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

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

Andere Möglichkeit, die Maschinengenaugkeit zu definieren:
Größte positive Zahl $y=2^{-k}$, so dass

$$1.0 + y = 1.0$$

Beispiel t=2, $\varepsilon = ¼ = (0.01)_2$; $(1.0|1)_2 \rightarrow (1.0)$

Mantissenlänge (Bits) ←→ Genauigkeit

Zusammenfassung

Relativer Rundungsfehler: $$\left| f_{rel}(x) \right| := \left| \frac{f_{rd}(x)}{x} \right| = \left| \frac{x - rd(x)}{x} \right| \leq 2^{-t} = \varepsilon$$

Maschinengenauigkeit ϵ, Mantissenlänge t, durch Normierung erstes Bit gleich 1.

Was könnte problematisch sein?

Für jede sauber implementierte Operation/Funktion op(x) gilt :

$$op_M(x) = rd(op(x)) = op(x)(1+\varepsilon'), \qquad |\varepsilon'| \leq \varepsilon$$

Beispiel: Stelle die Zahl 1/10 binär dar und runde auf 3 Mantissen-Stellen.

MATLAB und Heisenberg Effekt

```
function x = myrealmin()
x = 1;
temp = x;
while eps * temp / 2 > 0
    temp = (eps * temp / 2)  ;
    if (temp > 0)
        x = temp;
    end
end
```

heisen.m

```
a = 1;
while a*eps>0
        last = a;
        a = a/2.0;
end
a = last
```

Verschiedene Ergebnisse abhängig von Ausgabe oder Script.

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.

Beispiel: x=7/4 und y=3/8 → x+y=17/8

Mantisse mit t=3

$(1.11)_2\ 2^0 +_M (1.10)_2\ 2^{-2} =$

$= (111)_2\ 2^{-2} + (1.10)_2\ 2^{-2}$

$= (1000.10)_2\ 2^{-2}$

$= (1.00010)_2\ 2^1$

$= (1.00)_2\ 2^1$.

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

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

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

Zum Vergleich: Bei t=3 ist die Maschinengenauigkeit

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

Der auftretende Fehler ε_+ der Gleitpunktaddition entsteht also durch die abschließenden Rundung!

Nach 2.10 und 2.11 gilt daher

2.13. $x +_M y = rd(x+y) = (x+y)(1+\varepsilon_+)$

mit $|\varepsilon_+| \leq \varepsilon$ = die 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.
Vortrag bugs.

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 km1.m)**

AUFGABE: Finde bessere Art der Berechnung!

Einführendes Beispiel zur Epsilontik:

Addition dreier Maschinenzahlen y=a+b+c

Zerlege Gesamtrechnung in zwei Grundoperationen:

1. $e = a +_M b$ und 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 ε^2, ε^3, ...):

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

Damit ergibt sich für den relativen Fehler in erster Näherung:

$$f_{rel}(y)=\frac{y-f}{y}$$

$$\doteq \frac{a+b+c-(a+b+c+(a+b)\varepsilon_1+(a+b+c)\varepsilon_2)}{a+b+c}$$

$$=-\frac{a+b}{a+b+c}\varepsilon_1-\varepsilon_2$$

und die Abschätzung

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

Ist da was problematisch?

Wann wird der relative Fehler groß?

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

Andere Reihenfolge der Berechnung liefert Faktoren

$|(b+c)/(a+b+c)|$ oder $|(a+c)/(a+b+c)|$;

z.B. $b+c \approx 0 \approx a+b+c$. Welche Reihenfolge wäre dann gut?

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

Beispiel: Maschinenzahlen
$a=(1.11)_2 * 2^{-1}$, $b= - (1.10)_2 * 2^{-1}$ und
$c=(1.10)_2 * 2^{-3}$ bei dreistelliger Mantisse.

Addition:

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

Dabei tritt kein Fehler auf!
Andere Reihenfolge?

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

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 kann wichtig sein!

Bisher waren a, b und c Maschinenzahlen
Jetzt betrachten wir Eingangszahlen, die schon selbst mit Rundungsfehler behaftet sind:
a → a(1+ε_a) mit $|\varepsilon_a| <= \varepsilon$, 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!

