Floating-point numbers

Ágnes Baran, Csaba Noszály

Octave/Matlab basics

On-line documentation:

- Octave at www.gnu.org
- Matlab at www.mathworks.com

You can write expressions,statements in the command window:

```
>> 1+1

ans =

     2

>> 2*2

ans =

     4
```

The result will get into the variable named `ans`, unless we used assignment.

We can define our own variables:

```
>> a=2*3

a =

    6

>> b=3; c=a+b;
```

As you see, writing a semicolon after the expression, will turn off echoing the result - although the evaluation will be done. The value of a variable can be accessed by typing its name:

```
>> c

c =

    9
```

Variable names

- For the name of a variable one can use a sequence of characters that begins with a letter (of the english alphabet), and consists only of letters, digits and underscores. It is case sensitive!
- It is forbidden (and impossible) to use for naming variables the so called keywords: `if, for, while, function, ...`. For a full list keywords type `iskeyword`.
- It is strongly discouraged (but possible) to use the names of the so called built-in's: `size,sin,cos,exp,...`
- You can query the system about the existence of a particular name: `exist cos`
- You can destroy a variable with: `clear yourVariableName` You can destroy all variables in the workspace with: `clear all`
- for further details, see Variable names.

Relational operators

The result of a comparison is a logical `1` (=true) or logical `0` (=false).

- `a<b` is true iff. *a* is less than *b*
- `a<=b` is true iff. *a* is less than or equal to *b*
- `a>b` is true iff. *a* is greater than *b*
- `a>=b` is true iff. *a* is greater than or equal to *b*
- `a==b` is true iff. *a* is equal to *b*
- `a~=b` is true iff. *a* is not equal to *b*

For matrices of the same size the comparison is performed elementwise, i.e. comparing elements in the same location. The result is a logical matrix of the same shape.

m-files

`script`: A series of commands that you write into a file. It can be executed as a complete unit.

- Open in the editor window a new script and write our program here.

  Comments: Everything is ignored (not parsed,executed) after the `%` sign.
- Note that each of the statements are executed as it were typed in the command window, so without using `;` the results are printed on the screen.
- Save the file.
- Execute the script: either by pressing the `run` button in the top of editor window, or switching back to the command window by typing the `name` of script (without the .m extension)

The for-loop

```
for variable = vector
  statements
end
```

```
s=0;
for k=1:100
  s=s+k;
end
```

```
s=0;
for k=[ 1 3 -2 5 ]
  s=s+1/k;
end
```

```
s=0;
for k=100:-3:1
  s=s+k^2
end
```

The while-loop

```
while logical-expression
  statements
end
```

```
s=0; k=1;
while k<=100
  s=s+k; k=k+1;
end
```

```
s=1; k=10;
while k>1
  s=s*k; k=k-1;
end
```

```
s=0; k=100;
while k>=1
  s=s+k^2; k=k-3;
end
```

How can we trust in machine computations?

Exercise 1

Examine the value of the (logical) expression: $0.4 - 0.5 + 0.1 == 0$.
What is the value of $0.1 - 0.5 + 0.4 == 0$?

Exercise 2

What is the theoretical (expected) value of x after performing the following algorithm:

```
x=1/3;
for i=1:40
   x=4*x-1;
end
```

Exercise 3

Examine values of the following expressions:

$$2^{66} + 1 == 2^{66},\ 2^{66} + 100 == 2^{66},\ 2^{66} + 10000 == 2^{66}$$

Exercise 4

What are the results of algorithms below?

```
a=0;
for i=1:5
   a=a+0.2;
end
a==1
```

```
a=1;
for i=1:5
   a=a-0.2;
end
a==0
```

Try to explain!

