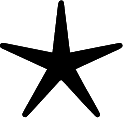
**LEZIONE 4: OPERAZIONI ARITMETICHE CON NUMERI APPROSSIMATI**

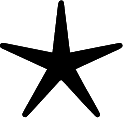
Nelle lezioni precedenti abbiamo discusso una delle basi del calcolo numerico: il sistema floating-point di rappresentazione dei numeri reali nel calcolatore. Il concetto cardine è quello di precisione di macchina, il massimo errore relativo di arrotondamento di un numero reale rappresentabile (ovvero non in overflow o underflow) con il reale-macchina “più vicino”.

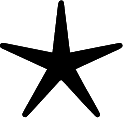
In questa lezione inizieremo ad entrare nel cuore del calcolo numerico, trattando innanzitutto le operazioni aritmetiche nel sistema floating-point, cioè quale sia l’effetto degli errori sui dati, sul risultato dell’operazione.

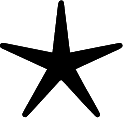
L’analisi che faremo sarà però più generale e permetterà di studiare la risposta agli errori sui dati delle operazioni aritmetiche qualunque sia la fonte di errore (ad esempio errori di misura sperimentale).

Cominciamo col definire il concetto di OPERAZIONE-MACCHINA.

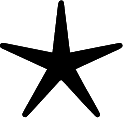
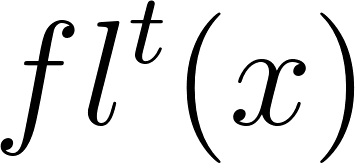
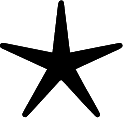
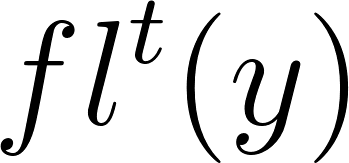
Sia [](http://www.texrendr.com/?eqn=%5Cstar#0) un’operazione aritmetica sui reali, ovvero

[](http://www.texrendr.com/?eqn=%5Cstar#0)= ± addizione,sottrazione

[](http://www.texrendr.com/?eqn=%5Cstar#0)= \* moltiplicazione

[](http://www.texrendr.com/?eqn=%5Cstar#0)= / divisione

Allora, dato un insieme di reali-macchina il modo in cui il processore realizza l’operazione tra due reali rappresentabili x,y è

x [](http://www.texrendr.com/?eqn=%5Cstar#0)y= [](https://www.codecogs.com/eqnedit.php?latex=fl%5Et(x)#0)[](http://www.texrendr.com/?eqn=%5Cstar#0)[](https://www.codecogs.com/eqnedit.php?latex=fl%5Et(y)#0)

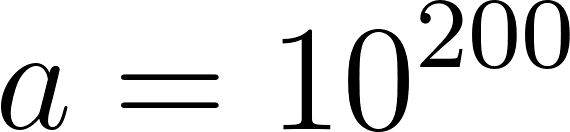
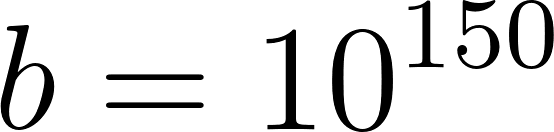
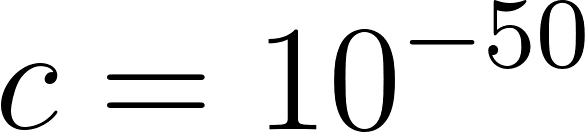
ovvero il modello è il seguente:

1. i due reali vengono arrotondati
2. viene fatta l’operazione tra gli arrotondamenti
3. il risultato viene a sua volta arrotondato

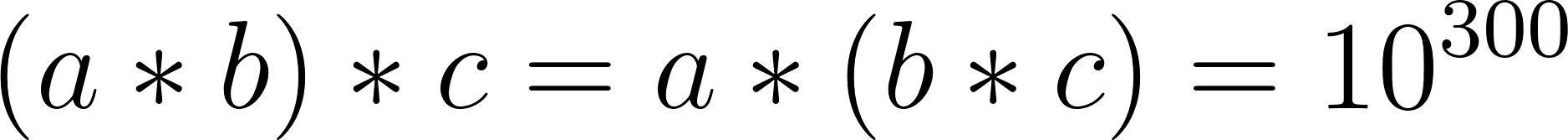
È importante osservare da subito che le operazioni-macchina nel sistema floating-point perdono varie proprietà delle operazioni aritmetiche teoriche, ad esempio, mentre la proprietà commutativa di addizione e moltiplicazione resta valida, in generale non sono più valide la proprietà associativa e distributiva.

Facciamo un esempio in cui non vale la proprietà associativa della moltiplicazione, per problemi di overflow.

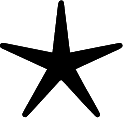
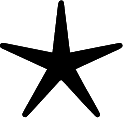
Consideriamo =(10,16,-308,308) che abbiamo visto corrispondere sostanzialmente all’interfaccia del Matlab.

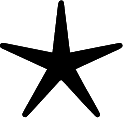
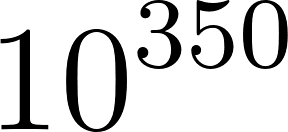
Siano [](https://www.codecogs.com/eqnedit.php?latex=a%3D10%5E%7B200%7D#0), [](https://www.codecogs.com/eqnedit.php?latex=b%3D10%5E%7B150%7D#0), [](https://www.codecogs.com/eqnedit.php?latex=c%3D10%5E%7B-50%7D#0)

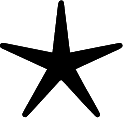
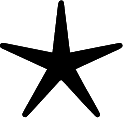
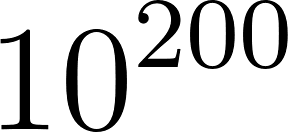
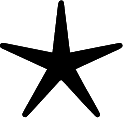
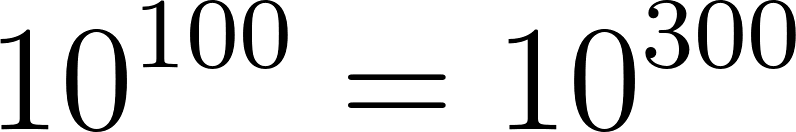
in aritmetica reale

[](https://www.codecogs.com/eqnedit.php?latex=(a*b)*c%3Da*(b*c)%3D10%5E%7B300%7D#0)

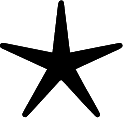
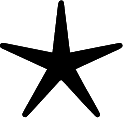
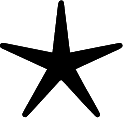
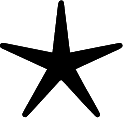
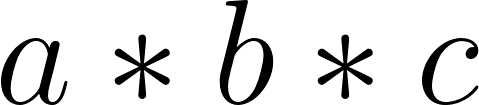
Ma con le limitazioni date per gli esponenti

(a[](http://www.texrendr.com/?eqn=%5Cstar#0)b)[](http://www.texrendr.com/?eqn=%5Cstar#0)c=overflow

perché a[](http://www.texrendr.com/?eqn=%5Cstar#0)b=[](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B350%7D#0) e 350>308

invece a[](http://www.texrendr.com/?eqn=%5Cstar#0)(b[](http://www.texrendr.com/?eqn=%5Cstar#0)c)=[](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B200%7D#0)[](http://www.texrendr.com/?eqn=%5Cstar#0)[](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B100%7D%3D10%5E%7B300%7D#0)

Quindi

(a[](http://www.texrendr.com/?eqn=%5Cstar#0)b)[](http://www.texrendr.com/?eqn=%5Cstar#0)c ≠ a[](http://www.texrendr.com/?eqn=%5Cstar#0)(b[](http://www.texrendr.com/?eqn=%5Cstar#0)c)= [](https://www.codecogs.com/eqnedit.php?latex=a*b*c#0)

^non associativa

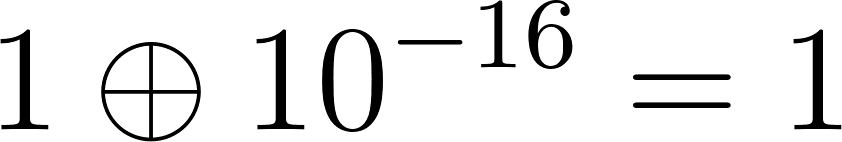
Come vedremo più avanti, la proprietà associativa (e anche la distributiva) possono saltare anche per effetto degli errori di arrotondamento.

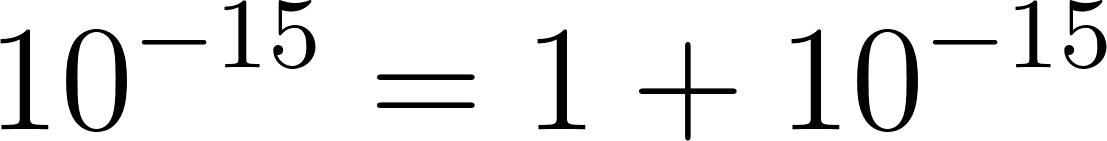
D’altra parte, c’è un un’altra proprietà che non è più valida a causa dell’arrotondamento, l’unicità dell’elemento neutro dell’addizione.

