What can go wrong:

- modeling errors
- discretization and truncation errors
- human error
- **roundoff and data errors**

Computers represent numbers using *floating point number systems*, $\mathbf{F}(\beta, k, m, M)$, characterized by

- β: the base
- k: the number of digits in the base β expansion
- m: the minimum exponent
- M: the maximum exponent

$$\mathbf{F}(\beta, k, m, M) := \{\pm(0.b_1 b_2 ... b_k)_\beta \times \beta^e \text{ with } m \leq e \leq M\}$$

Terminology:

- $b_1 b_2 ... b_k$ is called the **mantissa**
- e is called the **exponent**
- If $b_1 \neq 0$, or else $b_1 = b_2 = ... = b_k = 0$, $\mathbf{F}(\beta, k, m, M)$ is said to be **normalized**

2.1 Floating Point Arithmetic

Example:

$$\mathbf{F}(10,1,0,1) = \{\pm(0.b_1)_{10} \times 10^e \text{ with } 0 \le e \le 1\}$$
$$= \{0, \pm 0.1, \pm 0.2, ..., \pm 0.9, \pm 1, \pm 2, ..., \pm 9\}$$

Properties:

- The number of elements of (normalized) $\mathbf{F}(\beta, k, m, M)$ is $1 + 2(\beta - 1)\beta^{k-1}(M - m + 1)$.
- The largest positive number of (normalized) $\mathbf{F}(\beta, k, m, M)$ is $(0.\beta - 1\beta - 1...\beta - 1)_\beta \times \beta^M = (1 - \beta^{-k})\beta^M$.
- The smallest positive number of (normalized) $\mathbf{F}(\beta, k, m, M)$ is $(0.10...0)_\beta \times \beta^m = \beta^{m-1}$.

A number that has a magnitude outside the above computer range is called an **underflow** or an **overflow**.

2.1 Floating Point Arithmetic

Most computers use the binary system ($\beta = 2$). The two binary digits 0 and 1 are usually called **bits**, and the fixed-length group of binary bits is called a **computer word**.

Example: the floating-point number system of a 32-bit word length microcomputer. The internal representation of a word is as following:

sign (1 bit)	exponent (7 bits)	normalized mantissa (24 bits)

- the leftmost bit is used for the sign of the number ($0 \to +$ and $1 \to -$)
- the next seven bits represent the exponent, with the first bit used for its sign
- the final 24 bits represent the normalized mantissa

2.2 Roundoff Errors

$y = \pm(0.b_1 b_2 ... b_k b_{k+1} ...)_\beta \times \beta^e$ with $b_1 \neq 0$ and $m \leq e \leq M$.
Denote by $fl(y) \in \mathbf{F}(\beta, k, m, M)$ the *floating point equivalent* of y.
There are two natural ways to define $fl(y)$:

- *chopping* the number, i.e. $fl_{chop}(y) = \pm(0.b_1 b_2 ... b_k)_\beta \times \beta^e$;
- *rounding* the number, i.e.

$$fl_{round}(y) = \begin{cases} \pm(0.b_1 b_2 ... b_k)_\beta \times \beta^e & \text{if } b_{k+1} < \beta/2 \\ \pm[(0.b_1 b_2 ... b_k)_\beta + \beta^{-k}] \times \beta^e & \text{if } b_{k+1} \geq \beta/2. \end{cases}$$

Definition. The error introduced by converting a real number y to its floating point equivalent $fl(y)$ is called *roundoff error*.

- Absolute roundoff error:

$$|fl_{chop}(y) - y| \leq \beta^{e-k}, \quad |fl_{round}(y) - y| \leq \frac{\beta^{e-k}}{2}.$$

- Relative roundoff error:

$$\frac{|fl_{chop}(y) - y|}{|y|} \leq \beta^{1-k}, \quad \frac{|fl_{round}(y) - y|}{|y|} \leq \frac{\beta^{1-k}}{2}.$$

2.2 Roundoff Errors

Definition. The *machine precision* is given by

$$u = \begin{cases} \beta^{1-k}, & \text{chopping} \\ \frac{1}{2}\beta^{1-k}, & \text{rounding,} \end{cases}$$

where β is the base and k is the number of digits in the implemented floating point number system.

