

# Computer Arithmetic

## CS/SE 4X03

Ned Nediankov

McMaster University

September 16, 2021

# What is the output

```
a(1) = (1/cos(100*pi+pi/4))^2; % (1/cos(100π + π/4))^2 = 2
a(2) = 3*tan(atan(1e7))/1e7; % 3tan(arctan(10^7))/10^7 = 3
x = 4;
for i=1:20 x = sqrt(x); end
for i=1:20 x = x*x; end
a(3) = x; % = 4
a(4) = 5*(1+exp(-100)-1)/(1+exp(-100)-1); % 5 * (1+e^-100-1)/(1+e^-100-1) = 5
a(5) = log(exp(6e3))/1e3; % ln(e^6000)/1000 = 6
for i = 1:5
    fprintf('%d: %.16f\n', i+1, a(i));
end
```

# Outline

Fixed-point vs. floating-point

Floating-point number system

Rounding

Machine epsilon

IEEE 754

Cancellations

## Useful links

- [IEEE 754 double precision visualization](#)
- [C. Moler. Floating Point Numbers](#)
- [IEEE 754](#)
- [N. Higham. Half Precision Arithmetic: fp16 Versus bfloat16](#)
- [GNU Multiple Precision Arithmetic Library](#)
- [The GNU Multiple Precision \(GMP\) Library, in Maple](#)
- [Quadruple-precision floating-point format](#)

## Fixed-point vs. floating-point

- Fixed-point: the position of the radix (e.g. decimal, binary) point is fixed
  - Consider unsigned, 6 decimal digit representation with 4 digits for the integer part and 2 digits for the fractional part; e.g. 1234.56, 12.34
  - Largest number 9999.99, smallest 0.01
  - Can be interpreted as integers with implicit scaling factor, here 1/100
- Floating-point: the radix point can “float” between the digits
  - E.g. with 6 digits we can represent 0.123456, 0.00123456, 12345600000
  - Position of radix point is determined from an exponent  
E.g.  $0.123456 = 1.23456 \times 10^{-1} = 1234.56 \times 10^{-4}$
  - To ensure uniqueness of representation, assume first digit nonzero, followed by “.” followed by 5 digits

## Floating-point number system

A floating-point (FP) system is characterized by four integers  $(\beta, t, L, U)$ , where

- $\beta$  base of the system or radix
- $t$  number of digits or **precision**
- $[L, U]$  exponent range

A number  $x$  in the system is represented as

$$x = \pm \left( d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_{t-1}}{\beta^{t-1}} \right) \times \beta^e$$

where

- $0 \leq d_i \leq \beta - 1, i = 0, \dots, t - 1$
- $e \in [L, U]$

## Floating-point number system cont.

- The string of base  $\beta$  digits  $d_0d_1 \cdots d_{t-1}$  is called **mantissa** or **significand**
- $d_1d_2 \cdots d_{t-1}$  is called **fraction**
- A common way of expressing  $x$  is

$$\pm d_0.d_1 \cdots d_{t-1} \times \beta^e$$

- A FP number is **normalized** if  $d_0$  is nonzero  
**denormalized** otherwise

## Floating-point number system cont.

**Example 1.** Consider the FP  $(10, 3, -2, 2)$ .

- Numbers are of the form

$$d_0.d_1d_2 \times 10^e, \quad d_0 \neq 0, e \in [-2, 2]$$

- largest positive number  $9.99 \times 10^2$
- smallest positive normalized number  $1.00 \times 10^{-2}$
- smallest positive denormalized number  $0.01 \times 10^{-2}$
- denormalized numbers are e.g.  $0.23 \times 10^{-2}$ ,  $0.11 \times 10^{-2}$
- 0 is represented as  $0.00 \times 10^0$



# Rounding

How to store a real number

$$x = \pm d_0.d_1 \cdots d_{t-1}d_t d_{t+1} \cdots \times \beta^e$$

in  $t$  digits?

