Technische Universität Braunschweig

Institute for Numerical Analysis

Ramp Up Mathematics — Numerical Analysis Ramp Up for Data Science

Matthias Bollhöfer, SS 2024

Contents I

Contents II

Representation of real numbers

Theorem 0.1 (p–adic expansion)

Every real number $r \in \mathbb{R}$ can be written as

$$r = \pm(\underline{d_t}\, p^t + \underline{d_{t-1}}\, p^{t-1} + \underline{d_{t-2}}\, p^{t-2} + \cdots),$$

for some number $t \in \mathbb{Z}$, where $p \in \{2,\ 3,\ 4,\ \ldots\}$ is referred to as basis and $d_k \in \{0,\ \ldots,\ b-1\}$ are called digits.

Remark. typical bases are $p = 10$ (because human beings have ten fingers), $p = 2$ or $p = 16$ for computers

Conversion between different bases

Example 0.1

Consider $r = 18.5$ *and* $p = 2$. *Then we only have digits* 0 *and* 1.

$$
\begin{aligned}
r &= \\
&= \textit{dual (or binary) representation} \\
p &= 16,\ \textit{digits}\,(0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ \underbrace{A\ B\ C\ D\ E\ F}_{\textit{representing } 10,11,12,13,14,15}\ ).
\end{aligned}
$$

$$
\begin{aligned}
r &= \\
&= \textit{Hexadecimal representation}
\end{aligned}
$$

Computer Arithmetics

Conversion between different representations can be computed via

- dividing modulo p for the integer part and
- multiplying by p for the remainder.

$$r \neq 0 \Rightarrow r = \underbrace{\pm(\underline{z_t}\, p^{-1} + \underline{z_{t-1}}\, p^{-2} + \underline{z_{t-2}}\, p^{-3} + \cdots)}_{a} p^{t+1}, \text{ where } \frac{1}{p} \leqslant |a| < 1, \text{ since } z_t \neq 0.$$

Definition 0.1

The representation $r = a \cdot p^b$ is referred to as normalized floating point representation of $r \neq 0$ with respect to the basis p. a is called mantissa and b is the exponent of r.

$$\begin{aligned} a &= \pm(\underline{z_1}\, p^{-1} + \underline{z_2}\, p^{-2} + \underline{z_3}\, p^{-3} + \cdots) \text{ where } z_1 \neq 0 \\ b &= \pm(\underline{\beta_1}\, p^{e-1} + \underline{\beta_2}\, p^{e-2} + \cdots + \underline{\beta_e}\, p^0). \end{aligned}$$

- On a computer we only have a *fixed* number of n digits
- split representation into m digits for the mantissa, e digits for the exponent

$$\pm(z_1 p^{-1} + z_2 p^{-2} + \cdots + z_m p^{-m}) p^{\pm(\beta_1 p^{e-1} + \cdots + \beta_e p^0)}$$

coding scheme: $\pm z_1\ z_2 \cdots z_m\ \pm \beta_1 \cdots \beta_e$

Computer Arithmetics

Example 0.2

Consider

$$\begin{aligned} r_1 &= -1234.567890,\ r_2 = -0.01234567890, \\ p &= 10,\ m = 9,\ e = 2. \end{aligned}$$

Then the normalized floating point point representation is

$$\begin{aligned} r_1 &= \quad \widehat{=} \quad , \\ r_2 &= \quad \widehat{=} \end{aligned}$$

Definition 0.2

$\mathbb{M}(p, m, e)$ *denotes the set of machine numbers with basis p, m digits for the mantissa and e digits for the exponent (each of them w.r.t. the basis p).*

Unfortunately $\mathbb{M}(p, m, e)$ only contains a finite number of numbers, even these are sparsely distributed.

