ECS 231
Computer Arithmetic

Outline

Outline

1. Floating-point numbers and representations
2. Floating-point arithmetic
3. Floating-point error analysis
4. Further reading

Floating-point numbers and representations

1. Floating-point (FP) representation of numbers (scientific notation):

$$\underset{\substack{\uparrow \\ \text{sign}}}{-} \quad \underset{\substack{\uparrow \\ \text{significand}}}{3.1416} \times \underset{\substack{\uparrow \\ \text{base}}}{10}^{1} \quad \leftarrow \text{exponent}$$

2. FP representation of a nonzero **binary** number:

$$x = \pm\, b_0.b_1 b_2 \cdots b_{p-1} \times 2^E. \tag{1}$$

- It is *normalized*, i.e., $b_0 = 1$ (the hidden bit)
- *Precision* $(= p)$ is the number of bits in the significand (mantissa) (including the hidden bit).
- *Machine epsilon* $\epsilon = 2^{-(p-1)}$, the gap between the number 1 and the smallest FP number that is greater than 1.

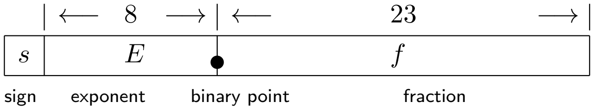
3. Special numbers: 0, -0, ∞, $-\infty$, NaN(="Not a Number").

IEEE standard

- All computers designed since 1985 use the *IEEE Standard for Binary Floating-Point Arithmetic* (ANSI/IEEE Std 754-1985), represent each number as a binary number and use binary arithmetic.
- Essentials of the IEEE standard:
  - consistent representation of FP numbers
  - correctly rounded FP operations (using various rounding modes)
  - consistent treatment of exceptional situation such as division by zero.

IEEE single precision format

- **Single** format takes 32 bits (=4 bytes) long:



- It represents the number

$$(-1)^s \cdot (1.f) \times 2^{E-127}$$

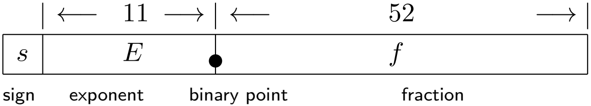
- The leading 1 in the fraction need not be stored explicitly since it is always 1 (*hidden bit*)
- $E_{\min} = (00000001)_2 = (1)_{10}$, $E_{\max} = (11111110)_2 = (254)_{10}$.
- "$E - 127$" in exponent avoids the need for storage of a sign bit.
- The range of positive normalized numbers:

$$N_{\min} = 1.00 \cdots 0 \times 2^{E_{\min}-127} = 2^{-126} \approx 1.2 \times 10^{-38}$$
$$N_{\max} = 1.11 \cdots 1 \times 2^{E_{\max}-127} \approx 2^{128} \approx 3.4 \times 10^{38}.$$

- Special repsentations for 0, $\pm\infty$ and NaN.

IEEE double pecision format

- **Double** format takes 64 bits (= 8 bytes) long:



- It represents the numer

$$(-1)^s \cdot (1.f) \times 2^{E-1023}$$

- The range of positive normalized numbers is from

$$N_{\min} = 1.00\cdots0 \times 2^{1022} \approx 2.2 \times 10^{-308}$$
$$N_{\max} = 1.11\cdots1 \times 2^{1023} \approx 2^{1024} \approx 1.8 \times 10^{308}.$$

- Special repsentations for 0, $\pm\infty$ and NaN.

Summary I

- Precision and machine epsilon of IEEE single, double and extended formats

Format	Precision p	Machine epsilon $\epsilon = 2^{-p-1}$
single	24	$\epsilon = 2^{-23} \approx 1.2 \times 10^{-7}$
double	53	$\epsilon = 2^{-52} \approx 2.2 \times 10^{-16}$
extended	64	$\epsilon = 2^{-63} \approx 1.1 \times 10^{-19}$

- Extra: Higham's lecture for additional formats, such as half (16 bits) and quadruple (128 bits).

Rounding modes

- Let a positive real number x be in the normalized range, i.e., $N_{\min} \leq x \leq N_{\max}$, and write in the normalized form

$$x = (1.b_1 b_2 \cdots b_{p-1} b_p b_{p+1} \ldots) \times 2^E,$$

- Then the closest fp number less than or equal to x is

$$x_- = 1.b_1 b_2 \cdots b_{p-1} \times 2^E$$

i.e., x_- is obtained by *truncating*.

