Fließkomma-Arithmetik und Fehlerfortpflanzung

Fließkomma-Arithmetik

Definition einer Maschinenoperation:

1. Berechne für Maschinenzahlen das Ergebnis der Operation mit höherer Genauigkeit (quasi exakt)

2. 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

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

Beispiel

x=7/4 und y=3/8 → x+y=17/8 Mantisse mit t=3

Beispiel

x=7/4 und y=3/8 → x+y=17/8 Mantisse mit t=3

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

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

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

$= (1.00010)_2\ 2^1$

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

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: $\left|\frac{1/8}{17/8}\right| = 0,0588\cdots \approx 6\%$

Maschinengenauigkeit: $\epsilon = 2^{-3} = 12,5$

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: $\left|\frac{1/8}{17/8}\right| = 0,0588\cdots \approx 6\%$

Maschinengenauigkeit: $\epsilon = 2^{-3} = 12,5$

Der Fehler entsteht durch die abschließende Rundung!

Fließkomma-Operationen

Für allgemeine Fließkomma-Operationen $\circ$ gilt:

$$\mathrm{rd}(r \circ s) = (r \circ s)(1 + \epsilon_\circ)$$

wobei $|\epsilon_\circ| \leq \epsilon$

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.

Rundungsfehleranalyse

Problem:

Rundungsfehler in der Eingabe und bei jeder durchgeführten Fließkommaoperation 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.

Addition

Addition dreier Maschinenzahlen y=a+b+c

Zerlege Gesamtrechnung in zwei Grundoperationen:

1. $e=a+_M b$ und 2. $f=e+_M c$

Addition

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

Addition

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 (ε^2, ε^3 , ...):

Addition

Ergebnis in erster Näherung:

$$f = a + b + c + (a + b)\epsilon_1 + (a + b + c)\epsilon_2$$

Relativer Fehler:

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

Addition

Ergebnis in erster Näherung:

$$f = a + b + c + (a + b)\epsilon_1 + (a + b + c)\epsilon_2$$

Relativer Fehler:

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

$$= \frac{a + b + c - (a + b + c + (a + b)\epsilon_1 + (a + b + c)\epsilon_2)}{a + b + c}$$

$$= -\frac{a + b}{a + b + c}\epsilon_1 - \epsilon_2$$

Addition

Ergebnis in erster Näherung:

$$f = a + b + c + (a + b)\epsilon_1 + (a + b + c)\epsilon_2$$

Relativer Fehler:

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

$$= \frac{a + b + c - (a + b + c + (a + b)\epsilon_1 + (a + b + c)\epsilon_2)}{a + b + c}$$

$$= -\frac{a + b}{a + b + c}\epsilon_1 - \epsilon_2$$

Damit gilt die Abschätzung

$$|f_{rel}(y)| \leq \left|\frac{a + b}{a + b + c}\epsilon_1 + \epsilon_2\right| \leq \left(1 + \left|\frac{a + b}{a + b + c}\right|\right)\epsilon$$

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

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

Beispiel

Addition von drei Maschinenzahlen:

$a=(1.11)_2 * 2^{-1}$, $b= -(1.10)_2 * 2^{-1}$ und $c=(1.10)_2 * 2^{-3}$

bei dreistelliger Mantisse.

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

$$= (1.00)_2 * 2^{-3} +_M (1.10)_2 * 2^{-3}$$

$$= (1.01)_2 * 2^{-2}$$

Dabei tritt kein Fehler auf!

Andere Reihenfolge?

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 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 $|\varepsilon_a| <= \varepsilon$, usw.

Mit Eingangsfehlern

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:

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

Erste Terme: Auswirkung der Eingabefehler

Vierter Term: Auswirkung 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

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

Auslöschung

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.

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

Auslöschung

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

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

Fehler: 100%,

bei Maschinengenauigkeit 0.125=1/8 oder 12.5%

Relativer Fehler bei Differenz y = a - b:

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

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

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

Auslöschung

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.

Beispiel Exponentialfunktion

Berechne $\exp(x) = \sum x^k / k!$ mit diesem 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?

Beispiel Exponentialfunktion

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!

Kondition und Stabilität

Kondition und Stabilität

Definition 19: Eine *Berechnungsmethode* ist eine festgelegte, wohldefinierte Folge von mathematischen Elementarberechnungen $(+, -, \cdot, /)$, die aus den Eingangsdaten $x \in \mathbb{R}, n \in \mathbb{N}^*$ das Ergebnis

$$y = f(x) \in \mathbb{R}$$

berechnet.