Computer Arithmetics

Example 0.3

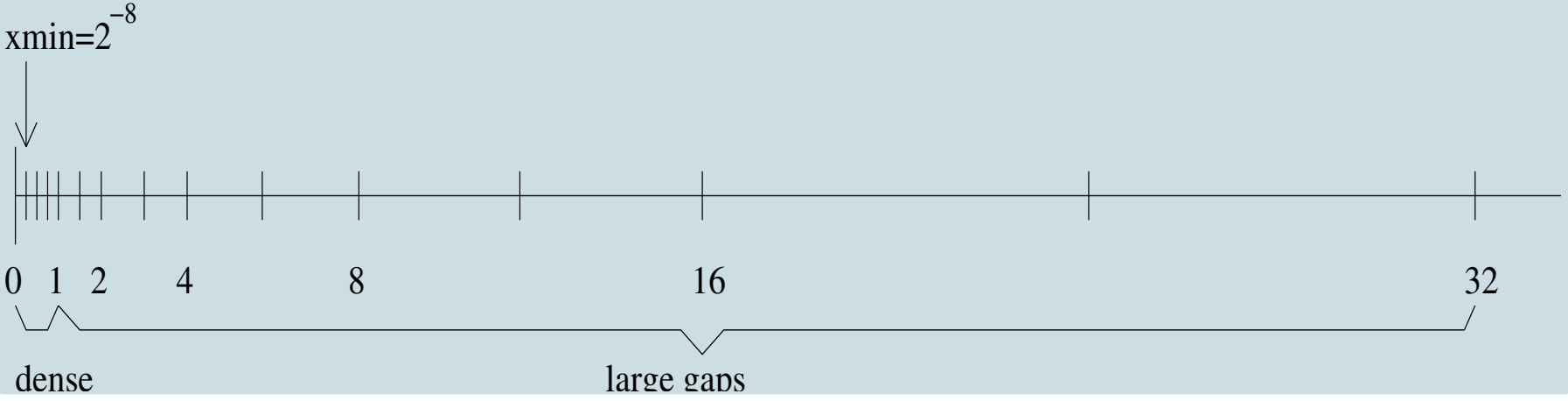
Consider $\mathbb{M}(2,3,3)$*, i.e.,* $p = 2$ *and* $m = e = 3$*.*

available mantissas			*available exponents*			
			± 000	$\widehat{=}$	± 0	$1, \quad 1$
± 100	$\widehat{=}$	$\pm \frac{1}{2}$	± 001	$\widehat{=}$	± 1	$2, \quad \frac{1}{2}$
			± 010	$\widehat{=}$	± 2	$4, \quad \frac{1}{4}$
± 101	$\widehat{=}$	$\pm \frac{5}{8}$	± 011	$\widehat{=}$	± 3	$8, \quad \frac{1}{8}$
			± 100	$\widehat{=}$	± 4	$16, \quad \frac{1}{16}$
± 110	$\widehat{=}$	$\pm \frac{3}{4}$	± 101	$\widehat{=}$	± 5	$32, \quad \frac{1}{32}$
			± 110	$\widehat{=}$	± 6	$64, \quad \frac{1}{64}$
± 111	$\widehat{=}$	$\pm \frac{7}{8}$	± 111	$\widehat{=}$	± 7	$128, \quad \frac{1}{128}$

Example 0.4 (continued)

smallest positive number $x_{\min} = \frac{1}{2}2^{-7} = 2^{-8}$, *largest positive number* $x_{\max} = \frac{7}{8}2^{7} = 112.$

The machine numbers are ordered as follows:



Example 0.5 (MATLAB)

$\gg$ realmin, realmax

Round off errors

from now on: either $p = 10$ or $p = 2^k$.
rounding numbers when projecting into the set of machine numbers:

$$fl : \mathbb{R} \longrightarrow \mathbb{M}(p, m, e).$$

Let $x_{\min} \leqslant |x| \leqslant x_{\max}$, $x = \pm(z_1 p^{-1} + z_2 p^{-2} + z_3 p^{-3} + \cdots)p^t$ where $z_1 \neq 0$.
traditional round off strategy to obtain m digits: *floating point number* $fl(x)$.

$$fl(x) = \pm p^t \cdot \begin{cases} (z_1 p^{-1} + z_2 p^{-2} + \cdots + z_m p^{-m}), & \text{if } z_{m+1} < \frac{p}{2}, \\ (z_1 p^{-1} + z_2 p^{-2} + \cdots + z_m p^{-m} + p^{-m}), & \text{if } z_{m+1} \geqslant \frac{p}{2}. \end{cases}$$

$$\begin{array}{lcll} |x| & < & x_{\min} & \left\langle \begin{array}{lcl} fl(x) & = & 0 \\ fl(x) & = & \text{sign}\,(x) x_{\min} \end{array} \right. \quad \text{'underflow'} \\ |x| & > & x_{\max} & \left\langle \begin{array}{lcl} fl(x) & = & \infty \\ fl(x) & = & \text{sign}\,(x) x_{\max} \end{array} \right. \quad \text{'overflow'} \end{array}$$

Example 0.6 (MATLAB)