- The next fp number bigger than x_- (also the next one that bigger than x) is

$$x_+ = ((1.b_1 b_2 \cdots b_{p-1}) + (0.00 \cdots 01)) \times 2^E$$

- If x is negative, the situtation is reversed.

Correctly rounding modes:

- *round down*:

$$\text{round}(x) = x_-$$

- *round up*:

$$\text{round}(x) = x_+$$

- *round towards zero*:

$$\text{round}(x) = x_- \text{ if } x \geq 0$$
$$\text{round}(x) = x_+ \text{ if } x \leq 0$$

- *round to nearest*:

$$\text{round}(x) = x_- \quad \text{or} \quad x_+$$

whichever is nearer to x.[1]

[1] except that if $x > N_{\max}$, $\text{round}(x) = \infty$, and if $x < -N_{\max}$, $\text{round}(x) = -\infty$. In the case of tie, i.e., x_- and x_+ are the same distance from x, the one with its least significant bit equal to zero is chosen.

Rounding error

- When the *round to nearest* (IEEE default rounding mode) is in effect,

$$\text{relerr}(x) = \frac{|\text{round}(x) - x|}{|x|} \le \frac{1}{2}\epsilon.$$

- Therefore, we have

$$\text{relerr} = \begin{cases} \frac{1}{2} \cdot 2^{1-24} = 2^{-24} \approx 5.96 \cdot 10^{-8}, & \text{single} \\ \frac{1}{2} \cdot 2^{-52} \approx 1.11 \times 10^{-16}, & \text{double.} \end{cases}$$

Outline

Floating-point arithmetic

- IEEE rules for correctly rounded fp operations:
  if x and y are correctly rounded fp numbers, then

$$
\begin{aligned}
\mathrm{fl}(x+y) &= \mathsf{round}(x+y) = (x+y)(1+\delta) \\
\mathrm{fl}(x-y) &= \mathsf{round}(x-y) = (x-y)(1+\delta) \\
\mathrm{fl}(x\times y) &= \mathsf{round}(x\times y) = (x\times y)(1+\delta) \\
\mathrm{fl}(x/y) &= \mathsf{round}(x/y) = (x/y)(1+\delta)
\end{aligned}
$$

  where

$$|\delta| \leq \frac{1}{2}\epsilon$$

- IEEE standard also requires that correctly rounded remainder and square root operations be provided.

Floating-point arithmetic, cont'd

IEEE standard response to exceptions

Event	Example	Set result to
Invalid operation	$0/0$, $0 \times \infty$	NaN
Division by zero	Finite nonzero/0	$\pm\infty$
Overflow	$\lvert x\rvert > N_{\max}$	$\pm\infty$ or $\pm N_{\max}$
underflow	$x \neq 0, \lvert x\rvert < N_{\min}$	± 0, $\pm N_{\min}$ or subnormal
Inexact	whenever $\mathrm{fl}(x \circ y) \neq x \circ y$	correctly rounded value

Floating-point arithmetic error

- Let $\widehat{x}$ and $\widehat{y}$ be the fp numbers and that

$$\widehat{x} = x(1+\tau_1) \quad \text{and} \quad \widehat{y} = y(1+\tau_2), \quad \text{for } |\tau_i| \leq \tau \ll 1$$

  where τ_i could be the relative errors in the process of "collecting/getting" the data from the original source or the previous operations, and
- **Question: how do the four basic arithmetic operations behave?**

Floating-point arithmetic error: $+, -$

Addition and subtraction:

$$
\begin{aligned}
\mathrm{fl}(\widehat{x}+\widehat{y}) &= (\widehat{x}+\widehat{y})(1+\delta) \\
&= x(1+\tau_1)(1+\delta) + y(1+\tau_2)(1+\delta) \\
&= x + y + x(\tau_1+\delta+O(\tau\epsilon)) + y(\tau_2+\delta+O(\tau\epsilon)) \\
&= (x+y)\left(1+\frac{x}{x+y}(\tau_1+\delta+O(\tau\epsilon))+\frac{y}{x+y}(\tau_2+\delta+O(\tau\epsilon))\right) \\
&\equiv (x+y)(1+\widehat{\delta}),
\end{aligned}
$$

where

$$
|\delta| \le \frac{1}{2}\epsilon, \qquad |\widehat{\delta}| \le \frac{|x|+|y|}{|x+y|}\left(\tau+\frac{1}{2}\epsilon+O(\tau\epsilon)\right).
$$

