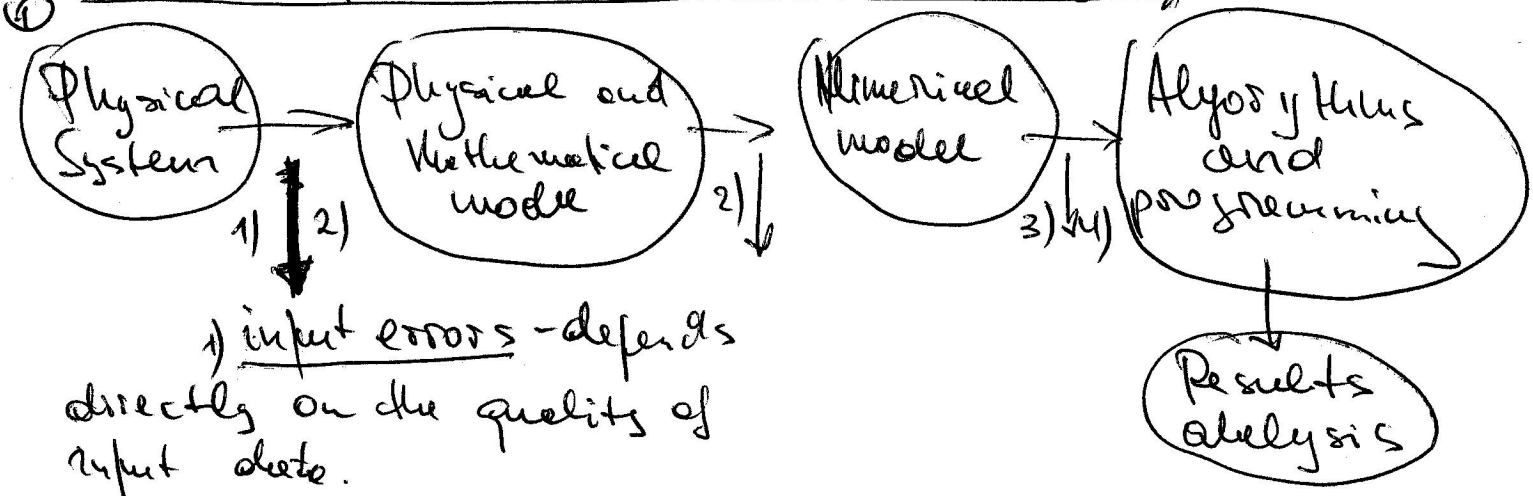
Numerical Methods I. – Lecture 01.

Intro.: Computer Numbers. / Error Analysis



1) input errors – depends directly on the quality of input data.

2) the approximate problem must be formulated in such a way that a) it can be solved numerically and b) its solution does not excessively differ from one for the given "exact" problem.

3) all numerical models are accompanied by "procedural" errors, e.g., truncation error, discretization error;
along with

4) propagation error – the error in the output of a problem, which is caused by the error in the input data.

② Definition of Errors:

DFN: If A is an approximate value of a number a, then the difference $\boxed{\Delta_a = a - A}$ is called the true error of A, while its absolute value

$\boxed{|\Delta_a| = |a - A|}$ is called the absolute error of A.

The number a is usually unknown, so one is interested in an upper bound $\delta_a > 0$ instead for the absolute error $|\Delta a|$ of A, i.e., $|\Delta_a| \leq \delta_a$.

DF2: If $|\Delta a|$ is the absolute error of an approximate value A of a, and if $\delta_a > 0$ is an upper bound for $|\Delta a|$, i.e., $|\Delta a| \leq \delta_a$, then δ_a is called an error bound for the absolute error of A.

Since $|\Delta a| = |a - A| \leq \delta_a \Rightarrow$ one can ensures, that

$$\boxed{A - \delta_a \leq a \leq A + \delta_a}.$$

DF3: if $|\Delta a|$ is the error of an approximation A for the a, where both a and A are nonzero, then the ratios:

$$\boxed{|\varepsilon_a| := \frac{|\Delta a|}{|a|} = \frac{|a - A|}{|a|}} \text{ for } a \neq 0 \text{ or}$$

$$\boxed{|\varepsilon_a| := \frac{|\Delta a|}{|A|} = \frac{|a - A|}{|A|}} \text{ for } A \neq 0 \text{ is called the true relative error of } A.$$

③ Decimal representation of numbers

DF4: for every integer a, there is exactly one decimal representation in powers of ten of the form:

$$\boxed{a = \pm (a_n 10^n + \cdots a_1 10^1 + a_0 10^0) = \pm \sum_{k=0}^{n} a_k 10^k},$$

with coeffs. $a_k \in \{0,1,2,3,4,5,6,7,8,9\}$, and $n > 0$.

