Rounding errors

Example

Show demo: "Waiting for 1".

Determine the double-precision machine representation for 0.1

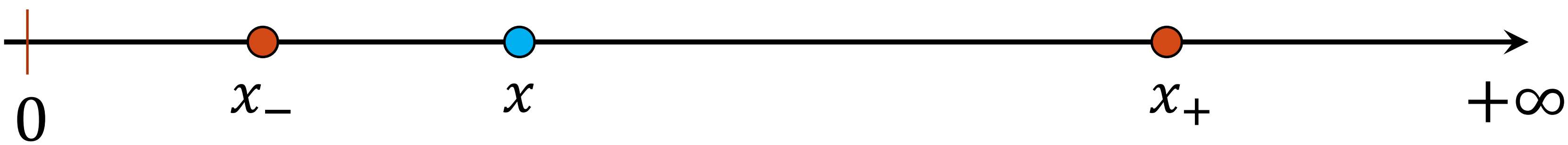
$$0.1 = (0.000110011\,\overline{0011}\ldots)_2 = (1.100110011\ldots)_2 \times 2^{-4}$$

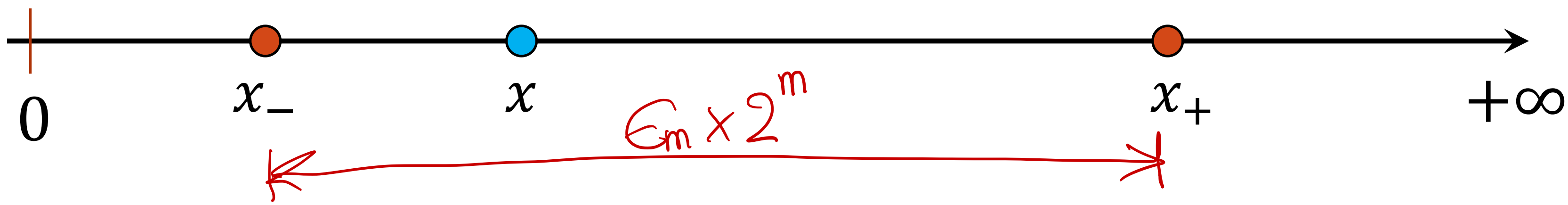
Machine floating point number

- Not all real numbers can be exactly represented as a machine floating-point number.
- Consider a real number in the normalized floating-point form:

$$x = \pm 1.b_1 b_2 b_3 \dots b_n \dots \times 2^m$$

- The real number x will be approximated by either x_- or x_+, the nearest two machine floating point numbers.





Exact number: $x = 1.b_1 b_2 b_3 \ldots b_n \ldots \times 2^m$

$$x_- = 1.b_1 b_2 b_3 \ldots b_n \times 2^m$$

$$x_+ = 1.b_1 b_2 b_3 \ldots b_n \times 2^m + \underbrace{0.000\ldots01}_{\epsilon_m} \times 2^m$$

Gap between x_+ and x_-: $|x_+ - x_-| = \epsilon_m \times 2^m$

Examples for single precision:

x_+ and x_- of the form $q \times 2^{-10}$

x_+ and x_- of the form $q \times 2^{4}$:

x_+ and x_- of the form $q \times 2^{20}$:

x_+ and x_- of the form $q \times 2^{60}$:

The interval between successive floating point numbers is not uniform: the interval is smaller as the magnitude of the numbers themselves is smaller, and it is bigger as the numbers get bigger.

