Fundamentals of numerical analysis

Today: floating point arithmetic

& backward error analysis

Last time: Overview of the course: NLA

To do linear algebra on computers

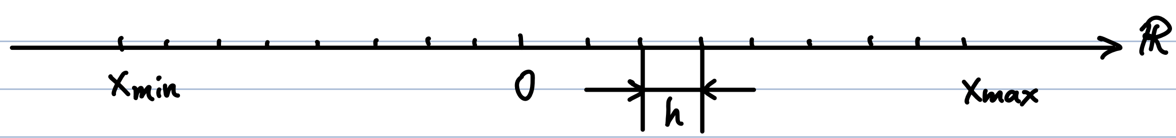
First step: store numbers on computers & do arthmetic

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

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

On a (binary) computer, $\underset{\uparrow\ \text{sign}}{\pm}\ \underbrace{1\,0\,0\,1}_{\substack{\text{integer part}\\ m\text{-digits}}}\,.\,\underbrace{0\,1\,1\,0}_{\substack{\text{fraction part}\\ n\text{-digits}}}$

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

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

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

For x in the range,

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

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

$$fi(0.25) = 0.25$$

$$fi(0.5) = 0.5$$

but $fi(0.5 \times 0.25) = fi(\frac{1}{8}) = 0$

No significant digits

Idea 2 (Floating point #s)

Mimics scientific notation 1.25×10^{-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 $e_{min} \le e \le e_{max}$ (exponent range)
- m: fraction $B^{t-1} \le m \le B^t - 1$

"normalized" ensure unique representation

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

A more common way of expressing floating point # is

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

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

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

  Less easy to overflow / underflow

  Range of nonzero floating point #s

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

example : IEEE 754 (1985, updated 2008)

	B	t	e_{min}	e_{max}	ε_{mach}
(FP32) single prec.	2	24	-126	127	$2^{-24} \approx 5.96 \times 10^{-8}$
(FP64) double prec.	2	53	-1022	1023	$2^{-53} \approx 1.11 \times 10^{-16}$

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

Single precision : 32 bits = 1 + 8 + 23

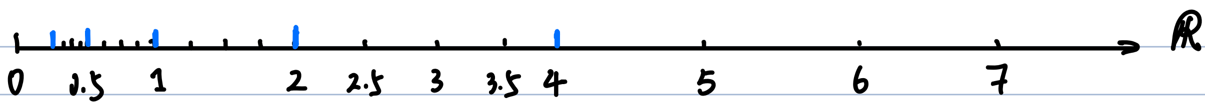
double precision : 64 bits = 1 + 11 + 52

sign exponent fraction

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

↑ multple format ...

- floating point numbers are not equally spaced



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

Floating point numbers:

$$2^3 \times .111 = 2^3 \times \left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3}\right) = 7$$

$$2^3 \times .110 = 2^3 \times \left(\frac{1}{2} + \frac{1}{2^2} + \frac{0}{2^3}\right) = 6$$

$$2^3 \times .101 = 2^3 \times \left(\frac{1}{2} + \frac{0}{2^2} + \frac{1}{2^3}\right) = 5$$

$$2^3 \times .100 = 2^3 \times \left(\frac{1}{2} + \frac{0}{2^2} + \frac{0}{2^3}\right) = 4$$

$$2^3 \times .011 = 2^2 \times .111 = 2^2 \times \left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3}\right) = 3.5$$

$$2^2 \times .110 = 3.0$$

$$2^2 \times .101 = 2.5$$

$$2^2 \times .100 = 2.0$$

$$2^2 \times .011 = 2^1 \times .111 = 1.75$$

$$\vdots$$

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

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

not a normalized representation

- So how to quantify accuracy of floating point # ?

machine epsilon (unit roundoff):

ϵ_{mach} = half distant from 1.0 to the next larger float

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

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

ε_{mach} = relative error of rounding $x \in \mathbb{R}$ to it nearest fp#

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

<u>Thm</u> For every $x \in \mathbb{R}$ (in exponent range)

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

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

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

$x \in [y_1, y_2]$

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

thus

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

∎

- 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 $\circledast$ be its floating point analogue

Standard model
(Fundamental Axiom of Floating Point Arithmetic)

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

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

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

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 $x \equiv fl(x)$ near $\frac{1}{3}$).

