Auslöschung

Kritischer Fall: Endergebnis nahe bei Null!

Folien-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

$$\mathbf{(\underline{1.0011}1)_2\ 2^{-1} - (\underline{1.0010}0)_2\ 2^{-1} =}$$
$$\mathbf{= (\underline{0.0001})_2\ 2^{-1} = (1.\mathit{0000})_2\ 2^{-5}}$$

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

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

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

Fehler: 100%,
bei Maschinengenauigkeit 0.125=1/8 oder 12.5%

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

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

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

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

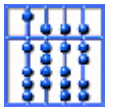
Aber:
Sind a und b exakt ohne Fehler, dann ist

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

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

Also Differenz mit <u>exakten</u> Zahlen ist OK!

Nur bei Differenz von <u>fehlerbehafteten</u> Zahlen droht Gefahr.

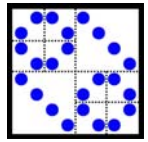


Berechnung der Exponential-Funktion an einer Stelle X mittels Programm:

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

```
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	$2.0611537*10^{-9}$



Für X = -15 ergibt sich:

1 – 15 + 112.5 - 562.5 +... - 312540.3 + 334864.6 – 334864.6 + 313935.5 - ... – 0.000000061660813 ... =

= 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!
Beispiel PPT: Patriot-Scud-Software-Bug

Kondition und Stabilität

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

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

Wir betrachten 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

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

und daher für den relativen Fehler

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

2.15. Definition:

Unter der Konditionszahl des Problems

y = f(x)

bezüglich Eingabewert x

versteht man den Betrag des Verstärkungsfaktors

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

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* ←→

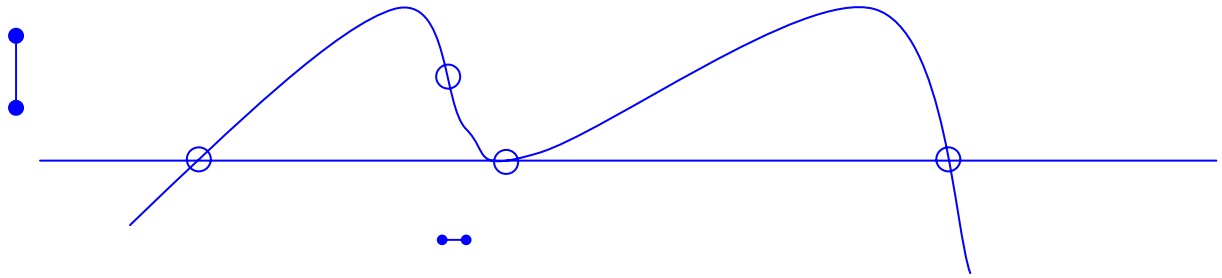
wenn kleine relative Fehler in x bei exakter Arithmetik (also ohne Rundungsfehler während der weiteren Rechnung) zu kleinen relativen Fehlern im Resultat y führen:
ε_y ungef. in der Größenordnung von ε_x

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: $\text{cond}(\exp(x)) = |x|$
$\text{cond}(\ln(x)) = |1/\ln(x)|$

Bild einer Funktion, Punkte schlechter Kondition:



Beispiel: Konditionszahlen zu **y=a+b+c**

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

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

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

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

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!
Ergäbe Verfeinerung 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.

Beispiel: Berechnung von

