Analisi degli errori

Problema reale → Modello Matematico → Risoluzione Numerica → Implementazione Software

Errori di semplificazione, errori di discretizzazione (o di troncamento analitico), errori di round-off

Errore assoluto ed errore relativo

Si vuole stimare l'influenza degli errori sulla soluzione cioè di quanto la soluzione calcolata y, e quindi approssimata, si discosta da quella *vera* x.

La bontà di tale approssimazione, in termini di cifre decimali, è data dall'*errore assoluto*: $|x-y|$

in termini di cifre significative è data dall'*errore relativo*: $\left|\frac{x-y}{x}\right|$,

$x \neq 0$.

Le **cifre significative** sono le cifre necessarie ad esprimere **il risultato di una misura con la precisione con cui è stata fatta**. Distinguiamo tra numeri interi e decimali.

1) Numeri interi. I numeri: 1236, 12360, 123600 hanno tutti 4 cifre significative.
2) Numeri decimali.

12.0 3 cifre significative

12.1 3 " "

12.20	4	"	"
12.200	5	"	"
0.012	2	"	"
0.133	3	"	"

Si ha che:

se y è un'approssimazione corretta di x a *p* cifre decimali:

$$|x - y| < 10^{-p}$$

se y è un'approssimazione corretta di x a *p* cifre significative:

$$\left|\frac{x - y}{x}\right| < 10^{-p+1}$$

Esempio: x = 624.428731, y = 624.420711

$$|x - y| = 0.8 \cdot 10^{-2} < 10^{-2}$$

ed infatti le prime 2 cifre decimali coincidono.

$$\left|\frac{x - y}{x}\right| = 1.3 \cdot 10^{-5} < 10^{-4}$$

ed infatti le prime 5 cifre significative coincidono.

In generale, l'errore relativo dà una stima più attendibile della bontà dell'approssimazione rispetto all'errore assoluto, tranne il caso in cui x sia prossimo a zero o 'piccolo'.

L'errore relativo generalmente è più importante. Mostriamolo con un esempio.

Calcolo A: $x = 5\ 10^{-5}$, $x_c = 4\ 10^{-5}$

$E_A = 10^{-5}$ $E_R = 0{,}2$ 20%

Calcolo B: $x = 5000$, $x_c = 4950$

$E_A = 50$ $E_R = 0{,}01$ 1%

Il numero di cifre significative dà una **stima dell'errore relativo**. Ad esempio, consideriamo i due numeri: 910 e 110 che hanno entrambi 2 cifre significative. I valori veri saranno: 910 ± 5 e 110 ± 5. Nel primo caso: $\frac{\Delta x}{x} = \frac{5}{910} \approx 0.5\%$, nel secondo: $\frac{\Delta x}{x} = \frac{5}{110} \approx 5\%$

Pertanto nel caso di 2 cifre significative, l'errore relativo è

$$\frac{\Delta x}{x} \approx 0.5 \div 5\%$$

Num. cifre signif.	$\frac{\Delta x}{x}$ teorico	Rozzamente
1	$5.5 \div 50\%$	10%
2	$0.5 \div 5\%$	1%
3	$0.055 \div 0.5\%$	0.1%

Ordine di convergenza e complessità computazionale

Generalmente i metodi che tratteremo sono di tipo iterativo ossia possono, teoricamente, essere eseguiti un numero infinito di volte. Per tali metodi dovremo quindi introdurre un ***criterio di arresto***. Inoltre spesso dovremo sostituire ad una serie la somma dei suoi primi *N* termini. Ciò darà luogo al cosiddetto ***errore di troncamento***

che dipenderà da un parametro, che diremo h, che tende a 0 quando $N \to \infty$. Se tale errore tende a zero come h^p (nel senso del limite per $h \to 0$) allora si dice che si ha ***ordine di convergenza p.***

Per ogni tipo di problema si possono inoltre progettare più algoritmi la cui valutazione dipenderà dalla sua ***stabilità***, di cui parleremo più avanti, e dalla sua *efficienza*, che è legata al suo ***costo***, che dipende dal numero di operazioni eseguite, e dall'occupazione dello ***spazio*** richiesto per la memorizzazione dei dati.

