

# Fundamentals of numerical analysis

Today: floating point arithmetic  
& backward error analysis

Last time: Overview of the course: NLA

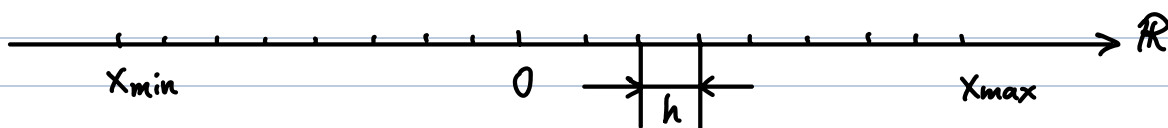
To do linear algebra on computers

First step: store numbers on computers & do arithmetic

Challenge:  $\mathbb{R}$  is unbounded and forms a continuum  
while computers are 'discrete' & finite memory

Idea 1 (Fixed point #s)

Discretize  $\mathbb{R}$  into equally spaced points



Denote the set of fixed point numbers

$$x = q B^{-n}, \quad x_{\min} \leq q B^{-n} \leq x_{\max}, \quad q \in \mathbb{Z}$$

On a (binary) computer,  $\pm \underbrace{1001}_{\substack{\uparrow \\ \text{sign} \\ m\text{-digits}}} . \underbrace{0110}_{\substack{\text{fraction part} \\ n\text{-digits}}}$

$$x = \pm \sum_{i=-n}^m \frac{k_i}{B^i}, \quad 0 \leq k_i \leq B-1$$

Nonzero fixed pt # range  $B^{-n} \leq |x| \leq B^{m+1} - B^{-n}$

Let  $f_i(\cdot)$  map  $\mathbb{R}$  to the nearest fixed point #

For  $x$  in the range,

$$f_i(x) = \underbrace{x + \delta}_{\text{absolute error is small}}, \quad |\delta| \leq h$$

cons: • Less suitable for representing very large / small #s

• Values can overflow / underflow easily

ex. On a binary computer with 1 integer digit and 2 fraction digits

$$f_i(0.25) = 0.25$$

$$f_i(0.5) = 0.5$$

$$\text{but } f_i(0.5 \times 0.25) = f_i\left(\frac{1}{8}\right) = 0$$

No significant digits

## Idea 2 (Floating point #s)

Mimics scientific notation  $1.25 \times 10^{-1}$

Floating point #'s

$$x = \pm \frac{m}{B^t} B^e$$

•  $t$ : precision

•  $B$ : base (usually  $B=2$  on a binary computer)

•  $e$ : exponent  $e_{\min} \leq e \leq e_{\max}$  (exponent range)

•  $m$ : fraction  $B^{t-1} \leq m \leq B^t - 1$

↑  
"normalized"  
ensure unique representation

"0" is a special case ( $m=0$ )

A more common way of expressing floating point # is

$$x = \pm B^e \times \left( \sum_{i=1}^t \frac{d_i}{B^i} \right) = \pm B^e \times .d_1 d_2 \dots d_t$$

each digit  $0 \leq d_i < B-1$  .  $d_1 \neq 0$  for normalized representation

- Decimal location "floats" depending on the size of #

Less easy to overflow / underflow

Range of nonzero floating point #s

$$B^{e_{\min}-1} \leq |x| \leq B^{e_{\max}} (1 - B^{-t})$$

example: IEEE 754 (1985, updated 2008)

	B	t	$e_{\min}$	$e_{\max}$	$e_{\max}$
single (FP32) prec.	2	24	-126	127	$2^{-24} \approx 5.96 \times 10^{-8}$
(FP64) double prec.	2	53	-1022	1023	$2^{-53} \approx 1.1 \times 10^{-16}$

↑  
 bits for fraction + 1    hidden bit  
(implicit digit  $d_1 \equiv 1$ )

Single precision: 32 bits = 1 + 8 + 23

double precision: 64 bits = 1 + 11 + 52

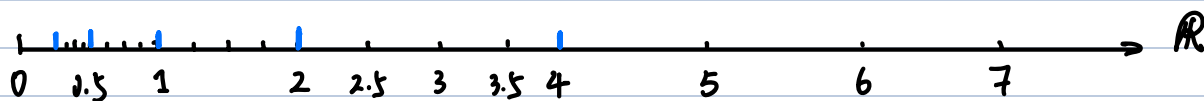
sign      exponent      fraction

Other precisions: FP8, FP16 (Half-prec.), single extended,

↑  
multiple format ...

double extended, ...

