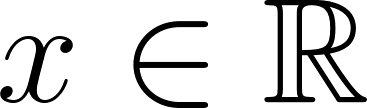
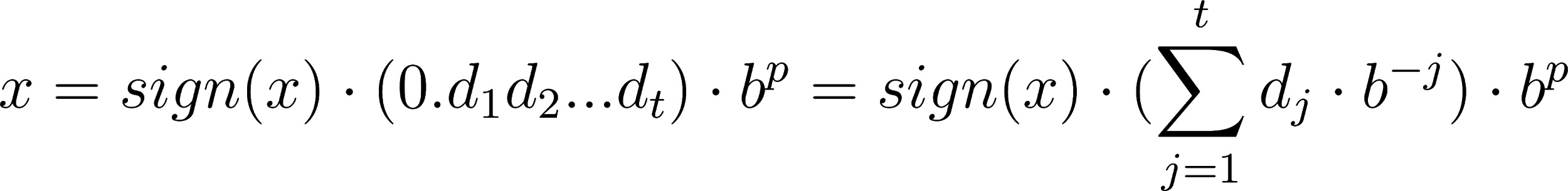
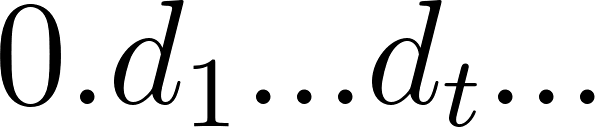
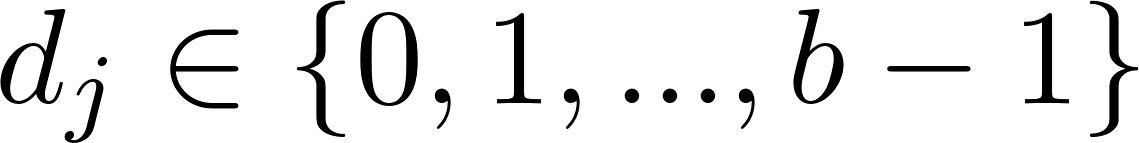
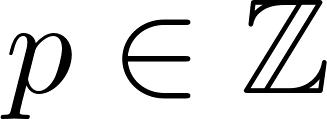
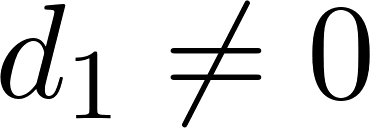
**LEZIONE 2: SISTEMA FLOATING POINT, PRECISIONE DI MACCHINA**

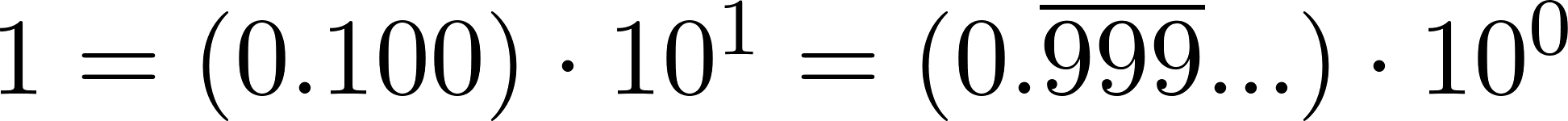
In questa lezione cominceremo ad occuparci dell’effettivo sistema di rappresentazione dei numeri reali del calcolatore, ovvero la rappresentazione “floating-point” (virgola mobile) arrivando a definire l’insieme dei “reali-macchina” e il concetto chiave di “precisione di macchina”.