Floating-point arithmetic error: $+, -$

Three possible cases:

1. If x and y have the same sign, i.e., $xy > 0$, then $|x + y| = |x| + |y|$; this implies
$$|\widehat{\delta}| \leq \tau + \frac{1}{2}\epsilon + O(\tau\epsilon) \ll 1.$$
Thus $\mathrm{fl}(\widehat{x} + \widehat{y})$ approximates $x + y$ well.
2. If $x \approx -y \Rightarrow |x + y| \approx 0$, then $(|x| + |y|)/|x + y| \gg 1$; this implies that $|\widehat{\delta}|$ could be nearly or much bigger than 1. This is so called **catastrophic cancellation**, it causes relative errors or uncertainties already presented in $\widehat{x}$ and $\widehat{y}$ to be magnified.
3. In general, if $(|x| + |y|)/|x + y|$ is not too big, $\mathrm{fl}(\widehat{x} + \widehat{y})$ provides a good approximation to $x + y$.

Catastrophic cancellation: example 1

- Computing $\sqrt{x+1} - \sqrt{x}$ straightforward causes substantial loss of significant digits for large n

x	$\mathrm{fl}(\sqrt{x+1})$	$\mathrm{fl}(\sqrt{x})$	$\mathrm{fl}(\mathrm{fl}(\sqrt{x+1}) - \mathrm{fl}(\sqrt{x})$
1.00e+10	1.00000000004999994e+05	1.00000000000000000e+05	4.99999441672116518e-06
1.00e+11	3.16227766018419061e+05	3.16227766016837908e+05	1.58115290105342865e-06
1.00e+12	1.00000000000050000e+06	1.00000000000000000e+06	5.00003807246685028e-07
1.00e+13	3.16227766016853740e+06	3.16227766016837955e+06	1.57859176397323608e-07
1.00e+14	1.00000000000000503e+07	1.00000000000000000e+07	5.02914190292358398e-08
1.00e+15	3.16227766016838104e+07	3.16227766016837917e+07	1.86264514923095703e-08
1.00e+16	1.00000000000000000e+08	1.00000000000000000e+08	0.00000000000000000e+00

- *Catastrophic cancellation can sometimes be avoided if a formula is properly reformulated.*
- In the present case, one can compute $\sqrt{x+1} - \sqrt{x}$ almost to full precision by using the equality

$$\sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}}.$$

Catastrophic cancellation: example 2

- Consider the function

$$f(x) = \frac{1-\cos x}{x^2}$$

  Note that

$$0 \le f(x) < 1/2 \quad \text{for all } x \neq 0$$

- Let $x = 1.2 \times 10^{-8}$, then the computed

$$\mathrm{fl}(f(x)) = 0.770988...$$

  is completely wrong!
- Alternatively, the function can be re-written as

$$f(x) = \left(\frac{\sin(x/2)}{x/2}\right)^2.$$

  Consequently, for $x = 1.2 \times 10^{-8}$, then the computed $\mathrm{fl}(f(x)) = 0.499999... < 1/2$ is fine!

Floating-point arithmetic error: $\times, /$

Multiplication and Division:

$$
\begin{aligned}
\mathrm{fl}(\widehat{x} \times \widehat{y}) &= (\widehat{x} \times \widehat{y})(1+\delta) \\
&= xy(1+\tau_1)(1+\tau_2)(1+\delta) \\
&\equiv xy(1+\widehat{\delta}_\times), \\
\mathrm{fl}(\widehat{x}/\widehat{y}) &= (\widehat{x}/\widehat{y})(1+\delta) \\
&= (x/y)(1+\tau_1)(1+\tau_2)^{-1}(1+\delta) \\
&\equiv xy(1+\widehat{\delta}_\div),
\end{aligned}
$$

where

$$
\widehat{\delta}_\times = \tau_1 + \tau_2 + \delta + O(\tau\epsilon), \qquad \widehat{\delta}_\div = \tau_1 - \tau_2 + \delta + O(\tau\epsilon).
$$

Thus

$$
|\widehat{\delta}_\times| \le 2\tau + \frac{1}{2}\epsilon + O(\tau\epsilon), \qquad |\widehat{\delta}_\div| \le 2\tau + \frac{1}{2}\epsilon + O(\tau\epsilon)
$$

we can conclude that *multiplication and division are very well-behaved!*

Outline

Floating-point error analysis

- Illustrate the basic idea of error analysis through a simple example. Consider the inner product:

$$x^T y = x_1 y_1 + x_2 y_2 + x_3 y_3,$$

  assuming already x_i's and y_j's are fp numbers.