Denote by  $\text{fl}(x)$  the FP representation of  $x$

- Rounding by chopping (also called rounding towards zero)
- Rounding to nearest.  $\text{fl}(x)$  is the nearest FP to  $x$   
If a tie, round to the even FP
- Rounding towards  $+\infty$ .  $\text{fl}(x)$  is the smallest FP  $\geq x$
- Rounding towards  $-\infty$ .  $\text{fl}(x)$  is the largest FP  $\leq x$

## Rounding cont.

**Example 2.** Consider the FP  $(10, 3, -2, 2)$ .

Let  $x = 1.2789 \times 10^1$

- chopping:  $\text{fl}(x) = 1.27 \times 10^1$
- nearest:  $\text{fl}(x) = 1.28 \times 10^1$
- $+\infty$ :  $\text{fl}(x) = 1.28 \times 10^1$
- $-\infty$ :  $\text{fl}(x) = 1.27 \times 10^1$

Let  $x = 1.275000 \times 10^1$ .

- nearest:  $\text{fl}(x) = 1.28 \times 10^1$

## Machine epsilon

**Machine epsilon:** the distance from 1 to the next larger FP number

E.g. in  $(10, 3, -2, 2)$ ,  $\epsilon_{\text{mach}} = 1.01 - 1 = 0.01 = 10^{-2}$

Another definition: smallest  $\epsilon > 0$  such that  $\text{fl}(1 + \epsilon) > 1$

These two definitions are not equivalent

**Unit roundoff:**  $u = \epsilon_{\text{mach}}/2$

When rounding to the nearest

$$\text{fl}(x) = x(1 + \epsilon), \quad \text{where } |\epsilon| \leq u$$

i.e.

$$\frac{\text{fl}(x) - x}{x} = \epsilon, \quad \left| \frac{\text{fl}(x) - x}{x} \right| \leq u$$

## IEEE 754

- IEEE 754 standard for FP arithmetic (1985)
- IEEE 754-2008, IEEE 754-2019
- Most common (binary) single and double precision since 2008 half precision

	bits	$t$	$L$	$U$	$\epsilon_{\text{mach}}$
single	32	24	-126	127	$1.2 \times 10^{-7}$
double	64	53	-1022	1023	$2.2 \times 10^{-16}$

	range	smallest	
		normalized	denormalized
single	$\pm 3.4 \times 10^{38}$	$\pm 1.2 \times 10^{-38}$	$\pm 1.4 \times 10^{-45}$
double	$\pm 1.8 \times 10^{308}$	$\pm 2.2 \times 10^{-308}$	$\pm 4.9 \times 10^{-324}$

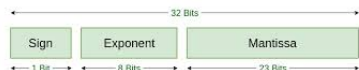
(These are  $\approx$  values)

## IEEE 754 cont.

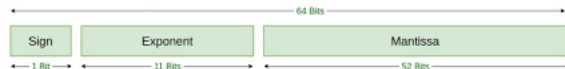
### Exceptional values

- **Inf**, **-Inf** when the result overflows, e.g.  $1/0.0$
- **NaN** "Not a Number" results from undefined operations e.g.  $0/0$ ,  $0*\text{Inf}$ ,  $\text{Inf}/\text{Inf}$   
**NaNs** propagate through computations

## IEEE 754 cont.



Single Precision  
IEEE 754 Floating-Point Standard



Double Precision  
IEEE 754 Floating-Point Standard

(From <https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/>):

[//www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/](https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/))

- sign 0 positive, 1 negative
- exponent is biased
- first bit of mantissa is not stored, sticky bit, assumed 1

## IEEE 754 cont.

## Single precision

- **Inf**
  - sign: 0 for +**Inf**, 1 for -**Inf**
  - biased exponent: all 1's, 255
  - fraction: all 0's
- **NaN**
  - sign: 0 or 1
  - biased exponent: all 1's, 255
  - fraction: at least one 1
- 0
  - sign: 0 for +0, 1 for -0
  - biased exponent: all 0's
  - mantissa: all 0's
- FP numbers
  - biased exponent: from 1 to 254, bias: 127
  - actual exponent:  $1 - 127 = -126$  to  $254 - 127 = 127$

## IEEE 754 cont.

## Double precision

- bias 1023
- biased exponent: from 1 to 2046
- actual exponent: from  $-1022$  to 1023
- rest similar to single



## IEEE 754 cont.

## FP arithmetic

For a real  $x$  and rounding to nearest

$$\text{fl}(x) = x(1 + \epsilon), \quad |\epsilon| \leq u$$

$u$  is the unit roundoff of the precision

The arithmetic operations are correctly rounded, i.e. for  $x$  and  $y$  IEEE numbers (e.g. in double or single precision) and rounding to the nearest

$$\text{fl}(x \circ y) = (x \circ y)(1 + \epsilon), \quad \circ \in \{+, -, *, /\}, \quad |\epsilon| \leq u$$

