Lecture 5 (hrs. 7,8) - October 1, 2025, 11:30 - 13:30 A13

(1.22) Remark (consequences of F(2,53) $\neq$ R).

Let M indicate the set of numbers that the computer can manipulate, the computer's '*machine numbers*'. Which set M exactly is *depends on* the computer you are considering. In the case of *Scilab* (and *Octave* and *Matlab*) the set M is 'substantially' F(2,53). Reserving the right to clarify the differences between the two sets later, we assume that:

in *Scilab* it is M = F(2,53)

Consider the following examples (the > character is the *Scilab* console *prompt*).

-
```
> x = 0.1;
```

  Since 0.1 = 1/10 $\notin$ F(2,53), after the assignment the value of x *is not equal to* 1/10.

-
```
> (1 - 9/10) * 10 - 1
ans = - 2.220D-16
```

  We have: 1, 9, 10 $\in$ F(2,53) but 9/10 $\notin$ F(2,53). That is:

  there exist x,y $\in$ F(2,53) s.t. x/y $\notin$ F(2,53)

- Let $f(x) = \dfrac{x(x-1)}{x - \mathrm{sqrt}(x)}$, whose domain is x > 0 and x $\neq$ 1.

  (A) It is: $$f(x) = \frac{x^2 - x}{x - \mathrm{sqrt}(x)} = \frac{(x + \mathrm{sqrt}(x))(x - \mathrm{sqrt}(x))}{x - \mathrm{sqrt}(x)} = x + \mathrm{sqrt}(x)$$

  (B) When x = 2 $\in$ F(2,53) we have:

```
> a = 2 * (2 - 1)/(2 - sqrt(2));

> b = 2 + sqrt(2);

> a == b

ans = F
```

(1.23) Definition (the rounding function).

The computer uses the elements of F(β,m) to approximate real numbers. The approximation is achieved by the *rounding function* rd: R → F(β,m) defined as follows:

rd(x) = the element of F(β,m) *closest* to x or, in case of ambiguity, that of the two elements of F(β,m) equidistant from x that has the fraction ending in an *even digit*.

(1.24) Remark.

The definition is well posed if β is even and m $\geqslant$ 2. In that case, if the last digit of the fraction of $\xi \in F(\beta,m)$ is *even* (respectively: *odd*), the last digit of the fraction of the successor of ξ is *odd* (respectively: *even*).

If β is even and m = 1 or β is odd, however, the definition is not well posed. For example, in F(3,2) the positive elements with zero exponent are:

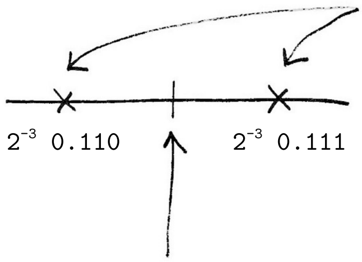
$$3^0\ 0.10\ ;\ 3^0\ 0.11\ ;\ 3^0\ 0.12\ ;\ 3^0\ 0.20\ ;\ \ldots$$

and the last two elements written are consecutive and *both* have an *even* fraction's last digit.

(1.25) Example.

Let x = 1/10. We want to determine the rounded value of x in F(2,3).

We already know (see Example (1.15)) that: $x = 2^{-3}\ 0.1\overline{100}$. Then we have the situation in figure:



elements of F(2,3) *adjacent* to x (the left one is obtained by *truncating* the fraction of x to the number of digits indicated by the precision - in this case 3 - the right one is the successor)

midpoint = $2^{-3}\ 0.1101 > x \Rightarrow rd(x) = 2^{-3}\ 0.110$ (= 3/32)

(1.26) Remark.

The rd function is *not* a function that the computer makes available to the user, but it is essential to understand how:

(1) the computer 'reads' real numbers;
(2) the computer performs operations on the elements of $F(\beta,m)$.

(1.27) Example.

Let's take up the first Example of the Remark (1.22). In *Scilab* the effect of assignment:

```
> x = 0.1
```

is: the value $rd(0.1) \in F(2,53)$ is assigned to the variable x (if the variable x does not exist at the time of the assignment, it is created).

The calculator approximates the real number with its rounded value in $F(\beta,m)$. *We are interested in how large an error is being made.*

(1.28) Theorem (bound on relative error).

Let rd the rounding function in F(β,m). For every real number x $\neq$ 0 it is:

$$\frac{|rd(x)-x|}{|x|} \leqslant \frac{1}{2}\,\beta^{1-m} = u \text{ (\textit{machine precision})}$$

(Proof...)

(1.29) Remark.

- The bound is *uniform*, in the sense that the quantity that limits the error is *independent of* x (it depends only on the parameters β and m that define the set of numbers).
- For F(2,53) it is $u = \frac{1}{2}\,2^{1-53} = 2^{-53} \approx 1.11\ 10^{-16}$.
- If we consider the *absolute* error, from the previous Theorem we obtain, for each real number x, the (*non-uniform*!) limitation:

  $$|rd(x) - x| \leqslant u\ |x|$$

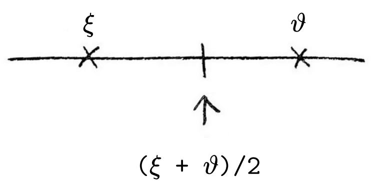
  It follows that *the further away* x is from zero, *the greater* the absolute error *can be*.

  The substantial difference between the two bounds - one is uniform and the other is not - is due to how the elements of F(β,m) are distributed on the real line. The distribution is *specifically designed* to achieve uniform bounding of the relative error.

(1.30) Example.

Let ξ be a positive element of F(2,53) and ϑ the successor of ξ. It is:

- $\xi/2 \in$ F(2,53) , $\vartheta/2 \in$ F(2,53)
- $\xi/2 + \vartheta/2 \notin$ F(2,53)



Choosing ξ = 1, in *Scilab* we have the following dialogue (for each t $\in$ F(2,53), nearfloat('succ',t) is the successor of t):

```
> c = 1/2 + nearfloat('succ',1)/2

c = 1

> c == 1

ans = T
```

To understand the dialogue, it is necessary to understand how *Scilab* sums two machine numbers. If $\xi,\vartheta \in$ F(2,53), we indicate with $\xi \oplus \vartheta$ the value assigned by *Scilab* to the expression $\xi + \vartheta$. It is, by definition:

$$\xi \oplus \vartheta = \mathrm{rd}(\xi + \vartheta)$$

The value is defined ‘as best as possible’ in the sense that the error between the exact value $\xi + \vartheta$ and the defined value $\xi \oplus \vartheta$ is *as small as possible*.

Let's go back to the Example. The value that *Scilab* assigns to c is, then:

1/2 ⊕ nearfloat(‘succ’,1)/2 = rd(1/2 + nearfloat(‘succ’,1)/2)

which, according to the definition of rounding, is equal to 1 (the one, between the two elements adjacent to the number to be rounded, which has the last digit of the fraction *even*).

What happens in the first assignment is: