Video 1: Rounding errors

A number system can be represented as $x = \pm 1.b_1 b_2 b_3 b_4 \times 2^m$ for $m \in [-6,6]$ and $b_i \in \{0,1\}$.

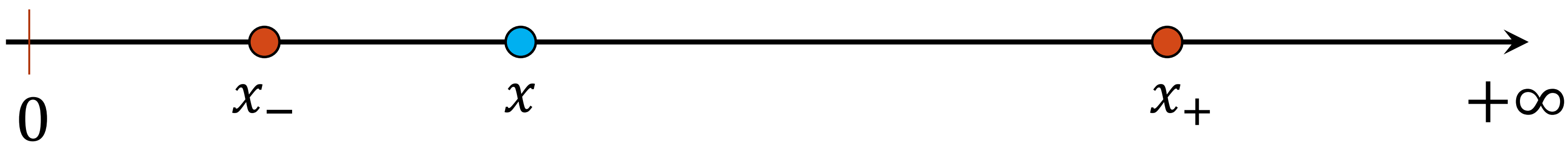
Let's say you want to represent the decimal number 19.625 using the binary number system above. Can you represent this number exactly?

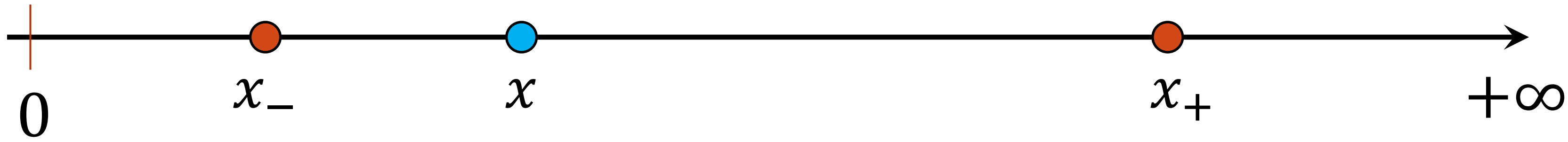
Machine floating point number

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

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

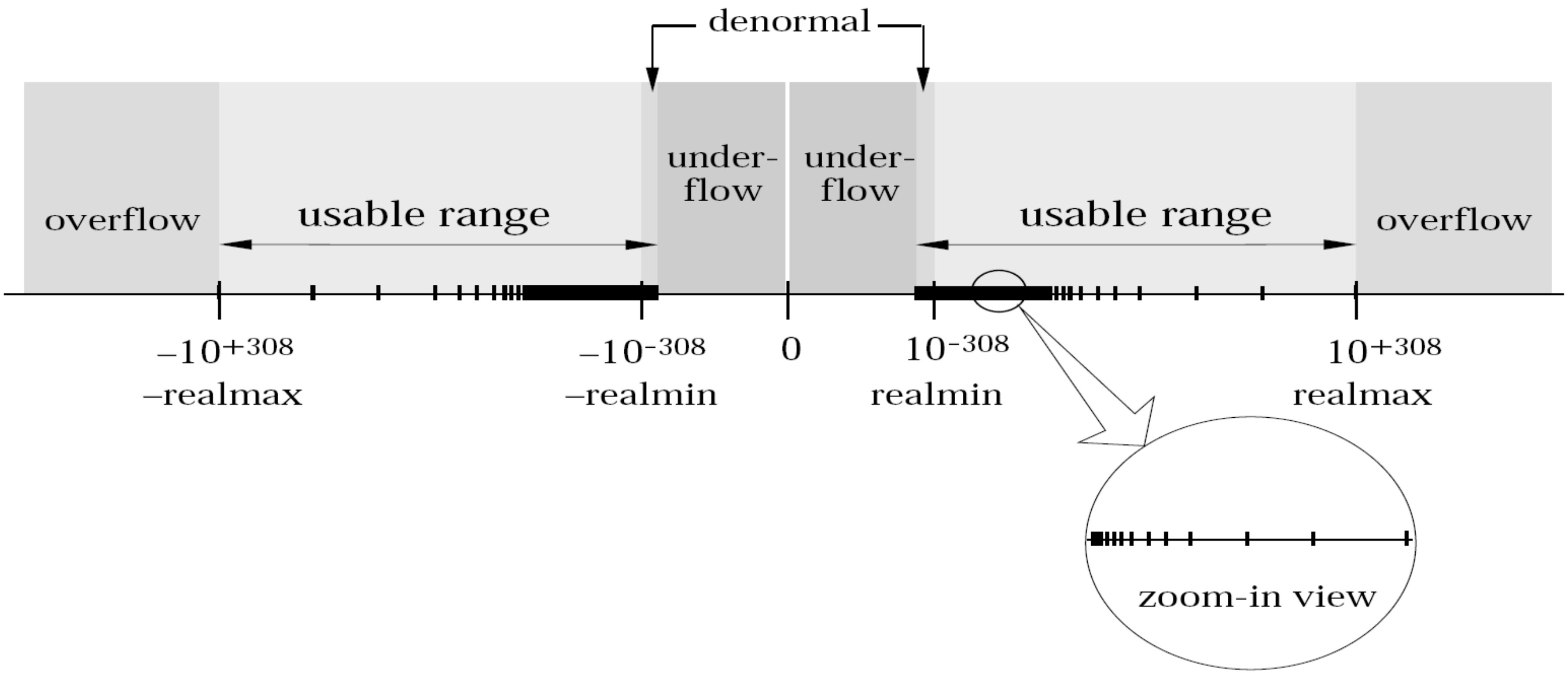
- The real number x will be approximated by either x_- or x_+, the nearest two machine floating point numbers.





