Machine numbers: how floating point numbers are stored?

Floating-point number representation

What do we need to store when representing floating point numbers in a computer?

$$x = \pm 1.f \times 2^{m}$$

$$x = \underbrace{\boxed{\pm}}_{\text{sign}}\underbrace{\boxed{\quad m \quad}}_{\text{exponent}}\underbrace{\boxed{\qquad f \qquad}}_{\text{significand}}$$

Initially, different floating-point representations were used in computers, generating inconsistent program behavior across different machines.

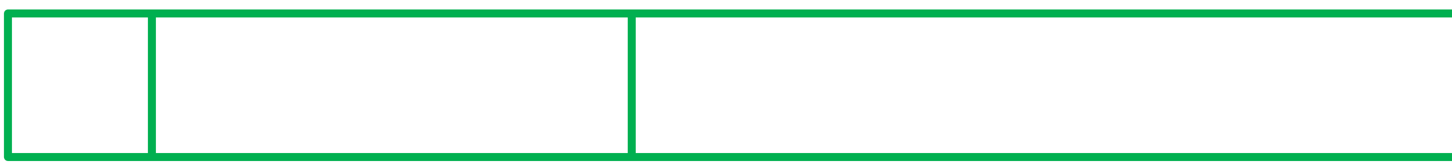
Around 1980s, computer manufacturers started adopting a standard representation for floating-point number: IEEE (Institute of Electrical and Electronics Engineers) 754 Standard.

Floating-point number representation

Numerical form:

$$x = \pm 1.f \times 2^{m}$$

Representation in memory:

$x =$ 

Precisions:

$$x = \pm 1.f \times 2^m$$

$m \in [L, U]$

↳ signed

IEEE-754 Single precision (32 bits):

	1 bit	8 bits	23 bits
$x =$	$\pm$	$c = m + shift$	f

unsigned

IEEE-754 Double precision (64 bits):

	1 bit	11 bits	52 bits
$x =$	$\pm$	$c = m + shift$	f

Special Values:

s = 1 ⇒ negative
s = 0 ⇒ positive

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

1) **Zero**:

8 / 11 23 / 52

$$x = \boxed{\pm \mid 00 \ldots 00 \mid 000 \ldots 000}$$

all zeros all zeros

2) **Infinity**: $+\infty$ $(s = 0)$ and $-\infty$ $(s = 1)$

8/11

$$x = \boxed{\pm \mid 111 \ldots 11 \mid 000 \ldots 000}$$

all ones all zeros

3) **NaN**: (results from operations with undefined results)

$$x = \boxed{\pm \mid 111 \ldots 11 \mid \text{anything but all zeros}}$$

all ones

all zeros in c but anything else in f

IEEE-754 Single Precision (32-bit)

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

s	$c = m + 127$	f
sign (1-bit)	exponent (8-bit)	significand (23-bit)

$c = (00 \ldots 0)_2 \longrightarrow$ reserved $\longrightarrow c = (0)_{10}$

$c = (11 \ldots 1)_2 \longrightarrow$ NaN or $\infty \longrightarrow c = (255)_{10}$

$1 \leq c \leq 254 \longrightarrow 1 \leq m + 127 \leq 254$

$-126 \leq m \leq 127$

$\boxed{\text{range of exponent}} \Longrightarrow \boxed{m \in [-126, 127]}$

IEEE-754 Single Precision (32-bit)

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

Example: Represent the number $x = -67.125$ using IEEE Single-Precision Standard

$$67.125 = (1000011.001)_2 = \underbrace{(1.000011001)_2 \times 2^6}$$

	8 bits	23 bits
1	1 0 0 0 0 1 0 1	0 0 0 0 1 1 0 0 1 0 0 0 ... 0

$m = 6 \longrightarrow c = m + 127 = (133)_{10} = (10000101)_2$

$n = 23 \Rightarrow p = 24$ $m \in [-126, 127]$

IEEE-754 Single Precision (32-bit)

$$x = (-1)^{s}\, 1.f \times 2^{m} = \boxed{s \mid c \mid f} \qquad c = m + 127$$

- **Machine epsilon** (ϵ_m): is defined as the distance (gap) between 1 and the next largest floating point number.

