

## Digits and Rounding

Establish a relationship between '*accurate digits*' and rounding error.

$$\pi = 3.1415\dots$$

$$\bar{\pi} \approx 3.142$$

$\underbrace{\hspace{2cm}}_{n=4}$

$$\text{relative error} = \frac{\pi - \bar{\pi}}{\pi} \leq 5 \cdot 10^{-4} \leq 10^{-3}$$

## Condition Numbers

Methods  $f$  take input  $x$  and produce output  $y = f(x)$ .

Input has (relative) error  $|\Delta x| / |x|$ .

Output has (relative) error  $|\Delta y| / |y|$ .

**Q:** Did the method make the relative error bigger? If so, by how much?

$$\kappa = \max_x \left( \frac{\text{relative change in } f(x)}{\text{rel. perturbation to } x} \right)$$
$$= \max_x \left( \frac{|f(x + \Delta x) - f(x)|}{|f(x)|} \bigg/ \frac{|\Delta x|}{|x|} \right)$$

absolute condition number

$$K_{abs} = \max_x \left( \frac{|f(x+\Delta x) - f(x)|}{|\Delta x|} \right)$$

$$= \max_{\text{input}} \left( \frac{\text{absolute change of output}}{\text{size of perturbation to input}} \right)$$

$$K(x) = \frac{|x| |f'(x)|}{|f(x)|}$$

$$f(x) = a + bx + cx^2 + \dots + y \cdot x^n$$

as  $x \rightarrow \infty$ ,  $f(x) \approx yx^n$

$$K(x) \approx \frac{|x| |nyx^{n-1}|}{|yx^n|} = n$$

## $n$ th-Order Accuracy

Often, *truncation error* is controlled by a parameter  $h$ .

Examples:

- ▶ distance from expansion center in Taylor expansions
- ▶ length of the interval in interpolation

A numerical method is called ' *$n$ th-order accurate*' if its truncation error  $E(h)$  obeys

$$E(h) = O(h^n).$$

# Outline

Python, Numpy, and Matplotlib  
Making Models with Polynomials  
Making Models with Monte Carlo

Error, Accuracy and Convergence

## Floating Point

Modeling the World with Arrays

The World in a Vector

What can Matrices Do?

Graphs

Sparsity

Norms and Errors

The 'Undo' Button for Linear Operations: LU

LU: Applications

Linear Algebra Applications

Interpolation

Repeating Linear Operations:  
Eigenvalues and Steady States

Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

SVD: Applications

Solving Funny-Shaped Linear Systems

Data Fitting

Norms and Condition

Numbers

Low-Rank Approximation

Iteration and Convergence

Solving One Equation

Solving Many Equations

Finding the Best: Optimization in 1D

Optimization in  $n$  Dimensions

## Wanted: Real Numbers... in a computer

Computers can represent *integers*, using bits:

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

How would we represent fractions, e.g. 23.625?

$$23.625 = 10111.101$$

$$\underbrace{\hspace{10em}}_{1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}}$$

10111.101      fixed-point

$$\underbrace{\hspace{10em}}_{32} . \underbrace{\hspace{10em}}_{32}$$

## Fixed-Point Numbers

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

$$2^{31} + 2^{30} + \dots + \frac{2^{-32}}{2^{-32}} \approx 2^{32} - 2^{-32} \approx 10^9$$

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

9 decimal digits

$$\underbrace{2^{31} \quad 2^{30} \quad \dots \quad 2^{-32}}_{1 \text{ accurate digit for } 2^{-32}}$$



## Floating Point numbers

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

$$1.2345 \dots \times 10^k$$

$$1.101 \times 2^3$$

$\underbrace{1101}_{\text{significant}} \underbrace{11}_{\text{exponent}}$

What pieces do you need to store an FP number?

