Numerical Analysis
GC / SIE
Floating point number representation and round-off errors

Daniel Kressner
Chair for Numerical Algorithms and HPC
Institute of Mathematics, EPFL
daniel.kressner@epfl.ch

EPFL

Slides by Pablo Antolín and Fabio Nobile.

Real number presentation in a computer

- Real numbers are stored in the following way (floating point representation) in a computer:

$$(-1)^s \cdot (0.a_1 a_2 \ldots a_t) \cdot \beta^e = (-1)^s \cdot m \cdot \beta^{e-t}; \quad a_1 \neq 0$$

  - sign: $s = 0$ ou 1
  - mantissa: $m = a_1 \cdots a_t$ (t integer digits $0 \leq a_i \leq \beta - 1$, with $a_1 \neq 0$)
  - base: β (usually 2)
  - exponent: e

Example

$$-23 \qquad \Longrightarrow \qquad (-1)^1 * (0.10111000\cdots) * 2^5$$

Single and double precision

- The floating point representation uses a fixed number of digits (bits) for the mantissa and exponent, plus an additional digit for the sign.

	t	e	bytes	bits
Single precision	24	$-125 \leq e \leq 128$	4	32
Double precision	53	$-1021 \leq e \leq 1024$	8	64

(1 byte = 8 bits)

- Common practice in scientific/engineering applications is to use **double precision**:
  - In base 10 this roughly correponds to 16 decimal digits and 3 digits for the exponent
  - The largest representable number: $1.797693134862316 \cdot 10^{+308}$
  - The smallest representable number: $2.225073858507201 \cdot 10^{-308}$
- In machine learning, GPUs, etc. uses single or even half precision.

Single and double precision

- The floating point representation uses a fixed number of digits (bits) for the mantissa and exponent, plus an additional digit for the sign.

	t	e	bytes	bits
Single precision	24	$-125 \leq e \leq 128$	4	32
Double precision	53	$-1021 \leq e \leq 1024$	8	64

(1 byte = 8 bits)

- Common practice in scientific/engineering applications is to use **double precision**:
  - In base 10 this roughly correponds to 16 decimal digits and 3 digits for the exponent
  - The largest representable number: $1.797693134862316 \cdot 10^{+308}$
  - The smallest representable number: $2.225073858507201 \cdot 10^{-308}$
- In machine learning, GPUs, etc. uses single or even half precision.

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 &= 0.1000000000000000 \quad 10^{1} \\ x = 10^{-16} & \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$1 \quad = 0.1000000000000000 \quad 10^1$

$x = 10^{-16}$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 \quad &= 0.1000000000000000 \quad 10^{1} \\ x = 10^{-16} \quad &= 0.1000000000000000 \quad 10^{-15} \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0100000000000000 & 10^{-14} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0010000000000000 & 10^{-13} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0001000000000000 & 10^{-12} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.9999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 &= 0.1000000000000000 \quad 10^{1} \\ x = 10^{-16} &= 0.0000100000000000 \quad 10^{-11} \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0000010000000000 & 10^{-10} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.9999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 \quad & = 0.1000000000000000 \quad 10^{1} \\ x = 10^{-16} \quad & = 0.0000001000000000 \quad 10^{-9} \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0000000100000000 & 10^{-8} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & =0.1000000000000000 & 10^{1} \\ x = 10^{-16} & =0.0000000010000000 & 10^{-7} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & =0.1000000000000000 & 10^{1} \\ x = 10^{-16} & =0.0000000001000000 & 10^{-6} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 \quad &= 0.1000000000000000 \quad 10^{1} \\ x = 10^{-16} \quad &= 0.0000000000100000 \quad 10^{-5} \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 \quad &= 0.1000000000000000 \quad 10^{1} \\ x = 10^{-16} \quad &= 0.0000000000010000 \quad 10^{-4} \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \dots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0000000000001000 & 10^{-3} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & =0.1000000000000000 & 10^{1} \\ x = 10^{-16} & =0.0000000000000100 & 10^{-2} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{array}{lll} 1 & = 0.1000000000000000 & 10^{1} \\ x = 10^{-16} & = 0.0000000000000010 & 10^{-1} \end{array}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} 1 \quad &= 0.1000000000000000 \quad 10^1 \\ x = 10^{-16} \quad &= 0.0000000000000001 \quad 10^0 \end{aligned}$$

Examples and dangers

Example 1: Try to evaluate in Python the function

$$f(x) = \frac{(1+x)-1}{x}$$

for $x = 10^{-1}, 10^{-2}, \ldots, 10^{-16}$.

In Python we get:

- $f(10^{-1}) = 1.000000000000001$
- $f(10^{-6}) = 0.999999999917733$
- $f(10^{-16}) = 0$

This phenomenon is known as **loss of significance**.

Explanation: when the computer does the addition $1 + 10^{-16}$ firstly it must "align the exponents" of both numbers

$$\begin{aligned} &1 &&= 0.1000000000000000 \quad 10^{1} \\ &x = 10^{-16} &&= 0.0000000000000000 \quad 10^{1} \end{aligned}$$

$$\Longrightarrow 1 + 10^{-16} = 1\,!!!$$

Examples and dangers

Example 2: Ariane 5, Guyane, June 1996

Photo Credit: Jeremy Beck / SpaceFlight Insider

Examples and dangers

Example 2: Ariane 5, Guyane, June 1996 . . . 37 seconds after the launch

Photo Credit: Pool VENTURIER/LE CORRE

Examples and dangers

Example 2: Ariane 5, Guyane, June 1996 → an overflow error

From SIAM News, Vol. 29. Number 8, October 1996:

> *The internal SRI software exception was caused during execution of a data conversion from a 64-bit floating-point number to a 16-bit signed integer value. The value of the floating-point number was greater than what could be represented by a 16-bit signed integer. The result was an operand error. The data conversion instructions (in Ada code) were not protected from causing operand errors, although other conversions of comparable variables in the same place in the code were protected.*

An OVERFLOW error with no human losses, but a cost of 500 million dollars and 10 years of work!!

Examples and dangers

Example 2: Ariane 5, Guyane, June 1996 → an overflow error

From SIAM News, Vol. 29. Number 8, October 1996:

> *The internal SRI software exception was caused during execution of a data conversion from a 64-bit floating-point number to a 16-bit signed integer value. The value of the floating-point number was greater than what could be represented by a 16-bit signed integer. The result was an operand error. The data conversion instructions (in Ada code) were not protected from causing operand errors, although other conversions of comparable variables in the same place in the code were protected.*

An OVERFLOW error with no human losses, but a cost of 500 million dollars and 10 years of work!!

Examples and dangers

Example 3: Let's consider

$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \\ \frac{1}{2} & \frac{1}{3} & \cdots & \cdots & \\ \frac{1}{3} & \vdots & \ddots & & \\ \vdots & \vdots & & & \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad x = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{n} \quad \text{et} \quad b = Ax.$$

We resolved the linear system $Ax = b$ for $n = 5, 10, 15, 20, \ldots$, i.e., $x = A^{-1}b$ is computed

What happens?