ANALISI DEGLI ERRORI

Nell'ambito dell'elaborazione numerica, è necessario esaminare gli errori che derivano dall'uso di uno strumento di calcolo: la loro origine, la loro propagazione e alcune tecniche per la loro valutazione.

Un calcolatore è in grado di rappresentare solo un numero finito di cifre: ne consegue la possibilità che un numero reale venga *approssimato* e che le operazioni forniscano risultati non esattamente rappresentabili. Allora una successione di operazioni (o algoritmo) eseguita su calcolatore da luogo alla creazione e alla propagazione di errori, detti *errori di arrotondamento*.

I risultati ottenuti sono diversi dai risultati esatti. Occorre dare una *stima dell'errore commesso*, per misurare la precisione dei risultati calcolati, dipendente dal numero di cifre usate e/o dall'ordine di esecuzione delle operazioni.

- Numeri di macchina o numeri finiti
- Operazioni sui numeri finiti
- Propagazione degli errori: errori inerenti ed errori di arrotondamento
- Malcondizionamento di un problema e stabilità di un algoritmo.

Definizione di errore

Sia $\alpha \in \mathbb{R}$ e α^* una sua approssimazione.

- ERRORE ASSOLUTO: $E_a = |\alpha - \alpha^*|$;
- ERRORE RELATIVO ($\alpha \neq 0$): $E_r = \frac{E_a}{|\alpha|}$;
- ERRORE PERCENTUALE: $E_p = (E_r \times 100)\%$

ESEMPI.

$\alpha = 0.3\ 10^1$ $\alpha^* = 0.31\ 10^1$	$E_a = 0.1$	$E_r = 0.3333..\ 10^{-1}$ $E_p = 3.33..\%$
$\alpha = 0.3\ 10^{-3}$ $\alpha^* = 0.31\ 10^{-3}$	$E_a = 0.1\ 10^{-4}$	$E_r = 0.3333..\ 10^{-1}$ $E_p = 3.33..\%$
$\alpha = 0.3\ 10^4$ $\alpha^* = 0.31\ 10^4$	$E_a = 0.1\ 10^3$	$E_r = 0.3333..\ 10^{-1}$ $E_p = 3.33..\%$

Sia $\alpha \in \mathbb{R}, \alpha \neq 0, \alpha = \pm(\sum_{i=1}^{\infty} a_i \beta^{-i})\beta^p$.

- TRONCAMENTO di α alla t-esima cifra:

$$\alpha_t = \pm(\sum_{i=1}^{t} a_i \beta^{-i})\beta^p$$

- ARROTONDAMENTO di α alla t-esima cifra (β pari):

$$\alpha_{rr} = \pm(\sum_{i=1}^{\infty} a_i \beta^{-i} + \frac{1}{2}\beta^{-t})_t \beta^p$$

ove

$\alpha_{rr} = \alpha_t$ se $0 \leq a_{t+1} < \beta/2$ round down

$\alpha_{rr} = \pm(\sum_{i=1}^{t-1} a_i \beta^{-i} + (a_t + 1)\beta^{-t})\beta^p$ se $\beta/2 \leq a_{t+1} \leq \beta - 1$ round up.

Valutazione degli errori nel caso di troncamento e arrotondamento

- TRONCAMENTO di α alla t-esima cifra:

$$\alpha_t = \pm .a_1 a_2 \ldots a_t \ \beta^p$$

$$E_a = |\alpha - \alpha_t| = .000\ldots 0 a_{t+1} \ldots \ \beta^p = .a_{t+1} a_{t+2} \ldots \ \beta^{-t+p}$$

Poichè $.a_{t+1} a_{t+2} \ldots < 1, \ \Rightarrow E_a < \beta^p \beta^{-t}$.

$$E_r = \frac{|\alpha - \alpha_t|}{|\alpha|} < \frac{\beta^p \beta^{-t}}{\beta^p \beta^{-1}} = \beta^{1-t}$$

perchè la mantissa di α è $\geq 1/\beta$.