Cominciamo con l’osservazione che a partire dalla rappresentazione in base b dei reali vista nella lezione 1, è sempre possibile scrivere [](https://www.codecogs.com/eqnedit.php?latex=x%20%5Cin%20%5Cmathbb%7BR%7D%0) in base b come

[](https://www.codecogs.com/eqnedit.php?latex=x%3Dsign(x)%5Ccdot%20(0.d_1d_2...d_t%E2%80%A6)%5Ccdot%20b%5Ep%20%3D%20sign(x)%5Ccdot%20(%5Csum%5Climits_%7Bj%3D1%7D%5Et%20d_j%20%5Ccdot%20b%5E%7B-j%7D)%20%5Ccdot%20b%5Ep%0)

dove p è l’esponente e [](https://www.codecogs.com/eqnedit.php?latex=0.d_1...d_t...%0) la mantissa e [](https://www.codecogs.com/eqnedit.php?latex=d_j%0) con [](https://www.codecogs.com/eqnedit.php?latex=1%3C%3Dj%3C%3D%5Cinfty%0) le cifre della mantissa, [](https://www.codecogs.com/eqnedit.php?latex=d_j%20%5Cin%20%5C%7B0%2C1%2C...%2Cb-1%5C%7D%0) e [](https://www.codecogs.com/eqnedit.php?latex=p%20%5Cin%20%5Cmathbb%7BZ%7D%0) è un intero che viene detto esponente (positivo, nullo o negativo).

Si adotta la convenzione che [](https://www.codecogs.com/eqnedit.php?latex=d_1%20%5Cneq%200%0), altrimenti ci sarebbero infinite rappresentazioni dello stesso numero, ad esempio in base 10 [](https://www.codecogs.com/eqnedit.php?latex=1%3D(0.100%E2%80%A6)%5Ccdot%2010%5E1%3D(0.0100%E2%80%A6)%5Ccdot%2010%5E2%3D(0.00100%E2%80%A6)%5Ccdot%2010%5E3%20...%0)

In questo modo esiste una sola rappresentazione per ogni numero reale (a meno di periodicità sulla cifra massima ad es. [](https://www.codecogs.com/eqnedit.php?latex=1%3D(0.100%E2%80%A6)%5Ccdot%2010%5E1%3D(0.%5Coverline%7B999%7D...)%5Ccdot%2010%5E0%0)).

Questa è quella che si chiama “rappresentazione a virgola mobile” (“floating point” in inglese) del numero. Per capire come funziona basta fare un paio di esempi in base 10

x = +1278.3405….

= (+0.12783405….) \* 104

oppure

x = -0.0003267….

= -(0.3267….) \* 10-3

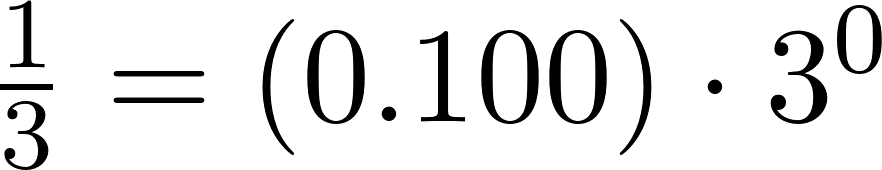
Come si vede semplicemente si sposta la virgola con un’opportuna potenza della base osserviamo che la mantissa sta in (0,1] in particolare appartiene a [0.1,1] (dove 0.1 è inteso in base b, cioè 0.1=1.b-1 e 1 si ottiene con mantissa periodica sulla cifra massima,

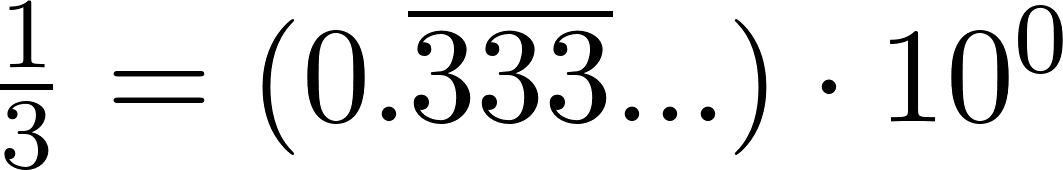
ad es. 0.…=1 in base 10).

Le cifre della mantissa si chiamano “cifre significative” del numero.