- floating point numbers are not equally spaced



If  $B=2$ ,  $t=3$ ,  $e_{\min} = -1$ ,  $e_{\max} = 3$

Floating point numbers:

$$2^3 \times .\underline{1}\underline{1}\underline{1} = 2^3 \times \left( \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} \right) = 7$$

$$2^3 \times .\underline{1}\underline{1}\underline{0} = 2^3 \times \left( \frac{1}{2} + \frac{1}{2^2} + \frac{0}{2^3} \right) = 6$$

$$2^3 \times .\underline{1}\underline{0}\underline{1} = 2^3 \times \left( \frac{1}{2} + \frac{0}{2^2} + \frac{1}{2^3} \right) = 5$$

$$2^3 \times .\underline{1}\underline{0}\underline{0} = 2^3 \times \left( \frac{1}{2} + \frac{0}{2^2} + \frac{0}{2^3} \right) = 4$$

$$2^3 \times .\underline{0}\underline{1}\underline{1} = 2^2 \times .\underline{1}\underline{1}\underline{1} = 2^2 \times \left( \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} \right) = 3.5$$

Set  $\frac{m}{B^t} = \frac{1}{2^2} + \frac{1}{2^3}$

$$m = 3 < m^{t-1}$$

not a normalized representation

$$2^2 \times .\underline{1}\underline{1}\underline{0} = 3.0$$

$$2^2 \times .\underline{1}\underline{0}\underline{1} = 2.5$$

$$2^2 \times .\underline{1}\underline{0}\underline{0} = 2.0$$

$$2^2 \times .\underline{0}\underline{1}\underline{1} = 2^1 \times .\underline{1}\underline{1}\underline{1} = 1.75$$

$\vdots$

- So how to quantify accuracy of floating point # ?  
machine epsilon (unit roundoff):

$\epsilon_{\text{mach}}$  = half distant from 1.0 to  
the next larger float

$$1.0 = \frac{B^{t-1}}{B^t} B \quad \text{next \#} = \frac{B^{t-1}+1}{B^t} B$$

$$\epsilon_{\text{mach}} = \frac{1}{2} B^{1-t}$$

$\epsilon_{\text{mach}}$  = relative error of rounding  $x \in \mathbb{R}$  to its nearest fp#

Let  $f(\cdot)$  map  $\mathbb{R}$  to the nearest floating point #

Thm For every  $x \in \mathbb{R}$  (in exponent range)

$$f(x) = x(1 + \underbrace{\delta}_{\text{relative error}}), \quad |\delta| \leq \epsilon_{\text{mach}}$$

Pf: w.l.o.g. assume that  $x > 0$

We write  $x = \mu \times B^{e-t}$ , where  $\beta^{t-1} \leq \mu < \beta^t$

$$x \in [y_1, y_2]$$

$$\text{where } y_1 = \lfloor \mu \rfloor B^{e-t}, \quad y_2 = \lceil \mu \rceil B^{e-t} = \frac{\lceil \mu \rceil}{B} B^{e-t+1}$$

thus

$$\left| \frac{f(x) - x}{x} \right| \leq \frac{1}{2} \left| \frac{y_2 - y_1}{x} \right| = \frac{1}{2} \frac{B^{e-t}}{\mu \times B^{e-t}} \leq \frac{1}{2} B^{1-t}$$



## • Floating point arithmetic

To carry out rounding analysis, we need to make some assumptions about the accuracy of the basic arithmetic operation

The most common assumptions are embodied in the following model

Let  $\mathbb{F}$  be the set of all floating point numbers

Let  $*$  be one of the operations  $+$ ,  $-$ ,  $\times$  or  $\div$

Let  $\oplus$  be its floating point analogue

Standard model  
(Fundamental Axiom of Floating Point Arithmetic)

$$\forall x, y \in \mathbb{F}, \quad x \oplus y = fl(x * y)$$

that is,  $\exists \delta$  with  $|\epsilon| \leq \epsilon_{mach}$  s.t.

$$x \oplus y = (x * y)(1 + \delta)$$

This model is valid for most computers, including IEEE standard arithmetic

example:  $f(x) = ((x - 0.5) + x) - 0.5 + x$

in exact arithmetic,  $f(\frac{1}{3}) = 0$

in double precision,  $f(x) \neq 0$ ,  $\forall x \in \mathbb{F}$

(\* Hint: Show that  $f(x) = 3x - 1$  for  $x \equiv f(x)$  near  $\frac{1}{3}$ .)