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.

Kondition

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

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:

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

Kondition

In erster Näherung gilt

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

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

Daher ist der relative Fehler des Resultats y

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

Definition 20: Die *Kondition* der Funktion $y = f(x)$ ergibt sich aus dem Verstärkungsfaktor

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

Kondition

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

κ groß, z.B. wenn:

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

Ein Problem heißt *gut konditioniert* ←→

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 ungefähr in der Größenordnung von ε_x

Kondition

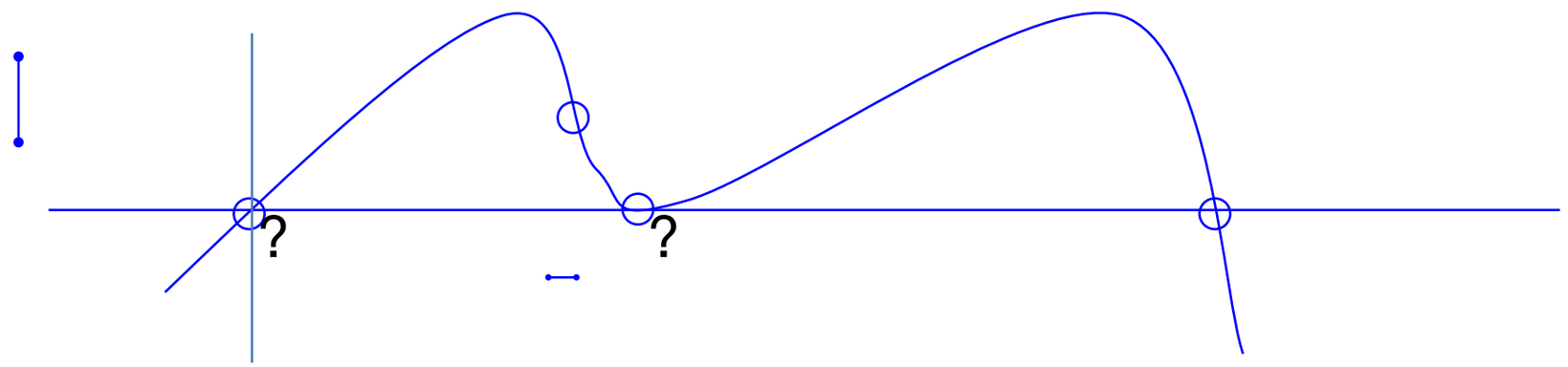
Andernfalls liegt schlechte Kondition bzgl. x vor.

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

Beispiel: $\kappa(\exp(x)) = |x|$

$\kappa(\ln(x)) = |1/\ln(x)|$

Bild einer Funktion, Punkte schlechter Kondition:



Beispiel: Addition

Berechne 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 relativen Fehler der Eingabedaten in der Formel für den relativen Fehler:

$$\frac{y-f}{y} \doteq \frac{a}{a+b+c}\epsilon_a + \frac{b}{a+b+c}\epsilon_b + \frac{c}{a+b+c}\epsilon_c +$$

$$+\frac{a+b}{a+b+c}\epsilon_1 + \epsilon_2$$

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

Unvermeidbarer Fehler!

Verkettete Berechnungsmethoden

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 Addition von (a+b) mit c.

Verkettete Berechnungsmethoden

Berechne die Konditionszahl für

$$z = g(y) = g(f(x)) = (g \circ f)(x)$$

Wir haben

$$\kappa_g = \left| \frac{y \cdot g'(x)}{z} \right| \qquad \kappa_f = \left| \frac{x \cdot f'(x)}{y} \right|$$

Verkettete Berechnungsmethoden

Berechne die Konditionszahl für

$$z = g(y) = g(f(x)) = (g \circ f)(x)$$

Wir haben

$$\kappa_g = \left| \frac{y \cdot g'(x)}{z} \right| \qquad \kappa_f = \left| \frac{x \cdot f'(x)}{y} \right|$$

$$\kappa_{g \circ f} = \left| \frac{x \cdot (g \circ f)'(x)}{z} \right| = \left| \frac{x \cdot g'(f(x)) \cdot f'(x) \cdot y}{z \cdot y} \right|$$

$$= \left| \frac{y \cdot g'(y)}{z} \right| \cdot \left| \frac{x \cdot f'(x)}{y} \right|$$

