“Yo, I know π to a thousand places”

... but how many places (digits) to **you** need to know?

What to learn (and quiz later)...

- **Binary**
  - convert binary to decimal (>1 and <1)
- **Floating point representation**
  - why is it called that?
  - how are 1.001E3 and 2.2E-1 added?
  - parts of a FP number: names and roles
- **Precision**
  - what is the relative error?
  - how many bits in a byte?
  - how many bytes in a double?
  - In math we have N, Z, Q, R, C
    - Which do we have on computers?
  - Is a+b+c = a+c+b? Why?
- **Roundoff error analysis**
  - what is 1/10 in binary? (why is that interesting?)
  - Which is best: a^2-b^2 or $(a-b)(a+b)$?
  - Which should make you more nervous: $\prod_i x_i$ or $\sum_i x_i$ why?
  - Why is subtracting two nearly equal numbers "bad?"

Binary

In decimal, we write:

$$101325 = \mathbf{1} \cdot 10^{5} + \mathbf{0} \cdot 10^{4} + \mathbf{1} \cdot 10^{3} + \mathbf{3} \cdot 10^{2} + \mathbf{2} \cdot 10^{1} + \mathbf{5} \cdot 10^{0}$$

Convert binary 11010 to decimal

$$110101 = \mathbf{1} \cdot 2^{5} + \mathbf{1} \cdot 2^{4} + \mathbf{0} \cdot 2^{3} + \mathbf{1} \cdot 2^{2} + \mathbf{0} \cdot 2^{1} + \mathbf{1} \cdot 2^{0}$$

$$= 32 + 16 + 0 + 4 + 0 + 1$$

$$= 53$$

What about 101.111?

$$101.111 = \mathbf{1} \cdot 2^{2} + \mathbf{0} \cdot 2^{1} + \mathbf{1} \cdot 2^{0} + \mathbf{1} \cdot 2^{-1} + \mathbf{1} \cdot 2^{-2} + \mathbf{1} \cdot 2^{-3}$$

$$= 4 + 0 + 1 + 1/2 + 1/4 + 1/8$$

$$= 5.875$$

Floating Point

Because the point floats: 101325 =

101325.E0 =

101.325E3 =

1.01325E4

float and **double**

- float = 4 bytes = 32 bits
- **double = 8 bytes = 64 bits**

Computers add like you do:

1.57E2 + 2.3E0 =

157.0E0 + 2.3E0 = ***(same power of 10)***

157.0
+ 2.3

159.3

That is, line up the decimals, then add

Double Precision Representation

S (1)	E (11)	M (52)
0	00000000000	0000000000000000000000000000000000000000000000000000

- Format: $(-1)^S \times 1.M \times 2^{E-1023}$
  - S is a sign bit (1 bit)
  - M is the mantissa (52 bits), the number part
    - Numbers between 0 and $2^{52} - 1$.
    - $2^{52} = 4.5E15 \rightarrow 15 + 1$ **=16 digits of accuracy**
    - Note, binary doubles are *normalized* meaning they are left shifted until the left-most bit is 1. This is assumed, giving a *bit* more accuracy.
  - E is the exponent (11 bits)
    - 11 bits $\rightarrow 2^{11} = 2048 \rightarrow 10^{2048-1}$.
  - The 1023 is a bias (shift), allowing negative exponents.
    - So, instead of 0 to 2047, have roughly 10^{-1023} to 10^{1023}.

From Numerical Recipes

$$S \times M \times b^{E-e}$$

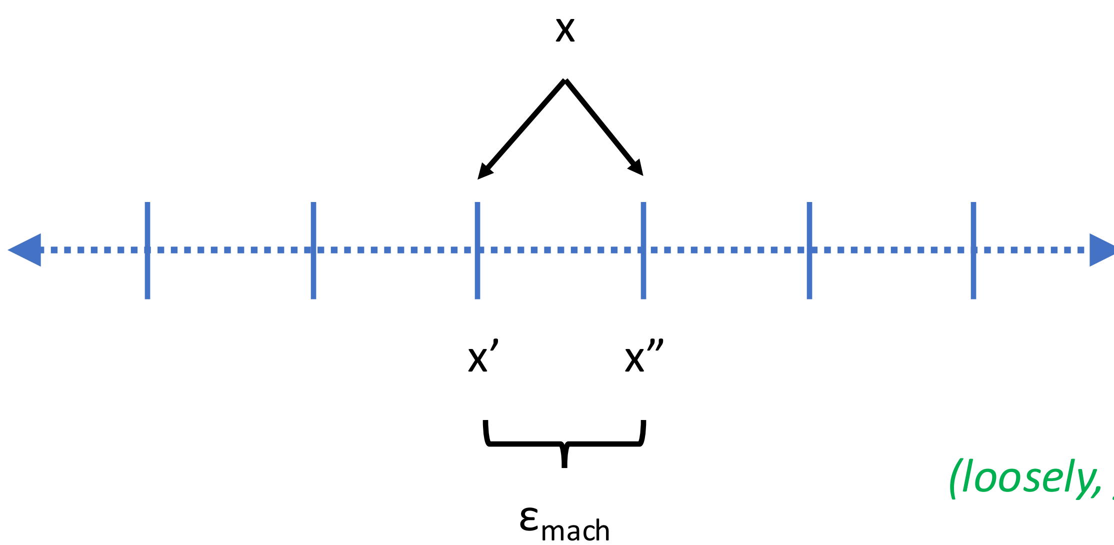
S	E	F	Value
any	1-2046	any	(-1)$^S \times$ 2$^{E-1023} \times$ 1.F
any	0	nonzero	(-1)$^S \times$ 2$^{E-1022} \times$ 0.F
0	0	0	$+0.0$
1	0	0	-0.0
0	2047	0	$+\infty$
1	2047	0	$-\infty$
any	2047	nonzero	NaN