Other important points:

(See iJulia notebook)

- Input/output rounding
- Nonassociativity
- Catastrophic cancellation, etc.

# • Rounding error analysis

Consider the inner product

$$x^T y \quad , \quad x, y \in \mathbb{F}^n \quad \text{special case: summation}$$

Naïve summation algorithm

$$\begin{cases} S_1 = fl(x_1 y_1) \\ S_i = fl(S_{i-1} + fl(x_i y_i)) \quad i = 2, \dots, n \end{cases}$$

$$S_1 = x_1 y_1 (1 + \delta_1) \quad , \quad |\delta_1| \leq \epsilon_{mach}$$

$$S_2 = (S_1 + x_2 y_2 (1 + \delta_2)) (1 + \delta'_2) \quad |\delta_2|, |\delta'_2| \leq \epsilon_{mach}$$

$$= x_1 y_1 (1 + \delta_1) (1 + \delta'_2) + x_2 y_2 (1 + \delta_2) (1 + \delta'_2)$$

$$S_n = x_1 y_1 (1 + \delta_1) \prod_{i=2}^n (1 + \delta'_i) + \sum_{j=2}^n x_j y_j (1 + \delta_j) \prod_{i=j}^n (1 + \delta'_i)$$

$$|\delta'_i|, |\delta_i| \leq \epsilon_{mach}$$

Lemma: If  $|\delta_i| \leq \epsilon_{mach}$ , and  $n \epsilon_{mach} < 1$ , then

$$\prod_{i=1}^n (1 + \delta_i) = 1 + \theta_n$$

$$\text{with } |\theta_n| \leq \frac{n \epsilon_{mach}}{1 - n \epsilon_{mach}} =: \gamma_n \quad \leftarrow \text{linear in } n$$

Pf: By induction



By this Lemma, we obtain

$$S_n = x_1 y_1 (1 + \theta'_n) + \sum_{j=2}^n x_j y_j (1 + \theta_j)$$

$$\text{with } |\theta'_n| \leq \gamma_n, |\theta_j| \leq \gamma_{n-j+2}$$

$$\text{Let } \Delta x = (\theta'_1 x_1, \theta'_2 x_2, \dots, \theta'_n x_n)^T$$

$$\text{or } \Delta y = (\theta'_1 y_1, \theta'_2 y_2, \dots, \theta'_n y_n)^T$$

$$\text{then } s_n = (x + \Delta x)^T y = x^T (y + \Delta y)$$

$$\text{Note that } |\Delta x| \leq \gamma_n |x|, |\Delta y| \leq \gamma_n |y|, |x| = (|x_i|)_{i=1}^n$$

$$|x^T y - s_n| = |\Delta x^T y| \leq \gamma_n \underline{|x|^T |y|}$$

Remark 1: Better algorithm can do better  
For example,

$$1) \quad \left| \begin{array}{l} x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{n/2} \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}^{n/2} \end{array} \right.$$

$$\left| \begin{array}{l} \text{do } S_{n/2}^{(1)} \text{ for } x_1, y_1, \quad S_{n/2}^{(2)} \text{ for } x_2, y_2, \text{ then } s_{n/2}^{(1)} + s_{n/2}^{(2)} \end{array} \right.$$

$$\text{Error: } |f(s_{n/2}^{(1)} + s_{n/2}^{(2)}) - x^T y| \leq \gamma_{n/2+1} |x|^T |y|$$

(Hint: perform the induction as above)

2) The error can be further improved by partitioning into  $k$  pieces

$$\text{Error: } \text{Error} \leq \gamma_{n/k+K-1} |x|^T |y|$$

take  $k = \sqrt{n}$  minimize the dependence in  $n$

3) Use pairwise summation recursively can improve to

$$\text{Error: } \text{Error} \leq \gamma_{\lceil \log_2 n \rceil} |x|^T |y| \quad \uparrow \text{ exercise}$$