Per capirlo, consideriamo di nuovo un sistema floating-point “virtuale” =(10,16,-308,308)

(qui in realtà conta essenzialmente il numero di cifre mantissa).

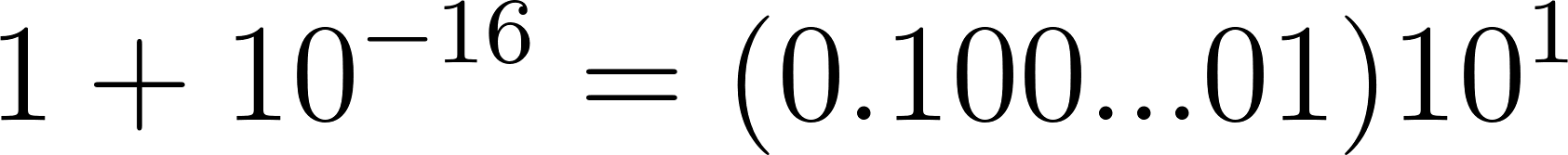
Si ha

[](https://www.codecogs.com/eqnedit.php?latex=1%20%5Coplus%2010%5E%7B-16%7D%20%3D%201#0) cioè 10-16 si comporta come 0 nell’addizione con 1, mentre

1⊕[](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B-15%7D%3D1%2B10%5E%7B-15%7D#0)

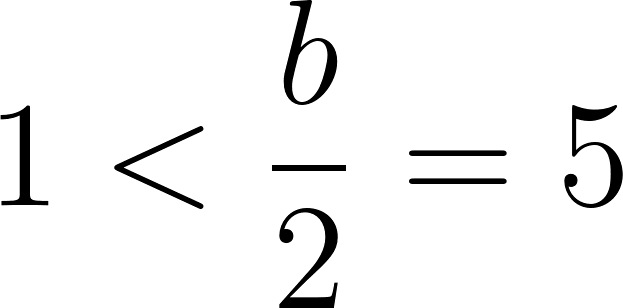
Perchè?

Basta riflettere sul fatto che le cifre di mantissa sono 16, ma

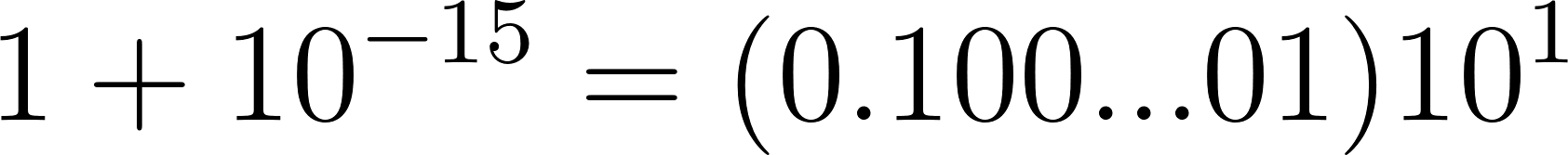
[](https://www.codecogs.com/eqnedit.php?latex=1%2B10%5E%7B-16%7D%3D(0.100...01)10%5E%7B1%7D#0)

**17° cifra** ↑

viene arrotondato ad 1 perché la prima cifra trascurata è

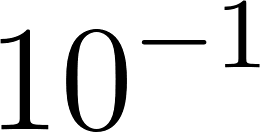
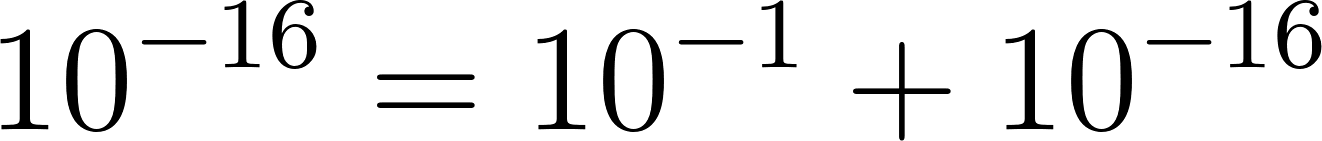
[](https://www.codecogs.com/eqnedit.php?latex=1%3C%20%5Cfrac%7Bb%7D%7B2%7D%3D5#0)

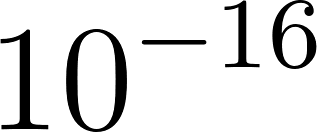
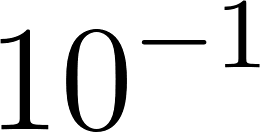
invece

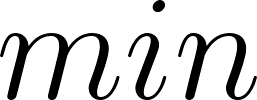
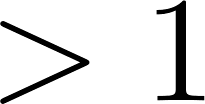
[](https://www.codecogs.com/eqnedit.php?latex=1%2B10%5E%7B-15%7D%3D(0.100...01)10%5E%7B1%7D#0)

**16° cifra** ↑

e non c’è bisogno di alcun arrotondamento.

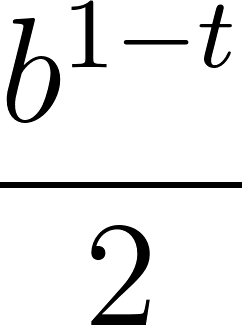
D’altra parte, si vede anche subito che [](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B-1%7D#0)⊕[](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B-16%7D%3D10%5E%7B-1%7D%2B10%5E%7B-16%7D#0)

Cioè [](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B-16%7D#0) non è l’elemento neutro per [](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B-1%7D#0). In questo contesto si può dare una seconda caratterizzazione della precisione di macchina (che non dimostriamo)

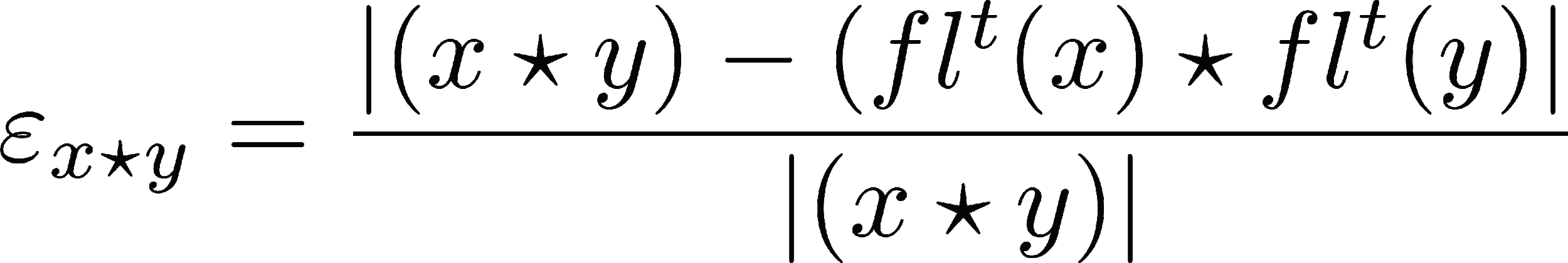
εM=[](https://www.codecogs.com/eqnedit.php?latex=min%7B#0){μ ∈ [](https://www.codecogs.com/eqnedit.php?latex=F%5E%7B%2B%7D#0) :1⊕μ [](https://www.codecogs.com/eqnedit.php?latex=%CE%BC%3E1%7D#0)}

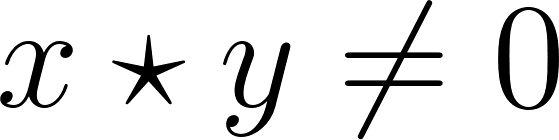
Passiamo alla questione chiave: la risposta delle operazioni agli errori sui dati.

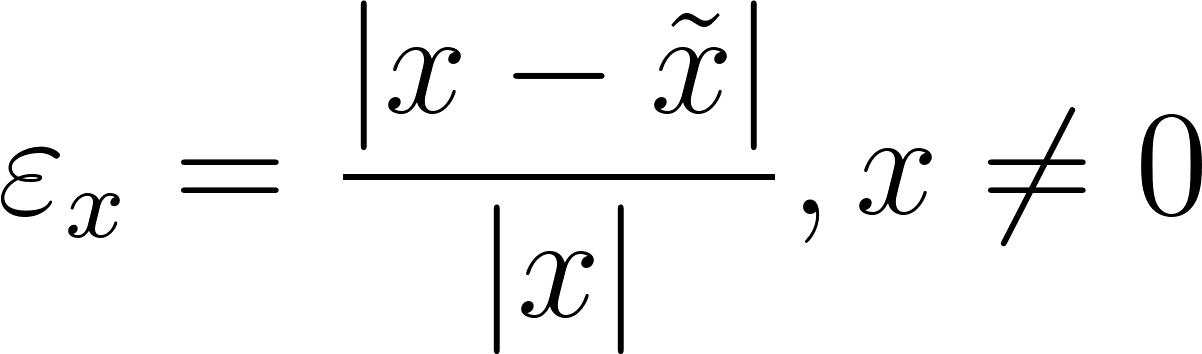
Possiamo osservare fin da subito che l’arrotondamento finale del risultato ha un errore che non può superare la precisione di macchina

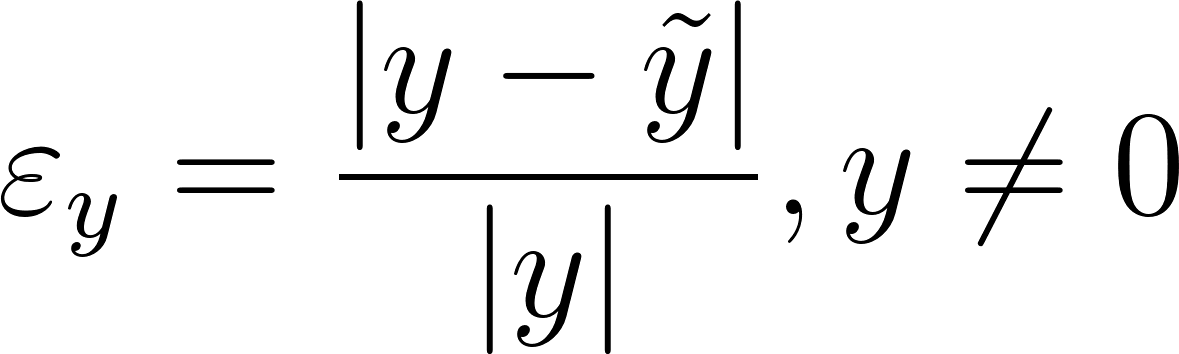
εM=[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7Bb%5E%7B1-t%7D%7D%7B2%7D#0)

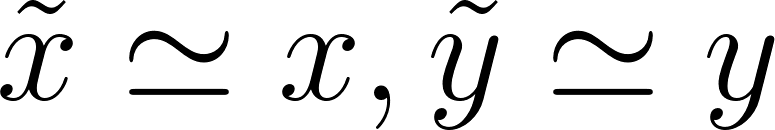
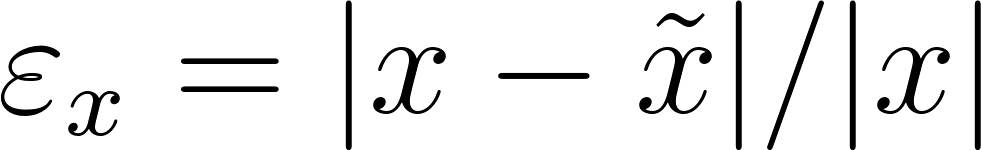
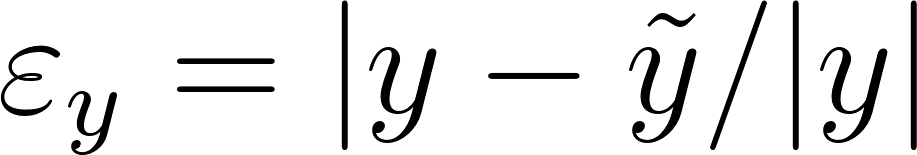
e quindi è ininfluente ai fini dell’analisi dell’effetto degli errori sui dati. Invece l’errore chiave da stimare è l’errore relativo

[](http://www.texrendr.com/?eqn=%5Cvarepsilon_%7Bx%5Cstar%20y%7D%20%3D%20%5Cfrac%7B%7C(x%20%5Cstar%20y)%20-%20(fl%5Et(x)%20%5Cstar%20fl%5Et(y)%7C%7D%7B%7C(x%20%5Cstar%20y)%7C%7D#0)

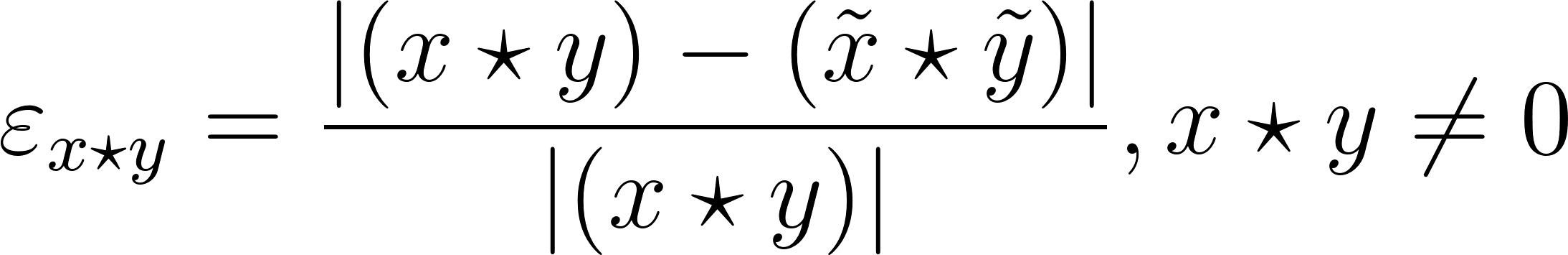
(purché [](https://www.codecogs.com/eqnedit.php?latex=x%20%5Cstar%20y%20%5Cneq%200#0)) in funzione degli errori relativi sui dati

[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_x%20%3D%20%5Cfrac%7B%7Cx-%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%7C%7D%2C%20x%20%5Cneq%200#0)

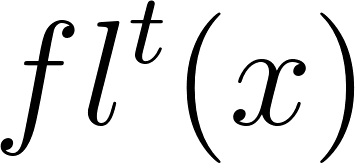
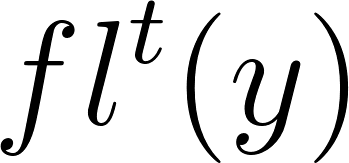
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_y%20%3D%20%5Cfrac%7B%7Cy-%5Ctilde%7By%7D%7C%7D%7B%7Cy%7C%7D%2C%20y%20%5Cneq%200#0)

più in generale, dati due numeri reali x, y ≠ 0 e due loro approssimazioni [](https://www.codecogs.com/eqnedit.php?latex=%5Ctilde%7Bx%7D%20%5Csimeq%20x%2C%20%5Ctilde%7By%7D%20%5Csimeq%20y#0) (dove supponiamo di saper stimare gli errori relativi [](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_x%20%3D%20%7Cx-%5Ctilde%7Bx%7D%7C%2F%7Cx%7C#0) e [](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_y%20%3D%20%7Cy-%5Ctilde%7By%7D%2F%7Cy%7C#0)) andiamo a stimare l’errore relativo sul risultato dell’operazione [](https://www.codecogs.com/eqnedit.php?latex=%5Cstar#0), commesso.

Utilizzando i dati approssimati invece dei dati esatti

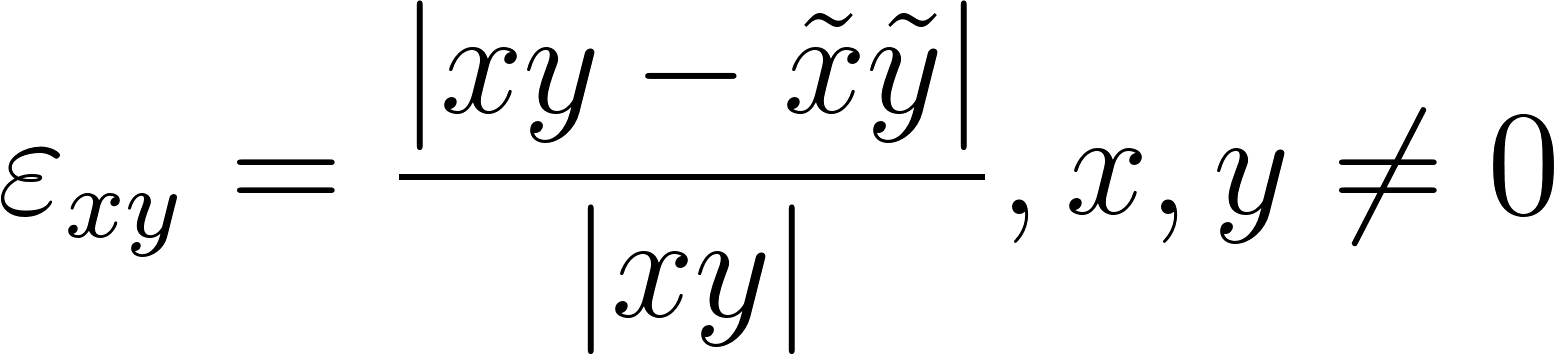
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon%20_%7Bx%5Cstar%20y%7D%3D%5Cfrac%7B%7C(x%5Cstar%20y)-(%5Ctilde%7Bx%7D%5Cstar%5Ctilde%7By%7D)%7C%7D%7B%7C(x%5Cstar%20y)%7C%7D%2C%20x%5Cstar%20y%5Cneq%200#0)

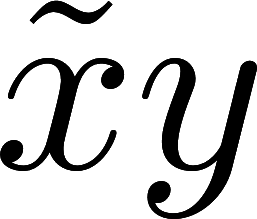
in funzione di εx, εy (nel caso delle operazioni macchina si ha

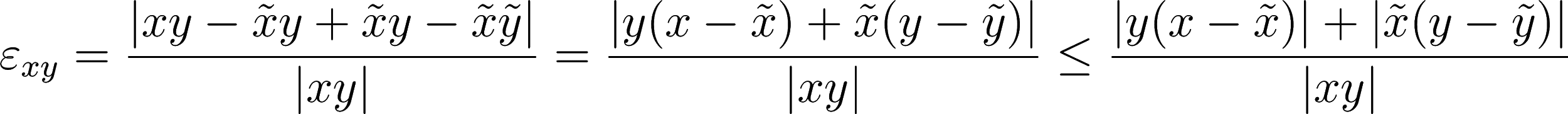
x^~=[](https://www.codecogs.com/eqnedit.php?latex=fl%5E%7Bt%7D(x)#0), ỹ=[](https://www.codecogs.com/eqnedit.php?latex=fl%5E%7Bt%7D(y)#0) εx ,εy ≤ εM )

diremo STABILE un’operazione aritmetica per cui l’errore sul risultato ha lo stesso ordine di grandezza dell’errore (massimo) sui dati.

Iniziamo con la moltiplicazione (indichiamo il prodotto con la notazione standard xy, sapendo che nei linguaggi di calcolo il simbolo delle moltiplicazione è \*)

[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bxy%7D%20%3D%20%5Cfrac%7B%7Cxy-%5Ctilde%7Bx%7D%5Ctilde%7By%7D%7C%7D%7B%7Cxy%7C%7D%2C%20x%2Cy%20%5Cneq%200#0)

Usiamo la stessa tecnica che si usa per dimostrare che il limite del prodotto di due successioni o funzioni è il prodotto dei limiti, aggiungendo e togliendo a numeratore ad esempio [](http://www.texrendr.com/?eqn=%5Ctilde%7Bx%7Dy#0)

[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon%20_%7Bxy%7D%3D%5Cfrac%7B%7Cxy-%5Ctilde%7Bx%7Dy%2B%5Ctilde%7Bx%7Dy-%5Ctilde%7Bx%7D%5Ctilde%7By%7D%7C%7D%7B%7Cxy%7C%7D%3D%5Cfrac%7B%7Cy(x-%5Ctilde%7Bx%7D)%2B%5Ctilde%7Bx%7D(y-%5Ctilde%7By%7D)%7C%7D%7B%7Cxy%7C%7D%20%5Cleq%20%5Cfrac%7B%7Cy(x-%5Ctilde%7Bx%7D)%7C%2B%7C%5Ctilde%7Bx%7D(y-%5Ctilde%7By%7D)%7C%7D%7B%7Cxy%7C%7D#0) \*

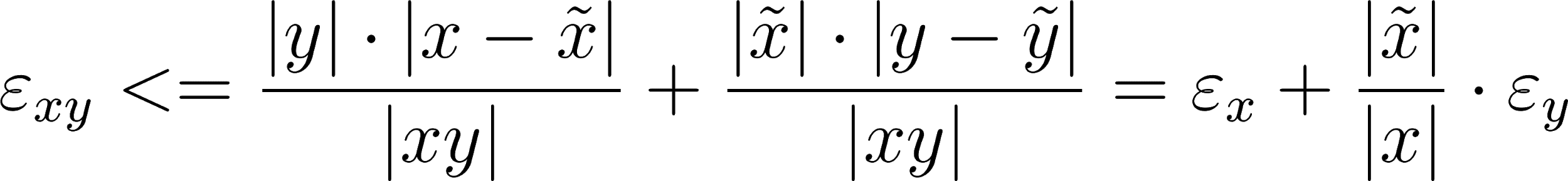
dove abbiamo utilizzato la DISUGUAGLIANZA TRIANGOLARE che è uno strumento chiave per fare stime.

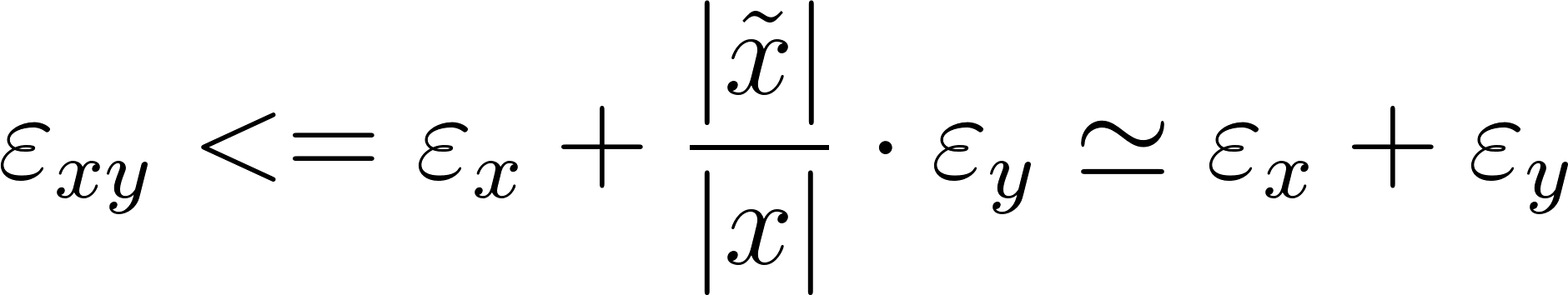
Dati a,b appartengono a R

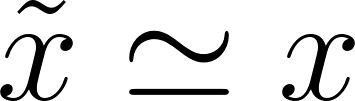
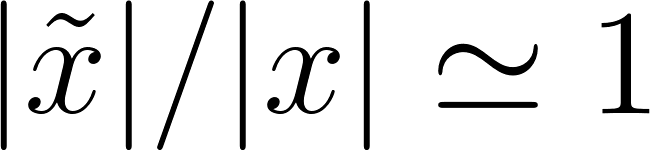
| |a|-|b| | ≤ |a+b| ≤ |a| + |b|

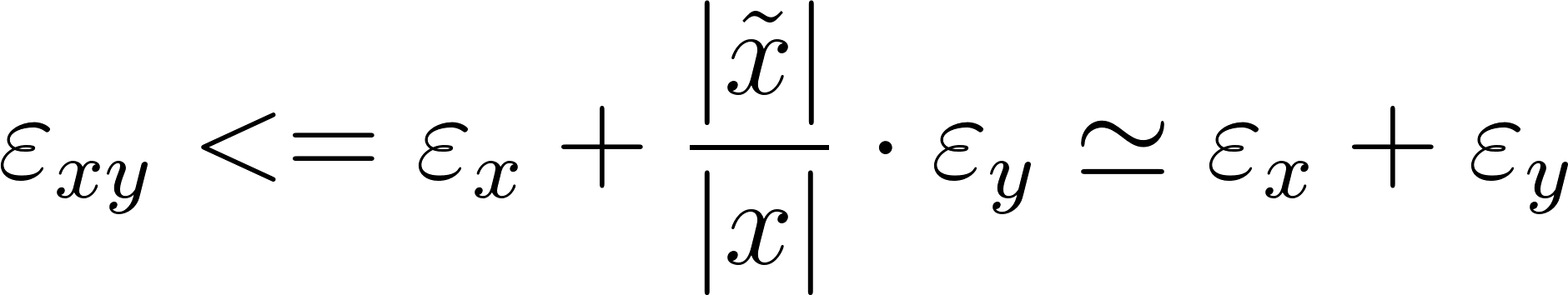
↑ sopra abbiamo usato

quindi da \* otteniamo (ricordando che il modulo del prodotto è il prodotto dei moduli)

[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bxy%7D%20%3C%3D%20%5Cfrac%7B%7Cy%7C%20%5Ccdot%20%7Cx-%5Ctilde%7Bx%7D%7C%7D%7B%7Cxy%7C%7D%20%2B%20%5Cfrac%7B%7C%5Ctilde%7Bx%7D%7C%5Ccdot%20%7Cy-%5Ctilde%7By%7D%7C%7D%7B%7Cxy%7C%7D%20%3D%20%5Cvarepsilon_x%20%2B%20%5Cfrac%7B%7C%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%7C%7D%5Ccdot%20%5Cvarepsilon_y#0)

[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bxy%7D%20%3C%3D%20%5Cvarepsilon_x%20%2B%20%5Cfrac%7B%7C%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%7C%7D%5Ccdot%20%5Cvarepsilon_y%20%5Csimeq%20%5Cvarepsilon_x%20%2B%20%5Cvarepsilon_y#0)

ora siccome [](https://www.codecogs.com/eqnedit.php?latex=%5Ctilde%7Bx%7D%20%5Csimeq%20x#0) possiamo dire almeno qualitativamente che [](https://www.codecogs.com/eqnedit.php?latex=%7C%5Ctilde%7Bx%7D%7C%20%2F%20%7Cx%7C%20%5Csimeq%201#0) e quindi

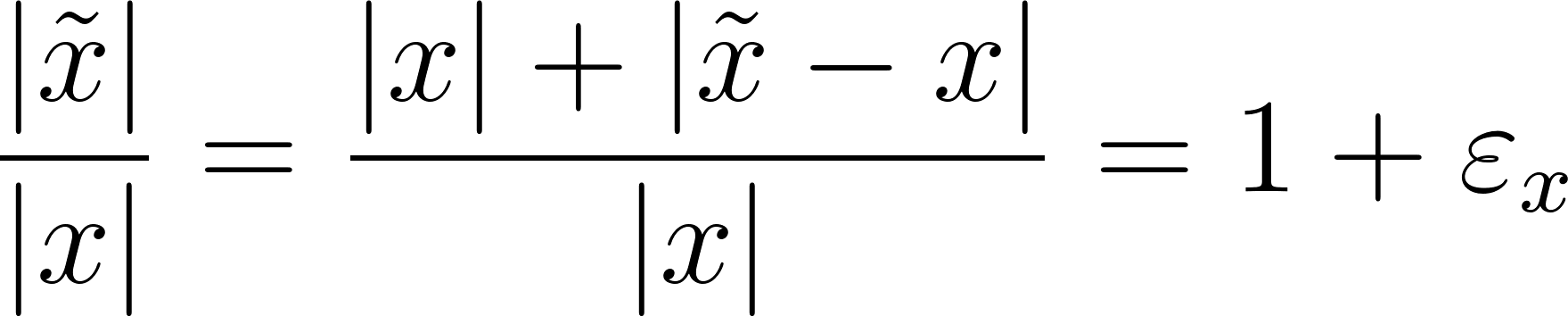
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bxy%7D%20%3C%3D%20%5Cvarepsilon_x%20%2B%20%5Cfrac%7B%7C%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%7C%7D%20%5Ccdot%20%5Cvarepsilon_y%20%5Csimeq%20%5Cvarepsilon_x%20%2B%20%5Cvarepsilon_y#0)

cioè che l’operazione di moltiplicazione è **stabile** (l’errore relativo sul risultato è maggiorato da una quantità che è dell’ordine dell’errore sui dati).