$$1.\underbrace{000 \ldots 00}_{23 \text{ bits}} \times 2^{0}$$

$$1.0000 \ldots 01 \times 2^{0}$$

$$0.000 \ldots 01 \times 2^{0}$$

$2^{-1}\, 2^{-2}\, 2^{-3} \ldots 2^{-23}$

$$\boxed{\epsilon_m = 2^{-23} \approx 1.2 \times 10^{-7}}$$

$$\boxed{\epsilon_m = 2^{-n}}$$

- **Smallest positive normalized FP number:**

$$UFL = 1.00 \ldots 0 \times 2^{-126} \qquad UFL = 2^{L} \approx 10^{-38}$$

- **Largest positive normalized FP number:**

$$OFL = 2^{U+1}(1 - 2^{-p}) = 2^{127+1}(1 - 2^{-24}) \approx 10^{38}$$

IEEE-754 Double Precision (64-bit)

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

s	$c = m + 1023$	f
sign (1-bit)	exponent (11-bit)	significand (52-bit)

$c = (\overbrace{000 \ldots 00}^{11\text{ bits}})_2 = (0)_{10} \longrightarrow$ reserved for zero

$c = (111 \ldots 11)_2 = (2047)_{10} \longrightarrow$ NaN and ∞

$1 \le c \le 2046 \longrightarrow 1 \le m + 1023 \le 2046$

$\boxed{m \in [-1022, 1023]}$ $\boxed{-1022 \le m \le 1023}$

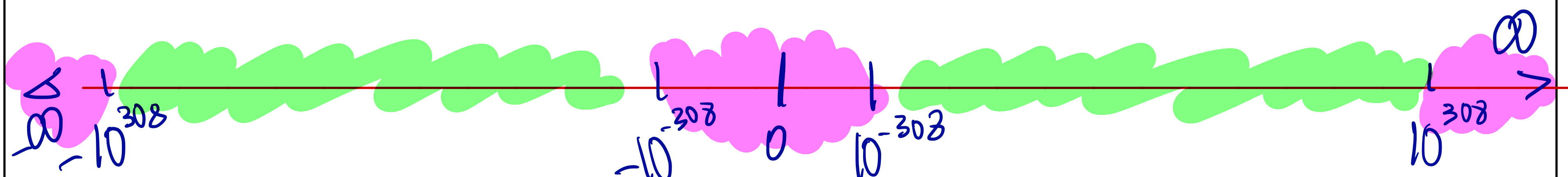
$m \in [-1022, 1023] \quad n = 52 \longrightarrow p = 53$

IEEE-754 Double Precision (64-bit)

$$x = (-1)^{s}\, 1.f \times 2^{m} = \boxed{s \mid c \mid f} \qquad c = m + 1023$$

- **Machine epsilon** (ϵ_m): is defined as the distance (gap) between 1 and the next largest floating point number.

$$\epsilon_m = 2^{-n} \longrightarrow \epsilon_m = 2^{-52} \approx 2.2 \times 10^{-16}$$



