COL100: Introduction to Computer Science

11: Floating-point arithmetic

Computing with real numbers

In general, we cannot represent arbitrary real numbers *exactly* using a finite amount of space

$$e^{\pi} = 23.140692632779269005729086\ldots$$

(Arithmetic would also take an infinite amount of time!)

Scientific notation:

$$e^{\pi} \approx 2.31407 \times 10^{1}$$

Floating-point numbers

n-digit floating-point number in base β:

$$x = \pm(d_0.d_1d_2d_3\ldots d_{n-1})_\beta \times \beta^e$$

with digits $0 \le d_i < \beta$, and integer **exponent** e.

The rational $(d_0.d_1d_2d_3\ldots d_{n-1})_\beta = \sum d_k\, \beta^{-k}$ is the **mantissa**.

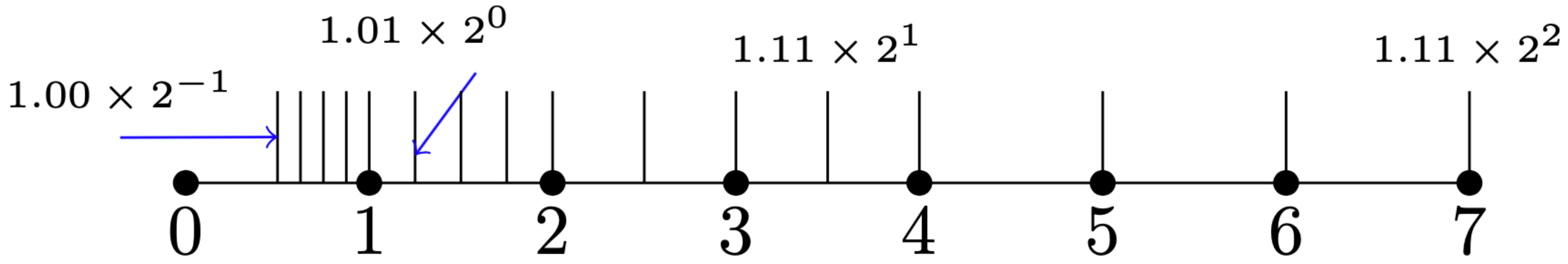
- In decimal ($\beta = 10$), $n = 6$: $e^\pi \approx 23.1407_{10} = +2.31407_{10} \times 10^1$
- In binary ($\beta = 2$), $n = 6$: $e^\pi \approx 10111.0_2 = +1.01110_2 \times 2^4$

The number is **normalized** when either $d_0 \ne 0$, or all $d_i = 0$.

Floating-point numbers

$$x = \pm(d_0.d_1d_2d_3\ldots d_{n-1})_\beta \times \beta^e$$

All positive floats for $\beta = 2$, $n = 3$, $-1 \le e \le 2$:



All modern machines have built-in support for $\beta = 2$, $n = 53$, $-1022 \le e \le 1023$:
IEEE 754 **double-precision** floating-point format

Rounding

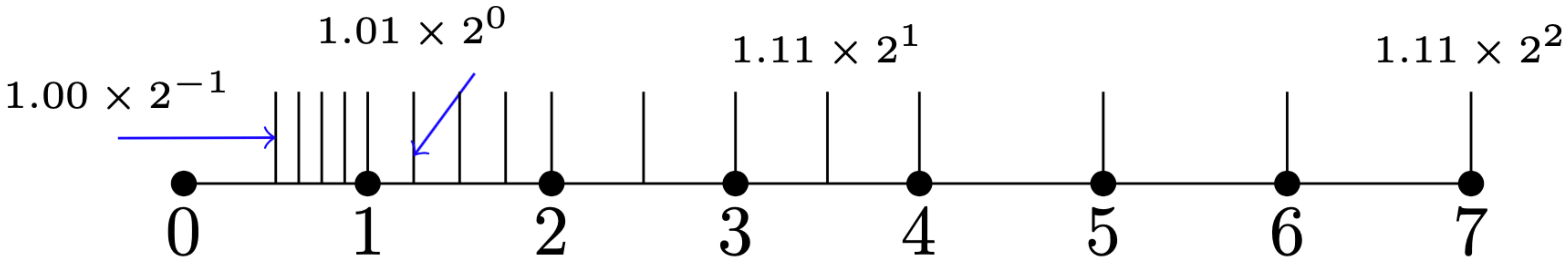
An arbitrary real number $x \in \mathbb{R}$ can only be *approximated* by a floating-point number $\mathrm{fl}(x) \in \mathbb{F}$ (e.g. by rounding to nearest)

(*Note:* $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$ is a theoretical function, not something we can implement!)

e.g. For decimal floating-point with $n = 3$,

- $\mathrm{fl}(2/3) = 6.67 \times 10^{-1}$
- $\mathrm{fl}(2021) = 2.02 \times 10^{2}$

Machine epsilon



Absolute error fl(x) – x varies a lot. But **relative error** (fl(x) – x)/x is bounded! For rounding to nearest,

$$|(\mathrm{fl}(x) - x)/x| \leq \tfrac{1}{2}\,\beta^{-(n-1)}$$

This upper bound is called **machine epsilon** or **unit roundoff** and denoted u.

