Scientific Computing I

Error Analysis and Machine Numbers

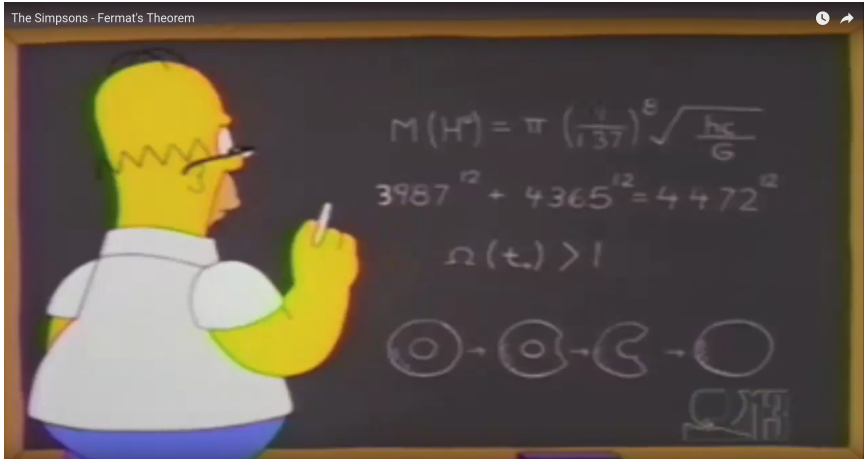
Martin Köhler

Computational Methods in Systems and Control Theory (CSC) Max Planck Institute for Dynamics of Complex Technical Systems

Winter Term 2024/2025

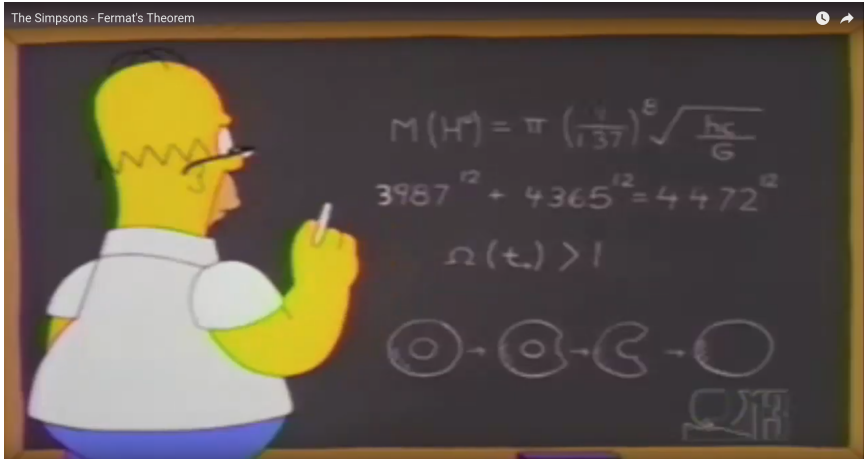
This Lecture:

Error Analysis and Machine Numbers



Homer claims: $3987^{12} + 4365^{12} = 4472^{12}$

[1]Image: The Simpsons S10E2 "The Wizard of Evergreen Terrace" ©FOX



Homer claims: $3987^{12} + 4365^{12} = 4472^{12}$

Using 3 significant digits: $1.61 \cdot 10^{43} + 4.78 \cdot 10^{43} = 6.39 \cdot 10^{43}$!!!

[1]Image: The Simpsons S10E2 "The Wizard of Evergreen Terrace" ©FOX

Machine Numbers

Machine Numbers

We are not having an ideal world:

- computers only provide finite memory,
- storing numbers is limited in the number of digits and accuracy,
- simple numbers like 1.0 or 0.5 can be stored easily,
- numbers like π but even $\frac{1}{3}$ require truncation,
- truncation causes an error in the representation.

p–adic expansion

Machine Numbers

p–adic expansion

Theorem (p–adic expansion)

For $x \in \mathbb{R}$, $p \in \mathbb{N}\backslash\{1\}$ there exist uniquely determined $j \in \{0,1\}$, $\ell \in \mathbb{Z}$ and $\forall k \in \mathbb{Z}$ with $k \leqslant \ell$ unique $\gamma_k \in \{0,\ldots,p-1\}$, such that

$$x = (-1)^j \sum_{k=-\infty}^{\ell} \gamma_k p^k, \tag{1}$$

where $\gamma_\ell \neq 0$ for $x \neq 0$, $j = \ell = 0$ for $x = 0$, and $\gamma_k < p-1$ for infinitely many $k \leqslant \ell$.

Machine Numbers

p–adic expansion

Theorem (p–adic expansion)

For $x \in \mathbb{R}$, $p \in \mathbb{N}\backslash\{1\}$ there exist uniquely determined $j \in \{0,1\}$, $\ell \in \mathbb{Z}$ and $\forall k \in \mathbb{Z}$ with $k \leqslant \ell$ unique $\gamma_k \in \{0,\ldots,p-1\}$, such that

$$x = (-1)^j \sum_{k=-\infty}^{\ell} \gamma_k p^k, \tag{1}$$

where $\gamma_\ell \neq 0$ for $x \neq 0$, $j = \ell = 0$ for $x = 0$, and $\gamma_k < p-1$ for infinitely many $k \leqslant \ell$.

- → most representations are based on this theorem
- → the expression "$\gamma_k < p-1$ for infinitely many k" means that, e.g., for $p = 10$ the number $3.\overline{9} = 3.99999\ldots$ is represented as 4.0
- → all summands are positive
- → $x = 0 \leftrightarrow \forall k \quad \gamma_k = 0$ and with $j = \ell = 0$ the representation of 0 is unique

Machine Numbers

p–adic expansion

The p–adic representation of a number given in a different number system can be expressed using the following representation:

$$(x)_p := \pm\gamma_\ell\gamma_{\ell-1}\cdots\gamma_0.\gamma_{-1}\gamma_{-2}\cdots,$$

where the digits following the separating "." are called the **mantissa**.

Machine Numbers

p–adic expansion

The p–adic representation of a number given in a different number system can be expressed using the following representation:

$$(x)_p := \pm\gamma_\ell\gamma_{\ell-1}\cdots\gamma_0\cdot\gamma_{-1}\gamma_{-2}\cdots,$$

where the digits following the separating "." are called the **mantissa**.

Systems like *Roman Numbers* (`IX, MC, ...`) does not fit in this system.

Decimal System

Machine Numbers

Decimal System

- used in everyday life
- $p = 10$
- digits: $\gamma_k \in \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\}$

$$x = \pm \sum_{k=-\infty}^{\ell} \gamma_k \cdot 10^k = \pm \gamma_\ell \gamma_{\ell-1} \dots \gamma_0 . \gamma_{-1} \gamma_{-2} \dots = (x)_{10}$$

Machine Numbers

Decimal System

- used in everyday life
- $p = 10$
- digits: $\gamma_k \in \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\}$

$$x = \pm \sum_{k=-\infty}^{\ell} \gamma_k \cdot 10^k = \pm \gamma_\ell \gamma_{\ell-1} \ldots \gamma_0 . \gamma_{-1} \gamma_{-2} \ldots = (x)_{10}$$

Definition

The expression $(x)_p$ means: The number x interpreted in the p–adic representation with base p.

Binary System

Machine Numbers

Binary System

- $p = 2$
- $\gamma_k \in \{0, 1\}$ or $\gamma_k \in \{L, H\}$
- easily representable by electrical signals
- often prefixed with `0b`

Machine Numbers

Binary System

- $p = 2$
- $\gamma_k \in \{0, 1\}$ or $\gamma_k \in \{L, H\}$
- easily representable by electrical signals
- often prefixed with `0b`

Example

The decimal number $x = 1123$ is translated into the binary system as follows:

$$\begin{aligned} 1123 &= 1024 + 99 = 2^{10} + 64 + 35 \\ &= 2^{10} + 2^6 + 32 + 3 = 2^{10} + 2^6 + 2^5 + 2^1 + 2^0, \end{aligned}$$

i.e., $(1123)_2 = 10001100011$.

Machine Numbers

Binary System

On the other hand, we have decimal number $\frac{1}{10}$ and get

$$\left(\frac{1}{10}\right)_2 = 0.0\overline{0011}.$$

Machine Numbers

Binary System

On the other hand, we have decimal number $\frac{1}{10}$ and get

$$\left(\frac{1}{10}\right)_2 = 0.0\overline{0011}.$$

So $\frac{1}{10}$ can not be written in a finite number of digits in the mantissa.

Octal System

Machine Numbers

Octal System

- $p = 8$
- digits: $\gamma_k \in \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\}$
- not so commonly used in computing, but in systems where binary is too lengthy and hexadecimal is too compact.
- often prefixed with a zero (e.g., `075`) or a leading `0o` (e.g., `0o75`)
- used in early computing systems and still relevant in some contexts today, such as file permissions in Unix/Linux systems.

Hexadecimal System

Machine Numbers

Hexadecimal System

- $p = 16$
- $\gamma_k \in \{0, 1, \ldots, 15\}$, usual representation uses $A = 10$, $B = 11$, $\ldots$, $F = 15$, and therefore the standard digits are $\gamma_k \in \{0, 1, \ldots, 9, A, B, \ldots, F\}$.

Machine Numbers

Hexadecimal System

- $p = 16$
- $\gamma_k \in \{0, 1, \ldots, 15\}$, usual representation uses $A = 10$, $B = 11$, $\ldots$, $F = 15$, and therefore the standard digits are $\gamma_k \in \{0, 1, \ldots, 9, A, B, \ldots, F\}$.

Example

The hexadecimal number $x = A1E$ turns into

$$(A1E)_{10} = 10 \cdot 16^2 + 1 \cdot 16^1 + 14 \cdot 16^0 = 10 \cdot 256 + 16 + 14 = 2590.$$

Machine Numbers

Hexadecimal System

- $p = 16$
- $\gamma_k \in \{0, 1, \ldots, 15\}$, usual representation uses $A = 10$, $B = 11$, $\ldots$, $F = 15$, and therefore the standard digits are $\gamma_k \in \{0, 1, \ldots, 9, A, B, \ldots, F\}$.