## IEEE 754 cont.

**Example 3.** Consider a decimal floating-point system with  $t = 5$  and rounding to nearest

- The machine epsilon is  $1.0001 - 1.0000 = 0.0001 = 10^{-4}$
- Unit roundoff is  $u = 10^{-4}/2 = 5 \times 10^{-5}$
- Let  $x = 1.162611735194631$

With rounding to nearest,  $\text{fl}(x) = 1.1626$

$$\text{fl}(x) = x(1 + \epsilon)$$

$$\epsilon = \frac{\text{fl}(x) - x}{x} = \frac{1.1626 - 1.162611735194631}{1.162611735194631} \approx -1.0094 \times 10^{-5}$$

$$|\epsilon| \approx 1.0094 \times 10^{-5} < 5 \times 10^{-5}$$

## IEEE 754 cont.

## Example 3. cont.

Let  $x = 1.162611735194631 \times 10^8$ . Then  $\text{fl}(x) = 1.1626 \times 10^8$

$$\epsilon = \frac{\text{fl}(x) - x}{x} = \frac{1.1626 \times 10^8 - 1.162611735194631 \times 10^8}{1.162611735194631 \times 10^8}$$
$$\approx -1.0094 \times 10^{-5}$$

$$|\epsilon| \approx 1.0094 \times 10^{-5} < 5 \times 10^{-5}$$

Let  $x = 1.00005000000000000001$ . Then  $|\epsilon| \approx 4.9998 \times 10^{-5} < 5 \times 10^{-5}$

## IEEE 754 cont.

**Example 4.** Assume  $x, y, z$  machine numbers. Find the error in  $\text{fl}(z(x + y))$

$$\begin{aligned}
 \text{fl}(z(x + y)) &= \text{fl}(z) \text{fl}(x + y) (1 + \delta_1) \\
 &= z(\text{fl}(x) + \text{fl}(y))(1 + \delta_1)(1 + \delta_2) \\
 &= z(x + y)(1 + \delta_1)(1 + \delta_2) \\
 &= z(x + y)(1 + \delta_1 + \delta_2 + \delta_1\delta_2) \\
 &= z(x + y)(1 + \delta_1 + \delta_2),
 \end{aligned}$$

where  $|\delta_{1,2}| \leq u$ .  $|\delta_1\delta_2| \ll u$  is very small, so we neglect it

Denoting  $\delta = \delta_1 + \delta_2$

$$\text{fl}(z(x + y)) = z(x + y)(1 + \delta), \quad \text{where } |\delta| \leq 2u$$

## IEEE 754 cont.

**Example 5.** Assume  $x, y$  real. What is the error in  $\text{fl}(xy)$ ?

$$\begin{aligned}
 \text{fl}(xy) &= \text{fl}(x) \text{fl}(y) (1 + \delta_1) \\
 &= x(1 + \delta_2)y(1 + \delta_3)(1 + \delta_1) \\
 &= xy(1 + \delta_1 + \delta_2 + \delta_3 + \underbrace{\delta_1\delta_2 + \delta_1\delta_3 + \delta_2\delta_3 + \delta_1\delta_2\delta_3}_{\text{very small}}) \\
 &\approx xy(1 + \delta_1 + \delta_2 + \delta_3),
 \end{aligned}$$

where  $\text{fl}(x) = x(1 + \delta_2)$ ,  $\text{fl}(y) = y(1 + \delta_3)$ ,  $|\delta_{1,2,3}| \leq u$

Denoting  $\delta = \delta_1 + \delta_2 + \delta_3$ ,

$$|\delta| \leq |\delta_1| + |\delta_2| + |\delta_3| \leq 3u$$

and

$$\text{fl}(xy) = xy(1 + \delta), \quad \text{where } |\delta| \leq 3u$$

### Example 6 (Computing $\sqrt{x^2 + y^2}$ ).

- One can do `sqrt(x*x+y*y)`
- Assume double precision and suppose  $x = 1\text{e}200$  and  $y = 1\text{e}100$
- $x*x$  will overflow and the result is `Inf`
- `sqrt(Inf+1e200)` gives `Inf`
- Let  $M = \max\{|x|, |y|\}$  and assume  $M = |x|$ . Then

$$\sqrt{x^2 + y^2} = M \sqrt{1 + (y/M)^2}$$

- Setting  $M=1\text{e}200$ ,  $y1=y/M$ , compute  $M*\text{sqrt}(1+y1*y1)$ , which gives  $1\text{e}200$

