MAT 460 Numerical Differential Equations

Version 21.10.14

Michael Karow, Andrea Dziubek

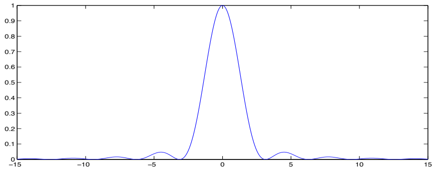
Lecture 1

Number Representation in Computers

Three examples of wrong computations with MATLAB.

Example 1: Consider the function

$$f(x) = \frac{1-\cos(2x)}{2\,x^2}$$



The plot shows that $f(10^{-9}) \approx 1$, but MATLAB computes $f(10^{-9}) = 0$.

We rewrite the problem:

$$f(x) = \frac{1-\cos(2x)}{2\,x^2} = \frac{(\sin(x)^2+\cos(x)^2)-(\cos(x)^2-\sin(x)^2)}{2\,x^2} = \left(\frac{\sin(x)}{x}\right)^2. \qquad (*)$$

Computing the right expression in $(*)$ with MATLAB gives $f(10^{-9}) = 1$.

Example 2:

For every real number $k \in \mathbb{R}$

$$(345 + 10^k) - 10^k = 345.$$

MATLAB computes

$$\begin{aligned}(345+10^{15})-10^{15} &= 345\\ (345+10^{16})-10^{16} &= 344\\ (345+10^{17})-10^{17} &= 352\\ (345+10^{18})-10^{18} &= 384\\ (345+10^{19})-10^{19} &= 0\end{aligned}$$

Explanation:

For large k the number $345 + 10^k$ cannot be saved exactly in a computer (mantissa size is too large).

Example 3:

MATLAB computes $1234 * (0.1 + 0.1 + 0.1 - 0.3)^{1/10} = 29.22\ldots$.

But the exact value is 0.

Explanation:

The numbers 0.1 and 0.3 can not be represented exactly in the computer.

MATLAB computes instead

$$0.1 + 0.1 + 0.1 - 0.3 = 5.55 * 10^{-17}$$

We see a large increase in this small error if we take the 10^{th} square root.

Questions:

1. How does the computer save numbers? How does it perform computations?
2. What makes errors worse?

Floating point numbers

An decimal number example:

$$\begin{aligned} 273.534 &= 2*10^2+7*10^1+3*10^0+5*10^{-1}+3*10^{-2}+4*10^{-3} \\ 1/3 &= 0.33333\ldots = 0.\overline{3} = 3*10^{-1}+3*10^{-2}+3*10^{-3}+\ldots \\ 1 &= 0.9999\ldots = 0.\overline{9} \end{aligned}$$

Theorem: Let $b \geq 2$ be an integer number and $x \in \mathbb{R}$, $x \geq 0$. Then a $k \in \mathbb{Z}$ and an infinite sequence $z_1, z_2, z_3 \ldots \in \{0,1,2,\ldots,b-1\}$ with $z_1 \neq 0$ exist, such that

$$\begin{aligned} x &= z_1 * b^{k-1} + z_2 * b^{k-2} + z_3 * b^{k-3} + z_4 * b^{k-4} \ldots \\ &= (z_1 * b^{-1} + z_2 * b^{-2} + z_3 * b^{-3} + z_4 * b^{-4} \ldots) * b^k \end{aligned}$$

The sequence z_j is unique, if we exclude the case $z_j = b-1$ for all $j \geq j_0$.

Notation:

$$\begin{aligned} x &= (0.z_1\, z_2\, z_3 \ldots)_b * b^k && (*) \\ &= (z_1\, z_2\, \ldots\, z_k\, .\, z_{k-1}\, z_{k-2} \ldots)_b && \text{for } k > 0 \end{aligned}$$

Definitions:

b = basis, k = exponent, symbols for the digits = z_j, sequence of digits = mantissa.

$(*)$ is called normalized floating point representation, $z_1 \neq 0$.

The following bases are often used in computations with computers:

$b = 2, 8, 16$ (dual-, octal- and hexa-decimal system)

Convert an decimal number < 1 into another number system

Example: dual system (synonym: binary system). digits: 0,1

Problem: Represent the decimal number 0.7 as dual number.

