# Numerisches Programmieren (IN0019)

**Frank R. Schmidt**

Winter Semester 2016/2017









## 2. Kondition und Stabilität 2 / 38

## Fließkomma-Arithmetik 3 / 38

### Fließkommazahlen (Wdh.)

Eine Fließkommazahl benutzt die folgende Zahlendarstellung

$$x = (-1)^s \cdot m \cdot 2^e,$$

wobei

| | |
|---|---|
| $s \in \mathbb{B}$ | das Vorzeichen, |
| $m \in [0, 2[$ | die Mantisse und |
| $e \in \mathbb{N}$ | den Exponenten |

kodieren. Weiter gehen wir davon aus, dass Fließkommazahlen üblicherweise normiert sind, d.h. für $x \neq 0$ gilt $m \in [1, 2[$.

Im Folgenden wollen wir uns die grundlegenden Operationen (+, −, ⋆ und /) für Fließkommazahlen ansehen.



### Fließkomma-Arithmetik

Im Computer wird eine Maschinenoperation üblicherweise wie folgt definiert:

1. Fasse die Fließkommazahl als reelle Werte auf und berechne die mathematische Operation exakt.
2. Runde das Ergebnis wieder auf eine Maschinenzahl.

In der Praxis wird dies dadurch erreicht, dass man die Operationen nicht exakt, sondern lediglich mit einer höheren Genauigkeit berechnet wird, so dass es keinen Unterschied macht, ob man die fast exakte Lösung oder die exakte Lösung rundet.

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







### Addition $x + y$

Wenn eine der beiden Summanden gerade 0 ist, ist die Summe der entsprechende andere Summand, d.h. $x +_{\mathbb{M}} 0 = x$.

Wir können also davon ausgehen, dass beide Fließkommazahlen normiert sind. Dann berechnet man die Summe wie folgt:

1. Verschiebe bei einer Zahl den Exponenten, so dass diese Zahl nicht mehr normiert ist.
2. Addiere die beiden Mantissen (mit doppelter Genauigkeit).
3. Normalisiere das Ergebnis (verschiebe das Komma)
4. Runde das Ergebnis.

Um das exakte Ergebnis zu berechnen, müsste man im Schritt 2 mit maximaler Genauigkeit rechnen. Allerdings reicht die doppelte Genauigkeit aus, um alle Effekte auf die Rundung feststellen zu können.





### Addition (Beispiel)

Betrachten wir

$$x = 1.75 \qquad y = 0.375 \qquad \Rightarrow \qquad x + y = 2.125$$

Bei einer Mantissenlänge von 2 erhalten wir

$$x = (1.11)_2 \cdot 2^0 \qquad y = (1.10)_2 \cdot 2^{-2} \qquad \Rightarrow \qquad x + y = (1.0001)_2 \cdot 2^1$$

Die Maschinenoperation $+_{\mathbb{M}}$ wird wie folgt berechnet

$$\begin{aligned}
&(1.11)_2 2^0 + (1.10)_2 2^{-2} && \text{(Eingabe)}\\
&\rightsquigarrow [(1.1100)_2 + (0.0110)_2]\, 2^0 && \text{(Schritt 1)}\\
&\rightsquigarrow [(10.0010)_2]\, 2^0 && \text{(Schritt 2)}\\
&\rightsquigarrow [(1.00010)_2]\, 2^1 && \text{(Schritt 3)}\\
&\rightsquigarrow [(1.00)_2]\, 2^1 && \text{(Schritt 4)}
\end{aligned}$$





### Addition (Beispiel)

Wir haben also

$$1.75 + 0.375 = 2.125 \qquad 1.75 +_{\mathbb{M}} 0.375 = 2$$

Damit erhalten wir

| | |
|---|---|
| einen absoluten Fehler von | $2.125 - 2 = 0.125$ |
| einen relativen Fehler von | $\frac{2.125 - 2}{2.125} \approx 5.88\%$ |

Der Fehler entsteht nur durch die Rundung.

