Representing real numbers in the computer

Floating point

$\underbrace{3.271}_{\text{mantissa}} \times \underbrace{10}_{\text{base}}^{-4}$

Signed integer exponent

$\left.\begin{array}{l} 32.71 \times 10^{-5} \\ 0.327 \times 10^{-3} \end{array}\right\}$ point can "float"

scientific notation.

Usually, we write 3.271×10^{-4}

only one digit to the left of the decimal point.

If exponent is 0, we don't usually write 3.271×10^{0} but instead write just 3.271

Scientific Notation vs fixed point

5.781×10^{-4} vs 0.0005781

Which can store more information in least amount of space?

Floating Point $\underbrace{5.781}_{\text{4 digits}} \times 10^{\overset{\text{one exponent}}{-4}}$

Fixed point. $\underbrace{0.0005781}_{\text{7 digits}}$ ← hard to store really small numbers or really large numbers.

Format statements

Matlab

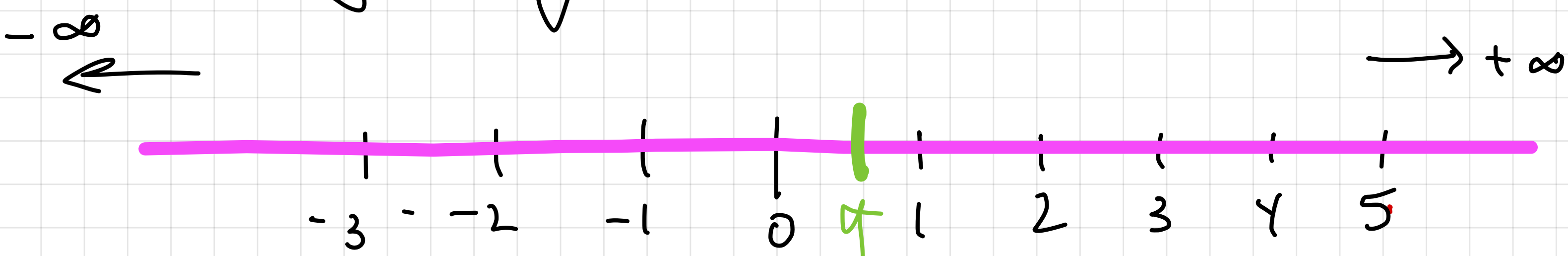
fprintf('%.4e', 5.781e-4)

fprintf('%.8f', 5.781e-4)

```
>> x = 5.781e-4;
>> fprintf('%.4e\n',x)
5.7810e-04
>> fprintf('%.7f\n',x)
0.0005781
```

Digital representation of real numbers

① infinity many real numbers

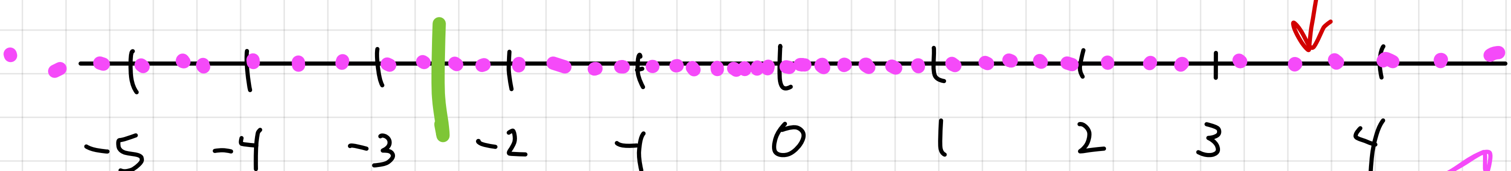


always hit a real number

"infinite precision"

② We can only represent a finite set of real numbers on the computer.

gaps



numbers are more densely packed near 0

might not hit exact representation.

less dense for large positive/negative numbers.

What numbers can we represent on the computer?