Determinare il costo delle operazioni in termini di un numero caratteristico del problema significa determinare la ***complessità computazionale*** dell' algoritmo.

Ad esempio, nel caso di un *sistema lineare*, il numero caratteristico del problema è il numero di equazioni del sistema che, in un sistema con matrice dei coefficienti quadrata, è uguale al numero delle incognite n. Nel caso del metodo di Gauss, che esamineremo più avanti, la complessità computazionale è $\frac{4}{3}n^3$.

Vediamo adesso uno schema, lo ***schema di Hörner,*** che serve ad abbassare il costo computazionale del calcolo di un polinomio.

Schema di Hörner *(nested multiplication)*

Si vuole eseguire il calcolo di:

$$f(x)=ax^3+bx^2+cx+d$$

Sono necessarie tre somme e sei moltiplicazioni per un totale di 9 operazioni floating point (9 flops).

In generale per calcolare:

$$f(x)=a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1 x+a_0$$

occorrono n somme ed $\frac{n(n+1)}{2}$ moltiplicazioni, per un totale di

$$n+\frac{n(n+1)}{2}=\frac{n(n+3)}{2} \sim n^2 \text{ flops}$$

Se invece si applica lo ***schema di Hörner***:

$$f(x)=a_0+x(a_1+x(a_2+\ldots+x(a_{n-1}+xa_n)\ldots))$$

otteniamo n somme ed n moltiplicazioni per un totale di: n flops.

Sistemi di numerazione

Una fonte di errore è data dal passaggio da un sistema di numerazione ad un altro.

I sistemi di rappresentazione numerica sono posizionali: ogni cifra occupa una posizione corrispondente ad una potenza della base del sistema adottato.

Sistema decimale (base 10)

$10258{,}5 = 1 \times 10^4 + 0 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 8 \times 10^0 + 5 \times 10^{-1}$

Sistema binario (base 2)

$(10010{,}1)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} = (18{,}5)_{10}$

Sistema esadecimale (base 16)

$(2A1)_{16} = 2 \times 16^2 + 10 \times 16^1 + 1 \times 16^0 = (673)_{10}$

Le cifre di questi sistemi sono :

decimale: 0, 1, …, 9

binario:0, 1

esadecimale: 0, 1, …, 9, A, B, C, D, E, F

Quindi, scelta una base b, ogni reale a può essere scritto:

$$a = \pm (a_m b^m + a_{m-1} b^{m-1} + \ldots + a_0 + a_{-1} b^{-1} + \ldots)$$

$$0 \leq a_i \leq b - 1$$

Tale rappresentazione è unica tranne se la parte frazionaria contiene infinite cifre consecutive $a_{-k} = b - 1$. In tal caso, una rappresentazione equivalente è quella di considerare il nuovo numero ottenuto sopprimendo la successione e aggiungendo un'unità all'ultima cifra rimasta.

Per esempio nel sistema decimale 0, 72999…9… e 0, 73 rappresentano lo stesso numero.

Per la rappresentazione dei numeri in diverse basi si ha il seguente

Teorema

Sia $b \in N$, $b \geq 2$, $x \in R$, $x \neq 0 \Rightarrow \exists_1 e \in Z$, $\{a_i\}_{i=1,2,\ldots}$ $a_i \in N$:

$$x = \pm\left(\sum_{i=1}^{\infty} a_i b^{-i}\right) b^e$$

$0 \leq a_i \leq b - 1$, $a_1 \neq 0$, a_i definitivamente $\neq b - 1$.

Rappresentazione numerica in un calcolatore

Poiché in un calcolatore lo spazio di memoria è finito, la sommatoria precedente può estendersi fino a t (numero finito):

$$x = \pm(\sum_{i=1}^{t} a_i b^{-i}) b^e$$

con $t < \infty$, $L < e < U$.