Beispiel:

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

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

Rundung von x und y liefert
für $\mathbf{(1.0011001\ldots)_2\ 2^{-1}}$ und $\mathbf{(1.001001\ldots)_2\ 2^{-1}}$
die Näherungen $\mathbf{(1.0011)_2\ 2^{-1}}$ und $\mathbf{(1.0010)_2\ 2^{-1}}$

Damit ergibt sich die Rechnung

$$\mathbf{(1.0011)_2\ 2^{-1} - (1.0010)_2\ 2^{-1} =}$$
$$\mathbf{= (0.0001)_2\ 2^{-1} = (1.0000)_2\ 2^{-5}}$$

Dabei sind unterstrichene Stellen noch exakt, während nicht unterstrichene, rote Stellen durch Rundung verfälscht sind.

Die kursiven, blauen Nullen im Ergebnis sind wertlos!

Das berechnete Ergebnis lautet also **1/32.**

Relativer Fehler:
(1/35 - 1/32) / (1/35) = - 0.0938

entspricht ca. 9.4% Abweichung.
Vgl. Maschinengenauigkeit für t = 5 von 0.031 ca. 3.1%

Die unterstrichenen, ‚guten' Stellen gehen durch die Differenz verloren und es bleiben die unsicheren Stellen übrig.

$(\underline{1.001}1)_2\ 2^{-1} - (\underline{1.001}0)_2\ 2^{-1} = (\underline{0.000}1)_2\ 2^{-1} = (1.\mathit{0000})_2\ 2^{-5}$

Bei **t=3** zeigt sich dieser Effekt noch stärker:

Rechnung: $(1.0\underline{1})_2\ 2^{-1} - (1.0\underline{1})_2\ 2^{-1} = 0$

Fehler: 100%,

bei Maschinengenauigkeit 0.125=1/8 oder 12.5%

Relativer Fehler bei Differenz y = a - b nach 2.13:

$$\varepsilon_y = \frac{a - b - \boxed{(a(1+\varepsilon_a) - b(1+\varepsilon_b)) \cdot (1+\varepsilon_-)}}{a-b}$$

$$\doteq -\frac{a}{\boxed{a-b}}\varepsilon_a + \frac{b}{\boxed{a-b}}\varepsilon_b - \varepsilon_-$$

Eingabefehler werden extrem verstärkt, wenn a-b nahe bei Null ist, also falls sich a und b fast auslöschen!

Der Fehler bei der Differenz selbst ist unkritisch!

Aber:

Sind a und b exakt ohne Fehler, dann ist

$$\varepsilon_a = 0 \quad \text{und} \quad \varepsilon_b = 0 \quad .$$

Daher ergibt sich dann nur ein relativer Fehler in der Größenordnung der Maschinengenauigkeit!

Also Differenz mit exakten Zahlen ist OK!

Nur bei Differenz von fehlerbehafteten Zahlen droht Gefahr.

Berechnung der Exponential-Funktion

$$\exp(x) = \sum x^k / k!$$

an einer Stelle X mittels Programm:

```
Y:=1.0 ;   T=1.0;    K=1;
WHILE ( Y  ≠ Y + T*X / K )
          T = T * X / K ;   Y = Y+ T ;  K = K + 1 ;
END
```

X	Y	$EXP(X)$
1	2.718282	2.718282
20	$4.8516531*10^{8}$	$4.8516520*10^{8}$
-10	$-1.6408609*10^{-4}$	$4.5399930*10^{-5}$
-20	1.202966	$2.0611537*10^{-9}$

Erklärung?

Für X = -15 ergibt sich:

1 – 15 + 112.5 - 562.5 +... - 312540.3 + 334864.6 – 334864.6 +
+313935.5 - ... – 0.00000061660813 ... =
= 3.050..*10^{-7}

Auslöschung durch wiederholte Differenz im Schritt **T = T + Y** !