Numbers that can be represented in binary as:

$$\pm \left(1 + b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \cdots \right) \times 2^{e}$$

where $b_i = 0$ or 1, $i = 1, 2, 3, \ldots$ and

where e is an integer. $e = \pm 1, \pm 2, \pm 3, \ldots$

Example: $x = 0.5$ (in decimal)

$x = 1.0 \times 2^{-1}$ (base 2) $b_1 = b_2 = b_2 = \ldots$

$e = -1$

Example: $x = 2$

$x = 1.0 \times 2^{1}$ $b_1 = b_2 = b_3 = \ldots = 0$

$e = 1$

Example:

$13 = 1 + 4 + 8 = 1 + 2^2 + 2^3$

$.25 = 2^{-2}$:

$$X = 13.25 = \begin{aligned} 13.25 &= 2^{-2} + 1 + 2^2 + 2^3 \\ &= \underbrace{2^3\left(2^{-5} + 2^{-3} + 2^{-1} + 1\right)}_{\text{factor out } 2^3} \end{aligned}$$

$$= \left(1 + 2^{-1} + 2^{-3} + 2^{-5}\right) \times 2^3$$

$b_1 = 1 \quad b_2 = 0 \quad b_3 = 1 \quad b_4 = 0 \quad b_5 = 1$

$b_6 = b_7 = \cdots = 0$

$e = 3$

Example: $X = 0.1$

$X = 0.1 = 1 \times 10^{-1} \Rightarrow$ cannot be represented exactly!

$$X = \left(1 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} + \ldots\right) \times 2^{-4}$$

continues for ever!

$$= \underbrace{0.09999\ldots\ldots = 0.0\overline{9}}_{= 0.1}$$

Example: $X = 0$

$$(1 + \cdots) \times 2^0$$

do we have to have this 1 here? ☹

Example: $X = -5$

How do we get a negative value?

How do we encode the binary representation on the computer?

- Need a "signed exponent"
- Need a mantissa
- Need a $\pm$ sign
- Need a 0

- Encode as a string of 0s and 1s

IEEE Floating point standard. 754 standard.

1 bit	11 bits	52 bits
sign ±	exponent	mantissa

"64-bit precision" = "double precision"

Sign bit : 1 bit   0 : positive
1 : negative

exponent : 11 bits   "(biased)" to get negative exponents

mantissa : $b_1 b_2 b_3 \cdots b_{52}$

More on the exponent

11 bits allows us to represent

$$E = e_0 2^0 + e_1 2^1 + e_2 2^2 + \cdots e_{10} 2^{10}$$

$e_i = 0$ or 1

Example: $E = 6$

e_{10} ... e_3 e_2 e_1 e_0

$$E = 2 + 4 = 2^1 + 2^2$$

$$e_0 = 0 \quad e_1 = e_2 = 1 \quad e_3 = e_4 = \cdots e_{10} = 0$$

Smallest E : $e_0 = e_1 = \cdots e_{10} = 0$ $\quad \underline{E = 0}$

Largest E : $e_0 = e_1 = \cdots e_{10} = 1$

$$E = 2^{11} - 1 = 2047$$

But how do we get a negative exponent? Shift the exponent

First half represent negative numbers

second half represents positive numbers

$$\underbrace{0 \quad 1 \quad 2 \quad 3 \quad 4 \quad \cdots\cdots\cdots \quad 2047}_{\text{2048 possible values for } E.}$$

Designate $0, 1, \ldots 1022$ as "negative"

1023 as 0

$1024, 1025, \ldots 2047$ as "positive"