Der relative Fehler ist kleiner als die Maschinengenauigkeit
$\varepsilon_{\mathbb{M}} = \frac{1}{2} 2^{-2} = 12.5\%$.



### Fehler bei Fließkomma-Arithmetik

**Lemma 1.** *Gegeben sei eine allgemeine mathematische Operation* $\circ \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. *Für die entsprechende Fließkomma-Operation* $\circ_{\mathbb{M}} \colon \mathbb{M} \times \mathbb{M} \to \mathbb{M}$ *gilt*

$$x \circ_{\mathbb{M}} y = \mathit{round}(x \circ y) = (x \circ y) \cdot (1 + \varepsilon_0),$$

*wobei* $|\varepsilon_0| \leqslant \varepsilon_{\mathbb{M}}$

Diese Beobachtung führt dazu, dass die konkreten Umsetzung der grundlegenden Operationen für unsere Fehleranalysen nicht so wichtig sind. Wir müssen lediglich wissen, dass jede Maschinenoperation einen relativen Fehler produziert, der kleiner ist als die Maschinengenauigkeit.

Der Standard IEEE 754 fordert dies für Addition/Subtraktion, Multiplikation/Division sowie für die Wurzelberechnung.





## Fehlerfortpflanzung 10 / 38

### Fehlerfortpflanzung

**Problem** Rundungsfehler in der Eingabe und bei jeder durchgeführten Fließkommaoperation können sich so auswirken, dass am Ende einer Berechnung ein vollkommen falsches Resultat herauskommt!

**Praktisches 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.

**Lösung** Bevor wir einen mathematischen Ausdruck im Computer berechnen, macht es Sinn, ihn in einen mathematisch äquivalenten Ausdruck umzuformulieren, der zu einem kleineren Fehler führt.



### Addition $x + y + z$

Zerlege Gesamtrechnung $x + y + z$ in zwei Grundoperationen:
$e = x +_{\mathbb{M}} y$ und $f = e +_{\mathbb{M}} z$.

Dann erhalten wir

$$\begin{aligned}
f &= (e + z)(1 + \varepsilon_1)\\
&= ((x + y)(1 + \varepsilon_0) + z)(1 + \varepsilon_1)\\
&= (x + y)(1 + \varepsilon_0)(1 + \varepsilon_1) + z(1 + \varepsilon_1)\\
&= x + y + z + \varepsilon_0(x + y) + \varepsilon_1(x + y + z) + \varepsilon_0\varepsilon_1(x + y)
\end{aligned}$$

mit $|\varepsilon_0|, |\varepsilon_1| \leqslant \varepsilon_{\mathbb{M}}$.

Üblicherweise ignorieren wir Fehler, die mindestens quadratisch in $\varepsilon_{\mathbb{M}}$ sind, da sie bei der abschließenden Rundung entfernt werden, wenn $\varepsilon_{\mathbb{M}}$ klein genug ist.

## Relativer Fehler von $x + y + z$

Betrachten wir das Ergebnis in erster Näherung

$$f = x + y + z + \varepsilon_0(x + y) + \varepsilon_1(x + y + z),$$

so erhalten wir für den relativen Fehler $\varepsilon_{\text{rel}}$

$$\begin{aligned}\varepsilon_{\text{rel}} &= \frac{f - (x + y + z)}{x + y + z} \\ &= \frac{\varepsilon_0(x + y) + \varepsilon_1(x + y + z)}{x + y + z} \\ &= \varepsilon_0 \frac{x + y}{x + y + z} + \varepsilon_1\end{aligned}$$

Damit gilt die Abschätzung

$$|\varepsilon_{\text{rel}}| \leqslant \left|\varepsilon_0 \frac{x + y}{x + y + z} + \varepsilon_1\right| \leqslant \left(1 + \frac{|x + y|}{|x + y + z|}\right) \varepsilon_{\mathbb{M}}$$



## Analyse des Relativen Fehlers

Um den relativen Fehler analysieren zu können, müssen wir uns den Faktor $\kappa := 1 + \frac{|x+y|}{|x+y+z|}$ genauer ansehen.

