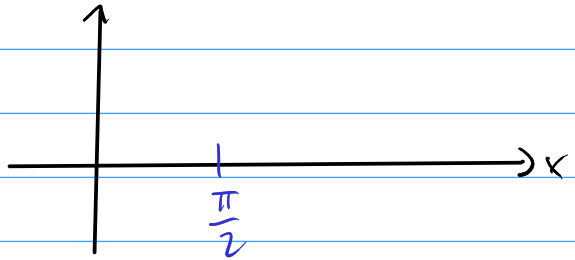
$$K \approx \left| \frac{x \cdot f'(x)}{f(x)} \right|$$



$f(x) = \sin x \rightarrow \dfrac{x \cdot \cos(x)}{\sin(x)}$

biggest near 0 $\rightarrow \dfrac{x \cdot 1}{x} \approx 1$

Stability and Accuracy

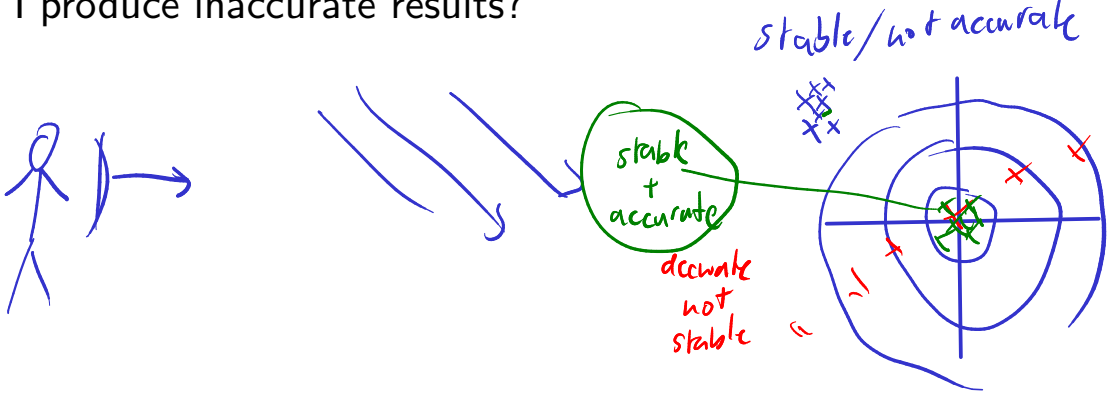
- When is a method stable?

  If it sensitivity to variation in input is no (or not much) greater than that of the underlying problem.

- When is a method accurate?

  Closeness of method output to the acutal answer for completely accurate input

- How can I produce inaccurate results?



1.2 Floating Point

Wanted: Real Numbers... in a computer

- Computers can represent *integers*, using bits:

$$23 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (10111)_2$$

How would we represent fractions?

16 8 4 2 1

1 0 1 1 1

$$23.625 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

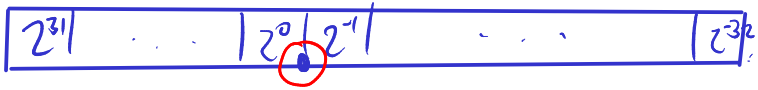
0.5 0.25 .125

.125 .125 0

Fixed-point arithmetic

Fixed-Point Numbers

- Suppose we use units of 64 bits, with 32 bits for exponents $\geqslant 0$ and 32 bits for exponents <0. What numbers can we represent?



biggest: $2^{32} - 2^{-32}$

smallest: 2^{-32}

- How many 'digits' of relative accuracy (think relative rounding error) are available for the smallest vs. the largest number?

Largest: 64 bits: $2^{64} \approx 10^{19}$ $\rightsquigarrow$ 19 digits

Smallest: 0 bits $\rightsquigarrow$ 0 digits

Floating Point numbers

- Convert $13 = (1101)_2$ into floating point representation.

$13 = (1.101)_2 \cdot 2^3$

lg. 10^{15}

- What pieces do you need to store an FP number?

$(1.101)_2$ → "significand" stored; $-1023 + \underline{1026}$

1 = not stored

101 = stored

3 → "exponent"

+/− → sign

$12.25 = 1100.01$

$1.10001 \cdot 2^3$

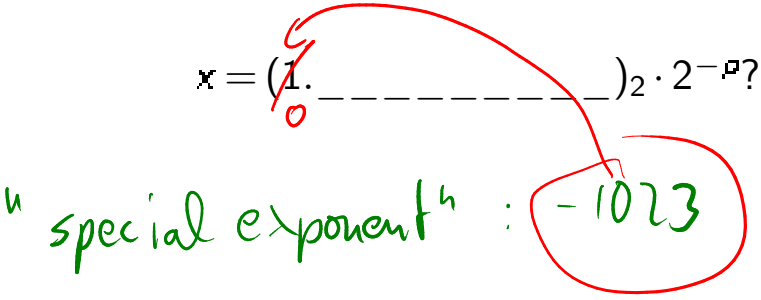
$.125 = 2^{-3}$

In-class activity: Floating Point

Unrepresentable numbers?

- Can you think of a somewhat central number that we cannot represent as

$$x = (1.\_\_\_\_\_)_2 \cdot 2^{-\square}?$$



Demo: Picking apart a floating point number