\dots x_{t-2}1 + 2^{-t+1}$$
- } Rundung so, dass letztes Bit 0 wird



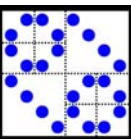


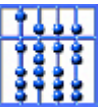
*Es kann ev. Overflow auftreten (zu großer Exponent).  
Dieser Fall wird aber im Folgenden ignoriert (Fehlermeldung?)*

*Es kann Underflow auftreten.  
In der Regel wird dann einfach die kleine Zahl zu 0 gesetzt.*

Die Rundungsfehleranalyse in den folgenden Abschnitten wird nur für Normalfall durchgeführt (ohne Over/Underflow)

Dabei ist nur die Mantisse wichtig!  
Der Exponent spielt keine Rolle, weil dabei keine Fehler auftreten.





## 2.8. Absoluter Rundungsfehler:

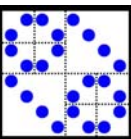
$$f_{rd}(x) = x - rd(x)$$

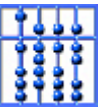
$$|f_{rd}(x)| = |x - rd(x)| \leq \frac{1}{2} 2^{-t+1} 2^e = 2^{e-t}$$

Problem: Ein absoluter Fehler von der Größe 0.1 ist

- bei der Zahl 2.1 recht groß, aber
- bei der Zahl 123456.7 sehr klein.

Daher sinnvollere Definition durch





## 2.9. Relativer Rundungsfehler:

$$f_{rel}(x) := \varepsilon_x := \frac{f_{rd}(x)}{x} = \frac{\delta_x}{x} = \frac{x - rd(x)}{x} \quad \text{für } x \neq 0$$

Dann gilt:

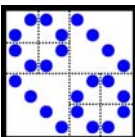
$$|f_{rel}(x)| \leq \frac{2^{e-t}}{|x|} \leq \frac{2^{e-t}}{2^e} = 2^{-t}$$

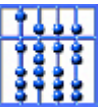
da wegen Normierung die Mantisse in  $[1,2]$  liegt:

Mantisse stets  $(1.bbbbb\dots)_2 < 2$

also  $m \geq 1$ , d.h.  $|x| \geq 2^e$

Außerdem gilt durch Umformung von 2.9:





$$\mathbf{2.10.}: \quad rd(x) = x - x\varepsilon_x = x(1 - \varepsilon_x)$$

$$\mathbf{mit} \quad |\varepsilon_x| \leq 2^{-t}$$

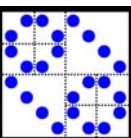
Gerundete Zahl =  
Ausgangszahl, bis  
auf Faktor  $(1 \pm \varepsilon)$

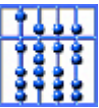
**2.11. Def.:** Die obere Schranke für den relativen Fehler, der bei der Rundung mit t-stelliger Mantisse auftreten kann, heißt **Maschinengenauigkeit**  $\varepsilon$ , und ergibt sich als

$$\varepsilon = 2^{-t}$$

Andere Möglichkeit, die Maschinengenauigkeit zu definieren:

Größte positive Zahl  $y=2^{-k}$ , so dass  $1.0 + y = 1.0$



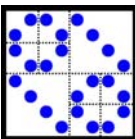


Beispiel  $t=2$ ,  $\varepsilon = \frac{1}{4} = (0.01)_2$ ;  $(1.0|1)_2 \rightarrow (1.0)_2$

**Mantissenlänge (Bits)  $\leftrightarrow$  Genauigkeit**

## Gleitpunktarithmetik

- 2.12. Def. (Realisierung einer Maschinenoperation):**
- Berechne für Maschinenzahlen das Ergebnis der Operation mit höherer Genauigkeit (quasi exakt)
  - Runde dieses Resultat wieder auf Maschinenzahl.





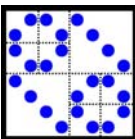


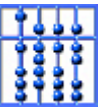
Dadurch ist der auftretende Fehler ausschließlich gegeben durch den Rundungsfehler, der im letzten Schritt auftritt!

### **Beispiel Addition $+_M$ :**

**Ausgangspunkt: Normierte Gleitpunktdarstellung beider Zahlen.**

- Verschiebe bei einer Zahl den Exponenten, so dass beide Zahlen den gleichen Exponenten haben.**
- Addiere nun die Mantissen.**
- Normalisiere das Ergebnis (verschiebe das Komma).**
- Runde das Ergebnis.**





Folien-Beispiel:  $x=7/4$  und  $y=3/8 \rightarrow x+y=17/8$

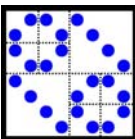
Mantisse mit  $t=3$

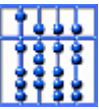
$$\begin{aligned}(1.11)_2 2^0 +_M (1.10)_2 2^{-2} &= \\ &= (111)_2 2^{-2} + (1.10)_2 2^{-2} \\ &= (1000.10)_2 2^{-2} \\ &= (1.00010)_2 2^1 \\ &= (1.00)_2 2^1 .\end{aligned}$$

Also  $x + y = 17 / 8$  , aber  $x+_M y = 2$  .

Absoluter Fehler :  $|17/8 - 2| = 1/8$

Relativer Fehler:  $\left| \frac{1/8}{17/8} \right| = 0.0588... \cong 6\%$





Zum Vergleich: Bei  $t=3$  ist die Maschinengenauigkeit

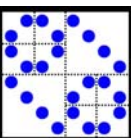
$$\varepsilon = 2^{-3} \cong 12.5\%$$

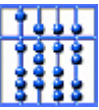
Der auftretende Fehler  $\varepsilon_+$  der Gleitpunktaddition beruht also auf der abschließenden Rundung, und nach 2.10 und 2.11 gilt daher:

$$\mathbf{2.13.} \quad x +_M y = rd(x + y) = (x + y)(1 + \varepsilon_+)$$

mit  $|\varepsilon_+| \leq \varepsilon$  **Maschinengenauigkeit**

In der Praxis ersetzt man die exakte Addition der Mantissen (Schritt 2) durch eine Addition mit höherer Genauigkeit, meist mit doppelter Genauigkeit. Danach Rundung auf Maschinenzahl.





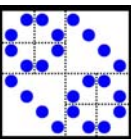
Ähnliches Modell bei Multiplikation / Division und auch bei anderen Funktionsauswertungen.

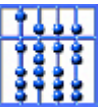
*Beispiel: Realisierung der Gleitpunkt-Division in INTEL-Prozessor und INTEL-Pentium-Bug 1994.*

## Fehlerfortpflanzung und Rundungsfehleranalyse

### **Problem:**

Rundungsfehler in der Eingabe und bei jeder durchgeführten Gleitpunktoperation können sich so auswirken, dass am Ende einer Berechnung ein vollkommen falsches Resultat herauskommt.





## Beispiel:

Mit Taschenrechner starte mit Zahl 2 und wiederhole k-mal die Wurzeloperation. Danach starte mit diesem Endresultat und wiederhole k-mal das Quadrieren.

Endresultat sollte stets wieder 2 sein.

Für k genügend groß erhält man aber 1. **(MATLAB)**

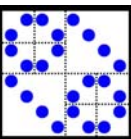
**AUFGABE: Finde bessere Art der Berechnung!**

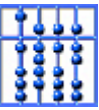
## Einführendes Folien-Beispiel zur Epsilontik:

Addition dreier Maschinenzahlen  $y=a+b+c$

Zerlege Gesamtrechnung in zwei Grundoperationen:

$$1. e=a+_M b \quad \text{und} \quad 2. f=e+_M c$$



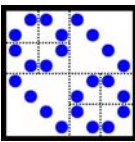


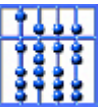
$$\begin{aligned} f &= e +_M c \\ &= (e + c)(1 + \varepsilon_2) \\ &= ((a +_M b) + c)(1 + \varepsilon_2) \\ &= ((a + b)(1 + \varepsilon_1) + c)(1 + \varepsilon_2) \\ &= a + b + c + (a + b)\varepsilon_1 + (a + b + c)\varepsilon_2 + (a + b)\varepsilon_1\varepsilon_2 \end{aligned}$$

mit  $|\varepsilon_1|, |\varepsilon_2| \leq \varepsilon$  Maschinengenauigkeit

Vernachlässigung der Terme höherer Ordnung  
(in  $\varepsilon^2, \varepsilon^3, \dots$ ):

$$f \doteq a + b + c + (a + b)\varepsilon_1 + (a + b + c)\varepsilon_2$$



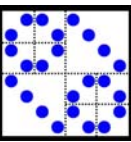


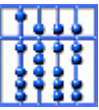
Damit ergibt sich für den relativen Fehler in erster Näherung:

$$\begin{aligned}f_{rel}(y) &= \frac{y - f}{y} \\ &\doteq \frac{a + b + c - (a + b + c + (a + b)\varepsilon_1 + (a + b + c)\varepsilon_2)}{a + b + c} \\ &= -\frac{a + b}{a + b + c}\varepsilon_1 - \varepsilon_2\end{aligned}$$

und die Abschätzung

$$|f_{rel}(y)| \doteq \left| \frac{a + b}{a + b + c}\varepsilon_1 + \varepsilon_2 \right| \leq \left( 1 + \left| \frac{a + b}{a + b + c} \right| \right) \varepsilon$$





Wann wird der relative Fehler groß?