Example

The hexadecimal number $x = A1E$ turns into

$$(A1E)_{10} = 10 \cdot 16^2 + 1 \cdot 16^1 + 14 \cdot 16^0 = 10 \cdot 256 + 16 + 14 = 2590.$$

The translation decimal number $\leftrightarrow$ hexadecimal number is done via the binary system:

$$(1123)_2 = \underbrace{0100}_{4 \cdot 16^2} \; \underbrace{0110}_{6 \cdot 16^1} \; \underbrace{0011}_{3 \cdot 16^0} \quad \Rightarrow \quad (1123)_{16} = 463.$$

$\rightarrow$ a group of four binary digits give one hexadecimal digit.

Normalized Floating Point Representation

Machine Numbers

Normalized Floating Point Representation

The p–adic representation (1) is equivalent to

$$x = \underbrace{\left\{(-1)^j \sum_{k=-\infty}^{\ell} \gamma_k p^{k-\ell-1}\right\}}_{=:s} \cdot p^{\ell+1} =: \underbrace{\left\{(-1)^j \sum_{i=1}^{\infty} \frac{\alpha_i}{p^i}\right\}}_{=:a} p^b, \tag{2}$$

where $\alpha_i := \gamma_{\ell-i+1}$, $i = 1, \ldots$ and $b := l + 1$. In (1) we have $\gamma_\ell \neq 0$ and thus we immediately get $\frac{1}{p} \leqslant |s| < 1$.

Machine Numbers

Normalized Floating Point Representation

Definition

The representation of any $x \in \mathbb{R}$ as in (2) is called *normalized floating point representation* of x with respect to p. Here

$$a \quad := \quad (-1)^j \sum_{i=1}^{\infty} \frac{\alpha_i}{p^i} \quad \text{where } \alpha_i \in \{0, 1, \ldots, p-1\} \tag{3}$$

is called the **significand** and

$$b \quad := \quad (-1)^s \sum_{i=1}^{m} \beta_i p^{m-i}, \quad \text{for } s \in \{0, 1\},\ \beta_i \in \{0, 1, \ldots, p-1\} \tag{4}$$

the **exponent**.

This floating point representation is called normalized since $\alpha_1 \neq 0$.

Machine Numbers

Normalized Floating Point Representation

In contrast to $i = 1, \ldots, \infty$, a computer can only store finitely many digits in the significand.

In case $\alpha_i = 0$, $\forall\ i > t \in \mathbb{N}$, x can be encoded by saving j, s (for determining the signs of significand and exponent) and the digits in the p-adic representation of significand and exponent.

$\rightarrow$ Schematic representation:

j	α_1	$\ldots$	α_t	s	β_1	$\ldots$	β_m

Thus we require $1 + t + 1 + m$ memory positions.

Machine Numbers

Normalized Floating Point Representation

Example

For $p = 10$ the normalized floating point representation of the real number 35 657.23 is given as

$$0.3565723 \cdot 10^5 = \left(\frac{3}{10^1} + \frac{5}{10^2} + \frac{6}{10^3} + \frac{5}{10^4} + \frac{7}{10^5} + \frac{2}{10^6} + \frac{3}{10^7} \right) \cdot 10^5,$$

encoded as

0	3	5	6	5	7	2	3	0	5
j	α_1	α_2	α_3	α_4	α_5	α_6	α_7	s	β_1

.

In this example $t = 7$ and $m = 1$.

Computer Representable Numbers

Machine Numbers

Computer Representable Numbers

Definition

For $p \in \mathbb{N}\backslash\{1\}$, $e_{\min}, e_{\max} \in \mathbb{Z}$, $t \in \mathbb{N}$ we denote the *set of normalized floating point numbers of length t with respect to the base p and range of exponents* $\{e_{\min}, e_{\min}+1, \ldots, e_{\max}\} \subset \mathbb{Z}$ by

$$\mathbb{M}(p, t, e_{\min}, e_{\max}) := \{\ \pm 0.\alpha_1\alpha_2\ldots\alpha_t \cdot p^b \quad | \alpha_i \in \{0, \ldots, p-1\}, \alpha_1 \neq 0, \\ e_{\min} \leqslant b \leqslant e_{\max}\} \cup \{0\}\,.$$

$x \in \mathbb{M}(p, t, e_{\min}, e_{\max})$ is called **computer number** or **machine number**.

Machine Numbers

Computer Representable Numbers

Definition

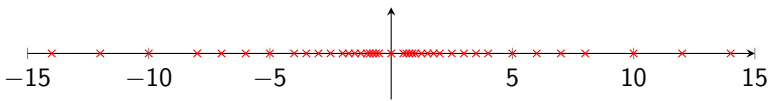
For $p \in \mathbb{N}\backslash\{1\}$, $e_{\min}, e_{\max} \in \mathbb{Z}$, $t \in \mathbb{N}$ we denote the *set of normalized floating point numbers of length t with respect to the base p and range of exponents* $\{e_{\min}, e_{\min}+1, \ldots, e_{\max}\} \subset \mathbb{Z}$ by

$$\mathbb{M}(p,t,e_{\min},e_{\max}) := \{\ \pm 0.\alpha_1\alpha_2\ldots\alpha_t \cdot p^b \quad | \alpha_i \in \{0,\ldots,p-1\}, \alpha_1 \neq 0, \\ e_{\min} \leqslant b \leqslant e_{\max}\} \cup \{0\}\,.$$

$x \in \mathbb{M}(p,t,e_{\min},e_{\max})$ is called **computer number** or **machine number**.

Example

The elements in $\mathbb{M}(2,3,-1,4)$ are shown in the following number ray



→ Note that machine numbers are not equally distributed.

Rounding Errors and Error Propagation

Rounding Rules

Rounding Errors and Error Propagation

Rounding Rules

Definition

The **rounding function**

$$\gamma : \mathbb{R} \to \mathbb{M}(p, t, e_{\min}, e_{\max})$$

for $x \in Z := [-x_{\max}, -x_{\min}] \cup \{0\} \cup [x_{\min}, x_{\max}]$ is determined by

$$\gamma(x) = \underset{\tilde{x} \in \mathbb{M}(p,t,e_{\min},e_{\max})}{\arg\min} |x - \tilde{x}|, \tag{5}$$

where

$$\begin{aligned} x_{\min} &:= \min \{|x| \mid x \in \mathbb{M}(p, t, e_{\min}, e_{\max}) \backslash \{0\}\}, \\ x_{\max} &:= \max \{|x| \mid x \in \mathbb{M}(p, t, e_{\min}, e_{\max})\}. \end{aligned}$$

Rounding Errors and Error Propagation

Rounding Rules

Let $x = \pm \sum_{i=1}^{\infty} \frac{\alpha_i}{p^i} \cdot p^b \in Z$ with $\alpha_1 \neq 0$. By truncating to t significant digits we get

$$\gamma(x) = \begin{cases} \pm \sum_{i=1}^{t} \frac{\alpha_i}{p^i} \cdot p^b, & \alpha_{t+1} < \frac{p}{2}, \\ \pm \left(\sum_{i=1}^{t} \frac{\alpha_i}{p^i} + \frac{1}{p^t} \right) \cdot p^b, & \alpha_{t+1} > \frac{p}{2}. \end{cases}$$

$\rightarrow$ What happens if $\alpha_{t+1} = \frac{p}{2}$?

Rounding Errors and Error Propagation

Rounding Rules

Let $x = \pm \sum_{i=1}^{\infty} \frac{\alpha_i}{p^i} \cdot p^b \in Z$ with $\alpha_1 \neq 0$. By truncating to t significant digits we get

$$\gamma(x) = \begin{cases} \pm \sum_{i=1}^{t} \frac{\alpha_i}{p^i} \cdot p^b, & \alpha_{t+1} < \frac{p}{2}, \\ \pm \left(\sum_{i=1}^{t} \frac{\alpha_i}{p^i} + \frac{1}{p^t} \right) \cdot p^b, & \alpha_{t+1} > \frac{p}{2}. \end{cases}$$

$\rightarrow$ What happens if $\alpha_{t+1} = \frac{p}{2}$?

We need some additional rules.

Rounding Errors and Error Propagation

Rounding Rules

Round up: Handle $\gamma(x)$ as if $\alpha_{t+1} > \frac{p}{2}$.

Round down: Handle $\gamma(x)$ as if $\alpha_{t+1} < \frac{p}{2}$.

Round-to-even: Rounds towards the closest machine number with an α_t that is even.

Rounding Errors and Error Propagation

Rounding Rules

Round up: Handle $\gamma(x)$ as if $\alpha_{t+1} > \frac{p}{2}$.

Round down: Handle $\gamma(x)$ as if $\alpha_{t+1} < \frac{p}{2}$.

Round-to-even: Rounds towards the closest machine number with an α_t that is even.

Example

For example for $p = 2$, $t = 3$:

$$\gamma(0.1001) = 0.100 \qquad \text{(round down)}$$
$$\gamma(0.1011) = 0.110 \qquad \text{(round up)}$$

$\rightarrow$ Round-to-even results in a (statistically) more equal distribution of rounding errors then using round-up or round-down permanently.

Overflows and Underflows

Rounding Errors and Error Propagation

Overflows and Underflows

We need to define $\gamma(x)$ for $x \notin Z$.

Two cases are possible:

1. $|x| < x_{\min}$: This case is called **underflow**.

   Either we round towards the closest valid machine number:

$$\gamma(x) = \begin{cases} 0 & \text{or rather} \\ \operatorname{sign}(x)\, x_{\min} \end{cases}$$

   or we use the so called **gradual underflow**. This allows **non-normalized** floating point numbers, i.e., floating point numbers allowing $\alpha_1 = 0$ to circumvent the underflow. The smallest number representable in this way is

$$0.\underbrace{0\ldots01}_{t} \cdot p^{e_{\min}}.$$

   In this case the same rounding rules as for $x \in Z$ are used.