Then decimal representation is written down in terms of digits a_k, as

$$\boxed{a = \pm a_n a_{n-1} \cdots a_1 a_0}$$

DF5: If a is not integer, then a has representation in decimal powers of ten with at least one term of the form: $\boxed{\delta_n 10^{-n}}$, n $K>0$ and integer, and

$$\boxed{\delta_n \in \{1,2,3,4,5,6,7,8,9\}} \quad \square$$

To separate the nonnegative powers of ten from the negative ones, one inserts a "decimal point" between a_0 and δ_1 in the "decimal fraction" of a:

$$a = \pm\left(\sum_{k=0}^{n} a_k 10^k + \sum_{k=1}^{\infty} \delta_k 10^{-k}\right) = \pm a_n a_{n-1} \cdots a_1 a_0 . \underbrace{\delta_1 \delta_2 \delta_3 \ldots}_{\text{are called decimals}}$$

DF6: If for a given number a there is an integer j with $\delta_j \neq 0$, while $\forall j' > j \; \delta_{j'} = 0$, then a is said to have a finite decimal representation, otherwise a has an infinite decimal repr.

DF7: All digits of decimal repr., beginning with the first nonzero digit, are called significant digits

DF8: Normalized decimal floating point representation

Every real number $a \neq 0$ can be represented in the form:

$$\boxed{a = \pm(.d_1 d_2 \ldots d_z d_{z+1} \ldots) \cdot 10^k = \pm m \cdot 10^k, \text{ with } 0.1 \leq m < 1}, \quad k \in \mathbb{Z} \; (\text{integers positive and negative})$$

here m is called "mantissa",
k is called "exponent".

In computer we always have a finite number of digits to represent "mantisses" m and exponents k.

$m = 7$ to 8 digits
$|k| \leq 100$ } typical.

If number a is to large, i.e., $a > .9999999 \cdot 10^{100}$ (1 2 3 4 5 6 7)
one has overflow => fatal

If number a is to small, i.e. $a < .9999999 \cdot 10^{-100}$
one has under flow => not fatal $\boxed{a = 0.0}$

In any case, one truncate a number a to its closest computer representation. (m=7)

Ex. 1 $525.48287 \cdot 10^5$ $.5\,2\,5\,4\,8\,2\,9\ 10^{0\,0\,8}$

$.525 \cdot 10^3 10^5$

$0.52548287 \cdot 10^3 10^5 =$

$0.5254828|7 \cdot 10^8$
(1 2 3 4 5 6 7)

$0.5254829 \cdot 10^8$ → rounding.

truncation error

Def: 8 Correct decimals

A number a in the decimal form with more than t decimals is correctly rounded to the machine number A with t decimals, if $\boxed{|a - A| \leq \frac{1}{2} \cdot 10^{-t}}$ or $\boxed{\Delta_a \leq \frac{1}{2} \cdot 10^{k-t}}$

Correct digits:

If A is obtained by correct rounding to t decimals then digits in A that are associated with the power 10^{-t} or higher are called correct digits of A. Leading zeros are omitted.

② gives us the absolute error of A, then the relative error ε_a is if $a \neq 0 \Rightarrow$

$$|\varepsilon_a| = \frac{|a-A|}{|a|} \leq \frac{|a-A|}{|m| \cdot 10^{k-1}} \leq \frac{5 \cdot 10^{k-t}}{|m| \cdot 10^{k-1}} \Rightarrow$$

$$\boxed{|\varepsilon_a| = \frac{|a-A|}{|a|} \leq 5 \cdot 10^{-t}}.$$

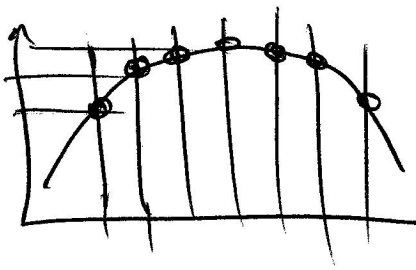
DF 3: If computer uses floating point numbers with t digits mantissas for power of 10, then

$\boxed{\rho = 5 \cdot 10^{-t}}$ is called machine constant or the elementary rounding error.

!! gives us the best accuracy possible on a given computer.