$$\begin{aligned}
0 \;\; 01111111111 \;\; 0000000000000000000000000000000000000000000000000000 &= +1 \times 2^{1023-1023} \times 1.0_2 && = 1 \\
1 \;\; 01111111111 \;\; 0000000000000000000000000000000000000000000000000000 &= -1 \times 2^{1023-1023} \times 1.0_2 && = -1 \\
0 \;\; 01111111111 \;\; 1000000000000000000000000000000000000000000000000000 &= +1 \times 2^{1023-1023} \times 1.1_2 && = 1.5 \\
0 \;\; 10000000000 \;\; 0000000000000000000000000000000000000000000000000000 &= +1 \times 2^{1024-1023} \times 1.0_2 && = 2 \\
0 \;\; 10000000001 \;\; 1010000000000000000000000000000000000000000000000000 &= +1 \times 2^{1025-1023} \times 1.1010_2 && = 6.5
\end{aligned}$$

```python
import bitstring
bitstring.BitArray(float=1.5, length=64).bin
```

```
'0011111111111000000000000000000000000000000000000000000000000000'
```

Roundoff Error



(loosely, for illustration)

Numbers have to be rounded to the nearest number that can be represented

3.14159265358979 3**2**

ε_{mach}

Machine Precision

ε is the smallest number for which fl(1+ ε) > 1

```
import sys

eps = sys.float_info.epsilon

eps
2.220446049250313e-16

2**-52
2.220446049250313e-16

1.0+eps
1.0000000000000002

1.0+eps/2
1.0
```

ε is the "relative error"

RE = (# - $\#_{exact}$) / $\#_{exact}$

Suppose ε_{mach} = 0.001,

$$RE = \frac{1.001 \times 10^6 - 1.000 \times 10^6}{1.000 \times 10^6}$$

$$= \frac{0.001 \times 10^6}{1.000 \times 10^6} = 0.001 = \epsilon_{mach}$$

The exponent part cancels, so ε_{mach} is just the smallest nonzero number in the mantissa

Roundoff error

- Floating point operations: $x\Box y$, where $\Box$ is one of + - * /
  - $fl(x\Box y) = round(x\Box y)$, that is, have to round.
  - $x + y \rightarrow$ need the same exponents $\rightarrow$ lose digits of the smaller number.
    - Suppose we had 4 digits to work with: $1000. + 7.200 \rightarrow 1.000E3 + 0.0072E3 \rightarrow 1.007E3$. So we lose the 2.
  - $x * y \rightarrow$ Add the exponents and multiply the mantissas $\rightarrow$ rounding error, but not as severe.
  - $a + b = b + a$, but $(a + b) + c \neq a + (b + c)$.
    - Commutative, but not associative.
    - For example, for $\epsilon < \frac{1}{2}\epsilon_{mach}$, $(1 + \epsilon) + \epsilon = 1$, but $1 + (\epsilon + \epsilon) > 1$.
    - Or, another way: suppose $|\epsilon| < \epsilon_{mach}$, then $(1 + \epsilon) - (1 - \epsilon) = 2\epsilon$, but on a computer it is 0

Roundoff error disasters

The Patriot and the Scud.

Sources
1. General Accounting Office Report GAO/IMTEC-92-26.
2. Robert Skeel, "Roundoff Error Cripples Patriot Missile," SIAM News, July 1992.

On February 25, 1991, during the Gulf War, a Patriot missile defense system let a Scud get through. It hit a barracks, killing 28 people. The problem was in the differencing of floating point numbers obtained by converting and scaling an integer timing register. The GAO report has less than the full story. For that see Skeel's excellent article.

https://web.ma.utexas.edu/users/arbogast/misc/disasters.html

Roundoff error disasters

The Vancouver Stock Exchange.

Sources

1. The Wall Street Journal November 8, 1983, p.37.
2. The Toronto Star, November 19, 1983.
3. B.D. McCullough and H.D. Vinod Journal of Economic Literature Vol XXXVII (June 1999), pp. 633-665. (References communicated by Valerie Fraysse)

In 1982 (I figure) the Vancouver Stock Exchange instituted a new index initialized to a value of 1000.000. The index was updated after each transaction. Twenty two months later it had fallen to 520. The cause was that the updated value was truncated rather than rounded. The rounded calculation gave a value of 1098.892.

https://web.ma.utexas.edu/users/arbogast/misc/disasters.html

Roundoff error disasters

Parliamentary elections in Schleswig-Holstein.

Source

1. Rounding error changes Parliament makeup, Debora Weber-Wulff, The Risks Digest, Volume 13, Issue 37, 1992.

In German parliamentary elections, a party with less than 5.0% of the vote cannot be seated. The Greens appeared to have a cliff-hanging 5.0%, until it was discovered (after the results had been announced) that they really had only 4.97%. The printout was to two figures, and the actual percentage was rounded to 5.0%.

https://web.ma.utexas.edu/users/arbogast/misc/disasters.html

Floating Point Analysis

See Jupyter Notebook