Write: $(0.7)_{10} = (0.z_1\, z_2\, z_3\, \ldots)_2$ [multiply both sides by 2 and compare the expressions before and after the dot]

$$\begin{aligned} \Rightarrow\ (1.4)_{10} &= 2*(0.7)_{10} = (z_1.z_2\, z_3 \ldots)_2 \Rightarrow z_1 = 1 \text{ and } (0.4)_{10} = (0.z_2\, z_3\, \ldots)_2 \\ \Rightarrow\ (0.8)_{10} &= 2*(0.4)_{10} = (z_2.z_3\, z_4 \ldots)_2 \Rightarrow z_2 = 0 \text{ and } (0.8)_{10} = (0.z_3\, z_4\, \ldots)_2 \quad (\\ \Rightarrow\ (1.6)_{10} &= 2*(0.8)_{10} = (z_3.z_4\, z_5 \ldots)_2 \Rightarrow z_3 = 1 \text{ and } (0.6)_{10} = (0.z_4\, z_5\, \ldots)_2 \\ \Rightarrow\ (1.2)_{10} &= 2*(0.6)_{10} = (z_4.z_5\, z_6 \ldots)_2 \Rightarrow z_4 = 1 \text{ and } (0.2)_{10} = (0.z_5\, z_6\, \ldots)_2 \\ \Rightarrow\ (0.4)_{10} &= 2*(0.2)_{10} = (z_5.z_6\, z_7 \ldots)_2 \Rightarrow z_5 = 0 \text{ and } (0.4)_{10} = (0.z_6\, z_7\, \ldots)_2 \\ \Rightarrow\ (0.8)_{10} &= 2*(0.4)_{10} = (z_6.z_7\, z_8 \ldots)_2 \Rightarrow z_6 = 0 \text{ and } (0.8)_{10} = (0.z_7\, z_8\, \ldots)_2 \quad (\\ &\vdots \end{aligned}$$

Row (1) and (2) are identical ⇒ periodic digit sequence.

Result: $7/10 = (0.7)_{10} = (\,0.1\overline{0110}\,)_2$

Digits of the hexadecimal system: $0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f$

Further examples:

$$\begin{aligned} 1/4 &= (0.25)_{10} = (0.01)_2 \\ 3/8 &= (0.375)_{10} = (0.011)_2 \\ 10/16 &= (0.625)_{10} = (0.a)_{16} \\ 1/7 &= (0.\overline{142857})_{10} = (0.1)_7 \end{aligned}$$

Convert an integer decimal number into another number system

Example: dual system (synonym: binary system). digits: 0,1

Problem: Represent the decimal number 27 as dual number.

Write: $(27)_{10} = (\ldots z_2\, z_1\, z_0)_2$

$$\begin{aligned} (27)_{10} = 2*(13)_{10}+1 &= 2*(\ldots z_2\, z_1)_2 + z_0 \Rightarrow (13)_{10} = (\ldots z_2\, z_1)_2 \text{ and } z_0 = 1 \\ (13)_{10} = 2*(6)_{10}+1 &= 2*(\ldots z_3\, z_2)_2 + z_1 \Rightarrow (6)_{10} = (\ldots z_3\, z_2)_2 \text{ and } z_1 = 1 \\ (6)_{10} = 2*(3)_{10}+0 &= 2*(\ldots z_4\, z_3)_2 + z_2 \Rightarrow (3)_{10} = (\ldots z_4\, z_3)_2 \text{ and } z_2 = 0 \\ (3)_{10} = 2*(2)_{10}+1 &= 2*(\ldots z_5\, z_4)_2 + z_3 \Rightarrow (2)_{10} = (\ldots z_5\, z_4)_2 \text{ and } z_3 = 1 \\ (2)_{10} = 2*(1)_{10}+0 &= 2*(\ldots z_6\, z_5)_2 + z_4 \Rightarrow z_j = 0 \text{ für } j \geq 6,\ z_5 = 1 \text{ and } z_4 = \end{aligned}$$

Result: $(27)_{10} = (11011)_2$

Check: $(11011)_2 = 1*1+1*2+0*4+1*8+1*16$

Machine numbers (general)

Definitions: For given basis b, mantissa size ℓ and exponent limits $k_{\min} < 0 < k_{\max}$ we call

$$\mathbb{M}(b,\ell,k_{\min},k_{\max}) := \{\ \sigma * (0\,.\,z_1\,z_2\,\dots z_\ell)_b * b^k \ \mid$$

$$\sigma \in \{+,-\},\ \ z_j \in \{0,1,\dots,b-1\},\ \ z_1 \neq 0,\ \ k_{\min} \le k \le k_{\max}\ \} \cup \{0\}$$

the set of machine numbers in normalized floating point representation.