Addieren wir nur nicht-negative Werte, erhalten wir stets $\kappa \in [1, 2]$. In dieser Situation haben wir also eine gutartige Situation, da der Eingabefehler nur geringfügig verstärkt wird.

Allerdings wird $\kappa$ dann besonders groß, wenn

$$|x + y| \gg |x + y + z| \qquad \text{bzw.} \qquad x + y + z \approx 0$$

Diese Situation kann abgemildert werden, wenn wir die Reihenfolge der Summanden ändern.

Das liegt daran, dass nur der Fehler der ersten Summe verstärkt wird. Wenn dieses Teilergebnis nahe am Endergebnis ist, wird $\kappa$ wesentlich kleiner.



## $x + y + z$ (Beispiel)

Betrachten wir

$$x = 0.875 = (1.11)_2 2^{-1} \qquad y = -0.75 = -(1.10)_2 2^{-1} \qquad z = 0.1875 = (1.10)_2 2^{-3}$$

so erhalten wir

$$\begin{aligned}(x +_{\mathbb{M}} y) +_{\mathbb{M}} z &= (1.00)_2 2^{-3} +_{\mathbb{M}} (1.10)_2 2^{-3} = (1.01)_2 2^{-2} = 0.3125 \\ x +_{\mathbb{M}} (y +_{\mathbb{M}} z) &= (1.11)_2 2^{-1} +_{\mathbb{M}} (-(1.00)_2 2^{-1}) = (1.10)_2 2^{-2} = 0.375\end{aligned}$$

Bei der ersten Berechnung tritt kein Fehler auf.
Bei der zweiten Berechnung tritt folgender relativer Fehler auf:

$$\frac{0.375 - 0.3125}{0.3125} = \frac{\frac{1}{16}}{\frac{5}{16}} = 20\%$$

**Die Reihenfolge der Operationen ist wichtig!**



## Eingabefehler bei $x + y$

Bisher waren die Eingaben nicht fehlerbehaftet. Der Code `float x=0.1;` führt aber bereits zu einem Fehler, da $0.1$ keine Maschinenzahl ist.

Statt $x$ und $y$ betrachten wir nun

$$x \cdot (1 + \varepsilon_x) \qquad y \cdot (1 + \varepsilon_y) \qquad |\varepsilon_x|, |\varepsilon_y| \leqslant \varepsilon_{\mathbb{M}}$$

Dann erhalten wir für die Summe

$$\begin{aligned}s &= (x(1 + \varepsilon_x) + y(1 + \varepsilon_y)) \cdot (1 + \varepsilon_0) \\ &= x + y + \varepsilon_x x + \varepsilon_y y + \varepsilon_0(x + y) + o(|\varepsilon_{\mathbb{M}}|)\end{aligned}$$

Der relative Fehler (in erster Näherung) ist

$$\varepsilon_{\text{rel}} = \frac{s - (x + y)}{x + y} = \underbrace{\varepsilon_x \frac{x}{x + y} + \varepsilon_y \frac{y}{x + y}}_{\text{Auswirkung der Eingabefehler}} + \varepsilon_0$$



## Auslöschung

Man redet bei der Addition von **Auslöschung**, wenn das Ergebnis der Summe nahe bei Null ist.

Betrachten wir das Beispiel $x = \frac{9}{35} = (0.\overline{1001})_2$ und $y = -\frac{4}{7} = -(0.\overline{100})_2$. Dann ist $x + y = \frac{1}{352} = 0.\overline{000001110101}$.

Allerdings gilt mit vier bzw. zwei Nachkommastellen:

$$\begin{aligned}(1.0011)_2 2^{-1} -_{\mathbb{M}} (1.0010)_2 2^{-1} &= (1.0000)_2 2^{-5} = \frac{1}{32} \\ (1.01)_2 2^{-1} -_{\mathbb{M}} (1.01)_2 2^{-1} &= (0.00)_2 = 0\end{aligned}$$

Damit erhalten wir einen relativen Fehler von $\frac{1/32 - 1/35}{1/35} = 9.38...\%$ bzw. $100\%$.
Beide Fehler sind größer als die Maschinengenauigkeit von $3.125\%$ bzw $12.5\%$.