Der Term T wächst zunächst, um am Ende einen sehr kleinen Wert anzunehmen!
Große Zwischenwerte + kleine Endwerte → Auslöschung!
Problematisch!

Kondition und Stabilität

2.14 Definition:
Ein Berechnungsverfahren ist eine Folge von mathematischen Berechnungen zur Lösung eines Problems mit Eingangsdaten $x \in IR^n$ **und dem Ergebnis** $y = f(x) \in IR$

Zur Berechnung von y wird es verschiedene Algorithmen geben, die sich z.B. in der Reihenfolge der Operationen unterscheiden (vgl. Addition a+b+c).

Zum Vergleich verschiedener Algorithmen betrachtet man die entstehenden Rundungsfehler.
Dazu kann man u.a. Taylor-Entwicklung oder Epsilontik verwenden.

Wir definieren die **Kondition** eines Problems.
Dazu betrachten wir

- Eingabedaten x_i , versehen mit absoluten Rundungsfehlern δ_{x_i}, i=1,...,n. (Zur Vereinfachung: n=1, also nur ein x)

- f(x) als black box; wir sind nur an der Ein- und Ausgabe interessiert!

Rundungsfehler **innerhalb** der Ausführung von f(x) sollen zunächst nicht auftreten!
Für den absoluten Fehler im Resultat gilt dann – unter Vernachlässigung der während der Berechnung sonst auftretenden Rundungsfehler:

$$y + \delta_y = f(x + \delta_x) = f(x) + f'(x)\delta_x + \mathrm{O}(\delta_x{}^2).$$

In erster Näherung gilt daher

$$y+\delta_y = f(x+\delta_x) = f(x)+ f'(x)\delta_x + \mathrm{O}(\delta_x{}^2) \Rightarrow$$

$$\delta_y \doteq f'(x)\delta_x$$

Daher ist der relative Fehler des Resultats y

$$\varepsilon_y = f_{rel}(y) = \frac{\delta_y}{y} \doteq \frac{xf'(x)}{y}\cdot\frac{\delta_x}{x} = \frac{xf'(x)}{y}\, f_{rel}(x) = \frac{xf'(x)}{f(x)}\cdot\varepsilon_x$$

2.15. Definition:

Unter der Konditionszahl des Problems

y = f(x)

bezüglich Eingabewert x

versteht man den Betrag des Verstärkungsfaktors

$$cond_x := \left|\frac{x\cdot f'(x)}{f(x)}\right|$$

$$cond_x := \left| \frac{x \cdot f'(x)}{f(x)} \right|$$

Die Konditionszahl misst die Sensibilität des Resultats y in Abhängigkeit von den Fehlern in der Eingabe x.

cond groß, z.B. wenn:

- große Eingabe gegenüber kleinem Endwert
- nahezu senkrechte Tangente (| f ’(x) | groß)

Ein Problem heißt *gut konditioniert* ←→

wenn kleine relative Fehler in x bei exakter Arithmetik (also ohne Rundungsfehler während der weiteren Rechnung) zu kleinen relativen Fehlern im Resultat y führen:
ε_y ungef. in der Größenordnung von ε_x

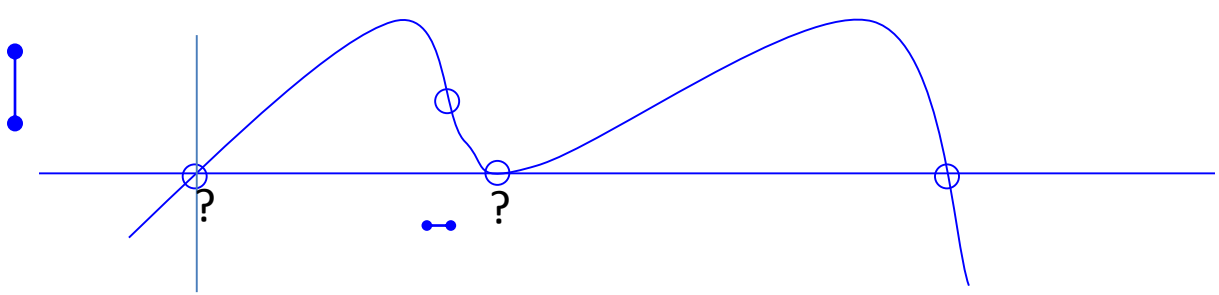
Andernfalls liegt schlechte Kondition bzgl. x vor.