$$= \kappa_g \cdot \kappa_f$$

Verkettete Berechnungsmethoden

$$\underbrace{f_n(f_{n-1}(\cdots(f_3(f_2}_{g_2}\underbrace{(f_1(x)))}_{(h_1(x))})\cdots))$$

$$f(x) = g_1(x)$$

$$\underbrace{f_n(f_{n-1}}_{g_{n-1}}\underbrace{(\cdots(f_3(f_2(f_1(x)))))\cdots))}_{(\quad h_{n-2}(x) \quad)}$$

$$f_n(\underbrace{f_{n-1}(\cdots(f_3(f_2(f_1(x)))))\cdots)}_{h_{n-1}(x)})$$

$$g_n(\quad h_{n-1}(x) \quad)$$

Alle Funktionen g_j müssen gut konditioniert sein, da sie Teilschritte implementieren!

Stabilität

Definition 21: Sei $y = f(x)$ ein gut konditioniertes Problem. Wenn es ein Berechnungsverfahren für f gibt, das die relativen Eingabefehler nicht vergrößert, dann ist dieses Berechnungsverfahren *numerisch stabil*.

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

Stabilität

Erste Frage: Konditionszahl OK?

Wenn ja, finde 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 $\varepsilon^2, \varepsilon^3, \varepsilon^4, \ldots$).

Damit erhält man das gestörte Endergebnis.
Berechne und diskutiere dann den relativen Fehler in erster Ordnung durch Abschätzen der Beträge der Einzelterme $\left|f_{rel}\right| \leq \left|Term\right| \cdot eps + \left|Term\right| \cdot eps + \ldots$

Stabilität

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 1

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

Problematisch?

Kondition ist OK, da

$$cond_x = \left| \frac{x^2}{(1 - \sqrt{1 - x^2})\sqrt{1 - x^2}} \right| \rightarrow \quad 2 \qquad für \qquad x \rightarrow 0$$ (L'Hospital)

Allerdings ist die Auswertung in dieser Form numerisch instabil

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

da Auslöschung im letzten Schritt!

Beispiel 1

Bessere Formulierung:

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

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

Für $x \approx 0$ keine Subtraktion mehr!
Alle Einzelschritte sind gut konditioniert!

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

Beispiel 2

f(x) = 1 - cos(x) in der Nähe von **x=0**

f(x) ist wieder gut konditioniert bei 0, da

$$\kappa_x = \left|\frac{x f'(x)}{f(x)}\right| = \left|\frac{x \cdot \sin(x)}{1 - \cos(x)}\right| \to 2 \quad \text{für } x \to 0$$

Aber bei 0 ist cos(x) nahe bei 1 → wieder Auslöschung!

In MATLAB: 1 - cos(10^(-8)) ergibt 0;

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

verliert man bei der Differenz 6 signifikante Stellen

Beispiel 2

Anderer Berechnungsweg:

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

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 3

$y = a^2 - b^2$ bei **$|a|=|b|$**

Anwendung der Epsilontik; seien a,b Maschinenzahlen:

Berechne erst beide Produkte, dann die Differenz.

Relativer Fehler:

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

$$\epsilon_y \doteq \boxed{-\frac{2a^2}{a^2-b^2}\epsilon_a + \frac{2b^2}{a^2-b^2}\epsilon_b} \boxed{-\frac{a^2}{a^2-b^2}\epsilon_1 + \frac{b^2}{a^2-b^2}\epsilon_2} - \epsilon_3$$

Fehler: Eingabefehler Produktfehler Differenzfehler

Beispiel 3

Konditionszahlen:

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

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

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

$$f = (a(1+\epsilon_a) - b(1+\epsilon_b))(1+\epsilon_-)(a(1+\epsilon_a) + b(1+\epsilon_b))(1+\epsilon_+)(1+\epsilon_\cdot)$$

$$\doteq (a^2(1+2\epsilon_a) - b^2(1+2\epsilon_b)) \cdot (1+\epsilon_- + \epsilon_+ + \epsilon_\cdot)$$

Relativer Fehler in erster Näherung:

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

Beispiel 3

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 Zwischen-werten und kleinen Endwerten (vgl. exp, Teilfunktion schlecht konditioniert).

Zusammenfassung

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.

Zusammenfassung

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!

Zusammenfassung

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.

Zusammenfassung

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

Zusammenfassung

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.