Die "guten" vorderen Stellen wurden **ausgelöscht** und es bleiben die fehlerbehafteten hinteren Stellen übrig!



## Zusammenfassung ($x + y$)

Der relative Fehler

$$\varepsilon_{\text{rel}} = \varepsilon_x \frac{x}{x + y} + \varepsilon_y \frac{y}{x + y} + \varepsilon_0$$

kann also bei der Differenz von Zahlen zu großen Fehlern führen.

Das gilt aber **nur bei fehlerbehafteten Eingaben!**

Rechnet man mit exakten Werten, so ist $\varepsilon_x = \varepsilon_y = 0$
und es tritt lediglich der Rundungsfehler $\varepsilon_0$ auf.

Wir erhalten also keinen Widerspruch zum IEEE Standard.



## Exponentialfunktion

Die Exponentialfunktion ist wie folgt definiert

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Daher ist es mathematisch sinnvoll, folgende Berechnung durchführen:

```
res  = 1.0;
term = x;
n    = 1.0;
while (res != res + term) {
  res  += term;
  n    += 1.0;
  term *= x/n;
}
```



## Exponentialfunktion (Auslöschung)

Die beschriebene Methode erzeugt die folgenden Ergebnisse:

| $x$ | res | $\exp(x)$ |
|---|---|---|
| 1 | 2.718282 | 2.718282 |
| 20 | $4.8516531 \cdot 10^8$ | $4.8516520 \cdot 10^8$ |
| -10 | $-1.6408609 \cdot 10^{-4}$ | $4.5399930 \cdot 10^{-5}$ |
| -20 | 1.202966 | $2.0611537 \cdot 10^{-9}$ |

Für $x = -20$ sehen die Berechnung wie folgt aus

$$1 - 20 + 200 - \ldots \pm \ldots + 43\,099\,804 - \ldots \pm \ldots + 10^{-28} - \ldots$$

Die Zwischenergebnisse wachsen zu Beginn, um am Ende sehr kleine Werte anzunehmen.

Diese Fehler treten für $x < 0$ auf, da hier durch wiederholte Differenzen Auslöschungen entstehen.

## Exponentialfunktion (Lösung)

Um Auslöschungen bei der Exponentialfunktion zu vermeiden, gibt es verschiedene Lösungsmöglichkeiten.

Eine einfache Vorgehensweise **für die Exponentialfunktion** wäre:

```
    y    = (x<0.0)?-x:x;
    res  = 1.0;
    term = y;
    n    = 1.0;
    while (res != res + term) {
      res  += term;
      n    += 1.0;
      term *= y/n;
    }
    if (x<0.0) {
      res = 1.0/res;
    }
```







## Berechnungsmethoden

**Definition 1.** Gegeben sei eine Funktion $f\colon \mathbb{R}^n \to \mathbb{R}^m$, wobei $n, m \in \mathbb{N}$ ist. Eine **Berechnungsmethode von** $f$ ist eine festgelegte, wohldefinierte Folge von mathematischen Elementarberechnungen $(+, -, \cdot, /)$, die aus Eingangsdaten $x \in \mathbb{R}^n$ das Ergebnis

$$y = f(x)$$

berechnet.

Zur Berechnung von $f$ wird es verschiedene Berechnungsmethoden (Algorithmen) geben (vgl. $x + y + z$ oder $\exp(x)$).

Beim Vergleich verschiedener Algorithmen betrachtet man die resultierenden Rundungsfehler.





## Absolute Kondition

Zuerst wollen wir untersuchen, wie sich $f$ verändert, wenn man die Eingabe $x \in \mathbb{R}^n$ durch $\delta \in \mathbb{R}^n$ stört, d.h., wenn man $f(x + \delta)$ statt $f(x)$ berechnet.