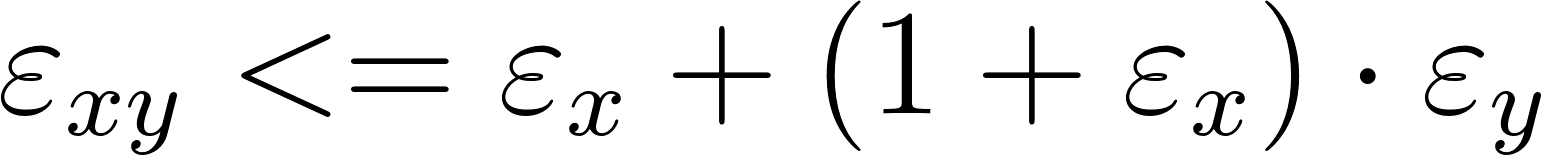
Per esprimere questo fatto possiamo usare la notazione

εxy ≲ εx+εy

dove ≲ non è una disuguaglianza esatta ma va intesa nel senso indicato sopra. In realtà possiamo dare anche una stima quantitativa osservando che per la disuguaglianza triangolare

[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B%7C%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%7C%7D%20%3D%20%5Cfrac%7B%7Cx%7C%2B%7C%5Ctilde%7Bx%7D-x%7C%7D%7B%7Cx%7C%7D%3D1%2B%5Cvarepsilon_x#0)

e quindi

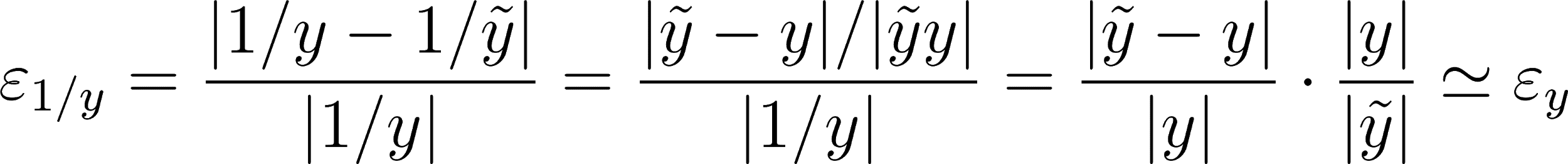
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bxy%7D%20%3C%3D%20%5Cvarepsilon_x%20%2B%20(1%2B%5Cvarepsilon_x)%5Ccdot%20%5Cvarepsilon_y#0)

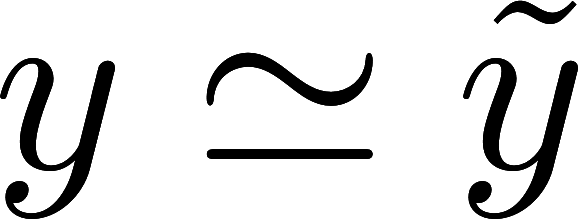
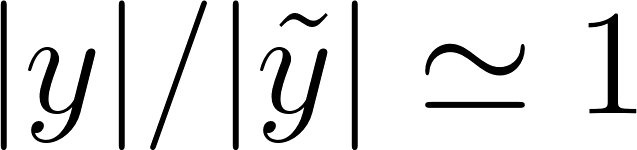
Nel caso della moltiplicazione macchina in precisione doppia εx≤εM circa 10-16

e quindi 1+εx è vicinissimo ad 1.

Ma anche con εx ≈ 10-1 (ad esempio 10% è un errore di misura sperimentale grande) avremmo che εx ≈ 1.1 e quindi la sostanza della stabilità non cambia.

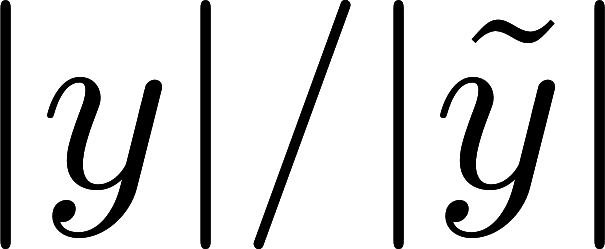
Passiamo ora alla DIVISIONE: siccome la divisione , y≠0 è la moltiplicazione per il reciproco, = x, ci basta analizzare la stabilità dell’operazione di reciproco.

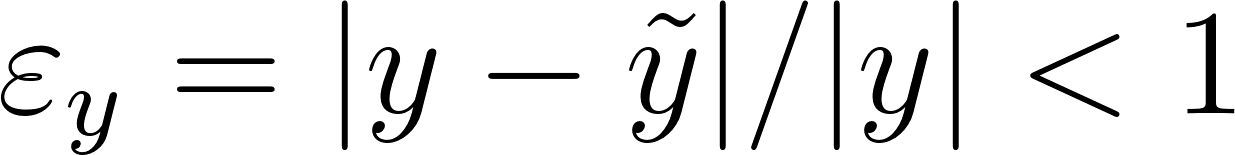
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7B1%2Fy%7D%20%3D%20%5Cfrac%7B%7C1%2Fy%20-%201%2F%5Ctilde%7By%7D%7C%7D%7B%7C1%2Fy%7C%7D%20%3D%20%5Cfrac%7B%7C%5Ctilde%7By%7D-y%7C%20%2F%20%7C%5Ctilde%7By%7Dy%7C%7D%7B%7C1%2Fy%7C%7D%20%3D%20%5Cfrac%7B%7C%5Ctilde%7By%7D-y%7C%7D%7B%7Cy%7C%7D%20%5Ccdot%20%5Cfrac%7B%7Cy%7C%7D%7B%7C%5Ctilde%7By%7D%7C%7D%20%5Csimeq%20%5Cvarepsilon_y#0)

con l’ipotesi qualitativa che [](http://www.texrendr.com/?eqn=y%20%5Csimeq%20%5Ctilde%7By%7D#0) e quindi [](https://www.codecogs.com/eqnedit.php?latex=%7Cy%7C%20%2F%20%7C%5Ctilde%7By%7D%7C%20%5Csimeq%201#0)

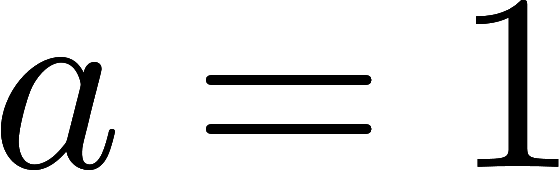
ma deduciamo che anche che la DIVISIONE è un’operazione STABILE

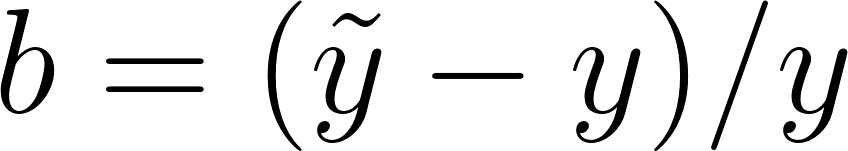
perchè il reciproco è stabile e la moltiplicazione è stabile.

