09/17/2018. # Error and Computer arithmetic

Baojian Zhou

— CSI-401

Numbers must be stored in computers and arithmetic operations must be performed on these numbers?

Two types of numbers:

1. fixed-point format: integer, long integer in C/Java/Python
2. floating-point format: float, double ..

We first consider the <u>decimal floating-point</u>:

For any $x \neq 0$, $x$ can be uniquely represented as:

$$x = \sigma \cdot \bar{x} \cdot 10^{e}, \quad \text{(scientific notation)}$$

where
1. $\sigma \in \{+1, -1\}$ : sign
2. $\bar{x} \in [1, 10)$ : significand (mantissa)
3. $e \in \mathbb{Z}$ : exponent.

Similarly, we can define the <u>binary floating-point</u>:

For any $x \neq 0$, $x = \sigma \cdot \bar{x} \cdot 2^{e}$, (uniquely represented)

where
1. $\sigma \in \{+1, -1\}$ : sign
2. $(1)_2 \leq \bar{x} < (10)_2$ : significand (mantissa)
3. $e \in \mathbb{Z}$ : integer, exponent

Examples of decimal:

$x = 2018.401 \Rightarrow \sigma = +1, \bar{x} = 2.018401, e = 3$

$x = -0.02018 \Rightarrow \sigma = -1, \bar{x} = 2.018, e = -2$

Examples of binary:

$x = (1011.1)_2 \Rightarrow \sigma = +1, \bar{x} = (1.0111)_2, e = 3$

$x = -(0.000101)_2 \Rightarrow \sigma = -1, \bar{x} = (1.01)_2, e = -4$

---

Question: How to store a binary floating-point number into a computer? How to store $\sigma, \bar{x}, e$ ?

IEEE 754 Single-precision floating-point representation:

| $b_1$ | $b_2 \cdots b_9$ | $b_{10} b_{11} \cdots b_{32}$ |
|---|---|---|

(For 32-bit computer, it's a word).

1. $\sigma = \begin{cases} +1 & \text{if } b_1 = 0 \\ -1 & \text{if } b_1 = 1 \end{cases}$ (1 bit)

2. $E = e + 127$, where $E = (b_2 \cdots b_9)_2$ (8 bits)

3. $\bar{x} = 1.b_{10} b_{11} \cdots b_{32}$ (23 bits).

So, $x = \sigma \cdot \bar{x} \cdot 2^e = (-1)^{b_1} \cdot 2^{b_2 b_3 \cdots b_9} \cdot 2^{-127} \cdot (1.b_{10} b_{11} \cdots b_{32})$

$= (-1)^{b_1} \cdot 1.b_{10} b_{11} \cdots b_{32} \cdot 2^{b_2 b_3 \cdots b_9 - 127}$

When $E = 0$, $\sigma = 0$, $b_{10} b_{11} \cdots b_{32} = 00 \cdots 0 \Rightarrow x = 0$

$E = 1$, $e = 1 - 127 = -126$ : minimum exponent

$E = 254$, $e = 254 - 127 = 127$ : maximum exponent

$E = 255$, if $b_{10} b_{11} \cdots b_{32} = 00 \cdots 0 \Rightarrow x = \pm\infty$, otherwise NaN.

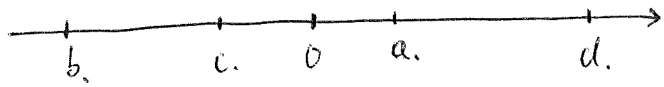
For single-precision format:

a. smallest positive number: $x = 1\cdot(1.00\cdots0)\cdot2^{-126} \approx 1.2\times10^{-38}$

b. smallest number: $x = -1\cdot(1.11\cdots1)\cdot2^{127} \approx -3.4\times10^{38}$

c. largest negative number: $x = -1\cdot(1.00\cdots0)\cdot2^{-126} \approx -1.2\times10^{-38}$

d. largest number: $x = 1\cdot(1.11\cdots1)\cdot2^{127} \approx 3.4\times10^{38}$




---

Question: How accurately a number $x$ can be stored in single/double precision? / How to measure the accuracy of a floating-point format? machine epsilon / largest integer M

machine epsilon:

For any particular floating-point format, the difference between 1. and the next larger number that can be stored in that format.

Example: For single-precision.

$$eps = x_2 - x_1, \text{ where } \begin{cases} x_1 = 1.\underbrace{00\cdots00}_{23\text{ bits}} \\ x_2 = 1.\underbrace{00\cdots0}_{22\text{ bits}}1 \end{cases}$$

$$= 1.+2^{-23} - 1.0$$

$$= 2^{-23} \approx 1.19\times10^{-7}.$$

$x = 1.0 + 2^{-23}$ ✓ $x = 1.0 + 2^{-24}$ × (cannot be exactly stored).