Abbiamo due tipi di rappresentazione:

Rappresentazione in virgola fissa

Sono fissati il numero di cifre N che rappresenta il numero e il numero di cifre prima e dopo la virgola N_1, N_2 : $N = N_1 + N_2$

Ad esempio: N = 10, N_1 = 4, N_2 = 6

27,325 → 0027 325000

0,024 → 0000 024000

Quindi, se N sono le posizioni di memoria ed una è per il segno, N - k - 1 per la parte intera e k per quella decimale, si ha:

$$x = (-1)^s (a_{N-2} a_{N-3} \ldots a_k . a_{k-1} \ldots a_0) = (-1)^s b^{-k} \sum_{i=0}^{N-2} a_i b^i$$

dove *b* è la base ed *s*, che può essere 0 oppure 1, è scelto in base al segno di *x*.

Rappresentazione in virgola mobile

In questo caso la posizione della virgola di un numero decimale non è fissa ma è data dall'esponente.

$$x = (-1)^s (0.\ a_1\ a_2 \ldots a_t)\, b^e, \quad x = (-1)^s\, m\, b^{e-t}, \qquad m = a_1 a_2 \ldots a_t$$

$$t \in N, \quad 0 \le a_i \le b - 1, \quad b^{t-1} \le m \le b^t - 1\ , \ L < e < U$$

In tal caso chiamiamo *m* mantissa ed *e* esponente di *x*.

Lo spazio riservato ad ogni $x \in R$ è:

s	*e*	*|m|*

L'insieme dei numeri la cui mantissa è rappresentabile da t cifre e il cui esponente è compreso tra L ed U, interi che variano per ogni calcolatore, è detto *insieme dei numeri macchina*.

$$F = F(t, b, L, U) = \{0\} \cup \left\{x \in R : x = \pm\left(\sum_{i=1}^{t} a_i b^{-i}\right) b^e\right\}$$

poiché 0 ha una rappresentazione particolare.

F è un insieme finito e numerabile la cui cardinalità è:

$$card(F) = 1 + 2\,(b - 1)\, b^{t-1}\,(U - L + 1)$$

Tale rappresentazione non è unica. Infatti:

$$x = m b^e = m' b^{e-1} = m'' b^{e+1} = \ldots$$

$$m' = m b, \quad m'' = \frac{m}{b}$$

Tale rappresentazione si dice normalizzata se:

$a_1 \neq 0$ ed $m \geq b^{t-1}$.

Per memorizzare e spesso si agisce nel seguente modo: fissato L si memorizza $e^* = e - L$ che è sempre non negativa, tenendo presente che il numero è memorizzato a meno di b^L.

I bits della mantissa m determinano la *precisione* con cui viene rappresentato il numero mentre i bits dell'esponente e determinano il *massimo e il minimo numero rappresentabili*.

Inoltre sistemi differenti di rappresentazione di un numero sono determinati da come sono trattati l'underflow, l'overflow, NaN, $\pm\infty$ all'interno del sistema.

In un calcolatore a 32 bit si hanno 23 bit per la mantissa, 1 bit per il segno e 8 bit per l'esponente: L = -127, U = +127, $-127 \le e \le 127 \Rightarrow 0 \le e^* \le 255$.

Il più grande e il più piccolo numero rappresentabili sono:

$\beta^{L-1} = x_{min} = (.1)_2\, 2^{-127} = \frac{1}{2}\, 2^{-127} = 2^{-128} \approx 10^{-38}$

$\beta^{U}(1-\beta^{-t}) = x_{max} = (.1\ldots1)_2\, 2^{127} \approx 2^{127} \approx 10^{+38}$

infatti: $(.1\ldots1)_2 = \sum_{i=1}^{23}\left(\frac{1}{2}\right)^i = \frac{\frac{1}{2}-\left(\frac{1}{2}\right)^{24}}{1-\frac{1}{2}} = (1-2^{-23}) \approx 1$ $\qquad \left(\sum_{k=m}^{n} x^k = \frac{x^m - x^{n+1}}{1-x}\right)$

Cambiando l'ultimo bit della mantissa di x_{min} si ha:

$\left(\frac{1}{2}+2^{-23}\right)\cdot 2^{-127} = x_{min} + 2^{-150} \approx x_{min} + 10^{-45}$

che è una differenza molto piccola.