If we add numbers with $k = k_{\min}$ and $z_1 = 0$, we call the set the extended set of machine numbers $\widehat{\mathbb{M}}(b,\ell,k_{\min},k_{\max})$

This are the numbers that can be reprepresented exactly on a computer.

All other numbers $x \in \mathbb{R}$ are rounded (truncated) to $\pm\infty$ or to 0:

$$x \longmapsto x_M \in \widehat{\mathbb{M}}(b,\ell,k_{\min},k_{\max}) \cup \{\pm\infty\}$$

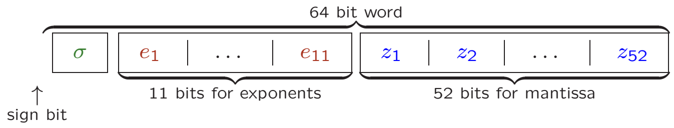
Theorem: Let $x_{\max}$ and $x_{\min}$ the largest and the smallest positive number in $\mathbb{M}(b,\ell,k_{\min},k_{\max})$
The smallest relative rounding error is then:

$$\frac{|x - x_M|}{|x|} \le \frac{1}{2}\ \underbrace{b^{-\ell+1}}_{=:\texttt{eps}} \qquad \text{for } x_{\min} < |x| < x_{\max}.$$

Definition: `eps` is called machine precision.

Alternative definition: 1+`eps` is the smallest machine number that is larger than 1.

MATLAB follows IEEE-Standard 754 (1985) for double precision format



This memory is used to represent the number

$$x = (-1)^\sigma * (1.\,z_1\,z_2\,\dots z_{52})_2 * 2^{(e_1\,e_2\,\dots e_{11})_2 - (1023)_{10}}$$

for

$$(0,0,\dots 0) \neq (e_1\,e_2\,\dots e_{11}) \neq (1,1,\dots,1)$$

If $(e_1\,e_2\,\dots e_{11}) = (0,0,\dots 0)$, then

$$x = (-1)^\sigma * (0.z_1 z_2\,z_3\,\dots z_{52})_2 * 2^{-(1022)_{10}}.$$

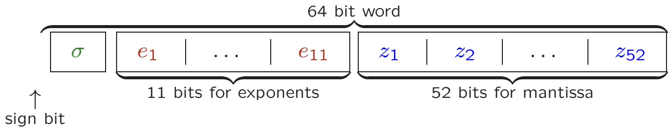
If $(e_1\,e_2\,\dots e_{11}) = (1,1,\dots,1)$ and $z_1 = z_2 = \dots = z_{52} = 0$, then $x = \pm\texttt{INF}$.
If $(e_1\,e_2\,\dots e_{11}) = (1,1,\dots,1)$ and $z_1 = 1,\ z_2 = \dots = z_{52} = 0$, then $x = \texttt{NAN}$.

(`INF`='infinity' denotes numbers (including ∞) which are greater than the largest real number (`realmax`). Example: $1/0 = \texttt{INF}$. `NAN` means 'not a number'.
`NAN` is the result of computations that doesn't make sense. Example: 0/0=`NAN`)

Note: IEEE = Institute of Electrical and Electronics Engineers

The smallest positive 'double precision'-number

Memory content of x:



If $(e_1\,e_2\,\dots e_{11}) = (0,0,\dots 0)$, then

$$x = (-1)^\sigma * (0.z_1 z_2\,z_3\,\dots z_{52})_2 * 2^{-(1022)_{10}}.$$

The smallest positive number we get for

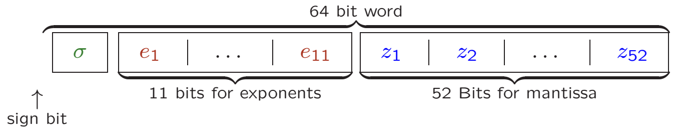
$$z_1 = z_2 = \dots = z_{51} = 0, \qquad z_{52} = 1.$$