For double-presion.

$eps = 2^{-52} \approx 2.22\times10^{-16}$

$x = 1.0 + 2^{-52}$ ✓ $x = 1.0 + 2^{-53}$ ×.

## largest integer $M$:

$M$ has the following property:

For any $x$ satisfying $0 \le x \le M$ can be stored exactly in the floating-point format.

$x = \sigma \cdot \bar{x} \cdot 2^e$. If $\bar{x}$ has $n$ bits, then $\bar{x}_{max} = 1.\underbrace{11\cdots1}_{n-1}$, $e = n-1$, so $x = 2^n - 1$ can be stored exactly. Also, $x = 2^n$ can also be stored. But $x = 2^n + 1$ does NOT.

For single-precision, $M = 2^{24} = 16777216$

For double-precision, $M = 2^{53} \doteq 9.0 \times 10^{15}$.

---

Question: How to deal with numbers that cannot be stored exactly in a computer?

$$\begin{cases} x = 1. + 2^{-24} \\ x = 1. + 2^{-53} \end{cases}$$

## Rounding and chopping:

Suppose $\bar{x} = 1.b_1 \cdots b_{n-1} b_n b_{n+1} \cdots$, we have two ways:

1. truncate / chop : ignore $b_n b_{n+1} \cdots$
2. round : $\begin{cases} \text{if } b_n = 0, & \text{chop } \bar{x} \text{ to } n \text{ digits.} \\ \text{if } b_n = 1, & \text{add 1 to the last digit, } b_{n-1}. \end{cases}$

(Find out, other 5 rounding methods defined in IEEE 754).

Notation : $x = x_T$ : true number,

$x_A$, $fl(x)$ : machine floating-point number / approximated number.

Let's assume that :

$$fl(x) = x \cdot (1+\varepsilon) \quad , \quad x = \sigma \cdot \bar{x} \cdot 2^e$$

$$\bar{x} = 1.b_1 b_2 \cdots b_n b_{n+1} \cdots$$

For chopping :

1. $\varepsilon \le 0$ For any $x$.

$fl(x) - x = x \cdot \varepsilon$

if ~~$x < 0$~~, then $fl(x) - x = x \cdot \varepsilon$.

$= \sigma \cdot \bar{x} \cdot 2^e \cdot \sigma \cdot (1.b_1 b_2 \cdots b_{n-1} - 1.b_1 b_2 \cdots b_{n-1} b_n b_{n+1} \cdots) \cdot 2^e$

$= \sigma^2 \cdot \bar{x} \cdot 2^{2e} \cdot (-0.\underbrace{00\cdots 0}_{n-1} b_n b_{n+1} \cdots)$.

$\le 0$.

2. Since $\varepsilon \le 0$, the errors could accumulate to be a large number.

3. $-2^{-n+1} \le \varepsilon \le 0$.

$$\varepsilon = \frac{fl(x) - x}{x} = \frac{\sigma \cdot (-0.00\cdots 0\, b_n b_{n+1} \cdots) \cdot 2^e}{\sigma \cdot 1.b_1 b_2 \cdots b_n b_{n+1} \cdots 2^e}$$

$$= -\frac{0.00\cdots 0\, b_n b_{n+1} \cdots}{1.b_1 b_2 \cdots b_n b_{n+1} \cdots} \qquad (-1 \le \varepsilon).$$

So, $-0.00\cdots 0\, b_n b_{n+1} \cdots \le \varepsilon$.

where $-0.00\cdots 0\, b_n b_{n+1} \cdots = -\sum_{i=n}^{\infty} 2^{-i} = -2^{-n+1}$

Therefore, $-2^{-n+1} \le \varepsilon \le 0$.

For rounding:

$$fl(x) = \begin{cases} \sigma \cdot 1.b_1 b_2 \cdots b_{n-1} \cdot 2^e & \text{if } b_n = 0 \\ \sigma \cdot (1.b_1 b_2 \cdots b_{n-1} + 2^{-n+1}) \cdot 2^e & \text{if } b_n = 1 \end{cases}$$

Case 1 : $b_n = 0$

$$|\varepsilon| = \left| \frac{fl(x) - x}{x} \right| = \left| \frac{\sigma \cdot (-0.\underbrace{00\cdots0}_{n-1}\,0\,b_{n+1} b_{n+2} \cdots) \cdot 2^e}{\sigma \cdot 1.b_1 b_2 \cdots b_{n-1} b_n \cdots \cdot 2^e} \right|$$

$$= \frac{0.00\cdots0\,0\,b_{n+1} b_{n+2} \cdots}{1.b_1 b_2 \cdots b_{n-1} b_n \cdots} \leq 0.00\cdots0\,0\,b_{n+1} b_{n+2} \cdots$$

$$= \sum_{i=n+1}^{\infty} 2^{-i} < 2^{-n}, \quad \text{so} \quad |\varepsilon| \leq 2^{-n}.$$