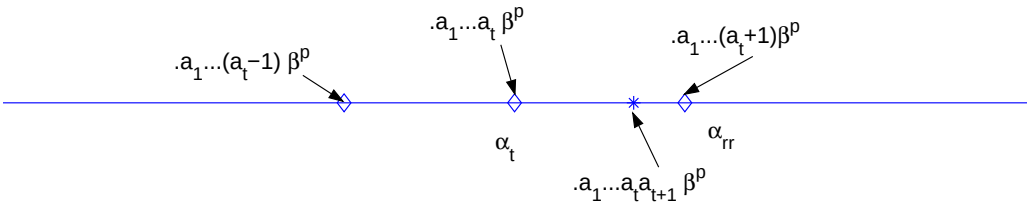
- ARROTONDAMENTO di α alla t-esima cifra (β pari):

$$\alpha_{rr} = \pm (.a_1 \ldots a_t a_{t+1} \ldots + \frac{1}{2}\beta^{-t})_t \beta^p =$$

$$= \begin{array}{cc} \nearrow & \pm .a_1 \ldots a_t \ \beta^p \quad \text{se } 0 \leq a_{t+1} < \beta/2 \\ \searrow & \pm .a_1 \ldots (a_t + 1) \ \beta^p \quad \text{se } \beta/2 \leq a_{t+1} \leq \beta - 1 \end{array}$$

$$E_a = |\alpha - \alpha_{rr}| \leq \frac{1}{2}\beta^{-t}\beta^p$$

$$E_r = \frac{|\alpha - \alpha_{rr}|}{|\alpha|} \leq \frac{1}{2}\frac{\beta^p \beta^{-t}}{\beta^p} = \frac{1}{2}\beta^{1-t}$$



Si dice che un numero α è una approssimazione del numero α^* con t cifre significative nella sua base di rappresentazione se

$$\frac{|\alpha - \alpha^*|}{|\alpha^*|} \leq \frac{1}{2}\beta^{1-t}$$

Esempio.
$\pi = 3.141592654... = 0.3141592654...10^1$

- mantissa: $0.3141592654...$
- Parte esponente: 10^1

Troncamento alla sesta cifra ($\beta = 10$, $t = 6$, $p = 1$):
$\pi \simeq .314159\ 10^1$
$E_a = .2654...10^{-5} < 10^{-5}$
$E_r = \frac{.2654...10^{-5}}{\pi} = .84...10^{-6} < 10^{-5}$

Arrotondamento alla sesta cifra ($\beta = 10$, $t = 6$, $p = 1$):
$\pi \simeq .314159\ 10^1$
$E_a = .2654...10^{-5} < .5\ 10^{-5}$
$E_r = \frac{.2654...10^{-5}}{\pi} = .84...10^{-6} < .5\ 10^{-5}$

Arrotondamento alla settima cifra ($\beta = 10$, $t = 7$, $p = 1$):
$\pi \simeq .3141593\ 10^1$
$E_a = .346...10^{-6} < .5\ 10^{-6}$
$E_r = \frac{.346...10^{-6}}{\pi} = .11...10^{-6} < .5\ 10^{-6}$
3.141593 è un'approssimazione di π con 7 cifre significative.

Ordine di accuratezza

Definizione. La funzione $f(x)$ è detta un **o-piccolo** della funzione $g(x)$ per $x \longrightarrow x_0$ e denotata con $f(x) = o(g(x))$ se esiste una funzione $k(x) \geq 0$ tale che

$$|f(x)| \leq k(x)|g(x)|, \qquad \lim_{x \to x_0} k(x) = 0$$

In tal caso si dice che $f(x)$ è trascurabile rispetto a $g(x)$ per x che tende a x_0.

In pratica f/g tende a 0 per $x \longrightarrow x_0$. La notazione $f(x) = o(1)$ indica che $f(x)$ tende a 0 per $x \longrightarrow x_0$.

Definizione. La funzione $f(x)$ è detta un **O-grande** della funzione $g(x)$ per $x \longrightarrow x_0$ e denotata con $f(x) = O(g(x))$ se esiste una costante $C > 0$ tale che

$$|f(x)| \leq C|g(x)|$$

per x in un intorno di x_0.

In pratica f/g si mantiene limitato in un intorno di x_0. La notazione $f(x) = O(1)$ indica che $f(x)$ si mantiene limitata in un intorno di x_0.