E’ importante osservare che la mantissa NON è la parte frazionaria, chi siano parte intera e parte frazionaria è determinato dall’esponente che sposta la virgola tramite la potenza bp (a destra per p>0 e a sinistra per p<0). E’ chiaro che in generale un reale ha mantissa infinita, in particolare gli irrazionali hanno mantissa infinita in qualsiasi base (perchè hanno parte frazionaria infinita). I numeri con mantissa finita sono invece razionali; che un razionale abbia mantissa finita o meno dipende dalla base,

come abbiamo già visto, ad esempio

[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B1%7D%7B3%7D%20%3D%20(0.100%E2%80%A6)%5Ccdot%203%5E0%0) in base 3

[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B1%7D%7B3%7D%3D(0.%5Coverline%7B333%7D...)%5Ccdot%2010%5E0%0) in base 10

A questo punto siamo in grado di definire l’insieme dei “reali-macchina”, che in un sistema di calcolo con una macchina che lavora in base “b” sono i numeri con una quantità finita di cifre di mantissa e l'esponente che varia in un intervallo finito di interi.

Formalmente l’insieme dei reali-macchina è un’insieme di razionali definiti da 4 parametri b (la base), t (il numero di cifre della mantissa), L (Lower), U (Upper) che sono gli estremi dell’intervallo di esponenti interi, dove L < 0 e U > 0

insieme dei numeri macchina

↓

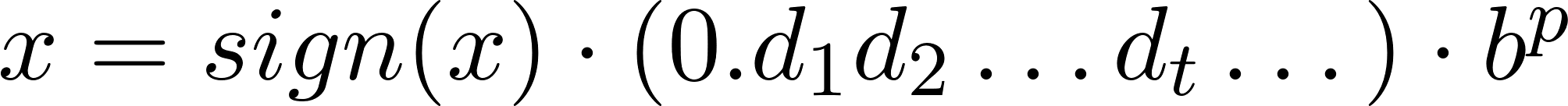
𝐅(b, t, L, U) = { μ ∈ ℚ : μ = sgn(μ)(0.μ1μ2….μt)\*bp, μj ∈ { 0, 1, 2, …., b - 1 }, μ1 ≠ 0, p ∈ [L, U] ⊂ 𝕫 }

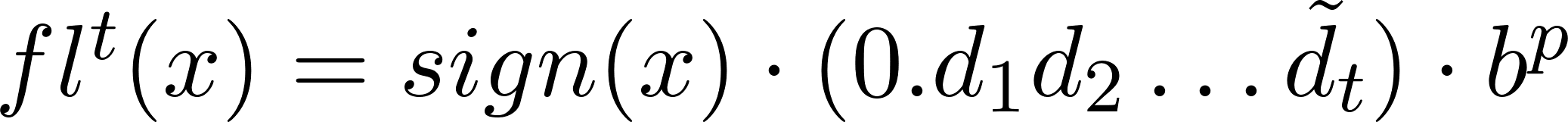
Il fatto che il numero di cifre di mantissa sia finito e l’intervallo di esponenti sia finito, permette la memorizzazione utilizzando sequenze di bit (e tipicamente base b=2).

Per avere un’idea (poi faremo un modellino) con variabili (zone di memoria) a 64 bit per i reali-macchina (come accade ad es. in Matlab) si ha b=2, t=53 e in modello semplificato L=-1023, U=+1023

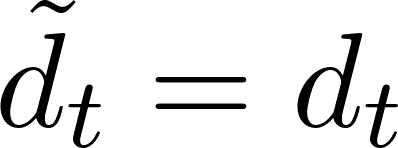
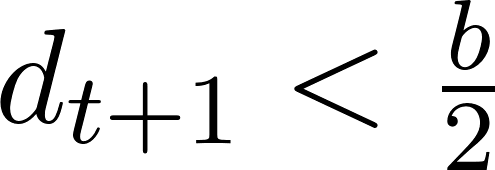
In una parte successiva andremo a studiare più in dettaglio la struttura dell’insieme dei reali-macchina. Qui invece facciamo il passo fondamentale che ci permette di studiare l’ERRORE che si commette approssimando un numero reale con un reale-macchina. Questa approssimazione avviene per arrotondamento della mantissa al numero di cifre disponibili.

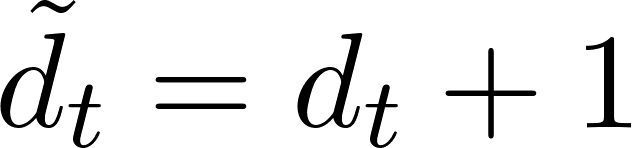
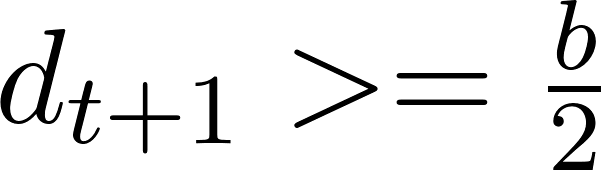
Definiamo arrotondamento a t cifre un numero reale scritto in notazione floating-point

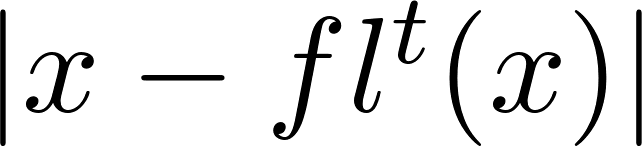
[](https://www.codecogs.com/eqnedit.php?latex=x%3Dsign(x)%5Ccdot%20(0.d_1d_2%5Cdotso%20d_t%5Cdotso)%5Ccdot%20b%5Ep%0)

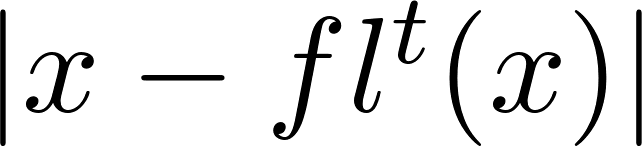
il numero [](https://www.codecogs.com/eqnedit.php?latex=fl%5Et(x)%3Dsign(x)%5Ccdot%20(0.d_1d_2%20%5Cdotso%20%5Ctilde%7Bd_t%7D)%5Ccdot%20b%5Ep%0)

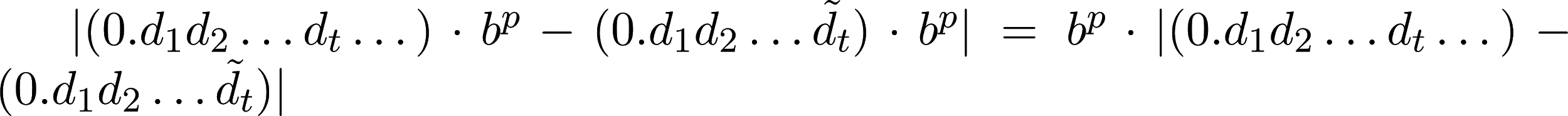
dove la mantissa è stata arrotondata alla t-esima cifra

[](https://www.codecogs.com/eqnedit.php?latex=%5Ctilde%7Bd_t%7D%3Dd_t%0) se [](https://www.codecogs.com/eqnedit.php?latex=d_%7Bt%2B1%7D%3C%5Cfrac%7Bb%7D%7B2%7D%0)

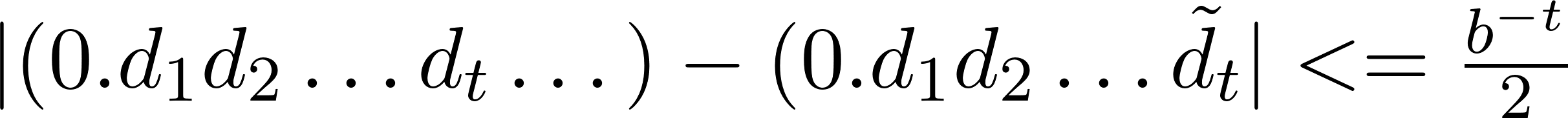
[](https://www.codecogs.com/eqnedit.php?latex=%5Ctilde%7Bd_t%7D%3Dd_t%2B1%0) se [](https://www.codecogs.com/eqnedit.php?latex=d_%7Bt%2B1%7D%20%3E%3D%20%5Cfrac%7Bb%7D%7B2%7D%0)

si tratta a questo punto di stimare se l’errore [](https://www.codecogs.com/eqnedit.php?latex=%7Cx-fl%5Et(x)%7C%0)

