# Numerical methods

## Session 2: Machine representation of numbers

Pedro González Rodríguez

Universidad Carlos III de Madrid

February 5, 2020

# Floating point representation of a number

Any machine operation is affected by rounding errors or roundoff. Given a base $\beta \in \mathbb{N}$ fixed with $\beta \geq 2$, and a real number $x$ with a finite number of digits $x_k$ with $0 \leq x_k < \beta$ for $k = -m, \cdots, n$. The positional representation of x with respect to the base $\beta$ is:

$$x_\beta = (-1)^s [x_n x_{n-1} \cdots x_1 x_0 . x_{-1} x_{-2} \cdots x_{-m}], \quad x_n \neq 0.$$

This relation actually means

$$x_\beta = (-1)^s \left( \sum_{k=-m}^{n} x_k \beta^k \right)$$

# Floating point representation of a number

**Example:** The conventional writing $x_{10} = 425.33$ denotes the number $x = 4 \cdot 10^2 + 2 \cdot 10 + 5 + 3 \cdot 10^{-1} + 3 \cdot 10^{-2}$, while $x_6 = 425.33$ would denote the real number $x = 4 \cdot 6^2 + 2 \cdot 6 + 5 + 3 \cdot 6^{-1} + 3 \cdot 6^{-2}$.

A rational number can have a finite number of digits in a base and an infinite number of digits in another base. For example, the fraction $1/3$ has infinite digits in base 10, being $x_{10} = 0.\bar{3}$, while it has only one digit in base 3, being $x_3 = 0.1$.

# Floating point representation of a number

Any real number can be approximated by numbers having a finite representation, with as much accuracy as we want, as long as we can increase the number of digits enough.
Basis in use:

- 2 (binary): 0 and 1 (called *bits*).
- 10 (decimal): The human basis.
- 16 (hexadecimal): $0, 1, \cdots, 8, 9, A, B, \cdots, E, F$.

# Floating point representation of a number

Computers use the *floating-point representation* which is given by

$$x = (-1)^s \cdot (0.a_1 a_2 \cdots a_t) \cdot \beta^e = (-1)^s \cdot m \cdot \beta^{e-t}$$

where:

- $s$ will give the sign.
- $\beta$ is the basis.
- $m = a_1 a_2 \cdots a_t$ an integer number called mantissa such that $0 \leq m \leq \beta^{t-1}$ and $0 \leq a_i \leq \beta - 1$).
- $t \in \mathbb{N}$ is the number of significant digits.
- $e$ is the exponent ($L \leq e \leq U$).

The number zero has a separate representation.

# Floating point representation of a number

Let us define the set of floating-point numbers with $t$ significant digits, base $\beta \geq 2, 0 \leq a_i \leq \beta - 1$, and range $(L, U)$ with $L \leq e \leq U$:

$$\mathbb{F}(\beta, t, L, U) = \{0\} \cup \left\{ x \in \mathbb{R} : x = (-1)^s \beta^e \sum_{i=1}^{t} a_i \beta^{-i} \right\}$$

In order to force uniqueness in the representation of the number $x$ we can impose $a_1 \neq 0$ and $m \geq \beta^{t-1}$. This representation of $x$ is called normalized.

For example, $x = 1$, would admit the representations $0.1 * 10^1$, $0.0001 * 10^4$, etc...but only one normalized representation 1.

# Floating point representation of a number

It is evident that if $x \in \mathbb{F}$, then $-x$ is also in $\mathbb{F}$, and that $\mathbb{F}$ has the following bounds:

$$x_{min} = \beta^{L-1} \leq x \leq \beta^{U}(1 - \beta^{-t}) = x_{max}$$

The cardinality (number of elements) of $\mathbb{F}$ is
$card\ \mathbb{F} = 2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$.