Die Konditionszahl misst den sog. unvermeidbaren Fehler, der durch **das Problem selbst** an einer Stelle x gegeben ist.

Beispiel: cond(exp(x)) = |x|
cond(ln(x)) = |1/ ln(x)|

$$cond_x := \left| \frac{x \cdot f'(x)}{f(x)} \right|$$

Bild einer Funktion, Punkte schlechter Kondition:



$$f(x) = \begin{cases} x \\ x^2 \\ x^{(10^6)} \\ (x-1)^2 \end{cases}$$

Frage: Schlecht konditionierte Probleme im Alltag?

Beispiel: Konditionszahlen zu **y=f(a,b,c)=a+b+c** $cond_x := \left| \frac{x \cdot f'(x)}{f(x)} \right|$

$$cond_a = \left| \frac{a}{a+b+c} \right|, \qquad cond_b = \left| \frac{b}{a+b+c} \right|, \qquad cond_c = \left| \frac{c}{a+b+c} \right|,$$

Das sind gerade die Verstärkungsfaktoren der rel. Fehler der Eingabedaten in der Formel für den relativen Fehler:

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

Konditionszahl bzgl . der zweiten Addition ***f(a+b,c)=(a+b)+c***

Unvermeidbarer Fehler!

Betrachten wir die Gesamtrechnung, so lassen sich Konditionszahlen zu jedem einzelnen Rechenschritt angeben.

Damit ist es möglich, für den gesamten Algorithmus das Fehlerverhalten zu bestimmen.

Dies ist meist zu aufwändig oder gar nicht möglich!

Es ermöglicht aber eine mehr mathematische Formulierung der *Epsilontik*.

z.B. ist der vierte blaue Term gleich der Konditionszahl der Teilfunktion, die die Addition von (a+b) mit c beschreibt.

2.16. Definition:

Sei das Problem y=f(x) gut konditioniert.

Existiert dann zusätzlich auch ein gutartiges Berechnungsverfahren, bei dem die relativen Fehler nicht zusätzlich stark vergrößert werden, so spricht man von einem *numerisch stabilen* Algorithmus.

Ein Berechnungsverfahren, das trotz kleiner Konditionszahl zu vergrößerten relativen Fehlern im Resultat führen kann, heißt *numerisch instabil*.

Erste Frage: Konditionszahl OK?

Wenn ja, formuliere numerisch stabiles Berechnungsverfahren:

Prüfe das Berechnungsverfahren mit Epsilontik:

Ersetze dazu jede Eingangsvariable x durch $x(1+\varepsilon_x)$ und jede auszuführende Operation

$$(x\ op_M\ y) = (x\ op\ y)*(1+\varepsilon_{op})$$

mit $|\varepsilon_x| <= \varepsilon$ und $|\varepsilon_{op}| <= \varepsilon$.

Vernachlässige dabei Terme höherer Ordnung in ε (also $\varepsilon^2, \varepsilon^3, \varepsilon^4, ...$).

Damit erhält man das gestörte Endergebnis.

Berechne und diskutiere dann den relativen Fehler in erster Ordnung durch Abschätzen der Beträge der Einzelterme

$$\left| f_{rel} \right| \leq \left| Term \right| \cdot eps + \left| Term \right| \cdot eps + \ldots$$

Ist das Problem schlecht konditioniert, dann ist nur Schadensbegrenzung möglich:

Verwende ev. höhere Genauigkeit:

	Eingabefehler	*10^(-12)*
mit	*Konditionszahl*	*10^(8)*
ergibt	*Ausgabefehler*	*10^(-4)*

Ist dieser Ausgabefehler noch tolerierbar?
Wenn nein, dann kann zu einer Verbesserung nur der Eingabefehler verkleinert werden.

Patriot Bug! Vortrag.