Scientific Computing Using Python

Floating-point representation and arithmetic – the basics

13th of June, 2023

Thomas Arildsen
tari@its.aau.dk

CLAAUDIA
Aalborg University

Floating-point representation and arithmetic – the basics
Agenda

- Learning objectives for this lecture:
  - ... know floating point representation.
  - ... understand the critical issue of cancellation in numerical computing.

Introduction

Floating point representation
- Machine precision

Numerical Issues
- Finite difference approximation

Conclusions

Floating point representation

Floating point representation
Intro

- Any real number $x \in \mathbb{R}$ needs a mapping to be representable in a digital computer.
- Need a finite word length, e.g. 8 bytes (double precision).
- The finite set of representable numbers $\mathbb{F}_t$:

$$\mathbb{F}_t \subset \mathbb{R}$$
$$|\mathbb{F}_t| < |\mathbb{R}| = \infty$$

- For double precision binary representation the largest number we can represent is approximately $1.80 \cdot 10^{308}$.
- $2.00 \cdot 10^{308}$ yield overflow, IEEE 754 [IEE08] format this as 'inf' for infinite.

Floating point representation
Representation

- One way of representing floating point numbers in the computer:

$$\check{x} = (-1)^s \cdot m \cdot b^e$$

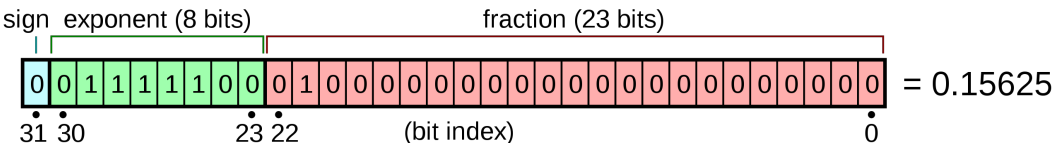


Figure: From WikiPedia by user Stannered (CC BY-SA 3.0)

s is the sign which is an integer belonging to the set $s \in \{0, 1\}$. $s = 0$ for positive numbers, and $s = 1$ for negative numbers.

Floating point representation
Representation

- One way of representing floating point numbers in the computer:

$$\check{x} = (-1)^s \cdot m \cdot b^e$$

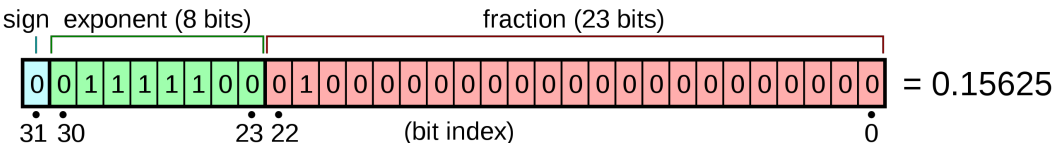


Figure: From WikiPedia by user Stannered (CC BY-SA 3.0)

b is the base or radix with $b \in \{2, 10\}$. For binary types $b = 2$ (*our context here*) and for decimal types $b = 10$.

Floating point representation
Representation

- One way of representing floating point numbers in the computer:

$$\check{x} = (-1)^s \cdot m \cdot b^e$$

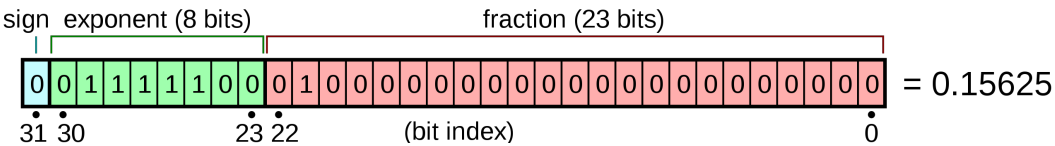


Figure: From WikiPedia by user Stannered (CC BY-SA 3.0)

m is a scalar number which can be represented by the string '$d_0.d_1 \cdots d_{p-1}$' as:

$$m = d_0 \cdot b^0 + d_1 \cdot b^{-1} + \cdots + d_{p-2} \cdot b^{-(p-2)} + d_{p-1} \cdot b^{-(p-1)}$$

where $d_0, d_1, \ldots, d_{p-1} \in \{0, 1, \ldots, b-1\}$. It should be noted that the binary encoding is such that $d_0 = 1$ for normal numbers. Only sub-normal numbers (and zero) can have $d_0 = 0$.