Computers perform calculations within their f.p.n.s.:

$$x @_{fl} y = fl(fl(x) @ fl(y)),$$

where @ represents a binary arithmetic operators (e.g., +, -, x, /). Floating point arithmetic does not satisfy many of the properties of real arithmetic, such as (addition) associativity and distributivity.

Floating Point Calculations

Example: In 4 decimal digit rounding arithmetic:

$$(0.1329 + 1.543) + 23.21 = 1.676 + 23.21 = 24.89 \quad \text{but}$$

$$0.1329 + (1.543 + 23.21) = 0.1329 + 24.75 = 24.88$$

$$(0.1351 + 23.21) \times 1.543 = 23.35 \times 1.543 = 36.03 \quad \text{but}$$

$$0.1351 \times 1.543 + 23.21 \times 1.543 = 0.2085 + 35.81 = 36.02$$

Accumulation of Roundoff Errors

$$\begin{aligned} x @_{fl} y - x @ y &= fl(fl(x) @ fl(y)) - x @ y \\ &= [fl(fl(x) @ fl(y)) - fl(x) @ fl(y)] + [fl(x) @ fl(y) - x @ y] \\ &= \text{introduced error} + \text{propagated error} \end{aligned}$$

The introduced error is small; it is bounded by machine precision. Unfortunately the propagated error can be large.

Floating Point Number Systems: The IEEE Standard

A widely used internal representation of numbers in almost all new computers is the **IEEE Standard**.

- The single-precision format

$$\text{Floating-point number} = (-1)^s \, x \, (1.f)_2 \, x \, (2^{c-127})_{10}$$

  uses 32 bits:
  - first bit is reserved for the sign bit s ($s = 0 \to +$, $s = 1 \to -$);
  - next eigth bits are reserved for the (biased) exponent c;
  - the remaining 23 bits are used for the fractional part f of the normalized mantissa.
- The double-precision format

$$\text{Floating-point number} = (-1)^s \, x \, (1.f)_2 \, x \, (2^{c-1023})_{10}$$

  uses 64 bits:
  - first bit is reserved for the sign bit s ($s = 0 \to +$, $s = 1 \to -$);
  - next 11 bits are reserved for the (biased) exponent c;
  - the remaining 52 bits are used for the fractional part f of the normalized mantissa.

2.3 Truncation Error

Round-off errors arise in considering the floating point equivalent of numbers. In contrast, the **truncation error** terminates a process, usually related to considering only a finite number of terms of infinite series or sequences. An important tool is the Taylor series expansion of $f(x)$ about a point x_0:

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1},$$

for some $\xi = \xi(x)$ between x_0 and x.

Examples:

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \frac{e^{\xi}}{(n+1)!}x^{n+1}$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + \frac{\sin^{(n+1)}(\xi)}{(n+1)!}x^{n+1}$$

2.4 Interval Arithmetic

Definition. Let $\star$ be one of the symbols $\{+, -, \cdot, \div\}$. If A and B are intervals, we define arithmetic operations on intervals by

$$A \star B = \{x \star y |\ x \in A,\ y \in B\}$$

except that we do not define $A \div B$ if $0 \in B$.

If $A = [a_1, a_2]$ and $B = [b_1, b_2]$, then
$A + B = [a_1 + b_1, a_2 + b_2]$
$A - B = [a_1 - b_2, a_2 - b_1]$
$A \cdot B = [\min\{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\},\ \max\{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\}]$
$A \div B = [a_1, a_2] \cdot [1/b_2, 1/b_1]$ provided that $0 \notin [b_1, b_2]$.