Per esempio, date le funzioni $f(x) = \frac{x^3}{1+x}$ e $g(x) = x^2$, avremo per $x \longrightarrow 0$:

$$\frac{x^3}{1+x} = O(x^2)$$

Infatti $\frac{x^3}{1+x} \leq \frac{x^3}{x} = x^2$ per $x \geq 0$. In pratica la notazione **O-grande** consente di descrivere il comportamento di una funzione in termini di funzioni elementari note $(x^n, x^{1/n}, a^x, log_a x, ...)$.

Definizione. La successione $\{x_n\}$ è detta un **O-grande** della successione $\{y_n\}$ se esistono costanti C ed N tali che

$$|x_n| \leq C|y_n| \qquad n \geq N$$

Ad esempio la successione

$$\frac{n^2-1}{n^3} = O(\frac{1}{n})$$

perchè $(n^2-1)/n^3 \leq n^2/n^3 = 1/n$ per $n \geq 1$.

Numeri finiti o numeri di macchina

A causa della limitata lunghezza della parola di memoria, sono rappresentabili effettivamente su calcolatore:

- un intervallo limitato di interi (numeri fixed point o a punto fisso);
- un insieme finito di numeri razionali (numeri floating point o a virgola mobile).

Numeri fixed point (β, t)

Sia β un intero maggiore di 1 e $t+1$ il numero di cifre a disposizione per la rappresentazione di un intero. β si usa come base di rappresentazione e t cifre si usano per la rappresentazione del valore assoluto del numero.

1. Sia N un intero non negativo. Sia $N = d_p d_{p-1} \ldots d_1$ la rappresentazione di N in base β.

$$fi(N) = \begin{cases} 000\ldots00 d_p d_{p-1} \ldots d_1 & t \geq p \quad fi(N) = N \\ t - p \text{ zeri} & \\ d_t d_{t-1} \ldots d_1 & t < p \quad fi(N) \neq N \end{cases}$$

   Allora $fi(N) = N$ se $t \geq p$, altrimenti $fi(N)$ è congruo N modulo β^t nelle prime t cifre. In questo caso si verifica un OVERFLOW intero, che non arresta la macchina. Sono rappresentabili solo le t cifre meno significative.
   Il più grande intero rappresentabile esattamente è dato da:

$$0ccc\ldots ccc = (\beta - 1)\beta^{t-1} + \ldots + (\beta - 1)\beta^0 = \beta^t - 1$$

   ove $c = \beta - 1$. Il punto radice è omesso poichè è pensato a destra della cifra meno significativa in posizione fissa.
   Sono rappresentato esattamente solo gli interi non negativi compresi tra $[0, \beta^t - 1]$.

2. Sia N un intero negativo. Allora

$$fi(N) = (\beta^{t+1} - |N|)_{t+1}$$

Questa si dice **rappresentazione complemento alla base β in $t+1$ cifre**. Per ottenere $fi(N)$, si prende il valore assoluto di N rappresentato in $t+1$ cifre e poi si complementa alla base la cifra meno significativa diversa da 0 e a $\beta - 1$ le altre cifre più significative. Pertanto la cifra $t+1$-esima è uguale a $c = \beta - 1$.

Il più piccolo intero esattamente rappresentabile è:

$$c000...00 = fi(-1000...00) = -\beta^t$$

Infatti $\beta^{t+1} - |\alpha| = (\beta - 1)\beta^t \Rightarrow |\alpha| = (\beta - \beta + 1)\beta^t$. Dunque $\alpha = -\beta^t$.

Sono esattamente rappresentabili $2\beta^t$ interi, appartenenti all'intervallo $[-\beta^t, \beta^t - 1]$. Al di fuori dell'intervallo si incorre nell'overflow intero.

ESEMPI

- $\beta = 2, t + 1 = 32$; sono rappresentabili esattamente gli interi in $[-2^{31}, 2^{31} - 1]$.
- $\beta = 2, t + 1 = 16$; sono rappresentabili gli interi in $[-32768, 32767]$. Per esempio

$$\begin{aligned} fi((1235)_{10}) &= fi((1001101001)_2) = 0000001001101001 \\ fi(-(1235)_{10}) &= fi(-(1001101001)_2) = 1111110110010111 \\ fi(-1) &= 1111111111111111 \end{aligned}$$

