Lecture Notes 06: Numerical Algorithms & Errors

CPSC 302: Numerical Computation for Algebraic Problems

Jessica Bosch
jbosch@cs.ubc.ca
http://www.cs.ubc.ca/~jbosch

University of British Columbia
Department of Computer Science

2017/2018 Winter Term 1

Outline

Outline

Goals

- Describe how numbers are stored in a floating point system.
- Understand how standard floating point systems are designed and implemented: IEEE standard.

Real Number Representation: Decimal & Binary

The decimal system is convenient for humans.

$$(0.0007396)_{10} = (7.396)_{10} \cdot 10^{-4} = \left(\frac{7}{10^0} + \frac{3}{10^1} + \frac{9}{10^2} + \frac{6}{10^3} \right) \times 10^{-4}$$

But computers prefer binary systems.

$$\begin{aligned}
(101.001)_2 = (1.01001)_2 \cdot 2^2 &= \left(\frac{1}{2^0} + \frac{0}{2^1} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{0}{2^4} + \frac{1}{2^5} \right) \times 2^2 \\
&= \left(\frac{1}{2^0} + \frac{1}{2^2} + \frac{1}{2^5} \right) \times 2^2 \\
&= \frac{2^2}{2^0} + \frac{2^2}{2^2} + \frac{2^2}{2^5} \\
&= 4 + 1 + 0.125 = (5.125)_{10}
\end{aligned}$$

Floating Point Number System $\mathbb{F}$

$$\mathbb{F} = \mathbb{F}(\beta, t, L, U)$$

$$x \in \mathbb{F}: \qquad x = \pm \left(\frac{\tilde{d}_0}{\beta^0} + \frac{\tilde{d}_1}{\beta^1} + \frac{\tilde{d}_2}{\beta^2} + \cdots + \frac{\tilde{d}_{t-1}}{\beta^{t-1}} \right) \times \beta^e$$

$$= \pm \left(\tilde{d}_0 . \tilde{d}_1 \tilde{d}_2 \cdots \tilde{d}_{t-1} \right)_\beta \times \beta^e$$

$$x = (\text{sign})(\text{"mantissa", i.e., significant digits})_\beta \times (\text{base}^{\text{exponent}})$$

- Base of the number system: $\beta \in \mathbb{N}$, $\beta > 1$
- Precision, i.e., number of digits with which a number is expressed: t
- Exponent range: $e \in \mathbb{Z}$ with $L \leq e \leq U$
- Digits: $\tilde{d}_i \in \mathbb{N}_0$, $0 \leq \tilde{d}_i \leq \beta - 1$

  Note that $\tilde{d}_0 > 0$: normalized floating point representation

Standard Floating Point Number System

Typical floating point number systems:

type	β	t	e
IEEE standard single precision (C float)	2	24	[–126, +127]
IEEE standard double precision (C double, MATLAB)	2	53	[–1022, +1023]

Lots of technical details: normalization ($\tilde{d}_0 \neq 0$), rounding, subnormals and gradual underflow, cancellation, special values (0, `Inf`, `NaN`), etc.

IEEE Standard Word

Sign, exponent, and mantissa stored in separate fields of a given floating point word.

Double precision (64 bit word)

$s = \pm$	$b =$11-bit exponent	$f =$52-bit "mantissa"
$s \in \{0, 1\}$	$b = e + U$ in binary representation	$f = \tilde{d}_1 \tilde{d}_2 \cdots \tilde{d}_{t-1}$

Single precision (32 bit word)

$s = \pm$	$b =$8-bit exponent	$f =$23-bit "mantissa"

Example

How do we store $(1.01101)_2 \times 2^4$ as IEEE single precision word?

s	b	f
0	10000011	01101000000000000000000

Rounding

How do we approximate an $x \in \mathbb{R}$ for which $x \notin \mathbb{F}$?

Example: Consider $\mathbb{F}(10, 4, -2, 2)$ and $x = \left(\frac{8}{3}\right)_{10}$. Obviously, $x \notin \mathbb{F}$.

$$\left(\frac{8}{3}\right)_{10} = 2.66\ldots6 \times 10^0 = \underbrace{\left(\frac{2}{10^0} + \frac{6}{10^1} + \frac{6}{10^2} + \frac{6}{10^3} + \frac{6}{10^4} + \ldots\right)}_{\text{in } \mathbb{F}\text{: only 4 significant digits}} \times 10^0$$

Rounding: $\text{fl}(x) \approx x$ where $\text{fl}(x) \in \mathbb{F}$

- *Chop:* truncate after the 3rd digit

$$x = \left(\frac{8}{3}\right)_{10} \approx \quad \text{fl}(x) = \left(\frac{2}{10^0} + \frac{6}{10^1} + \frac{6}{10^2} + \frac{6}{10^3}\right) \times 10^0 = 2.666 \times 10^0$$

- *Round to nearest:* default rule in IEEE standard systems

$$x = \left(\frac{8}{3}\right)_{10} \approx \quad \text{fl}(x) = \left(\frac{2}{10^0} + \frac{6}{10^1} + \frac{6}{10^2} + \frac{7}{10^3}\right) \times 10^0 = 2.667 \times 10^0$$

Examples

Number	Chop	Round to nearest
1.49	1.4	1.5
1.51	1.5	1.5
1.99	1.9	2.0
1.55	1.5	1.6
1.45	1.4	1.4

Machine Precision ϵ_{mach}

Accuracy of $\mathbb{F}$ is measured by the machine precision ϵ_{mach}.

