Welcome to Ma221 !

Floating Point + Error Analysis

Last time: evaluated $p(x) = (x-2)^{13}$ using Horner's rule: $O(10^{-8})$ errors, $\gg |p(x)|$ near $x=2$

Floating Point - How to represent real numbers

Long ago (< 1985) all computer were different $\Rightarrow$ hard to write portable code

Prof Kahan led IEEE Standard Committee.

First Standard: 1985, then 2008, 2019

Scientific Notation: $\pm d.d \cdots d \cdot \text{radix}^e$

Usually radix $= 2$ (or 10, for finance)

Store
- signbit ($\pm$)
- exponent (e)
- mantissa ($d.d \cdots d$)
- $p = \#$ digits in mantissa

Both p and #bit in e are limited to fit in 16, 32, 64, 128 bits.

Historically only hardware support for 32 (single) and 64 (double). Lately 16 (half) popular for machine learning. Google, Nvidia, Intel, .. buiding accelerators for 16 bits.

Some use bfloat16, differs from IEEE 16 bit format in using more e bits, smaller p. Current research on how to use bfloat16 for linear algebra

For simplicity, initially ignore limits on size of e, so assume no overflow or underflow

Normalization: use

$3.100e0$, not $0.0031e3$

i.e. leading digit nonzero

Normalization $\Rightarrow$ unique representations
and $\Rightarrow$ in binary, leading digit $= 1$
$\Rightarrow$ don't need to store it
$\Rightarrow$ free extra bit of precision
"hidden bit"

Def: $\text{rnd}(x)$ = nearest floating point number to x
(Note: Default IEEE rounding rule for breaking ties: round to "nearest even"
$\Rightarrow$ last bit (digit) is even, i.e. 0 in binary

Def: Relative Representation Error (RRE)

$$RRE(x) = \frac{|x - \text{rnd}(x)|}{|\text{rnd}(x)|}$$

Def: Maximum RRE $= \max_{x \neq 0} RRE(x)$

aka machine epsilon, macheps, ε

Max RRE = half distance from 1 to next largest FP number $= 1 + (\text{radix})^{1-p}$
$= .5 \cdot \text{radix}^{1-p} = 2^{-p}$ in binary

Roundoff model (no over/underflow)

(*) $fl(a \text{ op } b) = rnd(a \text{ op } b)$

= true result rounded to nearest even

$= (a \text{ op } b)(1+\delta)$, $|\delta| \le \varepsilon$

op can be $+, -, *, /$

(*) also true for complex arithmetic but with larger ε (Q1.12 for details)

Existing IEEE binary formats

single(S), double(D), quad(Q), half(H)

S: 32 bits = 1 (sign)
+ 8 (exponent)
+ 23 (mantissa)

$p = 23+1 = 24 \Rightarrow \varepsilon = 2^{-24} \sim 6 \cdot 10^{-8}$

$-126 \le e \le 127 \Rightarrow$ OV = overflow threshold $\approx 2^{128} \approx 10^{38}$

UN = underflow threshold $= 2^{-126} \approx 10^{-38}$

D: $64 = 1 + 11 + 52 \Rightarrow p = 53$, $\varepsilon = 2^{-53} \approx 10^{-16}$

$-1022 \le e \le 1023 \Rightarrow$ OV $\sim 2^{1024} \approx 10^{308}$, UN $= 2^{-1022} \approx 10^{-308}$

Q: $128 = 1 + 15 + 112 \Rightarrow \varepsilon \approx 10^{-34}$

OV $\approx 10^{4932} \approx \frac{1}{UN}$

H: $16 = 1 + 5 + 10 \Rightarrow \varepsilon \approx 5 \cdot 10^{-4}$

$OV \approx 10^4$, $UN \approx 10^{-4}$

Bfloat16: $16 = 1 + \underbrace{8}_{\text{same as single}} + 7 \Rightarrow \varepsilon \sim 4 \cdot 10^{-3}$

Lecture 1: "guaranteed correct" except in "rare cases": common approach is to do most work in low precision (fast) and few more steps in high precision (eg. a few steps of Newton) to get "usual precision" when done, as though all work done in single (refs on class web page)