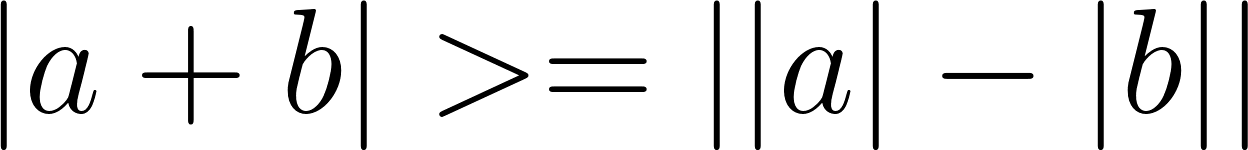
Anche in questo caso possiamo però quantificare, stimando meglio [](http://www.texrendr.com/?eqn=%7Cy%7C%20%2F%20%7C%5Ctilde%7By%7D%7C#0)

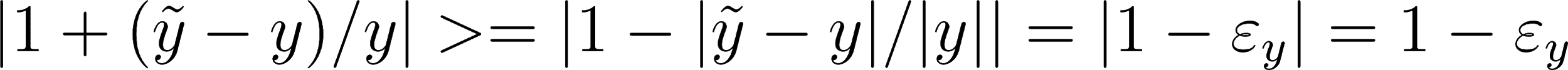
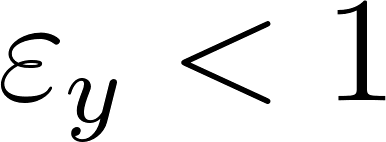
Assumiamo [](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_y%20%3D%20%7Cy-%5Ctilde%7By%7D%7C%20%2F%20%7Cy%7C%20%3C%201#0)

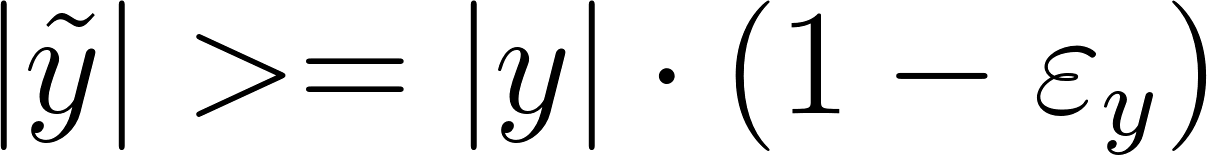
(cioè che l’errore relativo sia minore di 1, il che vuol dire < 100%, che sarà vero in tutte le situazioni “ragionevoli”, visto che tipicamente l’errore sarà molto più piccolo di 1, allora [](http://www.texrendr.com/?eqn=%7C%5Ctilde%7By%7D%7C%3D%7Cy%2B%5Ctilde%7By%7D-y%7C%20%3D%20%7Cy%7C%20%5Ccdot%20%7C1%2B(%5Ctilde%7By%7D-y)%20%2F%20y)%7C#0) usando la stima da sotto nella disuguaglianza triangolare

[](http://www.texrendr.com/?eqn=a%3D1#0)

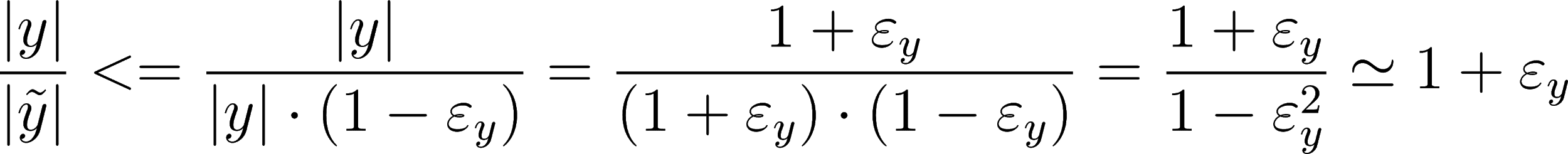
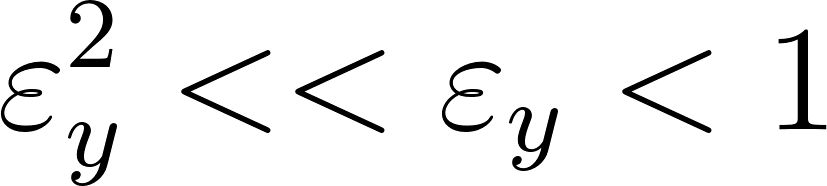
[](https://www.codecogs.com/eqnedit.php?latex=b%3D(%5Ctilde%7By%7D-y)%2Fy#0)

[](https://www.codecogs.com/eqnedit.php?latex=%7Ca%2Bb%7C%20%3E%3D%20%7C%7Ca%7C-%7Cb%7C%7C#0)

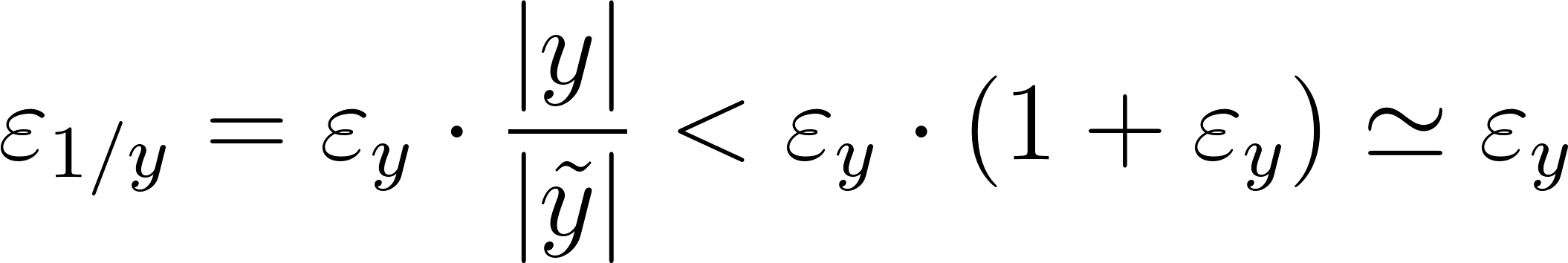
[](https://www.codecogs.com/eqnedit.php?latex=%7C1%2B(%5Ctilde%7By%7D-y)%2Fy%7C%20%3E%3D%20%7C1-%7C%5Ctilde%7By%7D-y%7C%20%2F%20%7Cy%7C%7C%20%3D%20%7C1-%5Cvarepsilon_y%7C%20%3D%201-%5Cvarepsilon_y#0) perché [](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_y%3C1#0)

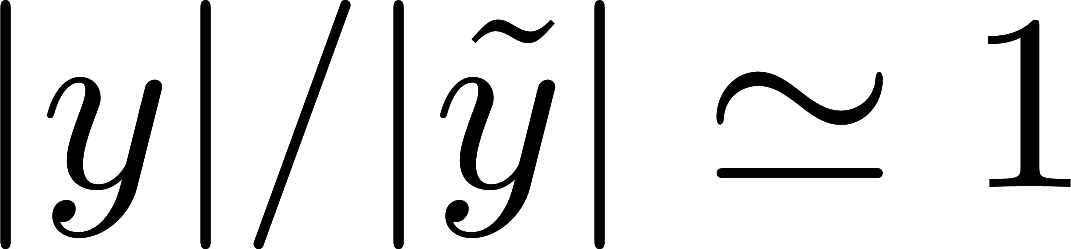
da cui si ottiene [](https://www.codecogs.com/eqnedit.php?latex=%7C%5Ctilde%7By%7D%7C%20%3E%3D%20%7Cy%7C%20%5Ccdot%20(1-%5Cvarepsilon_y)#0)

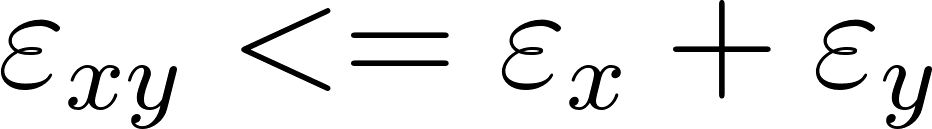
e quindi

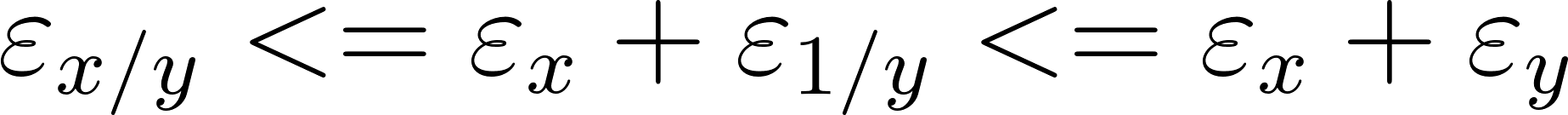
[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7B%7Cy%7C%7D%7B%7C%5Ctilde%7By%7D%7C%7D%20%3C%3D%20%5Cfrac%7B%7Cy%7C%7D%7B%7Cy%7C%20%5Ccdot%20(1-%5Cvarepsilon_y)%7D%20%3D%20%5Cfrac%7B1%2B%5Cvarepsilon_y%7D%7B(1%2B%5Cvarepsilon_y)%5Ccdot%20(1-%5Cvarepsilon_y)%7D%20%3D%20%5Cfrac%7B1%2B%5Cvarepsilon_y%7D%7B1-%5Cvarepsilon_y%5E2%7D%20%5Csimeq%201%2B%5Cvarepsilon_y#0) perché [](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_y%5E2%20%3C%3C%20%5Cvarepsilon_y%20%3C1#0)

(per la prima volta usiamo qui il simbolo << molto minore) alla fine otteniamo

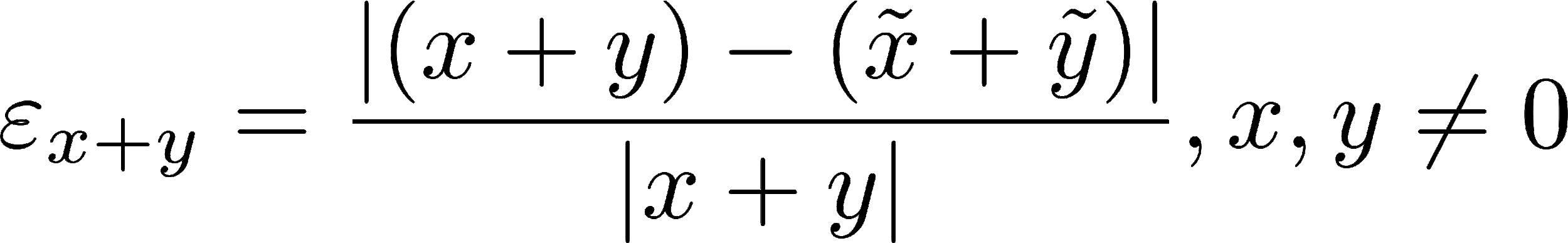
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7B1%2Fy%7D%20%3D%20%5Cvarepsilon_y%20%5Ccdot%20%5Cfrac%7B%7Cy%7C%7D%7B%7C%5Ctilde%7By%7D%7C%7D%20%3C%20%5Cvarepsilon_y%20%5Ccdot%20(1%2B%5Cvarepsilon_y)%20%5Csimeq%20%5Cvarepsilon_y#0)

