

## Lecture 03: Sources of Error cont'd

### Life in *Floating-Point Land*

#### Outline

- 1) Truncation error vs. Floating Point Error
- 2) Floating Point Systems:
  - Basic definitions
  - Basic Operations
  - Rounding vs. Chopping
- 3) Example: a  $p=1$ ,  $p=2$  toy decimal system
  - Machine epsilon
- 4) IEEE single and double Precision floating point
- 5) The great exp challenge -- truncation & Floating Point error

## Review: Sources of Error

- 1) "Truncation Error": e.g. Errors arising from approximating a function with a simpler function e.g.

$$\sin(x) \approx x - x^3/3! + x^5/5! + \mathbf{O}(h^7)$$

- 2) "Floating Point Error": Errors arising from approximating real numbers with finite-precision numbers e.g.

$$\pi \approx 3.14 \text{ (a 3 digit floating-point systems)}$$

## Floating Point Systems: definitions

All *normalized floating point systems* can be described as the following

$$\hat{f} = \pm d_1.d_2d_3d_4\dots d_p \times \beta^E$$

where:

- 1)  $\pm$  is a sign bit
- 2)  $d_1.d_2\dots d_p$  is the *Mantissa* (and  $.d_2\dots d_p$  is the "*fraction*" with  $p$  digits of precision (normalized FP systems assumes  $d_1 \neq 0$ ))
- 3)  $\beta$  is the *base* (e.g. binary fp  $\beta=2$ , decimal  $\beta=10$ )
- 4)  $E$  is the *exponent*, an *integer* in the range  $[E_{\min}, E_{\max}]$

Comments:

- 1) Floating point systems form a *discrete and finite set* of numbers
- 2) Floating point numbers are *not* uniformly distributed on the real line
- 3) Arithmetic in FP systems is different from real math

## Example: a Toy system

Consider the toy 2-digit precision decimal systems

$$f = \pm \underbrace{d_1}_{\text{red}} \underbrace{d_2}_{\text{blue}} \times 10^E \quad \text{Normalized}$$

with  $E$  in  $[-2, 0]$

$$\begin{aligned} \text{URL} &= 1 \times 10^{-2} \\ \text{URL} &= 9 \times 10^0 \end{aligned}$$

- 1) How many numbers in  $f$ ? 270
- 2) Distribution on the real line



## Example: a Toy system

### Basic Arithmetic

Multiplication

1 digit system

$$\begin{array}{r} 6 \times 10^{-1} \\ 8 \times 10^{-1} \\ \hline 4.8 \times 10^{-1} \end{array}$$

chop  $4 \times 10^{-1}$   
round  $5 \times 10^{-1}$

$$\begin{array}{r} 8 \times 10^{-1} \\ 5 \times 10^6 \\ \hline 4 \times 10^6 \end{array} = \text{inf}$$

$$4 \times 10^{-2} \quad 1 \times 10^{-2} \quad 1 \times 10^{-2} = 10^2 \text{ inf}$$

## Example: a Toy system

### Basic Arithmetic

Addition

$$\begin{array}{r} 6 \times 10^{-1} \\ + 8 \times 10^{-1} \\ \hline 1.4 \times 10^0 = 1 \times 10^0 \end{array}$$

$$a, b \quad \bar{x} = \frac{(a+b)}{2} = \frac{1 \times 10^0}{2} = 5 \times 10^{-1}$$

$$\bar{x} = a + \frac{(b-a)}{2} = 6 \times 10^{-1} + \frac{2 \times 10^{-1}}{2} = 7 \times 10^{-1}$$

$$\begin{array}{r} 1 \times 10^0 \\ + 1 \times 10^{-2} \\ \hline 1.01 \times 10^0 \end{array}$$

$\epsilon_{mach}$  = the number such that  
 $|+ \epsilon_{mach} > | \epsilon_{mach} = |$

Danger

adding large + small #'s lose precision

cancellation

$$\begin{array}{r} 3.1415 \\ 3.1414 \\ \hline 1 \times 10^{-4} \end{array}$$

$$p=2 \quad \begin{array}{r} 5.6 \times 10^{-1} \\ 5.8 \times 10^{-1} \end{array}$$

## Example: a Toy system

Basic Arithmetic

machine epsilon:

## Properties of Floating Point Systems

All FP systems are characterized by several important numbers

- 1) a smallest normalized number (underflow)

$$\text{UFL} = 1.000\dots x \beta^{E_{\min}}$$

*OFL +  $\epsilon_{\text{mach}}$   
= inf*

- 2) a largest normalized number (overflow)

$$\text{OFL} = d.\text{dddddd}\dots x \beta^{E_{\max}} \quad (\text{where } d = \beta - 1)$$

- 3) Zero:

$$0 = 0.000000\dots x \beta^0$$

- 4) Machine epsilon  $\epsilon_{\text{mach}}$

the smallest number such that  $1 + \epsilon_{\text{mach}} > 1$

- 5) Inf and NaN:

*o/o*

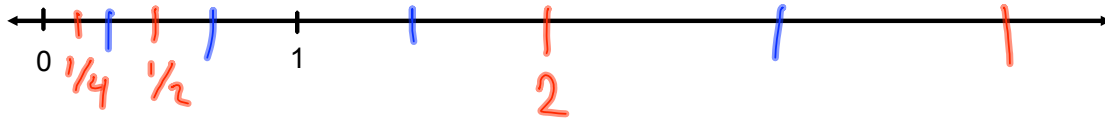
- 6) Subnormal numbers

$$f = 0.d_1 d_2 \dots d_p \times \beta^{E_{\min}}$$

## Binary Systems:

Consider the toy 2-digit precision base 2 system

$$f = \pm d_1.d_2 \times 2^E \quad \text{with } E \text{ in } [-2,2]$$



## Real systems: IEEE 754 Binary floating point systems

Single Precision: total storage 32 bits

Exponent 8 bits:  $E = [-126, 127]$

Fraction 23 bits ( $p=24$ )

S EEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFF  
01 89 31

$$\text{OFL} = 2^{127} \sim 3.4 \times 10^{38}$$

$$\text{UFL} = 2^{-126} \sim 1.2 \times 10^{-38}$$

$$\epsilon_{\text{mach}} = 2^{-23} \sim 1.2 \times 10^{-7}$$

$$\text{smallest subnormal } 2^{-149} \sim 1.4 \times 10^{-45}$$



**Big Points:**

Floating point math is not commutative or associative

Floating point errors can destroy precision

even IEEE double precision can break

Some Examples:

$$\varepsilon < \varepsilon_{mach}$$

$$e = \frac{1}{2} \times eps$$

$$(1 + \varepsilon) - 1 =$$

$$1 - 1 = 0$$

$$1 - 1 + \varepsilon = \varepsilon$$

$$A = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \quad \underline{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$A\underline{x} = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Solve  $\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$   $\varepsilon < \varepsilon_{mach}$

$$\begin{bmatrix} \varepsilon & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} \varepsilon & 1 & 1 \\ 0 & 1 - \frac{1}{2} & 2 - \frac{1}{2} \end{bmatrix}$$

$$\begin{bmatrix} \varepsilon & 1 & 1 \\ 0 & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix} \quad -\frac{1}{2}y = -\frac{1}{2} \quad y = 1$$

$$\varepsilon x + 1 = 1 \quad x = 0$$

$$\underline{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

1 ulps  $r = \frac{|f - \hat{f}|}{|f|} = \frac{|f - \hat{f}|}{eps(f)}$

## Floating Point and Truncation Error

In many numerical schemes both truncation error and floating point error can contribute to the overall error

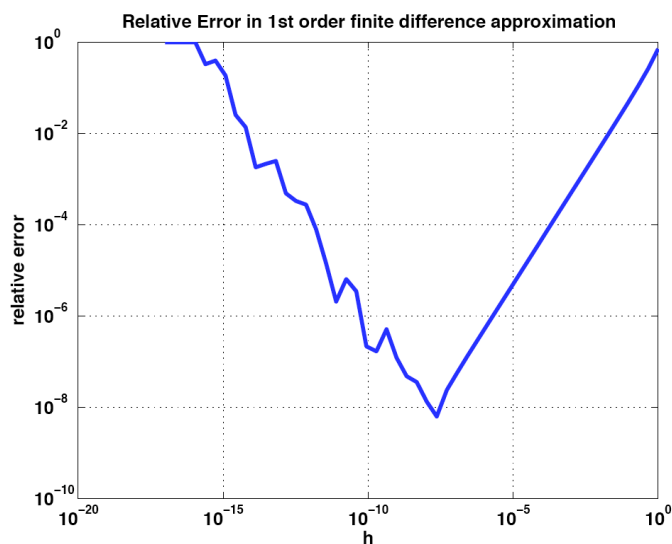
Example #1: Finite Difference approximation to  $f'(x)$

let  $f(x)=\exp(x)$ ,  $f(1)=e$ ,  $f'(1)=e$

## Floating Point and Truncation Error

Example #1: Finite Difference approximation to  $f'(x)$

let  $f(x)=\exp(x)$ ,  $f(1)=e$ ,  $f'(1)=e$



## Floating Point and Truncation Error

Example #2: The great  $\exp(x)$  challenge:

given the Taylor polynomial expansion of  $\exp(x)$  around 0

$$\exp(x) \sim 1 + x + \frac{x^2}{2!} + \dots + \sum_{n=0}^N \frac{x^n}{n!} + R_N$$

provide a matlab function `estimateExp(x)` such that the average relative error is less than 8 ulps over the range  $x = [-20,20]$  (that's not that hard...overachievers can do this on  $x=[-700,700]$ ).

A simple (but not ideal way to try)

```
N=100;  
n=0:N;  
estimateExp = @(x) sum(x.^n./factorial(n))
```