Rounding Errors and Error Propagation

Overflows and Underflows

2. $|x| > x_{\max}$: This case is called **overflow**.

   Again, we have the two variants

$$\gamma(x) = \begin{cases} \operatorname{sign}(x)\, x_{\max} \\ \operatorname{sign}(x) \cdot \infty. \end{cases}$$

   $\rightarrow$ require to extend our definition of $\mathbb{M}(p, t, e_{\min}, e_{\max})$ by a symbol for ∞.

   This is used in the IEEE 754 standard for floating point arithmetic.

Rounding Errors

Rounding Errors and Error Propagation

Rounding Errors

Definition

Let $x \in \mathbb{R}$ and $\tilde{x} \in \mathbb{M}(p, t, e_{\min}, e_{\max})$, the we define the **absolute error** by

$$\|x - \tilde{x}\|$$

and the **relative error** by

$$\frac{\|x - \tilde{x}\|}{\|x\|}.$$

Rounding Errors and Error Propagation

Rounding Errors

Definition

Let $x \in \mathbb{R}$ and $\tilde{x} \in \mathbb{M}(p, t, e_{\min}, e_{\max})$, the we define the **absolute error** by

$$\|x - \tilde{x}\|$$

and the **relative error** by

$$\frac{\|x - \tilde{x}\|}{\|x\|}.$$

Lemma

The absolute rounding error fulfills

$$|\gamma(x) - x| \leqslant \frac{p^{-t}}{2} \cdot p^b \quad \forall x \in Z.$$

Rounding Errors and Error Propagation

Rounding Errors

Proof.

Let $x := \pm \sum_{i=1}^{\infty} \frac{\alpha_i}{p^i} p^b$ and define

$$y_1 := \operatorname{sign}(x) \sum_{i=1}^{t} \frac{\alpha_i}{p^i}\ p^b \qquad \text{(round towards zero)}$$

$$y_2 := \operatorname{sign}(x) \left(\sum_{i=1}^{t} \frac{\alpha_i}{p^i} + \frac{1}{p^t} \right) p^b \qquad \text{(round away from zero)}$$

Then apparently we have $\gamma(x) \in \{y_1, y_2\}$ and

$$x \in \begin{cases} [y_1, y_2], & x > 0, \\ [y_2, y_1], & x < 0. \end{cases}$$

Let $a_1 < a_2 \in \{y_1, y_2\}$, since $|x - a_j| \leqslant \frac{1}{2}|a_2 - a_1| = \frac{1}{2}|y_2 - y_1|$ either for $j = 1$, or for $j = 2$, or both, if $x \in [a_1, a_2]$, we find

$$|\gamma(x) - x| \leqslant \frac{1}{2}\ |y_2 - y_1| = \frac{1}{2} \frac{p^b}{p^t}.$$

□

Rounding Errors and Error Propagation

Rounding Errors

Lemma

Let $Z = [-x_{\max}, -x_{\min}] \cup \{0\} \cup [x_{\min}, x_{\max}]$, *as above. The relative rounding error for all* $x \in Z \backslash \{0\}$ *fulfills*

$$\frac{|\gamma(x) - x|}{|x|} < \frac{1}{2} p^{1-t}.$$

Rounding Errors and Error Propagation

Lemma

Let $Z = [-x_{\max}, -x_{\min}] \cup \{0\} \cup [x_{\min}, x_{\max}]$*, as above. The relative rounding error for all* $x \in Z \backslash \{0\}$ *fulfills*

$$\frac{|\gamma(x) - x|}{|x|} \;<\; \frac{1}{2} p^{1-t}.$$

Proof.

The significand a of x fulfills $|a| \geqslant \frac{1}{p}$. Thus we have $|x| \geqslant \frac{1}{p} \cdot p^b$. From the previous Lemma we, therefore, find

$$\frac{|\gamma(x) - x|}{|x|} \leqslant \frac{1}{p^{b-1}} \frac{1}{2}\; p^{b-t} = \frac{1}{2}\; p^{1-t}.$$

From $|x| > \frac{1}{p} p^b$ we have strict inequality unless $x = \pm \frac{1}{p} \cdot p^b$. In the latter case, however, $x \in \mathbb{M}(p, t, e_{\min}, e_{\max})$ and so $\gamma(x) = x$, i.e., $\frac{|\gamma(x)-x|}{|x|} = 0$. □

Rounding Errors and Error Propagation

Rounding Errors

Definition

The quantity $\mathrm{u} := \frac{1}{2} p^{1-t}$ is called **unit round off**.

Rounding Errors and Error Propagation

Rounding Errors

Definition

The quantity $\mathrm{u} := \frac{1}{2} p^{1-t}$ is called **unit round off**.

- The unit round off describes the relative error that can result from rounding operations.
- Similar quantity: The **machine epsilon** (eps):

$$\mathrm{eps} := \min\{|\tilde{x} - 1| \mid \tilde{x} \in \mathbb{M}(p, t, e_{\min}, e_{\max}), \tilde{x} > 1\} = p^{1-t} = 2\mathrm{u},$$

  which determines the distance from 1 to next larger machine number.

Rounding Errors and Error Propagation

Rounding Errors

The relative error gives a hint about the accuracy:

Example

$$x = 25.317,\ \tilde{x} = 25.313 \quad \text{(i.e., } \tilde{x} \text{ has 4 correct digits)}$$
$$\Longrightarrow \frac{|x - \tilde{x}|}{|x|} = \frac{0.004}{25.317} \approx 0.16 \cdot 10^{-3}.$$

$\rightarrow$ It is an easy argumentation to find that the number of correct digits coincides with the negative exponent of the relative error (± 1).

Rounding Errors and Error Propagation

Rounding Errors

The relative error gives a hint about the accuracy:

Example

$$x = 25.317,\ \tilde{x} = 25.313 \quad \text{(i.e., } \tilde{x} \text{ has 4 correct digits)}$$
$$\Longrightarrow \frac{|x - \tilde{x}|}{|x|} = \frac{0.004}{25.317} \approx 0.16 \cdot 10^{-3}.$$

$\rightarrow$ It is an easy argumentation to find that the number of correct digits coincides with the negative exponent of the relative error (± 1).

The absolute error does not carry any information about the accuracy!

Example

Let $y = 0.001, \tilde{y} = 0.002$: $|y - \tilde{y}| = 10^{-3}$ is rather small, but $\tilde{y}$ has no correct digit as we can see from the relative error

$$\frac{|y - \tilde{y}|}{|y|} = 1.$$

Rounding Errors and Error Propagation

Rounding Errors

The rounding behaviour can be influenced within a C program. [1]

The rounding function $\gamma(.)$ can be influenced using the functions

```
int fegetround(void);
int fesetround(int round);
```

from

```
#include <fenv.h>
```

Available rounding models, i.e. values for the `round` argument, are

- `FE_DOWNWARD`, (round down)
- `FE_UPWARD`, (round up)
- `FE_TONEAREST` (default, "natural rounding"),
- `FE_TOWARDZERO` (round up for $x < 0$ and round down for $x > 0$).

[1] see `man fenv`

Computer Arithmetic

How do the rounding errors (absolute and relative) evolve under elementary arithmetic operations ($+$, $-$, $\cdot$, $/$)?

Rounding Errors and Error Propagation

Computer Arithmetic

From the lemmas, we conclude:

$$\gamma(x) = x(1+\varepsilon), \quad |\varepsilon| \leqslant \mathsf{u} \quad \forall x \in Z.$$

Rounding Errors and Error Propagation

Computer Arithmetic

From the lemmas, we conclude:

$$\gamma(x) = x(1+\varepsilon), \quad |\varepsilon| \leqslant \mathsf{u} \quad \forall x \in Z.$$

Example

For $p = 2$ we have know $(0.1)_2 = 0.0\overline{0011}$. After normalization we have $0.11\overline{0011} \cdot 2^{-3}$. Restricting to six digits, i.e. $t = 6$, we get

$$(\gamma(0.1))_2 = 0.110011 \cdot 2^{-3}.$$

Regarding its decimal representation, we have $\gamma(0.1) = \frac{51}{512} = 0.099609375$.

$\rightarrow$ it a result of storing values into a finite amount of memory.

Rounding Errors and Error Propagation

Computer Arithmetic

Computers are only equipped with a so called **pseudo arithmetic**. We can not expect in general that the result of $x \triangle y$ for $\triangle \in \{+, -, \cdot, /\}$ and machine numbers $x, y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$ will also be a number in $\mathbb{M}(p, t, e_{\min}, e_{\max})$.

Rounding Errors and Error Propagation

Computer Arithmetic

Computers are only equipped with a so called **pseudo arithmetic**. We can not expect in general that the result of $x \triangle y$ for $\triangle \in \{+, -, \cdot, /\}$ and machine numbers $x, y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$ will also be a number in $\mathbb{M}(p, t, e_{\min}, e_{\max})$.

Example

Both $x = 0.12$ and $y = 0.34$ are from the set of machine numbers $\mathbb{M}(10, 2, e_{\min}, e_{\max})$, but for their product we easily see

$$x \cdot y = 0.0408 = 0.408 \cdot 10^{-1},$$

which requires a 3 digit mantissa and thus is not in $\mathbb{M}(10, 2, e_{\min}, e_{\max})$.

$\rightarrow$ Representing 0.408 in $\mathbb{M}(10, 2, e_{\min}, e_{\max})$ requires rounding.