In base 2, il complemento alla base in $t + 1$ cifre si ottiene ponendo 1 in corrispondenza della cifra meno significativa diversa da 0 e poi scambiando 1 con 0 e 0 con 1 per le cifre più significative. Un'altra regola è di scambiare 0 con 1 e 1 con 0 e poi di aggiungere 1.

$$\begin{aligned} fi(-8) &= fi(-(1000)_2) = 1111111111110111 + 1 \\ &= 1111111111111000 \end{aligned}$$

ESERCIZIO. Sia $\beta = 2$ e $t + 1 = 16$. Rappresentare in fixed point i numeri 1023 e -31128.

$$\begin{aligned} fi(1023) &= fi((1111111111)_2) = 0000001111111111 \\ fi(-(31128)_{10}) &= fi(-111100110011000)_2) \\ &= 1000011001101000 \end{aligned}$$

Quando la base è 2, valgono le seguenti proprietà:

$$\begin{aligned} fi(2^t - 1) + 1 &= 011..111 + 1 = \\ &= 100...000 \\ &= fi(-2^t) \\ fi(-2^t) - 1 &= 100...000 - 1 = \\ &= 0111...111 = \\ &= fi(2^t - 1) \end{aligned}$$

Le proprietà servono a individuare le caratteristiche della rappresentazione fixed point su una macchina.

ESEMPIO. Il seguente codice permette di vedere le caratteristiche dei numeri fixed point (int) sulla macchina su cui è eseguito;:

```
#include <stdio.h>
#include <math.h>
main()
{int num; double it,pr;
num=1;
while(num>=0) ++ num;
-- num;
pr=num;
it=log10(pr+1)/log10(2);
printf("max int rappresentabile=\%d cifre=\%lf",num,it+1);
num=-1;
while(num<0)--num;
++num;
printf("min intero rappresentabile=\%d ",num);
}
```

Un possibile risultato è:

```
[rgv@dns ~]\$ ./prova.x
max int rappresentabile=2147483647 cifre=32.000000
min intero rappresentabile=-2147483648
```

Per $\beta = 2$, la cifra $t + 1$-esima permette di stabilire in modo veloce il segno di un numero: se il bit $t + 1$ vale 0 il numero è positivo o nullo, altrimenti è negativo.

INSIEME DEI NUMERI FIXED POINT INTERI RAPPRESENTABILI con $\beta = 2, t + 1 = 4$

$fi(N)$	N
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Aritmetica dei numeri fixed point

Si assume di operare con interi non negativi tali che $fi(N) = N$.

SOMMA

$$fi(N_1 + N_2) = fi(fi(N_1) + fi(N_2)) = (N_1 + N_2)_{t+1}.$$

Si può avere un overflow intero quando si ha un riporto sulla cifra $t+1$-esima. In questo caso la somma di due positivi fornisce un numero che è la rappresentazione di un negativo.

ESEMPI. $\beta = 2, t = 5$.

- $N_1 = 010010 (= 18_{10}),\ N_2 = 000101 (= 5_{10}) \Rightarrow$
  $fi(N_1 + N_2) = 010111 (= 23_{10})$.

  $$\begin{array}{r} 010010 \\ 000101 \\ \hline 010111 \end{array}$$

- $N_1 = 010011 (= 19_{10}),\ N_2 = 001110 (= 14_{10}) \Rightarrow$
  $fi(N_1 + N_2) = 100001 = fi(-31_{10})$ invece di 33_{10}.

  $$\begin{array}{r} 010011 \\ 001110 \\ \hline 100001 \end{array}$$

DIFFERENZA

$$fi(N_1 - N_2) = (fi(N_1) + fi(-N_2))_{t+1}$$

In pratica:

$$fi(N_1 - N_2) = (N_1 + \beta^{t+1} - N_2)_{t+1} = (\beta^{t+1} + (N_1 - N_2))_{t+1}$$

Il risultato ottenuto differisce da quello esatto di β^{t+1}. Ma se non si toglie β^{t+1} e si tronca alla $t+1$-esima cifra significativa si ottiene esattamente la rappresentazione fixed del risultato; nel caso di differenza tra due numeri dello stesso segno non si ha mai overflow.

