Stability. Condition number.

Computational problems and algorithms

- Stability
- Condition number

A computational problem: maps inputs $x \in \mathcal{X}$ to results $y \in \mathcal{Y}$.

A problem is well-defined, iff

- the solution $y \in \mathcal{Y}$ exists for all $x \in \mathcal{X}$
- this solution is unique
- this solution is *stable* for small perturbations of inputs: (small perturbations of inputs produce small changes in outputs)

Stability: rank of a matrix

Example: Computation of a matrix rank is unstable.

$$\mathbf{M} = \begin{bmatrix} 2 & 0 \\ 0 & \epsilon \end{bmatrix}$$

$\forall \epsilon > 0$, $\operatorname{rank} \mathbf{M} = 2$, while $\operatorname{rank} \mathbf{M}(\epsilon = 0) = 1$.

Stability: derivatives

Example: computation of derivatives is unstable.

Consider $g(x) = f(x) + A \sin \omega x$ with $A \ll 1$.
Then,

$$g'(x) = f'(x) + A\,\omega \cos \omega x$$

For $\omega \gg 1$,

$$\max_x \left| g'(x) - f'(x) \right| \gg 1$$

Stability: polynomials

Consider the equation

$$(x - 1)^8 = 0 \ .$$

Change the lowest order coefficient by 10^{-8}

$$(\bar{x} - 1)^8 = 10^{-8}$$

so that $|x - \bar{x}|/x = 0.1$

Condition number

Let the error in input Δx, changed the result by Δy. Then the *condition number* R

$$\Delta y \leqslant R \Delta x$$

- If $R \gg 1$, the problem is *poorly conditioned*.
- Roughly, a computation loses $\sim \log R$ significant figures.

Stability of a computational algorithm

An algorithm of computing the output y given input x is stable iff

- the result y is stable w.r.t. small perturbations in x
- the result y is *computationally stable*

Arithmetics is subject to round-off, characterized by e.g. the machine epsilon, ϵ. An algorithm is *computationally stable* if computational errors tend to zero for $\epsilon \to 0$.

Stability of numerical algorithms

Example: stable and unstable recursion

Example: an unstable recursion

Consider

$$I_n = \int_0^1 x^n \, e^{1-x} \, dx \; , \qquad n = 1, 2, \ldots$$

Integrating by parts, we find

$$I_n = n \, I_{n-1} - 1 \; , \qquad n \geqslant 1$$

and $I_0 = e - 1$.

Compute the values of the integrals for a range of n using the upwards recursion.

Use fixed-width arithmetics with one decimal place after the decimal dot.

Example: an unstable recursion

$$I_0 = e - 1 \; , \qquad I_n = n\, I_{n-1} - 1 \; , \qquad n \geqslant 1$$

$$\begin{aligned} I_0 &= e - 1 \approx 1.7 \\ I_1 &= 1 \times I_0 - 1 \approx 0.7 \\ I_2 &= 2 \times I_1 - 1 \approx 0.4 \\ I_3 &= 3 \times I_2 - 1 \approx 0.2 \\ I_4 &= 4 \times I_3 - 1 \approx -0.2 \end{aligned}$$

...which is nonsense: obviously, $I_n > 0$ for all $n \geqslant 0$.

N.B. The problem is not the truncation error itself. The problem is that the algorithm is *poorly conditioned*.

Downwards recursion

Solution: use downwards recursion.

Note that $I_n \to 0$ for $n \to \infty$. Rewrite the recursion identically

$$I_{n-1} = \frac{I_n + 1}{n}$$

Start from large enough n (e.g., $n_0 = 50$). Set $I_{n_0} = 0$ and recurse downwards.

The initial truncation error tends to zero.

The downwards recursion is well conditioned.