Definition

Let $x \triangle y$ be an operation on $x, y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$, $\triangle \in \{+, -, \cdot, /\}$. Then the result of the **floating point operation**, i.e., the result of a calculation $x \triangle y$ in a system of machine numbers by $x \circledcirc y$, is determined by

$$x \circledcirc y = \gamma(x \triangle y), \quad \triangle \in \{+, -, \cdot, /\}. \tag{6}$$

Rounding Errors and Error Propagation

Definition

Let $x \triangle y$ be an operation on $x, y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$, $\triangle \in \{+, -, \cdot, /\}$. Then the result of the **floating point operation**, i.e., the result of a calculation $x \triangle y$ in a system of machine numbers by $x \circledcirc y$, is determined by

$$x \circledcirc y = \gamma(x \triangle y), \quad \triangle \in \{+, -, \cdot, /\}. \tag{6}$$

Standard Model of the Floating Point Arithmetic

For all floating point numbers x, $y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$ and any arithmetic operation $\triangle \in \{+, -, \cdot, /\}$ it holds:

$$x \circledcirc y = (x \triangle y)(1 + \delta), \quad \text{for a } |\delta| \leqslant \mathrm{u}. \tag{7}$$

Rounding Errors and Error Propagation

Definition

Let $x \triangle y$ be an operation on $x, y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$, $\triangle \in \{+, -, \cdot, /\}$. Then the result of the **floating point operation**, i.e., the result of a calculation $x \triangle y$ in a system of machine numbers by $x \circledcirc y$, is determined by

$$x \circledcirc y = \gamma(x \triangle y), \quad \triangle \in \{+, -, \cdot, /\}. \tag{6}$$

Standard Model of the Floating Point Arithmetic

For all floating point numbers x, $y \in \mathbb{M}(p, t, e_{\min}, e_{\max})$ and any arithmetic operation $\triangle \in \{+, -, \cdot, /\}$ it holds:

$$x \circledcirc y = (x \triangle y)(1 + \delta), \quad \text{for a } |\delta| \leqslant \mathrm{u}. \tag{7}$$

We always assume for $\sqrt{x}$ that $\gamma(\sqrt{x}) = \sqrt{x}(1 + \delta)$ for a $\delta \in \mathbb{R}$ with $|\delta| \leqslant \mathrm{u}$

Error Propagation

Main question: How does the Standard Model of Floating Point Arithmetic influences more complex operation? Especially how does δ propagate through a set of operations?

Rounding Errors and Error Propagation

Error Propagation – Addition

Let $x, y \in \mathbb{R}\backslash\{0\}$, $\mathrm{sign}(x) = \mathrm{sign}(y)$ and

$$\begin{aligned} \tilde{x} := \gamma(x) &= x(1+\delta_x), & |\delta_x| &\leqslant \mathsf{u}, \\ \tilde{y} := \gamma(y) &= y(1+\delta_y), & |\delta_y| &\leqslant \mathsf{u}. \end{aligned}$$

Then we have

$$\begin{aligned} \tilde{x} \oplus \tilde{y} &= (\tilde{x} + \tilde{y})(1+\delta_{x+y}) \qquad (\text{where } |\delta_{x+y}| \leqslant \mathsf{u}) \\ &= (x(1+\delta_x) + y(1+\delta_y))(1+\delta_{x+y}) \\ &= ((x+y) + (x\delta_x + y\delta_y))(1+\delta_{x+y}) \end{aligned}$$

and

$$\begin{aligned} |\tilde{x} \oplus \tilde{y} - (x+y)| &= |(x+y)\delta_{x+y} + (x\delta_x + y\delta_y)(1+\delta_{x+y})| \\ &\leqslant |x+y|\mathsf{u} + (|x| \cdot \mathsf{u} + |y| \cdot \mathsf{u})(1+\mathsf{u}) \\ &= |x+y|\mathsf{u} + |x+y|\mathsf{u}(1+\mathsf{u}) \qquad (\text{using } \mathrm{sign}(x) = \mathrm{sign}(y)) \\ &= |x+y|(2\mathsf{u} + \mathsf{u}^2). \end{aligned}$$

Rounding Errors and Error Propagation

Error Propagation – Addition

Thus we find

$$\frac{|(\tilde{x} \oplus \tilde{y}) - (x + y)|}{|x + y|} \leqslant 2\mathrm{u} + \mathrm{u}^2.$$

The relative error is (up to a negligible higher order term u^2) at most twice as large as the relative representation errors of the summands x and y.

Rounding Errors and Error Propagation

Error Propagation – Addition

Thus we find

$$\frac{|(\tilde{x} \oplus \tilde{y}) - (x + y)|}{|x + y|} \leqslant 2\mathrm{u} + \mathrm{u}^2.$$

The relative error is (up to a negligible higher order term u^2) at most twice as large as the relative representation errors of the summands x and y.

Many additions may lead to a noticeable accumulated error.

Rounding Errors and Error Propagation

Error Propagation – Subtraction

Similar to adding but with $\text{sign}(x) \neq \text{sign}(y)$. Instead of adding two numbers with different signs here we treat the subtraction of two numbers with a common sign.

Using the standard model we obtain

$$\begin{aligned}\tilde{x} \ominus \tilde{y} &= (\tilde{x} - \tilde{y})(1 + \delta_{x-y}) \qquad (\text{where } |\delta_{x-y}| \leqslant \mathsf{u}) \\ &= ((x - y) + (x\delta_x - y\delta_y))(1 + \delta_{x-y})\end{aligned}$$

It follows

$$\begin{aligned}|(\tilde{x} \ominus \tilde{y}) - (x - y)| &= |(x - y)\delta_{x-y} + (x\delta_x - y\delta_y)(1 + \delta_{x-y})| \\ &= |(x - y)\delta_{x-y} + (x\delta_x - y\delta_x + y\delta_x - y\delta_y)(1 + \delta_{x-y})| \\ &= |(x - y)\delta_{x-y} + (x - y)\delta_x + y(\delta_x - \delta_y) \\ &\qquad + (x - y)\delta_x\delta_{x-y} + y(\delta_x - \delta_y)\delta_{x-y}| \\ &\leqslant 2|x - y| \cdot \mathsf{u} + 2|y|\mathsf{u} + |x - y| \cdot \mathsf{u}^2 + 2|y|\mathsf{u}^2\end{aligned}$$

and

$$\frac{|(\tilde{x} \ominus \tilde{y}) - (x - y)|}{|x - y|} \leqslant \left(\frac{2|y|}{|x - y|} + 2\right)\mathsf{u} + \left(\frac{2|y|}{|x - y|} + 1\right)\mathsf{u}^2.$$

Rounding Errors and Error Propagation

Error Propagation – Subtraction

Thus for $x \approx y$ we have to expect an especially large relative error. This effect is called **cancellation**.

Rounding Errors and Error Propagation

Error Propagation – Subtraction

Thus for $x \approx y$ we have to expect an especially large relative error. This effect is called **cancellation**.

To avoid cancellation it is necessary to try and rewrite the expression in a way that avoids the subtraction of two almost equal numbers.

Rounding Errors and Error Propagation

Error Propagation – Subtraction

Example

Let $p = 10,\ t = 10,\ x = 1.2 \cdot 10^{-5} = 0.12 \cdot 10^{-4}$ and $y = f(x) = \frac{1-\cos(x)}{x^2}$.

The evaluation of f in x gives

$$\begin{array}{lcl} \cos(x) & = & 0.99999999992800 \cdot 10^{0} =: c \approx 1 \\ \Longrightarrow \tilde{c} & := & \gamma(c) = 0.9999999999 \\ \Longrightarrow \tilde{y} & = & (1 \ominus \tilde{c}) \oslash (x \odot x) = 10^{-10} \oslash (0.144 \cdot 10^{-9}) = 0.6944444444. \end{array}$$

But the correct result rounded to ten digits of accuracy, however, is

$$\gamma(f(x)) = 0.4999997300.$$

The evaluation of $1 \ominus \tilde{c}$ causes the error: The result here has only one correct digit, while the remaining information got lost, when rounding c. The subsequent subtraction is performed exact, but the error $1 \ominus \tilde{c}$ is amplified by a factor of 10^{10}.

$$1 \ominus \tilde{c} = 0.1\underline{000000000} \cdot 10^{-9}$$

$\uparrow$ information about these values is lost

Using the alternative formulation

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

which uses the identity $\cos x = 1 - 2\sin^2\left(\frac{x}{2}\right)$, one gets the much better result $\tilde{y} = 0.5$.

Rounding Errors and Error Propagation

Error Propagation – Multiplication

Let x, y, $\tilde{x}$, and $\tilde{y}$ give as before. With a $|\delta_{x\cdot y}| \leqslant \mathsf{u}$ we have

$$\begin{aligned}\tilde{x} \odot \tilde{y} &= \tilde{x}\tilde{y}(1+\delta_{x\cdot y}) \\ &= x(1+\delta_x)y(1+\delta_y)(1+\delta_{x\cdot y}) \\ &= xy(1+\delta_x)(1+\delta_y)(1+\delta_{x\cdot y}) \\ &= xy + xy(\delta_x+\delta_y+\delta_{x\cdot y}) + \mathcal{O}(\mathsf{u}^2).\end{aligned}$$

It follows:

$$\frac{|\tilde{x} \odot \tilde{y} - x \cdot y|}{|x \cdot y|} \leqslant 3\mathsf{u} + \mathcal{O}(\mathsf{u}^2).$$

→ Multiplications behave a bit worse than addition.

Rounding Errors and Error Propagation

Error Propagation – Multiplication

Let x, y, $\tilde{x}$, and $\tilde{y}$ give as before. With a $|\delta_{x\cdot y}| \leqslant \mathsf{u}$ we have

$$\begin{aligned}\tilde{x} \odot \tilde{y} &= \tilde{x}\tilde{y}(1+\delta_{x\cdot y}) \\ &= x(1+\delta_x)y(1+\delta_y)(1+\delta_{x\cdot y}) \\ &= xy(1+\delta_x)(1+\delta_y)(1+\delta_{x\cdot y}) \\ &= xy + xy(\delta_x+\delta_y+\delta_{x\cdot y}) + \mathcal{O}(\mathsf{u}^2).\end{aligned}$$

It follows:

$$\frac{|\tilde{x} \odot \tilde{y} - x \cdot y|}{|x \cdot y|} \leqslant 3\mathsf{u} + \mathcal{O}(\mathsf{u}^2).$$

→ Multiplications behave a bit worse than addition.