Cambiando l'ultimo bit della mantissa di x_{max} si ha:

$((.1\ldots1)_2 - 2^{-23})2^{127} = x_{max} - 2^{104} \approx x_{max} - 10^{31}$

che è una differenza molto grande.

Quindi è meglio rappresentare numeri piccoli.

Un numero decimale per essere rappresentato nel computer viene convertito in binario. Tale conversione può comportare una rappresentazione approssimata. Per vedere ciò vediamo come si opera tale conversione.

Si hanno due casi:

1) numero maggiore di 1. Si divide per due: se c'è il resto si mette 1 altrimenti 0 e si assegnano potenze di due crescenti.
   Esempio:

37	2	1×2^0
18	2	0×2^1
9	2	1×2^2
4	2	0×2^3
2	2	0×2^4
1	2	1×2^5
0		

$\rightarrow \quad (100101)_2 = (37)_{10}$

2) numero minore di 1. Si divide per ½, ovvero si moltiplica per 2. Se il prodotto è minore di uno si ha 0 altrimenti 1. Quando il numero diventa maggiore di uno si sottrae 1 procedendo come prima.

0,2	2	0×2^{-1}
0,4	2	0×2^{-2}
0,8	2	1×2^{-3}
$1,6 \rightarrow 0,6$	2	1×2^{-4}
$1,2 \rightarrow 0,2$	2	0×2^{-5}
0,4	2	0×2^{-6}
0,8	2	1×2^{-7}
$1,6 \rightarrow 0,6$	2	1×2^{-8}

$\rightarrow \quad (0,2)_{10} = (\overline{0011})_2$

Si ha quindi una rappresentazione finita in decimale e una rappresentazione approssimata in binario e ciò dà luogo ad un *errore di arrotondamento (round-off)*.

Conversione da base 10 a base qualunque *

Sia n un numero in base 10, vogliamo convertirlo in un numero in base $b \in \mathbb{N}$. Sia $n = (a_j a_{j-1} \cdots a_1 a_0)_b$

Dividendo n per b si ha:

$$\frac{n}{b} = (a_j\, b^{j-1}) + (a_{j-1}\, b^{j-2}) + \ldots + (a_1\, b^0) + \frac{a_0}{b}$$

$\Rightarrow$ $n = bn_0 + a_0$ dove $n_0 < n$

Quindi a_0 , l'ultima cifra della rappresentazione, è il resto intero di n/b.

Per ottenere la penultima cifra si divide n_0/b:

$$\frac{n_0}{b} = (a_j\, b^{j-2}) + \ldots + (a_2\, b^0) + \frac{a_1}{b} = n_1 + \frac{a_1}{b}$$

Il procedimento si arresta ad a_j tale che $n_j = 0$.

Esempio: $741 = (a_9 a_8 a_7 \ldots a_1 a_0)_2 = (1011100101)_2$

****

Nell'aritmetica in virgola mobile (floating point) si ha il problema:

dato $x \in R$ come scegliere $fl(x) \in F = \{0\} \cup \{x \in R : x = \pm\left(\sum_{i=1}^{t} a_i b^{-i}\right) b^e\}$

Si fanno i seguenti casi:

1) $e < L$ *underflow* $fl(x) = 0$ "warning"

2) $e > U$ *overflow* segnale di errore e arresto del programma.

3) $L \le e \le U$

i) $a_k = 0$, $k \ge t + 1 \Rightarrow x \in F$

ii) $x \notin F$, $fl(x) \neq x$ e si ha:

a) *chopping o troncamento*: si esclude la parte destra della t-esima cifra

$$fl(x) = \pm\left(\sum_{i=1}^{t} a_i b^{-i}\right) b^e \qquad fl(x) < x$$

b) *rounding o arrotondamento*

$$x = \pm\left(\frac{a_1}{b} + \ldots + \frac{a_t}{b^t} + \frac{a_{t+1}}{b^{t+1}} + \ldots\right) b^e$$

$$fl(x) = \pm\left(\frac{a_1}{b} + \ldots + \frac{a_{t-1}}{b^{t-1}} + \frac{\alpha_t}{b^t}\right) b^e$$

con $\alpha_t = \begin{cases} a_t & 0 \le a_{t+1} < b/2 \qquad fl(x) < x \\ a_t + 1 & \frac{b}{2} \le a_{t+1} \le b - 1 \qquad fl(x) > x \end{cases}$

cioè si aggiunge $\frac{1}{2} b^{-t}$ e si tronca alla t-esima cifra.

Poiché le mantisse dei numeri macchina $\bar{p}$, con $b^{-1} \le \bar{p} < 1$, non hanno più di t cifre, la distanza tra due mantisse di macchina consecutive è b^{-t}. Quindi, se $\bar{p}_1$, $\bar{p}_2 > 0$, e $\bar{p}_2$ è consecutivo a $\bar{p}_1$ si ha:

$\bar{p}_2 = \bar{p}_1 + b^{-t}$

Quindi con **a**) tutti tali p vengono sostituiti con $\bar{p}_1$ e quindi $\left|p - \overline{p_1}\right| < b^{-t}$, invece con **b**) tutte le mantisse in $(\bar{p}_1, \bar{p}_1 + \frac{1}{2}b^{-t})$ sono sostituite con $\bar{p}_1$ e tutte le mantisse in $(\bar{p}_1 + \frac{1}{2}b^{-t}, \bar{p}_2)$ con $\bar{p}_2$. Pertanto se $\bar{p}$ è la mantissa associata a p si ha: $|p - \bar{p}| \le \frac{1}{2}b^{-t}$.

Quindi, per l'errore assoluto si ha:

$$|x - fl(x)| \le \begin{cases} b^{-t}b^{e} & \textit{chopping} \\ \frac{1}{2}b^{-t}b^{e} & \textit{rounding} \end{cases}$$

Per l'errore relativo:

$$\left|\frac{x - fl(x)}{x}\right| \le \begin{cases} b^{-t+1} & \textit{chopping} \\ \frac{1}{2}b^{-t+1} & \textit{rounding} \end{cases}$$

Si definisce precisione o ***epsilon macchina*** il sup dell'errore relativo, cioè:

$$\varepsilon_M = \begin{cases} b^{-t+1} & \textit{chopping} \\ \frac{1}{2}b^{-t+1} & \textit{rounding} \end{cases}$$

Pertanto $\varepsilon_M = .5 \cdot b^{1-t}$ è il massimo errore della rappresentazione quando l'arrotondamento è il rounding che è quindi la distanza tra 1 e il prossimo numero più grande in flop: $1 + b^{1-t}$.

Non ha senso cercare delle approssimazioni con precisione inferiore ad ε_M.

L'aritmetica standard IEEE include 2 tipi di numeri flop:

la single precision con 32 bits e la double precision con 64 bits.

I numeri rappresentati, l' ε_M e il range sono dati risp., da:

$(-1)^s \cdot 2^{e-127} \cdot (1+m)$, $\varepsilon_M = 2^{-24} \cong 6 \cdot 10^{-8}$, $2^{-126} < x < 2^{128} \leftrightarrow 10^{-38} < x < 10^{38}$

$(-1)^s \cdot 2^{e-1023} \cdot (1+m)$, $\varepsilon_M = 2^{-53} \cong 10^{-16}$, $2^{-1022} < x < 2^{1023} \leftrightarrow 10^{-308} < x < 10^{308}$

Distribuzione dei numeri in floating point *

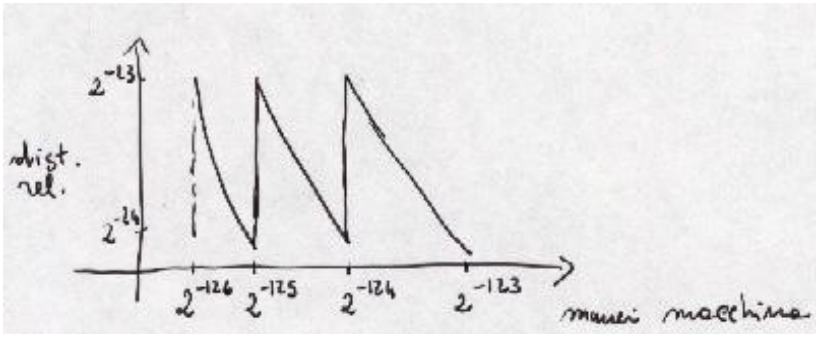
I numeri in floating point non sono equispaziati ma si addensano in prossimità del più piccolo numero rappresentabile.