1. Sia $N_1 \geq N_2$. Allora $N_1 - N_2$ è positivo o nullo. $\beta^{t+1} + N_1 - N_2$ ha una cifra 1 nella $t+2$-esima posizione significativa. Troncare alla $t+1$-esima cifra equivale a ottenere la rappresentazione corretta del numero positivo o nullo $N_1 - N_2$.
   Se $\beta = 2, t+1 = 6, N_1 = 001111 (= 15_{10}), N_2 = 000111 (= 7_{10})$,

$$\begin{aligned} fi(N_1 - N_2) &= (001111 + 111001)_6 = (1001000)_6 \\ &= 001000 = fi(8_{10}) \end{aligned}$$

2. Sia $N_1 < N_2$. Allora il risultato è il numero negativo $-(N_2 - N_1)$, di valore assoluto $N_2 - N_1$.

$$\beta^{t+1} + N_1 - N_2 = \beta^{t+1} - (N_2 - N_1) = fi(-(N_2 - N_1))$$

   Tale numero esprime la rappresentazione complemento alla base in $t+1$-cifre del risultato.
   Se $N_1 = 000111 (= 7_{10}), N_2 = 001111 (= 15_{10})$,

$$\begin{aligned} fi(N_1 - N_2) &= (000111 + 110001) = (111000) \\ &= fi(-001000) = fi(-8_{10}). \end{aligned}$$

PRODOTTO

Riconducibile ad addizioni e scorrimenti delle cifre a destra. Poichè il prodotto di interi in $t+1$ cifre può produrre un risultato in $2(t+1)$ cifre, spesso si incorre in overflow intero. Si usa un accumulatore B di $2(t+1)$ cifre, la cui parte più significativa R è inizialmente nulla.
$\beta = 2, t+1 = 6$.

$N_1 = 000011 = 3_{10}, N_2 = 000101 = 5_{10}; fi(N_1 * N_2) = 15_{10}$.

000011	$\Leftarrow$ A	moltiplicando
000000	000101	$\Leftarrow$ B
R	moltiplicatore	

Si fanno $t+1 = 6$ passi.

000011			
000001	100010	$1 \times fi(N_1) + R \rightarrow R$	$\Rightarrow$
000000	110001	$0 \times fi(N_1) + R \rightarrow R$	$\Rightarrow$
000001	111000	$1 \times fi(N_1) + R \rightarrow R$	$\Rightarrow$
000000	111100	$0 \times fi(N_1) + R \rightarrow R$	$\Rightarrow$
000000	011110	$0 \times fi(N_1) + R \rightarrow R$	$\Rightarrow$
000000	001111	$0 \times fi(N_1) + R \rightarrow R$	$\Rightarrow$

QUOZIENTE

Si usano due accumulatori A e B. In A si mette il divisore e nella parte R di B il dividendo. Si riconduce a sottrazioni e scorrimenti. Il primo passo è di eseguire s scorrimenti a destra di R fino a che $fi(N_2) \leq R < \beta fi(N_2)$. Si eseguono poi $s+1$ passi uguali (l'ultimo senza scorrimento verso sinistra).

$\beta = 2, t+1 = 6.$

$N_1 = 010000 = 16_{10}, N_2 = 000101 = 5_{10}; fi(N_1/N_2) = 3_{10}$, con resto 1.

	000101	A		
B	010000	000000	s=1	
B	001000	000000		
B	000011	000001	q=1	$\leftarrow R - fi(N_2)$
B	000110	000010	$\leftarrow$	
B	000001	000011	q=1	$R \leftarrow R - fi(N_2)$
	resto R	quoziente		

L'aritmetica tra numeri fixed point è esatta purchè si resti nell'intervallo rappresentabile.

NUMERI FLOATING POINT

L'insieme dei numeri reali è SIMULATO su un calcolatore mediante un insieme di numeri finiti F o numeri floating point; tale insieme dipende da quattro parametri:

- β: base di rappresentazione;
- t: numero di cifre per la rappresentazione della mantissa;
- L: valore del più piccolo esponente rappresentabile;
- U: valore del più grande esponente rappresentabile.