Gap between two successive machine floating point numbers

A "toy" number system can be represented as $x = \pm 1.b_1 b_2 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

$(1.00)_2 \times 2^0 = 1$
$(1.01)_2 \times 2^0 = 1.25$
$(1.10)_2 \times 2^0 = 1.5$
$(1.11)_2 \times 2^0 = 1.75$

$(1.00)_2 \times 2^1 = 2$
$(1.01)_2 \times 2^1 = 2.5$
$(1.10)_2 \times 2^1 = 3.0$
$(1.11)_2 \times 2^1 = 3.5$

$(1.00)_2 \times 2^2 = 4.0$
$(1.01)_2 \times 2^2 = 5.0$
$(1.10)_2 \times 2^2 = 6.0$
$(1.11)_2 \times 2^2 = 7.0$

$(1.00)_2 \times 2^3 = 8.0$
$(1.01)_2 \times 2^3 = 10.0$
$(1.10)_2 \times 2^3 = 12.0$
$(1.11)_2 \times 2^3 = 14.0$

$(1.00)_2 \times 2^4 = 16.0$
$(1.01)_2 \times 2^4 = 20.0$
$(1.10)_2 \times 2^4 = 24.0$
$(1.11)_2 \times 2^4 = 28.0$

$(1.00)_2 \times 2^{-1} = 0.5$
$(1.01)_2 \times 2^{-1} = 0.625$
$(1.10)_2 \times 2^{-1} = 0.75$
$(1.11)_2 \times 2^{-1} = 0.875$

$(1.00)_2 \times 2^{-2} = 0.25$
$(1.01)_2 \times 2^{-2} = 0.3125$
$(1.10)_2 \times 2^{-2} = 0.375$
$(1.11)_2 \times 2^{-2} = 0.4375$

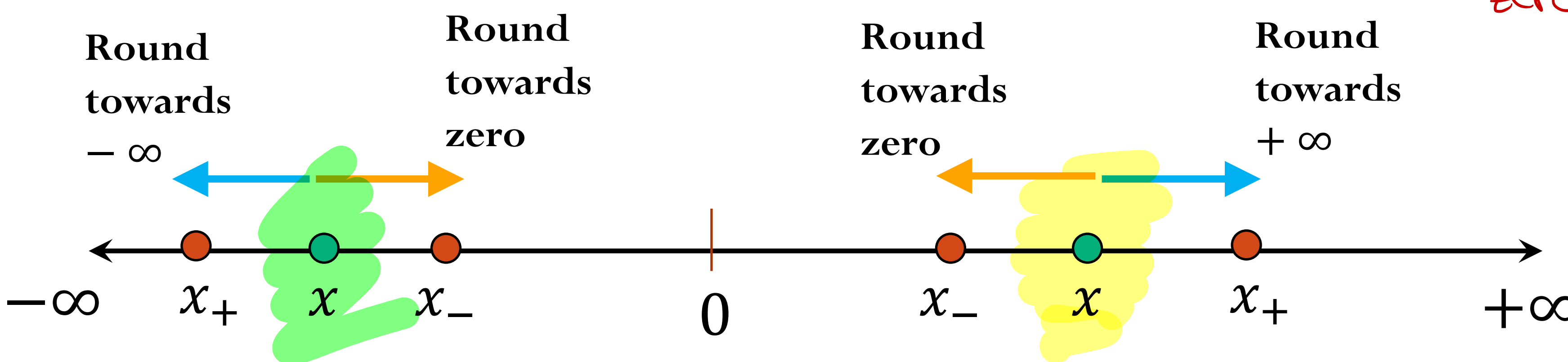
$(1.00)_2 \times 2^{-3} = 0.125$
$(1.01)_2 \times 2^{-3} = 0.15625$
$(1.10)_2 \times 2^{-3} = 0.1875$
$(1.11)_2 \times 2^{-3} = 0.21875$

$(1.00)_2 \times 2^{-4} = 0.0625$
$(1.01)_2 \times 2^{-4} = 0.078125$
$(1.10)_2 \times 2^{-4} = 0.09375$
$(1.11)_2 \times 2^{-4} = 0.109375$

Rounding $\hat{x} = fl(x) = round(x)$

★ Tie-break rule:
- default: round to nearest even
- other: round away from zero

The process of replacing x by a nearby machine number is called rounding, and the error involved is called **roundoff error.**



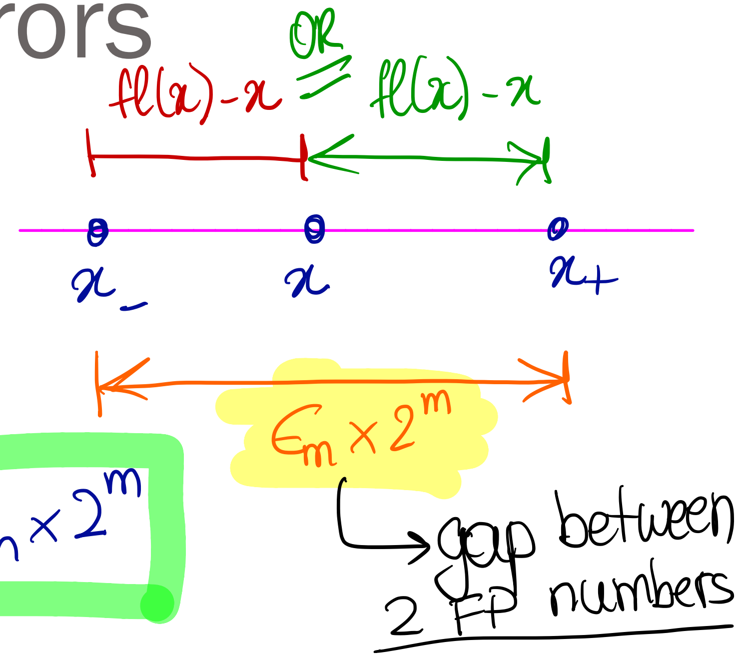
Round by chopping:

	x is positive number	x is negative number
Round up (ceil)	round towards $+\infty$ $fl(x) = x_+$	round toward zero $fl(x) = x_-$
Round down (floor)	round towards zero $fl(x) = x_-$	round towards $-\infty$ $fl(x) = x_+$

★ **Round to nearest**: round towards closest FP. (down or up)

Rounding (roundoff) errors

Consider rounding by chopping:



- **Absolute error:**

$$|fl(x) - x| \leq |x_+ - x_-|$$

or

$$\boxed{|fl(x) - x| \leq \epsilon_m \times 2^m}$$

- **Relative error:**

$$\underbrace{\frac{|fl(x) - x|}{x}}_{e_r} \leq \frac{|x_+ - x_-|}{x} = \frac{\epsilon_m \times 2^m}{x} = \frac{\epsilon_m \times 2^m}{q \times 2^m} \quad (1 \leq q < 2)$$

$$e_r \leq \frac{\epsilon_m \times \cancel{2^m}}{1.b_1b_2\ldots \times \cancel{2^m}} \Rightarrow \boxed{e_r \leq \epsilon_m}$$

Relative error due to rounding (get FP representation) is less than machine epsilon.

Rounding (roundoff) errors

$\boxed{e_r \leq 5 \times 10^{-n}}$

$e_r \lesssim 10^{k} \Rightarrow k = -n+1$

x_- $\qquad$ $x = 1.b_1 b_2 b_3 \ldots b_n \ldots \times 2^m$ $\qquad$ x_+

$$\frac{|\tilde{x} - x|}{|x|} \leq 2^{-23} \approx 1.2 \times 10^{-7}$$

$$\frac{|\tilde{x} - x|}{|x|} \leq \underbrace{2^{-52}}_{\epsilon_m} \approx 2.2 \times 10^{-16}$$

$< 5 \times 10^{-n}$

Single precision: Floating-point math consistently introduces relative errors of about 10^{-7}. Hence, single precision gives you **about $\boxed{7}$ (decimal) accurate digits.**

$\boxed{\text{Rule of thumb!}}$

Double precision: Floating-point math consistently introduces relative errors of about 10^{-16}. Hence, double precision gives you **about $\boxed{16}$ (decimal) accurate digits.**

$e_r \leq \epsilon_m$

Single Precision $\Rightarrow e_r \leq 2^{-23} \approx 1.2 \times 10^{-7} \leq 5 \times 10^{-n}$