La spaziatura tra due $x_1, x_2 \in F$ è almeno $b^{-1} \varepsilon_M |x_2|$ ed al più vale $\varepsilon_M |x_2|$ se $\varepsilon_M = b^{1-t}$

All'interno dell'intervallo $[b^e, b^{e+1}]$ i punti sono invece equispaziati e la loro distanza è b^{e-t}.

Quindi ogni volta che si aumenta (o diminuisce) *e* di una unità si ha un aumento (o una diminuzione) di un fattore b della distanza tra due numeri consecutivi. Per questo si prediligono basi piccole.

Il fenomeno, detto *wobbling precision*, ha quindi un andamento oscillatorio:



****

Condizionamento e stabilità

Consideriamo il problema: trovare x tale che F(x) = d, dove d è il dato o i dati da cui dipende la soluzione x ed F relazione funzionale che lega x e d. Diremo che tale problema è *ben posto* se, per un certo dato, la soluzione esiste, è unica e dipende con continuità dai dati. La dipendenza continua dei dati significa che piccole perturbazioni sui dati danno luogo a piccole variazioni della soluzione, dove "piccolo" può essere inteso in senso relativo

o assoluto. D'altronde abbiamo visto che si commettono errori sia nel rappresentare i numeri reali, sia nell'esecuzione di operazioni aritmetiche.

Nascono a questo punto due problemi: ci si chiede

1) alterando i dati del problema di quanto si altera la soluzione;
2) come si propagano gli errori.

Il primo problema è connesso con la dipendenza continua dai dati della soluzione e può essere stimato con il ***numero di condizionamento*** del problema, numero che non dipende dall'uso dell'aritmetica finita del calcolatore ma dal tipo di problema.

Supponiamo di alterare di δd i dati del problema precedente e ci chiediamo di quanto si altera la soluzione:

$d + \delta d \rightarrow x + \delta x$

Diremo *numero di condizionamento relativo*:

$$K = \frac{\|\delta x\|/\|x\|}{\|\delta d\|/\|d\|}$$

Se $x = 0$, $d = 0$ si calcola il *numero di condizionamento assoluto*:

$$K_{ASS} = \|\delta x\|/\|\delta d\|$$

Se K è grande il problema è mal condizionato. Se un problema è ben posto ma K è grande basta riformulare il problema.

Il secondo problema dipende dalla stabilità dell'algoritmo.

Ad ogni problema numerico si possono associare più algoritmi. Un algoritmo è stabile se la propagazione degli errori dovuti all'aritmetica di macchina è limitata. Un algoritmo è più stabile di un altro se in esso l'influenza degli errori è minore.

Errori dovuti alle operazioni aritmetiche

Quale delle quattro operazioni può provocare una perdita di precisione?

In generale, i risultati di operazioni aritmetiche tra numeri di macchina non sono numeri di macchina. Indichiamo con $a \oplus b$ il valore vero di un calcolo, dove $\oplus$ è una delle 4 operazioni. Quando tale risultato $\notin F$, indichiamo la sua approssimazione con $fl(a \oplus b)$.

La differenza $a \oplus b - fl(a \oplus b)$ è il *round-off error*.

Per minimizzare tale errore si possono usare degli accorgimenti, illustriamone uno nel caso della somma: in tal caso si rendono uguali gli esponenti, si sommano le mantisse in un accumulatore con $2m$ cifre e poi si aggiustano gli esponenti.

Ciò può portare alla non validità delle proprietà commutativa, distributiva ed associativa della somma e della moltiplicazione.