Si denota tale insieme con F(β, t, L, U).
In particolare, si descrive il formato IEEE (Institute of Electrical and Eletronic Engineerings) standard dell'aritmetica binaria floating point, definita dalle convenzioni contenute nel documento 754 dell'ANSI. Tale aritmetica usa $\beta = 2$.
Si scrive un numero reale $\alpha \neq 0$ espresso in base 2 nella notazione speciale (normalizzazione IEEE):

$$\alpha = (-1)^s(1.a_2a_3...a_ta_{t+1}...)2^p$$

Si rappresenta:

- segno: si rappresenta in un bit il valore di s, pari a 0 se $\alpha > 0$ e a 1 se $\alpha < 0$;
- esponente p: è un intero che deve essere compreso tra L e U; la rappresentazione di p è per traslazione in l bit, ossia:

  $$\text{rappresentazione di } p = p + bias$$

  ove il bias è, dunque, la rappresentazione dello 0;
- mantissa: vengono rappresentate i t bit più significativi, troncando; fisicamente poichè il primo bit è sempre 1, vengono rappresentati solo $t - 1$ bit ($a_2 a_3 ... a_t$).

	SEMPLICE PRECISIONE	DOPPIA PRECISIONE
N. totale di bit	32 (4 byte)	64 (8 byte)
segno s	1 bit	1 bit
t	24	53
l	8	11
bias	127 (01111111)	1023 (01111111111)
U	127	1023
L	-126	-1022

ESEMPI. Si usano le convenzioni della singola precisione.

- Il numero $1 = 1.00\ 2^0$ è rappresentato come
  0 01111111 00000000000000000000000
  segno esponente mantissa
- Il numero $(4.25)_{10} = (100.01)_2 = 1.0001\ 2^{10}$ è rappresentato come
  0 10000001 00010000000000000000000
  segno esponente mantissa
  127+2=129
- Il numero $(3.141592)_{10} = (11.001001000011111110101111..)_2 =$
  $= 1.10010010000111111110101111\ 2^1$ è rappresentato come
  0 10000000 10010010000111111010111
  segno esponente mantissa
  127+1=128
- Il numero $(0.333333)_{10} = (0.01010101010101010000111010..)_2 =$
  $= 1.01010101010101000111010\ 2^{-10}$ è rappresentato come
  0 01111101 01010101010101000111010
  segno esponente mantissa
  127-2=125

Più piccolo numero rappresentabile in valore assoluto= $1.0000\ldots.\ 2^{-126} \simeq 10^{-38}$

0 00000001 00000000000000000000000

segno esponente mantissa

-126+127=1

Più grande numero rappresentabile in valore assoluto= $1.1111\ldots1\ 2^{127} = (1 + 1 - 2^{-23})2^{127} \simeq 10^{38}$

0 11111110 11111111111111111111111

segno esponente mantissa

127+127=254

Se si cerca di rappresentare un numero in valore assoluto pi'u piccolo del più piccolo rappresentabile, si incorre nell'UNDERFLOW floating point; il calcolatore rappresenta il numero con 0 e prosegue l'elaborazione. Se si cerca di rappresentare un numero più grande del più grande rappresentabile in valore assoluto, si incorre nell'OVERFLOW floating point che arresta l'elaborazione.

Rappresentazione degli esponenti
$p = 0$ è rappresentato con 01111111.
$p = 1$ è rappresentato con 100000000.
$p = -1$ è rappresentato con 01111110.
$p = 127 = U$ è rappresentato con 11111110=$(254)_{10}$.
$p = -126 = L$ è rappresentato con 00000001=$(1)_{10}$.

Gli esponenti negativi o nulli hanno il bit più significativo uguale a 0, gli esponenti positivi uguale a 1.

RAPPRESENTAZIONI SPECIALI

mantissa	esponente		
0	0	rappresenta	$(-1)^s 0$
0	$11111111 = (255)_{10}$	rappresenta	$\pm\infty$
$\neq 0$	$11111111 = (255)_{10}$	rappresenta	NaN

NUMERI DENORMALIZZATI: sono numeri più piccoli di 2^{-126}.