Recall that $e_r \leq 5 \times 10^{-n}$

$n = 7$ (single precision gives about 7 decimal digits of accuracy)

* Even if we were to write

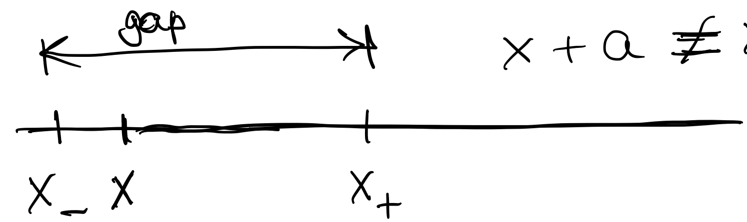
$$e_r \leq 10^{-n+1} \Rightarrow \log_{10}(e_r) \leq \log_{10}(10^{-n+1}) = (-n+1)1$$

$$n \leq 1 - \log_{10}(e_r) = 1 - \log_{10}(1.2 \times 10^{-7})$$

$\boxed{n \leq 7.92}$ → so it does not quite give 8 decimal digits of accuracy!

Iclicker question

in general ...



$x + a \neq x$

Assume you are working with IEEE single-precision numbers. Find the smallest number a that satisfies

$$2^8 + a \neq 2^8$$

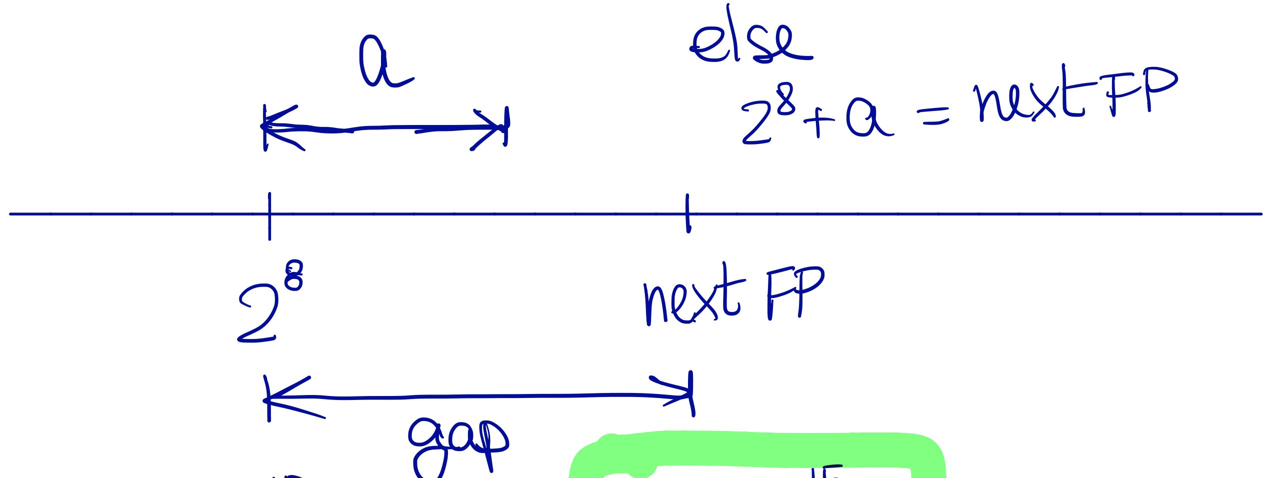
A) 2^{-1074}
B) 2^{-1022}
C) 2^{-52}
D) 2^{-15}
E) 2^{-8}

if $a <$ gap:

$2^8 + a = 2^8$

else

$2^8 + a = \text{nextFP}$



$$\text{gap} = \epsilon_m \times 2^8 = 2^{-23} \times 2^8 = 2^{-15} \longrightarrow \boxed{a > 2^{-15}}$$

Rule of thumb: $\underbrace{q \times 2^m}_{x} + a \neq q \times 2^m \Rightarrow a > \epsilon_m 2^m$