Esempio: Supponiamo di avere 4 cifre per la mantissa e vogliamo eseguire la: $\sum_{i=1}^{11} x_i$ con

$x_1 = 0{,}5055\ 10^4$, $x_i = 0{,}4000\ 10^0$ $\quad i = 2,\ldots,11$

$x_1 + x_2 = 0{,}5055\ 10^4 + 0{,}00004\ 10^4 = 0{,}50554\ 10^4 = 0{,}5055\ 10^4$

e quindi $\sum_{i=1}^{11} x_i = 0{,}5055\ 10^4$

Se invece si esegue prima la: $\sum_{i=2}^{11} x_i = 0{,}4\ 10^1$ si avrà:

$$\sum_{i=1}^{11} x_i = x_1 + \sum_{i=2}^{11} x_i = 0{,}5055\ 10^4 + 0{,}0004\ 10^4 = 0{,}5059\ 10^4$$

Quindi, per minimizzare la propagazione degli errori nel caso di somma di molti numeri dello stesso segno, tali numeri vanno sommati dal più piccolo al più grande.

Se $fl(a \oplus b)$ è arrotondato correttamente, come nel sistema IEEE in cui tale numero è il floating più vicino, allora a meno di over- o under-flow si ha:

$$fl(a \oplus b) = (a \oplus b)(1+\delta) \quad \text{con } |\delta| < \varepsilon_M .$$

In realtà, per evitare errori dovuti all'underflow si aggiunge una quantità piccolissima (10^{-45} in s.p.) che dà luogo ad una "eccezione" che viene segnalata da un flag.
Esaminiamo quindi gli errori delle 4 operazioni.

Per ***l'errore assoluto*** si ha:

Somma algebrica: $x_1(1+\varepsilon_1) \pm x_2(1+\varepsilon_2)$

Prodotto: $x_1(1+\varepsilon_1) \cdot x_2(1+\varepsilon_2)$

Divisione: $x_1(1+\varepsilon_1) / x_2(1+\varepsilon_2)$

Per ***l'errore relativo*** si ha:

prodotto: $$\frac{x_1 x_2 - x_1(1+\varepsilon_1)x_2(1+\varepsilon_2)}{x_1 x_2} = -\varepsilon_1 - \varepsilon_2 - \varepsilon_1\varepsilon_2 \sim -\varepsilon_1 - \varepsilon_2$$

divisione: $$\frac{\frac{x_1}{x_2} - \frac{x_1(1+\varepsilon_1)}{x_2(1+\varepsilon_2)}}{x_1/x_2} = \frac{-\varepsilon_1 + \varepsilon_2}{1+\varepsilon_2} \sim -\varepsilon_1 + \varepsilon_2$$

somma algebrica: $$\frac{(x_1 \pm x_2) - [x_1(1+\varepsilon_1) + x_2(1+\varepsilon_2)]}{x_1 \pm x_2} = \frac{-x_1\varepsilon_1 - x_2\varepsilon_2}{x_1 \pm x_2}$$

Pertanto:

Sottrazione: ok per l'errore assoluto, ma si ha un errore relativo grande se $x_1 \approx x_2$

Prodotto : ok per l'errore relativo, l'errore assoluto dipende dall'ordine di grandezza dei fattori

Divisione: ok per l'errore relativo, l'errore assoluto è grande se $x_2 \approx 0$

Cancellazione

Esempio: $f(x) = x(\sqrt{x+1} - \sqrt{x})$ con 6 cifre decimali per la mantissa

x	$\tilde{f}(x)$	f(x)
1	0,414210	0,414214
10	1,54340	1,54347
10^2	4,99000	4,98756
10^3	15,8000	15,8074
10^4	50,0000	49,9988
10^5	100,000	198,113

Le cifre sottolineate sono affette da errore. Per 10^5 l'informazione è persa. Per evitare l'errore commesso nel calcolare la differenza di numeri vicini si può procedere così:

$$f(x) = x\left(\sqrt{x+1} - \sqrt{x}\right) = \frac{x\left(\sqrt{x+1} - \sqrt{x}\right)\left(\sqrt{x+1} + \sqrt{x}\right)}{\sqrt{x+1} + \sqrt{x}} = x\frac{x+1-x}{\sqrt{x+1} + \sqrt{x}} = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

Altro esempio: $p(x) = ax^2 + bx + c$

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, \qquad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Algoritmo stabile:

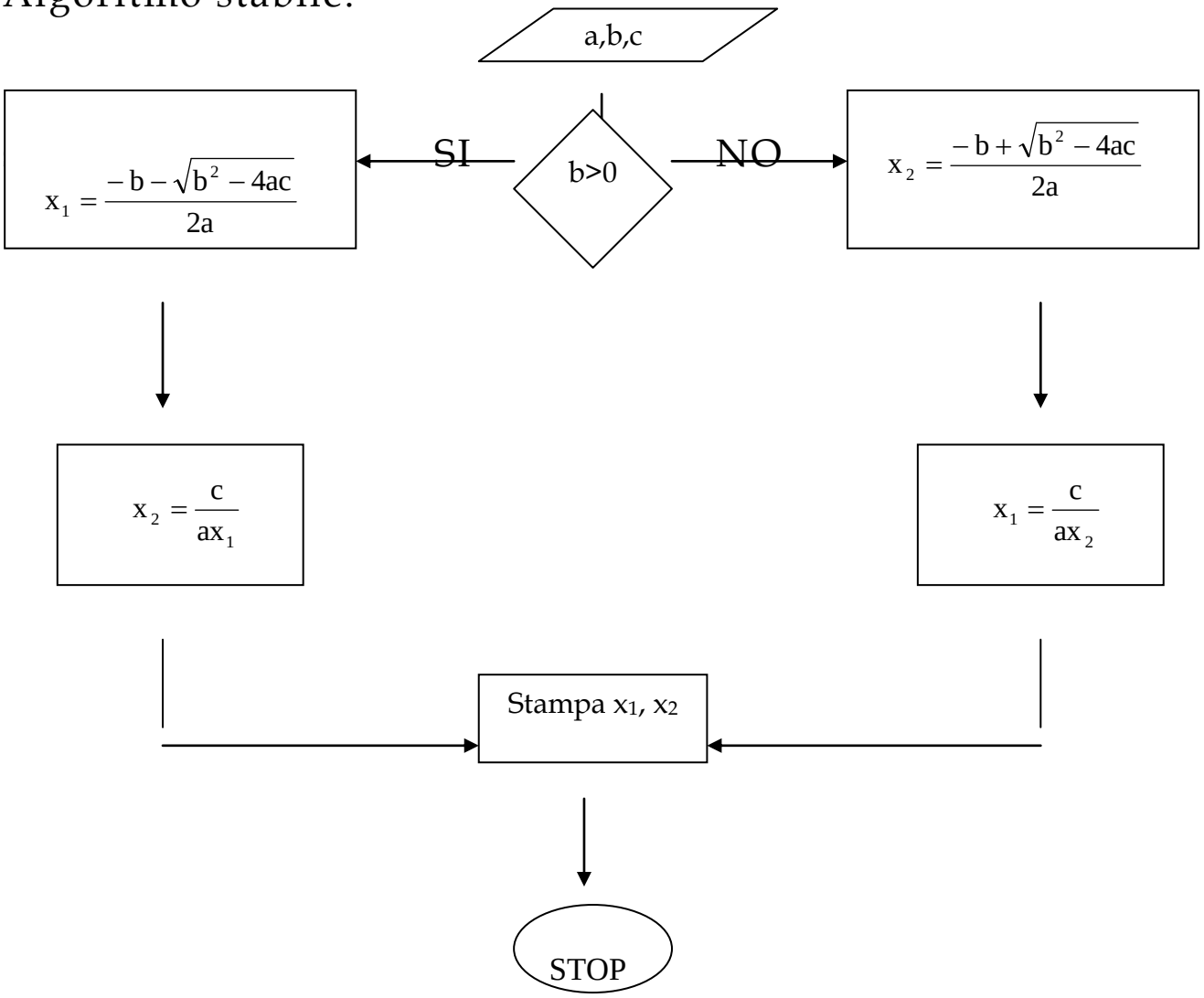


Diagramma di flusso per il calcolo di ε_M nel caso in cui la base sia b.

ε_M è il più piccolo numero per cui si ha 1 + ε_M > 1