Division:

- similar to multiplication
- division by very small numbers should be avoided to avoid cancellation
- but: in contrast to substraction only the absolute error is affected

Rounding Errors and Error Propagation

Error Propagation

Regarding the previous error estimates, we end up with the following problem:

Rounding Errors and Error Propagation

Error Propagation

Regarding the previous error estimates, we end up with the following problem:

Computer arithmetic is neither associative nor distributive.

Rounding Errors and Error Propagation

Error Propagation

Regarding the previous error estimates, we end up with the following problem:

Computer arithmetic is neither associative nor distributive.

That means:

$$(x \mathbin{\textcircled{\triangle}} y) \mathbin{\textcircled{\triangle}} z \neq x \mathbin{\textcircled{\triangle}} (y \mathbin{\textcircled{\triangle}} z)$$
$$x \odot (y \oplus z) \neq (x \odot y) \oplus (x \odot z).$$

Rounding Errors and Error Propagation

Error Propagation

Example

Given $\mathbb{M}(10, 5, e_{\min}, e_{\max})$ and $a = 4.2832, b = 4.2821, c = 5.7632$, we want to evaluate the expression $d := (a - b) \cdot c$. In exact calculation we find:

$$d = (0.0011) \cdot 5.7632 = 0.00633952 \quad \Longrightarrow \quad \gamma(d) = 0.63395 \cdot 10^{-2}.$$

The relative error is

$$\frac{|d - \gamma(d)|}{|d|} \approx 0.3 \cdot 10^{-6}.$$

In pseudo arithmetic using $\mathbb{M}(10, 5, e_{\min}, e_{\max})$ we have two options:

(i) $(a \ominus b) \odot c = (0.11 \cdot 10^{-2}) \odot (0.57632 \cdot 10^{1}) = 0.63395 \cdot 10^{-2} = \gamma(d)$, which gives the correct rounded result.

(ii) $(a \odot c) \ominus (b \odot c) =: e \ominus f =: g$

$$e = a \odot c = \gamma(0.24684932824 \cdot 10^{2}) = 0.24685 \cdot 10^{2}$$

$$f = b \odot c = \gamma(0.2467859872 \cdot 10^{2}) = 0.24679 \cdot 10^{2}$$

$$\Longrightarrow g = e \ominus f = \gamma(0.00006 \cdot 10^{2}) = 0.6 \cdot 10^{-2}$$

$$\Longrightarrow \frac{|d - g|}{|d|} \approx 0.054,$$

The IEEE Standard 754

The IEEE Standard 754

Error Propagation

Until mid of the 1980s each CPU vendor has its own machine number formats:

Computer	p	t	$e_{\min}$	$e_{\max}$
Univac 1108	2	27	-128	127
PDP – 11	2	24	-128	127
Cray – 1	2	48	-16384	16383
HP – 45	10	10	-98	100
TI – SR5x	10	12	-98	100
IBM System/360	16	6	-64	63

The IEEE Standard 754

Error Propagation

Until mid of the 1980s each CPU vendor has its own machine number formats:

Computer	p	t	$e_{\min}$	$e_{\max}$
Univac 1108	2	27	-128	127
PDP – 11	2	24	-128	127
Cray – 1	2	48	-16384	16383
HP – 45	10	10	-98	100
TI – SR5x	10	12	-98	100
IBM System/360	16	6	-64	63

$\rightarrow$ manufacturers standardize the use of computer arithmetic to make results comparable. To this end, the IEEE Standard 754 was born in 1985.

IEEE 754-1985

The IEEE Standard 754

IEEE 754-1985

The standard prescribes that $\mathbb{M}$ should be closed under the operations $+$, $-$, $\cdot$, $/$, $\sqrt{\ }$. That means any of these operations has to lead to a result in $\mathbb{M}$. Further contributions of the standard are:

- rounding is performed as "round-to-even".
- the standard model for floating point arithmetic holds, i.e., the result of an elementary operation is behaving as if the exact result had been rounded.
- overflows result in $\gamma(x) = \pm\infty$.
- underflows with non-normalized numbers and gradual underflow.
- two data types `double` (8 byte, `fp64`) and `single` (4 byte, `fp32`), both using $p = 2$.
- Normalization forces $\alpha_1 = 1$, thus it is not stored. $\rightarrow$ an extra bit for the significand.
- The **`single`** data type has the following properties (**`double`** analogously):
  - An exponent $E = 255$ is used to encode the elements $\pm\infty$ or **`NaN`** (not-a-number) that are necessary to ensure closedness of $\mathbb{M}$.
  - The exponent b of the machine number is derived from E via $b = E - 127$, which saves another bit for the sign of the exponent.
  - $E = 0$ is used to encode subnormal numbers.

The IEEE Standard 754

IEEE 754-1985

`half`: (16 bit)

```
S EEEEE MMMMMMMMMM
0 1   5 6        15
```

`single`: (32 bit)

```
S EEEEEEEE MMMMMMMMMMMMMMMMMMMMMMM
0 1       8 9                     31
```

`double`: (64 bit)

```
S EEEEEEEEEEE MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
0 1          11 12                                                63
```

Figure: Storage patterns for **`half`**, **`single`** and **`double`** precision variables.

The IEEE Standard 754

IEEE 754-1985

Summarizing we get the representation

$$x = (-1)^S \cdot (1.\gamma_2 \dots \gamma_{24}) \cdot p^{E-127}.$$

for **single** precision and

$$x = (-1)^S \cdot (1.\gamma_2 \dots \gamma_{53}) \cdot p^{E-2047}.$$

for **double** precision. This slightly differs from Definition of the machine numbers.

For the minimal value $E = 1$ it follows in **single**

$$x_{\min} = 1.\underbrace{0 \dots 0}_{23} \cdot 2^{1-127} = 0.1 \cdot 2^{-125} \qquad \Longrightarrow \qquad e_{\min} = -125.$$

Further, we get

$$e_{\max} = 1 + (254 - 127) = 128.$$

The IEEE Standard 754

IEEE 754-1985

Numbers in IEEE-754 **`single`**:

`0 11111111 00000000000000000000000`	=	$+\infty$
`1 11111111 00000000000000000000000`	=	$-\infty$
`0 11111111 00000100000000000000000`	=	**NaN**
`1 11111111 00100010001001010101010`	=	**NaN**
`0 10000000 00000000000000000000000`	=	$+1.0 * 2^{128-127} = 2$
`0 10000001 10100000000000000000000`	=	$+1.101 * 2^{129-127} = 6.5$
`1 10000001 10100000000000000000000`	=	$-1.101 * 2^{129-127} = -6.5$
`0 00000001 00000000000000000000000`	=	$+1.0 * 2^{1-127} = 2^{-126} = x_{\text{min}}$
`0 00000000 10000000000000000000000`	=	$+0.1 * 2^{-126} = 2^{-127}$
`0 00000000 00000000000000000000001`	=	$+0.0\ldots01 * 2^{-126} = 2^{-149}$
	=	smallest representable number
`0 00000000 00000000000000000000000`	=	$+0$
`1 00000000 00000000000000000000000`	=	-0
`0 01111111 00000000000000000000000`	=	$1.0 * 2^{127-127} = 1.0$
`1 01111111 00000000000000000000000`	=	$-1.0 * 2^{127-127} = -1.0$

The IEEE Standard 754

IEEE 754-1985

Further assumptions:

- The value of a variable can be tested for **NaN** since this is the only "number" for which $x \neq x$ is true.
- Whenever an incorrect result or a number is not matching the machine number definition, this causes an **exception** and raise a flag**flag** in the CPU:

Flag	Example	Result
invalid	$0/0, \quad 0 \cdot \infty, \quad \sqrt{-1},$ $\infty/\infty, \quad +\infty + (-\infty)$	**NaN** ("not a number")
overflow	$x_{\max} * x_{\max}$	$\pm\infty$ usually denoted: $\pm$**Inf**
division by zero	$x/0$ for $x \neq 0$	$\pm\infty$
underflow	$x_{\min}/p^s$, $1 < s < t$	subnormal number
inexact	$\mathrm{rd}(x \circ y) \neq x \circ y$	correctly rounded result

Table: IEEE Standard 754, Exception Handling.

IEEE 754-2008

The IEEE Standard 754

IEEE 754-2008

The revised edition of the standard serves a multitude of purposes:

- It merges IEEE 754–1985 with IEEE 845 (a standard defining decimal floating point numbers important in finance).
- It reduces the possible implementation alternatives, as well as ambiguous formulations.
- It adds two additional $p = 2$-based precision levels for **`half`** (also known as **`fp16`**) (2 byte) and **`quad`**ruple (16 byte) precision.
- It extends **`min`** and **`max`** for the special cases ± 0 and $\pm\infty$.
- The formerly *denormalized* numbers for *gradual underflow* treatment are now consistently called *subnormal* numbers. (non-normalized numbers)
- Also, a combined multiplication and addition operation called fused multiply add and performing $a \leftarrow a \pm (b \times c)$ was added to the set of basic operations fulfilling the standard model for floating point arithmetic.

The IEEE Standard 754

IEEE 754-2008

precision	p	t	$e_{\min}$	$e_{\max}$	u	$x_{\min}$	$x_{\max}$
half	2	$10+1$	-13	16	$\approx 4.88 \cdot 10^{-4}$	$\approx 6 \cdot 10^{-5}$	$\approx 1 \cdot 10^{5}$
single	2	$23+1$	-125	128	$\approx 5.96 \cdot 10^{-8}$	$\approx 1 \cdot 10^{-38}$	$\approx 3 \cdot 10^{38}$
double	2	$52+1$	$-1\,021$	$1\,024$	$\approx 1.11 \cdot 10^{-16}$	$\approx 10^{-308}$	$\approx 10^{308}$
quad	2	$112+1$	$-16\,381$	$16\,384$	$\approx 9.63 \cdot 10^{-35}$	$\approx 10^{-4\,932}$	$\approx 10^{4\,932}$

Table: IEEE standard 754-2008, data types.