IAlso:

$$x = 2^{-(1074)_{10}} \approx 4.94 * 10^{-324}.$$

The MATLAB number `realmin`

Memory content of x:



This memory is used to represent the number

$$x = (-1)^\sigma * (1.\,z_1\,z_2\,\dots z_{52})_2 * 2^{(e_1\,e_2\,\dots e_{11})_2 - (1023)_{10}}$$

for

$$(0,0,\dots 0) \neq (e_1\,e_2\,\dots e_{11}) \neq (1,1,\dots,1)$$

The smallest positive number we get for

$$e_1 = e_2 = \dots = e_{10} = 0, \qquad e_{11} = 1 \qquad z_1 = z_2 = \dots = z_{52} = 0.$$

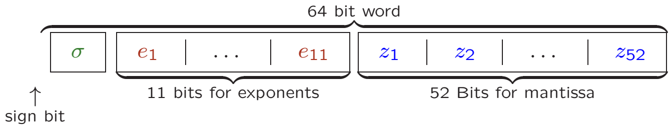
Also:

$$x = 2^{-(1022)_{10}} =: \texttt{realmin} \approx 2.22 * 10^{-308}.$$

Machine precision eps

Definition: eps is the difference between the number 1 and smallest machine number which is larger than 1.

Memory content of x:


64 bit word

σ | e_1 | ... | e_{11} | z_1 | z_2 | ... | z_{52}

11 bits for exponents

52 Bits for mantissa

sign bit

This memory is used to represent the number

$$x = (-1)^{\sigma} * (1.z_1\, z_2 \ldots z_{52})_2 * 2^{(e_1\, e_2 \ldots e_{11})_2 - (1023)_{10}}$$

for

$$(0,0,\ldots 0) \neq (e_1\, e_2 \ldots e_{11}) \neq (1,1,\ldots,1)$$

The smallest machine number x which is larger than 1 we get for

$$e_1 = 0, \qquad e_2 = e_3 = \ldots = e_{11} = 1, \qquad z_1 = z_2 = \ldots = z_{51} = 0, \qquad z_{52} = 1.$$

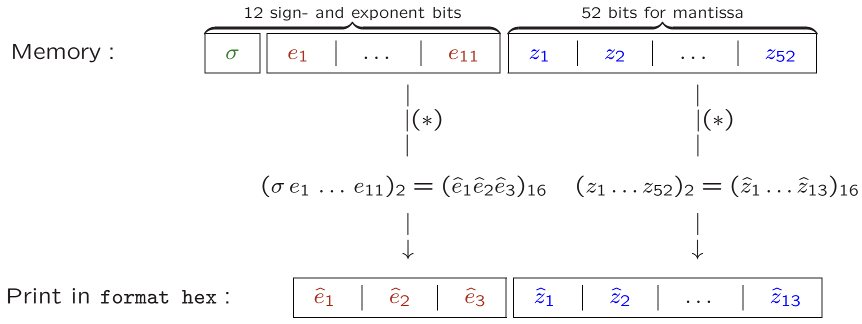
Also:

$$x = 1 + 2^{-52} \qquad \Rightarrow \qquad \texttt{eps} = 2^{-52} \approx 2.22 * 10^{-16}.$$

Conclusion: We can save a (not too small and not too large) real number in 'double precision'-format with exact 15 decimal places.

Print memory in MATLAB with format hex

To print the memory of a variable x, first type format hex and then type x. This will print a 16-digit long hexadecimal number, as shown below. (Note: After typing the command format or format long all numbers will be printed again as decimal numbers.)


12 sign- and exponent bits

52 bits for mantissa

Memory :

σ | e_1 | ... | e_{11} | z_1 | z_2 | ... | z_{52}

$(*)$

$(*)$

$(\sigma\, e_1 \ldots e_{11})_2 = (\hat{e}_1 \hat{e}_2 \hat{e}_3)_{16}$

$(z_1 \ldots z_{52})_2 = (\hat{z}_1 \ldots \hat{z}_{13})_{16}$

Print in format hex :

$\hat{e}_1$ | $\hat{e}_2$ | $\hat{e}_3$ | $\hat{z}_1$ | $\hat{z}_2$ | ... | $\hat{z}_{13}$

$(*)$ Interpret bits as digits of a binary number and write as hexadecimal number. This means, represent each 4 successive bits as digits $\hat{e}_k, \hat{z}_k \in \{0,1,\ldots,9,a,b,c,d,e,f\}$.