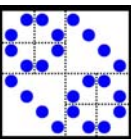
Wenn  $|a+b| \gg |a+b+c|$ , oder  $a+b+c \approx 0$

Andere Reihenfolge der Berechnung liefert Faktoren  
 $|(b+c)/(a+b+c)|$  oder  $|(a+c)/(a+b+c)|$ ;

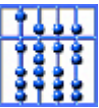
Es wird jeweils der Fehler, der bei der ersten Addition auftritt, verstärkt.

Folien-Beispiel: Maschinenzahlen

$$\begin{aligned} a &= (1.11)_2 * 2^{-1}, & b &= - (1.10)_2 * 2^{-1} \text{ und} \\ c &= (1.10)_2 * 2^{-3} & & \text{bei dreistelliger Mantisse.} \end{aligned}$$







Addition:

$$\begin{aligned}\tilde{y} &= \left( (1.11)_2 * 2^{-1} +_M (-(1.10)_2 * 2^{-1}) \right) +_M (1.10)_2 * 2^{-3} \\ &= (1.00)_2 * 2^{-3} +_M (1.10)_2 * 2^{-3} \\ &= (1.01)_2 * 2^{-2}\end{aligned}$$

Dabei tritt kein Fehler auf!

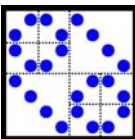
Andere Reihenfolge:

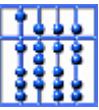
$$\begin{aligned}\hat{y} &= (1.11)_2 * 2^{-1} +_M \left( (-(1.10)_2 * 2^{-1}) +_M (1.10)_2 * 2^{-3} \right) \\ &= (1.11)_2 * 2^{-1} +_M (-(1.00)_2 * 2^{-1}) \\ &= (1.10)_2 * 2^{-2}\end{aligned}$$

mit relativem Fehler

$$\left| \frac{(1.01)_2 * 2^{-2} - (1.10)_2 * 2^{-2}}{(1.01)_2 * 2^{-2}} \right| = 20\%$$

**Merke:** Reihenfolge der Operationen ist wichtig!





Bisher waren  $a$ ,  $b$  und  $c$  Maschinenzahlen

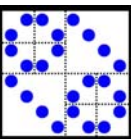
Jetzt betrachten wir Eingangszahlen, die schon selbst mit Rundungsfehler behaftet sind:

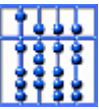
$a \rightarrow a(1+\varepsilon_a)$  mit  $|\varepsilon_a| \leq \varepsilon$ , usw.

1.  $e = (a \cdot (1 + \varepsilon_a)) +_M (b \cdot (1 + \varepsilon_b))$

2.  $f = e +_M (c \cdot (1 + \varepsilon_c))$ .

**Relativer Fehler in erster Näherung:**



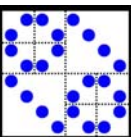


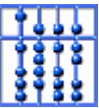
$$\frac{f - y}{y} \doteq \frac{a}{a+b+c} \varepsilon_a + \frac{b}{a+b+c} \varepsilon_b + \frac{c}{a+b+c} \varepsilon_c + \frac{a+b}{a+b+c} \varepsilon_1 + \varepsilon_2$$

**Erste Terme: Auswirkung der Eingabefehler**

**Vierter Term: Auswirkung des Fehlers bei der ersten Addition**

**Fünfter Term: Fehler bei der zweiten Addition**





## Auslöschung

Kritischer Fall: Endergebnis nahe bei Null!

### Folien-Beispiel:

Differenz zwischen  $x=3/5$  und  $y=4/7$  bei fünf-stelliger Mantisse.

Exakte Rechnung:  $x - y = 1/35 = (0.11101\dots)_2 2^{-5}$

Rundung von  $x$  und  $y$  liefert

für  $(1.0011001\dots)_2 2^{-1}$  und  $(1.001001\dots)_2 2^{-1}$

die Näherungen  $(1.0011)_2 2^{-1}$  und  $(1.0010)_2 2^{-1}$

Damit ergibt sich die Rechnung

$$\begin{aligned} \underline{(1.0011)}_2 2^{-1} - \underline{(1.0010)}_2 2^{-1} &= \\ &= \underline{(0.0001)}_2 2^{-1} = (1.0000)_2 2^{-5} \end{aligned}$$