"bias the exponent" $e = E - 1023$

To represent $X = 0.5$ $\quad$ $E = e + 1023$

$$X = 1 \times 2^{-1} \qquad e = -1$$

$$E - 1023 = -1 \Rightarrow E = 1022$$

$$1022 = 2^1 + 2^2 + 2^3 + \cdots\cdots 2^9 \qquad \begin{matrix} 1024 - 2 \\ 2^{10} - 2 \end{matrix}$$

$$E = \underline{e_{10}}\ \underline{e_9}\ \underline{e_8}\ \underline{e_7}\ \underline{e_6}\ \underline{e_5}\ \underline{e_4}\ \underline{e_3}\ \underline{e_2}\ \underline{e_1}\ \underline{e_0}$$

$$= (0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0)_b$$

Full representation

$$- \,|\, \underline{e_{10}}\ \_\ \_\ \_\ \_\ \_\ \_\ \_\ \_\ \underline{e_2}\ \underline{e_1}\ \underline{e_0} \,|\, \underline{b_1}\ \underline{b_2}\ \underline{b_3}\ \_\ \_\ \_\ \ldots\ \underline{b_{52}} \,|$$

sign bit

exponent

mantissa

"normalized mantissa"

$$1.b_1 b_2 b_3 \ldots b_{52}$$

"1" is assumed

Example: $X = 0.5$

$$X = 1.0 \times 2^{-1}$$

$$= (0 \,|\, \underbrace{01111111110}_{\text{biased exponent } = 1022} \,|\, \underbrace{00000 \ldots 0}_{52\ 0\text{s}} \,|)_b$$

'+'

Example $x = 2$

$10^3 - 1$

$$\begin{array}{r} 1000 \\ -\quad 1 \\ \hline 0999 \end{array}$$

$$x = 1.0 \times 2^1$$

exponent: $1 = 1 \times 2^0$

$$e = E - 1023 = 0$$

$$E = 1023 = \underbrace{2^{10} - 1}$$

?? !!

0 1 1 1 1 1 1 1 1 1 1

$$(0|01111111111|0000\,0\cdots\cdots 0)_b$$

+1 $E = e - 1023$ mantissa = 52 0s

$$1 = 2 - 1 = 2^1 - 1 = (01)_b$$

$$3 = 4 - 1 = 2^2 - 1 = (011)_b$$

$$7 = 8 - 1 = 2^3 - 1 = (0111)_b$$

$$15 = 16 - 1 = 2^4 - 1 = (01111)_b$$

$$1023 = 1024 - 1 = 2^{10} - 1 = (01111111111)_b$$

How do we represent 0 ?

$x =$

$$0 \,|\, 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \,|\, 0\,0\,0\,0\,0\,0 \ldots 0$$

exponent — mantissa

means: $x = 0$

Not $E = 0 \Rightarrow e = E - 1023$
$= -1023$

mantissa 1.0 ?

$$x = \cancel{1.0 \times 2^{-1023}}$$ NO!

The IEEE standard tells us how 0 is represented $\Rightarrow$ all bits set to 0.

The mantissa is "denormalized"

If all bits in the exponent are 0, then we drop the "1" on the mantissa.

not 1023

instead of $1.0 \times 2^{-1023} \Rightarrow 0.0 \times 2^{-1022}$

Denormalized numbers

non-zero

$$\underline{0} \;|\; \underbrace{0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0}_{E=0} \;|\; \underbrace{0\,0\,1\,0\,0\,0\,0 \cdots\cdots 0}_{b_3 = 1 \;\neq 0}$$

If $E=0 \Rightarrow$ exponent is -1022 (not -1023)

Drop implied "1" on mantissa

$\Rightarrow$ Get a "denormalized number"

denormal number

$$0.125 \times 2^{-1022} \approx 10^{-308}$$

2^{-3}

Denormalized numbers are less precise, because they will have fewer digits in the mantissa.

Something like: 0.00000034

vs $\underbrace{3.4713671}_{\text{these extra digits are lost.}} \times 10^{-7}$

Other special numbers in the IEEE standard

±0 – all bits set to 0
– possible to have +0 -0 depending on how sign bit is set.

±∞ – all exponent bits set to 1
– all mantissa bits set to 0
– sign bit can be 1 or 0

NAN – all exponent bits set to 1
– mantissa bits can be set to any non-zero value
– "quiet Nans" – propagate through calculations
– "signaling Nans" – signals a "floating point exception"

"not a number"

the end