Note Floating-point numbers have limitations:

- limitations on significand
- limitations on exponent
  - ↳ **overflow** ⇒ exponent is too large (positive)
    → set as Inf
  - ↳ **underflow** ⇒ exponent is too large (negative)
    → set as zero

Rmk. With proper scaling, overflow / underflow can often be avoided.

1.2 Errors

Suppose $p^* \in \mathbb{R}$ is an approximation of $p \in \mathbb{R}$.

- **absolute error:** $e_a(p, p^*) = |p - p^*|$
- **relative error:** $e_r(p, p^*) = \dfrac{|p - p^*|}{|p|}$ for $p \neq 0$

Error bounds

- **absolute error bound:** $e_a(p, p^*) \leq \varepsilon_a(p, p^*)$
- **relative error bound:** $e_r(p, p^*) \leq \varepsilon_r(p, p^*)$

Rmk. Often we can only obtain a bound on error produced by algorithms.

Ways to reduce errors in finite digit precision:

I. Avoid subtraction of 2 nearly equal numbers

- reason: causes cancellation of significant digits (catastrophic cancellation)

Ex. 1 Given 2 numbers x and y, with $x > y$ and k-digit representation:

Then $fl(x) = 0.d_1 d_2 \ldots d_p \alpha_{p+1} \alpha_{p+2} \ldots \alpha_k \times 10^n$

$$fl(y) = 0.d_1 d_2 \dots d_p \beta_{p+1} \beta_{p+2} \dots \beta_k \times 10^n$$

$$\Rightarrow \quad fl(fl(x) - fl(y)) = 0.\underbrace{00\dots0}_{p \text{ times}} \sigma_{p+1} \sigma_{p+2} \dots \sigma_k \times 10^n$$

$$= 0.\sigma_{p+1} \sigma_{p+2} \dots \sigma_k \times 10^{n-p}$$

Then $fl(fl(x) - fl(y))$ has k-p significant digits (loss of accuracy).

Ex. 2 $\quad f(x) = \dfrac{1-\cos x}{x^2}$

Fact $\quad 0 < f(x) \le \frac{1}{2}$, $\quad \lim_{x\to 0} f(x) = \lim_{x \to 0} \dfrac{\sin(x)}{2x}$ $\quad$ (L' Hospital)

$$= \lim_{x\to 0} \frac{\cos(x)}{2} = \frac{1}{2}$$

On MATLAB: $\quad f(1.2 * 10^{-8}) \approx 0.77098$

$f(1.2 * 10^{-9}) = 0 \quad$ NOT $\approx 1/2$!

Remedy: use trig identity $\cos x = 1 - 2\sin^2(\frac{x}{2})$

$$\rightarrow \quad f(x) = \frac{1-\cos x}{x^2} = \frac{2\sin^2(\frac{x}{2})}{x^2}$$

(no longer requires subtraction in numerator)

II. Avoid division by small numbers or multiplying by large numbers

- reason: causes overflow

Ex. Consider $\quad c = \sqrt{a^2 + b^2}$

If $a = 10^{170}$, $b = 1$, then correctly rounded solution: $c = 10^{170}$

However, with double-precision arithmetic:

$$a^2 = \text{Inf}, \quad a^2 + b^2 = \text{Inf} + 1 = \text{Inf}, \quad c = \sqrt{\text{Inf}} = \text{Inf}$$

Remedy: scale the data

$$c = s\sqrt{\left(\frac{a}{s}\right)^2 + \left(\frac{b}{s}\right)^2}, \quad \text{where } s = \max\{|a|, |b|\}$$

Here, $s = 10^{170}$ and $c = 10^{170}\sqrt{(1)^2 + (\frac{b}{s})^2} = 10^{170}$

↳ underflow to 0 (ok)

III. Reduce the number of arithmetic computations $(+, -, \cdot, \div)$

reason: more computation → more rounding errors

Ex. Evaluate $f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ at $x = 4.71$

mult. 2 2 1

add. 1 1 1 } 8

Now consider nested formulation:

$$f(x) = (x^3 - 6.1x^2 + 3.2x) + 1.5$$
$$= (x^2 - 6.1x + 3.2)x + 1.5$$
$$= ((x - 6.1)x + 3.2)x + 1.5$$

2 multiplications + 3 additions

→ Horner's Method (Ch.2)

Truncation error: truncate infinite sum by finite sum

Thm. Taylor's Theorem

$$f(x) = \underbrace{f(x^*) + f'(x^*)(x - x^*) + \frac{f''(x^*)}{2!}(x - x^*)^2 + \dots + \frac{f^{(n)}(x^*)}{n!}(x - x^*)^n}_{p_n(x)} + \underbrace{\frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x^*)^{n+1}}_{R_n(x)}$$

where ξ is between x and x^*

Truncation error: $R_n(x) = f(x) - p_n(x)$