Floating point representation
Representation

- One way of representing floating point numbers in the computer:

$$\check{x} = (-1)^s \cdot m \cdot b^e$$

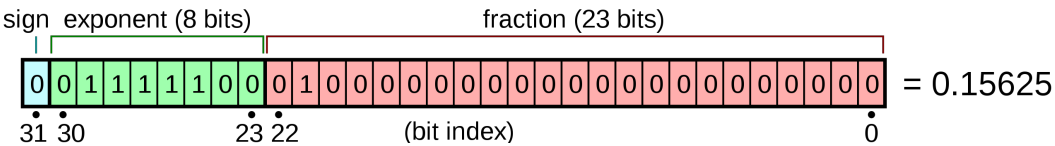


Figure: From WikiPedia by user Stannered (CC BY-SA 3.0)

e is an exponent integer given by $e \in \{e_{\min}, 2 - e_{\max}, \ldots, e_{\max}\}$, represented by w digits and $e_{\min} = 1 - e_{\max}$:

$$e = e_0 \cdot b^{w-1} + e_1 \cdot b^{w-2} + \cdots + e_{w-1} \cdot b^0 - e_{\max}$$

Floating point representation
Types

- Representing floating point numbers in the computer:

IEEE type	Base b	p [bits]	e_{max} [—]
Decimal-64	10	16	+384
Decimal-128	10	34	+6144
Binary-16	2	10 (+1)	+15
Binary-32	2	23 (+1)	+127
Binary-64	2	52 (+1)	+1023
Binary-128	2	112 (+1)	+16383

- Table: +1 in the binary formats is that the mantissa m has an implicit $d_0 = 1$, unless the written exponent is all zeros.
- Normally we use the binary format $\rightarrow$ hardware supported. Decimal for finance.

Floating point representation
Example: overflow I

- Consider the following function:

$$\begin{aligned} y &= \sum_{\ell=0}^{L} a_\ell \cdot x^\ell \\ &= a_0 + a_1 \cdot x + \cdots + a_{L-1} \cdot x^{L-1} + a_L \cdot x^L \\ &= a_0 + x \cdot (a_1 + x \cdot (a_2 + x \cdot (\cdots (a_{L-1} + a_L \cdot x) \cdots))) \end{aligned}$$

- where we have:

$$\text{sgn}(a_\ell) \quad = \quad \begin{cases} +1 & \text{for } \ell \text{ even} \\ -1 & \text{else} \end{cases}$$

- and $a_0 = 1200.0$, $a_1 = -150.0$, $a_2 = 11.0$, $a_3 = -101.0$, $a_4 = 12.0$, $x = 10.0$, and $L = 4$.
- $y = 1.98 \cdot 10^4$

Floating point representation
Example: overflow II

- Using a Binary-16 type from IEEE 754 with a maximum value of $6.55 \cdot 10^4$ and sequential use of:

$$y = a_0 + a_1 \cdot x + \cdots + a_{L-1} \cdot x^{L-1} + a_L \cdot x^L$$

$$\begin{aligned}
a_0 + a_1 \cdot x &= 1200.0 - 150.0 \cdot 10^1 &\rightarrow\ & x_1 = -3.00 \cdot 10^2 \\
x_1 + a_2 \cdot x^2 &= x_1 + 11.0 \cdot 10^2 &\rightarrow\ & x_2 = 8.00 \cdot 10^2 \\
x_2 + a_3 \cdot x^3 &= x_2 - 101.0 \cdot 10^3 &\rightarrow\ & x_3 = -\text{inf} \\
x_3 + a_4 \cdot x^4 &= x_3 + \text{inf} &\rightarrow\ & y = \text{NaN}
\end{aligned}$$

- Intermediate values also has a limit!

Floating point representation
Example: overflow III

- Alternative formula (Horners method):

$$y = a_0 + x \cdot (a_1 + x \cdot (a_2 + x \cdot (\cdots (a_{L-1} + a_L \cdot x) \cdots)))$$

- Alternative method of computing:

$$\begin{aligned}
a_3 + a_4 \cdot x &= -101.0 + 12.0 \cdot 10.0 \quad \rightarrow \quad x_3 = 1.90 \cdot 10^1 \\
a_2 + x_3 \cdot x &= 11.0 + x_3 \cdot 10.0 \quad \rightarrow \quad x_2 = 2.01 \cdot 10^2 \\
a_1 + x_2 \cdot x &= -150.0 + x_2 \cdot 10.0 \quad \rightarrow \quad x_1 = 1.86 \cdot 10^3 \\
a_0 + x_1 \cdot x &= 1200.0 + x_1 \cdot 10.0 \quad \rightarrow \quad y = 1.98 \cdot 10^4
\end{aligned}$$

- The same mathematical equation may render different results when implemented as a computer program using floating-point representation and arithmetic.