Case 2 : $b_n = 1$.

$$|\varepsilon| = \left| \frac{fl(x) - x}{x} \right| = \left| \frac{\sigma \cdot (2^{-n+1} - 0.00\cdots0\,1\,b_{n+1} b_{n+2} \cdots) \cdot 2^e}{\sigma \cdot 1.b_1 b_2 \cdots b_{n-1} b_n \cdots \cdot 2^e} \right|$$

$$= \frac{2^{-n+1} - 0.00\cdots0\,1\,b_{n+1} b_{n+2} \cdots}{1.b_1 b_2 \cdots b_{n-1}\,1\,b_{n+1} \cdots} = \frac{2^{-n} - 0.00\cdots0\,b_{n+1} b_{n+2} \cdots}{1.b_1 b_2 \cdots b_{n-1}\,1\,b_{n+1} \cdots}$$

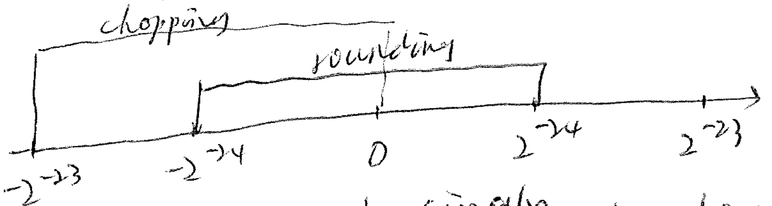
$$\leq \frac{2^{-n} - 0.00\cdots0\,b_{n+1} b_{n+2} \cdots}{1. + 2^{-n}} \leq \frac{2^{-n}}{1. + 2^{-n}} \leq 2^{-n}.$$

For both cases, we have, $|\varepsilon| \leq 2^{-n}$.

$$-2^{-n} \leq \varepsilon \leq 2^{-n}.$$

Summary for rounding and chopping:

1. chopping could be as twice (large as rounding




: single precision.

| | single | double |
|---|---|---|
| rounding | $[-2^{-24}, 2^{-24}]$ | $[-2^{-53}, 2^{-53}]$ |
| chopping | $[-2^{-23}, 0]$ | $[-2^{52}, 0]$ |

$\mathbb{E}\, \varepsilon_r = 0$. $\mathbb{E}\, \varepsilon_c = -2^{-24}$, rounding is better.

---

Question: How to measure errors?

Let $x_T$ : true number, $x_A$ : machine number.

Absolute error: $Err(x_A) = x_T - x_A$

Relative error: $Rel(x_A) = \dfrac{x_T - x_A}{x_T}$

Example: $x_T = \pi$, $x_A = 22/7$.

$Err(x_A) \approx -0.00126$

$Rel(x_A) \approx -0.000403$

---

Question: Which one is better?

Ex 1: Albany — NYC

$x_T = 152$ mi $\quad Err(x_A) = 1$

$x_A = 151$ mi $\quad Rel(x_A) = 1/152 \approx 0.658\%$

Albany — Crossgates

$x_T = 2$ miles

$x_A = 1$ mile

7

There are 8 types of errors. Check: example-02.ipynb

1. modeling errors:

$$N(t) = N_0 e^{kt}, \quad N_0 = 3924000 \times e^{-1790k}, \quad k = 0.02975$$

$$1790 \le t \le 1860.$$

2. blunders and mistakes.

3. physical measurment : speed of light in a vacuum

$$c = (2.997925 + \varepsilon) \cdot 10^{10} \text{ cm/sec}, \ |\varepsilon| \le 0.000003.$$

4. machine representation error : rounding / chopping.

5. mathematical approximation errors.

6. loss-of-significant errors.

$$\cos(2\theta) = 2\cos^2(\theta) - 1 = 1 - 2\sin^2(\theta).$$

$$f(x) = \frac{1-\cos x}{x^2} = \frac{2\sin^2(x/2)}{x^2} = \frac{1}{2}\left[\frac{\sin(x/2)}{x/2}\right]^2.$$

7. noise in function evalution :

8. overflow / underflow.

Errors in arithmetic operations:

Propagated error: $E = (x_T \, \omega \, y_T) - (x_A \, \omega \, y_A)$.