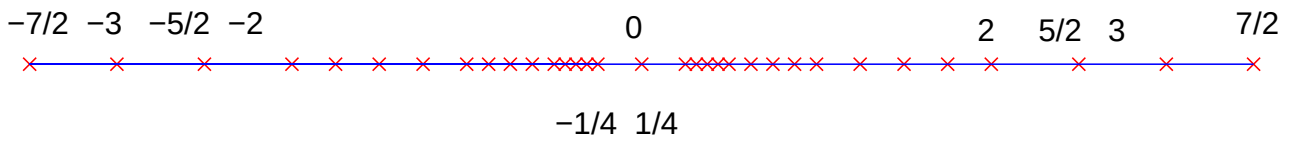
esponente	mantissa	numero
00000000	10...0	2^{-127}
00000000	010...0	2^{-128}
....		
00000000	0...001	2^{-149}

L'insieme dei NUMERI FINITI $F(\beta, t, L, U)$ non è CONTINUO, bensì è un insieme FINITO e LIMITATO.
Possiede esattamente $2(\beta-1)\beta^{t-1}(U-L+1)+1$ elementi: essi sono ugualmente spaziati tra le potenze della base β; infatti tra potenze successive di β ci sono $(\beta-1)\beta^{t-1}$ elementi.

ESEMPIO. Se si considera $\beta = 2, t = 3, L = -2, U = 1$, resta definito un insieme di numeri finiti che possiede 33 elementi in totale. Se si usano le regole di normalizzazione di un reale dell'ANSI 754, tutti gli elementi dell'insieme di numeri finiti definito sono dati dallo zero e da:

$$\pm 1.a_1a_2 \cdot 2^p, \quad p = -2, -1, 0, 1; a_1 = 0, 1; a_2 = 0, 1$$

$m \backslash p$	-2	-1	0	1	
1.00	$\pm 1/4$	$\pm 1/2$	± 1	± 2	
1.01	$\pm 5/16$	$\pm 5/8$	$\pm 5/4$	$\pm 5/2$	$+0$
1.10	$\pm 6/16$	$\pm 3/4$	$\pm 3/2$	± 3	
1.11	$\pm 7/16$	$\pm 7/8$	$\pm 7/4$	$\pm 7/2$	



Attorno a 0 si trova un intervallo $(-\beta^L, \beta^L)$ rappresentato come 0 (underflow). Numeri maggiori di $(2 - \beta^{-t+1})\beta^U$ e minori di $-(2 - \beta^{-t+1})\beta^U$ non sono rappresentabili (overflow).

I numeri più piccoli sono meglio rappresentati.

Di conseguenza, è importante lo scaling dei dati negli algoritmi.

Ogni elemento di F rappresenta se stesso e un intero intervallo di numeri reali.

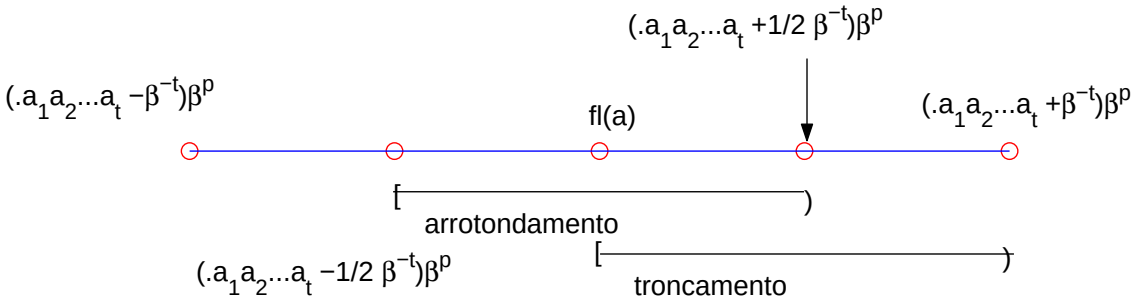
Se $fl(\alpha) = (\sum_{i=1}^{t} a_i \beta^{-i})\beta^p$, allora $fl(\alpha)$ rappresenta l'intervallo

$$[(\sum_{i=1}^{t} a_i \beta^{-i})\beta^p, (\sum_{i=1}^{t} a_i \beta^{-i} + \beta^{-t})\beta^p)$$

nel caso di RAPPRESENTAZIONE PER TRONCAMENTO,

$$[(\sum_{i=1}^{t} a_i \beta^{-i} - \frac{1}{2}\beta^{-t})\beta^p, (\sum_{i=1}^{t} a_i \beta^{-i} + \frac{1}{2}\beta^{-t})\beta^p)$$

nel caso di RAPPRESENTAZIONE PER ARROTONDAMENTO.



Se $\alpha \in F$, $fl(\alpha) = \alpha$.