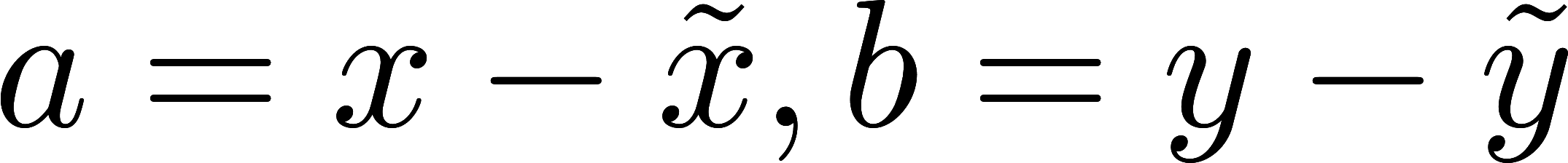
cioè abbiamo quantificato in modo più preciso la stima qualitativa [](http://www.texrendr.com/?eqn=%7Cy%7C%2F%7C%5Ctilde%7By%7D%7C%20%5Csimeq%201#0). Riassumendo, moltiplicazione e divisione sono operazioni stabili visto che

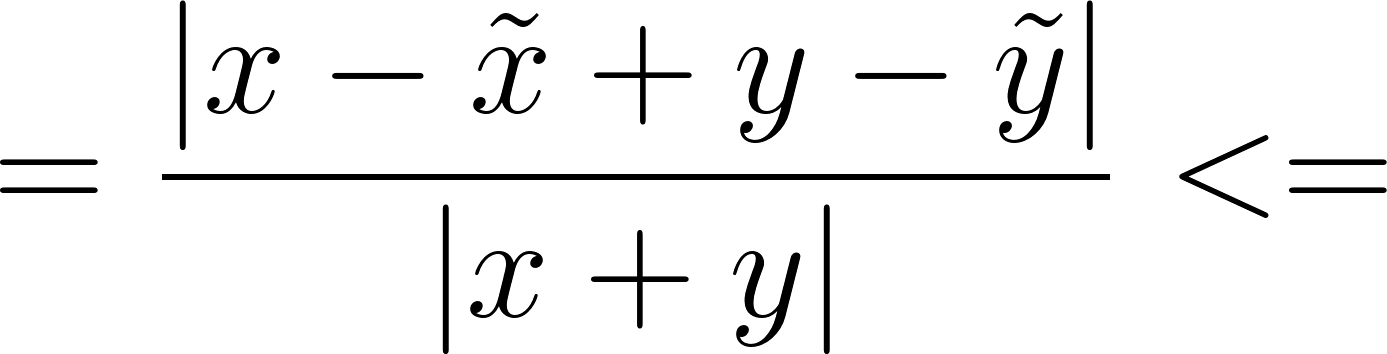
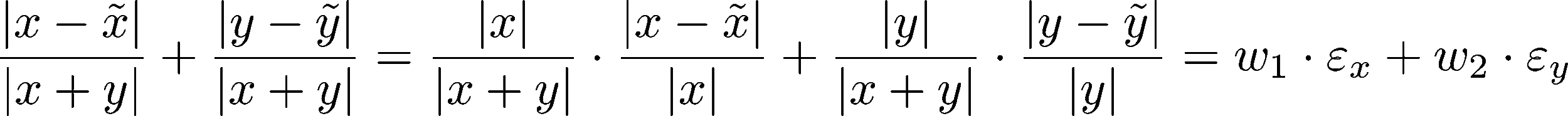
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bxy%7D%20%3C%3D%20%5Cvarepsilon_x%20%2B%20%5Cvarepsilon_y#0)

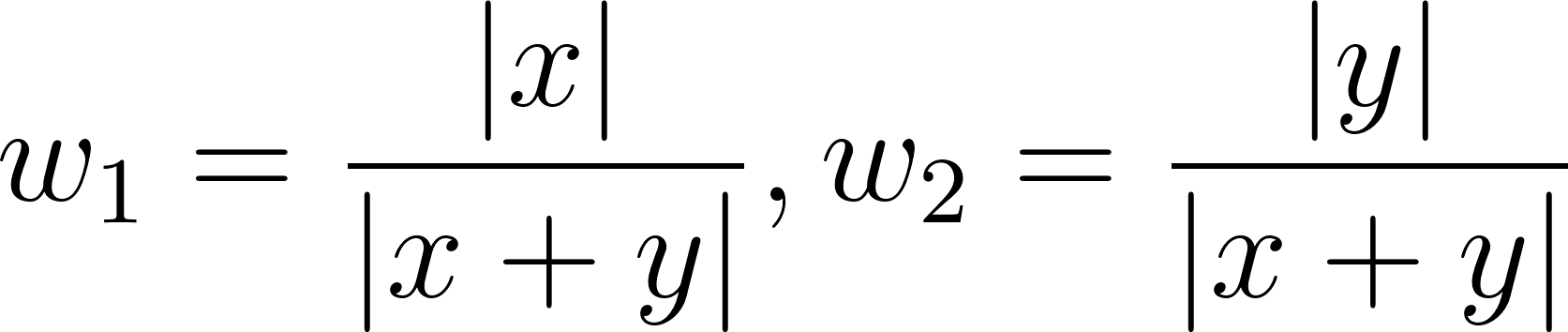
[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bx%2Fy%7D%20%3C%3D%20%5Cvarepsilon_x%20%2B%20%5Cvarepsilon_%7B1%2Fy%7D%3C%3D%5Cvarepsilon_x%20%2B%20%5Cvarepsilon_y#0)

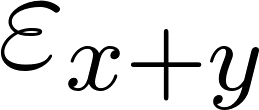
Restano da analizzare addizione e sottrazione. Ma quando parliamo di addizione e quando di sottrazione visto che i numeri possono avere segno qualsiasi ? Quello che faremo è analizzare la risposta agli errori sui dati della somma algebrica x+y ma tale somma (non importa il segno del risultato, è effettivamente un addizione se x e y hanno lo stesso segno è invece una sottrazione se hanno segni diversi)

[](http://www.texrendr.com/?eqn=%5Cvarepsilon_%7Bx%2By%7D%3D%5Cfrac%7B%7C(x%2By)-(%5Ctilde%7Bx%7D%2B%5Ctilde%7By%7D)%7C%7D%7B%7Cx%2By%7C%7D%2C%20x%2Cy%20%5Cneq%200#0)

[](http://www.texrendr.com/?eqn=a%3Dx-%5Ctilde%7Bx%7D%2C%20b%3Dy-%5Ctilde%7By%7D#0)

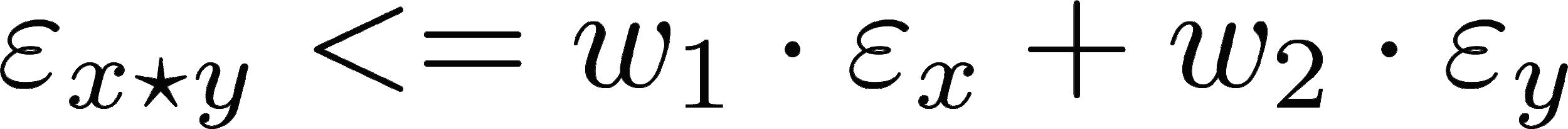
[](https://www.codecogs.com/eqnedit.php?latex=%3D%20%5Cfrac%7B%7Cx-%5Ctilde%7Bx%7D%2By-%5Ctilde%7By%7D%7C%7D%7B%7Cx%2By%7C%7D%20%3C%3D#0) (disuguaglianza triangolare) [](http://www.texrendr.com/?eqn=%5Cfrac%7B%7Cx-%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%2By%7C%7D%20%2B%20%5Cfrac%7B%7Cy-%5Ctilde%7By%7D%7C%7D%7B%7Cx%2By%7C%7D%3D%5Cfrac%7B%7Cx%7C%7D%7B%7Cx%2By%7C%7D%20%5Ccdot%20%5Cfrac%7B%7Cx-%5Ctilde%7Bx%7D%7C%7D%7B%7Cx%7C%7D%20%2B%20%5Cfrac%7B%7Cy%7C%7D%7B%7Cx%2By%7C%7D%20%5Ccdot%20%5Cfrac%7B%7Cy-%5Ctilde%7By%7D%7C%7D%7B%7Cy%7C%7D%7D%20%3D%20w_1%20%5Ccdot%20%5Cvarepsilon_x%20%2B%20w_2%20%5Ccdot%20%5Cvarepsilon_y#0)

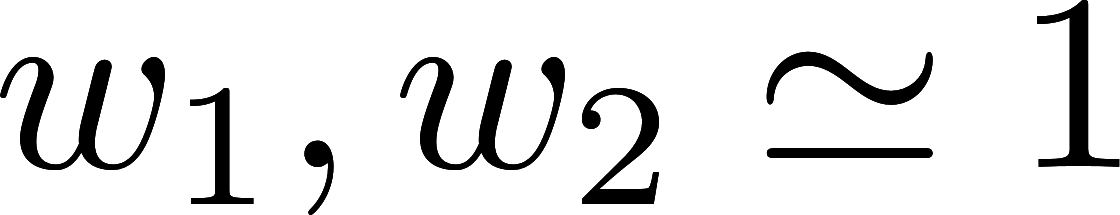
dove [](https://www.codecogs.com/eqnedit.php?latex=w_1%20%3D%20%5Cfrac%7B%7Cx%7C%7D%7B%7Cx%2By%7C%7D%2C%20w_2%3D%5Cfrac%7B%7Cy%7C%7D%7B%7Cx%2By%7C%7D#0)

abbiamo quindi maggiorato [](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bx%2By%7D#0) con una somma pesata degli errori sui dati, con pesi w1 e w2.