Remark 2: Worst case bound is often pessimistic

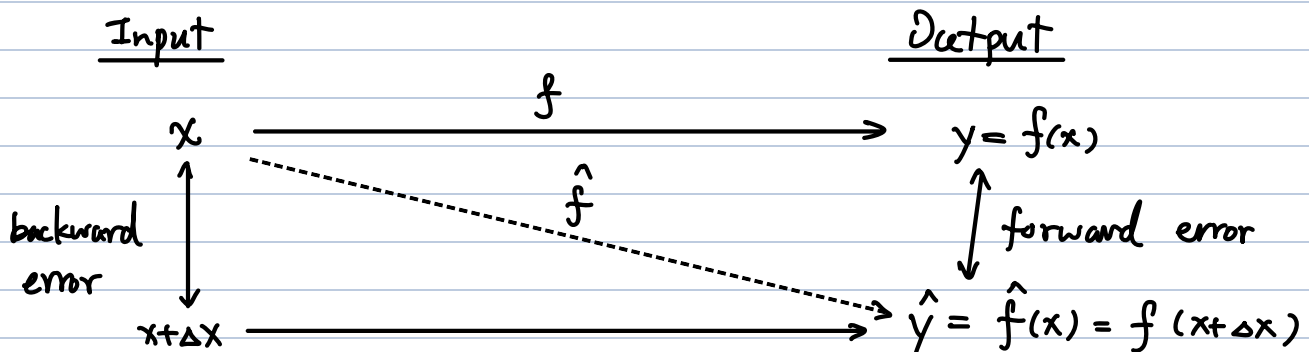
can be improved by rounding probabilistically

$$\text{error} \approx \sqrt{n \log n} \epsilon_{\text{mach}} \quad (\text{Higham-Mary, 2019})$$



$y = f(x)$ ,  $x \in \mathbb{R}^n$ ,  $y \in \mathbb{R}^m$  approximated by  $\hat{f}(x)$

- How should we measure the "quality" of  $\hat{y}$ ?



$$\text{relative forward error} = \frac{\|\hat{f}(x) - f(x)\|}{\|f(x)\|}$$

$$\text{relative backward error} = \frac{\|\Delta x\|}{\|x\|} \quad \text{s.t. } f(x + \Delta x) = \hat{f}(x)$$

Why "backward" analysis?

- Uncertainty in data  $\Leftrightarrow$  rounding error in algorithm
- Reduce bounding forward error to perturbation theory which is well understood.

Def: An algorithm is

- backward stable if  $\frac{\|\Delta x\|}{\|x\|}$  is small
- numerically stable if  $\hat{f}(x) + \Delta y = f(x + \Delta x)$

$\|\Delta y\|/\|y\|$  and  $\|\Delta x\|/\|x\|$  both small

ex. Inner product is backward stable

ex. Outer product is not backward stable (Hint: check the rank)

but satisfies  $f(x y^T) = x y^T + \Delta$ ,  $\|\Delta\| \leq u \|x y^T\|$

hence numerically stable

• Under additional conditions,

backward stability implies numerical stability

$$\text{relation: } \hat{f}(x) - f(x) = f(x + \Delta x) - f(x) = \frac{f(x + \Delta x) - f(x)}{\|\Delta x\|} \|\Delta x\|$$

$$\underbrace{\frac{\|\hat{f}(x) - f(x)\|}{\|f(x)\|}}_{\text{forward error}} \leq \underbrace{\sup_{\Delta x} \left( \frac{\|f(x + \Delta x) - f(x)\| / \|f(x)\|}{\|\Delta x\| / \|x\|} \right)}_{\substack{\text{Condition} \\ \text{number}}} \underbrace{\frac{\|\Delta x\|}{\|x\|}}_{\text{backward error}} + O(\|\Delta x\|^2)$$

$$\begin{aligned} & \text{Condition number} \\ & \text{differentiable} \quad \frac{\|x\| \|Jf(x)\|}{\|f(x)\|} \end{aligned}$$

ex.  $f(b) = A^{-1}b$

$$K = \frac{\|b\| \|A^{-1}\|}{\|A^{-1}b\|} \leq \|A\| \|A^{-1}\|$$

ex. Inner product is numerically stable  $\leftarrow$  exercise.