$\gg \text{realmin}/2, \text{realmax} * 2, \text{inf}, \text{NaN}$

Round Off Errors

Proposition 0.1 (Absolute / Relative round off error)

Let $x_{\min} \leqslant |x| \leqslant x_{\max}$. *Then we have*

$$\begin{aligned} |fl(x) - x| &\leqslant \frac{p}{2}^{-m} p^t \text{ (absolute round off error)} \\ \frac{|fl(x) - x|}{|x|} &\leqslant \frac{p}{2} p^{-m} \text{ (relative round off error)} \end{aligned}$$

When dividing by $|x|$, the order of magnitude p^t of x disappears, since $|x| \geqslant \frac{1}{p} p^t$.

Definition 0.3

The number $\epsilon := \frac{p}{2} p^{-m}$ *is referred to as roundoff unit,* ***u*** *or macheps.*

Proposition 0.2

We always have $\epsilon = \min\{\delta \in [x_{\min}, x_{\max}] : \ fl(1+\delta) > 1\}$ *and*

$$fl(x) = x \cdot (1 + \varepsilon) \text{ such that } |\varepsilon| \leqslant \epsilon, \text{ where } \varepsilon = \frac{fl(x) - x}{x}.$$

Round Off Errors

Example 0.7 (MATLAB)

$\gg \text{eps}, 1 + \text{eps} > 1, 1 + \text{eps}/2 > 1$

Example 0.8

Consider 0.2 *as usual decimal number when using* $m = 6$ *and* $p = 2$*. This yields* $0.\overline{0011}$ *in dual representation, respectively, when being normalized* $0.110\overline{011} * 2^{-2}$*. Round off when using* $m = 6$ *digits will be* $0.110011 * 2^{-2}$*. Transformed back we obtain* $\frac{51}{256} = 0.19921875$*.*

Whenever an operation is performed, e.g., reading, writing, conversion and computing we are faced with round off errors.

On a computer we only have a *pseudo arithmetics*, since $\mathbb{M}$ is not a closed set w.r.t. $+, -, *, \div$.

Round Off Errors

Example 0.9

Consider $\mathbb{M}(10, 5, 1)$ *and the numbers*

$$\begin{aligned} x &= +25684 + 1 \;\widehat{=}\; 2.5684, \\ y &= +32791 - 2 \;\widehat{=}\; 0.0032791. \end{aligned}$$

Without *round off we obtain*

$$\begin{aligned} x + y &= \underbrace{2.5716}_{5}791 \notin \mathbb{M}, \\ x * y &= 0.00\underbrace{84220}_{5}4044 \notin \mathbb{M}, \\ x/y &= \underbrace{783.26}_{5}37004 \notin \mathbb{M}. \end{aligned}$$

Round Off Errors

- arithmetic operations in $\mathbb{R}$ have to be mapped to $\mathbb{M}$ using round off
- result of the *pseudo operation* $\textcircled{\nabla}$, $\nabla \in \{+, -, *, \div\}$

$$x \textcircled{\nabla} y := fl(x \nabla y), \text{ for } x, y \in \mathbb{M}.$$

- a single floating point operation is called flop
- flop is used as measure for complexity analyses
- internally, the CPU operates with a slightly longer mantissa, then numbers are normalized, finally round off

Be aware that there are exist computers where the computed result does not refer to the round off of the exact result

Proposition 0.3

If $x \textcircled{\nabla} y = fl(x \nabla y)$ *and* $x_{\min} \leqslant |x \nabla y| \leqslant x_{\max}$, *then we have for the relative error*

$$\left| \frac{x \textcircled{\nabla} y - x \nabla y}{x \nabla y} \right| = \left| \frac{fl(x \nabla y) - x \nabla y}{x \nabla y} \right| < \epsilon$$

Round Off Errors

For real numbers $\mathbb{R}$ we have valid associative, commutative and distributive laws, on a computer based on $\mathbb{M}$ usually *only* the commutativity applies.