Significant - binary digits  
 $s_0 \dots s_n$

exponent - magnitude  
 $e_1 \dots e_k$

$n+k \approx \text{bits}$

$$s_0.s_1s_2s_3\dots s_n \cdot 2^{e_1\dots e_k}$$

$$s_0 = 1 \quad s_1, \dots, s_n = 0$$

$$e_1, e_2, \dots, e_k = 1$$

$$1.0 \cdot 2^{\overbrace{k-1}^{11111}} = 2^{k-1}$$

$$k=10 \quad 2^{1024-1} = 2^{1023}$$

$$1.01011$$

significand

$$1.s_0s_1\dots s_n \cdot 2^{e_1\dots e_k}$$

$$= 0.1s_0\dots s_n \cdot 2^{e_1\dots e_k + 1}$$

$$1011101.11001$$

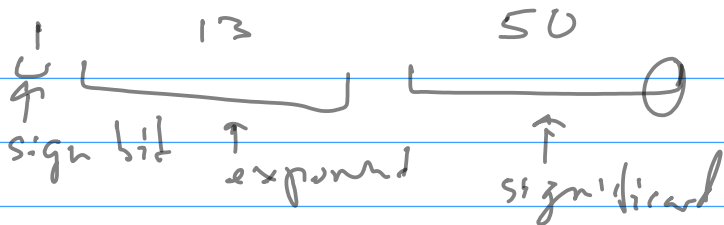
$$\begin{array}{l} \rightarrow \\ \end{array} 1.01110111001 \times 2^6$$

$$.0001101$$

$$\begin{array}{l} \leftarrow \\ \end{array} -4$$

$$1.101 \cdot 2$$

64-bit



if exponent = 0000000

$$\rightarrow 2^{-2^{13}}$$

roundoff error based machine

$\epsilon$  is smallest such epsilon  
that  $fp(1 + \epsilon) \neq fp(1)$

$$\text{exp} = 000 \Rightarrow 2^{-4}$$

$$100 \Rightarrow 2^0$$

$$111 \Rightarrow 2^3$$

$$f_p(1) = \underbrace{0}_{\text{sign}} \underbrace{100}_{\text{exp}} \underbrace{0 \dots 0}_{\text{sig.}}$$

$$\text{exp} = 100 \Rightarrow \cancel{2^0} \quad 2^{-4}$$

$$5 = 1.01 \cdot 2^2$$

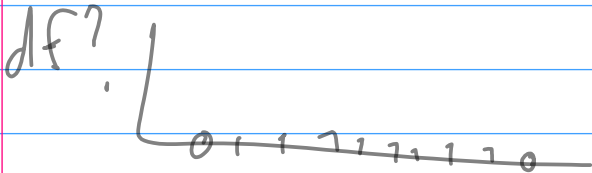
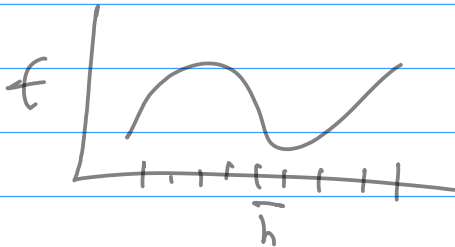
"M  
s<sub>1</sub> s<sub>2</sub>

$$6 = 1.10 \cdot 2^2$$

s<sub>1</sub>

$$7 = 1.11$$

$$8 = 1.00 \cdot 2^3 \quad 9$$



$$f_p(1) = \underbrace{0}_{\text{sign}} \underbrace{000}_{\text{exponent}} \underbrace{000000}_{\text{significand}}$$

$$f_p(1 + \epsilon) = 0 \quad 000 \quad 00001$$

$$\begin{aligned}
 f_p(\text{exp}) = 0/00 &\Rightarrow 2^{-4} \Rightarrow 2^{-111} \\
 &= 1/00 \Rightarrow 2^0 \\
 &= 1/11 \Rightarrow 2^3
 \end{aligned}$$



## **In-class activity:** Floating Point