# Zahlendarstellung und Maschinengenauigkeit eps

October 26, 2018

## 0.1 Bemerkungen zur Fehlerrechnung der Vorlesung

Maschinen rechnen im allgemeinen nicht genau. Ein einfaches Beispiel zeigt das bereits:

```
In [2]: h=1e-18
        print(1+h)
        print((h+1)-1)
        print(h+(1-1))
```

```
1.0
0.0
1e-18
```

Für ausreichend kleines $h$ gilt $1 + h = 1$ auf dem Rechner. Dort ist der relative Fehler $h/1$ natürlich dann klein. Bei $(h+1)-1$ beträgt der relative Fehler aber schon 100%. Dieses Phänomen gilt es zu verstehen, wir wollen zunächst den Rundungsfehler in diesem Kapitel abschätzen.

Kurze Nebenbemerkung: Auch Rechengesetze werden auf dem Rechner verletzt. Die letzte Zeile zeigt, dass das Assoziativgesetz auf dem Rechner nicht gilt.

Auf dem Rechner werden alle (Fließkomma-) Zahlen in der Form

$$\pm 0.m_1 m_2 \dots m_p \cdot b^{\pm e}$$

abgespeichert (in der Basis $b$), $m_1 \neq 0$. Dabei geht natürlich Genauigkeit verloren bei Zahlen, die keine endliche $b$-adische Darstellung besitzen.

Nehmen wir als Beispiel die 1 in der Darstellung mit $p = 2$ und $b = 10$. Die Darstellung ist

$$0.10 \cdot 10^1.$$

Die auf dem Rechner nächstgrößere Zahl wäre

$$0.11 \cdot 10^1.$$

Alles dazwischen muss nach oben oder unten gerundet werden. Den größten Fehler bekommt man für die Zahl genau in der Mitte zwischen den beiden, also

$$0.105 \cdot 10^1.$$

Der maximale absolute Fehler ist 0.05. Ausgedrückt mit $b$ und $p$ ist das

$$\frac{b}{2} b^{-p} = \frac{1}{2} b^{-p+1}.$$

Wir betrachten nun irgendeine Zahl zwischen 1 und 10. Für alle diese Zahlen ist nach dieser Betrachtung der maximale absolute Fehler $0.05 = \frac{1}{2}b^{-p}$. Die Rundung dieser Zahl ist mindestens 1. Also ist der maximale relative Fehler

$$\frac{|x - rd(x)|}{|rd(x)|} \leq \frac{b}{2}b^{-p} = \frac{1}{2}b^{-p+1}.$$

Diese Zahl bezeichnen **wir** als eps oder Maschinengenauigkeit und folgen damit der klassischen Literatur (etwa Stoer/Bulirsch, Numerische Mathematik). Matlab und Python definieren diese Zahl, leider, anders, nämlich als den Abstand von 1 bis zur nächstgrößeren Zahl, und das sind nach unserer Definition 2 eps.

Wir benutzen hier gern $b = 10$. Auf dem Rechner benutzt man $b = 2$. Damit ist dort die Rechengenauigkeit $2^{-p}$. Zusätzlich gibt es dort noch eine Spezialität: Da an der ersten Stelle hinter dem Komma immer eine 1 steht, speichert man die nicht mit ab. Also benutzt man in Wirklichkeit $p + 1$ Stellen. Damit ist der Wert von eps in diesem Fall $2^{-(p+1)}$.

Im IEEE-Standard 754 für doppelte Genauigkeit wird festgelegt: $p = 52$. Wir erhalten damit

```
In [3]: import math
        math.exp(-53*math.log(2))

Out[3]: 1.1102230246251573e-16
```

Die relative Genauigkeit einer doppelt genauen Gleitkommazahl ist also $1.1\,10^{-16}$.

Aus der Herleitung hier sehen wir, dass $1 + eps$ die größte Zahl ist, die noch zur 1 gerundet wird. Wir können also ein ganz einfaches Programm schreiben, das eps berechnet. Dazu nehmen wir ein großes eps und machen es so lange kleiner, bis $1 + eps = 1$ auf dem Rechner.

```
In [4]: factor=0.9
        eps=1
        while (1+eps>1):
            eps=eps*factor
        print(eps)

1.0730891267324267e-16
```

Die Größenordnung kennen wir damit schon mal. Mit einem etwas aufwändigeren Programm lässt sich der Wert auch genauer bestimmen.

```
In [10]: factor=0.9999
         eps1=eps
         for i in range(1,14):
             eps=eps1
             while (1+eps>1):
                 eps1=eps
                 eps=eps*factor
             factor=factor+(1-factor)*0.9
         print(eps1)
         print(eps)
```

```
1.1102230246251562e-16
1.1102230246251562e-16
```

Und das ist genau das, was wir oben auch schon berechnet haben.

Die Ingenieure benutzen eine andere Definition für eps, deshalb bekommt Python einen anderen Wert heraus.

```
In [11]: import numpy as np
         print(np.finfo(float).eps)
         print(2*eps)
```

```
2.220446049250313e-16
2.2204460492503123e-16
```

Der zweite in IEEE754 definierte Zahlentyp ist single precision. Wir wiederholen unser Programm in single precision.

```
In [7]: factor=0.9
        eps=1
        while (np.float32(1+eps)>1):
            eps=eps*factor
        print(eps)
```

```
5.8927293063124204e-08
```

Die Genauigkeit von single precision ist also ungefähr $5 \cdot 10^{-8}$. Tatsächlich ist hier $p = 23$, also ist eps nach unserer Rechnung oben genau bestimmt durch

```
In [8]: math.exp(-24*math.log(2))
```

```
Out[8]: 5.960464477539072e-08
```

Mit diesen Bemerkungen ist unsere Betrachtung von Maschinenzahlen beendet. Was Sie sich merken sollten: Beim Speichern/Verarbeiten einer Zahl auf dem Rechner muss gerundet werden. Bei diesem Runden entsteht ein relativer Fehler, der durch eps nach oben begrenzt ist. eps liegt in der Größenordnung von $10^{-8}$ (single precision) bzw. $10^{-16}$ (double precision).