$f$ wird hier nur als "Black Box" verstanden und wir gehen davon aus, dass $f(x)$ exakt ohne Rundungsfehler berechnet werden kann.

$f$ verstärkt also den absoluten Fehler der Eingangsdaten um den Faktor

$$\frac{\|f(x+\delta) - f(x)\|}{\|\delta\|}.$$

Der Grenzwert für $\delta \to 0$ wird als **absolute Kondition** $\kappa_{\text{abs}}$ bezeichnet.

Insbesondere haben wir für differenzierbare Funktionen $f\colon \mathbb{R} \to \mathbb{R}$ gerade

$$\kappa_{\text{abs}} = |f'(x)|$$



## Limes superior

Wenn wir eine beschränkte Folge $(x_n)_{n\in\mathbb{N}}$ von Zahlen $x_n \in \mathbb{R}$ betrachten, dann ist es möglich, dass sie nicht konvergiert (z.B. $x_n = (-1)^n$).

Eine Verallgemeinerung des Grenzwertes ist der **Limes superior**, der auch für beschränkte Folgen berechnet werden kann, die nicht konvergieren.

Hier betrachtet man für jedes $n \in \mathbb{N}$ die kleinste obere Schranke $s_n$ der Menge $\{x_n, x_{n+1}, \ldots\}$. Diese Folge $(s_n)_{n\in\mathbb{N}}$ ist **monoton fallend und beschränkt**, d.h. sie konvergiert. Wir definieren nun den **Limes superior** von $x_n$ als

$$\limsup_{n\to\infty} x_n := \lim_{n\to\infty} s_n.$$

Da die absolute Kondition als die größte Verstärkung des Eingabefehlers definiert werden soll, definieren wir

$$\kappa_{\text{abs}} = \limsup_{\delta\to 0} \frac{\|f(x+\delta) - f(x)\|}{\|\delta\|}.$$







## Kondition

Da wir uns mit Fließkommazahlen beschäftigen, sind wir daran interessiert, wie der relative Fehler verstärkt wird.

Der relative Eingabefehler ist $\|\delta\| / \|x\|$ und der relative Ausgabefehler ist $\|f(x+\delta) - f(x)\| / \|f(x)\|$.

Daher definieren wir die **Kondition** von $f$ als

$$\kappa = \limsup_{\delta\to 0} \frac{\|f(x+\delta) - f(x)\| / \|f(x)\|}{\|\delta\| / \|x\|} = \frac{\|x\|}{\|f(x)\|} \cdot \limsup_{\delta\to 0} \frac{\|f(x+\delta) - f(x)\|}{\|\delta\|}$$
$$= \frac{\|x\|}{\|f(x)\|} \cdot \kappa_{\text{abs}}$$

Insbesondere haben wir für differenzierbare Funktionen $f\colon \mathbb{R} \to \mathbb{R}$ gerade

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



## Interpretation der Kondition

Die Konditionszahl misst die Sensibilität des Resultats $f(x)$ in Abhängigkeit von den Fehlern in der Eingabe $x$.

Konditionszahlen $\kappa < 1$ führen dazu, dass der Anfangsfehler gedämpft wird.

Große Konditionszahlen erhält man, wenn

- die absolute Kondition sehr hoch ist, d.h für große $|f'(x)|$
- die Ausgabe viel kleiner ist als die Eingabe ($\|f(x)\| \ll \|x\|$)

Ein Problem heißt **gut konditioniert**, wenn kleine **relative Fehler** in $x$ bei **exakter Arithmetik** (also ohne Rundungsfehler) zu kleinen relativen Fehlern im Resultat $f(x)$ führen, d.h. wenn $\kappa$ in der Größenordnung von 1 ist.

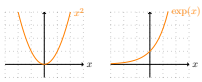
Andernfalls heißt das Problem **schlecht konditioniert**.