Si noti che questi pesi dipendono da x e da y, ma non dipendono dagli errori

in realtà anche con la moltiplicazione e la divisione siamo arrivati in sostanza a una stima del tipo

[](http://www.texrendr.com/?eqn=%5Cvarepsilon_%7Bx%20%5Cstar%20y%7D%20%3C%3D%20w_1%5Ccdot%20%5Cvarepsilon_x%20%2B%20w_2%20%5Ccdot%20%5Cvarepsilon_y#0)

dove [](http://www.texrendr.com/?eqn=w_1%2C%20w_2%20%5Csimeq%201#0)

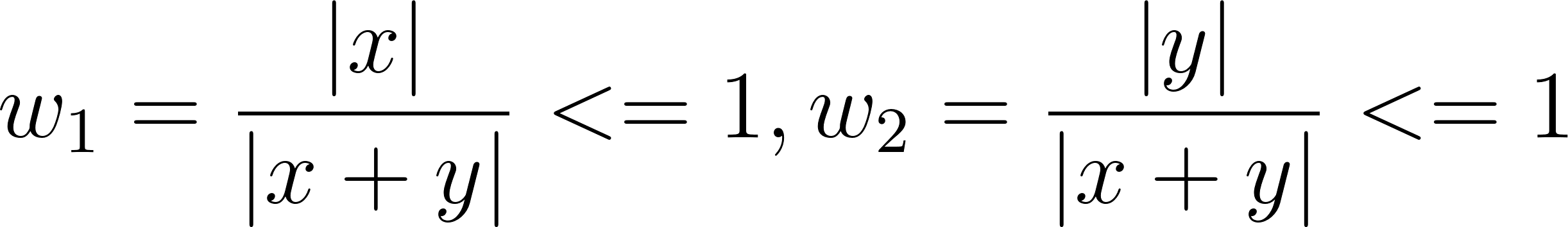
Vedremo ora che questa è anche la situazione con l’addizione,

mentre le cose possono cambiare radicalmente con la sottrazione

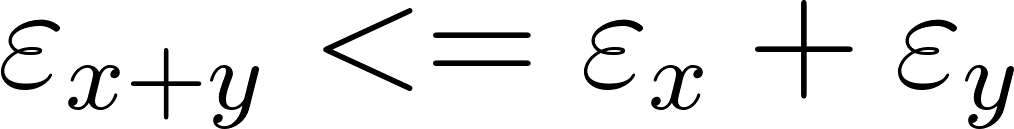
ADDIZIONE (sgn(x)=sgn(y))

in questo caso |x+y| >= |x|, |y| si pensi per semplicità al caso x,y>0, è chiaro che x+y > x e x+y > y

quindi

[](https://www.codecogs.com/eqnedit.php?latex=w_1%20%3D%20%5Cfrac%7B%7Cx%7C%7D%7B%7Cx%2By%7C%7D%20%3C%3D%201%2C%20w_2%20%3D%20%5Cfrac%7B%7Cy%7C%7D%7B%7Cx%2By%7C%7D%20%3C%3D%201#0)

cioè

[](https://www.codecogs.com/eqnedit.php?latex=%5Cvarepsilon_%7Bx%2By%7D%20%3C%3D%20%5Cvarepsilon_x%20%2B%20%5Cvarepsilon_y#0)

ovvero l’addizione è STABILE (l’errore relativo sul risultato è maggiorato da una quantità che è dell’ordine degli errori sui dati)

SOTTRAZIONE(sgn(x)!=sgn(y))

in questo caso |x+y| < |x| oppure |x+y| < |y| quindi max {w1,w2}>1

questo ci dice che la sottrazione può far perdere precisione rispetto agli errori sui dati, ma quanta?

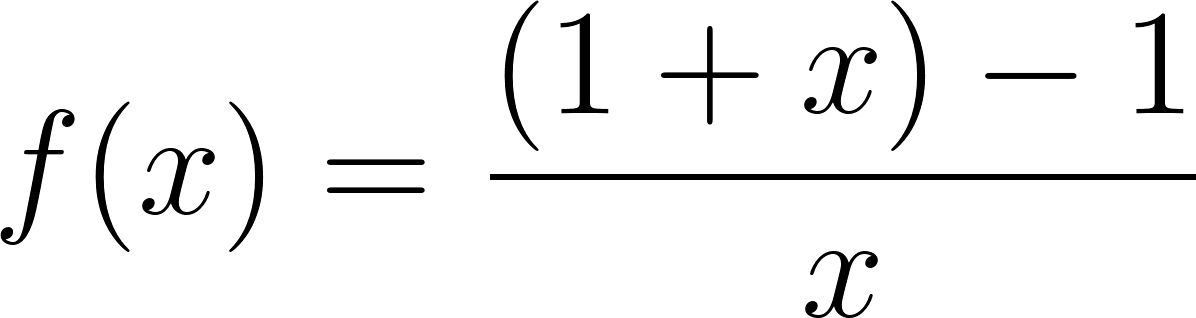
in effetti può succedere che |x| e |y| sono molto vicini in termini relativi cioè che

|x+y| << |x|,|y| in queste situazioni w1,w2 >> 1 e la sottrazione diventa INSTABILE si noti che w1 e w2 possono essere arbitrariamente grandi, dipende dai dati.

È importante osservare che si tratta di un problema di “vicinanza” relativa, non assoluta, tra le due quantità che vengono sottratte, cioè i casi instabili non sono quelli in cui |x+y| è “piccolo”, ma quelli in cui è piccolo rispetto a |x|, |y|.  
Ad esempio sono analoghi:



Detta a parole, due numeri dell’ordine delle unità che distano di qualche milionesimo, sono altrettante vicini in termini relativi di due numeri dell’ordine del milione che distano di qualche unità.  
In entrambi i casi i pesi w1 , w2 sono fattori di amplificazione dell’errore dell’ordine di 106  
Possiamo sintetizzare che la SOTTRAZIONE è potenzialmente (non sempre) INSTABILE infatti se |x| e |y| sono distanti in termini relativi, la sottrazione perderà poca precisione.  
Se invece sono vicini perderà molta precisione, tanta più quanto più sono vicini.  
Questo fenomeno (che si chiama anche “cancellazione numerica”) è il primo e importante esempio di possibile instabilità di un algoritmo ( in questo caso la semplice operazione di sottrazione tra numeri approssimati) è un fenomeno che studieremo meglio con degli esempi e che si può fronteggiare in modi:  
1) cercare di riscrivere le espressioni e gli algoritmi in modo da evitare sottrazioni instabili ( lo vedremo ad esempio con la formula risolutiva per le equazioni di 2º grado)  
2) aumentando la precisione (cioè diminuendo l’errore sui dati) in funzione della grandezza di w1 e w2  
In campo sperimentale, questo significa aumentare la precisione dello strumento di misura nel caso dell’arrotondamento, questo significa andare in un sistema floating-point a precisione estesa, se ad esempio w1 e w2 sono così grandi da mettere in crisi un sistema a precisione doppia si tenga conto che come vedremo w1 e w2 possono essere arbitrariamente grandi e quindi far perdere completamente di significato al risultato quando   
Come già osservato però, usare precisioni estese può avere un costo computazionale molto elevato in termini di tempo di calcolo e di occupazione di memorie.  
Nella prossima lezione faremo vari esempi di perdita di precisione dovuta ad una sottrazione e di stabilizzazione (quando possibile) dell’algoritmo di calcolo evitando tale sottrazione.  
Per fissare le idee, facciamo però subito un esempio significativo: consideriamo la funzione

[](https://www.codecogs.com/eqnedit.php?latex=f(x)%3D%5Cfrac%7B(1%2Bx)-1%7D%7Bx%7D#0) è evidente che (è la funzione costante 1) però, calcolando in MatLab si ottiene cioè l’errore relativo sul risultato è >11% (un errore enorme rispetto all’arrotondamento che non supera )  
Vedremo che la spiegazione sta nella sottrazione a numeratore, visto che 1+10-15 è vicinissimo ad 1 invece , pur essendo perché?

Tratteremo entrambi i casi nella prossima lezione.