Example 0.10

Consider $\mathbb{M}(10, 5, 1)$ *and check the laws for their applicability.*

1. *associativity*

$$
\begin{aligned}
0.98765 + 0.012424 - 0.0065432 &= 0.\underline{9935}308, \\
0.98765 \oplus (0.012424 \ominus 0.0065432) &= \\
0.98765 \oplus 0.0058808 &= \underbrace{0.\underline{99353}(08)}_{0.99353}, \\
(0.98765 \oplus 0.012424) \ominus 0.0065432 &= \\
\underbrace{1.000074}_{1.0001} \ominus 0.0065432 &= 0.\underline{99356}.
\end{aligned}
$$

Round Off Errors

Example 0.11 (continued)

2. *distributivity*

$$
\begin{aligned}
(4.2832 - 4.2821) * 5.7632 &= 0.00\underline{633952}, \\
(4.2832 \ominus 4.2821) \otimes 5.7632 &= \\
0.0011 \otimes 5.7632 &= 0.00\underline{63395}(2), \\
(4.2832 \otimes 5.7632) \ominus (4.2821 \otimes 5.7632) &= \\
\underbrace{24.684(93824) \ominus 24.678(59872)}_{24.685 \ominus 24.679} &= 0.00\underline{6}0000
\end{aligned}
$$

Mathematically equivalent algorithms can lead to completely different results on a computer.

- operations like $\sqrt{}$, sin, exp are implemented by software libraries, usually the computed result refers to the exact result after round off
- sometimes $\div$ is not implemented in straight forward fashion, but you may select either “quick and dirty” or “slow and accurate”

Error Propagation

- errors typically propagate in further computations
- What is happening when applying elementary operations $\circledcirc, \nabla \in \{*, \div, \pm\}$?

$$\tilde{x} = x(1 + \Delta x),\ \tilde{y} = y(1 + \Delta y),\ |\Delta x|, |\Delta y| \leqslant \epsilon$$

1. multiplication

$$\begin{aligned} \tilde{x} \otimes \tilde{y} &= \\ &= \\ &= \\ &\equiv \quad xy(1 + \delta) \\ |\delta| &\leqslant \end{aligned}$$

$\rightarrow$ multiplication

Error Propagation

$$\tilde{x} = x(1+\Delta x),\ \tilde{y} = y(1+\Delta y),\ |\Delta x|, |\Delta y| \leqslant \epsilon$$

2. division

$$\begin{aligned}
\tilde{x} \oslash \tilde{y} &= \\
&= \\
&= \\
&= \\
&\equiv \frac{x}{y}(1+\delta) \\
|\delta| &\leqslant
\end{aligned}$$

$\rightarrow$ division

Error Propagation

$$\tilde{x} = x(1+\Delta x),\ \tilde{y} = y(1+\Delta y),\ |\Delta x|, |\Delta y| \leqslant \epsilon$$

3. addition/subtraction

$$\begin{aligned}
\tilde{x} \oplus \tilde{y} &= \\
&= \\
&= \\
&= \\
&\equiv \quad (x \pm y)(1+\delta) \\
|\delta| &\leqslant \\
&= \\
\end{aligned}$$

Error Propagation

$$\tilde{x} = x(1+\Delta x),\ \tilde{y} = y(1+\Delta y),\ |\Delta x|, |\Delta y| \leqslant \epsilon$$

3. addition/subtraction [continued]

$$\delta \approx \underbrace{\frac{x}{x \pm y}}_{\text{amplification factors}} \cdot \underbrace{\triangle x}_{\text{relative error on entry}} \pm \frac{y}{x \pm y} \cdot \triangle y + \varepsilon_{\pm} \leftarrow \text{round off of the result}$$

- If $|x \pm y| \ll |x|$ or $|x \pm y| \ll |y|$: massive error propagation, *Cancellation*!
- When writing algorithms try to avoid this as much as possible

When you compute the difference of two numbers of almost the same size, you will encounter cancellation!

Cancellation

Example 0.12

$a = \frac{1}{101}$, $b = \frac{1}{102}$. Round off after four digits of mantissa in $\mathbb{M}(10, 4, 1)$ gives

$$\tilde{a} = a(1 + \Delta a) = 9.901e - 3, \ \tilde{b} = b(1 + \Delta b) = 9.804e - 3.$$

Here we obtain $|\Delta a| \leqslant 10^{-6}$, $|\Delta b| \lessapprox 8 \cdot 10^{-6}$, $\epsilon = 5 \cdot 10^{-4}$.
Exact result $c = a - b = \frac{1}{10302} \approx 9.70685 \cdot 10^{-5}$

$$\tilde{c} = \tilde{a} \ominus \tilde{b} = \tilde{a} - \tilde{b} = 9.700e - 5.$$

Relative error $(c - \tilde{c})/c \approx 7.1 \cdot 10^{-4}$. Error increases by a factor of about 90!