$$f(x) = 1 - \sqrt{1 - x^2}, \quad x \approx 0$$

Kondition ist OK, da

$$cond_x = \left| \frac{x^2}{(1 - \sqrt{1 - x^2})\sqrt{1 - x^2}} \right| \to 2 \quad \text{für} \quad x \to 0$$

(L'Hospital)

Allerdings ist die Auswertung in dieser Form numerisch instabil

$$x \to x^2 \to 1 - x^2 \to \sqrt{1 - x^2} \to 1 - \sqrt{1 - x^2}$$

da Auslöschung im letzten Schritt!

Bessere Formulierung:

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

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

Entsprechend lässt sich die Berechnung der Exponentialfunktion für große negative x ‚retten',
indem wir exp(-1000) ersetzen durch 1/exp(1000) .

Beispiel: **f(x) = 1 - cos(x)** in der Nähe von **x=0**
f(x) ist wieder gut konditioniert bei 0, da

$$cond_x = \left|\frac{xf'}{f}\right| = \left|\frac{x\cdot\sin(x)}{1-\cos(x)}\right| = \left|\frac{x^2+\cdots}{1-(1-x^2/2+\cdots)}\right| \to \quad 2, \quad x \to 0$$

Aber bei 0 ist cos(x) nahe bei 1 → wieder Auslöschung!
In MATLAB: 1 - cos(10^(-8)) ergibt 0;

oder in **cos(10^(-3))=0.999999500000004**

verliert man bei der Differenz 6 signifikante Stellen

Anderer Berechnungsweg:

$$\mathbf{1 - \cos(x) = 2 \sin^2(x/2)}$$

oder Reihenentwicklung des Cosinus

$$1-\cos(x) = 1-(1-\frac{x^2}{2}+\frac{x^4}{4!}-\frac{x^6}{6!}+\cdots) = \frac{x^2}{2}-\frac{x^4}{4!}+\frac{x^6}{6!}-\cdots$$

Beispiel: $\mathbf{y = a^2 - b^2}$ bei $\mathbf{|a|=|b|}$

Anwendung der Epsilontik; seien a,b Maschinenzahlen:
Berechne erst Produkte, dann Differenz

$$f = \left((a \cdot a) \cdot (1+\varepsilon_1) - (b \cdot b) \cdot (1+\varepsilon_2)\right) \cdot (1+\varepsilon_3)$$

Relativer Fehler:

$$\varepsilon_y = \frac{y-f}{y} \doteq -\frac{a^2}{a^2-b^2}\varepsilon_1 + \frac{b^2}{a^2-b^2}\varepsilon_2 - \varepsilon_3$$

Nun seien auch a und b fehlerhaft: **$a(1+\varepsilon_a)$, $b(1+\varepsilon_b)$**

$$\varepsilon_y \doteq \boxed{-\frac{2a^2}{a^2-b^2}\varepsilon_a + \frac{2b^2}{a^2-b^2}\varepsilon_b} \boxed{-\frac{a^2}{a^2-b^2}\varepsilon_1 + \frac{b^2}{a^2-b^2}\varepsilon_2} - \varepsilon_3$$

Fehler: Eingabefehler Produktfehler Differenzfehler

Konditionszahlen:

$$cond_a = \left|\frac{a \cdot \frac{dy}{da}}{a^2-b^2}\right| = \left|\frac{2a^2}{a^2-b^2}\right| \quad , \quad cond_b = \left|\frac{b \cdot \frac{dy}{db}}{a^2-b^2}\right| = \left|\frac{-2b^2}{a^2-b^2}\right|$$

Problem ist schlecht konditioniert für |a| ≅ |b|

Andere Art der Berechnung: **y = (a – b)(a + b)**

$$f = \left(a(1+\varepsilon_a) - b(1+\varepsilon_b)\right)(1+\varepsilon_-)\left(a(1+\varepsilon_a) + b(1+\varepsilon_b)\right)(1+\varepsilon_+)\cdot(1+\varepsilon_*)$$
$$\doteq \left(a^2(1+2\varepsilon_a) - b^2(1+2\varepsilon_b)\right)\cdot(1+\varepsilon_- + \varepsilon_+ + \varepsilon_*)$$

Relativer Fehler in erster Näherung:

$$\frac{-2a^2}{a^2-b^2}\varepsilon_a + \frac{2b^2}{a^2-b^2}\varepsilon_b - \varepsilon_- - \varepsilon_+ - \varepsilon_*$$

Vergleich mit erstem Algorithmus:

Das neue Verfahren ist besser, da i.W. nur der unvermeidbare Fehler (durch Eingabefehler) auftritt!

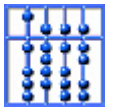
Grund: Auslöschung in $a - b$ geringer als in $a^2 - b^2$,
da Fehler in a und b kleiner als in a^2 und b^2.

Zusammenfassung

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 großer Zahl und sehr kleiner Zahl, bei der die signifikanten Stellen in der kleinen Zahl stecken (vgl. wiederholtes Wurzelziehen)**
- **Allgemein Operationsfolgen mit großen Zwischenwerten und kleinen Endwerten**

(vgl. exp, Teilfunktion schlecht konditioniert).

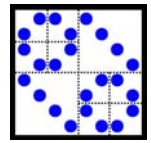


Vorsicht! Gesundes Misstrauen!

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

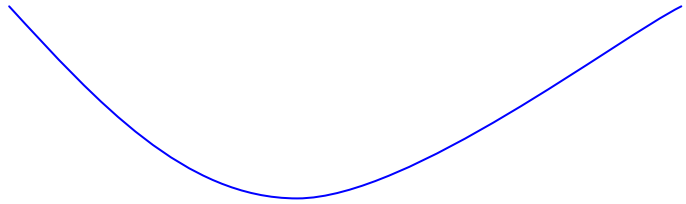
Umformen eines numerisch instabilen Verfahrens durch

- **andere Reihenfolge der Berechnung**
- **Anfang der Taylorentwicklung**
- **Trigonometrische Formeln**
- **algebraische Umformung (binomische F.)**
- **....**
- **Ev. double precision rechnen, damit trotz schlechter Kondition oder Rundungsfehler noch brauchbares Resultat übrigbleibt.**



Systematische Fehler und große Zahl der Operationen können zu schlechten Ergebnissen führen!
(Siehe Beispiel Börsenindex)

Ev. Modellfehler gegen Rundungsfehler abwägen:
Feineres Modell → Mehr Rechnung → Mehr Rundungsfehler!
Man muss die optimale Balance finden!
Beispiel Übungsaufgabe Differenzenquotient.
Gesamtfehler:



Grob diskretisiert →Modellfehler Optimum fein diskretisiert→Rundungsfehler

Beispiel: Verbesserte Fehleranalyse für den numerisch instabilen Fall großer Zwischenwerte

Zerlege Problem f(x) in zwei Schritte

$$y = f(x) = f_2(f_1(x)) = f_2(z)$$

wobei $z = f_1(x)$ großer Zwischenwert und

$y = f_2(z)$ kleiner Endwert.

Daher ist Teilproblem $f_2(z)$ für diese Werte schlecht konditioniert, da $|z / f_2(z)|$ groß ist!

Daher ist Gesamtverfahren nicht numerisch stabil für x.

Verfahren ist numerisch stabil, wenn für jede Zerlegung in Teilprobleme $f_2(f_1(x)) = f_2(z)$, $z = f_1(x)$, $f_2(z)$ stets gut konditioniert ist!

Konditionszahl ←→ Gesamtproblem

Numerisch stabil ←→ Berechnungsform

Genauere Analyse der numerischen Stabilität durch Bestimmung der Konditionszahlen und Ableitungen aller Teilschritte:

Zerlege Algorithmus in Teilprobleme $f(x) = f_2(f_1(x))$ und berechne alle auftretenden Konditionszahlen $\text{cond}(f_2)$!

Meist zu aufwändig oder unmöglich.

Epsilontik genügt für uns.
(Ersetze $x \rightarrow x(1+\varepsilon)$, $x \text{ op } y \rightarrow (x \text{ op } y)(1+\varepsilon)$
Streiche Terme höherer Ordnung in $\varepsilon^2, \varepsilon^3, \varepsilon^4, \ldots$
Bestimme damit den rel. Fehler des Resultats $(f - y)/f$
in erster Näherung und schätze Beträge ab nach oben
Diskutiere die einzelnen Terme.

Ziel:

Erkenne aus Formel (Programm), bzw. berechneten (Zwischen)werten,

- ob das Problem gut konditioniert ist, und

- ob das verwendete Verfahren numerisch stabil ist,

- bzw. wie das Verfahren ev. verbessert werden kann.