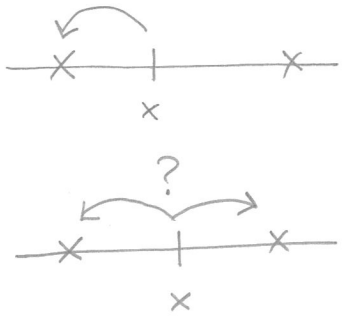
2) come il calcolatore utilizza gli elementi di $F(\beta, m)$ ...

... per APPROSSIMARE numeri reali

- funzione ARROTONDATO $\quad rd : \mathbb{R} \to F(\beta, m)$

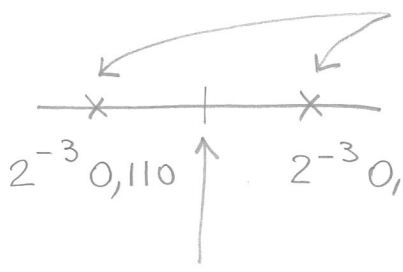
~~def~~ : $\forall x \in \mathbb{R}$, $rd(x) = \ldots$

... l'elemento di $F(\beta, m)$ più vicino ad x



o, se c'è ambiguità, quello dei due che ha frazione che termina con 0 ($\beta = 2$)

Es : $F(2,3)$, $x = \frac{1}{10}$, $b = -3$, $g = 0,\overline{1100}$



$\Rightarrow rd(x) = 2^{-3}\,0,110$

Oss Se β pari e $m \geq 2$ allora: se l'ultima cifra della frazione di $\xi \in F(\beta, m)$ è PARI (risp. DISPARI), l'ultima cifra della frazione del successore di ξ è DISPARI (risp. PARI).

se β dispari non è vero!

Es: elem consecutivi in $F(3,2)$ sono

$3^0\ 0,1\textcircled{0}$ pari
$3^0\ 0,1\textcircled{1}$ dispari
$3^0\ 0,1\textcircled{2}$ pari
$3^0\ 0,2\textcircled{0}$ PARI!

- rd non è una funzione che il calcolatore mette a disposizione dell'utilizzatore, ma è indispensabile per capire come...

1) il calcolatore "legge" i numeri reali
2) il calcolatore fa operazioni sugli elementi di $F(\beta, m)$.

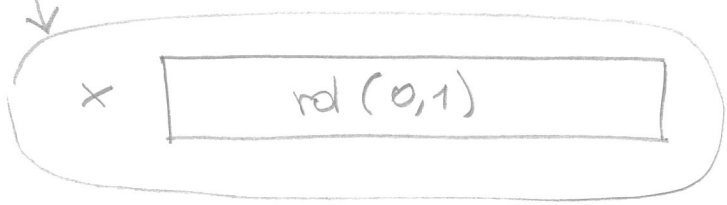
Es ①: > x = 0,1 (comando Scilab)

effetto...

a) se non esiste già, viene creata una variabile di nome x

b) alla variabile viene assegnato il valore

$rd(0,1) \in F(2,53)$



Il calcolatore APPROSSIMA il numero reale con il suo arrotondato in $F(\beta, m)$

Pb: che errore viene commesso?

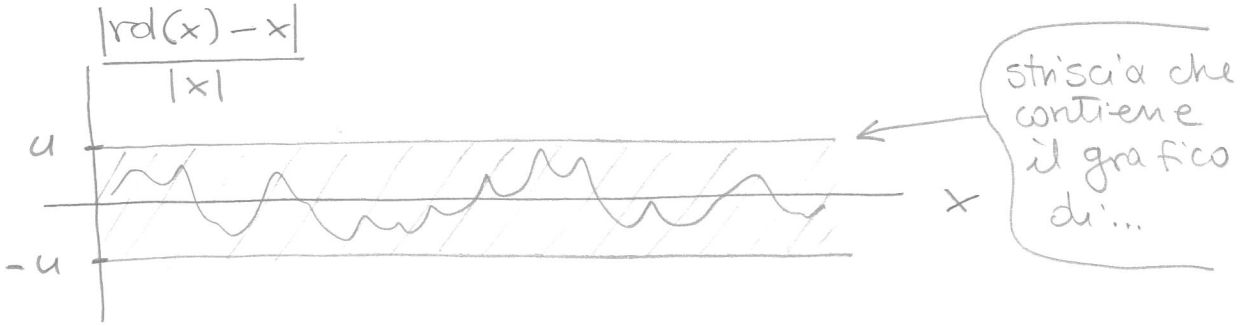
Soluzione:

Teo: $\forall x \in \mathbb{R}, x \neq 0, \quad \frac{|rd(x) - x|}{|x|} \leq \frac{1}{2}\beta^{1-m} \equiv u$

u: PRECISIONE di MACCHINA

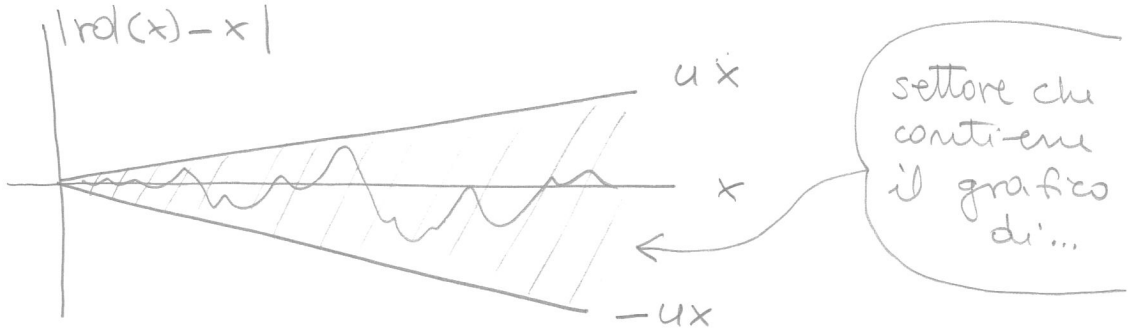
Oss:

- La stima è <u>uniforme</u> nel senso che la quantità che limita l'errore è <u>indipendente da x</u> (dipende solo dai parametri che definiscono $F(\beta, m)$)



- in $F(2,53)$ è $u = \frac{1}{2} 2^{1-53} = 2^{-53} \simeq 1.11\ 10^{-16}$
  [Scilab: number_properties ("eps")]
- l'errore considerato è quello RELATIVO; per l'errore ASSOLUTO si ottiene la stima (<u>non uniforme</u>!):

$$|rd(x) - x| \leq u|x| \quad (\text{vale } \forall x \in \mathbb{R})$$

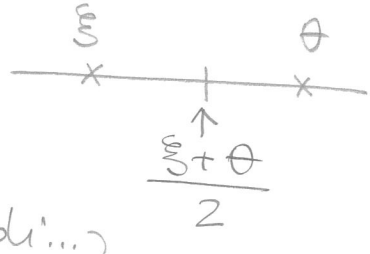


Questo accade per come sono distribuiti gli elementi di $F(\beta, m)$. Questi ultimi sono pesati appositamente per ottenere la stima uniforme dell'errore relativo.

(Nota: per i numeri in virgola fissa accade l'opposto: la stima dell'errore assoluto è uniforme, quella dell'errore relativo no.)

Es (2): ξ elem positivo di $F(2,53)$
θ successore di ξ ($\Rightarrow \theta \in F(2,53)$)

- $\frac{1}{2}\xi \in F(2,53)$, $\frac{1}{2}\theta \in F(2,53)$
- $\frac{1}{2}\xi + \frac{1}{2}\theta \notin F(2,53)$



Si ha ($\xi = 1$):

il successore di...

