Numerical Methods: brief introduction to floating point numbers

Elena Celledoni

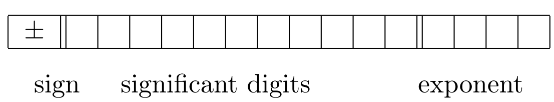
January 7, 2020

1 Representation of numbers on a computer: Floating point model

Computers have finite memory hence not every number can be represented exactly on a computer. For example $\sqrt{2}$, π have infinite number of digits hence it is impossible to represent them exactly in a computer. To fit in a computer, real numbers are approximated via the **floating point model**:

- binary system is used: $r = \pm(\alpha_k\, 2^k + \alpha_{k-1}\, 2^{k-1} + \alpha_{k-2}\, 2^{k-2} + \cdots + \alpha_0\, 2^0)$ where $\alpha_0, \ldots, \alpha_k \in \{0, 1\}$, $\alpha_k \neq 0$.
- a **fixed** amount of memory is allocated to represent each number:

$$\begin{aligned} r &= \pm\, 0.\, \alpha_k\, \alpha_{k-1}\, \ldots \alpha_{k-m-1}\, \alpha_{k-m}\, \ldots\, \alpha_0 \quad \cdot 2^{k+1} \\ fl(r) &= \pm\, 0.\, \alpha_k\, \alpha_{k-1}\, \ldots \alpha_{k-m-1}\, \tilde{\alpha}_{k-m} \quad \cdot 2^{E} \end{aligned}$$

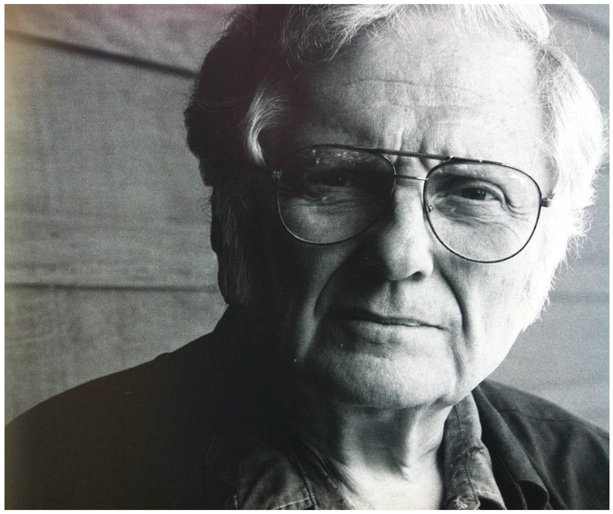


2 Floating point numbers

A *bit* is a basic unit of information in information theory, the name bit comes from *binary digit*.

Double precision IEEE 745

1 bit	52 bits	11 bits
sign	significant digits	exponent



Kahan, Turing award in 1989, was the primary architect behind the IEEE 754-1985

Some concepts and definitions:

- **Rounding**: is a procedure for determining the floatingpoint counterpart $fl(r)$ of a real number r. An alternative approach is called chopping.
- **Roundoff error** is $r - fl(r)$.
- **Machine epsilon**: ϵ is the smallest floating point number such that

$$1 + \epsilon \neq 1$$

  in the computer.
- **Loss of significant digits**: loss of precision due to subtraction of floating point numbers very close to each other.

2.1 Rounding

Rounding is the most used method to approximate real numbers in a computer. Assume $b = 10$. Given

$$0.d_1\, d_2 \ldots d_p\, d_{p+1} \ldots d_{p+k} \cdot b^e$$

rounding to p-digits gives

$$0.d_1\, d_2 \ldots d_p\, d_{p+1} \ldots d_{p+k} \cdot b^e = \begin{cases} 0.d_1\, d_2 \ldots d_p \cdot b^e & \text{if} \quad 0.d_{p+1} \ldots d_{p+k} < \frac{1}{2} \\ 0.d_1\, d_2 \ldots \tilde{d}_p \cdot b^e & \text{if} \quad 0.d_{p+1} \ldots d_{p+k} = \frac{1}{2} \quad \tilde{d}_p \text{ nearest even digit to } d_p.d_{p+1} \ldots d_{p+k} \\ 0.d_1\, d_2 \ldots \hat{d}_p \cdot b^e & \text{if} \quad 0.d_{p+1} \ldots d_{p+k} > \frac{1}{2} \quad \hat{d}_p = d_p + 1 \end{cases}$$

3 Loss of significant digits

Given the two real numbers

$$x = 0.3721478693$$

$$y = 0.37202300572$$

their difference is

$$x - y = 0.0001248121$$

We perform **rounding** at 5 digits, this gives

$$fl(x) = 0.37215$$

$$fl(y) = 0.37202$$

now the difference of the two floating point numbers is

$$fl(x) - fl(y) = 0.00013$$

in memory we can store 5 digits for $fl(x) - fl(y)$ but we really know only two of them, the others are lost.

4 Avoid propagation of roundoff error

Stability: study how the error propagates due to perturbations in the initial data.

- stability of the problem
- stability of the algorithm

Example 4.1 **Problem**: *find* x *such that* $ax + b = c$ *where* a, b, c *are given numbers and* $a \neq 0$.

Algorithm 1:

1. *divide by* a: $x + \frac{b}{a} = \frac{c}{a}$;
2. *subtract* $\frac{b}{a}$: $x = \frac{c}{a} - \frac{b}{a}$

Algorithm 2:

1. *subtract* b: $ax = c - b$;
2. *divide by* a $x = \frac{c-b}{a}$

Stability of the problem answers the question: What happens to the solution of $ax + b = c$ if $a \to a(1 + \delta_a)$, $b \to b(1 + \delta_b)$, $c \to c(1 + \delta_c)$?

Stability of the algorithm answers the question: What happens to the output of the algorithm if $a \to a(1 + \delta_a)$, $b \to b(1 + \delta_b)$, $c \to c(1 + \delta_c)$?

5 Stability and condition numbers

A problem is stable when the relative error in the output solution is of the same size of the relative error in the input data. Given a stable problem only if we choose a stable algorithm to solve it we get errors in the output which are of the same size as the errors in the input.

Definition 5.1 *Condition numbers are constants used to bound the amplification of the relative error in the output by means of the relative error in the input.*

Condition numbers are useful for quantifing the stability of a problem as well as the stability of an algorithm.

Example 5.2 (Stability of the arithmetic operation "+") *Let* $x > 0$ *and* $y > 0$ *real. Let* $fl(x) = x(1 + \delta_x)$, $fl(y) = y(1 + \delta_y)$ *with* $|\delta_x| \leq \epsilon$ *and* $|\delta_y| \leq \epsilon$. *Look at the relative error:*

$$\left| \frac{x + y - (fl(x) + fl(y))}{x + y} \right| = \left| \frac{x + y - (x + x\delta_x + y + y\delta_y)}{x + y} \right|$$

$$= \left| -\frac{x}{x + y}\delta_x - \frac{y}{x + y}\delta_y \right| \leq C \cdot \bar{\delta}$$

where $C = \max\{\frac{x}{x+y}, \frac{y}{x+y}\}$ *and* $\bar{\delta} = 2 \cdot \max\{|\delta_x|, |\delta_y|\} \leq 2\epsilon$. *Addition of positive numbers is a stable operation.* C *here is the condition number.*