- **Smallest positive normalized FP number:**

$$UFL = 2^{L} \longrightarrow UFL = 2^{-1022} \approx 2.2 \times 10^{-308}$$

- **Largest positive normalized FP number:**

$$OFL = 2^{U+1}(1 - 2^{-p}) = 2^{1024}(1 - 2^{-53}) \approx 10^{308}$$

Can we represent # smaller than UEL

$$x = 1.f \times 2^m$$

$$\boxed{x = 0.f \times 2^{L}}$$

denormalized / subnormal

Let's make

$c = (00 \ldots 00)_2$ all zeros

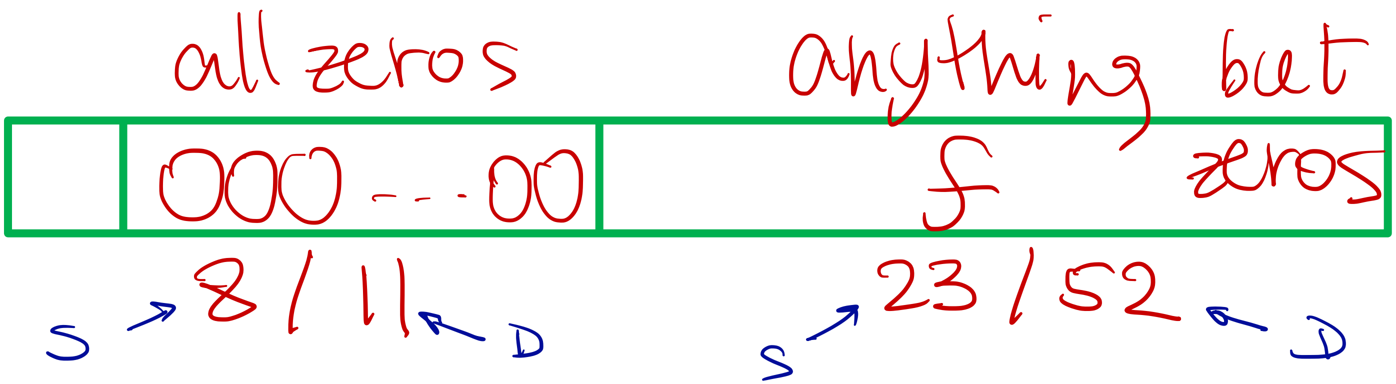
f = anything but zeros

used to "indicate" that exponent is $m = L$ (and <u>NOT</u> to evaluate m)

Subnormal (or denormalized) numbers

- Noticeable gap around zero, present in any floating system, due to normalization
- Relax the requirement of normalization, and allow the leading digit to be zero, only when the exponent is at its minimum ($m = L$)
- Computations with subnormal numbers are often slow.

Representation in memory (another special case):

$x =$ 

Numerical value:

$$X = \pm 0.f \times 2^{L}$$

$m \in [-126, 127]$ single

$m \in [-1022, 1023]$ double

Subnormal (or denormalized) numbers

IEEE-754 Single precision (32 bits):

$c = (00000000)_2 = 0$

smallest subnormal : $(0.000 \ldots 001)_2 \times 2^{-126} = 2^{-23} \times 2^{-126} \approx 1.4 \times 10^{-45}$

$2^{-1}\ 2^{-2}\ 2^{-3}$ 2^{-23}

$n = 23$

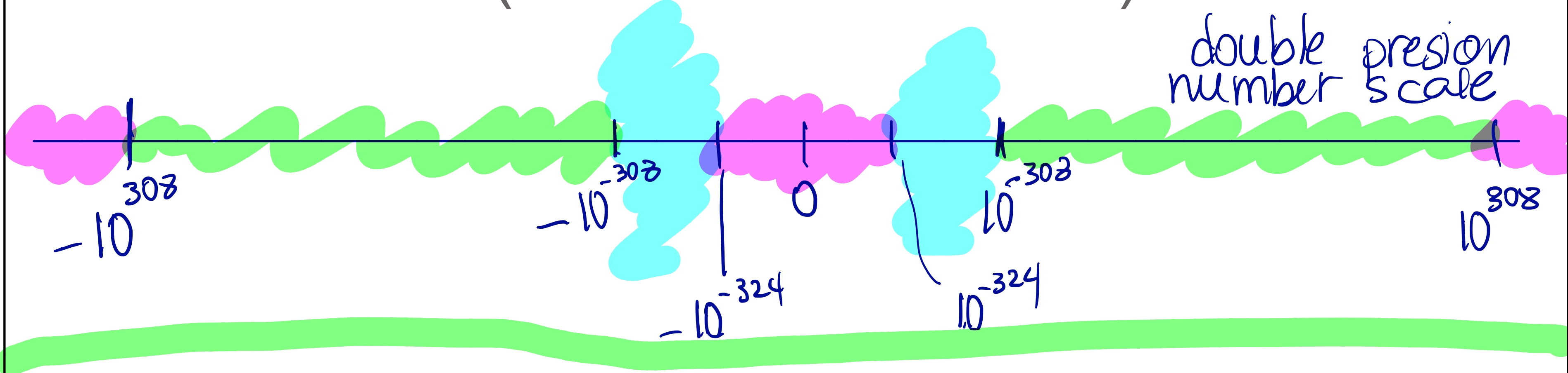
IEEE-754 Double precision (64 bits):