Floating point representation

ϵ: machine precision I

- Floating point "quantization"(representation) can be described as:

$$u \mapsto \breve{u} = (-1)^s \cdot m \cdot 2^e : \quad u - \breve{u} \in \left[-2^{e-p};\, 2^{e-p}\right)$$

- Define a relative error between a real number u and the finite normal representation $\breve{u}$ as:

$$E = \left| \frac{u - \breve{u}}{u} \right|$$

- The largest relative error for $b = 2$ is then:

$$\epsilon_2 = \max E \simeq 2^{-p}$$

Floating point representation

ϵ: machine precision II

- The largest relative error for $b = 2$ is then:

$$\epsilon_2 = \max E \simeq 2^{-p}$$

- Confirms our intuition. Larger p gives higher accuracy.
- Sometimes called machine precision.
- An alternative definition is:

$$\epsilon' = 2 \max_{\check{1}+\gamma=\check{1}} \gamma = 2 \cdot \epsilon_2$$

- This is used in Python.
- Use e.g. `numpy.finfo(numpy.float64).eps/2.0` to get ϵ_2.

Miniexercise

Describe the difference between computing the sum:

```
>>> x = [1.0] + [1e-16]*10000
>>> sum(x)
```

or (continuing)

```
>>> x.reverse()
>>> sum(x)
```

Why?

Floating point representation

ϵ: machine precision III

- Interpreted as the number of significant decimal digits.

Format	Name	Machine ϵ_2	Sign. dec. digits
B16	half precision	$4.88 \cdot 10^{-4}$	$\simeq$ 3–5
B32	single precision	$5.96 \cdot 10^{-8}$	$\simeq$ 6–9
B64	double precision	$1.11 \cdot 10^{-16}$	$\simeq$ 15–17

Numerical Issues

Numerical issues
Numerical aspects

- When we do math we normally have a variable $x \in \mathbb{R}$. When we do scientific computing we are limited to having $x \in \mathbb{F}_t$ where $\mathbb{F}_t \subset \mathbb{R}$.
- Does it matter . . .? the answer is not black/white. Often it does not matter:
  - . . . rounding errors meaning that the true mathematically correct number $x = \hat{x} + \epsilon$ where $\hat{x} \in \mathbb{F}_t$ and ϵ is a hopefully small difference.
- – but sometimes it does:
  - . . . cancellation effects where two variables very close to each other in value are subtracted. The relative error may be relatively large which may lead to trouble.

Numerical issues
Example: finite difference approximation

- The example considers the function:

$$y(x) = 1 - x^2 + 3 \cdot x^3 - 5 \cdot x^4 + 4 \cdot x^5$$

- We compare the theoretical derivative:

$$\dot{y}(x_0) = -2 \cdot x_0 + 9 \cdot x_0^2 - 20 \cdot x_0^3 + 20 \cdot x_0^4$$

- with a finite difference approximation:

$$\dot{y}_{\text{approx.}}(x_0; x_\delta) = \frac{y(x_0 + x_\delta) - y(x_0 - x_\delta)}{2 \cdot x_\delta}$$

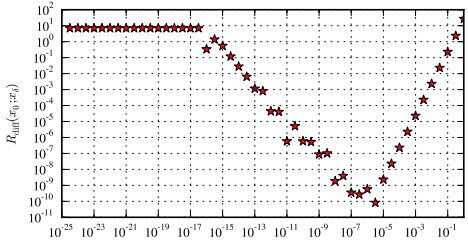
- We then observe the absolute value of the relative difference:

$$R_{\text{diff}}(x_0;\ x_\delta) = \left| \frac{\dot{y}_{\text{approx.}}(x_0;\ x_\delta) - \dot{y}(x_0)}{\dot{y}(x_0)} \right|$$

Numerical issues

Example: finite difference approximation

- Result for double precision binary:



- Rule of thumb for optimum value is $x_\delta \simeq x_0 \cdot \sqrt{\epsilon_2} \simeq 1.1 \cdot 10^{-8}$ where $\epsilon_2 \simeq 1.11 \cdot 10^{-16}$ is the machine ϵ_2 for double precision.
- From the figure, the optimum is around $x_\delta \simeq 10^{-6}$.

Conclusions

Conclusions

- Conclusions:
  - ... floating point representation matters ... sometimes.
  - ... numerical issues can be hard to identify. Can easily be interpreted as other errors.
  - ... mathematical equivalent algorithms are not equivalent when implemented using floating point representation/arithmetic.

Sources

[IEE08] IEEE. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standards 754–2008. IEEE Computer Society. Aug. 2008.