## IEEE 754 cont.

## Note

expression	produces
$y1*y1$	$1e-100$
$1+y1*y1$	1
$\text{sqrt}(1+y1*y1)$	1

# Cancellations

**Example 7.** Assume a decimal FP system with  $t = 5$  digits and rounding to nearest. Let  $x = 1.234567$  and  $y = 1.234512$

- Then

$$\text{fl}(x) = \text{fl}(1.234567) = 1.234\mathbf{6} \quad \text{roundoff error}$$

$$\text{fl}(y) = \text{fl}(1.234512) = 1.234\mathbf{5} \quad \text{roundoff error}$$

$$\text{fl}(x) - \text{fl}(y) = 0.0001 = \mathbf{1.0000} \times 10^{-4} \quad \text{NO roundoff error}$$

- $\text{fl}(x) - \text{fl}(y) = 1.0000 \times 10^{-4}$  has no correct digits: **catastrophic cancellation**
- $x - y = 5.5000 \times 10^{-5}$
- error in  $\text{fl}(x) - \text{fl}(y)$  is  $\frac{[\text{fl}(x) - \text{fl}(y)] - (x - y)}{x - y} \approx 8.1818 \times 10^{-1}$
- Repeat the same computation with  $x = 4.894080$  and  $y = 5.384576$ . Is the error  $\leq$  unit roundoff?



## Cancellations cont.

## Example 8.

- Consider FP arithmetic with 8 decimal digits
- Let  $x = 1$ ,  $h = 10^{-4}$
- Compute  $\text{fl}[\sin(x + h) - \sin(x)]$

$$\text{fl}[\sin(x + h)] = 8.4152501 \times 10^{-1}$$

$$\text{fl}[\sin(x)] = 8.4147098 \times 10^{-1}$$

$$\begin{aligned}\text{fl}[\sin(x + h)] - \text{fl}[\sin(x)] &= 0.0005403 \times 10^{-1} \\ &= 5.4030000 \times 10^{-5}\end{aligned}$$

$$\sin(x + h) - \sin(x) = 5.402602314186211 \times 10^{-5}$$

- We lose 3 digits: cancellations. Relative error  $\approx 7.3 \times 10^{-5}$
- Note there are no roundoff errors in the subtraction

## Cancellations cont.

## Example 9.

- Consider FP arithmetic with 8 decimal digits
- Let  $x = 1$ ,  $h = 10^{-1}$
- Compute  $\sin(x + h) - \sin(x)$

$$\text{fl}[\sin(x + h)] = 8.9120736 \times 10^{-1}$$

$$\text{fl}[\sin(x)] = 8.4147098 \times 10^{-1}$$

$$\begin{aligned}\text{fl}[\sin(x + h)] - \text{fl}[\sin(x)] &= 0.4973638 \times 10^{-1} \\ &= 4.9736380 \times 10^{-2}\end{aligned}$$

$$\sin(x + h) - \sin(x) = 4.9736375 \times 10^{-2}$$

- We lose 1 digit. Relative error  $\approx 9.5 \times 10^{-8} \approx 10^{-7}$
- Note there are no roundoff errors in the subtraction

## Cancellations cont.

**Example 10.** Consider the equivalent expressions  $x^2 - y^2$  and  $(x - y)(x + y)$ . Suppose  $|x| \approx |y|$ . Which one is better to evaluate? Assume  $x, y > 0$ ; the case  $x, y < 0$  is similar

- $x - y$  may have cancellations;  $x + y$  does not
- $x^2$  and  $y^2$  would have (in general) roundoff errors from the multiplications
- due to them, the cancellations in  $x^2 - y^2$  can be worse than in  $(x - y)(x + y)$

Try

```
x = 10000 * rand; y = x * (1 + 1e-14);
eval1 = (x - y) * (x + y); eval2 = x * x - y * y;
%compute more accurate result using
xv = vpa(x); yv = vpa(y); acc = (xv - yv) * (xv + yv);
fprintf('error in (x-y)*(x+y) = % .2e\n', double(acc - eval1));
fprintf('error in x*x - y*y      = % .2e\n', double(acc - eval2));
```