# Floating point representation of a number

**Example:** The positive numbers in the set $\mathbb{F}(2, 3, -1, 2)$ are:

$$
\begin{array}{llll}
(0.111)\cdot 2^2 = \frac{7}{2}, & (0.110)\cdot 2^2 = 3, & (0.101)\cdot 2^2 = \frac{5}{2}, & (0.100)\cdot 2^2 = 2, \\
(0.111)\cdot 2 = \frac{7}{4}, & (0.110)\cdot 2 = \frac{3}{2}, & (0.101)\cdot 2 = \frac{5}{4}, & (0.100)\cdot 2 = 1, \\
(0.111) = \frac{7}{8}, & (0.110) = \frac{3}{4}, & (0.101) = \frac{5}{8}, & (0.100) = \frac{1}{2}, \\
(0.111)\cdot 2^{-1} = \frac{7}{16}, & (0.110)\cdot 2^{-1} = \frac{3}{8}, & (0.101)\cdot 2^{-1} = \frac{5}{16}, & (0.100)\cdot 2^{-1} = \frac{1}{4}.
\end{array}
$$

They are included between $x_{min} = \beta^{L-1} = 2^{-2} = 1/4$ and $x_{max} = \beta^{U}(1 - \beta^{-t}) = 2^2(1 - 2^{-3}) = 7/2$.

# Floating point representation of a number

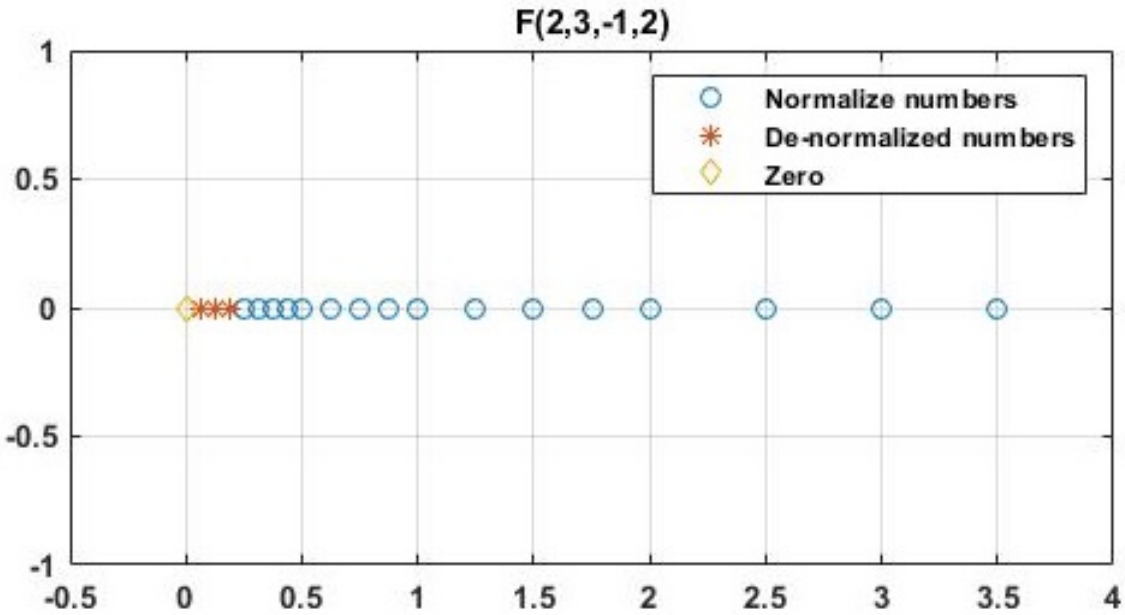
As a whole, we have $(\beta-1)\beta^{t-1}(U-L+1)=(2-1)2^{3-1}(2+1+1)=16$ strictly positive numbers. Their opposites must be added to them, as well as the number zero. We notice that when $\beta=2$, the first significant digit in the normalized representation is necessarily equal to 1 and thus it may not be stored in the computer (in such an event, we call it hidden bit). When considering also the positive de-normalized numbers, we should complete the above set by adding the following numbers:

$$(.011)_2\cdot 2^{-1}=\frac{3}{16},\quad (.010)_2\cdot 2^{-1}=\frac{1}{8},\quad (.001)_2\cdot 2^{-1}=\frac{1}{16}.$$

According to what previously stated, the smallest de-normalized number is $\beta^{L-t}=2^{-1-3}=1/16$.

# Distribution of floating-point numbers

The floating-point numbers are not equally spaced along the real line, but they get dense close to the smallest representable number.




The spacing between a number $x \in \mathbb{F}$ and its next nearest $y \in \mathbb{F}$, where both $x$ and $y$ are non null, is at least $\beta^{-1}\epsilon_M|x|$ and at most $\epsilon_M|x|$, being $\epsilon_M = \beta^{1-t}$ the machine epsilon.