Demo

$a = 10^5 \quad \beta = 1.0$

while $(a+\beta) > a$:

$\beta = \beta/2$

print (β)

Loop will terminate when $a + \beta = a$

double precision: $\beta = \text{gap} = \underbrace{10^{-16}}_{\epsilon_m} \; 10^5 = 10^{-11}$

Mathematical properties of FP operations

Not necessarily associative:

For some x , y, z the result below is possible:

$$(x + y) + z \neq x + (y + z)$$

Not necessarily distributive:

For some x , y, z the result below is possible:

$$z\,(x + y) \neq z\,x + z\,y$$

Not necessarily cumulative:

Repeatedly adding a very small number to a large number may do nothing

Demo: FP-arithmetic

Floating point arithmetic

Consider a number system such that $x = \pm 1.b_1 b_2 b_3 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

Rough algorithm for addition and subtraction:

1. Bring both numbers onto a common exponent
2. Do "grade-school" operation
3. Round result

- **Example 1: No rounding needed**

$$
\begin{array}{rl}
+ & a = (1.101)_2 \times 2^1 \\
 & b = (1.001)_2 \times 2^1 \\
\hline
\end{array}
$$

$10.110 \times 2^1 = 1.0110 \times 2^2 = 1.011 \times 2^2$ ✓

Floating point arithmetic

Consider a number system such that $x = \pm 1.b_1b_2b_3 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

- **Example 2: Require rounding**

$$
\begin{array}{rl}
 & a = (1.101)_2 \times 2^0 \\
+ & b = (1.000)_2 \times 2^0 \\
\hline
 & 10.101 \times 2^0 = 1.0101 \times 2^1 \xrightarrow{\text{chopping}} 1.010 \times 2^1
\end{array}
$$

- **Example 3:**

$$
\begin{array}{rl}
 & a = (1.100)_2 \times 2^1 \\
+ & b = (1.100)_2 \times 2^{-1} \longrightarrow 0.01100 \times 2^2 \times 2^{-1} = 0.01100 \times 2^1 \\
\hline
\end{array}
$$

$$
\begin{array}{rl}
 & 1.100 \times 2^1 \\
+ & 0.01100 \times 2^1 \\
\hline
 & 1.111 \times 2^1
\end{array}
$$

(no rounding needed)

Floating point arithmetic

Consider a number system such that $x = \pm 1.b_1b_2b_3b_4 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

- **Example 4:**

$$\left.\begin{aligned} a &= (1.1011)_2 \times 2^1 \\ b &= (1.1010)_2 \times 2^1 \end{aligned}\right\} \text{ numbers are "close" to each other}$$

$$c = a - b$$

$$\begin{array}{r} 1.1011 \times 2^1 \\ -\ 1.1010 \times 2^1 \\ \hline 0.0001 \times 2^1 \end{array} \xrightarrow[\text{normalize}]{} 1.\underline{????} \times 2^1$$

machine choice

$$1.\underbrace{000}\times 2^1$$

NOT SIGNIFICANT DIGITS!

Cancellation

Demo

$a = 1.a_1a_2a_3a_4a_5a_6 \ldots a_n \ldots \times 2^{m1}$

$b = 1.b_1b_2b_3b_4b_5b_6 \ldots b_n \ldots \times 2^{m2}$

lost precision due to rounding

Suppose $a \approx b$ and single precision (without loss of generality)

$a = 1.a_1a_2a_3a_4a_5a_6 \ldots a_{20}a_{21}10a_{24}a_{25}a_{26}a_{27} \ldots \times 2^m$

$b = 1.a_1a_2a_3a_4a_5a_6 \ldots a_{20}a_{21}11b_{24}b_{25}b_{26}b_{27} \ldots \times 2^m$

$fl(b-a) = 0.0000 \ldots 0001 \times 2^m$

$\Rightarrow$ normalize

$fl(b-a) \; 1.\underbrace{0000 \ldots 00}_{n \text{ bits}} \times 2^{-n+m}$

not significant bits (spurious zeros!)