1. $x_T, y_T$ are true numbers.
2. $x_A, y_A$ are machine numbers.
3. $\omega \in \{+, -, \times, \div\}$.
4. Suppose we know the rounding error $\varepsilon_x, \varepsilon_y$.

$$x_T = x_A + \varepsilon_x, \quad y_T = y_A + \varepsilon_y$$

①. For multiplication: $\omega = *$

We try to use $Rel(x_A * y_A)$ to bound the error.

$$Rel(x_A * y_A) = \frac{x_T y_T - x_A y_A}{x_T \cdot y_T} = \frac{x_T y_T - (x_T - \varepsilon_x)\cdot(y_T - \varepsilon_y)}{x_T y_T}$$

$$= \frac{x_T \varepsilon_y + y_T \cdot \varepsilon_x - \varepsilon_x \varepsilon_y}{x_T y_T} = \frac{\varepsilon_y}{y_T} + \frac{\varepsilon_x}{x_T} - \frac{\varepsilon_x}{x_T} \cdot \frac{\varepsilon_y}{y_T}$$

$$= Rel(x_A) + Rel(y_A) - \underbrace{Rel(x_A) \cdot Rel(y_A)}_{\text{small}}.$$

$$\approx Rel(x_A) + Rel(y_A).$$

Numerical stable.

②. For division:

$$Rel\left(\frac{x_A}{y_A}\right) = \frac{x_T/y_T - x_A/y_A}{x_T/y_T} = \frac{\left(x_T/y_T - \frac{x_A}{y_A}\right)\cdot \frac{y_A}{x_T}}{\left(\frac{x_T}{y_T}\right)\cdot \frac{y_A}{x_T}}$$

$$= \frac{\frac{y_A}{y_T} - \frac{x_A}{x_T}}{\frac{y_A}{y_T}} = \frac{1 - \frac{x_A}{x_T} - \left(1 - \frac{y_A}{y_T}\right)}{1 - \left(1 - \frac{y_A}{y_T}\right)} = \frac{Rel(x_A) - Rel(y_A)}{1 - Rel(y_A)}$$

As $\lim_{n\to\infty} \sum_{n=0}^{\infty} [Rel(y_A)]^n = \frac{1}{1-Rel(y_A)}$

So $Rel(\frac{x_A}{y_A}) = [Rel(x_A) - Rel(y_A)] \cdot \sum_{n=0}^{\infty} [Rel(y_A)]^n$

$$\approx Rel(x_A) - Rel(y_A) + \underbrace{Rel(x_A) \cdot Rel(y_A) - Rel(y_A)^2 + \cdots}_{\text{small}}$$

$$\approx Rel(x_A) - Rel(y_A).$$

So, division is also numerical stable.

Addition / substraction :

$$Rel(x_A + y_A) = \frac{x_T + y_T - (x_A + y_A)}{x_T + y_T} = \frac{x_T \cdot \frac{x_T - x_A}{x_T} + y_T \cdot \frac{y_T - y_A}{y_T}}{x_T + y_T}$$

$$= \frac{x_T \cdot Rel(x_A) + y_T \cdot Re(y_A)}{x_T + y_T}$$

- If $x_T, y_T$ have same sign :

$$|Rel(x_A + y_A)| \le |Rel(x_A)| + |Rel(y_A)|$$

- If $x_T, y_T$ have different sign :

Consider $x_T = x+1$, $y_T = -x$ . $x \geq 0$.

$$Rel(x_A + y_A) = \frac{(x+1) \cdot Rel(x_A) - x \cdot Rel(y_A)}{x+1-x}$$

$$= \underbrace{x \cdot [Rel(x_A) - Rel(y_A)]}_{\text{this could be large.}} + Rel(x_A)$$

So, addition is not numerical stable.

Propagated error in function evaluation:

$f(x)$ differentiable $[a, b]$.

$$f(x_T) - f(x_A) = f'(y)\cdot(x_T - x_A).$$

$$y \in [x_T, x_A] \text{ or } [x_A, x_T]$$

Since $x_T$ and $x_A$ are close to each other.

$$f(x_T) - f(x_A) \approx f'(x_T)\cdot(x_T - x_A)$$

$$Rel(f(x_A)) = \frac{f(x_T) - f(x_A)}{f(x_T)} = \frac{f'(y)\cdot(x_T - x_A)}{f(x_T)}$$

$$\approx \frac{f'(x_T)\cdot x_T}{f(x_T)} \cdot \frac{(x_T - x_A)}{x_T}$$

$$= \frac{f'(x_T)\cdot x_T}{f(x_T)} \cdot Rel(x_A)$$

Example: $f(x) = b^x$, $f'(x) = (\log b)\cdot b^x$.

So, $\cancel{Rel(f(x_A))} \approx \frac{(\log b)\cdot b^x}{b^x}$.

$$b^{x_T} - b^{x_A} \approx (\log b)\cdot b^{x_T}\cdot(x_T - x_A)$$

$$Rel(b^{x_A}) \approx (\log b)\cdot x_T \cdot Rel(x_T).$$

$K = (\log b)\, x_T$. condition number.

It could be large!