# Distribution of floating-point numbers.Standard formats

Nowadays the standard IEC559 endorses two formats for the floating-point numbers: a basic format, made by the system $\mathbb{F}(2, 24, -125, 128)$ for the single precision, and by $\mathbb{F}(2, 53, -1021, 1024)$ for the double precision, both including the de-normalized numbers, and an extended format, for which only the main limitations are fixed.

# Floating point operations

As previously stated, it is necessary to define on the set of machine numbers an arithmetic which is analogous, as far as possible, to the arithmetic in $\mathbb{R}$. Thus, given any arithmetic operation $\circ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ on two operands in $\mathbb{R}$ (the symbol $\circ$ may denote sum, subtraction, multiplication or division), we shall denote by $\boxdot$ the corresponding machine operation

$$\boxdot : \mathbb{R} \times \mathbb{R} \to \mathbb{F}, \quad x \boxdot y = fl(fl(x) \circ fl(y)).$$

# Floating point operations

From the properties of floating-point numbers one could expect that for the operations on two operands, whenever well defined, the following property holds: $\forall x, y \in \mathbb{F}, \exists \delta \in \mathbb{R}$ such that

$$x \boxdot y = (x \circ y)(1 + \delta) \tag{1}$$

with
$|\delta| \leq u$ (remember $u$ is the machine precision $u = \frac{1}{2}\beta^{1-t} = \frac{1}{2}\epsilon_M$).

In order for this to be satisfied when $\circ$ is the operator of subtraction, it will require an additional assumption on the structure of the numbers in $\mathbb{F}$, that is the presence of the so-called round digit (which is addressed at the end of this section). In particular, when $\circ$ is the sum operator, it follows that for all $x, y \in \mathbb{F}$

$$\frac{|x \boxplus y - (x + y)|}{|x + y|} \leq u(1 + u)\frac{|x| + |y|}{|x + y|} + u$$

so that the relative error associated with every machine operation will be small, unless x + y is not small by itself. An aside comment is deserved by the case of the sum of two numbers close in module, but opposite in sign. In fact, in such a case $x + y$ can be quite small, this generating the so-called cancellation errors.

**Example:** Let us work with $\mathbb{F}$ having $\beta = 10$ and $t = 2$. Let us subtract 1 and 0.99.
Without round digit we have:

$$\begin{array}{rcl} 10^1 \cdot 0.1 & & 10^1 \cdot 0.10 \\ 10^0 \cdot 0.99 & \rightarrow & \underline{10^1 \cdot 0.09} \\ & & 10^1 \cdot 0.01 \quad \rightarrow \quad 10^0 \cdot 0.10 \end{array}$$

With round digit:

$$\begin{array}{rcl} 10^1 \cdot 0.1 & & 10^1 \cdot 0.10 \\ 10^0 \cdot 0.99 & \rightarrow & \underline{10^1 \cdot 0.09\boxed{9}} \\ & & 10^1 \cdot 0.00\boxed{1} \quad \rightarrow \quad 10^0 \cdot 0.01 \end{array}$$

# Floating point operations

It is important to notice that, together with properties of standard arithmetic that are preserved when passing to floating-point arithmetic (like, for instance, the commutativity of the sum of two addends, or the product of two factors), other properties are lost. An example is given by the associativity of sum: it can indeed be shown that in general

$$x \boxplus (y \boxplus z) \neq (x \boxplus y) \boxplus z.$$

# Floating point operations

**Note:** The IEC559 standard defines a closed arithmetic on $\mathbb{F}$, this meaning that any operation on it produces a result that can be represented within the system itself. As an example, in the next table we report the results that are obtained in exceptional situations:

| Exception | Examples | Result |
|---|---|---|
| non valid operation | $0/0, 0 \cdot \infty$ | NaN |
| overflow | | $\pm\infty$ |
| division by zero | $1/0,$ | $\pm\infty$ |
| underflow | | subnormal numbers |

The presence of a NaN (Not a Number) in a sequence of operations automatically implies that the result is a NaN. General acceptance of this standard is still ongoing.

# Floating point operations

Round-off errors: Wikipedia.

In these links there are some examples of round-off disasters:
http://www-users.math.umn.edu/ arnold//disasters/
https://web.ma.utexas.edu/users/arbogast/misc/disasters.html