Conveniently, for any real x (with $\beta^{emin} \leq |x| \leq \beta^{emax}$) we can write

$$\mathrm{fl}(x) = x(1 + \delta) \quad \text{for some } \delta \text{ with } |\delta| \leq u.$$

Floating-point operations

Even if our input numbers are exactly representable as floating-point numbers, the results of arithmetic might not be!

$$x, y \in \mathbb{F} \quad \not\Rightarrow \quad x + y, x - y, x \times y, x \,/\, y \in \mathbb{F}$$

Computer arithmetic is designed to give the closest float result:

$$\begin{aligned} x \circledast y &= \mathrm{fl}(x * y) \\ &= (x * y)(1 + \delta) \quad \text{…for some } \delta \text{ with } |\delta| \leq u. \end{aligned}$$

So result of any floating-point arithmetic has some (bounded) relative error!

Error analysis example

Given x, $y \in \mathbb{F}$, compute $f(x, y) = x^2 - y^2$.

$$
\begin{aligned}
p &= x \otimes x = x^2 (1 + \delta_1) \quad \text{with } |\delta_1| \leq u \\
q &= y \otimes y = y^2 (1 + \delta_2) \quad \text{with } |\delta_2| \leq u \\
r &= p \ominus q \\
&= (p - q)(1 + \delta_3) \quad \text{with } |\delta_3| \leq u \\
&= (x^2 (1 + \delta_1) - y^2 (1 + \delta_2))(1 + \delta_3) \\
&= x^2 (1 + \delta_1)(1 + \delta_3) - y^2 (1 + \delta_2)(1 + \delta_3)
\end{aligned}
$$

This is exactly $f(\tilde{x}, \tilde{y}) = \tilde{x}^2 - \tilde{y}^2$ for some perturbed input $\tilde{x} = x \sqrt{((1+\delta_1)(1+\delta_3))}$, $\tilde{y} = y \sqrt{((1+\delta_2)(1+\delta_3))}$!

Relative **backward errors** $(\tilde{x} - x)/x$, $(\tilde{y} - y)/y$ can be shown to be $O(u)$ as $u \to 0$.

Loss of significance

Accumulation of error can have catastrophic results!

Example ($\beta = 10$, $n = 6$):

