Analisi degli errori

Sistemi di numerazione

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 ogni reale *a* può essere scritto:

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

$0 \le a_i \le N - 1$

Tale rappresentazione è unica tranne se la parte frazionaria contiene infinite cifre consecutive $a_{-k} = N - 1$. 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):

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

$t < \infty \quad L \leq e \leq U$

Abbiamo due tipi di rappresentazione:

<u>Rappresentazione in virgola fissa</u>

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: $N = 10$, $N_1 = 4$, $N_2 = 6$

$27{,}325 \rightarrow$ 0027 325000

$0{,}024 \rightarrow$ 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:

$$x = (-1)^s b^{-k} \sum_{j=0}^{N-2} a_j b^j$$

dove b è la base ed s 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.

$\forall\, x \in R$ si può scrivere come: $x = (-1)^s\, m \cdot b^e$ con $m \in R$

Tale rappresentazione non è unica. Infatti:

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

$$m' = m \cdot b_1, \quad m'' = \frac{m}{b}$$

Tale rappresentazione si dice normalizzata se: $b^{-1} \le |m| < 1$

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

$$x = (-1)^s\, m \cdot b^e, \quad m = .a_1 a_2 \dots a_t$$

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

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 la cui caratteristica è compresa tra L ed U, che sono interi che variano per ogni calcolatore è detto *insieme dei numeri macchina*.

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

poiché 0 ha una rappresentazione particolare.

F è un insieme finito e numerabile: $\text{card } F = 1 + 2\,(b - 1)\, b^{t-1}\,(U - L + 1)$

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.

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:

$x_{min} = (.1)_2 \cdot 2^{-127} = \frac{1}{2} \cdot 2^{-127} = 2^{-128}$

$x_{max} = (.1\ldots1)_2 \cdot 2^{127} \sim 2^{128}$

infatti: $(.1\ldots1)_2 = \sum_{i=1}^{23}\left(\frac{1}{2}\right)^i = \frac{1-\left(\frac{1}{2}\right)^{23}}{1-\frac{1}{2}} = 2\cdot\left(1-\left(\frac{1}{2}\right)^{23}\right) = 2\cdot(1-2^{-23}) \sim 2$

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

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

che è una differenza molto piccola.

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

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

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		

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

2) numero minore di 1. Si divide per 1/2, 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}

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

Si ha quindi una rappresentazione finita in decimale e una rappresentazione approssimata in binario: *errore di arrotondamento.*

Conversione da base 10 a base qualunque

Sia n un numero in base 10, vogliamo convertirlo in un numero in base $b \in N$.

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

Dividendo n per b si ha:

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

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

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:

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

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) $a_m = 0$, $m \ge t + 1 \Rightarrow x \in F$

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

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

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

$fl(x) < x$

b) rounding o arrotondamento

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

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

$$\text{con } \alpha_t = \begin{cases} a_t & 0 \le a_{t+1} < b/2 \\ a_k + 1 & \frac{b}{2} \le a_{t+1} \le b-1 \end{cases}$$

$fl(x) \ge x$

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

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

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

Quindi, per *l'errore assoluto* si ha:

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

Per l'errore relativo:

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

L'errore relativo generalmente è più importante.

Calcolo A: $x = 0{,}5 \cdot 10^{-4}$, $x_c = 0{,}4 \cdot 10^{-4}$

$E_A = 0{,}1 \cdot 10^{-4}$

$E_R = 0{,}2$ 20%

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

$E_A = 50$

$E_R = 0{,}01$ 1%

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

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

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

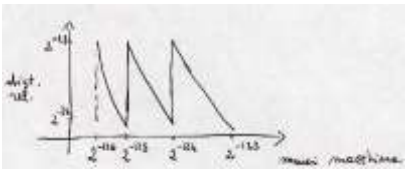
Distribuzione dei numeri in floating point

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 $\beta^{-1}\varepsilon_M |x_2|$ ed al più vale $\varepsilon_M |x_2|$ se $\varepsilon_M = \beta^{1-t}$

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

Quindi ogni volta che si aumenta (o diminuisce) e di una unità si ha un aumento (o una diminuzione) di un fattore β della distanza tra 2 numeri consecutivi. Per questo si prediligono basi piccole.

Il fenomeno (*wobbling precision*) ha quindi un andamento oscillatorio:



Altra fonte di errori: il risultato di operazioni aritmetiche non può essere un numero di macchina.