Higher precision than 128 (Quad) available (CARPREC, GMP, links on web page)

"Extended precision" (80 bit format) in original IEEE floating point standard, implemented in Intel x86, now deprecated

Error Analysis:

$$fl(a \text{ op } b) = rnd(a \text{ op } b)$$
$$= (a \text{ op } b)(1+\delta) \quad |\delta| \leq \varepsilon$$

Horner's rule: to evaluate $p(x) = \sum_{i=0}^{d} a_i x^i$

algorithm: $p = a_d$

for $i = d-1 : -1 : 0$

$p = x \cdot p + a_i$

label intermediate terms:

$p_d = a_d$

for $i = d-1 : -1 : 0$

$p_i = x \cdot p_{i+1} + a_i$

introduce roundoff

$p_d = a_d$

for $i = d-1 : -1 : 0$

$$p_i = [x \cdot p_{i+1}(1+\delta_i) + a_i](1+\delta_i')$$
$$|\delta_i| \leq \varepsilon, \quad |\delta_i'| \leq \varepsilon$$

Simplify:

$$p_0 = \sum_{i=0}^{d-1} \left[(1+\delta_i') \prod_{j=0}^{i-1} (1+\delta_j)(1+\delta_j')\right] a_i \cdot x^i$$
$$+ \left[\prod_{j=0}^{d-1} (1+\delta_j)(1+\delta_j')\right] \cdot a_d \cdot x^d$$

$$= \sum_{i=0}^{d-1} [\text{product of } 2i+1 \text{ terms like } 1+\delta] \cdot a_i x^i$$

$$+ [\text{product of } 2d \text{ terms like } 1+\delta]\, a_d x^d$$

$$= \sum_{i=0}^{d} a_i' \cdot x^i \qquad a_i' = a_i \cdot \underbrace{[\text{terms like } 1+\delta]}_{1 + \text{tiny number}}$$

In Words: Horner's Rule is backward stable: returns exact value of a polynomial at x with slightly different coefficients a_i'

Simplify to get error bound

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

$$= 1 + n\varepsilon + O(\varepsilon^2)$$

... usually ignore $O(\varepsilon^2)$

$$\le 1 + \frac{n\varepsilon}{1-n\varepsilon} \quad \text{if } n\varepsilon < 1$$

... proof left to students

$$\prod_{i=1}^{n} (1+\delta_i) \ge (1-\varepsilon)^n = 1 - n\varepsilon + O(\varepsilon^2)$$

$$\ge 1 - \frac{n\varepsilon}{1-n\varepsilon}, \; n\varepsilon < 1$$

$$\Rightarrow \left| \prod_{i=1}^{n} (1+\delta_i) - 1 \right| \le n\varepsilon$$

$$|\text{computed } p_d - p(x)| \le \sum_{i=0}^{d-1} (2i+1)\,\varepsilon\,|a_i x^i| + 2d\,\varepsilon\,|a_d x^d|$$

$$\text{relerr} = \frac{|\text{computed } p_d - p(x)|}{|p(x)|}$$

$$\le \underbrace{\frac{\sum_{i=0}^{d} |a_i x^i|}{|p(x)|}}_{\text{condition number}} \cdot \underbrace{2d\varepsilon}_{\text{backward error}}$$

How many correct digits can we guarantee?

k correct digits $\Leftrightarrow$ relative error bound $\le 10^{-k}$

$\Leftrightarrow -\log_{10}(\text{relative error}) \ge k$

Modify Horner's Rule to get error bound:

$p = a_d$, $\text{ebnd} = |a_d|$

for $i = d-1 : -1 : 0$

$p = x \cdot p + a_i$, $\text{ebnd} = |x| \cdot \text{ebnd} + |a_i|$

$\text{ebnd} = \text{ebnd} \cdot 2 \cdot d \cdot \varepsilon$ ... absolute error bound

(Fig 1.3 in text)