- It is the maximal possible relative error in representing an $x \neq 0$ in $\mathbb{F}$:

$$\left| \frac{\mathsf{fl}(x) - x}{x} \right| \leq \epsilon_{mach}$$

- Its actual value depends on rounding rule:
  - Chop: $\epsilon_{mach} = \beta^{1-t}$
  - Round to nearest: $\epsilon_{mach} = \frac{1}{2}\beta^{1-t}$

- IEEE standard single precision: $\epsilon_{mach} = 2^{-24} \approx 10^{-7}$
- IEEE standard double precision: $\epsilon_{mach} = 2^{-53} \approx 10^{-16}$

  MATLAB: ϵ_{mach}=eps/2

Floating Point Arithmetic & Rounding

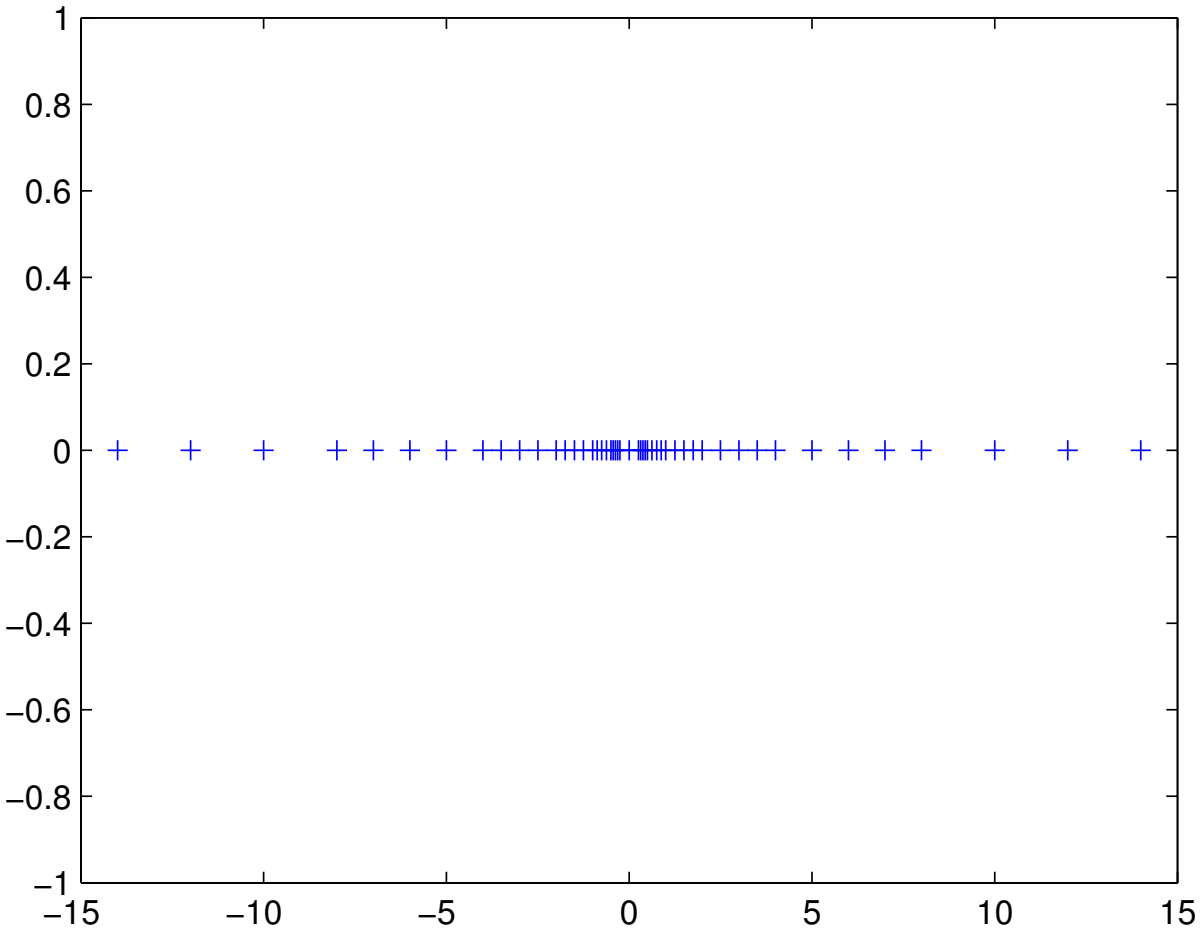
The result of an arithmetic operation on floating point numbers may not be a floating point number.

- Arithmetic operations are commutative (eg: $a + b = b + a$).
- Arithmetic operations are **not** associative (eg: $(a + b) + c \neq a + (b + c)$ for some a, b, c).
  - See Piazza: roundoff_error.ipynb
- For error analysis, usually assume (true for IEEE standard)

$$\frac{|\text{fl}(\text{fl}(x) \text{ op } \text{fl}(y)) - (\text{fl}(x) \text{ op } \text{fl}(y))|}{|(\text{fl}(x) \text{ op } \text{fl}(y))|} = |\delta| \leq \epsilon_{mach}$$

  where "op" is any standard arithmetic operation (eg: $+$, $-$, $\times$, :).

Spacing of Floating Point Numbers



Note the **uneven** distribution, both for large exponents and near 0.

Overflow, Underflow, NaN

- Overflow: when $e > U$ (fatal)
- Underflow: when $e < L$ (non-fatal: set to 0 by default)
- NaN: Not-a-number (e.g., $0/0$)

The Patriot Missile Failure

Poor handling of roundoff errors causes disaster!

1991, during the Gulf War, an American Patriot Missile battery in Dhahran, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile.

- Chopping of $1/10$ lead to an error of 0.34 seconds.
- Incoming Scud was outside the "range gate" that the Patriot tracked.
- It killed 28 Americans.