Floating point numbers

Example

$a = 10$

$$0.3721 = \frac{3}{10} + \frac{7}{10^2} + \frac{2}{10^3} + \frac{1}{10^4}$$

$$21.65 = 0.2165 \cdot 10^2 = \left(\frac{2}{10} + \frac{1}{10^2} + \frac{6}{10^3} + \frac{5}{10^4}\right) \cdot 10^2$$

$a = 2$

$$0.1101 = \frac{1}{2} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{1}{2^4}$$

$$0.001011 = 0.1011 \cdot 2^{-2} = \left(\frac{1}{2} + \frac{0}{2^2} + \frac{1}{2^3} + \frac{1}{2^4}\right) \cdot 2^{-2}$$

Floating point numbers

The form of non-zero floating point numbers:

$$\pm a^k \left(\frac{m_1}{a} + \frac{m_2}{a^2} + \cdots + \frac{m_t}{a^t} \right)$$

where

$a > 1$ is an integer, the `base`,

$t > 1$ is an integer, the length of the `mantissa`

$k_- \leq k \leq k_+$ are integers, k is the `characteristic`, $k_- < 0$ and $k_+ > 0$ are fixed.

$1 \leq m_1 \leq a - 1$ is an integer, (the number is in normalized form)

$0 \leq m_i \leq a - 1$ is an integer, for $i = 2, \ldots, t$

In short:

$$\boxed{\pm|k|m_1,\ldots,m_t}$$

The set of the representable numbers is uniquely determined by the numbers

$$\boxed{a, t, k_-, k_+}$$

Example

Let $a = 2$, $t = 4$, $k_- = -3$, $k_+ = 2$.

- Compute the floating-point form the numbers below:

$$0.6875, \quad 0.8125, \quad 3.25, \quad 0.875$$

- Of how many positive, normalized numbers can be represented in the given system?

Facts

For a given a, t, k_-, k_+

- the `largest` (positive) representable number:

$$\begin{gathered}
M_\infty = a^{k_+}\left(\frac{a-1}{a}+\frac{a-1}{a^2}+\cdots+\frac{a-1}{a^t}\right) = \\
= a^{k_+}\left(1-\frac{1}{a}+\frac{1}{a}-\frac{1}{a^2}+\cdots+\frac{1}{a^{t-1}}-\frac{1}{a^t}\right) = \\
= a^{k_+}\left(1-a^{-t}\right)
\end{gathered}$$

- the `smallest` (positive) representable number:

$$\varepsilon_0 = a^{k_-}\left(\frac{1}{a}+0+\cdots+0\right) = a^{k_- - 1}$$

- `subnormal` numbers: if $k = k_-$ and $m_1 = 0$.

Facts

- The number `1` is always representable:

$$1 = a^1 \cdot \frac{1}{a}$$

or

$$1 = [+|1|1, 0, \ldots, 0]$$

- The `right` neighbour of 1:

$$1 + \varepsilon_1 = [+|1|1, 0, \ldots, 0, 1]$$

or

$$1 + \varepsilon_1 = a\left(\frac{1}{a} + 0 + \cdots + 0 + \frac{1}{a^t}\right) = 1 + a^{1-t}$$

that is $\varepsilon_1 = a^{1-t}$ (**the machine epsilon**)

Exercise 5

(a) Write a code that computes the machine epsilon!

(b) Read the help of the function eps! What is the value of eps(1)?

Exercise 6

(a) Write a code that computes ε_0!

(b) What is the value of `eps(0)`?

Exercise 7

Examine the values of `realmin` and `realmax`! What is `realmin('single')` and `realmax('single')`?

The IEEE floating point standard:

	single precision	double precision
size	32 bits	64 bits
mantissa	23+1 bits	52+1 bits
characteristic	8 bits	11 bits
ε_1	$\approx 1.19 \cdot 10^{-7}$	$\approx 2.22 \cdot 10^{-16}$
M_∞	$\approx 10^{38}$	$\approx 10^{308}$

Note that here m_1 is 1 (a constant), so it is not stored explicitly. For the sign 1 bit is reserved.

For a given a, t, k_+, k_- the floating-point numbers is finite subset of the real interval $[-M_\infty, M_\infty]$

Exercise 8

Let $a = 2$, $t = 4$, $k_- = -3$, $k_+ = 2$.

(a) Draw all positive (normalized) numbers from the system!

(b) What is the value of M_∞, ε_0 és ε_1?

(c) What is the distance of two neighbouring numbers?

Example

The set of all positive normalized numbers in the system

$$a = 2,\ t = 4,\ k_- = -3,\ k_+ = 2$$

	$k=0$	$k=1$	$k=2$	$k=-1$	$k=-2$	$k=-3$
0.1000	$\frac{8}{16}$	$\frac{8}{8}$	$\frac{8}{4}$	$\frac{8}{32}$	$\frac{8}{64}$	$\frac{8}{128}$
0.1001	$\frac{9}{16}$	$\frac{9}{8}$	$\frac{9}{4}$	$\frac{9}{32}$	$\frac{9}{64}$	$\frac{9}{128}$
0.1010	$\frac{10}{16}$	$\frac{10}{8}$	$\frac{10}{4}$	$\frac{10}{32}$	$\frac{10}{64}$	$\frac{10}{128}$
0.1011	$\frac{11}{16}$	$\frac{11}{8}$	$\frac{11}{4}$	$\frac{11}{32}$	$\frac{11}{64}$	$\frac{11}{128}$
0.1100	$\frac{12}{16}$	$\frac{12}{8}$	$\frac{12}{4}$	$\frac{12}{32}$	$\frac{12}{64}$	$\frac{12}{128}$
0.1101	$\frac{13}{16}$	$\frac{13}{8}$	$\frac{13}{4}$	$\frac{13}{32}$	$\frac{13}{64}$	$\frac{13}{128}$
0.1110	$\frac{14}{16}$	$\frac{14}{8}$	$\frac{14}{4}$	$\frac{14}{32}$	$\frac{14}{64}$	$\frac{14}{128}$
0.1111	$\frac{15}{16}$	$\frac{15}{8}$	$\frac{15}{4}$	$\frac{15}{32}$	$\frac{15}{64}$	$\frac{15}{128}$

$M_\infty = 2^2(1 - 2^{-4}) = \frac{15}{4}$ and $\varepsilon_0 = 2^{-3-1} = \frac{1}{16}\ \left(= \frac{8}{128}\right)$

Let $y = a^k \cdot 0.m_1 m_2 ... m_t$.

The closest number that is greater than y is in distance:

$$a^k \cdot \frac{1}{a^t} = a^{k-t}$$

Bigger characteristic means bigger distance (stepsize) between neighbouring numbers.

If $k > t$, then the stepsize is larger than 1.

Exercise 9

Examine again the values of the following expressions:

$$2^{66}+1 == 2^{66},\ 2^{66}+10 == 2^{66},\ 2^{66}+100 == 2^{66},$$
$$2^{66}+1000 == 2^{66},\ 2^{66}+10000 == 2^{66}$$

Try to find the smallest $n > 0$ for which $2^{66} + n == 2^{66}$ is `false`! What is the value of `eps(2^66)`?

For double precision ($t = 53$):

y	distance of the right neighbour
1	$\approx 2.22 \cdot 10^{-16}$
16	$\approx 3.5527 \cdot 10^{-15}$
1024	$\approx 2.27 \cdot 10^{-13}$
$2^{20} \approx 10^6$	$\approx 2.33 \cdot 10^{-10}$
$2^{52} \approx 4.5 \cdot 10^{15}$	1
$2^{60} \approx 1.15 \cdot 10^{18}$	256
$2^{66} \approx 7.38 \cdot 10^{19}$	16384

Rounding

Not all numbers has an exact representation in a floating point number system.

Example

The binary representation of $\frac{1}{10}$:

$$0.00011001100110011 00\ldots.$$

The binary representation of $\frac{1}{3}$:

$$0.0101010101010\ldots.$$

Rounding

Let $x \in [-M_\infty, M_\infty]$ a real number, and denote by $fl(x)$ the corresponding floating-point number.

`Regular rounding`

$$fl(x) = \begin{cases} 0, & \text{if } |x| < \varepsilon_0 \\ \text{among the nearest floating point} \\ \text{numbers to } x\text{, the larger} \\ \text{in absolute value,} & \text{if } |x| \geq \varepsilon_0 \end{cases}$$

`Cutting, choping`

$$fl(x) = \begin{cases} 0, & \text{if } |x| < \varepsilon_0 \\ \text{the nearest floating point} \\ \text{number towards zero,} \text{if } |x| \geq \varepsilon_0 \end{cases}$$

Remark

The rounding rules implemented in todays processors are more involved. For simplicity we will use the rules above.

Example

Let $a = 2$, $t = 4$, $k_- = -3$, $k_+ = 2$. What is $fl(0.1)$ in case of choping and regular rounding?