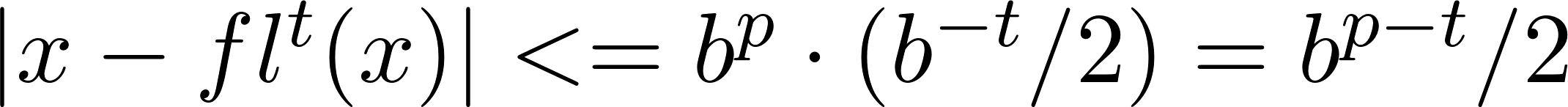
Ora, [](https://www.codecogs.com/eqnedit.php?latex=%7Cx-fl%5Et(x)%7C%0) è uguale a

[](https://www.codecogs.com/eqnedit.php?latex=%7C(0.d_1d_2%20%5Cdotso%20d_t%20%5Cdotso%20)%5Ccdot%20b%5Ep%20-%20(0.d_1d_2%20%5Cdotso%20%5Ctilde%7Bd_t%7D)%5Ccdot%20b%5Ep%7C%20%3D%20b%5Ep%5Ccdot%20%7C(0.d_1d_2%20%5Cdotso%20d_t%20%5Cdotso%20)%20-%20(0.d_1d_2%20%5Cdotso%20%5Ctilde%7Bd_t%7D)%7C%0)

ma abbiamo già stimato l’errore di arrotondamento a n cifre “dopo la virgola”, che è <=b-n/2 quindi siamo in grado di stimare la differenza in modulo tra mantissa esatta e mantissa arrotondata (posto n=t)

[](https://www.codecogs.com/eqnedit.php?latex=%7C(0.d_1d_2%20%5Cdotso%20d_t%20%5Cdotso)%20-%20(0.d_1d_2%20%5Cdotso%20%5Ctilde%7Bd_t%7D%7C%3C%3D%5Cfrac%7Bb%5E%7B-t%7D%7D%7B2%7D%0)

da cui ricaviamo

[](https://www.codecogs.com/eqnedit.php?latex=%7Cx-fl%5Et(x)%7C%20%3C%3D%20b%5Ep%20%5Ccdot%20(b%5E%7B-t%7D%2F2)%20%3D%20b%5E%7Bp-t%7D%2F2%0)

notiamo subito un aspetto: l’errore dipende da p, cioè dall’ordine di grandezza del numero (in base b).

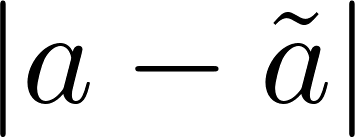
Numeri grandi in modulo (p positivo grande) avranno errori grandi, numeri piccoli in modulo (p negativo e grande in modulo) avranno errori piccoli.

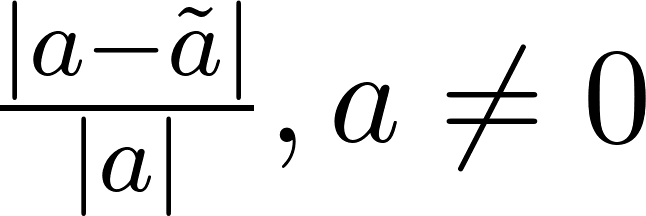
Gli errori diventano addirittura grandissimi o piccolissimi avvicinandosi agli estremi L ed U dell’intervallo di esponenti.

La domanda è: è accettabile una situazione di questo tipo, ovvero un errore variabile con l’ordine di grandezza del numero, nel modo descritto?

La riposta è SÌ, se ci spostiamo dall’ERRORE ASSOLUTO (che è quello stimato finora) al concetto di ERRORE RELATIVO.