$c = (00000000000)_2 = 0$

smallest subnormal : $(0.0000 \ldots 001)_2 \times 2^{-1022} = 2^{-52}\ 2^{-1022} \approx 4.9 \times 10^{-324}$

$2^{-1}\ 2^{-2}$ 2^{-52}

Subnormal (or denormalized) numbers



Example using single precision

$$1.\underbrace{000 \ldots 00}_{23 \text{ bits}} \times 2^{-126} \rightarrow \text{UFL} \qquad p = 24$$

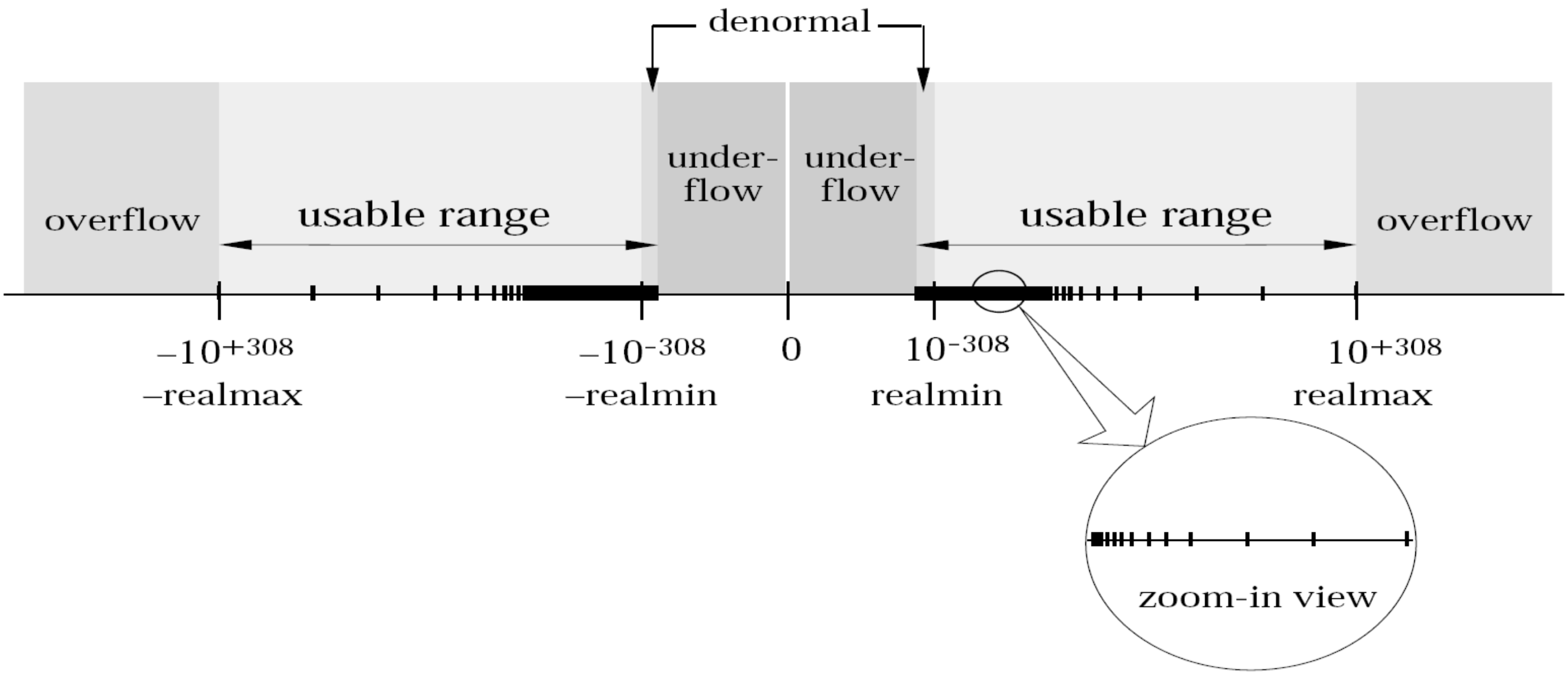
$$0.\underbrace{1111 \ldots 11}_{23 \text{ bits}} \times 2^{-126} \rightarrow p = 23$$