An alternate 16-bit floating point format

The IEEE Standard 754

An alternate 16-bit floating point format

The **half** format has a very limited number range due to small representable exponents.

Google Brain project created a 16-bit floating point format called **bfloat16** with key properties close to the **fp32** or **single** format:

- preserves the first 16 bits of fp32 layout,
- uses the same exponent range as fp32,
- reduces the significand to 7 bits (+1 bit from normalization),
- → inherits the good range of real numbers from single precision,
- → but has fewer numbers in that range due to the shorter significand,
- exception handling follows IEEE standards,
- conversion from fp32 to bfloat16 is easy, only requiring the copy of the first 16 bits
- is the most widely accepted quasi-standard among lower precision number formats.

```
S EEEEEEEE MMMMMMM
0 1      8 9    15
```

Figure: Storage pattern for **bfloat16** half precision variables.

Error Analysis

Error Analysis

A computed result is influenced by errors from different categories:

- **data errors:** input data are not known exactly, e.g., due to measurement inaccuracies.
- **rounding errors:** Errors resulting from the necessity to work with numbers from $\mathbb{M}(p, t, e_{\min}, e_{\max})$ instead of $\mathbb{R}$ and the evaluation of expressions with a finite significand.
- **methodological errors:** Methodological errors depend on different factors. On the one hand, the accuracy of the model underlying the computation plays a role. On the other hand, also the solution method applied to solve or evaluate the model has a crucial contribution to this type of error.

Error Analysis

A computed result is influenced by errors from different categories:

- **data errors:** input data are not known exactly, e.g., due to measurement inaccuracies.
- **rounding errors:** Errors resulting from the necessity to work with numbers from $\mathbb{M}(p, t, e_{\min}, e_{\max})$ instead of $\mathbb{R}$ and the evaluation of expressions with a finite significand.
- **methodological errors:** Methodological errors depend on different factors. On the one hand, the accuracy of the model underlying the computation plays a role. On the other hand, also the solution method applied to solve or evaluate the model has a crucial contribution to this type of error.

The methodological error in any case strictly depends on the task at hand and the way it is solved.

Conditioning / Condition Number / Stability

Error Analysis

Conditioning / Condition Number / Stability

$\rightarrow$ a property of the mathematical problem only

Example

We consider to affine linear functions

$$f_1(x) = a_1 x + b_1 \quad \text{and} \quad f_2(x) = a_2 x + b_2,$$

where $a_1 = 10^{-5}$ and $a_2 = 10^5$. b_1 and b_2 are determined such that $f_1(x^*) = f_2(x^*) = 0$. Some algorithm computing the root $\tilde{x}^*$ of f_1 (or f_2) stops once $|f_1(\tilde{x}^*)| \leqslant \delta$ (or $|f_2(\tilde{x}^*)| \leqslant \delta$), with $\delta = 10^{-3}$ is fulfilled. Now, we have for $\tilde{x}^*$:

$$f_1 : \tilde{x}^* \in [x^* - 100, x^* + 100] \text{ and } f_2 : \tilde{x}^* \in \left[x^* - 10^{-8}, x^* + 10^{-8}\right]$$

Error Analysis

$\rightarrow$ a property of the mathematical problem only

Example

We consider to affine linear functions

$$f_1(x) = a_1 x + b_1 \quad \text{and} \quad f_2(x) = a_2 x + b_2,$$

where $a_1 = 10^{-5}$ and $a_2 = 10^5$. b_1 and b_2 are determined such that $f_1(x^*) = f_2(x^*) = 0$. Some algorithm computing the root $\tilde{x}^*$ of f_1 (or f_2) stops once $|f_1(\tilde{x}^*)| \leqslant \delta$ (or $|f_2(\tilde{x}^*)| \leqslant \delta$), with $\delta = 10^{-3}$ is fulfilled. Now, we have for $\tilde{x}^*$:

$$f_1 : \tilde{x}^* \in [x^* - 100, x^* + 100] \text{ and } f_2 : \tilde{x}^* \in \left[x^* - 10^{-8}, x^* + 10^{-8}\right]$$

- f_1 corresponds to a badly conditioned problem
- f_2 corresponds to a well conditioned problem

Error Analysis

Conditioning / Condition Number / Stability

We consider the problem of evaluating the function $y = f(x)$, where $f : D \to V$ maps the data $x \in D$ onto the result $y \in V$.

Definition

Let Δx be the perturbation on the input data x. Then the evaluation of f computes

$$y + \Delta y = f(x + \Delta x),$$

where Δy covers the error of the perturbed computation. Then the **condition number** $c(f, x)$ is given by the bound on the relative error:

$$\frac{\|\Delta y\|}{\|y\|} \leqslant c(f, x) \cdot \frac{\|\Delta x\|}{\|x\|}.$$

Error Analysis

Conditioning / Condition Number / Stability

We consider the problem of evaluating the function $y = f(x)$, where $f : D \to V$ maps the data $x \in D$ onto the result $y \in V$.

Definition

Let Δx be the perturbation on the input data x. Then the evaluation of f computes

$$y + \Delta y = f(x + \Delta x),$$

where Δy covers the error of the perturbed computation. Then the **condition number** $c(f, x)$ is given by the bound on the relative error:

$$\frac{\|\Delta y\|}{\|y\|} \leqslant c(f, x) \cdot \frac{\|\Delta x\|}{\|x\|}.$$

$\rightarrow$ The condition number gives information about how an error in the input could be amplified by evaluating a function $f(x)$.

Error Analysis

Conditioning / Condition Number / Stability

We consider the problem of evaluating the function $y = f(x)$, where $f : D \to V$ maps the data $x \in D$ onto the result $y \in V$.

Definition

Let Δx be the perturbation on the input data x. Then the evaluation of f computes

$$y + \Delta y = f(x + \Delta x),$$

where Δy covers the error of the perturbed computation. Then the **condition number** $c(f, x)$ is given by the bound on the relative error:

$$\frac{\|\Delta y\|}{\|y\|} \leqslant c(f, x) \cdot \frac{\|\Delta x\|}{\|x\|}.$$

$\rightarrow$ The condition number gives information about how an error in the input could be amplified by evaluating a function $f(x)$.

$\rightarrow$ The corresponding property for an algorithm is called **stability**.

Forward Error Analysis

Error Analysis

Forward Error Analysis

Notation:

- $x \in D$ are the data for the problem,
- $f : D \to V$ is the mathematical problem mapping data to values,
- and $y = f(x) \in V$ is the exact result, whereas
- $\hat{y}$ is the numerically computed result (e.g. with the help of an algorithm).

Error Analysis

Forward Error Analysis

Notation:

- $x \in D$ are the data for the problem,
- $f : D \to V$ is the mathematical problem mapping data to values,
- and $y = f(x) \in V$ is the exact result, whereas
- $\hat{y}$ is the numerically computed result (e.g. with the help of an algorithm).

Obvious question: How do y and $\hat{y}$ relate to each other?

$$\|y - \hat{y}\| = ?, \qquad \frac{\|y - \hat{y}\|}{\|y\|} = ?$$

Error Analysis

Forward Error Analysis

Notation:

- $x \in D$ are the data for the problem,
- $f : D \to V$ is the mathematical problem mapping data to values,
- and $y = f(x) \in V$ is the exact result, whereas
- $\hat{y}$ is the numerically computed result (e.g. with the help of an algorithm).

Obvious question: How do y and $\hat{y}$ relate to each other?

$$\|y - \hat{y}\| = ?, \qquad \frac{\|y - \hat{y}\|}{\|y\|} = ?$$

→ **Forward error analysis** performs a step by step analysis of the propagation and the accumulation of the rounding errors.

Error Analysis

Forward Error Analysis

Example

Let the mathematical problem be that of solving the simple quadratic equation $y^2 - 2ay + b = 0$, for given $a, b \in \mathbb{M}(p, t, e_{\min}, e_{\max})$. The two solutions are known to be

$$y_1 = a - \sqrt{a^2 - b}, \quad \text{and} \quad y_2 = a + \sqrt{a^2 - b}.$$

We concentrate on the computation of y_1. Exactly following the solution formula above is giving the below algorithm in exact and finite arithmetic (following the standard model for floating point arithmetic):

	exact computation		numerical realization
1.	$c := a \cdot a$	$\Longrightarrow$	$\hat{c} = a^2(1 + \delta_1)$
2.	$d := c - b$	$\Longrightarrow$	$\hat{d} = (\hat{c} - b)(1 + \delta_2)$
3.	$e := \sqrt{d}$	$\Longrightarrow$	$\hat{e} = \sqrt{\hat{d}}(1 + \delta_3)$
4.	$y_1 := a - e$	$\Longrightarrow$	$\hat{y}_1 = (a - \hat{e})(1 + \delta_4)$

Here we have $|\delta_i| \leqslant \mathsf{u}$, $i = 1, \ldots, 4$ due to the standard model assumption.