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

c = 1

> c == 1        <- "c è uguale a 1 ?"

ans = T
```

ans: variabile "di appoggio" che "contiene la risposta"

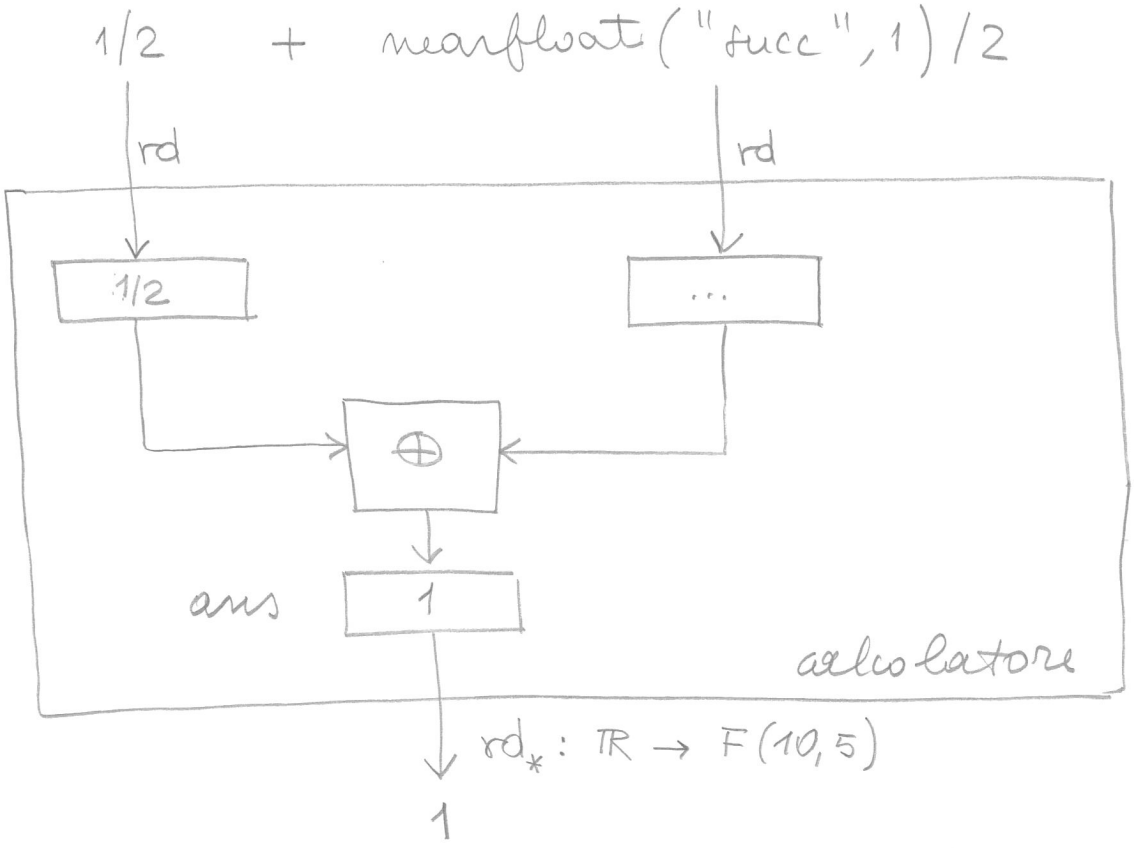
In Scilab (nel calcolatore) si ha:

$$\forall\, \xi_1, \xi_2 \in F(\beta, m)$$

$$\xi_1 \oplus \xi_2 = rd(\underbrace{\xi_1 + \xi_2}_{\text{somma "esatta"}})$$

$\oplus$: "pseudo somma"
si utilizza un simbolo non usuale per chiarezza.

e quello che accade è:



Es. Nell'Es finale della lez precedente, si utilizzano due sequenze diverse di op per calcolare i valori del polinomio $(x-2)^{13}$. Le due sequenze sono equivalenti IN $\mathbb{R}$ (ovvero "in teoria") ma NON lo sono operando in $F(2,53)$ (ovvero "con il calcolatore"), come evidente dal grafico allegato (in cui f_test_2 è relativo al calcolo dei valori con $(x-2)^{13}$ e f_test_2_h con horner...). Dunque il comportamento del calcolatore è una ragionevole conseguenza dell'aver applicato il metodo di bisezione a DUE FUNZIONI DIVERSE.

number_properties

determine floating-point parameters

Calling Sequence

```
pr = number_properties(prop)
```

Arguments

prop
: string

pr
: real or boolean scalar

Description

This function may be used to get the characteristic numbers/properties of the floating point set denoted here by F(b,p,emin,emax) (usually the 64 bits float numbers set prescribe by IEEE 754). Numbers of F are of the form:

```
sign * m * b^e
```

e is the exponent and m the mantissa:

```
m = d_1 b^(-1) + d_2 b^(-2) + .... + d_p b^(-p)
```

d_i the digits are in [0, b-1] and e in [emin, emax], the number is said "normalised" if d_1 ~= 0. The following may be gotten:

prop = "radix"
: then pr is the radix b of the set F

prop = "digits"
: then pr is the number of digits p

prop = "huge"
: then pr is the max positive float of F

prop = "tiny"
: then pr is the min positive normalised float of F

prop = "denorm"
: then pr is a boolean (%t if denormalised numbers are used)

prop = "tiniest"
: then if denorm = %t, pr is the min positive denormalised number else pr = tiny

prop = "eps"
: then pr is the epsilon machine (generally (b^(1-p))/2) which is the relative max error between a real x (such than |x| in [tiny, huge]) and fl(x), its floating point approximation in F

prop = "minexp"
: then pr is emin

prop = "maxexp"
: then pr is emax

Remarks

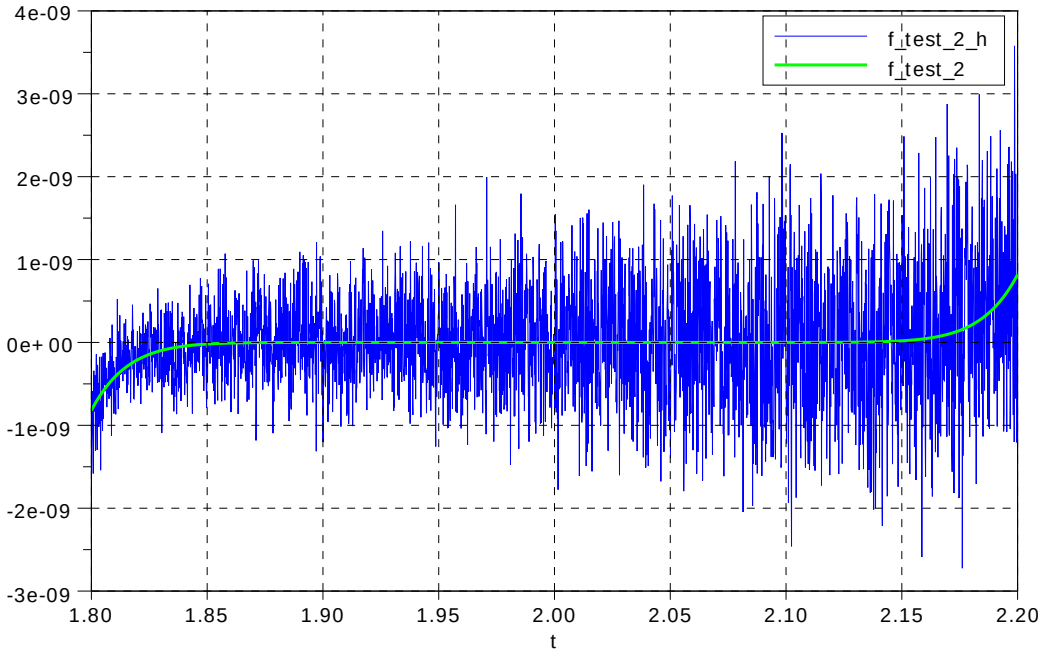
This function uses the lapack routine dlamch to get the machine parameters (the names (radix, digit, huge, etc...) are those recommended by the LIA 1 standard and are different from the corresponding lapack's ones) ; CAUTION: sometimes you can see the following definition for the epsilon machine : eps = b^(1-p) but in this function we use the traditionnal one (see prop = "eps" before) and so eps = (b^(1-p))/2 if normal rounding occurs and eps = b^(1-p) if not.

Examples

```
b = number_properties("radix")
eps = number_properties("eps")
```

See Also

- nearfloat — get previous or next floating-point number
- frexp — dissect floating-point numbers into base 2 exponent and mantissa


f_test_2_h
f_test_2
4e-09
3e-09
2e-09
1e-09
0e+00
-1e-09
-2e-09
-3e-09
1.80
1.85
1.90
1.95
2.00
2.05
2.10
2.15
2.20
t