$$0.\underbrace{000 \ldots 00111}_{23 \text{ bits}} \times 2^{-126} \rightarrow p = 3$$

not significant bits

Loss of precision !!

IEEE-754 Double Precision



Summary for Single Precision

$x = (-1)^{s}\, 1.f \times 2^{m} =$ $\boxed{s \mid c \mid f}$ $\quad m = c - 127$

Stored binary exponent (c)	Significand fraction (f)	value
00000000	0000…0000	zero
00000000	$any\ f \neq 0$	$(-1)^{s}\, 0.f \times 2^{-126}$
00000001	$any\ f$	$(-1)^{s}\, 1.f \times 2^{-126}$
⋮	⋮	⋮
11111110	$any\ f$	$(-1)^{s}\, 1.f \times 2^{127}$
11111111	$any\ f \neq 0$	NaN
11111111	0000…0000	infinity

What is the equivalent decimal number?

0 00000000 00000000000000000000000

1 11111111 00000000000000000000000

0 11111111 11111111110000111111111

0 00000000 11110000000000000000000

0 01111111 00000000000000000000000

Iclicker question

$p = n+1 = 4$

A number system can be represented as $x = \pm 1.b_1 b_2 b_3 \times 2^m$ for $m \in [-5,5]$ and $b_i \in \{0,1\}$.

$1.000 \times 2^{-5} = 2^{-5}$

1) **What is the smallest positive normalized FP number:**
a) 0.0625 b) 0.09375 $\boxed{\text{c) } 0.03125}$ d) 0.046875 e) 0.125

2) **What is the largest positive normalized FP number:**
a) 28 $\boxed{\text{b) } 60}$ c) 56 d) 32

1.111×2^5 or $2^{U+1}(1-2^{-p}) = 2^6(1-2^{-4})$

3) **How many additional numbers (positive and negative) can be represented when using subnormal representation?**
a) 7 $\boxed{\text{b) } 14}$ c) 3 d) 6 e) 16

0.001×2^{-5}
0.100
$.010$
$.011$
$.101$
$.110$
$.111$

4) **What is the smallest positive subnormal number?**
$\boxed{\text{a) } 0.00390625}$ b) 0.00195313 c) 0.03125 d) 0.0136719

5) **Determine machine epsilon**

$\epsilon_m = 2^{-n} = 2^{-3}$

a) 0.0625 b) 0.00390625 c) 0.0117188 $\boxed{\text{d) } 0.125}$

A number system can be represented as $x = \pm 1.b_1b_2b_3b_4 \times 2^m$ for $m \in [-6,6]$ and $b_i \in \{0,1\}$.

$n=4$ $p=5$

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?

$(19.625)_{10} = (10011.101)_2 = 1.0011101 \times 2^4$

1.0011×2^4

integer range $\longrightarrow$ until 2^p

double precision $\longrightarrow 2^{53}$

2) What is the range of integer numbers that you can represent exactly using this binary system?

$(1)_{10} = (1)_2 = 1.0000 \times 2^0$

$(2)_{10} = (10)_2 = 1.0000 \times 2^1$

$(3)_{10} = (11)_2 = 1.1000 \times 2^1$

$(15)_{10} = (\overset{2^3}{1}\overset{2^2}{1}\overset{2^1}{1}\overset{2^0}{1}.0)_2 = 1.1110 \times 2^3$

$(\overset{2^4}{1}\overset{2^3}{1}\overset{2^2}{1}\overset{2^1}{1}\overset{2^0}{1})_2 = (31)_{10} = 1.1111 \times 2^4$

$(32)_{10} = 1.0000 \times 2^5$

$1.0001 \times 2^5 = 34$

cannot represent $(33)_{10}$!!