$x = fl(x)$, $y = fl(y)$ se: $x \text{ op } y \notin F$ si deve definire op il cui risultato appartiene ad F:

$$x \text{ op } y = fl(x \text{ op } y)$$

valida se non c'è overflow.

Ad esempio nel caso della *somma*: si rendono uguali gli esponenti, si sommano le mantisse in accumulatore con 2m cifre, si aggiustano gli esponenti.

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

Esempio: 4 cifre per la mantissa $\sum_{i=1}^{11} x_i$

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

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

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

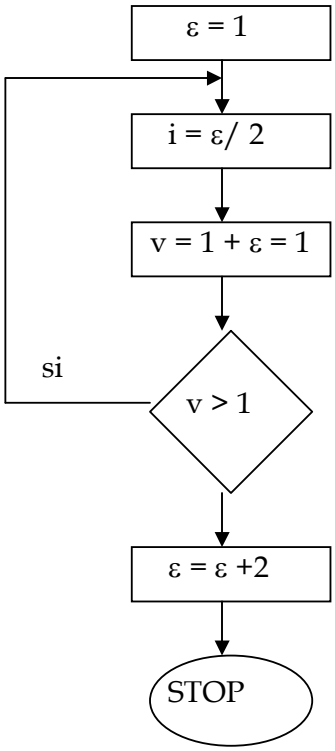
Se invece si fa: $\sum_{i=2}^{11} x_i = 0{,}4 \cdot 10^1$ e

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

Per sommare molti numeri dello stesso segno si devono sommare dal più piccolo al più grande per minimizzare la propagazione degli errori.

Diagramma di flusso per il calcolo di ε

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



Codice fortran:

```
            PROGRAM EPSMAC
C           CALCOLO APPROSSIMATO DI ε
            EPS = 1
1           EPS = EPS / 2
            EPS1 = EPS +1
            IF (EPS1.GT.1) GOTO 1
            EPS = 2 + EPS
        END
```

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 su 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*. Ricordiamo che per algoritmo intendiamo una successione di passi che trasformi un vettore di dati in un corrispondente output. 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.

Quale delle quattro operazioni può provocare una perdita di precisione? Abbiamo detto che i risultati di operazioni aritmetiche tra numeri di macchina in generale non sono

numeri di macchina. L'operazione di macchina associa a due numeri di macchina un terzo numero di macchina.

Indichiamo con $\oplus\ \ominus\ \odot\ \oslash$ le operazioni di macchina:

$\bar{a} \oplus \bar{b} = fl(\bar{a} +) = (\bar{a} +)(1 + \varepsilon_1)$

$\bar{a} \ominus \bar{b} = fl(\bar{a} - \bar{b}) = (\bar{a} - \bar{b})(1 + \varepsilon_2)$

$\bar{a} \odot \bar{b} = fl(\bar{a} \cdot \bar{b}) = (\bar{a} \cdot \bar{b})(1 + \varepsilon_3)$

$\bar{a} \oslash \bar{b} = fl(\bar{a} / \bar{b})(1 + \varepsilon_4)$

dove $\bar{a} = fl(a)$ e $\bar{b} = fl(b)$, $|\varepsilon_i| \le eps = \varepsilon_M$

Esaminiamo l'errore relativo:

$$\left|\frac{(x_1 op x_2) - (\overline{x_1} op \overline{x_2})}{x_1 op x_2}\right| = \left|\frac{(x_1 op x_2) - [x_2(1+\varepsilon_2)]}{x_1 op x_2}\right|$$

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

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

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

$\left|\dfrac{x_i}{x_1 + x_2}\right| \to \infty$ per $x_1 + x_2 \to 0$

Errore assoluto ed errore relativo nelle quattro operazioni:

$\delta x^* = x - x^*$

$\varepsilon_{x^*} = \frac{\delta x^*}{x^*}$

operazione	err. ass.	err. rel.
+	$\delta x_1 + \delta x_2$	$\frac{x_1}{x_1 + x_2}\varepsilon x_1 + \frac{x_2}{x_1 + x_2}\varepsilon x_2$
-	$\delta x_1 - \delta x_2$	$\frac{x_1}{x_1 - x_2}\varepsilon x_1 + \frac{x_2}{x_1 - x_2}\varepsilon x_2$
*	$x_2\delta x_1 - x_1\delta x_2$	$\varepsilon x_1 + \varepsilon x_2$
/	$\frac{\delta x_1}{x_2} - \frac{x_1}{x_2^2}\delta x_2$	$\varepsilon x_1 - \varepsilon x_2$

Osservazioni:

+ e - : nessun problema per l'errore assoluto, ma si ha un errore relativo grande se $x_1 \approx x_2$