0
x_-
x
x_+
$+\infty$

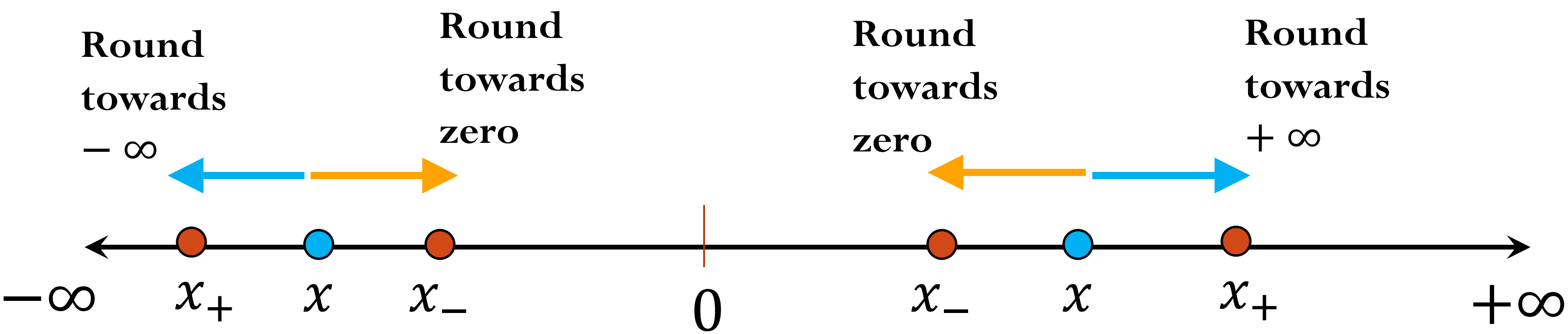
IEEE-754 Double Precision



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.

Rounding

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)	$fl(x) = x_+$ Rounding towards $+\infty$	$fl(x) = x_-$ Rounding towards zero
Round down (floor)	$fl(x) = x_-$ Rounding towards zero	$fl(x) = x_+$ Rounding towards $-\infty$

Round to nearest: either round up or round down, whichever is closer

Rounding (roundoff) errors

Consider rounding by chopping:

- **Absolute error:**

- **Relative error:**

Rounding (roundoff) errors

$$\frac{|fl(x) - x|}{|x|} \leq \epsilon_m$$

The relative error due to rounding (the process of representing a real number as a machine number) **is always bounded by machine epsilon.**

IEEE Single Precision

$$\frac{|fl(x) - x|}{|x|} \le 2^{-23} \approx 1.2\times10^{-7}$$

IEEE Double Precision

$$\frac{|fl(x) - x|}{|x|} \le 2^{-52} \approx 2.2\times10^{-16}$$

Gap between two machine numbers

Video 2: Arithmetic with machine numbers

Mathematical properties of FP operations

Not necessarily associative:

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

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

```
In [5]: (np.pi+1e100)-1e100
Out[5]: 0.0

In [6]: (np.pi)+(1e100-1e100)
Out[6]: 3.141592653589793

In [7]: b = 1e80
        a = 1e2
        print(a + (b - b) )
        print((a + b) - b )

100.0
0.0
```

Not necessarily distributive:

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

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

```
In [3]: print(100*(0.1 + 0.2))
        print(100*0.1 + 100*0.2)

30.000000000000004
30.0

In [4]: 100*(0.1 + 0.2) == 100*0.1 + 100*0.2
Out[4]: False
```

Not necessarily cumulative:

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

Floating point arithmetic (basic idea)

$$x = (-1)^{\boldsymbol{s}}\, 1.\boldsymbol{f} \times 2^{\boldsymbol{m}}$$

- First compute the exact result
- Then round the result to make it fit into the desired precision

- $x + y = fl(x + y)$
- $x \times y = fl(x \times y)$

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**

$$a = (1.101)_2 \times 2^1$$
$$b = (1.001)_2 \times 2^1$$

$$c = a + b = (10.110)_2 \times 2^1 = (1.011)_2 \times 2^2$$

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 $b_i \in \{0,1\}$.

- **Example 2: Require rounding**

$$a = (1.101)_2 \times 2^0$$
$$b = (1.000)_2 \times 2^0$$

$$c = a + b = (10.101)_2 \times 2^0$$

- **Example 3:**

$$a = (1.100)_2 \times 2^1$$
$$b = (1.100)_2 \times 2^{-1}$$

$$c = a + b = (1.100)_2 \times 2^1 + (0.011)_2 \times 2^1 = (1.111)_2 \times 2^1$$

Floating point arithmetic

Consider a number system such that $x = \pm 1.b_1 b_2 b_3 b_4 \times 2^m$ for $m \in [-4,4]$ and $b_i \in \{0,1\}$.

- **Example 4:**

$$a = (1.1011)_2 \times 2^1$$
$$b = (1.1010)_2 \times 2^1$$

$$c = a - b = (0.0001)_2 \times 2^1$$

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:**

$$a = (1.1011)_2 \times 2^1$$
$$b = (1.1010)_2 \times 2^1$$

$$c = a - b = (0.0001)_2 \times 2^1$$

Or after normalization: $c = (1.????)_2 \times 2^{-3}$

- There is not data to indicate what the missing digits should be.
- Machine fills them with its best guess, which is often not good (usually what is called spurious zeros).
- Number of significant digits in the result is reduced.
- This phenomenon is called **Catastrophic Cancellation**.

Loss of significance

Assume a and b are real numbers with $a \gg b$. For example

$$a = 1.a_1a_2a_3a_4a_5a_6 \ldots a_n \ldots \times 2^0$$
$$b = 1.b_1b_2b_3b_4b_5b_6 \ldots b_n \ldots \times 2^{-8}$$

In Single Precision, compute $(a + b)$

$$1.a_1a_2a_3a_4a_5a_6a_7a_8a_9 \ldots a_{22}a_{23} \times 2^0$$

Cancellation

Assume a and b are real numbers with $a \approx b$.

$$a = 1.a_1a_2a_3a_4a_5a_6 \ldots a_n \ldots\times 2^m$$
$$b = 1.b_1b_2b_3b_4b_5b_6 \ldots b_n \ldots\times 2^m$$

In single precision (without loss of generality), consider this example:

$$a = 1.a_1a_2a_3a_4a_5a_6 \ldots a_{20}a_{21}10a_{24}a_{25}a_{26}a_{27} \ldots \times 2^m$$
$$b = 1.a_1a_2a_3a_4a_5a_6 \ldots a_{20}a_{21}11b_{24}b_{25}b_{26}b_{27} \ldots\times 2^m$$

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

Examples:

1) ***a* and *b* are real numbers with same order of magnitude** $(a \approx b)$. They have the following representation in a decimal floating point system with 16 decimal digits of accuracy:

$$fl(a) = 3004.45$$
$$fl(b) = 3004.46$$

How many accurate digits does your answer have when you compute $b - a$?

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.

Assume you are performing this computation using a machine with 5 decimal accurate digits. Compute $f(10^{-3})$

Loss of Significance

Re-write the function $f(x) = \sqrt{x^2 + 1} - 1$ to avoid subtraction of two numbers with similar order of magnitude

Example:

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?