Rounding errors

Example

Show demo: "Waiting for 1".

Determine the double-precision machine representation for 0.1

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

$$\boxed{}\ \underbrace{\boxed{01111111011}}_{c}\ \underbrace{\boxed{\qquad\qquad\qquad\qquad}}_{f}$$

$$m = -4 \longrightarrow c = m + 1023 \longrightarrow c = (1019)_{10}$$

$$f = \underbrace{1001\ 1001\ 1001\ \ldots\ 1001}_{52}\ \big|\ 1001\ \ldots$$

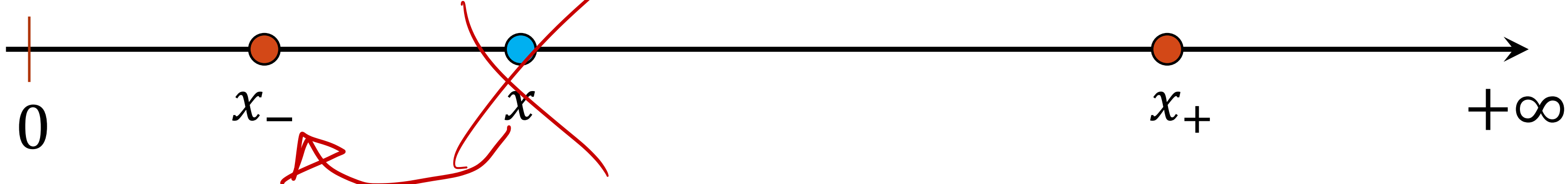
1010 | IGNORE

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.

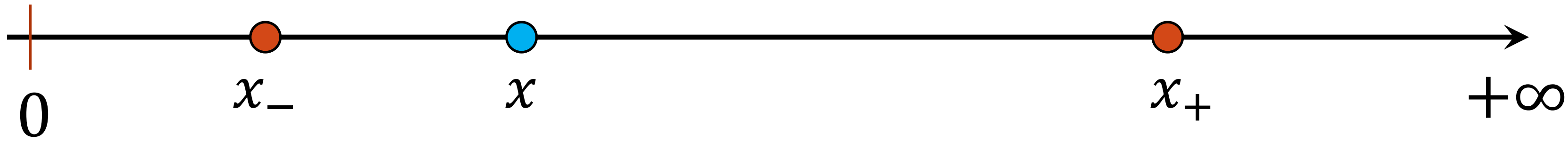


$$X_- = 1.b_1 b_2 b_3 \ldots b_n \times 2^m$$

$$X_+ = \underbrace{1.b_1 b_2 b_3 \ldots b_n \times 2^m}_{X_-} + \boxed{0.000 \ldots 01 \times 2^m}$$

n

$$X_+ - X_- = 2^{-n} \times 2^m$$



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

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

$$x_+ = 1.b_1 b_2 b_3 \ldots b_n \times 2^m + \underbrace{0.000 \ldots 01}_{\epsilon_m} \times 2^m$$

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

Examples for single precision:

x_+ and x_- of the form $q \times 2^{-10}$ $\quad 2^{-23} \times 2^{-10} = 2^{-33} \sim 10^{-10}$

x_+ and x_- of the form $q \times 2^{4}$: $\rightarrow 2^{4} \times 2^{-23} \sim 10^{-6}$

x_+ and x_- of the form $q \times 2^{20}$: $\rightarrow 2^{20} \times 2^{-23} \sim 0.125$

x_+ and x_- of the form $q \times 2^{60}$: $\rightarrow 2^{60} \times 2^{-23} = 2^{37} \sim 10^{11}$

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

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

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

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

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

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

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

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

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

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