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

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

Cancellazione

Esempio: $f(x) = \sqrt{x+1} - \sqrt{x}$ con sei 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

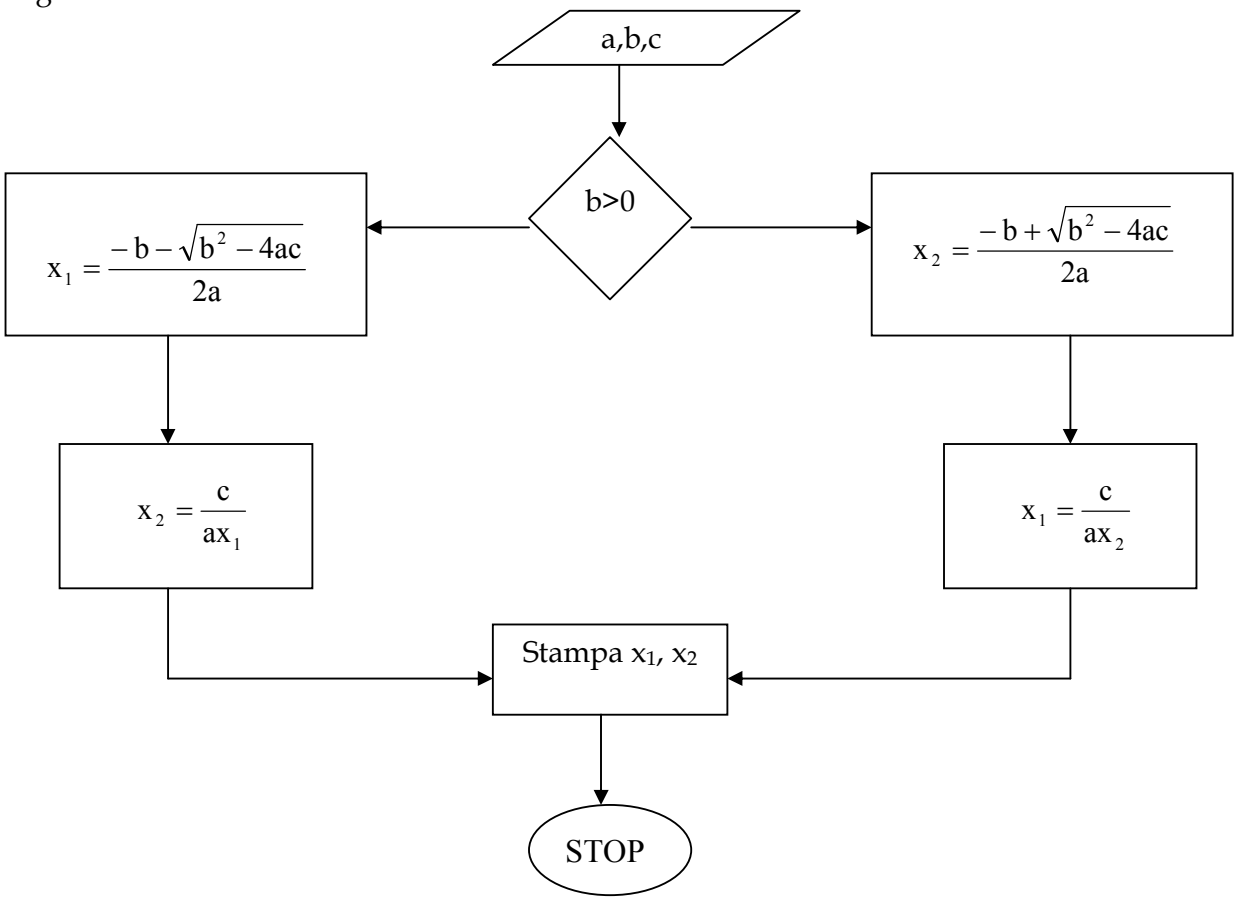
Le cifre sottolineate sono affette da errore. Per 10^5 l'informazione è persa. Per evitare l'errore commesso nel calcolare la differenza di numeri vicini si può procedere così:

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

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:



<u>Schema di Hörner (nested multiplication)</u>

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 $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ ricorrono n somme e $\frac{n(n+1)}{2}$ moltiplicazioni, per un totale di $n + \frac{n(n+1)}{2} = \frac{n(n+3)}{2} \sim n^2$ flops

Se invece si applica lo schema di Horner: $f(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n)\ldots))$

otteniamo n somme e n moltiplicazioni: #flops ~ n