Math 381 - Fall 2020

Jay Newby

University of Alberta

Week 2

Last Week

1. Course overview
2. eClass
3. Python
4. Jupyter
5. Programming
6. Hardware
7. Numerical errors

This Week

1. Representing numbers on a computer
2. Floating point overflow and underflow
3. Roundoff error
4. Examples

Review from last time

Suppose we have the exact solution to a problem $x \neq 0$, and an approximate solution $\hat{x} \in \mathbb{R}$.

Absolute Error:

$$\mathcal{E}_{\text{abs}} = |\hat{x} - x|$$

Relative Error:

$$\mathcal{E}_{\text{rel}} = \frac{|\hat{x} - x|}{|x|}$$

Representing integers

Instead of learning a unique symbol of every integer, we learn 10 symbols

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Polynomial representation of integers

Let r_j, $j = 1, \ldots, N$ be a sequence of base β integers. An integer $n \in \mathbb{N}$ can be represented as

between 0 & $\beta - 1$

102

$$n = \pm \sum_{j=0}^{N} r_j \beta^j$$

$\{r_0, r_1, r_2, \ldots$

How many symbols does a computer know?

Binary Example

Let $\beta = 2$ then

$$n = \pm \sum_{j=0}^{N} r_j 2^j$$

How about $n = 1074$?

$$r_{10-j} = \{10000110010\}$$

Representing Real numbers

Examples 2.5, $2+\frac{1}{2}$

How do we represent real numbers with pencil and paper?

$d_0 > 0$

6.023×10^{23} (normalized)

0.6023×10^{24}

mega 10^{6}

micro 10^{-6}

Representing Real numbers

$0.\overline{33}$ $\quad$ $0.\bar{3}$

Any real number $x \in \mathbb{R}$ can be represented by three components: a sign $\pm$, an infinite sequence of base β digits d_j, $j = 0, 1, \ldots$ called the *mantissa*, and an exponent β^e for integer e. Thus, we can write

$$x = \pm \left(\sum_{j=0}^{\infty} \frac{d_j}{\beta^j} \right) \times \beta^e,$$

where

$$d_j \in \{0, 1, \ldots \beta - 1\}.$$

$d_0 > 0$

Example for base 10

If $d_t \in \{0, 1, \ldots 9\}$, (base 10) then

$$x = \pm \left(\frac{d_0}{1} + \frac{d_1}{10} + \frac{d_2}{10^2} + \cdots + \frac{d_t}{10^t} + \cdots \right) \times 10^e,$$

where e is an integer exponent.

Example for binary

If $d_t \in \{0, 1\}$, (base 2) then $d_0 > 0$ by convention

$$x = \pm \left(1 + \frac{d_1}{2} + \frac{d_2}{2^2} + \cdots + \frac{d_t}{2^t} + \cdots \right) \times 2^e,$$

where e is an integer exponent.

Floating Point Number

A computer can store a finite number of bits ($\{0, 1\}$ values) to represent a single floating point number

$$\mathrm{fl}(x) = \mathrm{sign}(x) \times (1.\tilde{d}_1\tilde{d}_2\tilde{d}_3 \cdots \tilde{d}_{t-1}\tilde{d}_t) \times 2^e,$$

where e is an integer exponent and $\tilde{d}_n \in \{0, 1\}$.

Single and double precision

- Double Precision (Python default, AKA 'float64') 64bits total, with 1bit for the sign, $t = 52$ for the matissa, and 11bits to store the exponent e. The exponent is bounded by $L \leq e \leq U$, where $L = -1022$ and $U = 1023$.

- Single Precision (AKA 'float32') 32bits total, with 1bit for the sign, $t = 23$ for the matissa, and 8bits to store the exponent e (where $L = -126$ and $U = 127$).

Floating Point Number: roundoff error

For IEEE standard

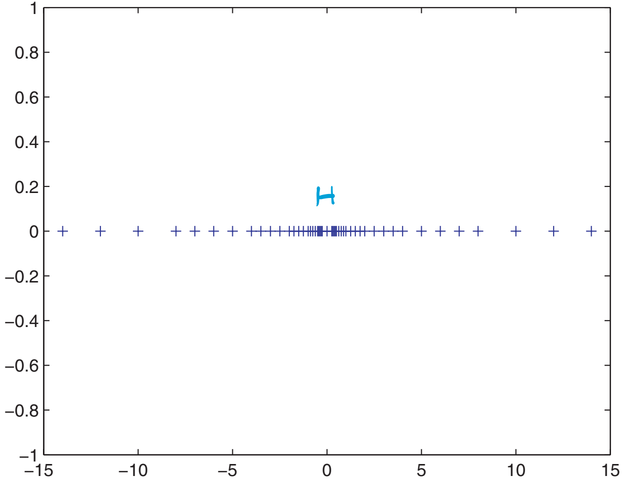
$$\mathcal{E}_{\text{rel}} = \frac{|\text{fl}(x) - x|}{|x|} \leq 2^{-t-1}$$

machine epsilon : ε (book uses η)

float32 : $\varepsilon \approx 10^{-8}$

float64 : $\varepsilon \approx 10^{-16}$

Distribution of floating point numbers on the real line



Smallest and largest possible (absolute) value of float32

Special numbers

$b = 0 \Rightarrow e = 0 - 127 = -127$ ← zero & subnormal we wont study

$b = 255 \Rightarrow e = 255 - 127 = 128$ ← NaN & inf

"not a number" like $1/0$ or $\log(0)$

Recall $L \le e \le U$, $L = -126$, $U = 127$

Let $e = b - 127$, where b is a non negative 8-bit integer (uint8)

Largest absolute value: $x^{(32)}_{\text{largest}} = (1.\tilde{d}_1\tilde{d}_2\tilde{d}_3 \cdots \tilde{d}_{t-1}\tilde{d}_t) \times 2^{127} \approx 10^{38}$

Smalest absolute value: $x^{(32)}_{\text{smallest}} = 2^{-126} \approx 10^{-38}$

Smallest and largest possible (absolute) value of float64

Recall $L \leq e \leq U$, $L = -1022$, $U = 1023$
Let $e = b - 1023$, where b is a non negative 8-bit integer (uint8)

Largest absolute value: $x^{(64)}_{\mathrm{largest}} \approx 2 \cdot 2^{1023}$

Smalest absolute value: $x^{(64)}_{\mathrm{smallest}} = 2^{-1022}$