Se $\alpha \notin F$, $|fl(\alpha) - \alpha| \leq |y - \alpha| \quad \forall y \in F$. $fl(\alpha)$ è il numero di F più prossimo ad α.

TEOREMA SULL'ERRORE DI RAPPRESENTAZIONE.
Sia $\alpha \in \mathbb{R}, \alpha \neq 0$.

$$|\frac{fl(\alpha) - \alpha}{\alpha}| \leq k\beta^{1-t}$$

con $k = 1$ nel caso di troncamento e $k = 1/2$ nel caso di arrotondamento, oppure

$$fl(\alpha) = \alpha(1 + \epsilon) \quad |\epsilon| \leq k\beta^{1-t}$$

.

La quantità $u = k\beta^{1-t}$ si dice PRECISIONE DI MACCHINA e dipende solo da β e da t. Essa è caratteristica del calcolatore che si usa e rappresenta il più piccolo numero sentito dalla macchina relativamente, ossia è caratterizzato dal fatto che

$$fl(1 + u) > 1 \text{ mentre } fl(1 + a) = 1 \; \forall a < u$$

Infatti

$$(.1\beta^1 + k\beta^{1-t}) = \beta(\beta^{-1} + k\beta^{-t}) > 1$$

ESEMPIO. Sia $\beta = 2, t = 3$.
Se $u = \beta^{-2} = 2^{-2} = 0.01$,allora:

$$\begin{aligned} 1 + 0.01 &= 1.01 > 1 \\ 1 + 0.001 &= 1.001, fl(1.001) = 1 \end{aligned}$$

Se $u = 1/2\beta^{-2} = 2^{-3} = 0.001$, allora:

$$\begin{aligned} 1 + 0.001 &= 1.001, fl(1.001) = 1.01 > 1 \\ 1 + 0.0001 &= 1.0001, fl(1.0001) = 1 \end{aligned}$$

Allora ponendo una variabile uguale a 1 e continuando a dimezzarla finchè non viene più sentita nella somma con 1, il ritorno indietro di un passo determina un elemento dell'ordine della precisione di macchina.

Per determinare la base della rappresentazione si osserva che:

$$fl(n+u) > n, n = 0, 1, ..., \beta - 1; \quad fl(\beta + u) = \beta$$

Infatti:

$$\begin{aligned} fl(n+u) &= \beta(n\beta^{-1} + k\beta^{-t}) > n \\ fl(\beta+u) &= \beta(1 + k\beta^{-t}) = \beta \end{aligned}$$

ESEMPIO. Sia $\beta = 10, t = 3$, $u = .510^{-2}$.

$$\begin{aligned} 9 + 0.005 &= 9.005, fl(9.005) = 9.01 > 9 \\ 10 + 0.005 &= 10.005, fl(10.005) = 10 \end{aligned}$$

Infine, per decidere se la macchina lavora per troncamento o per arrotondamento, si osserva che:
dati $\epsilon = \beta^{1-t}$, $\epsilon_1 = \beta^{-t}$,

$$(1+\epsilon) - \epsilon_1 = \begin{array}{ll} \nearrow & > 1 \quad \text{arrotondamento} \quad u = 1/2\epsilon \\ \searrow & = 1 \quad \text{troncamento} \quad u = \epsilon \end{array}$$

Infatti:

$$\begin{aligned} (1+\beta^{1-t}) - \beta^{-t} &= \beta(\beta^{-1} + \beta - t) - \beta^{-t} = \\ &= \beta(\beta^{-1} + \beta^{-t} - \beta^{-t-1}) = \\ &= \beta(\beta^{-1} + \beta^{-t-1}(\beta - 1)) \end{aligned}$$

Allora:

troncamento	$\Rightarrow$	$\beta\beta^{-1} = 1$
arrotondamento	$\Rightarrow$	$\beta(\beta^{-1} + \beta^{-t}) > 1$

ESEMPIO. Sia $\beta = 10, t = 3$, $\epsilon = 10^{-2}$, $\epsilon_1 = 10^{-3}$.

$$\begin{aligned} 1 + 0.01 &= 1.01 \\ 1.01 - 0.001 &= 1.009 \end{aligned}$$

$fl(1.009) = 1$ nel caso di troncamento ed è $= 1.01 > 1$ nel caso di arrotondamento.