From the binary expansion of 0.1, we get the form:

$$2^{-3} \cdot 0.1100110011001100....$$

Regular rounding:

$$fl(0.1) = 2^{-3} \cdot 0.1101$$

Choping:

$$fl(0.1) = 2^{-3} \cdot 0.1100$$

Exercise 10

Let $a = 2$, $t = 4$, $k_- = -3$, $k_+ = 2$. Compute the corresponding floating point numbers for:

$$0.4, \quad 0.3, \quad \frac{1}{3}, \quad 0.7, \quad \frac{1}{32}$$

Exercise 11

Examine the value of expression $0.4 - 0.5 + 0.1 == 0$! Explain! Examine the value of expression $0.1 - 0.5 + 0.4 == 0$! Explain!

Exercise 8 (cont.)

Let $a = 2$, $t = 4$, $k_- = -3$, $k_+ = 2$. Try to find positive $x \neq y$ floating point numbers, for which:

(f) $x + y < M_\infty$, but $x + y$ is not a floating point number.

(g) $fl(x + y) = x$.

Exercise 12

What will be the value of x after executing the code below?

```
x=1/3;
for i=1:40
   x=4*x-1;
end
```

Why is so different what we see?

Exercise 13

The code below modifies and restores the value of x by successive squarerooting and squareing. In theory x remains the same. What we see in practice? Why?

```
for i=1:60
   x=sqrt(x);
end
for i=1:60
   x=x^2;
end
```

Rounding

Estimating the `absolute` error
in case of `regular rounding`:

$$|fl(x) - x| \leq \begin{cases} \varepsilon_0, & \text{ha } |x| < \varepsilon_0 \\ \frac{1}{2}\varepsilon_1 |x|, & \text{ha } |x| \geq \varepsilon_0 \end{cases}$$

in case of `choping`:

$$|fl(x) - x| \leq \begin{cases} \varepsilon_0, & \text{ha } |x| < \varepsilon_0 \\ \varepsilon_1 |x|, & \text{ha } |x| \geq \varepsilon_0 \end{cases}$$

Rounding

Estimating the `relative` error
in case of `regular rounding`:

$$\boxed{\frac{|fl(x)-x|}{|x|} \le \frac{1}{2}\varepsilon_1}$$

in case of `choping`:

$$\boxed{\frac{|fl(x)-x|}{|x|} \le \varepsilon_1}$$

Addition

Example

Let $a = 10$, $t = 3$. Assuming 1 spare digit compute $fl(x+y) =!$
$x = 0.425 \cdot 10^{-1}$, $y = 0.677 \cdot 10^{-2}$

$y \to y = 0.0677 \cdot 10^{-1}$ **(1 spare digit)**

$x + y = 0.425 \cdot 10^{-1} + 0.0677 \cdot 10^{-1} = 0.4927 \cdot 10^{-1}$

$$fl(x+y) = \begin{cases} 0.492 \cdot 10^{-1}, & \text{choping} \\ 0.493 \cdot 10^{-1}, & \text{regular rounding} \end{cases}$$

Error and operations

Denote by $\triangle$ one of the `+,-,*/`, let x and y floating point numbers. Assuming that the computer performs the operations exactly and assigns a floating point number to the result. Then
in case of regular rounding we have:

$$|fl(x\triangle y) - x\triangle y| \leq \begin{cases} \varepsilon_0, & \text{if } |x\triangle y| < \varepsilon_0 \\ \frac{1}{2}\varepsilon_1 |x\triangle y|, & \text{if } |x\triangle y| \geq \varepsilon_0 \end{cases}$$

in case of choping we have:

$$|fl(x\triangle y) - x\triangle y| \leq \begin{cases} \varepsilon_0, & \text{if } |x\triangle y| < \varepsilon_0 \\ \varepsilon_1 |x\triangle y|, & \text{if } |x\triangle y| \geq \varepsilon_0 \end{cases}$$

$$|x\triangle y| > M_\infty \qquad \Longrightarrow \textbf{ overflow}$$
$$|x\triangle y| < \varepsilon_0 \qquad \Longrightarrow \textbf{ underflow}(fl(x\triangle y) = 0)$$