Error Analysis

Forward Error Analysis

Example

Now inserting all computed quantities we find

$$
\begin{aligned}
\hat{y}_1 &= \left\{ a - \sqrt{(a^2(1+\delta_1) - b)(1+\delta_2)}(1+\delta_3) \right\}(1+\delta_4) \\
&= a(1+\delta_4) \\
&\quad - \Big\{ a^2 \underbrace{(1+\delta_1)(1+\delta_2)(1+\delta_3)^2(1+\delta_4)^2}_{\substack{= 1+\delta_1+\delta_2+2\delta_3+2\delta_4+\mathcal{O}(\mathrm{u}^2) \\ =: 1+\varepsilon_1, \quad |\varepsilon_1| \leqslant 6\mathrm{u}+\mathcal{O}(\mathrm{u}^2)}} - b \underbrace{(1+\delta_2)(1+\delta_3)^2(1+\delta_4)^2}_{\substack{1+\delta_2+2\delta_3+2\delta_4+\mathcal{O}(\mathrm{u}^2) \\ =: 1+\varepsilon_2, \quad |\varepsilon_2| \leqslant 5\mathrm{u}+\mathcal{O}(\mathrm{u}^2)}} \Big\}^{\frac{1}{2}} \\
&= a + a\delta_4 - \sqrt{(a^2 - b) + (a^2\varepsilon_1 - b\varepsilon_2)} \\
&= a + a\delta_4 - \sqrt{a^2 - b} - \frac{1}{2\sqrt{a^2 - b}}(a^2\varepsilon_1 - b\varepsilon_2) + \mathcal{O}(\mathrm{u}^2)
\end{aligned}
$$

Error Analysis

Forward Error Analysis

Example

The last step exploits that using a Taylor expansion of $g(x) := \sqrt{x}$ at

$$x + \Delta x = \underbrace{a^2 - b}_{=:x} + \underbrace{a^2\varepsilon_1 - b\varepsilon_2}_{=:\Delta x},$$

we get

$$g(x + \Delta x) = \sqrt{x + \Delta x} = \sqrt{x} + \frac{1}{2\sqrt{x}}\Delta x + \mathcal{O}((\Delta x)^2),$$

where $|\Delta x| \leqslant 6(|a^2| + |b|)\mathrm{u} = \mathcal{O}(\mathrm{u})$.

Error Analysis

Forward Error Analysis

Example

Using this knowledge for the numerical result it follows

$$\hat{y}_1 = y_1 - \frac{1}{2\sqrt{a^2-b}}(a^2\varepsilon_1 - b\varepsilon_2) + a\delta_4 + \mathcal{O}(\mathrm{u}^2)$$

and thus for the relative error we get

$$\frac{|\hat{y}_1 - y_1|}{|y_1|} = \frac{1}{|a-\sqrt{a^2-b}|} \cdot \frac{1}{2\sqrt{a^2-b}} \underbrace{\left|a^2\varepsilon_1 - b\varepsilon_2 + 2a\delta_4\sqrt{a^2-b}\right|}_{\leqslant a^2\cdot 6\mathrm{u} + \underbrace{|b|\cdot 5\mathrm{u}}_{<|b|\cdot 6\mathrm{u}} + |a|\sqrt{a^2-b}\cdot 2\mathrm{u}} + \mathcal{O}(\mathrm{u}^2)$$

$$\leqslant 3\frac{a^2 + |b| + |a|\sqrt{a^2-b}}{\sqrt{a^2-b}\cdot|a-\sqrt{a^2-b}|}\mathrm{u} + \mathcal{O}(\mathrm{u}^2)$$

The forward error may be large if the denominator is small. This can happen in two cases that can both be traced back to cancellation happening in the computation of y_1.

(i) $a^2 \approx b \implies$ cancellation in 2. $d := a^2 - b$,
(ii) $|b| \ll a^2 \wedge a > 0 \implies$ cancellation in 4. $y_1 = a - e$.

Backward Error Analysis

Error Analysis

Backward Error Analysis

Given the result of the computation $\hat{y}$ — can we express $\hat{y}$ as the exact solution of a mathematical problem for slightly perturbed data? That means:

Does there exist a Δx, such that $\hat{y} = f(x + \Delta x)$?

Error Analysis

Backward Error Analysis

Given the result of the computation $\hat{y}$ — can we express $\hat{y}$ as the exact solution of a mathematical problem for slightly perturbed data? That means:

$$\textbf{Does there exist a } \Delta x\textbf{, such that } \hat{y} = f(x + \Delta x)\textbf{?}$$

Asking this question makes sense, since for inaccurate data x we only know the correct value up to, e.g., measurement errors. If the analysis for $\hat{y} = f(x + \Delta x)$ now provides a Δx that is of the magnitude of the data errors (i.e., measurement inaccuracies), then the computation result is as good as we can expect. An answer to the above question is derived by a so called **backward error analysis**.

Error Analysis

Backward Error Analysis

Definition

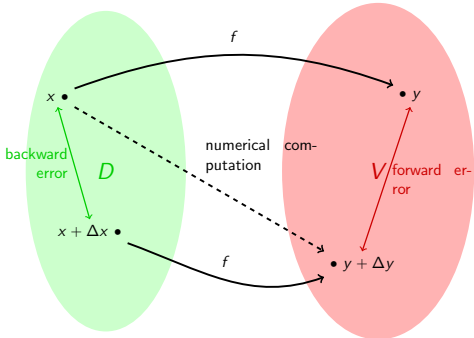
$\eta := \inf\{\|\Delta x\|; \hat{y} = f(x + \Delta x)\}$ is the **(absolute) backward error** of $\hat{y}$,
$\eta_{\text{rel}} := \eta / \|x\|$ is called the **relative backward error**, where $\|\,.\,\|$ is a suitable norm in the set of data D.

Error Analysis

Backward Error Analysis

Definition

$\eta := \inf\{\|\Delta x\|; \hat{y} = f(x + \Delta x)\}$ is the **(absolute) backward error** of $\hat{y}$,
$\eta_{\text{rel}} := \eta / \|x\|$ is called the **relative backward error**, where $\|\,.\,\|$ is a suitable norm in the set of data D.



Error Analysis

Backward Error Analysis

Definition

If for any $x \in D$ a method for computing $y = f(x)$ produces a $\hat{y} = f(x + \Delta x)$ for a small relative backward error $\left|\frac{\Delta x}{x}\right|$, then the method is said to be **(numerically) backward stable**. The concrete definition of small depends on the problem, but might, e.g., mean Δx is of the size of the unavoidable data errors.

On the other hand, a method is called **(numerically) forward stable** if it produces a relative forward error $\left|\frac{\Delta y}{y}\right|$ of the same magnitude that a backward stable method would.

Remark

- A forward stable method does not necessarily need to be backward stable to meet the definition.
- The definition primarily conveys that a forward stable algorithm produces an error approximately proportional to the data error, influenced by the condition number.
- Even if the backward error of the computed solution is small, this error can be amplified by a factor as large as the condition number when transitioning to the forward error in a forward stable method.

Remark

- A forward stable method does not necessarily need to be backward stable to meet the definition.
- The definition primarily conveys that a forward stable algorithm produces an error approximately proportional to the data error, influenced by the condition number.
- Even if the backward error of the computed solution is small, this error can be amplified by a factor as large as the condition number when transitioning to the forward error in a forward stable method.

We always have:

$$\textbf{backward stable} \Rightarrow \textbf{forward stable}$$

The opposite implication does, however, in general not hold.

Error Analysis

Backward Error Analysis

The verification of **backward stability** involves a **backward error analysis**, where the computed result $\hat{y}$ is considered as the exact computation for perturbed data. This perturbed data is then compared to the original data.

Example

Consider $y_1 = a - \sqrt{a^2 - b}$ and $\hat{y}_1$ the corresponding solution of the quadratic equation for perturbed data a and b

$$y^2 - 2(a + \Delta a)y + (b + \Delta b) = 0$$

To this end, we require an expression of the form

$$\hat{y}_1 = (a + \Delta a) - \sqrt{(a + \Delta a)^2 - (b + \Delta b)}.$$

Error Analysis

Backward Error Analysis

Example

As for the forward error analysis we get

$$\hat{y}_1 = a(1+\delta_4)$$

$$- \Big\{ a^2 \underbrace{(1+\delta_1)(1+\delta_2)(1+\delta_3)^2}_{\substack{=1+\delta_1+\delta_2+2\delta_3+\mathcal{O}(u^2) \\ =:1+\varepsilon_1, \quad |\varepsilon_1| \leqslant 4u+\mathcal{O}(u^2)}} (1+\delta_4)^2 - b \underbrace{(1+\delta_1)(1+\delta_3)^2(1+\delta_4)^2}_{=:1+\varepsilon_2, \quad |\varepsilon_2| \leqslant 5u+\mathcal{O}(u^2)} \Big\}^{\frac{1}{2}}$$

$$= a + a\delta_4 - \Big\{ (a+a\delta_4)^2 - b \Big(\underbrace{1+\varepsilon_2 - \frac{a^2}{b}\varepsilon_1(1+\delta_4)^2}_{\substack{=1+\varepsilon_2-\frac{a^2}{b}\varepsilon_1+\mathcal{O}(u^2) \\ =:1+\delta_b, \ |\delta_b| \leqslant 5u+\frac{4a^2}{|b|}u+\mathcal{O}(u^2)}} \Big) \Big\}^{\frac{1}{2}}$$

$$= (a+a\delta_4) - \sqrt{(a+a\delta_4)^2 - (b+b\delta_b)}$$

Error Analysis

Backward Error Analysis

Example

Now defining $\Delta a := a\delta_4$, $\Delta b := b\delta_b$ we can estimate the relative backward error as

$$\begin{array}{rcll} \dfrac{|\eta_a|}{|a|} & \leqslant & \dfrac{|\Delta a|}{|a|} & \leqslant |\delta_4| \leqslant \mathrm{u}, \\ \dfrac{|\eta_b|}{|b|} & \leqslant & |\delta_b| & \leqslant \underbrace{\left(5 + \dfrac{4a^2}{|b|}\right)}_{\text{amplification factor}} \mathrm{u} + \mathcal{O}(\mathrm{u}^2). \end{array}$$

Note that the relative error is the infimum over all possible errors $\Delta x = \Delta\,[a,\ b]$. A small backward error, as we would expect it from a numerically backward stable algorithm, is derived if $a^2 \approx |b|$. The error may get large in case $a^2 \gg b$.

Remark

The separate consideration of the backward errors in a and b is called **component-wise error analysis**. For a **norm-wise** consideration one tries to estimate $\frac{1}{\left\|\left[\frac{a}{b}\right]\right\|_2}\eta$.