Example for format hex

Question: A number x is represented in format hex as sequence of digits:

$$x_{\text{hex}} = \text{c04a8000000000}$$

Which number is it?

Answer:

Translate the first three digits:

$$\begin{aligned} \text{c04} &= 12 * 16^2 + 0 * 16 + 4 \\ &= (1100)_2 * 16^2 + (0000)_2 * 16 + (0100)_2. \end{aligned}$$

$\Rightarrow \sigma\, e_1\, e_2 \ldots e_{11} = 110000000100$

$\Rightarrow \sigma = 1, \quad (e_1 \ldots e_{11})_2 = (10000000100)_2 = 2^{10} + 2^2 = (1028)_{10}.$

Translate the remaining digits:

$a = (1010)_2,\ 8 = (1000)_2,\ 0 = (0000)_2 \Rightarrow$

$$a8000000000 = 10101000\underbrace{0\ldots\ldots0}_{44\text{Zeros}}$$

Putting both together gives

$$\begin{aligned} x &= -1 * (1.10101000\,0\ldots\ldots0)_2 * 2^{(1028)_{10} - (1023)_{10}} \\ &= -1 * (1 + 2^{-1} + 2^{-3} + 2^{-5}) * 2^5 \\ &= (-53)_{10} \end{aligned}$$

Addition of binary numbers

The addition of binary numbers follows the same rules as for addition of decimal numbers which we learned in primary school:
Start at the right side. Add digits per column. If the result is larger than a digit, carry over and add to the next digit.

This example shows how to add x=54 and y=39 as binary numbers.

$$\begin{array}{cccccccl}
 & 1 & 1 & 0 & 1 & 1 & 0 & \leftarrow x \\
 & 1 & 0 & 0 & 1 & 1 & 1 & \leftarrow y \\
1 & 0 & 0 & 1 & 1 & 0 & & \text{carry over} \\ \hline
1 & 0 & 1 & 1 & 1 & 0 & 1 & \leftarrow x + y
\end{array}$$

The computer comonly makes an error when computing the four basic arithmetic operations.

Example: Perform the following computation in the decimal system with mantissa length 4.

Problem for the computer: add 1 and $0.5431 * 10^{-2} = 0.00543$.

Compute: $(0.1000 + 0.0005431) * 10^1 = 0.1005 * 10^1$

We lost the last 3 digits
(mantissa has finite length).

Subtraction of two numbers of same order (catastrophic cancellation error)

Example:

	Exact values:	Values in the computer:
	$x_1 = 0.10024$	$\widetilde{x}_1 = 0.1002$
	$x_2 = 0.10011$	$\widetilde{x}_1 = 0.1001$
Difference:	$x_1 - x_2 = 0.00013$	$\widetilde{x}_1 - \widetilde{x}_2 = 0.0001$
	$= 0.13 * 10^{-3}$	$= 0.1 * 10^{-3}$

The difference $\widetilde{x}_1 - \widetilde{x}_2$ is computed exact (in this example).
However, we see a large increase in the relative error. We have

$$\left|\frac{\widetilde{x}_1 - x_1}{x_1}\right| \approx 4 * 10^{-4}, \qquad \left|\frac{\widetilde{x}_2 - x_2}{x_2}\right| \approx 1 * 10^{-4},$$

however

$$\left|\frac{(\widetilde{x}_1 - \widetilde{x}_2) - (x_1 - x_2)}{x_1 - x_2}\right| \approx 2 * 10^{-1}$$

We lost 3 digits precision.
(The first 4 digits of x_i and $\widetilde{x}_i$ are equal, but $x_1 - x_2$ and $\widetilde{x}_1 - \widetilde{x}_2$ are equal only up to the first nonzero digit.)

This phenomena is called **catastrophic cancellation error**.

⇒ Avoid subtraction of two numbers of same order.