Consequence I: Discretization error

$y = f(x)$



1) while not any values of y and x are possible to represent on computer.

2) As amount of computer memory is finite.

Consequence II: leading to the concept of propagation error

~~$a+(b+c) = (a+b)+c$~~

~~$0.11+(0.38+1.25) = (0.11+0.38)+1.25$~~

~~$0.11+0.1$~~

~~$0.1+(0.4+1.3) = 0$~~

!! The associated "floating-point" operations need not to be associative:

Ex: $a = 0.23371258 \cdot 10^{-4}$ $\quad t = 8$

$b = 0.33678429 \cdot 10^{2}$

$c = -0.33677811 \cdot 10^{2}$

$a + (b + c) = 0.23371258 \cdot 10^{-4} + 0.61800000 \cdot 10^{-3}$

$= 0.64137126 \cdot 10^{-3}$

as $b + c = 0.00000618 \cdot 10^{2} = 0.618 \cdot 10^{-5} \cdot 10^{2}$

$(a + b) + c = 0.33678452 \cdot 10^{2} - 0.33677811 \cdot 10^{2}$

$= 0.64100000 \cdot 10^{-3}$

as $a + b = 0.00000023 \cdot 10^{2} + 0.33678429 \cdot 10^{2} =$

$= 0.33678452 \cdot 10^{2}$

Let all computational procedures are represented of a real-valued function f, so output data is obtained as:

$$y = f(x_1, x_2 \dots x_n) := f(\vec{x}) \; ; \text{ with } \vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

↑ input data.

If the input x_i is known only approximately as X_i, then f will yield an app. output

$$Y = f(X_1, X_2 \dots X_n) := f(\vec{X}) \; ; \quad \vec{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}$$

Theorem:

$\Delta_y = y - Y$ gives "propagation error" for a given $\Delta_x = x - X$.

$$\Delta_y = y - Y = f(\vec{x}) - f(\vec{X}) = \sum_{n=1}^{n} \frac{\partial f(\vec{x})}{\partial x_i} \Delta_{x_i} .$$

↖ Taylor series

if $y = f(x) \neq 0$, and $x_i \neq 0$ for $\forall i$, then the relative propagation error is

$$\varepsilon_y := \frac{\Delta y}{y} = \sum_{i=1}^{n} \frac{x_i}{f(\vec{x})} \frac{\partial f(\vec{x})}{\partial x_i} \varepsilon_{x_i}$$

$$\kappa_i = \frac{x_i}{f(\vec{x})} \frac{\partial f(\vec{x})}{\partial x_i} \rightarrow \text{amplification factor.}$$

The problem of computing $f(\vec{x})$ from $\vec{x}$ is a well-conditioned problem for algorithm f, if its all amplification factors a reasonably bounded.

Ex:
a) Ideally $|\kappa_i| \leq 1$

b) if $|\kappa_i| \leq 10^4$ for all i, the 4 digits of accuracy would be lost.

Propagation errors

1) product $y = f(x_1, x_2) = x_1 \cdot x_2$

$$\varepsilon_y = \varepsilon_{x_1} + \varepsilon_{x_2}; \quad \kappa_i \approx 1$$

2) division

$$f(x_1, x_2) = \frac{x_1}{x_2}; \quad x_2 \neq 0$$

$$\partial_{x_1} = \frac{1}{x_2}; \; \partial_{x_2} = -\frac{x_1}{x_2^2} \Rightarrow \kappa_1 \approx 1; \; \kappa_2 \approx -1$$

$$\varepsilon_y = \varepsilon_{x_1} - \varepsilon_{x_2}$$

3) sum: $f(x_1, x_2) = x_1 + x_2$

$$\partial_{x_1} = 1; \quad \varepsilon_y = \frac{x_1}{x_1 + x_2} \varepsilon_{x_1} + \frac{x_2}{x_1 + x_2} \varepsilon_{x_2} = \sum_{i=1}^{2} \kappa_i \varepsilon_{x_i}$$

1) x_1, x_2 have the same sign, then $f(x)$ is well-conditioned; since $0 \leq \kappa_{x_i} \leq \frac{x_i}{x_1+x_2} < 1$

2) x_1, x_2 have opposite sign and are nearly equal in size, "+" is badly conditioning as $x_1 + x_2 \to 0$.

"Cancellation effect"

~~Def/example~~

Def: Numerical Stability

Algorithms that avoid accumulated comp. errors $\Rightarrow$ numerically stable.