Catastrophic cancellation

Example of cancellation:

Suppose $a = 1.1011a_5a_6a_7\ldots \times 2^1$

$b = 1.1010\,b_5b_6b_7\ldots \times 2^1$

using machine where $n=4$ $\Rightarrow$ $a = 1.1011 \times 2^1$

$b = 1.1010 \times 2^1$

$$a-b \Rightarrow \begin{array}{r} 1.1011\,a_5\,a_6\,a_7\ldots \times 2^1 \\ \ominus\ 1.1010\,b_5\,b_6\,b_7\ldots \times 2^1 \\ \hline 0.0001 \times 2^1 \end{array}$$

machine resulting with cancellation

$1.\underbrace{0000}_{\text{not significant digits}} \times 2^{-3}$

when done by "hand"

$1.\underbrace{c_1c_2c_3c_4}_{\text{significant digits}} \times 2^{-3}$

from $a_5a_6a_7a_8$

$\ominus\ b_5b_6b_7b_8$

Cancellation

$a = 1.a_1a_2a_3a_4a_5a_6 \ldots a_n \ldots \times 2^{m1}$
$b = 1.b_1b_2b_3b_4b_5b_6 \ldots b_n \ldots \times 2^{m2}$

For example, assume single precision and $m1 = m2 + 18$ (without loss of generality), i.e. $a \gg b$

$$fl(a) = 1.a_1a_2a_3a_4a_5a_6 \ldots a_{22}a_{23} \times 2^{m+18}$$

$$fl(b) = 1.b_1b_2b_3b_4b_5b_6 \ldots b_{22}b_{23} \times 2^{m}$$

$$\begin{array}{rl} & 1.a_1a_2a_3a_4a_5a_6 \ldots a_{22}a_{23} \times 2^{m+18} \\ + & 0.0000 \ldots 001b_1b_2b_3b_4b_5 \times 2^{m+18} \\ \hline \end{array}$$

In this example, the result $fl(a+b)$ only included 6 bits of precision from $fl(b)$. Lost precision!

Loss of Significance

How can we avoid this loss of significance? For example, consider the function $f(x) = \sqrt{x^2 + 1} - 1$

If we want to evaluate the function for values x near zero, there is a potential loss of significance in the subtraction.

Let's consider five-decimal digit arithmetic and evaluate $f(x)$ at $x = 10^{-3}$

$f(x) = \sqrt{10^{-6} + 1} - 1 = \text{zero}$! (since 10^{-6} is smaller than machine epsilon $\epsilon_m \approx 10^{-5}$)

How can we obtain better results and avoid cancellation?

Loss of Significance

Re-write the function as $f(x) = \frac{x^2}{\sqrt{x^2+1}-1}$ (no subtraction!)

Re-write the function to "eliminate" subtraction of similar numbers

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

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

$$f(10^{-3}) = \frac{10^{-6}}{\sqrt{10^{-6}+1}+1} = \frac{10^{-6}}{2}$$

(note that 10^{-6} is not zero, i.e.
$10^{-6} < \epsilon_m$ but not smaller than UFL)

Example:

exact values

If $x = 0.3721448693$ and $y = 0.3720214371$ what is the relative error in the computation of $(x - y)$ in a computer with five decimal digits of accuracy?

approximations using 5 decimal digits $\tilde{x} = 0.37214$

$\tilde{y} = 0.37202$

relative error due to rounding:

$$\frac{|x - \tilde{x}|}{|x|} \simeq 1.3 \times 10^{-5}$$

$$e_r = \frac{|(x-y) - (\tilde{x} - \tilde{y})|}{|x-y|}$$ (relative error of difference)

$$e_r = \frac{0.0001234322 - 0.00012}{0.0001234322} \simeq 3 \times 10^{-2}$$

→ the error due to the subtraction is "large" compared to the error due to rounding because of cancelation.