Foreshadows linear algebra

Horizontal axis still problem
to solve, usually n^2 dimensions

Error increases the closer you
get to "hardest possible problem"
for Horner's rule: computing $p(x)=0$
For matrix inversion: set of
singular matrices: n^2-1 dimensional
set in $\mathbb{R}^{n\times n}$: $\det(A)=0$

Later: compute distance from A
to nearest singular matrix (SVD)

For evaluating $p(x)$
rel error bound: $$\underbrace{\frac{\sum_{i=0}^{d} |a_i|\cdot|x^i|}{|p(x)|}}_{\text{condition number}} \cdot \underbrace{2d\varepsilon}_{\text{backward error}}$$

condition# $\to \infty$ as $x \to 2$
because $p(x)=0$

Same idea for linear algebra!
for A^{-1}: condition number
proportional to $\dfrac{1}{\text{distance to nearest singular matrix}}$

for $Ax=b$ what is backward error:
exact answer to $(A+E)\hat{x}=b$
where E "small" compared to A
(need matrix norms)

$\hat{x}-x=(A+E)^{-1}b-A^{-1}b$

Taylor expansion of $(A+E)^{-1}$

Homework: Extend error analysis of Horner to linear algebra!

Horner: $p=a_d$, for $i=d-1:-1:0$, $p=x\cdot p+a_i$

dot product of x,y $s=0$, for $i=1:d$, $s=x_i\cdot y_i+s$

So error analysis similar
(HW 1.10, 1.11)

Details of Floating Point

- Important to understand or write reliable software
- Lots of recent HW developments
- Analyzing code reliability hard
  - recent work on automatic tools to detect error
- see posted notes

① Exception Handling

IEEE Standard has rules for

Underflow: Tiny / Big = 0, or
"subnormal": special numbers with smallest exponent, with zero leading bits in mantissa

$$0.00101 \cdot 2^{\text{minexp}}$$

Overflow or Divide by 0

$\pm 1/0 = \pm \text{Inf}$ = "Infinity"

Natural rules:

Big + Big = Inf

3 - Inf = -Inf etc

Invalid

0/0 = NaN = "Not a number"

Rules: Inf - Inf = NaN, $\sqrt{-1}$ = NaN

3 + NaN = NaN

Flags available to check if an Inf or NaN created

Impact of Exceptions on Software:

Reliability:

Compute: $s = \|x\|_2 = \sqrt{\sum_i |x_i|^2}$

What could go wrong with

$s = 0$, for $i = 1:n$, $s = s + x_i^2$, $s = \sqrt{s}$

Overflow or underflow, could give wrong answer, even if true s "ok", in BLAS

Worst case examples! (see links on web page)

Crash of Ariane 5

Robotic car crash

Current work to make BLAS and LAPACK more reliable (class projects)

Error Analysis:

Before: $rnd(a \text{ op } b) = (a \text{ op } b)\cdot(1+\delta)$
$|\delta| \le \varepsilon$

With underflow

$rnd(a \text{ op } b) = (a \text{ op } b)(1+\delta) + \eta$

where $|\eta| \le$ tiny number related to UN

Speed:

Run "reckless" code, fast but ignores possible expections

Check flags to see if exception occured

In rare case of exceptions, redo slowly / carefully

(2) High Precision: various packages see web page. Some special tricks for fast high precision addition

Optional HW Q1.18

(3) Lower precision (16 bit or less) arithmetic

how low can precision go and still be trust worthy?

lots of variations being considered

New IEEE Standards committee on "Floating Point for ML"

Is 8 bits enough? 4, 2, 1?

Variable precision

(5) Reproducibility:

getting same answer for each run, for debugging + reliability

Not guaranteed on modern architectures:

Parallel computers may perform sums in different orders

$$\underbrace{\underbrace{(1-1)}_{0}+10^{-20}}_{10^{-20}} = 10^{-20} \neq 0 = \underbrace{1+\underbrace{(-1+10^{-20})}_{-1}}_{0}$$

Lots of recent work on reproducible summation! (see class web page)

Latest IEEE standard has new instruction for this

(class projects!)