## Kondition (Beispiele)




Für $f(x) = x^2$ erhalten wir für die Kondition

$$\kappa := \left| \frac{2x \cdot x}{x^2} \right| = 2$$

Für $f(x) = \exp(x)$ erhalten wir für die Kondition

$$\kappa := \left| \frac{\exp(x) \cdot x}{\exp(x)} \right| = |x|$$





## Kondition von $x + y$

Wenn wir davon ausgehen, dass die Eingabedaten $x$ und $y$ unabhängig voneinander Fehler aufweisen, so lässt sich die Kondition von $x + y$ bzgl. $x$ und $y$ berechnen und wir erhalten

$$\kappa_x = \left| \frac{x}{x+y} \right| \qquad \kappa_y = \left| \frac{y}{x+y} \right|$$

Dies sind gerade die Verstärkungsfaktoren des relativen Fehlers

$$\varepsilon_{\text{rel}} = \varepsilon_x \frac{x}{x+y} + \varepsilon_y \frac{y}{x+y} + \varepsilon_0$$

Diese Fehler sind unvermeidbar und hängen nicht von dem gewählten Algorithmus ab.

**Verkettung von Funktionen**

Betrachten wir einen konkreten Algorithmus, so lassen sich Konditionszahlen für jeden einzelnen Rechenschritt angeben.

Damit ist es möglich, für den gesamten Algorithmus das Fehler-Verhalten zu bestimmen.

Sei $f, g\colon \mathbb{R} \to \mathbb{R}$ gegeben, dann gilt für die Kondition von $f \circ g$ gerade

$$\begin{aligned}\kappa_{f\circ g} &= \left|\frac{x\cdot (f\circ g)'(x)}{(f\circ g)(x)}\right| = \left|\frac{x\cdot f'(g(x))\cdot g'(x)}{f(g(x))}\right| \\ &= \left|\frac{x\cdot g'(x)}{g(x)}\right| \cdot \left|\frac{g(x)\cdot f'(g(x))}{f(g(x))}\right| = \kappa_f \cdot \kappa_g\end{aligned}$$

Das heißt wir haben nicht nur eine Kettenregel für die absolute Konditionszahl sondern auch für die (relative) Konditionszahl.



**Stabilität**

**Definition 2.** Sei $f\colon \mathbb{R}^n \to \mathbb{R}^m$ ein gut kondioniertes Problem. Wenn es ein Berechnungsverfahren für $f$ gibt, das die relativen Eingabefehler nicht vergrößert, dann wir dieses Berechnungsverfahren numerisch stabil.

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

Wenn sich ein Berechnungsverfahren als Verkettung von Funktionen $g_k\colon \mathbb{R}^{n_k} \to \mathbb{R}^{m_k}$ darstellen lässt, dann sollte jede Funktion gut konditioniert sein, damit das Verfahren numerisch stabil ist.

Wenn es möglich ist, sollte man Auslöschungen vermeiden!



**Stabilitätsanalyse**

- Prüfe, ob das Problem gut konditioniert ist
- Finde bei guter Kondition ein numerisch stabiles Berechnungsverfahren.

Bei der Analyse des Verfahrens, ersetze jede Variable $x$ durch $x(1+\varepsilon_x)$ und jede Operation $x \circ y$ durch $(x \circ y)(1+\varepsilon_i)$. Ignoriere anschließend Terme höherer Ordnung in $\varepsilon$ ($\varepsilon^2$, $\varepsilon^3$, etc.).

Alternativ kann man auch direkt die Ableitungen jeder Operation berechnen, um die Skalierung der Fehler zu ermitteln.

Bei schlechter Kondition ist nur Schadensbegrenzung möglich, z.B. indem man die Genauigkeit der Mantisse erhöht oder den Eingabefehler durch bessere Sensoren verringert.