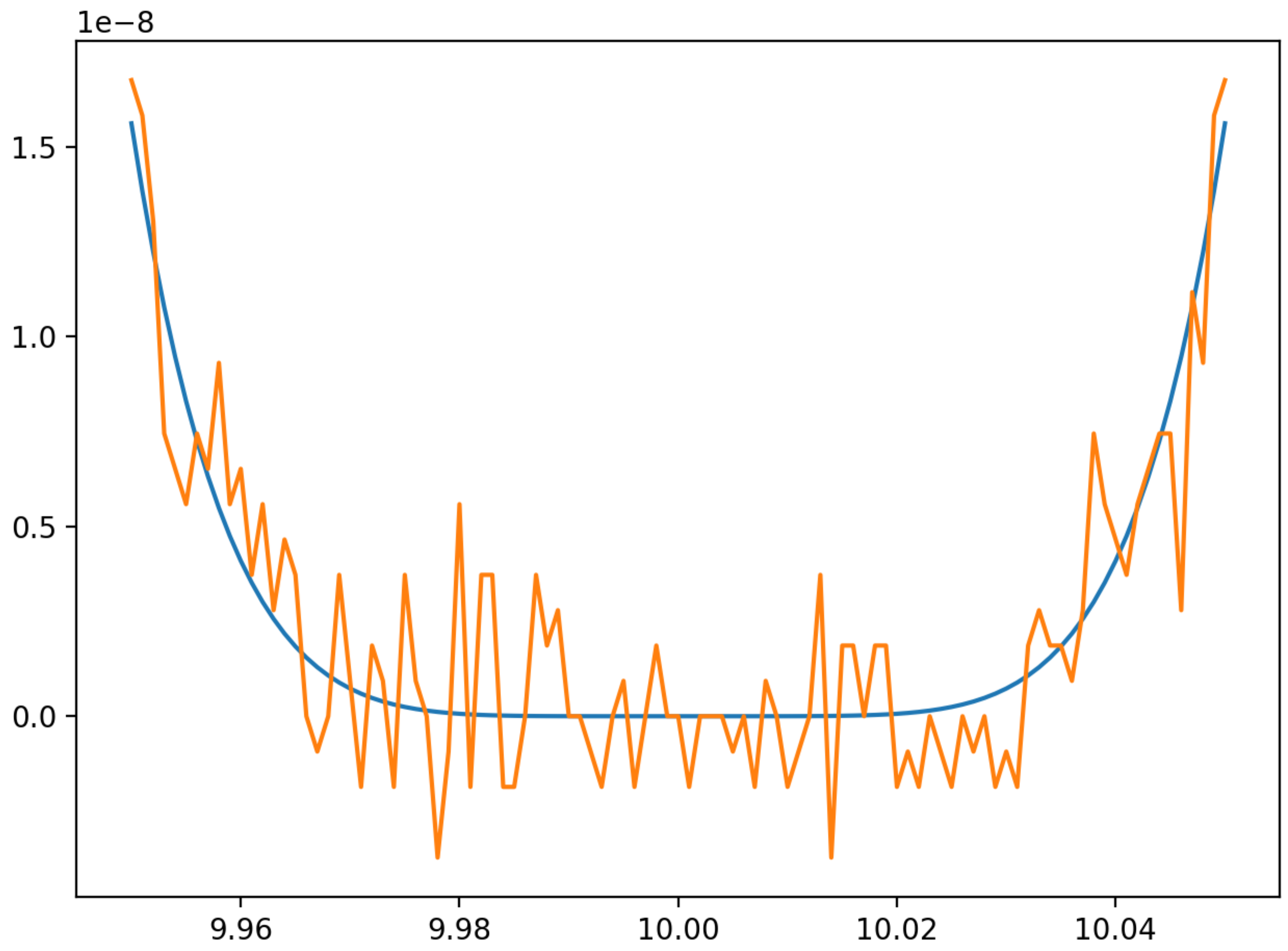
$$\begin{aligned} \tilde{x} &= 1.23456 \times 10^{2} \\ \tilde{y} &= 1.23321 \times 10^{2} \\ \tilde{x} \ominus \tilde{y} &= 0.00135 \times 10^{2} \\ &= 1.35000 \times 10^{-1} \end{aligned}$$

No rounding error in the subtraction! But result has only three significant digits

If relative error of $\tilde{x}$, $\tilde{y}$ is $\sim 10^{-5}$ then *relative* error of $\tilde{x} - \tilde{y}$ is $\sim 10^{-2}$

Example: Polynomial evaluation

$$p(x) = (x - 10)^6$$
$$= x^6 - 60x^5 + 1500x^4 - 20{,}000x^3 + 150{,}000x^2 - 600{,}000x + 1{,}000{,}000$$



Conditioning

Given inaccurate inputs and inaccurate calculations, how accurate of an answer can we expect?

Consider $f : \mathbb{R} \to \mathbb{R}$ and $x \in \mathbb{R}$. How sensitive is $f(x)$ to errors in x?

- Relative error in input = $(\tilde{x} - x)/x$
- Relative error in output = $(f(\tilde{x}) - f(x))/f(x)$
- **Condition number** = maximum ratio over all "nearby" $\tilde{x}$

$$\lim_{\delta \to 0} \sup_{|\tilde{x} - x| \leq \delta} \left(\frac{|(f(\tilde{x}) - f(x))/f(x)|}{|(\tilde{x} - x)/x|} \right)$$

Conditioning

$$\lim_{\delta \to 0} \sup_{|\tilde{x} - x| \leq \delta} \left(\frac{|(f(\tilde{x}) - f(x))/f(x)|}{|(\tilde{x} - x)/x|} \right)$$

$$= |x\, f'(x) \,/\, f(x)|$$

If large: small errors in input produce large errors in output!

Example: Let $r(c)$ = largest root of $x^2 - 2x + c$.

$$c = 1 \Rightarrow r(c) = 1$$

$$c = 0.9999 \Rightarrow r(c) = 1.01$$

Rel. output error = 100× rel. input error. At $c = 1$, condition number = ∞

Afterwards

Read the lecture notes on numerical computing by Prof. Suban