Other important points:
(See iJulia notebook)

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

- Rounding error analysis

Consider the inner product

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

Naive summation algorithm

$$s_1 = fl(x_1 y_1)$$
$$s_i = fl(s_{i-1} + fl(x_i y_i)) \qquad i = 2, \cdots, n$$

$$s_1 = x_1 y_1 (1+\delta_1) \quad , \qquad |\delta_1| \le \varepsilon_{mach}$$

$$s_2 = \left(s_1 + x_2 y_2 (1+\delta_2)\right)(1+\delta_2') \qquad |\delta_2|, |\delta_2'| \le \varepsilon_{mach}$$

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

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

$$|\delta_i'|, |\delta_i| \le \varepsilon_{mach}$$

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

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

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

Pf: By induction ▨

By this Lemma, we obtain

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

$$\text{with} \quad |\theta_n'| \le \gamma_n, \ |\theta_j| \le \gamma_j$$

Let $\Delta x = (\theta_n' x_1, \theta_2 x_2, \cdots, \theta_n x_n)^T$

or $\Delta y = (\theta_n' y_1, \theta_2 y_2, \cdots, \theta_n y_n)^T$

then $S_n = (x+\Delta x)^T y = x^T (y + \Delta y)$

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

$$| x^T y - S_n | = |\Delta x^T y| \le \gamma_n \underline{|x|^T |y|}$$

Remark 1: Better algorithm can do better

For example,

1) $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{matrix} n/2 \\ n/2 \end{matrix}$ $\quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} n/2 \\ n/2 \end{matrix}$

do $S^{(1)}_{n/2}$ for x_1, y_1. $S^{(2)}_{n/2}$ for x_2, y_2, then $S^{(1)}_{n/2} + S^{(2)}_{n/2}$

Error: $| fl(S^{(1)}_{n/2} + S^{(2)}_{n/2}) - x^T y | \le \gamma_{n/2+1} |x|^T |y|$

(Hint: perform the induction as above)

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

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

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

3) Use pairwise summation recursively can improve to

$$\text{Error} \le \gamma_{\lceil \log_2 n \rceil} |x|^T |y|$$

Remark 2: Worst case bound is often pessimistic

can be improved by rounding probabilistically

error $\approx \sqrt{n \log n}\ \varepsilon_{mach}$ (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 $\hat{y}$?

<u>Input</u> <u>Output</u>

$$\begin{tikzcd}[column sep=huge, row sep=large]
x \arrow[r, "f"] \arrow[d, leftrightarrow, "\text{backward error}"'] \arrow[dr, dashed, "\hat{f}"] & y = f(x) \arrow[d, leftrightarrow, "\text{forward error}"] \\
x+\Delta x \arrow[r] & \hat{y} = \hat{f}(x) = f(x+\Delta x)
\end{tikzcd}$$

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

relative backward error $= \max\limits_{\substack{\Delta x \\ \text{s.t. } f(x+\Delta x) = \hat{f}(x)}} \dfrac{\|\Delta x\|}{\|x\|}$

Why "backward" analysis ?

- Uncertainty in data $\Longleftrightarrow$ rounding error in algorithm
- Redue bounding forward error to perterbation theory which is well understood.

Def : An algorithm is

backward stable if $\dfrac{\|\Delta x\|}{\|x\|}$ 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 $fl(xy^T) = xy^T + \Delta$, $\|\Delta\| \le c u \|xy^T\|$

hence numerically stable

- Under additional conditions,

backward stability implies numerical stability

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

$$\underbrace{\frac{\|\hat{f}(x) - f(x)\|}{\|f(x)\|}}_{\text{forward error}} \le \underbrace{\sup_{\Delta x}\left(\frac{\|f(x+\Delta x) - f(x)\| / \|f(x)\|}{\|\Delta x\| / \|x\|}\right)}_{=:\ \kappa \ \text{condition number}} \cdot \underbrace{\frac{\|\Delta x\|}{\|x\|}}_{\text{backward error}} + O(\|\Delta x\|^2)$$

$$\kappa \underset{\text{differentiable}}{=} \frac{\|x\| \, \|Jf(x)\|}{\|f(x)\|}$$

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

$$\kappa = \frac{\|b\| \, \|A^{-1}\|}{\|A^{-1} b\|} \le \|A\| \, \|A^{-1}\|$$

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