**Stabilitätsanalyse (1. Beispiel)**

Wir wollen $f(x) = 1 - \sqrt{1-x^2}$ für $x \approx 0$ berechnen.
Das Problem ist gut konditioniert, da

$$\begin{aligned}\kappa &= \lim_{x\to 0}\left|\frac{x^2}{(1-\sqrt{1-x^2})(\sqrt{1-x^2})}\right| = \lim_{x\to 0}\left|\frac{2x}{2x-\frac{x}{\sqrt{1-x^2}}}\right| \\ &= \lim_{x\to 0}\left|\frac{2}{2-\frac{1}{\sqrt{1-x^2}}}\right| = 2\end{aligned}$$

Allerdings ist die Auswertung instabil (Auslöschung im letzten Schritt).

Folgender Ausdruck ist stabil, da jeder Einzelschritt gut konditioniert ist

$$f(x) = \frac{1+\sqrt{1-x^2}}{1+\sqrt{1-x^2}}(1-\sqrt{1-x^2}) = \frac{x^2}{1+\sqrt{1-x^2}}$$



**Stabilitätsanalyse (2. Beispiel)**

Wir wollen $f(x) = 1 - \cos(x)$ für $x \approx 0$ berechnen.
Das Problem ist wieder gut konditioniert, da

$$\begin{aligned}\kappa &= \lim_{x\to 0}\left|\frac{x\sin(x)}{1-\cos(x)}\right| = \lim_{x\to 0}\left|\frac{\sin(x)+x\cos(x)}{\sin(x)}\right| \\ &= \lim_{x\to 0}\left|\frac{\cos(x)-x\sin(x)+\cos(x)}{\cos(x)}\right| = 2\end{aligned}$$

Allerdings ist die Auswertung instabil (Auslöschung).

Folgender Ausdruck ist stabil, da jeder Einzelschritt gut konditioniert ist

$$f(x) = 1-\cos\left(\frac{x}{2}+\frac{x}{2}\right) = 1-\left(\cos\left(\frac{x}{2}\right)^2-\sin\left(\frac{x}{2}\right)^2\right) = 2\sin\left(\frac{x}{2}\right)^2$$



**Zusammenfassung (Problemstellung)**

Endlichkeit des Computers führt zu endlicher Menge von Maschinenzahlen.

In jedem Schritt treten Rundungsfehler auf.

Gefährlich sind Operationen, bei denen man signifikante Stellen verliert,
wie z.B.:

- Auslöschung (Differenz fast gleicher Zahlen)
- Summe zwischen sehr großer Zahl und sehr kleiner Zahl, bei der die signifikanten Stellen in der kleinen Zahl stecken (oft gefolgt von Auslöschungen in späteren Schritten)
- Allgemeiner: Operationsfolgen mit großen Zwischenwerten und kleinen Endwerten (vgl. $\exp(-20)$).





**Zusammenfassung (Lösungen)**

Algorithmus ist OK, wenn die Größenordnung der relativen Fehler im Resultat ungefähr gleich der Größenordnung der Eingabefehler bleibt.

Umformen eines numerisch instabilen Verfahrens durch

- andere Reihenfolge der Berechnung
- algebraische Umformung (binomische Formeln)
- trigonometrische Formeln
- Anfang der Taylorentwicklung

Systematische Fehler und große Zahl der Operationen können zu schlechten Ergebnissen führen!

Eventuell Modellfehler gegen Rundungsfehler abwägen:
Feineres Modell ⇒ Mehr Rechnung ⇒ Mehr Rundungsfehler!
Man muss die optimale Balance finden!



**Zusammenfassung (Lösungen)**

Verfahren ist numerisch stabil, wenn für jede Zerlegung
in Teilprobleme $f(x) = f_2 \circ f_1(x)$, das Teilproblem
$f_2$ in $z := f_1(x)$ gut konditioniert ist.

| | | |
|---|---|---|
| Konditionszahl | $\leftrightarrow$ | Gesamtproblem |
| Numerische Stabilität | $\leftrightarrow$ | Berechnungsform |

Erkenne aus Formel (Programm), bzw. berechneten Zwischenwerten,

- ob das Problem gut konditioniert ist
- ob das verwendete Verfahren numerisch stabil ist
  bzw. wie das Verfahren verbessert werden kann.