- $\mathrm{fl}(x^T y)$ is computed in the following order:

$$\mathrm{fl}(x^T y) = \mathrm{fl}\big(\,\mathrm{fl}(\mathrm{fl}(x_1 y_1) + \mathrm{fl}(x_2 y_2)) + \mathrm{fl}(x_3 y_3)\,\big).$$

- By the fp arithmetic model, we have

$$\begin{aligned}
\mathrm{fl}(x^T y) &= \mathrm{fl}\big(\,\mathrm{fl}(x_1 y_1(1+\epsilon_1) + x_2 y_2(1+\epsilon_2)) + x_3 y_3(1+\epsilon_3)\,\big) \\
&= \mathrm{fl}\big(\,(x_1 y_1(1+\epsilon_1) + x_2 y_2(1+\epsilon_2))(1+\delta_1) + x_3 y_3(1+\epsilon_3)\,\big) \\
&= \big(\,(x_1 y_1(1+\epsilon_1) + x_2 y_2(1+\epsilon_2))(1+\delta_1) + x_3 y_3(1+\epsilon_3)\,\big)(1+\delta_2) \\
&= x_1 y_1(1+\epsilon_1)(1+\delta_1)(1+\delta_2) + x_2 y_2(1+\epsilon_2)(1+\delta_1)(1+\delta_2) \\
&\quad + x_3 y_3(1+\epsilon_3)(1+\delta_2),
\end{aligned}$$

where $|\epsilon_i| \le \frac{1}{2}\epsilon$ and $|\delta_j| \le \frac{1}{2}\epsilon$.

Floating-point error analysis, cont'd

There are two ways to interpret the errors in the computed $\mathrm{fl}(x^T y)$:

- Forward error analysis
- Backward error analysis

Forward error analysis

- We have

$$\mathrm{fl}(x^T y) = x^T y + E,$$

where

$$\begin{aligned} E =& x_1 y_1(\epsilon_1 + \delta_1 + \delta_2) + x_2 y_2(\epsilon_2 + \delta_1 + \delta_2) \\ &+ x_3 y_3(\epsilon_3 + \delta_2) + O(\epsilon^2). \end{aligned}$$

- It implies that

$$\begin{aligned} |E| &\leq \frac{1}{2}\epsilon(3|x_1 y_1| + 3|x_2 y_2| + 2|x_3 y_3|) + O(\epsilon^2) \\ &\leq \frac{3}{2}\epsilon \cdot |x|^T |y| + O(\epsilon^2). \end{aligned}$$

- This bound on E tells the worst-case difference between the "exact" $x^T y$ and its computed value.

Backward error analysis

- We can also write

$$\mathrm{fl}(x^T y) = \widehat{x}^T \widehat{y} = (x + \Delta x)^T (y + \Delta y),$$

where

$$\begin{aligned} \widehat{x}_1 &= x_1(1+\epsilon_1), & \widehat{y}_1 &= y_1(1+\delta_1)(1+\delta_2) \equiv y_1(1+\widehat{\delta}_1), \\ \widehat{x}_2 &= x_2(1+\epsilon_2), & \widehat{y}_2 &= y_2(1+\delta_1)(1+\delta_2) \equiv y_2(1+\widehat{\delta}_2), \\ \widehat{x}_3 &= x_3(1+\epsilon_3), & \widehat{y}_3 &= y_3(1+\delta_2) \equiv y_3(1+\widehat{\delta}_3). \end{aligned}$$

and

$$|\widehat{\delta}_1| = |\widehat{\delta}_2| \leq \epsilon + O(\epsilon^2) \quad \text{and} \quad |\widehat{\delta}_3| \leq \frac{1}{2}\epsilon.$$

- This says the computed value $\mathrm{fl}(x^T y)$ is the *"exact"* inner product of a slightly perturbed $\widehat{x}$ and $\widehat{y}$.

Outline

Further reading

1. D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 18(1):5–48, 1991.
2. Rencet lecture by N. Higham on the latest development on low precision and multiprecision arithmetic.
   `http://bit.ly/kacov18`
3. Discussion of numerical disasters:
   - T. Huckle, Collection of software bugs
     `http://www5.in.tum.de/~huckle/bugse.html`
   - "Bits and Bugs: A Scientific and Historical Review of Software Failures in Computational Science" by T. Huckle and T. Neckel, SIAM, March 2019.