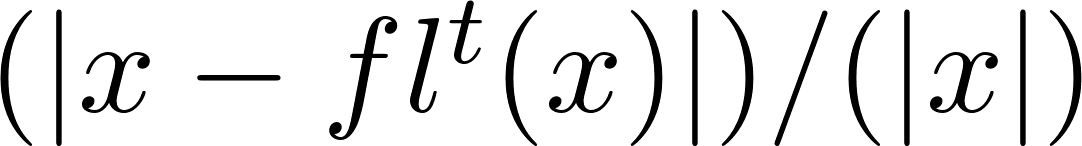
Dati due reali a e ã con ã ≈ a (ã è approssimazione di a)

[](https://www.codecogs.com/eqnedit.php?latex=%7Ca-%5Ctilde%7Ba%7D%7C%0) ERRORE ASSOLUTO

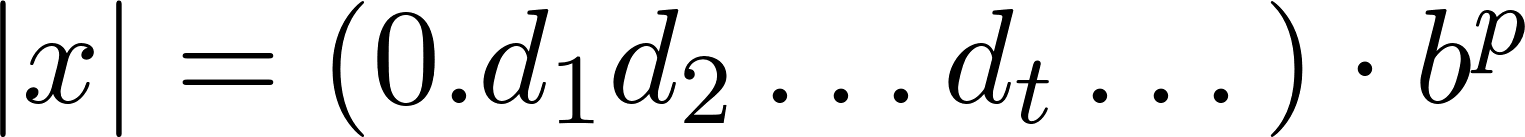
[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B%7Ca-%5Ctilde%7Ba%7D%7C%7D%7B%7Ca%7C%7D%2C%20a%5Cneq%200%0) ERRORE RELATIVO

l’errore relativo è l’errore assoluto su una quantità, pesato dalla grandezza della quantità.

È l’errore più importante in campo sperimentale e nelle applicazioni pratiche, l’errore che di solito si esprime in percentuale: ad esempio una certa quantità è nota con un errore del 10%(errore relativo 10-1), oppure dell’1% (errore relativo 10-2),oppure dello 0,01%(errore relativo 10-4).

Nel nostro caso dell’errore di arrotondamento a t cifre di mantissa, scopriremo che l’errore relativo non dipende più da p (cioè dall’ordine di grandezza del numero nella base fissata) quindi ci interessa ora stimare [](https://www.codecogs.com/eqnedit.php?latex=(%7Cx-fl%5Et(x)%7C)%2F(%7Cx%7C)%0) per x!=0

Abbiamo già stimato il numeratore(che è l’errore assoluto di arrotondamento a t cifre di mantissa), dobbiamo quindi stimare (da sopra) per fare questo si stima da sotto |x| infatti, se allora cerchiamo quindi un tale α

[](https://www.codecogs.com/eqnedit.php?latex=%7Cx%7C%20%3D%20(0.d_1d_2%20%5Cdotso%20d_t%20%5Cdotso)%5Ccdot%20b%5Ep%0)

dove d1≠ 0

fissato p, qual è il valore più piccolo possibile di |x|?

Corrisponde alla mantissa minima che è 0.100…=b-1 quindi otteniamo

La quantità 𝜀M , che si chiama PRECISIONE DI MACCHINA è il parametro chiave del sistema floating-point, inteso come insieme dei reali-macchina e della sua capacità di approssimazione. In sintesi, possiamo dire che la precisione di macchina è il massimo errore relativo di arrotondamento t” cifre di mantissa e dipende solo da “b” e da “t”, o in altre parole è la massima distanza relativa dal reale-macchina più vicino, distanza relativa che non dipende dall’ordine di grandezza del numero come vedremo, non tutti i reali sono approssimabili con un reale-macchina, perché il range di esponenti è finito ciononostante, su ogni reale approssimabile si fa un errore che non supera la precisione di macchina.

Nel sistema floating-point a 64bit è una quantità piccolissima (≈10-16).

Osserviamo qui che useremo il simbolo “≈”, che significa “circa uguale”, sia nel senso di approssimazione con errore “piccolo” sia nel senso di approssimazione dell’ordine di grandezza (come l’abbiamo appena usata qui sopra, per indicare che è dell’ordine di 10-16) nella prossima lezione studieremo la struttura dell’insieme 𝐅(b,t,L,U).