Perturbation Analysis

Error Analysis

Perturbation Analysis

Determine whether the problematic error amplification is:

- problem immanent,
- caused by the specific algorithmic approach chosen for solving the problem.

$\rightarrow$ reformulate the algorithm to avoid the second problem.

Error Analysis

Perturbation Analysis

Determine whether the problematic error amplification is:

- problem immanent,
- caused by the specific algorithmic approach chosen for solving the problem.

→ reformulate the algorithm to avoid the second problem.

→ Use perturbation analysis to find the condition number of the problem.

Error Analysis

Perturbation Analysis

Determine whether the problematic error amplification is:

- problem immanent,
- caused by the specific algorithmic approach chosen for solving the problem.

→ reformulate the algorithm to avoid the second problem.

→ Use perturbation analysis to find the condition number of the problem.

Again, let

$$f : D \to V, \quad f \in \mathcal{C}^2(D), \qquad y = f(x), \quad \hat{y} = f(x + \Delta x),$$

where $\mathcal{C}^2(D)$ is the set of two times differentiable functions on D.

Error Analysis

Perturbation Analysis

Question: How does the disturbanced of the data Δx influence the computed result $\hat{y}$?

Obviously, the large the slope of tangent of f in x the more $\hat{y}$ deviates from y when the input is perturbed.

Error Analysis

Perturbation Analysis

Question: How does the disturbanced of the data Δx influence the computed result $\hat{y}$?

Obviously, the large the slope of tangent of f in x the more $\hat{y}$ deviates from y when the input is perturbed.

Mathematically speaking, we have

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + o(\Delta x),$$

with $g \in o(\Delta x)$ means that $\lim_{\Delta x \to 0} \frac{g(\Delta x)}{\Delta x} = 0$. This yields:

$$\begin{aligned} \hat{y} - y &= f(x + \Delta x) - f(x) \\ &= f(x) + f'(x)\Delta x + o(\Delta x) - f(x) \\ &= f'(x) \cdot \Delta x + o(\Delta x) \approx f'(x) \cdot \Delta x. \end{aligned}$$

$\rightarrow$ the factor $|f'(x)|$ amplifies the data errors in the result $\hat{y}$
$\rightarrow$ **asymptotic** or **local perturbation analysis** since it focuses on a local neighborhood of x

Error Analysis

Perturbation Analysis

As seen before, the relative error is the more important one, we obtain:

$$\begin{aligned}\frac{\hat{y}-y}{y} &= \frac{f'(x)\Delta x}{y} + o(\Delta x) \\ &= \frac{f'(x)\cdot x}{f(x)} \cdot \frac{\Delta x}{x} + o(\Delta x)\end{aligned}$$

and thus

$$\frac{|\hat{y}-y|}{|y|} = \underbrace{\frac{\left|f'(x)\cdot x\right|}{|f(x)|}}_{=:c(f,x)} \cdot \frac{|\Delta x|}{|x|} + o(|\Delta x|). \tag{8}$$

Error Analysis

Perturbation Analysis

Definition

Let $f \in \mathcal{C}(D)$, $x, x + \Delta x \in D$ and $f(x + \Delta x) = \hat{y}$. The infimum of all numbers $c_{\text{abs}}(f, x)$ for which

$$\|y - \hat{y}\| \leqslant c_{\text{abs}}(f, x) \, \|\Delta x\| + o(\|\Delta x\|)$$

holds, is called **(absolute) condition number of f in x.**
Analogously, the infimum of all numbers $c(f, x) = c_{\text{rel}}(f, x)$, such that

$$\frac{\|y - \hat{y}\|}{\|y\|} \leqslant c_{\text{rel}}(f, x) \frac{\|\Delta x\|}{\|x\|} + o\left(\frac{\|\Delta x\|}{\|x\|}\right)$$

is true, is denoted as **(relative) condition number of f in x.**

Error Analysis

Perturbation Analysis

If f is differentiable then in analogy to (8)

$$c_{\text{abs}}(f, x) = \left\| f'(x) \right\|$$

and

$$c(f, x) = c_{\text{rel}}(f, x) \leqslant \frac{\|x\|}{\|f(x)\|} \left\| f'(x) \right\|,$$

where f' is the Jacobi matrix of $f : D \to V$ in x and the norms have to be compatible.
Optimal: The norm for f' is induced from the norm of x. (↗ later in the lecture)

Error Analysis

Perturbation Analysis

Example

Error Analysis

Perturbation Analysis

We consider the quadratic equation again: Here we have $x = \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right] \in \mathbb{R}^2$ and

$$f(a,b) = a - \sqrt{a^2 - b}, \quad y = f(a,b), \quad \hat{y} = f(a + \Delta a, b + \Delta b).$$

Further, let us assume

$$\max\left\{ \frac{|\Delta a|}{|a|}, \frac{|\Delta b|}{|b|} \right\} \leqslant \varepsilon \ll 1.$$

For the evaluation of the Taylor expansion we require the partial derivatives of f with respect to the data a, b:

$$\begin{aligned} \frac{\partial f}{\partial a}(a,b) &= 1 - \frac{1}{2}(a^2 - b)^{-\frac{1}{2}} \cdot 2a = 1 - \frac{a}{\sqrt{a^2 - b}} = \frac{\sqrt{a^2 - b} - a}{\sqrt{a^2 - b}} \\ &= -\frac{f(a,b)}{\sqrt{a^2 - b}}, \\ \frac{\partial f}{\partial b}(a,b) &= \frac{1}{2} \cdot \frac{1}{\sqrt{a^2 - b}}. \end{aligned}$$

Error Analysis

Perturbation Analysis

Example

Further assuming that $a^2 > b > 0$ or $b < 0$, such that $\sqrt{a^2 - b} \in \mathbb{R}$, we find

$$\begin{aligned}\hat{y} - y &= f(a,b) + \frac{\partial f}{\partial a}(a,b) \cdot \Delta a + \frac{\partial f}{\partial b}(a,b) \cdot \Delta b + o(\varepsilon) - f(a,b) \\ &= -\frac{f(a,b)a}{\sqrt{a^2 - b}} \cdot \frac{\Delta a}{a} + \frac{1}{2} \cdot \frac{b}{\sqrt{a^2 - b}} \frac{\Delta b}{b} + o(\varepsilon)\end{aligned}$$

and thus

$$\frac{|\hat{y} - y|}{|y|} \leqslant \underbrace{\frac{|a|}{\sqrt{a^2 - b}}}_{=:c_a(f,a,b)} \cdot \frac{|\Delta a|}{|a|} + \underbrace{\frac{|b|}{2\sqrt{a^2 - b} \cdot |a - \sqrt{a^2 - b}|}}_{=:c_b(f,a,b)} \cdot \frac{|\Delta b|}{|b|} + o(\varepsilon) \tag{9}$$

$$\leqslant \frac{1}{\sqrt{a^2 - b}} \left(|a| + \frac{|b|}{2|a - \sqrt{a^2 - b}|} \right) \cdot \varepsilon + o(\varepsilon). \tag{10}$$

Error Analysis

Perturbation Analysis

Example

The inequality (9) here represents the component-wise perturbation analysis and (10) the norm-wise one. A norm-wise consideration also follows from the Cauchy-Schwarz-Inequality applied to

$$\hat{y} - y = (\nabla f(a, b))^{\mathsf{T}} \begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix} + o(\varepsilon),$$

such that

$$|\hat{y} - y| \leqslant \|\nabla f(a, b)\| \cdot \left\| \begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix} \right\| + o(\varepsilon).$$

Here, we are only interested in the (usually more precise) component wise consideration.

Error Analysis

Perturbation Analysis

Example

case 1: $a^2 \approx b$ For $a^2 \to b$ it follows $c_a(f, a, b) \to \infty$ and also $c_b(f, a, b) \to \infty$.
The problem thus is ill-conditioned, i.e., we can not expect "good" results. A large forward error is "unavoidable". The large forward errors in this case are therefore caused by the bad conditioning of the problem. This corresponds to the observation that the backward error is still small in this case.

case 2: $a^2 \gg b$ In this case $c_a(f, a, b) \approx 1$. The same can easily be seen for $c_b(f, a, b)$ when considering $\frac{b}{a^2} \to 0 \Leftrightarrow b \to 0$ and applying L'Hôpitals rule. That means, we find that the problem is well conditioned in this case.

$\to$ Since the method for computing y_1 is performing well in most cases and only misbehaves in the case where $a^2 \approx b$, we also call the method **conditionally stable**.

Conclusion

Error Analysis

Conclusion

1. $c(f, x)$ in general not only depends on the problem but also on the data supplied to it. A mathematical problem thus is not generally good or bad, but it depends on where in D we evaluate it.
2. Condition numbers can be categorized as follows:
   $c(f, x) \approx 1 \quad \Rightarrow \quad$ well conditioned.
   $c(f, x) \gg 1 \quad \Rightarrow \quad$ ill-conditioned.
   $c(f, x) \ll 1$ may be bad as well since we can easily "lose information" due to the large possible backward errors.

Error Analysis

Conclusion

3. An unstable algorithm can result from the decomposition of a (possibly well conditioned) mathematical problem into a concatenation of sub-tasks, i.e.,

$$f(x) = (g_k \circ g_{k-1} \circ \ldots \circ g_1)(x),$$

where one or more of the g_j are ill-conditioned. For example, if the g_j are elementary operations and one of them is suffering from cancellation, then the loss of information resulting from the cancellation may prevail the remaining computation.

4. The main property of the connection between forward error, backward error and condition number is sketched by the rough rule:

$$\text{forward error} \;\approx\; \text{condition number} \times \text{backward error}.$$

This again illustrates the implication

$$\text{backward stability} \Rightarrow \text{forward stable}$$

good conditioning & stable algorithm $\Longrightarrow$ reliable result.

bad conditioning or unstable algorithm $\Longrightarrow$ unsure result.

Error Analysis

Conclusion

Error Analysis

Conclusion

Error Analysis

Conclusion